राहुल वाधवा

ऑपरेटिंग सिस्टम

OPERATING SYSTEM

एशियन पब्लिशर्स, मुज़फ़्फ़रनगर®

प्राविधिक शिक्षा परिषद, उत्तर प्रदेश के नवीनतम पाठ्यक्रम पर आधारित

ऑपरेटिंग सिस्टम

(OPERATING SYSTEM)

(प्रयोगात्मक भाग सहित)

(चतुर्थ सत्र, द्वितीय वर्ष कम्प्यूटर साईंस एवं इनफॉरमेशन टेक्नोलॉजी एवं PGDCA के छात्र-छात्राओं हेतु)

Vertexal.

लेखक:

राहुल वाधवा

बी० ई० (इलैक्ट्रॉनिक्स), ज्योतिषविद्

अध्यक्ष इलैक्ट्रॉनिक्स

राजकीय पॉलीटेक्निक

शाहजहांपुर

2023–2024

प्रकाशक:

एशियन पब्लिशर्स, मुज़फ़्फ़रनगर®

[illegible] वाला बाग, नई मण्डी, मुज़फ़्फ़रनगर–251 001 (उ०प्र०)

ऑपरेटिंग सिस्टम

• राहुल वाधवा

प्रकाशक:

एशियन पब्लिशर्स

46/20, कम्बल वाला बाग, नई मण्डी,
मुजफ्फरनगर-251 001 (उ० प्र०)
फोन: 0131-2660989
Visit us at : www.asianpublishers.co.in
email : spmittal@asianpublishers.co.in
sales@asianpublishers.co.in

प्रथम संस्करण : 2016-2017
द्वितीय संस्करण : 2018-2019
पुनः मुद्रित : 2020, 2021, 2022
पुनः मुद्रित : 2023-2024

मूल्य : ₹ 300.00

ISBN: 978-93-5502-139-7

लेजर टाइपसैटिंग:

सारा एसाईनमैन्टस्
शाहदरा, दिल्ली
फोन: 011-22572589

मुद्रक:

विमल प्रेस
मेरठ
फोन: 0-9412203584

प्रस्तावना

ता कहुँ प्रभु कछु अगम नहीं, जा पर तुम्ह अनुकूल।
तब प्रभावँ बड़वानलहि, जारि सकइ खलु तूल॥

अर्थात् हे प्रभु, जिस पर आप प्रसन्न हों, उसके लिये कुछ भी कठिन नहीं है। आपके प्रभाव से रूई (जो स्वयं बहुत जल्दी जल जाने वाली वस्तु है) बड़वानल को निश्चय ही जला सकती है (अर्थात् असम्भव भी सम्भव हो सकता है)। जब श्री हनुमान जी लंका को जलाकर प्रभु श्री राम जी के पास पहुँचे तो प्रभु ने उनसे यह प्रश्न किया कि हे हनुमान, बताओ तो, रावण के द्वारा सुरक्षित लंका और उसके बड़े बाँके किले को तुमने किस तरह जलाया। हनुमान जी ने अभिमान रहित होकर कहा कि हे प्रभु, बंदर का बस यही बड़ा पुरुषार्थ है कि वह एक डाल से दूसरी डाल पर चला जाता है। मैंने जो समुद्र लांघ कर सोने का नगर जलाया और अशोक वन को उजाड़ डाला, यह सब तो हे रधुनाथ जी! आप ही का प्रताप है, हे नाथ! इसमें मेरी बड़ाई तो कुछ भी नहीं है—

बार-बार प्रभु चहई उठावा। प्रेम मगन तेहि उठब न भावा।।
प्रभु कर पंकज कपि कें सीसा। सुमरि सो दसा मगन गौरीसा।।
सावधान मन करि पुनि संकर। लागे कहन कथा अति सुंदर।।
कपि उठाइ प्रभु हृदयँ लगावा। कर गहि परम निकट बैठावा।।
कहु कपि रावन पालित लंका। केहि बिधि दहेउ दुर्ग अति बंका।।
प्रभु प्रसन्न जाना हनुमाना। बोला बचन बिगत अभिमाना।।
साखामृग कै बड़ि मनुसाई। साखा तें साखा पर जाई।
नाघि सिंधु हाटकपुर जारा। निसिचर गन बधि बिपिन उजारा।।
सो सब तव प्रताप रघुराई। नाथ न कछु मोरि प्रभुताई।।

जीवन में हम जो भी करते हैं, जो भी पाते हैं, वह सभी परम् पिता परमेश्वर की कृपा से हो रहा है। मनुष्य तो बस निमित् मात्र है। इसीलिये श्रीमद्भागवत् गीता में भी कहा गया है कि फल की कामना किये बिना कर्म करते चले जाना ही मनुष्य का धर्म है।

जहाँ तक ऑपरेटिंग सिस्टम विषय पर लिखी इस पुस्तक का प्रश्न है, यह आपके सामने प्रस्तुत है। मुझे पूर्ण विश्वास है कि यह पुस्तक आपके लिये उपयोगी सिद्ध होगी। इस पुस्तक के लेखन में अपने बहुमूल्य सुझाव प्रेषित करने हेतु लेखक समस्त मित्रों, अध्यापकों व छात्र-छात्राओं का आभार व्यक्त करता है। लेखक प्रकाशक का आभार व्यक्त करता है जिनके सुझावों से इस पुस्तक को अधिक उपयोगी बनाया जा सका। पुस्तक लेखन में मेरे परिवार के सदस्यों का सहयोग अकथनीय है, जिसके बिना इस पुस्तक का लिखा जाना सम्भव नहीं था।

पुस्तक के विषय में पाठकों के विचार, सुझाव एवं आलोचनायें सादर आमंत्रित है।

- लेखक

e-mail : scorp2427@yahoo.co.in
Mobile : 9457269124
facebook : rahul.wadhva.92

TRIALS

No matter what trials come our way.
GOD will see us through each day.
If only we'll put our trust in Him.
The light of His love will never dim.
Don't ever think that He's forgotten you.
It's just, sometimes, we need test or two.
For, we'll never be able to help our brother.
If we're never been tested, one way or another.
Never ask the Lord for an easy road,
But only for strength to carry your load.
For through your sorrows, you will find,
The Lord is with you,
He's loving and kind.
And when your trials have all gone away,
"I've been there, too," is what you can say.
For, there'll be times when someone you meet,
Will praise God to walk in the steps of your feet.

Syllabus

ऑपरेटिंग सिस्टम
(Operating System)

L	T	P
4	–	4

Rationale

The course provides the students with an understanding of human computer interface existing in computer system and the basic concepts of operating system and its working. The students will also get hands-on experience and good working knowledge to work in windows and Linux environments. The aim is to gain proficiency in using various operating systems after undergoing this course. While imparting instructions, the teachers are expected to lay more emphasis on concepts and principles of operating systems, its features and practical utility.

Learning Outcomes

After undergoing the subject, students will be able to:

- describe various types and services of operating system.
- identify the concept of process, various states in the process and their scheduling.
- classify different types of schedulers and scheduling algorithms.
- identify the significance of inter-process communication and synchronization.
- describe deadlock and the various ways to recover from deadlock.
- identify memory management techniques.
- describe virtual memory and its underlying concepts.
- describe the features and brief history of Linux.
- use General purpose commands and filters of Linux.
- use of shell scripts in Linux.

TOPIC WISE DISTRIBUTION OF PERIODS

S. No.	Topics	Time Allotted (Periods)	Marks Allotted (%)
1.	Overview of Operating Systems	10	18
2.	Process Management	10	18
3.	Deadlocks	06	10
4.	Memory Management Function	10	18
5.	I/O Management Functions	04	08
6.	File Management	06	10
7.	Linux Operating System	10	18
	Total	56	100

DETAILED CONTENTS

1. **Overview of Operating Systems** (10 Periods)

Definition of Operating Systems, Types of Operating Systems, Operating System Services, User operating system interface, System Calls, Types of System Calls, System Programs, Operating System Structure, Virtual Machine, Benefits of Virtual Machine.

2. **Process Management** (Principles and Brief Concept) (10 Periods)

Process concept, Process State, Process Control Block, Scheduling Queues, Scheduler, Job Scheduler, Process Scheduler, Context Switch, Operations on Processes, Interprocess Communication, Shared Memory Systems, Message-Passing Systems, CPU Scheduler, Scheduling Criteria, Scheduling Algorithms, Preemptive and Non Preemptive, First come first serve (FCFS), Shortest Job first (SJF), Round Robin (RR), Multiprocessor scheduling, Process Synchronization.

3. **Deadlocks** (Principles and Brief Concept) (06 periods)

Deadlock, Conditions for Dead lock, Methods for handling deadlocks, Dead Prevention, Deadlock Avoidance, Deadlock detection, Recovery from deadlock.

4. **Memory Management Function** (Principles and Brief Concept) (10 periods)

Definition — Logical and Physical address Space, Swapping, Memory allocation, Contiguous Memory allocation, Fixed and variable partition, Internal and External fragmentation and Compaction, Paging — Principle of operation, Page allocation, Hardware support for paging, Protection and sharing, Disadvantages of paging, Segmentation, Virtual Memory.

5. **I/O Management Functions** (Principles and Brief Concept) (04 periods)

Dedicated Devices, Shared Devices, I/O Devices, Storage Devices, Buffering, Spooling.

6. **File Management** (Principles and Brief Concept) (06 periods)

Types of File System; Simple file system, Basic file system, Logical file system, Physical file system, Various Methods of Allocating Disk Space

7. **Linux Operating System** (10 Periods)

History of Linux and Unix, Linux Overview, Structure of Linux, Linux releases, Open Linux, Linux System Requirements, Linux Commands and Filters: mkdir, cd,rmdir,pwd, ls, who, whoami, date, cat, chmod, cp, mv, rm, pg, more, pr, tail, head, cut, paste, nl, grep, wc, sort, kill, write, talk, mseg, wall, merge, mail, news Shell: concepts of command options, input, output, redirection, pipes, redirecting and piping with standard errors, Shell scripts, vi editing commands.

LIST OF PRACTICALS

1. Demonstration of all the controls provided in windows control panel.
2. Exercise on Basics of windows.
3. Installation of Linux Operating System.
4. Usage of directory management commands of Linux: ls, cd, pwd, mkdir, rmdir.
5. Usage of File Management commands of Linux: cat, chmod, cp, mv, rm, pg, more, find.
6. Use the general purpose commands of Linux: wc, od, lp, cal , date, who, whoami.
7. Using the simple filters: pr, head, tail, cut, paste, nl, sort.
8. Communication Commands: news, write, talk, mseg, mail, wall.
9. Write a shell program that finds the factorial of a number.
10. Write a shell program that finds whether a given number is prime or not.
11. Write a shell program to find the average of three numbers.
12. Write a shell progrann that will convert all the text of the file from lowercase to uppercase.

विषय-सूची

1 Chapter

कम्प्यूटर का संक्षिप्त परिचय
(BRIEF INTRODUCTION OF COMPUTER)

THINK ABOUT IT

The whole world
Is in fetters bound
Men are slaves
To their passions and appetites!
He amongst me is truly free
Who hath renounced
His freedom to the Lotus Feet of the Guru!
The Guru is the greatest liberator
The redeemer of the soul
He leads out of untruth into truth!
Out of darkness into light!
Out of death into the deathless state!
Blessed be His name!

— J. P. Vasvani

§ 1.1. परिचय (Introduction) :

कम्प्यूटर आधुनिक तकनीक की महान खोज है। कम्प्यूटर सूचना क्रांति का अग्रदूत बन गया है। विद्यालयों में कम्प्यूटर शिक्षा अनिवार्य-सी होती जा रही है। आज देश के हर क्षेत्र में कम्प्यूटर प्रणाली से काम लेने पर ज़ोर दिया जा रहा है। बस, रेलवे और हवाई जहाज का किसी भी शहर का आरक्षण किसी भी शहर से कम्प्यूटर द्वारा संभव है। बैंकों के लिए कम्प्यूटर सचमुच एक वरदान साबित हुआ है। परीक्षा परिषदों और विश्वविद्यालयों द्वारा अपने परीक्षा परिणामों को तैयार करने में कम्प्यूटर का प्रयोग किया जाता है। चुनावों के परिणाम, टेलीफोन, बिजली, पानी आदि के बिल, भवन निर्माण के लिए नक्शा तैयार करने में भी कम्प्यूटर महत्वपूर्ण भूमिका निभाता है। चिकित्सा के क्षेत्र में भी कम्प्यूटर का विशेष योगदान है। यह एक ऐसा डिवाइस है जिसका इस्तेमाल कई सारे उद्देश्यों के लिये किया जाता है जैसे—सूचनाओं को सुरक्षित रखना, ई-मेल, मैसेजिंग, सॉफ्टवेयर प्रोग्रामिंग, गणना, डेटा प्रौसेसिंग आदि।

सूचना और प्रौद्योगिकी के इस युग में हर कोई कम्प्यूटर से जुड़ता जा रहा है। फेसबुक और इंटरनैट मे पल-पल जुड़ती जा रही इस दुनिया में ई-मेल, ई-कॉमर्स, नैट-बैंकिग, ऑनलाइन रिजर्वेशन, यानि जीवन का हर पहलू कम्प्यूटर से जुड़ता जा रहा है। तरह-तरह के सॉफ्टवेयर विकसित किये जा रहे हैं, जो कि हर क्षेत्र में कार्य को आसान बनाते जा रहे हैं। ऐसे में, देश के युवाओं में भी सॉफ्टवेयर इंजीनियर बनने की इच्छा जागृत होना स्वाभाविक है। कम्प्यूटर के क्षेत्र से जुड़ने के बाद देश के युवा इस क्षेत्र में अत्यन्त उत्साह के साथ नित नई ऊँचाईयों पर पहुँचने के सपने बुनने लगते हैं। लेकिन जब भी आप कोई ऊँची

इमारत बनाने का ख्वाब देखते हैं, तो उसके लिये सबसे पहले ध्यान देना होता है, उस इमारत की नींव पर। नींव यदि गहरी और मजबूत होगी तो इमारत को अधिक ऊँचाईयों तक निश्चित होकर ले जाया जा सकता है। कम्प्यूटर इंजीनियरिंग के साथ भी यही है। बड़े-बड़े कम्प्यूटर प्रोग्राम व सॉफ्टवेयर विकसित करने वाले कम्प्यूटर के महारथी भी एक दिन fundamentals से ही शुरू करते हैं और आप भी fundamentals से ही शुरू कर रहे हैं। अत:, एक सफल कम्प्यूटर इंजीनियर बनने के लिये Hardware व Software का समुचित ज्ञान आपको रखना ही होगा। Operating system व कम्प्यूटर भाषाओं में महारथ हासिल करेंगे तभी बन पायेंगे, एक सफल कम्प्यूटर इंजीनियर। अत: operaing sysem के विषय में गहनता से अध्ययन करना प्रारम्भ करें तथा मेरी राय तो यह है कि कम्प्यूटर की कोई language (जैसे C and C++) को भी प्रथम वर्ष से सीखना प्रारम्भ कर लें क्योंकि यदि प्रथम वर्ष में language के basics सीख लेंगे तो द्वितीय वर्ष तथा अंतिम वर्ष में Advanced Programming सीख सकेंगे तथा language पर आपकी बेहतरीन command हो जायेगी।

§ 1.2. कम्प्यूटर क्या है?

कम्प्यूटर एक इलैक्ट्रॉनिक डिवाइस है जो कि विभिन्न कार्यों को प्रोग्राम के द्वारा कर सकता है। निर्देशों का समूह प्रोग्राम कहलाता है।

"A computer is an electronic device that can perform a variety of operations in accordance with a set of instructions called program."

वास्तव में, कम्प्यूटर्स द्वारा डाटा को मनुष्य की तुलना में लाखों गुना तेजी से access (पहुँच बनाना) तथा process किया जा सकता है। कम्प्यूटर अपनी मैमोरी में डाटा तथा Information स्टोर कर सकता है, उसको process कर सकता है, तथा वाँछित results को उत्पन्न कर सकता है। अत: computer को Data processor के रूप में Use किया जाता है।

चूँकि, कम्प्यूटर के सम्बंध में Data तथा Information शब्दों का अक्सर प्रयोग किया जाता है, अत: आपके लिए यह जानना जरूरी है कि Data तथा Information क्या होते हैं?

Data—डेटा का तात्पर्य Raw facts तथा Figures से है। जैसे–Rajeev, 1983, –159, 723.6, Sunaina, 'B' आदि Data हैं। Data को Process करके Information प्राप्त की जा सकती है।

"The raw facts and figures are called data."

Information—डाटा को Process करने के बाद जो प्राप्त होता है, उसको सूचना या Information कहते हैं अर्थात् Meaningful Data अर्थात् Data जो कि किसी न किसी रूप में अर्थपूर्ण हो, Information कहलाती है। उदाहरणत: Rajeev was born in 1983, यह एक Information है। Rajeev got 'B' grade in Mathematics यह भी एक Information है। Rajeev and Sunaina live in Gorakhpur यह भी एक Information है।

"Information is what we got after processing the data ie. Meaningful data is known as information. Data are aggregated and summarized in various meaningful ways to form information."

अत: संक्षेप में आप यह समझ लें कि Computer के Input पर जो दिया जाता है, वह data है तथा Output पर जो प्राप्त होता है, वह Information है। आसान शब्दों में कहा जाये, तो data को आप किसी फैक्टरी में जाने वाले कच्चे सामान (raw material) की तरह समझ सकते हैं तथा Information को उस factory से बाहर आने वाले उत्पाद (Product) की तरह समझ सकते हैं। Data को Information में बदलने की प्रक्रिया Information Processing Cycle कहलाती है।

इनपुट, प्रोसेस व आउटपुट (Input, Process and Output)—कम्प्यूटर में Input, Process तथा Output आदि शब्दों का कई बार प्रयोग होता है। आपके लिये यह जानना आवश्यक है कि इन शब्दों का क्या अर्थ है। आइये, कुछ उदाहरणों की सहायता से इनका अर्थ समझने की कोशिश करते हैं—

(i) मान लीजिये कि आप अपना रूम Decorate करना (सजावट करना) चाहते हैं। आप Market जाकर Flowers, posters तथा Decorative items (सजावटी सामान) लाते हैं तथा अपने Room को Decorate करते हैं। अब इसमें Input, Process तथा आउटपुट क्या हुए? Input का अर्थ है कि आप क्या करना चाहते हैं, जैसे कि इस उदाहरण में इनपुट हैं, आपका अपना रूम Decorate करने की इच्छा। Process का अर्थ है कि वास्तव में क्या-क्या कार्य

किये गये, जैसे कि इस उदाहरण में Process है आपका Decorative सामान लेने के लिये बाजार जाना व सामान खरीदकर लाना। Output का अर्थ है कि परिणाम (Result) क्या हुआ जैसा कि इस उदाहरण में Result है आपका Decorate किया हुआ room।

(ii) आइये एक अन्य उदाहरण लेते हैं। मान लीजिए कि आपने अपने छोटे भाई को दिल्ली से कानपुर जाने को कहा तथा वह ट्रेन पकड़कर कानपुर चला गया। अब बताइये कि इसमें Input क्या है? Process क्या है तथा Output क्या है?

Input—आपका अपने छोटे भाई को ट्रेन में बैठकर जाने को कहना।

Process—आपके भाई का ट्रेन में बैठकर कानपुर जाना।

Result—आपके भाई का कानपुर पहुँचना।

(iii) **विचार प्रश्न**—मान लीजिये कि आप अपने मित्र से फोन पर बात करना चाहते है। आपने उसका नम्बर मिलाया और उससे बात की। इसमें इनपुट Process व आउटपुट क्या-क्या हैं?

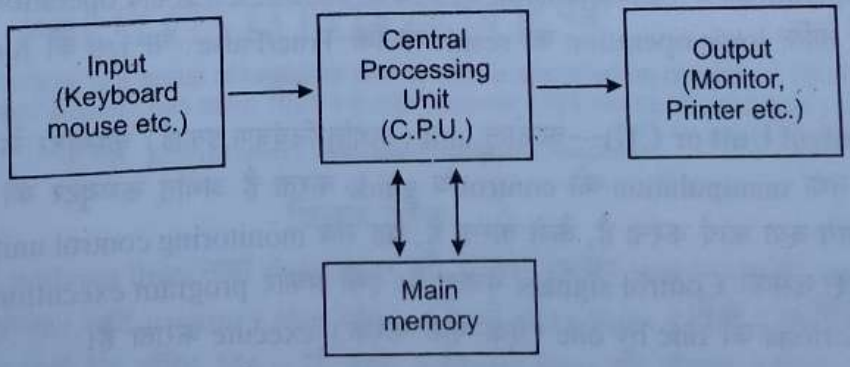
उक्त उदाहरणों से यह स्पष्ट है किसी कार्य को करने के लिये जो चीजें आवश्यक हैं उन्हें Input कहते हैं, जो Actual Work अर्थात् वास्तविक कार्य किया जाता है उसे Process कहते हैं तथा जो Final result प्राप्त होता है, उसे आउटपुट कहते हैं। अत:, हम जब कोई Work करते हैं तो यह Input-Process-Output Cycle जिसे Short में IPO कह सकते हैं क्रियान्वित होती है। Computer भी इसी प्रकार से कार्य करता है। इनपुट प्राप्त करता है, उसकी Processing करता है तथा आउटपुट प्रदान करता है।

अभ्यास प्रश्न

(i) What is data?
(ii) What is information?
(iii) What is process?
(iv) Which of the following statement is true—
(a) Data is processed to get information.
(b) Information is processed to get data.
(v) The full name of CPU is .....
(vi) डेटा की processing करके प्राप्त होती है तथा यह information अगली processing के लिये कार्य करती है।

§ 1.3. कम्प्यूटर के कार्यकारी घटक (Functional Components of a Computer) :

जैसा कि अभी आपने देखा कि कम्प्यूटर अपना कार्य करने हेतु INPUT-PROCESS-OUTPUT CYCLE को Follow करता है, अत: कम्प्यूटर में एक Input unit होती है, Processing करने हेतु Central Processing Unit या CPU होता है तथा Output करने हेतु Output Unit होता है। अत: Computer का block diagram चित्र 1.1 में प्रदर्शित है। Input data तथा Processing के समय Intermediate data आदि की Storage हेतु कम्प्यूटर में Memory होती है।



चित्र 1.1—Block diagram of a Computer

इनपुट यूनिट (Input Unit)—कम्प्यूटर में Data input करने हेतु कई Input Devices कम्प्यूटर से Attached होती है। जैसे Keyboard, mouse, magnetic ink character reader (MICR), optical mark reader (OMR),optical character reader (OCR), Joystick आदि को विभिन्न रूपों में कम्प्यूटर में Data input करने हेतु Use किया जाता है।

इनपुट यूनिट न केवल user से इनपुट प्राप्त करता है, बल्कि उसको उन forms (प्रारूप) में Convert कर देता है, जो कि कम्प्यूटर द्वारा समझी जा सके। यह तो आप अच्छी तरह से जानते होगें कि कम्प्यूटर केवल Binary भाषा (अर्थात् 1 तथा 0 की भाषा) जिसको machine language भी कहा जाता है ही जानते हैं और समझते हैं अर्थात् ON/OFF या High/ low या 1 और 0 इत्यादि। अत: user से प्राप्त इनपुट को Binary language मे Convert करना Input unit की ही जिम्मेदारी होती है। *"An Input unit takes the input from the user and converts it into binary form so that it can be understood by the computer."*

अत:, कम्प्यूटर की इनुपट पर डाटा तथा Instructions दिये जाते हैं तथा इन instructions को follow करते हुए ही कम्प्यूटर द्वारा data की processing की जाती है। उदाहरणत: यदि आपने कम्प्यूटर से कहा कि 5 तथा 6 को add करो, तो 5 व 6 data हैं तथा add एक instruction हैं ऐसे ही, यादि आपने कम्प्यूटर स्क्रीन पर Rajeev Print करना है तो 'Rajeev' data है तथा 'प्रिंट करो' Print Instruction। अब आप मुझे यह बताइये कि पिछले खण्ड में जो तीन example मैंने consider किये थे, उसमें data क्या है, तथा instruction क्या हैं?

सैंट्रल प्रोसैसिंग यूनिट (Central Processing Unit)—Central processing unit या 'CPU कम्प्यूटर का control center अर्थात् उसका दिमाग होता है। यह computer को guide, direct तथा govern करता है। यहाँ पर एक बात को आपने बहुत ध्यान से समझ लेना चाहिए कि हालांकि CPU कम्प्यूटर का Brain कहा जाता है किन्तु इसकी स्वंय की Decision making power (निर्णय लेने की क्षमता) नहीं होती न ही इसकी कोई IQ होती है। कम्प्यूटर अपना प्रत्येक step में user द्वारा दिये गये instruction व program को ही follow करता है तथा सभी निर्णय user द्वारा दिये गये प्रोग्राम (निर्देशों) के अनुसार ही लेता है। बिना instruction व programs के CPU कुछ नहीं कर सकता। अत: यदि आपको कम्प्यूटर द्वारा छोटे से छोटा कार्य भी कराना है तो उसके लिए आपको computer को instructions देने होंगे या program लिखना होगा। computer के लिये गये programms computer की भाषा में होते हैं जिनको computer समझ सकता है। अत: कम्प्यूटर से कोई भी कार्य करवाने हेतु आपको computer language सीखनी पड़ती है तथा programs लिखने पड़ते हैं, तभी आप कम्प्यूटर से कोई कार्य करवा सकते हैं। CPU के दो Components होते हैं जो कि विभिन्न फंक्शन्स करते हैं—Control unit या CU तथा Arthmetic logic unit (अर्थात् ALU)।

एरिथमैटिक तथा लॉजिक यूनिट (Arithmetic and Logic Unit is ALU)—Arithmetic operations अर्थात् गणितीय Operations अर्थात् plus, minus, multiply तथा divide। इसी प्रकार Logical operations अर्थात् तार्किक operations अर्थात् तर्क के आधार पर किये जाने वाली operations अर्थात् AND, OR, NOT। कम्प्यूटर का ALU Arithmetic व logic करता है। अत:, जब भी किसी data पर logic या Arithmetic operations करना होता है तो यह datas memory से ALU में भेजा जाता है ताकि ALU required operation को perform करके results को वापिस memory में भेज देता है।

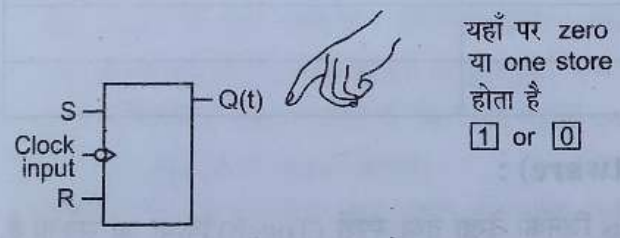
इसी प्रकार logical operations तथा Comparision <, >, = =, <=, >=, < > इत्यादि operations हेतु डाटा भी Memory से ALU में भेजा जाता है ताकि logic operation का result जो कि True/False, या 1/0 की form में होता है, memory को भेज दिया जाता हैं

कन्ट्रोल यूनिट (Control Unit or CU)—कन्ट्रोल unit (अर्थात् नियंत्रण इकाई) कम्प्यूटर के data तथा information के interpretation flow तथा manipulation को control व guide करता है अर्थात् कम्प्यूटर को किस sequence (क्रम) में कार्य करना है, किस समय क्या कार्य करना है, कैसे करना है, यह सब monitoring control unit द्वारा की जाती है। ALU द्वारा कार्य सम्पादन हेतु CU उसको Control signals भेजता है। इसी प्रकार program execution के समय control unit program के सभी instructions को one by one (एक-एक करके) execute कराता है।

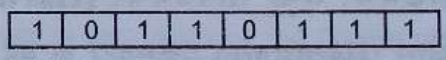
आधुनिक कम्प्यूटर्स में ALU तथा CU को single चिप पर Integrate कर दिया जाता है तथा यह चिप्स (अर्थात् ICs) Micro processors या म्यू-पी (μP) कहलाती है।

आउटपुट यूनिट (Output Unit)—कम्प्यूटर से आउटपुट प्राप्त करने हेतु कई आउटपुट डिवासेस (अर्थात आउटपुट युक्तियाँ) भी कम्प्यूटर से Attached (जुड़ी) रहते हैं। CPU से आने वाली आउटपुट electronic binary form में होती है जिसको कि ऐसा form में convert करना जरूरी होता है जो कि user द्वारा आसानी से समझी जा सके अर्थात Characters, graphical, audio visual इत्यादि। यह कार्य Output Unit द्वारा किया जाता है। Output Unit के उदाहरण हैं—VDU (Visual Display unit) या Monitor, Printer, Plotter, Coder, Speech Synthesizer इत्यादि।

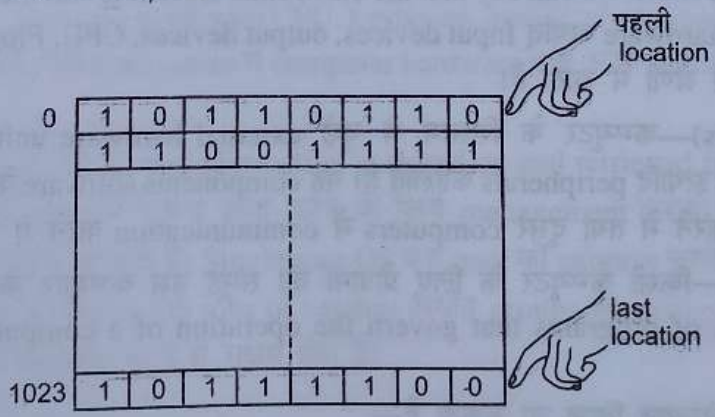
मैमोरी (Memory)—कम्प्यूटर के डाटा, प्रोग्राम व Information को स्टोर करने हेतु कम्प्यूटर की मैमोरी Semiconductor memory होती है तथा यह data को Binary form में अर्थात् 1 और 0 के form में स्टोर करती है। Binary digits को short में bits कहा जाता है। एक bit को store करने हेतु एक flip flop का use होता है (Fig. 1.2) flip flop भी electronic circuit होता है। अत: bit का अर्थ होता है 1 या 0। Flip flop के group को register कहा जाता है। 8 bit का समूह byte कहलाता है तथा 4 bit का समूह nibble कहलाता है। Bits का समूह word कहलाता है अर्थात् 1 register में एक word store होता है। यदि word size 4 bit है और memory में 4 bit के 512 word store करने की क्षमता है तो मैमोरी का size होगा 512×4 bits। अत: मैमोरी की संरचना को आप एक बहुमंजिली इमारत (Multistoreyed Building) के समान मान सकते हो जिसमे प्रत्येक floor में 4 या 8 कमरे हैं तथा ऐसी कई Floors हैं। For example: If the memory size is specified as 1024×8 bits, this means that word size is 8 bits and the memory is capable of storing 1024 words, each of bit size 8.



(a) Symbol of flip flop (flip flop in an electronic circuit which can use on binary digit (bit) i.e. 1 or 0.



(b) A register is a group of flip flops which store a set of bits called a binary word.



(c) A semiconductor memory is a group of registers and its size is specified as number of binary words it can store for example this memory can store 1024 × 8 bits (means 1024 words of word size 8 each).

चित्र 1.2—Memory organisation in a Computer

विचार प्रश्न

यदि किसी memory में address lines तथा data lines की संख्या उसका word size व capacity पता होने पर ज्ञात की जा सकती है, उदाहरणत: यदि memory size 256×4 है तो data lines 4 होगी (क्योंकि word size 4 है) तथा address lines हेतु आप देखें कि चूँकि $256 = 2^8$, अत: address lines की संख्या 8 होगी। अब निम्न तालिका को पूरा कीजिये—

Memoery Size	No of Address lines	No of Data lines
256 × 4	8	4
256 × 8	8	8
512 × 4	9	4
512 × 8		
1024 × 8		
2048 × 8		
4096 × 8		
4 8K × (1K = 1024)		
8K × 8		
16 K × 8		
64 K × 8		
65, 536 × 8		
32 K × 4		
64 kilobyte		
500 × 5		

§ 1.4. हार्डवेयर तथा साफ्टवेयर (Hardware and Software) :

हार्डवेयर (Hardware)—कम्प्यूटर के वह घटक Components जिनको देखा तथा स्पर्श (Touch) किया जा सकता है, Hardware कहलाते हैं अर्थात् Hardware represents the physical and tangible components of the computer that can be seen and touched. In short, we can say that the electronic, electrical and mechanical equipments that make a computer is called hardware अर्थात् Input devices, output devices, CPU, Floppy disc, Hard disc इत्यादि यह सभी कम्प्यूटर हार्डवेयर की श्रेणी में आता है।

पैरीफैरल्स (Peripherals)—कम्प्यूटर के सिस्टम से जुड़े external hardware units जैसे keyboard, mouse, speakers, printers, monitors इत्यादि peripherals कहलाते हैं। यह components software के साथ मिलकर Calculations करने में, data को organise करने में तथा दूसरे computers से communication करने में सहायक होते हैं।

साफ्टवेयर (Software)—किसी कम्प्यूटर के लिए प्रोग्रामों का समूह उस कम्प्यूटर का software कहलाता है i.e. software represents the set of programs that govern the operation of a computer system and make the hardware run.

साफ्टवेयर का वर्गीकरण निम्नवत् किया जा सकता है—

(i) Operating system
(ii) Language Processors
(iii) Application software

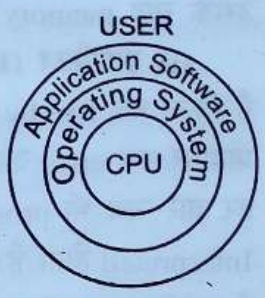
विचार प्रश्न—Without software, a computer will be of no use. Do you agree? अर्थात बिना Software के computer hardware की ठीक ऐसी ही स्थिति हो जायेगी, जैसे कि बिना जान का मानव शरीर। क्या आप सहमत हैं?

§ 1.5. ऑपरेटिंग सिस्टम (Operating System) :

ऑपरेटिंग सिस्टम एक प्रोग्राम होता है जो कि user तथा hardware के मध्य Interface (अर्थात् मध्यस्थ) का कार्य करता है।

"An operating system is a program which acts as an interface between a user and the hardware."

अत: operating system के बिना कम्प्यूटर से कोई भी कार्य लेना सम्भव नहीं है। अत: ऑपरेटिंग सिस्टम कम्प्यूटर का एक महत्वपूर्ण घटक है जो कि कम्प्यूटर के अन्य घटकों का नियन्त्रण करता है। अत: कम्प्यूटर के मुख्य कम्पोनैंट्स हैं—हार्डवेयर, ऑपरेटिंग सिस्टम, एप्लीकेशन साफ्टवेयर (चित्र 1.3 देखें)। हार्डवेयर में Basic computing resources होते हैं, application program routines द्वारा users computations problems solve करने के लिए computer resources को utilize किया जाता है तथा आपरेटिंग सिस्टम application program द्वारा hardware के use को control तथा coordinate करता है। कहने का तत्पर्य यह है कि user जब कोई application program run करता है तो इस सम्बन्ध में सभी क्रियायें कम्प्यूटर के हार्डवेयर के अंदर लगे परिपथों में सम्पादित होती हैं। हार्डवेयर को कन्ट्रोल तथा coordinate करने का कार्य operating system द्वारा किया जाता है। अत: operating system कम्प्यूटर के हार्डवेयर को useable (इस्तेमाल करने योग्य) बनाना है। अत: operating system एक ऐसा program है जो कि hardware को useable बनाता है। An operating system may also be defined as a set of programs that controls the computer. Some examples of operating systems are MSDOS, MS Windows, UNIX, VMS, Chicago etc.



चित्र 1.3—Operating System's role in a Computer System

अत: संक्षेप में कहा जाये तो ऑपरेटिंग सिस्टम निम्न महत्वपूर्ण कार्य करता है—

(i) Provide the instructions to prepare user interface अर्थात् user तथा computer के मध्य interaction (तालमेल) प्रदान करता है कि किस प्रकार user computer से communicate करें (typed commands द्वारा या Graphical symbols द्वारा)।

(ii) Loads necessary programs into the computer memory which are required for proper computer functioning अर्थात् computer को start करने के बाद वह सभी program जो कि computer की functioning के लिये आवश्यक होते हैं, उनको ऑपरेटिंग सिस्टम स्वंय लोड करता है।

(iii) Coordinates how programs work with CPU, keyboard, mouse, printer and other hardware as well as with other software. अर्थात् programs व computer hardware तथा अन्य Software के मध्य सामंजस्य स्थापित करना।

(iv) Manages the way in which the information is stored on and retrieved from the disks अर्थात् diskमें सूचना स्टोर करने या disk से सूचना प्राप्त करने के लिये management करना।

Operating System कई प्रकार के होते है। Single user OS एक user को support करते हैं, multiuser OS कई users को support करते है। Batch processing OS jobs of batches अर्थात् groups को process करते हैं। Multiprocessing OS कई CPU's को एक साथ Handle करने में सक्षम होते हैं।

§ 1.6. लैंगुएज प्रोसैसर (Language Processors) :

कम्प्यूटर प्रोग्रामर अपने Programs को high level language HLL's में लिखते हैं क्योंकि इन languages में code (coding अर्थात् कम्प्यूटर की भाषा में प्रोग्राम लिखना) करना अपेक्षाकृत आसान होता है। लेकिन कम्प्यूटर केवल machine language जो कि binary अर्थात् 1 तथा 0 के रूप में होती है समझता है। अत: कम्प्यूटर High level language जैसे C, C++, Java इत्यादि में लिखें प्रोग्राम को तभी समझ पायेगा जब वह प्रोग्राम machine language में बदल दिया जायेगा। वह प्रोग्रम जो कि HLL को binary form में बदलते हैं, language processors कहलाते हैं। Language processors मुख्यत: तीन प्रकार के होते हैं—Assemblers, Compilers तथा interpreters.

(i) कम्पाइलर्स (Compliers)—Compilers वह प्रोग्राम होते हैं जो कि High level language program को machine language प्रोग्राम में convert करते हैं यह पूरे High language program को एक ही बार में Convert कर देते हैं तथा उनके errors को report कर देते हैं। इन errors को प्रोग्रामर correct करता है, तथा इसके बाद प्रोग्राम पुन: compile किया जाता है। जब सब errors समाप्त हो जाते हैं तो उस प्रोग्रम को एक object program Binary रूप में बन जाता है। इसके बाद memory से Compiler को remove किया जा सकता है।

(ii) इटरप्रैटर (Interpreter)—Interpreter भी high level language को machine language में Convert करता है किन्तु यह line by line conversion करता है। यदि किसी line में कोई error मिलता है तो यह reporting करता है, तथा प्रोग्रामर उस error को correct करता है। उसके बाद ही अगली लाइन का conversion होता है। Interpretor program को हर बार जब भी program का execution होता है, Memory में होना चाहिये क्योंकि प्रत्येक बार प्रोग्राम run होने से पहले Interpreted होता है। Error debugging (debugging अर्थात् error खोजना) के लिये Interperter अत्यंत उपयोगी होता है क्योंकि वह तुरंत line by line error report करता है किन्तु errors के Correct हो जाने के बाद भी Interperter को मैमोरी में ही रखना पड़ता है, जिससे मैमोरी का unnecessary usage (बेवजह इस्तेमाल) होता है। अत: Interperter व compilers को यदि combine किया जाये तो यह HLL program को Object program में translate करने हेतु best combination है। Error removal हेतु Interperter को use किया जाये तथा सब errors remove हो जाने के बाद program को compile कर दिया जाये, और फिर language translator को memory से remove किया जा सकता है।

(iii) एसैम्बलर (Assembler)—Assemblers Assembly language programs को machine language में convert करता है। Assembly language program सामान्यत: microprocessors के लिये लिए जाते हैं तथा English like letters जिसको Mnemonics (निमोनिक्स) कहा जाता है, के form में होते हैं।

A Compiler checks the entire program written by user and if it is free from eror and mistakes then produces a complete program in Machine language known as object program. But if it founds some error in program then it does not execute the single statement of the program. So compiler translate whole program in Machine language before it starts execution.

Interpreters translates one statement at a time (i.e. line by line) and if it is error free then executes that statement. This continues till the last statement in the program has been translated and executed. Thus Interpreter translates and executes the statement before it goes to next statement. When it found some error in statement it will immediately stop the execution of the program.

Error finding is much easier in Interpreter because it checks and executes each statement at a time. So wherever it fine some error it will stop the execution. Where Compiler first check all the statement for error and provide lists of all the errors in the program.

Interpreter take more time for the execution of a program compared to Compilers because it translates and executes each statement one by one.

§ 1.7. एप्लीकेशन साफ्टवेयर (Application Software) :

एप्लीकेशन प्रोग्राम programs का समूह होता है जो कि किसी विशष एप्लीकेशन हेतु आवश्यक operation को करता है।

"An application software is a set of programs necessary to carry out operations for specific applications."

अत: प्रत्येक application के लिए अलग Software होता है। उदाहरणत: library management के software को Hotel management के लिये use नहीं किया जा सकता तथा Hotel management के software को library management के लिये use नहीं किया जा सकता । अत: प्रत्येक application के requirements को ध्यान में रखते हुये software engineers उन requirements के अनुसार उस particular application हेतु application software design करते हैं।

एप्लीकेशन साफ्टवेयर को निम्नवत् वर्गीकृत किया जा सकता है—

Customized Application Software—यह user की requirement के अनुसार tailor-made software होता है। अत: user द्वारा specify की गई सभी requirements को यह पूरा करता है। अत: यह किसी दूसरे user के work place पर directly install नहीं किया जा सकता क्योंकि हो सकता है कि दूसरे user की requirements पहले user से जिसके लिये वह particular customize application software develope किया गया है, से भिन्न हों तथा new user के requirements के अनुसार यह software fit न बैठता हो।

General Application Software—किसी कार्य को करने हेतु general requirements (सामान्य आवश्यकताओं) को ध्यान में रखते हुये general application software design किये जाते हैं इसको कई users use कर सकते हैं।

अत:, मोटे तौर पर, मुझे लगता है कि Customized Application Software तथा General application software में ठीक वैसी ही भिन्नता होती है जैसी दर्जी द्वारा सिले गये कपड़ों तथा readymade कपड़ों में होती है क्योंकि दर्जी आपका पूरा नाप लेकर तथा आपके बताये गये निर्देशों के अनुसार कपड़े सिलता है जबकि readymade garments एक आम user की general requirements को ध्यान में रखकर सिले जाते हैं। क्या आप सहमत हैं?

Some important examples of Applications Software—

(i) Inventory Control
(ii) Financial accounting
(iii) Railway reservation
(iv) Medical field
(v) Results preparation
(vi) Billing software etc.

विचार प्रश्न

(i) आपरेटिंग सिस्टम क्या होता है? इसके प्रमुख कार्य क्या होते हैं?

(ii) Compiler व Interpreter में क्या अन्तर होता है?

(iii) Compiler व Assembler में क्या अन्तर होता है?

(iv) निम्न में Hardware तथा Software छाँटिये–

(a) Memory	**(b) Motherboard**	**(c) Power Supply**
(d) Windows	**(e) Compiler**	**(f) Assembler**
(g) Interpreter	**(h) MS-Office**	**(i) MS-DOS**
(j) Mouse	**(k) Keyboard**	**(l) Printer**
(m) Monitor	**(n) C-program**	**(o) Unix**
(p) Linux	**(q) CPU**	**(r) Joystick**
(s) Visual Basic	**(t) Java**	**(u) Fortran**
(v) BASIC	**(w) HTML**	**(x) Modem**
(y) Pen drive	**(z) Hard Disk Drive (HDD)**	

(v) निम्न में से कौन-सा कथन असत्य है—

(a) Computer High level language नहीं समझ सकता।

(b) Compiler एक ऐसा Hardware device है जो कि High level language में लिखे गये program को machine level binary language में change कर देता है।

(c) MS-office एक application software है।

(d) DOS CUI operating system है जबकि Windows GUI operating system है।

§ 1.8. कम्प्यूटर की Strength तथा Weaknesses (Strength and Weaknesses of a Computer) :

आपने देखा होगा कि आज कम्प्यूटर का उपयोग हर जगह हो रहा हैं बैंकिंग, रेलवे, वैद्युत बिलिंग, होटल, कॉलेज, मॉल्स, डिफैन्स, डिजाइनिंग आदि लगभग हर क्षेत्र में कम्प्यूटर अपनी भूमिका निभा रहा है।

हर क्षेत्र में होने वाले कम्प्यूटरीकरण के मुख्य लाभ निम्नवत् हैं—

(i) **स्पीड (Speed)**—कम्प्यूटर की स्पीड मानवीय स्पीड से बहुत ज्यादा होती है। आधुनिक कम्प्यूटर्स एक सैकेण्ड में लाखों instructions excute कर सकते हैं।

(ii) **उच्च स्टोरेज कैपेसिटी (High Storage Capacity)**—कम्प्यूटर को memory में बहुत कम space में बहुत अधिक Information store की जा सकती है।

(iii) **विश्वसनीयता (Reliability)**—कम्प्यूटर को थकान या boredom नहीं होती है। वह Human beings से ज्यादा reliable (अर्थात् विश्वसनीय) होते हैं।

(iv) **परिशुद्धता (Accuracy)**—कम्प्यूटर अत्यधिक accuracy से कार्य कर सकते हैं।

(v) **वरसैलिटी (Versality)**—कम्प्यूटर हरफनमौला होते हैं। Hazardous environment (खतरनाक वातावरण) जहाँ Human beings के लिये खतरा हो सकता है, कम्प्यूटर्स instal करके काम लिया जा सकता है। कम्प्यूटर्स graphic audio, visual आदि information को भी provide कर सकते हैं।

कम्प्यूटर्स के इतने सारे लाभ होने के बावजूद कम्प्यूटर्स की कुछ कमियाँ भी हैं जैसे—

(i) Lack of decision making power.

(ii) Zero IQ

अत: कम्प्यूटर्स में न तो decision (निर्णय) लेने की क्षमता होती है तथा न ही उनकी कोई अपनी IQ (बुद्धिमत्ता) होती है। कम्प्यूटर से काम लेने के लिए छोटे से छोटे step को भी उन्हें बताना पड़ता है। अत: मानव और कम्प्यूटर में सबसे बड़ा अन्तर यही है कि मानव का छोटा बच्चा भी अपनी बुद्धि रखता है किन्तु बड़े से बड़ा कम्प्यूटर भी IQ या decision making power नहीं रखता । इसीलिये, programs के जरिये कम्प्यूटर्स को instructions देनी पड़ती है तभी वह कोई कार्य कर पाता है। इसीलिये इतनी सारी Programming languages हैं, जो कि एक software engineer को सीखनी पड़ती हैं इन languages के Data types, statements, syntax आप जब तक नहीं सीखेंगें, तब तक ना तो आप प्रोग्राम लिख पायेंगे तथा न ही आप कम्प्यूटर्स से कोई कार्य करवा पायेंगे। कम्प्यूटर language सीखने में आपको proper syntax (किसी भी कम्प्यूटर भाषा का grammar (अर्थात व्याकरण) उसका syntax कहलाता है) इसलिये सीखने पड़ते हैं क्योंकि आपने यदि program लिखने में छोटी सी भी error कर दी, तो compiler उसको नहीं समझ पायेगा (तथा error दे देगा) और program run नहीं कर पायेगा। इसीलिये, यदि आप कोई प्रोग्रामिंग भाषा (जैसे C, C++, Jawa इत्यादि) सीखते हैं तो उस language के सारे statements और command proper तरीके से सीखने होंगे।

§ 1.9. प्रोग्राम डैवलपमैन्ट के विभिन्न स्टैप्स (Steps of Program Development Process) :

जैसा कि आपने पढ़ा कि कम्प्यूटर से कोई भी कार्य कराने हेतु Programming करनी पड़ती है तथा इसके लिए आपको Computer Languages सीखनी पड़ती है। उस कम्प्यूटर भाषा में program लिखकर ही आप कम्प्यूटर से कोई कार्य करवा सकते हैं।

Program एक निर्देशों का समूह होता है जिसके अनुसार कम्प्यूटर कार्य करता है। अत: कम्प्यूटर से कोई भी कार्य कराने हेतु जो निर्देश आप देते हैं उसका Complete set program कहलाता है।

Program द्वारा inputs की processing करके उनको outputs में transform किया जाता है। अत: प्रोग्राम लिखने से पहले आपको इस बात की पूरी जानकारी होनी चाहिए कि कौन-कौन से इनपुट प्रोग्राम को देने हैं तथा कौन-कौन से आउटपुट आपको लेने हैं। किसी प्रोग्राम को लिखने से पहले आपको यह समझ होनी चाहिए कि आप उस प्रोग्राम द्वारा क्या-क्या करवाना चाहते हैं। वास्तव में, कम्प्यूटर का Intellengence Zero है, अत: वह खुद कुछ भी नहीं सोच सकता । जो-जो काम आप प्रोग्राम

द्वारा कराना चाहते हैं उसके instruction आप लिखते जाइये और जब आप यह प्रोग्राम Run करेंगे तो वही काम होते जायेंगे किन्तु जो भी आप कराना चाहते हैं, वह step चाहे कितना ही छोटा क्यों न हो, जब तक आप उसके लिए instruction नहीं लिखेगें, तब तक Computer कुछ नहीं कर पायेगाा । उदाहरणत:, यदि आप चाहते हैं कि आपका प्रोग्राम screen पर कुछ type कराये तो आपको प्रोग्राम में print कराने के लिए instruction लिखना होगा (जो कि C में printf तथा C++ में cout << होता है) तथा इसके आगे proper syntax में वह लिखना होगा जो कि आप type कराना चाहते हैं इसी प्रकार यदि आप चाहते हैं कि आपका program run किये जाने पर कोई इनपुट माँगे तो इसके लिए भी आपको program में instruction देना पड़ेगा के Input मांगो। आप कुछ calculate कराना चाहते हैं, तो उसके लिये भी Instruction लिखना पड़ेगा कि इस formulae से calculate करो । कहने का मतलब यह है कि जो-जो आप program में लिखेंगे, वह सब आपको computer करके दे देगा, न उससे एक भी step ज्यादा करेगा न ही एक भी step कम। तो कोई कार्य कराने के लिए आपको न केवल computer की language आनी चाहिये, बल्कि problem जिसको की आप solve कराना चाहते हैं, उसकी जानकारी होनी चाहिये तथा यह भी आपके दिमाग में clear होना चाहिये कि उस problem को solve कराने हेतु क्या-क्या steps हैं और आप कम्प्यूटर से क्या-क्या कराना चाहते हैं। कम्प्यूटर के प्रोग्राम्स जिन language में लिखे जाते हैं, उन्हें High level languages (HLL) कहते हैं। Fortran, Basic, C, C++, Visual Basic इत्यादि High level languages के उदाहरण हैं। हर Programming Language के अपने अलग instructions होते हैं, तथा जो भी language आप सीखना चाहते है, उससे related सभी instructions आपको सीखने पड़ते हैं, तभी आप एक अच्छे programmer बन सकते हैं अत: program developent process एक step by step process है तथा एक अच्छे effective व efficient program को बनाने में प्रत्येक step का एक महत्वपूर्ण योगदान है। प्रोग्राम डैवलपमैंट प्रौसेस के मुख्य स्टैप्स निम्नवत् हैं—

(i) Crack and Analysing the Problem—किसीं भी प्रोगग्रम को बनाने का सबसे पहला स्टैप यह है कि आप problem को समझें कि आप प्रोग्राम से क्या करवाना चाहते हैं चाहे problem छोटी सी हो या फिर बहुत बड़ी, किन्तु आपको वह problem पूरी तरह से clear होनी चाहिये । आपके दिमाग में problem की पूरी picture स्पष्ट होनी चाहिये कि आप कम्प्यूटर से क्या-क्या steps करवाना चाहते हैं क्योंकि तभी आप उन steps को करवाने के लिये Instructions लिख पायेंगे। देखिये, कम्प्यूटर का खुद का कोई दिमाग नहीं होता, वह तो आपके दिये गये instructions को ज्यो का त्यों करता चला जायेगा। वह तो आपके हुक्म का गुलाम है। अगर आप उसे कहेंगे कुऐं में कूद जाओ तो वह कुऐं में भी कूद जायेगा। अत: यदि आपको ही problem clear नही होगी तो आप कम्प्यूटर से कुछ भी नहीं करवा पायेंगे। अत: सबसे पहला step है, problem को समझना। problem को समझने के बाद उसका analysis किया जाता है कि क्या-क्या इनपुट देने हैं तथा कौन-कौन से आउटपुट प्राप्त करने हैं।

(ii) Writing the steps in Algrithm form and drawing the flowchart. "किसी problem को solve करने के लिये steps को logical sequence में लिखना Algorithm कहलाता है।"

"The logical sequence of precise steps to solve a given problem is known as Algorithm."

"किसी problem को solve करने हेतु विभिन्न steps को चित्र के रूप में दर्शाना flowchart कहलाता है।"

"A flowchart is a diagramatic representation of an algorithm and it describes the sequence of operations to be performed to solve a given problem."

"A flowchart is a visual presentation of data flow of an information processing system, representing the various steps operating performed and the sequence in which they are performed to get the solution of any given problem."

यदि कोई problem का solution निकालना है तथा उसके लिये program लिखना है तो उस problem को understand करने के बाद उसको solve करने हेतु क्रमबद्ध तरीके से जो steps लिखे जाते हें, उसको ही Algorithm कहा जाता है। यदि algorithm को picture या diagram के रूप में व्यक्त किया जाये तो इसको flowchart कहा जाता है।

(iii) Code the Algorithm—अब जबकि आपने problem को समझ लिया है, उसको analyse कर लिया है तथा उस problem को solve करने हतु सभी steps को क्रमबद्ध कर लिया है अर्थात् algorithm लिख ली है तो अब आप तैयार हैं,

किसी कम्प्यूटर भाषा में उस problem हेतु program लिखने को। Algorithm के विभिन्न steps को program में convert कर देना ही coding कहलाता है।

"Algorithm को computer programming language में convert कर देना कोडिंग कहलाता है अर्थात् problem को solve करने हेतु computer को जो instructions देने हैं, उन्हें किसी programming language अर्थात् High level language में program के रूप में लिख देना coding कहलाता है। फिर इस coded program को processing हेतु कम्प्यूटर में फीड किया जाता है।"

"The translation of algorithm to a computer high level language program is called coding. The program obtained is called the source code. The coded program is then fed into the computer for further processing."

(iv) Compiling the Program—प्रोग्राम को कम्प्यूटर में फीड करने के बाद उसकी compiling की जाती है। Compiling मतलब है, High level language में आप द्वारा जो प्रोग्राम लिखा गया है जिसको कि Source code भी कहते हैं उसको Machine language program (जो कि binary form में होता है तथा object code कहलाता है) में convert करना। इसके लिये कुछ विशेष programs होते हैं जिनको compilers कहा जाता है, अर्थात् compilers वह प्रोग्राम होते हैं जो किसी High level program को machine language program में convert करते हैं दूसरे शब्दों में कहा जाये, तो compilers वह program होते हैं जो source code को object code में convert करते हैं। यहाँ पर आपको यह भी समझ में आ जाना चाहिये कि हर language का compiler अलग होता है अर्थात् C का Compilier "C"भाषा के प्रोग्राम को machine language में बदल देगा। किसी दूसरी language के लिये अलग compiler होगा, इत्यादि।

"High level language program को machine language में convert करना compilation कहलाता है।"

"After the program source code is fed into the computer, it is translated into machine language (object code). This process is called compilation. For compilation appropriate compiler program is used, which can translate the program written in a specific language to the object code."

Program को compile करते समय compiler यह भी check करता है कि आपका program programming language के अनुरूप लिखा गया है तथा language के नियमों का पालन कर रहा है या नहीं । यदि programming language के किसी grammatic rule का उल्लंघन होता है (Syntax error) या कोई statement meaningful नहीं होती है (semantic error) तो compile indicate कर देता है कि program में error है। इन errors को compile time error कहते हैं ताकि programmer को इनको ठीक करना जरूरी होता है तथा तभी program execute होता है। errors को ठीक करने की प्रक्रिया debugging कहलाती है। Errors पर विस्तृत चर्चा आगे की जायेगी।

(v) Execute the Program—प्रोग्राम को compile करने के बाद उसको execute किया जाता है। इस phase को run time कहा जाता है, तथा इस दौरान program में लिखे गये instructions का execution होता है तथा आपको output प्राप्त हो जाती है। उपर्युक्त वर्णित सभी steps को ध्यानपूर्वक किया जाना चाहिये तथा चैक करते रहना चाहिये कि किसी भी step में कोई त्रुटि (error) न हो जाये। error होने पर आपको वांछित results प्राप्त नही हो पायेंगे तथा आपको program लिखने का कोई फायदा नहीं होगा। Errors किसी भी stage में होना सम्भव है—algorithm लिखने में, coding में, compiling के समय report किये गये errors इत्यादि। अत: program लिखने हेतु सभी steps में प्रोग्रामर को अत्यंत सावधानी बरतनी चाहिये।

§ 1.10. त्रुटियाँ (Errors) :

जैसा कि आपने पढ़ा, त्रुटियाँ प्रोग्राम डैवलपमैंट के किसी भी stage पर हो सकती हैं। इसलिये programmer को program development के सभी stages बहुत सावधानी पूर्वक करने चाहिये। Errors जिनको कि bug भी कहा जाता है, program के compilation तथा proper execution में बाधा पहुँचाता है। Erors सामान्यत: निम्न प्रकार के होते हैं—(i) Compile time errors (ii) Run time errors (iii) Logical errors

(i) Compile-time Errors—Compile-time में होने वाले errors Compile-time errors कहलाते हैं जब program को compile किया जाता है तो उसका source code check किया जाता है कि वह programing language के rules को follow कर रहा है या नहीं अर्थात किसी rule का उल्लंघन violation तो नहीं कर रहा। Compile time errors दो प्रकार के होते हैं—

(a) Syntax Errors—यदि language के grammatical rules का उल्लंघन (Computer भाषा के व्याकरण का उल्लंघन) होता है तो इसको Syntax errors कहते है अर्थात् किसी भी programming language की हर statement के कुछ rules (नियम) निर्धारित होते हैं तथा उन्हीं rules के अंतर्गत उस statements को लिखा जाना चाहिये। यदि उस rule में जरा भी गलती आपने कर दी तो compiler उस statement को नहीं समझ पायेगा और syntax error बता देगा अर्थात् उस statement को invalid बता देगा अत: programming language को सीखते समय आपको हर statement का proper syntax सीखना होगा, अन्यथा आप proper प्रोग्राम नहीं लिख पायेंगे।

"Syntax refers to the general rules governing the validity of statements of a computer language-Any violation of syntax of the statement results in syntax error."

(b) Semantic Errors—यदि program में कोई ऐसी statement लिखी है, जिसका कोई meaning नहीं निकाला जा सकता, तो इसको semantic error कहा जाता है।

"Semantic means set of rules which give meaning to a statement. If the statement has no meaning, Semantic errors occur."

(ii) Run-time Errors—Run-time errors वह होते हैं जो प्रोग्राम के execution के समय होता है। यह errors program के execution (कार्यान्वयन) को stop कर देते हैं ie. they result in program crash or abnormally terminated program ऐसे program syntax wise तो सही होते हैं किन्तु उसमें गलत data देने या infinite loop में फंस जाने के कारण इनको terminate करना पड़ता है।

Errors that occur during the execution of a program are run-time errors. Some run-time errors stop the execution of the program which is then called program crashed or abnormally terminated.

Most run-time errore are easy to identify because program halts when it encounters an infinite loop or wrong values (of different data type other than required) as input.

Normally, programming languages incorporate checks for run-time errors, However, C++ usually takes care of such errors by terminating the program, but a program that crashes whenever it detects an error condition is not desirable. Therefore, a program should be robust so as to recover and continue following an error.

(iii) Logical Errors—कभी-कभी ऐसा होता हैं कि प्रोग्राम compilation के दौरान तथा run-time में कोई error नहीं show करते किन्तु फिर भी आपको सही आउटपुट results प्राप्त नहीं होता इसका मतलब यह हुआ कि programmer ने problem को समझने में या उसका विश्लेषण करने में कोई गलती की है अर्थात् उसने गलत logic सोचा है। इस गलत logic को लगाकर उसने प्रोग्राम को लिख दिया है। अत: program का syntax तो सही है, तथा वह run भी हो जायेगा, किन्तु प्रोग्राम के गलत logic लगाने देने के कारण सही result नहीं दे पायेगा ऐसे errors logical errors कहलाते हैं।

Sometimes, even if we don't encounter any error during compile-time and run-time, but the program does not provide the correct result. This is because of the programmer's mistaken analysis the problem he is trying to solve. Such errors are logical errors. For instance, an incorrectly complemented algorithm, or use of a variable before its initialization, or unmarked end for a loop, or wrong parameters passed are often the hardest to prevent and to locate. These must be handled usefully, sometimes logical errors are treated as a subcategory of run-time errors.

Program solving methodology and techniques—
The steps to creating a working program are —
(i) Understand the problem well.
(ii) Analyze the problem to—
- *identify minimum number of inputs required for output.*
- *identify processing components.*

(iii) Design the program by—
- *deciding step by step solution.*
- *breaking down (disintigrating) solution into simple steps..*

(iv) code the program by—
- *identifying arithmetic and logical operations required for solutions.*
- *using appropriate control structures such as conditional or looping control structure.*

(v) Test and Debug the program by—
- *Finding errors in it.*
- *Correcting the errors.*

(vi) Complete your documentation.
(vii) Maintain your program.

§ 1.11. प्रोग्राम डाक्यूमैन्ट्स (Program Documentation) :

Documentation का तात्पर्य है कि प्रोग्राम के बीच में आप जगह-जगह पर comments लिख देते हैं, कि कोई statement प्रोग्राम में क्यों लिखी है, या कोई फंक्शन उत्पन्न किया है तो वह क्यों किया है। ठीक ऐसे ही, जैसे maths की problem solve करते time right side में bracket में reason लिखते जाते हैं ताकि दूसरे को समझ में आ जाये, कि यह step कैसे आता है। प्रोग्राम का documentation इसलिये जरूरी है ताकि बाद में यदि कोई दूसरा user उस प्रोग्राम को समझना चाहे तो उसको प्रोग्राम आसानी से समझ में आ जायें। यदि स्वयं programer भी उस प्रोग्राम को बाद में देखेगा, तो उसे भी आसानी से याद आ जायेगा कि जो statements है, वह उसने क्यों लिखी है। कई बार, जब प्रोग्राम लम्बा होता है, तो प्रोग्राम को दोबारा देखने पर documentation से प्रोग्राम को समझने में काफी सुविधा हो जाती है।

Internal documentation द्वारा प्रोग्राम में कुछ statements डाल दी जाती हैं, जो कि compiler execute नहीं करता है, तथा यह statements प्रोग्राम के किसी हिस्से का purpose अर्थात् उद्देश्य समझाती है। यह statements comments कहलाती हैं Documentation द्वारा प्रोग्राम के कार्य का विस्तृत विवरण या प्रोग्राम कैसे use किया जाना है, यह सब भी लिखा जाता है। व्यवसायिक प्रोग्राम (Commercial program) में documentation के अंतर्गत instruction manual भी शामिल होती है।

"Documentation means written descriptions, specification, design code and comment, internal and external to a program which make a program more understandable readable and more easily modifiable."

ध्यान रखें कि यदि आप program में comments डालते हैं तो केवल अपनी या दूसरे programmer की सुविधा हेतु (कि program समझने में आसानी रहे) compiler का comments से कोई लेना-देना नहीं होता तथा compilation करते समय compiler सभी comments को ignore कर देता है अर्थात् comments को compile नहीं किया जाता।

प्रोग्राम मैंटेनेन्स (Program Maintenance)—प्रोग्राम मेंन्टेनेन्स का तात्पर्य है, प्रोग्राम के पूरा हो जाने के बाद उसका Modification जिससे कि बदलती आवश्यकताओं की पूर्ति होती रहे या उसमें आने वाले Errors को ठीक किया जा सके। कई प्रकार के maintenance हो सकते है—

(i) **Corrective Maintenance**—जब प्रोग्राम पूरा हो जाता है तथा उसको Use करके operation कियो जाते हैं, तो कभी-कभी कुछ Unexpected error आ जाते हैं इनको ठीक करना Corrective Maintenance कहलाता है।

(ii) **Adaptive Maintenance**—स्थितियाँ चेंज होने पर प्रोग्राम को नयी स्थिति के अनुसार ढालने हेतु correction करना Adaptive Maintenance कहलाता है जैसे—Government change होने पर कोई कानून चेंज हो जाये, किसी Company की कोई Policy Change हो जाये या market की स्थिति चेंज हो जाये या फिर नये वित्तीय वर्ष में Income Tax Calculation के नये slabs आ जायें इत्यादि।

(iii) **Preventive Maintenance**—अर्थात् यदि किसी error के उत्पन्न होने से पहले ही आपको अंदाज़ा लग गया कि यह error उत्पन्न होगा तो उसको ठीक कर देना Preventive Maintenance कहलाता है।

(iv) **Perfective Maintenance**—तेज़ी से बदलती दुनिया के अनुसार software में नये features, नयी facilities तथा नयी capabilities डालते रहना perfective maintenance कहलाता है।

§ 1.12. Algorithm and Flowcharting :

जैसा कि आप जानते हैं कि किसी प्रोग्राम को solve (हल) करने के लिये steps को logical sequence (तार्किक क्रम) में लिखना Algorithm कहलाता है तथा इन steps को चित्र के रूप में दर्शाना flowcharting या flowchart कहलाता है।

इससे पहले कि हम आगे बढ़ें और C में program लिखना सीखें, Algorithms तथा flowcharts को समझ लेना बहुत जरूरी है। ऐसा इसलिये कि जब भी आप कम्प्यूटर की कोई भी language सीखते हैं तो किताब केवल आपको उसके language से सम्बन्धित विभिन्न statements, उसके प्रारूप जिसे syntax कहा जाता है तथा कुछ Example programs के साथ उन्हें use करना सिखा सकती हैं, किन्तु उसके बाद यदि आपको कोई नया प्रोग्राम लिखने को दिया जाये, तो प्रोग्राम का logic आपको खुद ही सोचना पड़ेगा। logic का मतलब यह कि जो problem आपको दी गई है अर्थात् जो program आपको लिखना है उसको computer से कैसे solve कराना हैं computer को कोन-कौन से instruction देने हैं तथा किस क्रम में देने हैंक्योंकि आप जानते हैं कि program instructions का क्रमबद्ध समूह होता है, जो कि computer execute करता है तथा समस्या solve होती हैं। अत: computer की भाषा में statements को लिखने से पहले अर्थात् program लिखने से पहले उन statements को (अर्थात् program के steps का क्रमबद्ध लिखना) आप अपनी भाषा में लिख लेते हैं तथा इसी को algorithm कहा जाता है।

इसके पहले कि हम कुछ problems की algorithm लिखें, एक छोटा सा example लेते हैं। मान लीजिये कि आपको बस द्वारा लखनऊ से दिल्ली जाना है। इसके लिये आप कौन-कौन से steps करेंगे इसकी algorithm आपको लिखनी है। इसको ऐसे लिखेंगे—

Step 1—लखनऊ के बस अड्डे जायेंगे।
Step 2—दिल्ली जाने वाली बस ढूँढेंगे।
Step 3—बस में जाकर बैठ जायेंगे।
Step 4—कन्डक्टर को रुपये देकर दिल्ली का टिकट लेंगे।
Step 5—दिल्ली आने पर बस से उतर जायेंगे।

तो यह है लखनऊ से दिल्ली जाने का algorithm अर्थात् किसी भी कार्य को करने के लिये जो steps हैं, उसको क्रमबद्ध तरीके से लिख देना ही algorithm कहलाता है। अब कुछ अन्य examples लेते हैं—

Example—Write an algorithm for inputting a number x, calcualting its square and printing the value of square of x. (एक संख्या x को इनपुट कराने, उसका वर्ग calculate करने तथा x^2 की value print कराने हेतु algorithm लिखिये)

Step 1—Input the value of x.
Step 2—calculate x^2
Step 3—Print the value of x.

फ्लोचार्ट्स (Flowcharts)

किसी problem को solve करने के लिये विभिन्न steps को चित्र के रूप में दर्शाना flow chart कहलाता है।

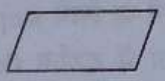
"A pictorial view of algorithm ie the flow of information in a processing system is called a flow chart."

यह तो आप जानते ही हैं कि चित्रों की क्या महत्ता होती है। आप हाथी पर दो पन्नों का निबन्ध लिखकर भी हाथी को उतनी अच्छी तरह से explain नहीं कर पायेंगे जितनी अच्छी तरह आप हाथी का चित्र बनाकर कर लेंगे अर्थात् किसी बात को हज़ारों शब्दों की अपेक्षा एक चित्र द्वारा अत्यधिक स्पष्ट रूप से व्यक्त किया जा सकता है। ऐसे ही, algorithm को यदि flowchart के रूप में दर्शा दिया जाये, तो वह ज्यादा स्पष्ट (clear) तथा समझने में आसान हो जाता है (easy to understand)

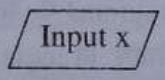
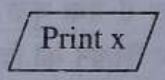
Flowchart हेतु कुछ standard symbols हैं, जो आपको याद करने पड़ेंगे—

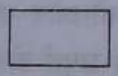
(i) Start तथा stop के लिये 

जैसे  तथा 

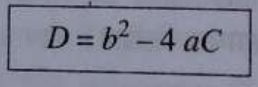
(ii) Input तथा Output के लिये  है,

जैसे

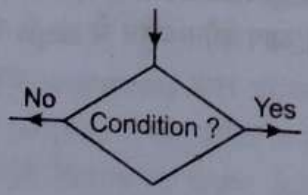
 या 

(iii) Process के लिये 

जैसे—

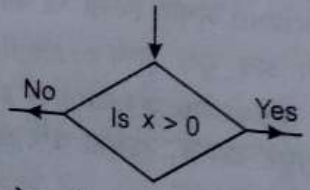


(iv) Decision making के लिये



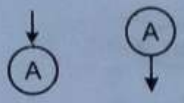
यदि condition true होगी तो program का control yes की ओर चलेगा तथा यदि control false होगी तो program का control no की ओर चलेगा।

जैसे—



अब यदि x का मान शून्य से बड़ा होगा तो program का flow yes की ओर चलेगा, यदि x का मान शून्य से बड़ा नहीं होगा (अर्थात् शून्य या निगेटिव होगा) तो program का flow no की ओर चलेगा।

(v) Connector अर्थात् यदि flow chart को एक page से next page में ले जाना है, या दो भागों में break करना है।



(vi) Comments, Explainations definitions के लिये

Question : What are the advantages of flowcharts?

Ans : Flowchart अर्थात् क्रमदर्शी आरेख या प्रवाह तालिका algorithm का चित्रात्मक प्रदर्शन होता है। इनकी सहायता से प्रोग्रामर समस्या को आसानी से समझ सकता है तथा प्रोग्रामर के लिये प्रोग्राम लिखना आसान हो जाता है।

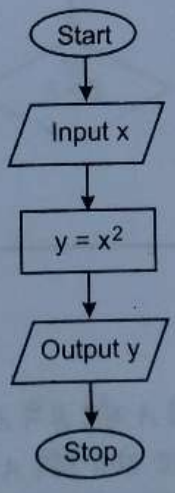
Flowcharts के मुख्य लाभ निम्नवत् हैं—

- Easy Communication अर्थात् information का flow आसानी से समझा जा सकता है।
- Effective Analysis अर्थात् समस्या का बेहतर तरीके से विश्लेषण किया जा सकता है।
- Proper documentation, अर्थात् Program का proper documentation करने में सहायता मिलती है।
- Efficient Coding अर्थात् flowchart से program की बेहतर coding सम्भव हो जाती है।

Proper debugging and maintenance अर्थात् program की debuging व maintenance हेतु flowchart से काफी सहायता मिलती है।

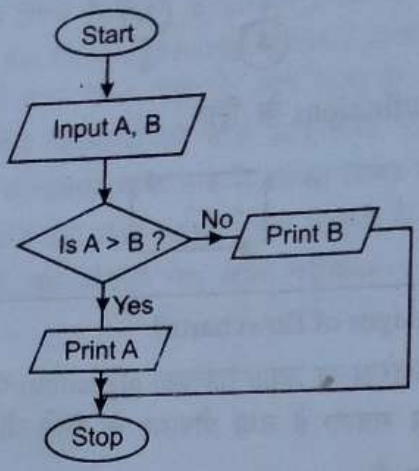
However, the limitations of flowcharts is that they become clumsy in case of complex logic, if some alteration has to be made, we have to readraw them completely, their reproduction is also a problem as they cannot be typed.

उदाहरण 1— Draw a flowchart to input the value of x, calculate and print the value of x^2.



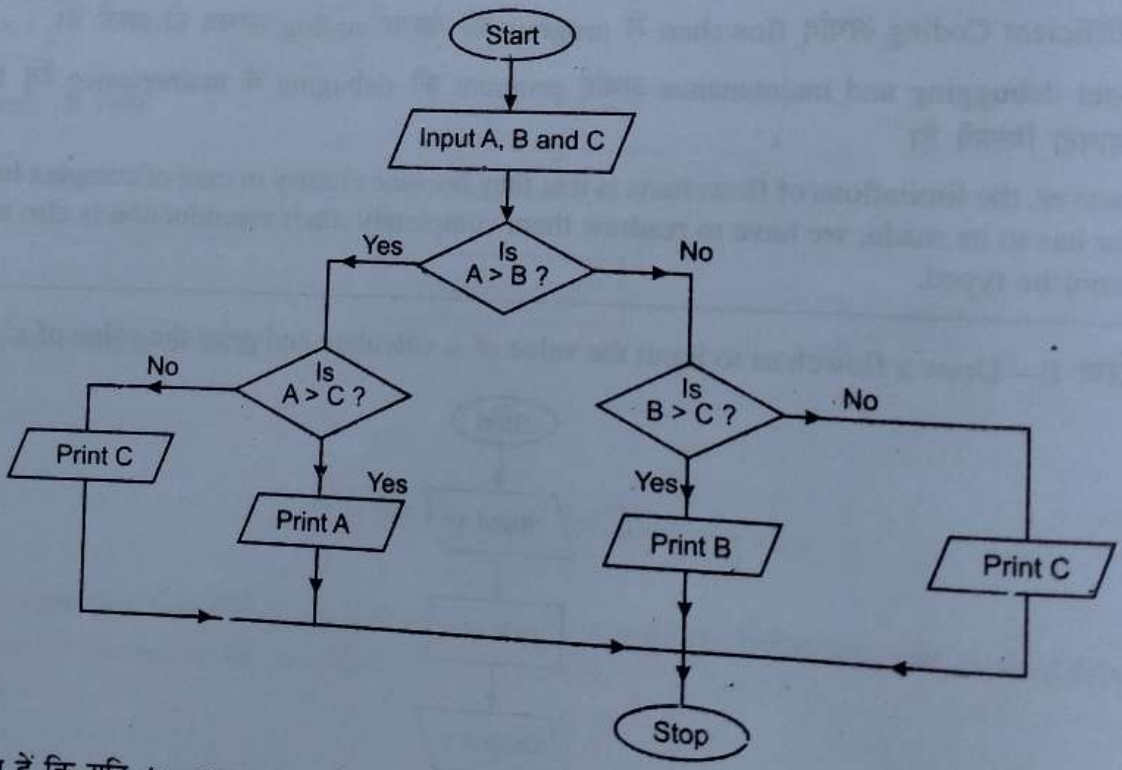
Problems, which involve decision making—सामान्यत:problems में ऐसी स्थिति उत्पन्न हो जाती है, जबकि दो options में से एक का चुनाव करना पड़ता है, अर्थात् यदि option का answer yes है तो कुछ steps करने होते हैं तथा यदि answer no हें तो कुछ अन्य steps होते हैं। अत: अब हम कुछ ऐसे problems की algorithm लिखेंगे व flowchart बनायेंगे, जहाँ decision making involved हो।

उदाहरण—दो संख्याओं को इनपुट कराने तथा उनमें से बड़ी संख्या को Print कराने हेतु Flow chart लिखें।



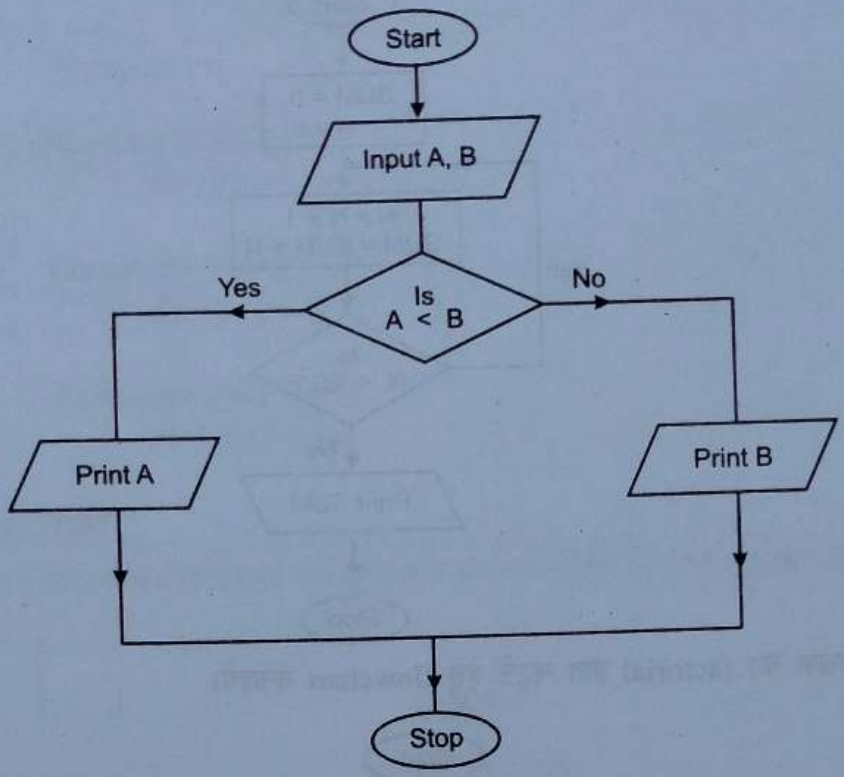
नोट—ध्यान दें यदि $A > B$ तो program का flow yes की ओर जायेगा A print होगा नहीं तो program का flow no की ओर जायेगा तथा B Print होगा।

उदाहरण—तीन संख्याओं को इनपुट कराने तथा उनमें से सबसे बड़ी संख्या प्रिंट कराने हेतु Flowchart बनायें।

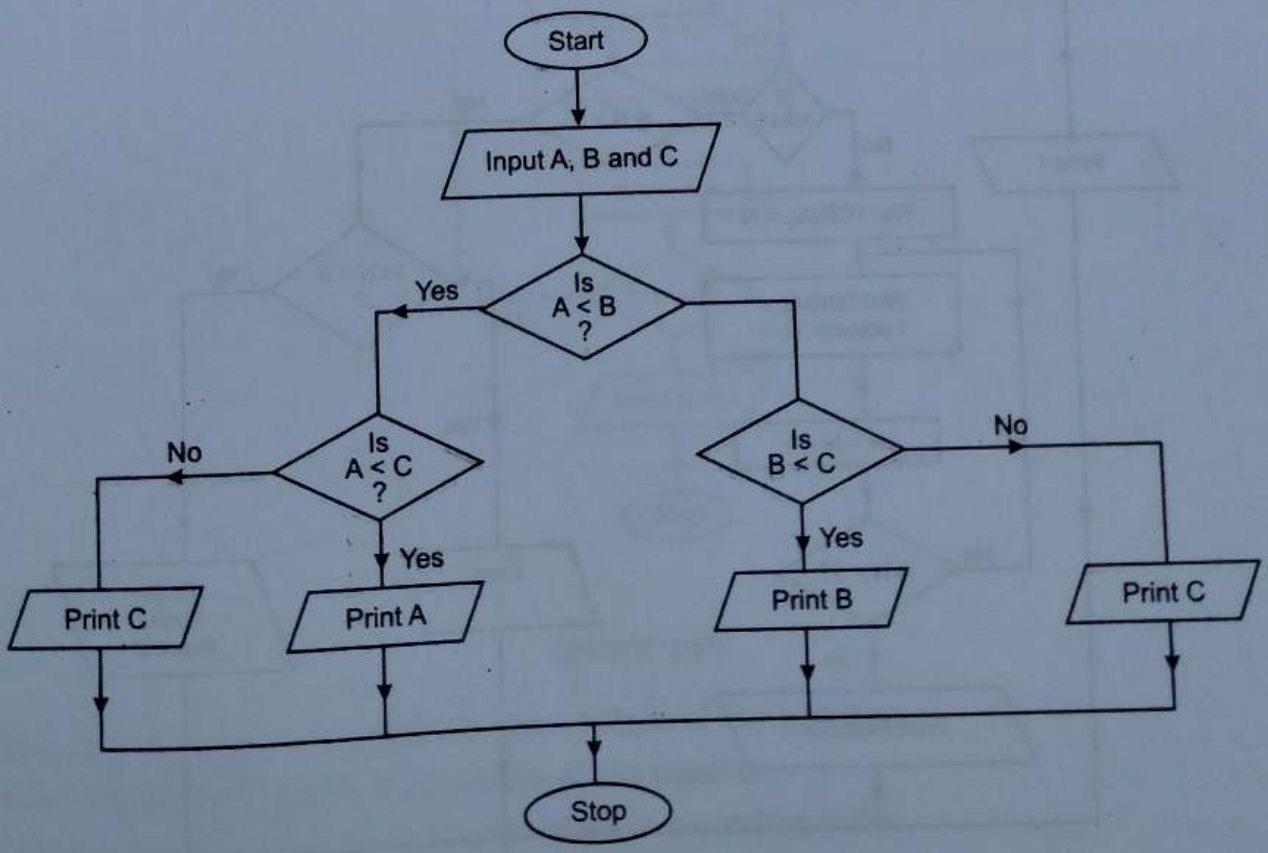


नोट—ध्यान दें कि यदि $A > B$ का उत्तर हाँ है, तो A और B में A बड़ा है, अब यदि $A > C$ का भी उत्तर हाँ है तो A ओर C में भी A बड़ा है, अत: A सबसे बड़ा है, अत: A प्रिंट होगा यदि $A > C$ का उत्तर ना है, तो इसका अर्थ हुआ कि A, C से छोटा है (किन्तु B से बड़ा है), अत: C प्रिंट होगा यदि $A > B$ का उत्तर ना है तो A और B में B बड़ा है, अब यदि $B > C$ का उत्तर हाँ है, तो B और C में भी B बड़ा है, अत: B सबसे बड़ा है तो B का प्रिंट होगा, यदि $A > B$ का उत्तर न है, तथा $B > C$ का उत्तर भी न है, तो C सबसे बड़ा है, अत: प्रिंट C होगा।

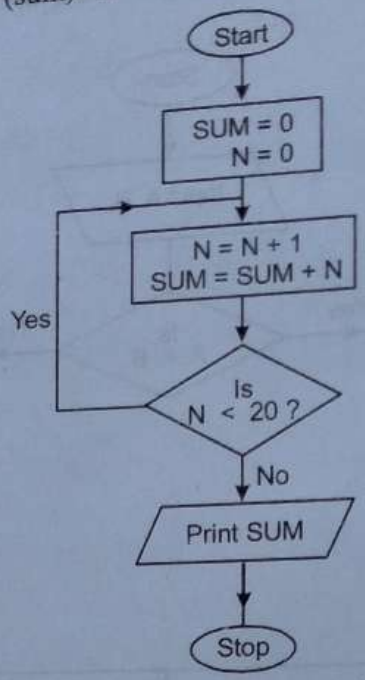
उदाहरण—दो संख्याओं को इनपुट कराने पर उनमें से छोटी संख्या प्रिंट कराने हेतु फ्लो चार्ट बनाइये।



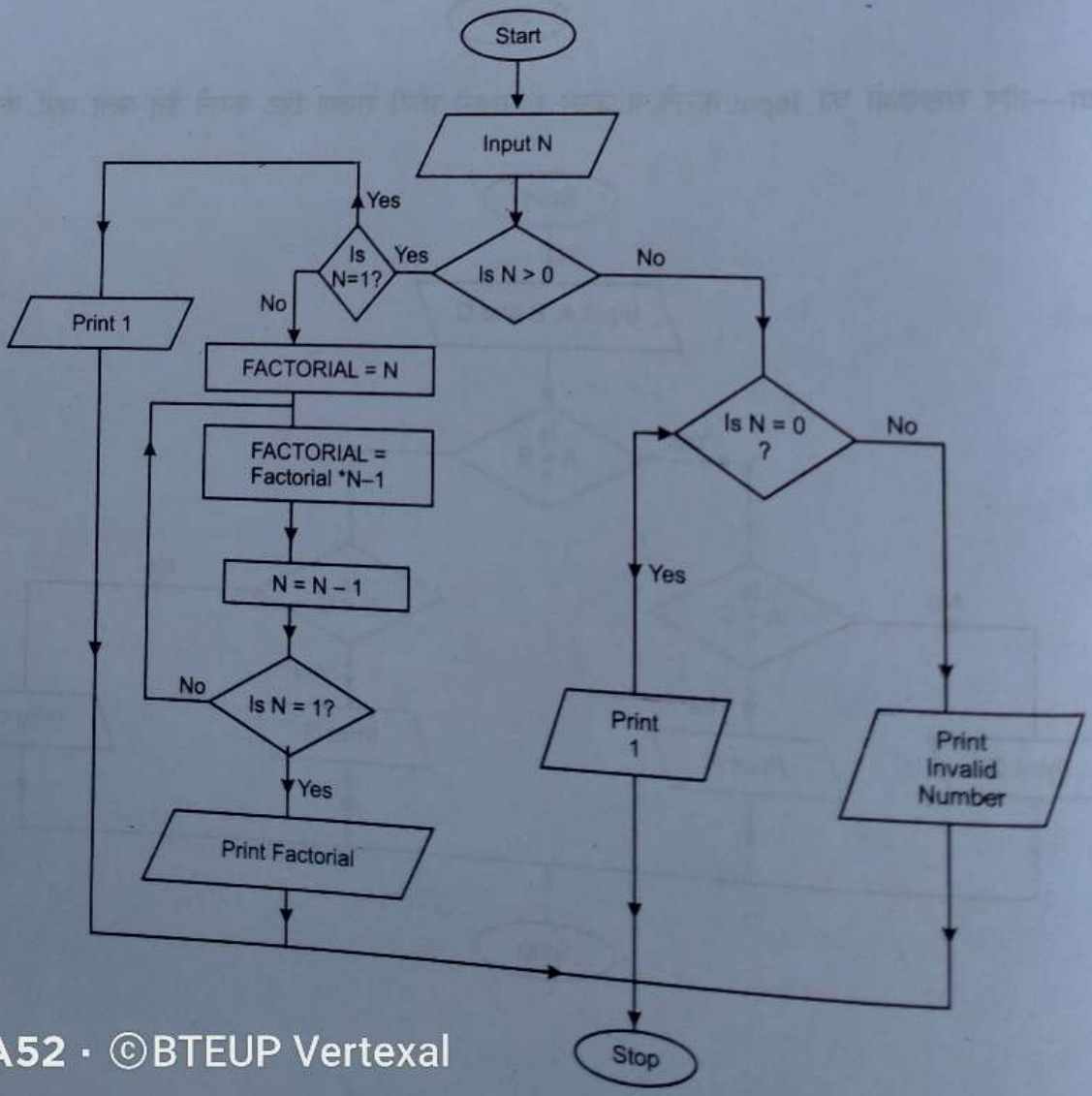
उदाहरण—तीन संख्याओं को Input कराने व उनमें से सबसे छोटी संख्या प्रिंट कराने हेतु फ्लो चार्ट बनाइये।



उदाहरण—पहली 20 संख्याओं के योग (sum) की गणना करने हेतु flowchart बनाइये।



उदाहरण—किसी संख्या का factorial ज्ञात करने हेतु flowchart बनाइये।



माना कि $N = 4$

$N > 0$ is yes

So

Factorial = 4

Next :

Factorial = $4 \times 3 = 12$

$N = N - 1 = 4 - 1 = 3$

So $N = 1$ is false

Factorial = $12 \times 2 = 24$

$N = 3 - 1 = 2$

Still $N = 1$ is false

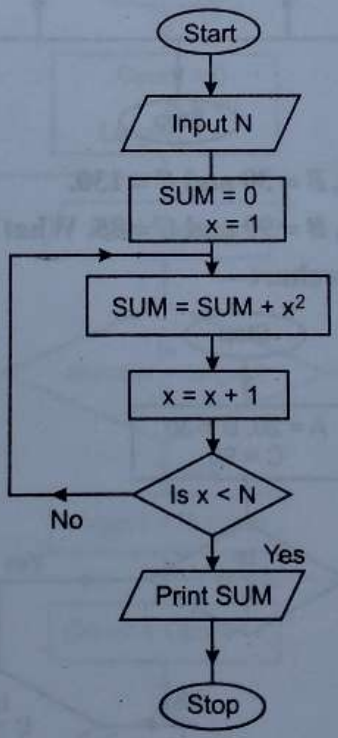
Factorial = $24 \times 1 = 24$

$N = 2 - 1 = 1$

$N = 1$ is true

So factorial = 24 print होगा।

Examples : Draw a flowchart to input a number N and find the sum of squares of numbers upto N i.e., $1^2 + 2^2 + ... + N^2$



अभ्यास प्रश्न

(i) Draw a flowchart to calculate the sum of digits of any number.

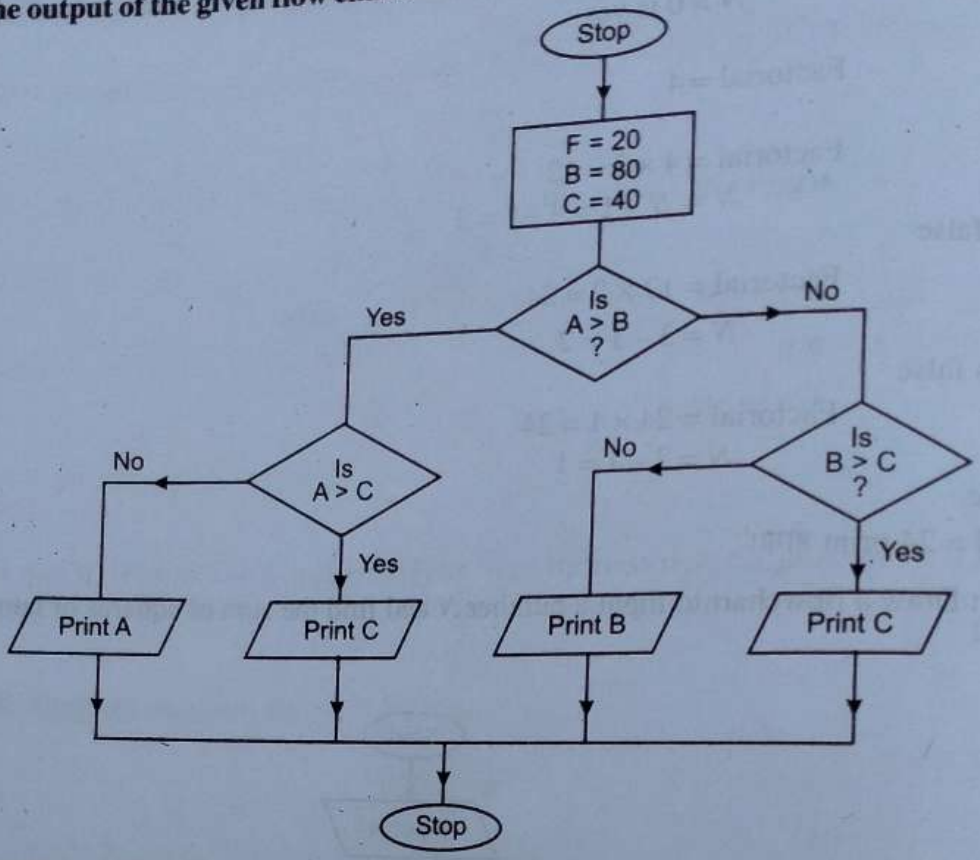
(ii) Draw a flowchart to reverse the digits of any number.

(iii) Draw a flowchart to arrange five numbers in ascending order.

(iv) Draw a flowchart to print the table of 5.

(v) Draw a flowchart to input a number and to print its table.

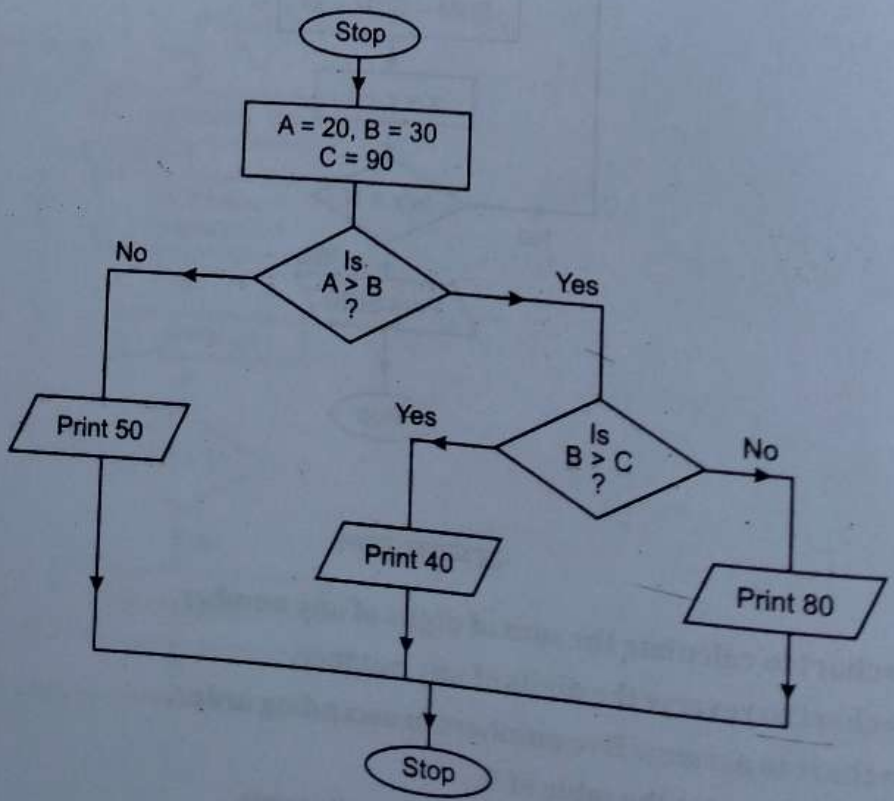
(vi) Find the output of the given flow chart—



(vii) Repeat problem (vi) for $A = 50, B = 30$ and $C = 130$.

(viii) Repeat problem (vi) for $A = 80, B = 90$ and $C = 85$. What do you observe.

(ix) Find the output of the given flowchart—



An algorithm (एल्गोरिद्म) is a step-by-step procedure for calculations. They are used for calculation, data processing and reasoning.

Example : If we want to input 50 numbers and return the largest number, the algorithm with be as follows :

Step 1 : Start

Step 2 : Let $N = 50$, Largest = 0, Count = 0

Step 3 : Input number

Step 4 : If number > Largest, then Largest = Number

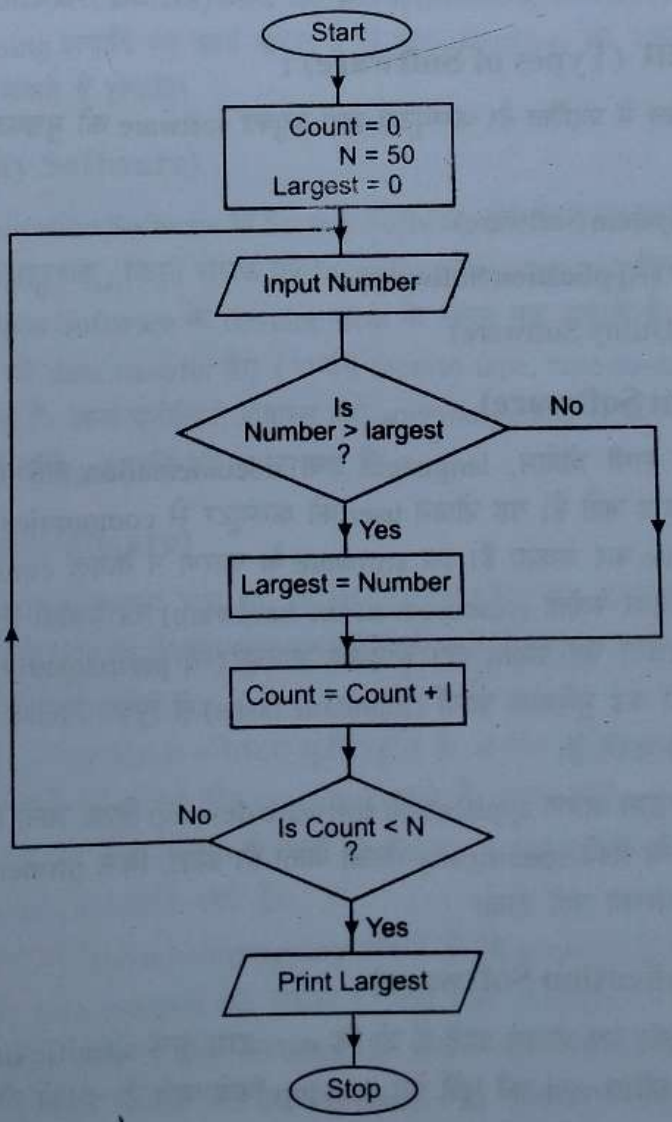
Step 5 : Increment Count

Step 6 : If count < 50, Go to Step 3

Step 7 : Print Largest

Step 8 : Stop.

A flowchart is a digramattic representation of an Algorithm. For example, if you wish to input 50 numbers and find the largest, then the flowchart will be as under :



§ 1.13. हार्डवेयर तथा साफ्टवेयर (Hardware and Software) :

अक्सर छात्रों को इस बात में confusion रहता है कि आखिर हार्डवेयर तथा सॉफ्टवेयर में क्या अन्तर है। वास्तव में, कम्प्यूटर के हार्डवेयर का उसके सॉफ्टवेयर से वही सम्बन्ध है जैसा कि मनुष्य के शरीर का सम्बन्ध उसी जान से है। जिस प्रकार मनुष्य का शरीर उसकी जान के बगैर हाड़-मांस के सिवा कुछ नहीं रह जाता, ठीक इसी प्रकार सॉफ्टवेयर के बिना कम्प्यूटर का हार्डवेयर भी एक लोहे के डिब्बे के समान ही है, जिसको कबाड़ी को बेचने के सिवा उसका और कोई उपयोग नहीं है।

"हार्डवेयर का तात्पर्य कम्प्यूटर सिस्टम के physical components से है जिनको डाटा प्रौसैसिंग हेतु प्रयुक्त किया जाता है।"

"Hardware refers to the physical components of a computer system (electronic or electrical) which are used for processing data."

साफ्टवेयर या प्रोग्राम इंस्ट्रक्शन्स (आदेशों) का एक complete set होता है जो कि प्रोग्रामर द्वारा लिखा जाता है और जिनकी सहायता से कम्प्यूटर किसी problem को solve करता है।

"A software or a program can be defined as complete set of written instructions by the programmer which enables the computer to obtain the solution of a problem (with or without data). Software is a general term that is used to describe any single program or group of programs."

§ 1.14. सॉफ्टवेयर के प्रकार (Types of Software) :

सॉफ्टवेयर का वर्गीकरण चित्र में प्रदर्शित है। कम्प्यूटर्स द्वारा प्रयुक्त software को मुख्यत: तीन श्रेणियों में विभाजित किया जा सकता है—

(i) सिस्टम सॉफ्टवेयर (System Software)

(ii) एप्लीकेशन सॉफ्टवेयर (Application Software)

(iii) यूटीलिटी सॉफ्टवेयर (Utility Software)

सिस्टम सॉफ्टवेयर (System Software)

सिस्टम सॉफ्टवेयर में वह सभी प्रोग्राम, languages तथा documentation होते हैं जो कि कम्प्यूटर के निर्माता (manufacturer) द्वारा सप्लाई किया जाते हैं। यह प्रोग्राम user को कम्प्यूटर से communicate करने में सहायता करते हैं तथा इनके द्वारा अपने प्रोग्राम्स को write कर सकता है। इस software के कारण न केवल computer को use करना आसान हो जाता है, बल्कि कम्प्यूटर के हार्डवेयर स्त्रोतों (resources of the hardware) का प्रभावी एवं दक्षतापूर्वक (efficient) प्रयोग सम्भव हो जाता है। सिस्टम सॉफ्टवेयर वह प्रोग्राम होते हैं जोकि कम्प्यूटर में permanently stored होते हैं जो कि न केवल programmer (अर्थात् user) को कई मुश्किल कार्यों (mundane tasks) से मुक्ति दिलाते हैं बल्कि resource utilization के improvement में भी सहायता करते हैं।

इस सॉफ्टवेयर के बिना user द्वारा अपना application software develop किया जाना सम्भव नहीं है। सिस्टम सॉफ्टवेयर को किसी particular computer के लिये specifically लिखा जाता है। अत:, बिना proper modification के, इसका दूसरे कम्प्यूटर्स पर प्रयोग किया जाना सम्भव नहीं होता।

एप्लीकेशन सॉफ्टवेयर (Application Software)

एप्लीकेशन सॉफ्टवेयर के अंतर्गत वह प्रोग्राम आते हैं जो कि user द्वारा कुछ specific functions को perform करने हेतु प्रयुक्त किये जाते हैं। यह किसी निर्धारित task की पूर्ति हेतु develop किये जाते हैं। इनको तीन श्रेणियों में विभाजित किया जा सकता है—

(i) Specific or Customised Program अर्थात् वह प्रोग्राम जो कि किसी विशेष कार्य हेतु लिखे जाते हैं उदाहरणत: आपने अपनी संस्था की marksheet बनाने हेतु कोई प्रोग्राम लिखा है, या Indian Railways का reservation हेतु लिखा गया प्रोग्राम या किसी Businessman की inventory control हेतु लिखा गया प्रोग्राम इत्यादि। कहने का तात्पर्य यह है कि इन प्रोग्राम को लिखकर user वांछित कार्य को पूर्ण कर सकता है।

(ii) Standard Application Software अर्थात् वह प्रोग्राम को जो कि ready to use होते हैं तथा यह लोगों की common problems को serve करने हेतु लिखे जाते हैं। यह standard software होते हैं तथा इनके users का दायरा बड़ा होता है। उदाहरणत: pay roll packages, accounting packages, traffic control, Educational softwares, Library Management Softwares इत्यादि।

यदि आप का उदाहरण लें तो स्पष्ट है कि यह Software अनेकों Libraries में catalogue बनाने, Books issue का data maintain करने हेतु लाभप्रद होगा। अत:, कोई भी institute अपनी Library हेतु इसे बाजार में खरीद सकता है। अत:, स्पष्ट है कि जहाँ specific या customised software किसी विशेष user को लाभान्वित करता है, जबकि Standard Application Software से अनेकों user लाभान्वित होते हैं।

(iii) General Application Software or Programs का तात्पर्य उन प्रोग्राम्स से है जो कि मार्केट में manuals के साथ उपलब्ध रहते हैं। इन softwares के users का दायरा बहुत अधिक बड़ा होता है। उदाहरणत: MS word, MS Exel, Computer Games Software, DMBS (Database Management System) packages जैसे foxpro इत्यादि। यदि आपको work processing इत्यादि का कार्य करना है तो MS Word use कर सकते हैं, spread sheet use करनी है तो Excel use कर सकते हैं इत्यादि।

यूटीलिटी सॉफ्टवेयर (Utility Software)

यूटीलिटी सॉफ्टवेयर वह Application Software या System Software होते हैं जो कि किसी program के developement में अक्सर प्रयुक्त किये जाते हैं। उदाहरणत:, किसी संख्या की logarithm या square root इत्यादि की गणना करने वाले प्रोग्राम की आवश्यकता किसी Application Software के development के समय पड़ सकती है। ऐसा प्रोग्राम Utility Program कहलाता है। Utility Software को data transfer हेतु (अर्थात् tape-to-tape, tape-to-disk, card-to-tape या tape-to-printer etc.) हेतु किया जा सकता है। अन्य यूटीलिटी प्रोग्राम्स जैसे sort/merge प्रोग्राम्स की सहायता से records की sorting, files की merging, files की updating इत्यादि की जा सकती है।

सिस्टम सॉफ्टवेयर (System Software)

जैसा कि आपको पिछले खण्ड में बताया गया था कि सिस्टम सॉफ्टवेयर के अंतर्गत वह सभी प्रोग्राम, language व documentation आते हैं जो कि कम्प्यूटर के निर्माता द्वारा सप्लाई किये जाते हैं। सिस्टम सॉफ्टवेयर दो प्रकार के होते हैं अर्थात् programming languages तथा System Utilities.

सिस्टम यूटिलिटीज़ (System Utilities)—सिस्टम यूटीलिटीज़ के अंतर्गत अनेकों ऐसे General Purpose Programs आते हैं जो कि कम्प्यूटर के प्रयोग को सरल एवं तीव्र (speedy) बनाते हैं। इनके द्वारा programming efficiency बढ़ती है तथा man-machine communication सुविधाजनक हो जाता है। Editor, Loader, Monitor, Debugger तथा Operating System यह सभी System Utilities के अंतर्गत आते हैं।

एडीटर (Editor)—Editors ऐसे interactive programs होते हैं जो कि memory में stored होते हैं तथा जिनकी सहायता से user program लिख सकता है, text generate कर सकता है, तथा इनमें changes कर सकता है। उदाहरणत:, यदि लिखे गये source program में corrections करने हैं (जो कि compiler द्वारा निकाले गये हैं) तो editor द्वारा यह changes किया जा सकता है। प्रोग्राम के compile होने के पश्चात् तथा उसके error free हो जाने के पश्चात् editor को final text को RAM में store करने हेतु भी कहा जा सकता है।

लोडर (Loader)—प्रोग्राम को execute करने से पूर्व उसको main storage में रखा जाना चाहिये। वह स्पेशल प्रोग्राम जो कि प्रोग्राम को input या storage devices से read करते हैं तथा उनको main storage में place करते हैं, Loaders कहलाते हैं। आधुनिक कम्प्यूटर्स में loaders ROM में permanently stored होते हैं।

मॉनीटर (Monitor)—मॉनीटर भी एक प्रोग्राम होता है जो कि ROM में stored होता है तथा निम्न कार्य करता है—

(i) Instruction तथा data को Enter (या load) करना
(ii) मैमोरी लोकेशन्स तथा रजिस्टर्स की Contents को display करना तथा change करना
(iii) प्रोग्राम को execute करना
(iv) युक्तियों के कंट्रोल को manage करना
(v) मैमोरी एलोकेशन को manage करना

डिबगर (Debugger)—प्रोग्राम में हुई mistake तथा errors को बग (bug) कहा जाता है तथा इन mistake को remove करने की प्रक्रिया debugging कहलाती है। Debugging प्रक्रिया में सहायता करने वाले प्रोग्राम Debuggers कहलाते हैं।

ऑपरेटिंग सिस्टम (Operating System)—ऑपरेटिंग सिस्टम System Software का एक अत्यंत महत्वपूर्ण भाग है। यह प्रोग्राम्स का एकीकृत समूह (Integrated Collection of Programs) है जो कि कम्प्यूटर को on करने के तुरन्त बाद कम्प्यूटर के ऑपरेशन (operation) को अपने हाथों में ले लेता है तथा बिना किसी human intervention के अनेकों प्रोग्राम्स कम्प्यूटर पर run कर जाते हैं। ऑपरेटिंग सिस्टम कम्प्यूटर सिस्टम के brain की तरह कार्य करता है तथा उसकी सभी components को control करता है।

§ 1.15. ऑपरेटिंग सिस्टम क्या है? (What is an Operating System) :

ऑपरेटिंग सिस्टम एक ऐसा प्रोग्राम होता है जो user, तथा computer hardware के बीच एक मध्यस्थ (mediator) की भूमिका निभाता है। यह प्रोग्राम एक ऐसा वातावरण प्रदान करता है जिसमें कि user अपने प्रोग्राम्स को execute कर सके। एक ऑपरेटिंग सिस्टम कम्प्यूटर को user के लिए आसान (user friendly) बनाता है तथा कम्प्यूटर हार्डवेयर के बेहतर प्रयोग हेतु वातावरण तैयार करता है।

"ऑपरेटिंग सिस्टम वह प्रोग्राम या सॉफ्टवेयर होता है जो यूज़र तथा हार्डवेयर के मध्य इंटरफेस या link का कार्य करता है। अत:, ऑपरेटिंग सिस्टम एक प्रोग्राम का होता है, जो कम्प्यूटर को कन्ट्रोल करता है।"

"An Operating System is a program that acts as an intermediatly (मध्यस्थ) between the user of the computer and the hardware of the computer. It provides an environment in which a user can execute the programs in efficiently and conviniently."

ऑपरेटिंग सिस्टम के मुख्य उद्देश्य (Main objectives of an Operating System)

(i) **Abstraction उत्पन्न करके हार्डवेयर की details को hide करना (To hide the details of hardware by creating abstraction)**—Abstraction ऐसा software होता है जो कि lower level details को hide करता है तथा higher level functions का set provide करता है। Operating System physical devices, instruction तथा memory को एक प्रकार के virtual world में transform कर देता है।

(ii) **Resources को मैनेज करना (Manage resources)**—ऑपरेटिंग सिस्टम कम्प्यूटर के विभिन्न resources को manage करता है तथा process को allocate करता है।

(iii) **एक effective तथा pleasant यूसर इंटरफेस प्रदान करना (Provide Pleasant and Effective user interface)**—ऑपरेटिंग सिस्टम यूसर के लिये एक friendly वातावरण तैयार करता है।

यूसर इंटरफेस के महत्वपूर्ण components हैं—Command interpreter, file system, on-line help तथा application integration.

विभिन्न प्रकार के ऑपरेटिंग सिस्टम (Different Types of Operating System)

आधुनिक कम्प्यूटर सिस्टम तीन ग्रुप्स में विभक्त किये जा सकते हैं—

बैच सिस्टम, टाइम शेयर्ड सिस्टम तथा रियल टाइम ऑपरेटिंग सिस्टम इनकों classify करने का आधार यह है कि कम्प्यूटर के user तथा उसके प्रोग्राम का किस प्रकार से प्रौसेसिंग के समय interaction होता है।

बैच प्रोसैसिंग ऑपरेटिंग सिस्टम में user एक central place पर अपने jobs को submit करता है जहाँ jobs को एक batch के रूप में collect किया जाता है। इनको कम्प्यूटर पर एक पंक्ति (queue) के रूप में रखा जाता है तथा एक एक करने process किया जाता है। अत:, processing के समय user का job से कोई interaction नहीं होता।

टाइम शेयरिंग सिस्टम में ऑपरेटिंग सिस्टम कम्प्यूटर एक समय में कई users को services प्रदान करता है। इस प्रकार कई users एक साथ central processor, memory तथा कम्प्यूटर के दूसरे resources को शेयर करते हैं। इस प्रकार के सिस्टम में user को प्रोग्राम से execution के समय full interaction रहता है।

रियल टाइम सिस्टम्स द्वारा उन applications को serve किया जाता है जहाँ कम्प्यूटर से immediate response की आवश्यकता होती है, अर्थात response time अत्यंत कम रखना होता है। Airline reservation systems, machine tool control, Nuclear Power Station की monitoring, getting a stock market quotation इत्यादि में real time system use किये जाते हैं।

§ 1.16. Operating System Terminology :

Multiprogramming—Multiprogramming is the allocation of more than one concurrent applications, jobs or user to a Computer System or its resources अर्थात मल्टी प्रोग्रामिंग द्वारा किसी कम्प्यूटर तथा उसके एक साथ कई कार्य सम्पन्न कर सकते हैं। एक साथ बहुत सारे कार्य सम्पन्न करने के लिये बारी बारी से प्रत्येक कार्य को समय दिया जाता है तथा फिर तेजी से प्रथम कार्य को समय दिया जाता है।

Time Sharing—Computer के resources को एक साथ कई users share करते हैं, उसे time sharing कहा जाता है। प्रत्येक user को CPU का समय दिया जाता है, तथा बारी बारी से CPU सभी users को थोड़ा-थोड़ा समय देता है।

Real Time Systems—इनमें प्रत्येक जॉब के लिये एक निर्धारित समय होता है तथा उस निर्धारित समय (deadline) पश्चात् उस जॉब को अनिवार्य रूप से समाप्त करना होता है तथा system की performance को इसी आधार पर मापा जाता है कि सिस्टम उस निर्धारित समय अवधि में कार्य का पूरा कर पाता है या नहीं। यदि job निर्धारित समयावधि में पूर्ण नहीं होता तो यह स्थिति deadline overrun कहलाती है। एक अच्छा व दक्ष (efficient) सिस्टम वह होता है जो न्यूनतम deadline overrun देता है।

Multitasking—एक साथ बहुत से कार्य करना Multitaslking कहलाता है। Multi processing में जहाँ कई CPU involve हो सकते हैं, वहीं Multitasking में एक ही CPU होता है, जो कि multiple task करने हेतु तेजी से एक task से दूसरे की ओर स्विच करता रहता है। Multitasking दो प्रकार की होती है, pre emptive जिसमें CPU सभी tasks को time slices allot कर देता है तथा cooperative (अर्थात् non-preemptive), जिसमें program स्वयं अपने लिये आवश्यक समय हेतु CPU को कन्ट्रोल कर सकता है।

File Systems—A file system is a method of storing data. A file may also be defined as collection of logically related information कम्प्यूटर के डाटा को स्टोर करने तथा पठनीय बनाने हेतु files का use किया जाता है।

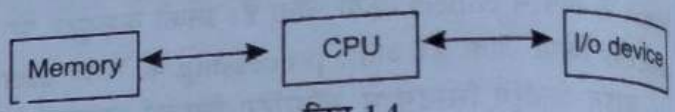
File access methods—फाइल से कोई record या डाटा प्राप्त करना (अर्थात फाइल तक अपनी पहुँच बनाना) file accessing कहलाता है। File access करने हेतु मुख्य विधियाँ निम्नवत् हैं—

(i) Sequential access methods—Sequential access विधि में फाइल को sequence में पढ़कर (record by record) access किया जाता है। यह एक सस्ती, आसान व efficient विधि है किन्तु निम्न activity ratio applications हेतु अनुपयुक्त है।

(ii) Direct access method—Direct access method में record को सीधे ढूँढा जा सकता है तथा एक एक करके सभी records खंगालने की आवश्यकता नहीं होती। अत:, यह विधि तीव्र परिणाम देती है।

(iii) Index Sequential access method—Index Sequential access विधि की सहायता से Indexed sequential files से डाटा access किये जा सकते हैं। Index Sequential file में records की कई स्तरीय Index (multi level indexes) होती है, master index, track indexes इत्यादि जिनकी सहायता से record तक पहुँचा जा सकता है। इस प्रकार record तक directly पहुँचना सम्भव हो जाता है।

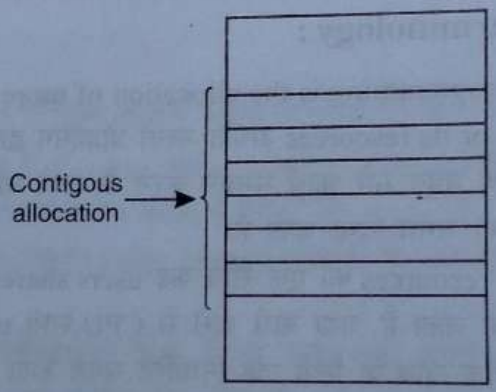
Allocation methods—आधुनिक कम्प्यूटर्स में मैमोरी की एक अत्यंत महत्वपूर्ण भूमिका है। CPU तथा I/O System दोनों का ही मैमोरी से Interaction होता है। (चित्र 1.4)



चित्र 1.4

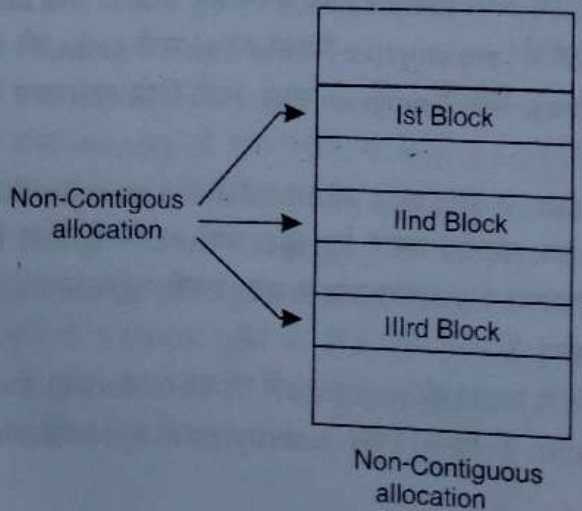
मैमोरी बाइनरी शब्दों (1 तथा 0 के रूप में) का समूह होती है तथा प्रत्येक location का एक unique binary address होता है। CPU memory से read या write करता है। जब program start होता है तो CPU memory allocated (आवंटित) करता है तथा प्रोग्राम समाप्त होने पर मैमोरी फ्री कर दी जाती है। मैमोरी के आवंटन हेतु कई विधियाँ हैं—

(i) Contiguous Storage Allocation—यदि program को contiguous storage memory allot की जाती है तो इसे contiguous storage allocation कहते हैं। *यह बिल्कुल इस तरह है जैसे कि किसी होटल में ठहरने हेतु आये दस व्यक्तियों को मैनेजर क्रम से (उदाहरणत: कमरा नं. 31 से 40) कमरे आवंटित कर देता है।*

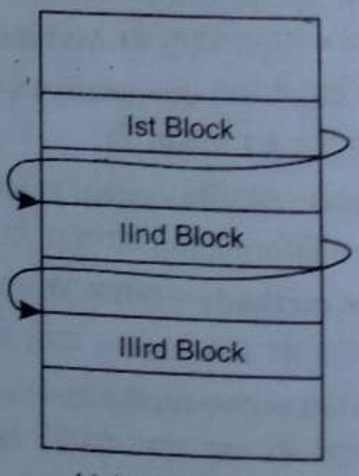


चित्र 1.5 (a)—Contigous allocation

(ii) Non Contiguous Allocation—यदि program को contiguous allocation नहीं किया जाता, तो यह non-contiguous allocation कहा जाता है। *यह बिल्कुल ऐसा ही है जैसे कि होटल का मैनेजर दस व्यक्तियों को क्रमिक रूप से कमरे आवंटित न करके यहाँ वहाँ (उदाहरणत: कमरा नम्बर 35, 48, 82 इत्यादि)। कमरे आवंटित कर दे। इसके लिये program को smaller components (छोटे size के) में ब्रेक किया जाता है।*



चित्र 1.5 (b)

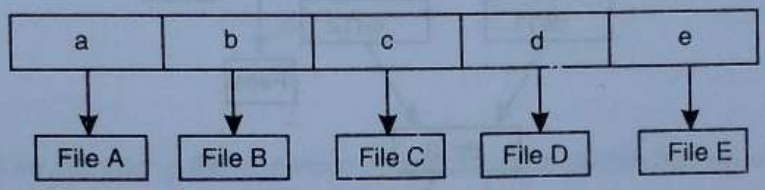


चित्र 1.5 (c)

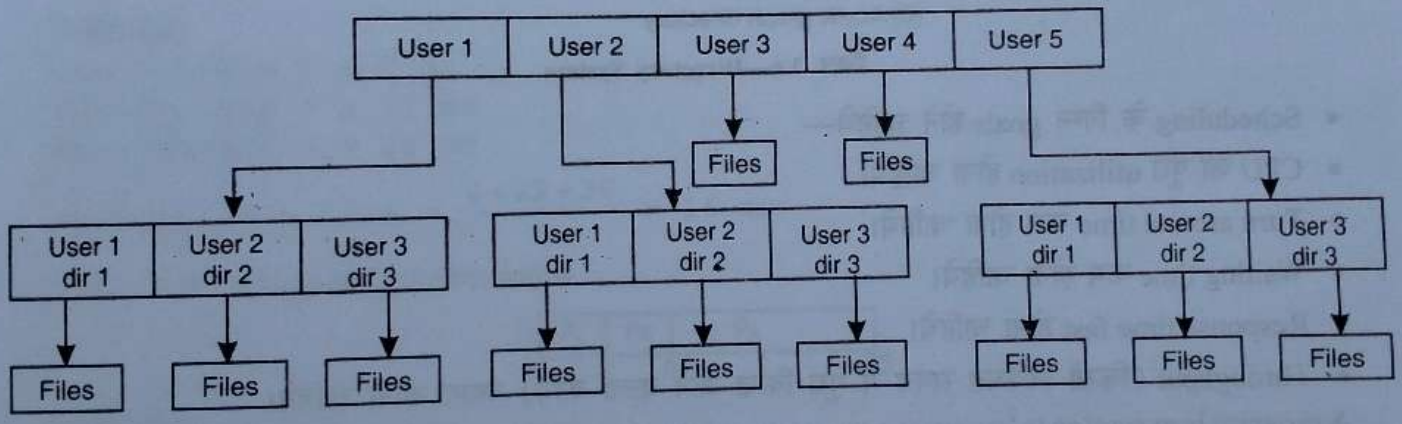
Linked allocation method—Linked allocation एक non contiguous allocation विधि है जिसमें प्रत्येक block उससे अगले block की ओर point करता है अर्थात प्रत्येक block के अंत में उससे अगले block का address होता है। *यह बिल्कुल ऐसे ही है जैसे कि होटल का मैनेजर पहले व्यक्ति को कमरा नम्र 35 आवंटित करे, कमरा नम्बर 35 में एक स्लिप रखी हो जिसमें उससे अगले व्यक्ति का कमरा नम्बर लिखा हो (माना कमरा नम्बर 48), कमरा नम्बर 48 में तीसरे व्यक्ति का कमरा नम्बर लिखा है (माना कमरा नम्बर 82) इत्यादि।)*

Indexed Allocation—इस विधि में allocation हेतु एक Index table बनी होती है (ठीक वैसे ही जैसे आपकी पुस्तक के शुरुआत में एक Index होती है, जो यह बताती है कि कौन-सा अध्याय किस पेज से शुरू है)। अत:, सभी pointer इधर उधर बिखरे होने के बजाये (linked allocation में) एक ही स्थान पर आ जाते हैं तथा blocks को direct access किया जा सकता है।

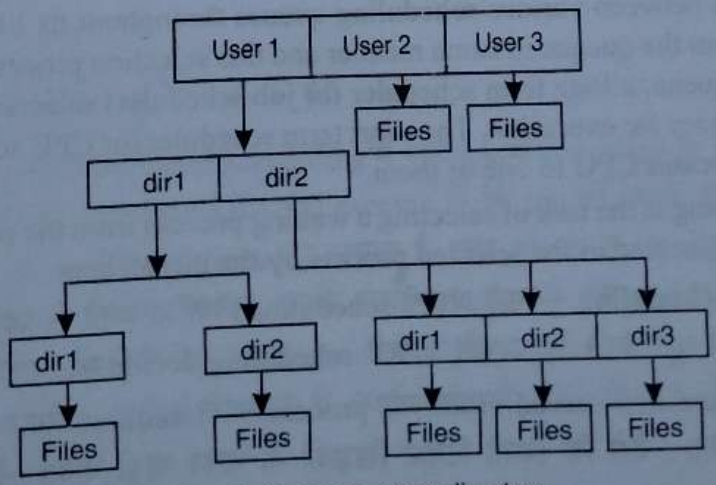
Directory System—कम्प्यूटर्स में फाइल्स को store करने हेतु directory system का use किया जाता है। किसी directories के अंदर sub-directories व उनके अंदर files होती हैं।



(a) Single level directory



(b) Two level directory

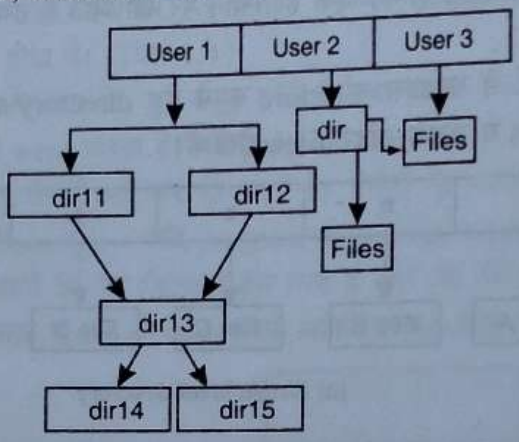


(c) Tree structure directory

चित्र 1.6

- Single level directory में सभी files एक ही directory में होती हैं।
- Two level directory में प्रत्येक user की अपनी user file directory (UFD) होती है।
- Tree structure directory में directory की कई sub-directories होती हैं।
- Acyclic graph direciory में graphs की कोई cycle नहीं होती इसमें directories, sub-directories व files को share कर सकती हैं।

Scheduling Concepts—Processor को विभिन्न कार्य allot करना तथा efficiency, good response time, good turn around time, good throughpu, fairness के साथ यह कार्य सम्पन्न कराना Scheduling कहलाता है।



(d) Acylic graph directory

चित्र 1.6—Directory System

- Scheduling के निम्न goals होने चाहिये—
- CPU का पूरा utilization होना चाहिये।
- Turn around time कम होना चाहिये।
- Waiting time कम होना चाहिये।
- Response time fast होना चाहिये।
- Throughput (किसी निश्चित समय में पूरा किया जाने वाला कार्य) ज्यादा होना चाहिये।

A program is execution is known as process. (or a job). As proccesses enter a system, they are put in a job queue.

A process migrates between various scheduling queues throughout its lifetime. The operating system selects the processes from the queque in some manner and this selection process is called scheduling. Out of many processes in the queue, a long term scheduler (or job scheduler) selects processes from this pool and loads them into the memory for execution. The short term scheduler (or CPU scheduler) selects among these ready processes and allocates CPU to one of them.

Hence CPU scheduling is the task of selecting a waiting process from the ready queue and allocation the CPU to it. The CPU is allocated to the selected process by the dispatchers.

Non Pre-emptive Scheduling—इसमें अगली scheduling निर्धारित करने से पहले पिछला कार्य पूरा किया जाता है अर्थात एक scheduled कार्य पूरा करने के पश्चात् अगली scheduling decide की जाती है।

- **FCFS**—First come first served अर्थात जो process पहले request लेकर आया, उसकी को पहले CPU allocate किया जायेगा (जैसे कि किसी टिकट खिड़की के बाहर खड़ी पंक्ति (queue) में होता है)।
- **Shortest Job First (SJF)**—अर्थात जितने jobs हैं उसमें से जिसका exeution समय सबसे कम है, उसे पहले मौका दिया जायेगा।

- **Deadline Scheduling**—जिसकी deadline (अर्थात पूरा करने हेतु निर्धारित समय सीमा) सबसे पहले है वह कार्य किया जायेगा। यदि job अपनी deadline को overshoot कर जाता है तो यह deadline over run कहलाता है—

K = C - D
K = deadline overrun
C = completion time
D = Deadline.

यह ठीक ऐसा ही है जैसे कि आप उस कार्य को सबसे पहले करें, जिसकी last date सबसे निकट हो।

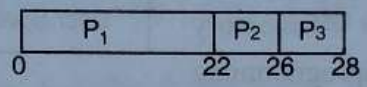
Note that non-pre emptive scheduling takes places under following circumstance.

- When a process switches from running state to waiting state.
- When a process terminates

Lets take an examples—

Process	Burst time (in ms)
P_1	22
P_2	4
P_3	2

FCFS—The Gantt chart (showing the sequence of operation) is shown below—

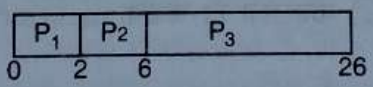


Wait time

For Process 1 = 0 ms
For Process 2 = 22 ms
For Process 3 = 26 ms

$$\text{Average wait time} = \frac{0+22+26}{2} = 16 \text{ ms}$$

SJF—The Gantt chart is shown below :



Wait time

For P_1 = 0
For P_2 = 2 ms
For P_3 = 6 ms

$$\text{Average wait time} = \frac{0+2+6}{3} = 2 \cdot 66 \text{ ms}$$

Pre emptive Scheduling—Pre emptive Scheduling में एक job के execution के दौरान ही scheduling निर्णय ले लिया जाता है। अत:, इस प्रकार की scheduling में CPU execute हो रहे job को किसी अन्य job की processing हेतु कुछ समय के लिये CPU को release करने हेतु force कर सकता है तथा इस प्रकार throughput के सुधार होता है—

- **Round Rob Scheduling**—इसमें एक एक करके सभी process को एक time span या time slot या time quantum (time slice) दिया जाता है। इस प्रकार queue के सभी programs को cyclic रूप से अपने समय प्राप्त होता रहता है। जिसका work complete हो जाता है, वह queue से बाहर हो जाता है।
- **Response ratio scheduling**—

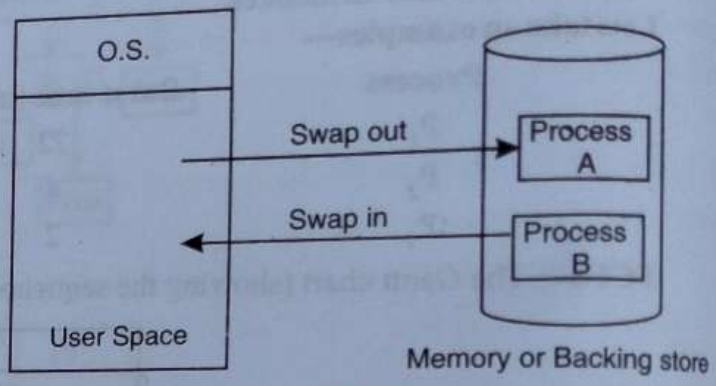
$$\text{Response Ratio} = \frac{\text{Elapsed time}}{\text{Execution time required}}$$

जिसकी response ratio अधिक होती है, उसे प्राथमिकता दी जाती है। अत: short job की response ratio अधिक होने के कारण उसे पहले मौका मिलता है।

- **Shortest seek time first**—जिसका seek time सबसे कम होता है, उसे मौका मिलता है।
- Note that a pre-emptive scheduling takes place under following circumstances :
  When a process switches from running state to ready state.
  When a process switches over from waiting state to ready state.

मैमोरी management—मैमोरी मैनेजमैंट का उद्देश्य है process को memory allocate करना, address space की shapping करना तथा access टाइम को कम करना disk से memory blocks लेकर उनको CPU में input करने हेतु कई तरीके हैं, जैसे—

Swapping—किसी suspended या pre-empted process को main memory से disk में डाल देना तथा आवश्यकता पड़ने पर उसे वापस ले आना Swapping कहलाता है। मान लीजिये आपकी tables पर कुछ ऐसी books हैं जिनकी अभी आवश्यकता नहीं है। अत: फिलहाल आपने उनका बंडल बनाकर अलमारी में रख दिया तथा जब फिर उनकी जरूरत पड़ी तो फिर table पर ले आये, यही swapping है। यदि memory की capacity सीमित है, तो swapping से multiprogramming implement करना सुविधाजनक हो जाता है।



चित्र 1.7

Multiple Partitions—जिस प्रकार किसी अलमारी में विभिन्न racks होते हैं उसी प्रकार memory की भी partitioning की जाती है। यदि partition program के execution से पहले की जाती है तो static कहलाती है तथा यदि programming के दौरान की जाती है, तो dynamic कहलाती है।

Paging—जिस प्रकार पुस्तक को pages में विभाजित करते हैं, ऐसे ही program को भी fixed size program components (छोटे छोटे घटकों में) विभाजित करते हैं, जिनको pages कहते हैं। Pages का address (0 ..... *n*) virtual address space (या logical address space) कहलाता है। जबकि main storage address को physical address space कहा जाता है।

"In Paging, the computer system splits the program into program segments called pages."

Virtual Storage Using Paging

In paging, each program is split into fixed size program components (घटक) called pages. Pages in a program, and words in a page, both are numbered from o....n, i.e., their logical address. the range o....n of logical addresses is called virtual address space or logical address space. The range of main storage addresses o....n is called physical address space. Thus virtual address of a program consists of p pages numbered as o....p-1 and the pages contains O to S-1 words, where S is the page size. The physical or main memory is broken into fixed-size blocks called 'frames'. For a program execution, its pages are loaded into any available frames and the page map table (PMT) which is defined to translate from user pages to memoryframes. The page map table stores an entry for every program page which contains the frame (storage block) number where the page presently resides. The page number (O....P-1) is used to index this table and the corresponding frame number is picked in order to access the page contents.

In order to execute a program, some pages of it are loaded of it are loaded in the main memory depending upon storage availability. During its execution, the program generates page number which is mapped to find its physical location and then the page is accessed.

Segmentation—यह भी पेजिंग के समान होती है किन्तु इसमें program segments का use होता है। Program segment program की एक logical unit होती है जो programmer define करता है।

Segment से कोई word access करने हेतु logical address की आवश्यकता होती है, जिसके दो भाग होते हैं— Segment number तथा word number.

This scheme is implemented in the same manner as paging is implemented. If differs from paging is use of program segments. A program segment is a logical unit of a program as defined by a programmer. (a part of program logic which performs a specific task). Responsibility lies on the programmer only for determining how many segments a program should have or which program units should comprise a segment.

For accessing a particular word in the segment, its logical address consists of two parts :

(i) segment number and (ii) word number.

This segment number is mapped on to segment table to get physical address of the segment and by adding the word's identification number, physical or actual address of the word is calculated. Again the hardware unit, Address Transformation Unit, is responsible for generation of actual address of the required word within the segment.

DOS files—कम्प्यूटर में सूचना files के रूप में store होती हैं। File एक logically related information का collection होता है। File name के दो भाग होते हैं—Primary name तथा Secondary name (या extension) उदाहरणत: sonia.txt, arshi.bak, neha.doc इत्यादि।

- Text files में .txt extension लगता है। Text files word processing softwares द्वारा बनती है। तथा ASCII code (American Standard Code for Information Interchange) के रूप में store होती है।
- bat files;
- bak files या Back files का extension .bak होता है तथा Safety purpose (means original file के delete, corrupt इत्यादि हो जाने की स्थिति) हेतु save की जाती है।
- Bas अर्थात basic files एक है, (BASIC एक programing language है) (Beginners ALL PURPOSE SYMBOLIC INSTRUCTION CODE) तथा इसकी files का extension .bas होता है।

§ 1.17. Windows :

Operating System के दो भाग होते हैं—Kernel तथा Shell. Kernel भाग hardware से interact करता है जबकि shell भाग user से interact करता है। Shell एक command interpreter है जो user से command प्राप्त करता है, उसका interpretation (अर्थ निकालना) करता है तथा उसी अनुसार action लेता है। user तथा computer आपस में किस प्रकार Interact करते हैं, इसको Interface कहा जाता है। यदि user द्वारा keyboard पर टाइप करके commands दी जाती है तो यह CUI अर्थात character user interface कहलाता है जैसे DOS। यदि user mouse click द्वारा commands देता है तो यह GUI अर्थात graphical user interface कहलाता है जैसे कि windows.

"GUI (Graphical User Interface) based operating system have shells that offer Graphical elements for interaction between computer hardware and user."

Windows के बारे में महत्वपूर्ण बातें निम्नवत् हैं—

- Windows XP एक GUI based ऑपरेटिंग सिस्टम है, तथा इसमें विभिन्न Windows पर multiple applications एक साथ run कर सकती हैं।
- जो पहली स्क्रीन Window open करने पर सामने आती है, जिस पर icons display होते हैं, desktop कहलाती है।
- Window एक rectangular area है, जो किसी application या document या dialog से संबंधित होता है।
- Icon एक graphical संकेत (symbol) होता है जो किसी window element (file, folder या shortcut) को represent (व्यक्त) करता है।
- Operating System को computer की memory में load करना booting कहलाता है।

- Windows XP is a GUI based operating system. In windows operating system, multiple applications can be simultaneously run in different windows.
- In Windows XP, the screen upon which icons, windows, etc. are displayed is known as desktop.
- A window is a rectangular area pertaining to an application or a document or a dialog.
- An icon (आइकन)is a graphic symbol that represents a window element like file, folder or shortcut.
- Loading up of operating system files into the computer's memory is called booting up (बूटिंग).
- We can explore your computer through Start button & Taskbar, My Computer and window Explorer. The Taskbar is a bar, which is usually at the bottom of the screen. The Start button is located at taskbar. By clicking at Start button, Start menu appears wherefrom you can start programs, open documents, customize your system, get help, search for items on your computer and more. My Computer is useful for viewing the contents of a single folder or drive. Windows Explorer is another way of seeing what is on your computer. Windows explorer shows the computer' contents as a hierarchy.
- A folder (फोल्डर) is a location in which you can store files and other folders.
- To create a new folder, File New Folder commands are clicked in My Computer window.
- To find files or folders, Start Find Files or Folder commands are clicked.
- To move a file, the file icon or name is firstly selected, then Edit Cut commands are clicked. Then the destination folder is opened in a My Computer window and there Edit Copy command is clicked. Then the destination folder is opened in a My Computer window, and there Edit Paste commands are clicked.
- To create a shortcut to a file, firstly select the file or folder, whosw shortcut is to be created. Then drag the file icon through right mouse button to desired location where shortcut is to be placed. And then select Create Shortcut(s).
- To shut down the computer (कम्प्यूटर को शट-डाउन करने हेतु), Start Turn Off Computer commands are clicked.

§ 1.18. Information Technology :

सूचना प्रौद्योगिकी का तात्पर्य है, सूचना को एकत्रित करना, प्रोसैस करना, उसको ध्वनि, चित्र, टैक्स्ट या न्यूमैरिक रूप में disseminate करना।

The aquisition, processing, storage and dissemination of pictorial, numeric, textual and vocal information by a mean of computer and tele communication devices is called Information technology.

Technology in Office Automation—विभिन्न offices automation हेतु वित नये IT उपकरणों से सुसज्जित हो रहे हैं; जैसे—

- Pentium-IV computers
- UPS
- Monitors
- LAPTOPS
- Printers
- FAX machine
- Speakers
- Modems
- The process of automation office tasks with the help of computer technology is called office automation (OA). Different types of tasks are performed in an office. These tasks include:

- ❑ Decision-making
- ❑ Data manipulation
- ❑ Document handling
- ❑ Communication & information storage

- **Office automation** refers to the varied computer machinery and software used to digitally create, collect, store, manipulate, and relay office information needed for accomplishing basic tasks.
- Raw data storage, electronic transfer, and the management of electronic business information comprise the basic activities of an office automation system.
- Office automation helps in optimizing or automating existing office procedures.
- The backbone of office automation is a LAN, which allows users to transfer data, mail and even voice across the network.
- Can get many tasks accomplished faster.
- Eliminates the need for a large staff.
- Less storage is required to store data.
- Multiple people can update data simultaneously in the event of changes in schedule.
- Office automation systems (OAS) are configurations of networked computer hardware and software.
- Applied to business and communication functions that used to be performed manually
- Types of functions integrated by office automation systems include (1) electronic publishing; (2) electronic communication; (3) electronic collaboration; (4) image processing; and (5) office management.
- LAN allows users to transmit data, voice, mail, and images across the network to any destination, whether that destination is in the local office on the LAN, or anywhere else through a connecting network.
- Makes office work more efficient and increases productivity.

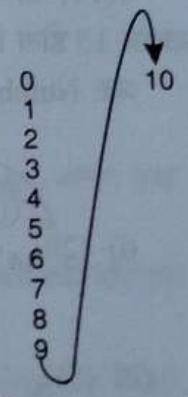
§ 1.19. Number Systems :

हालांकि हम सब decimal नम्बर सिस्टम से परिचित है किन्तु कम्प्यूटर सिस्टम्स में कई अन्य नम्बर सिस्टम भी प्रचलित है जैसे Binary, Octal तथा Hexadecimal.

What is a number system—नम्बर सिस्टम क्या होता है इसको जानने के लिये पहले समझिये, कि हम गिनती कैसे करते हैं। इसके लिये हमने संकेत (symbols) निर्धारित कर रखे हैं, जैसे 0, 1, 2, 3, 4, 5..... इत्यादि। किन्तु गिनती तो unlimited (असीमित) होती है जबकि संकेतों की संख्या सीमित होती है, अत: संकेतों को बार बार repeat करना आवश्यक हो जाता है।

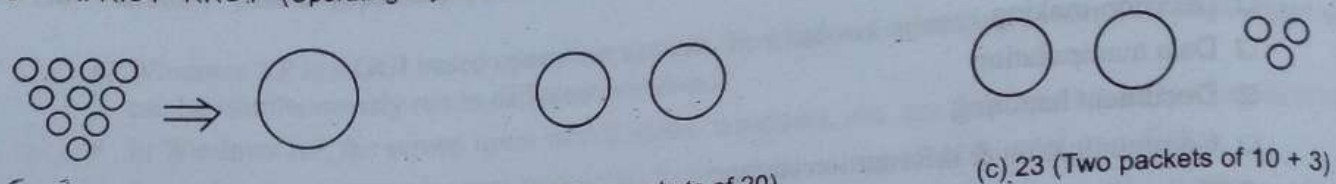
आमतौर पर, हम दशमलव संख्या प्रणाली (Decimal number system) का करते हैं, जिसमें दस संकेत होते हैं—0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

जैसे ही अंतिम संकेत समाप्त होता है, हम अगली गिनती लिखते हैं 10। आइये, समझें कि 10 का क्या अर्थ है (चित्र 1.8)



चित्र 1.8—Decimal Counting

आप बताइये कि 9 के बाद 10 क्यों लिखा गया। यहाँ 10 का 1 यह कर रहा है कि सभी संकेत एक बार समाप्त हो चुके हैं तथा फिर से पहले संकेत अर्थात 0 से शुरुआत की जा रही है। इसी प्रकार 20 यह indicate करता है कि सभी संकेत दो बार पूरे गिने जा चुके हैं तथा तीसरी बार शुरुआत है रही है इत्यादि।

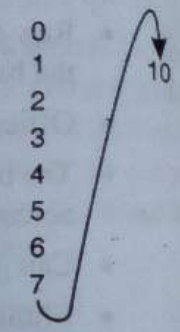


चित्र 1.9—Decimal number system

तो कहने का तात्पर्य यह है कि कुल संकेतों की संख्या दस है, इसीलिये decimal number सिस्टम का base (आधार) 10 है।

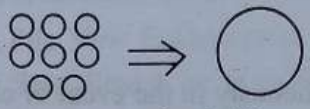
"The number of symbols used in a number system is called its base."

Octal Number System—आक्टल नम्बर सिस्टम की base 8 होती है—अत: चित्र 1.10 के अनुसार counting होगी।

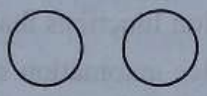


चित्र 1.10

(a) यहाँ का $(10)_8$ (one, zero octal) का मतलब 8 होगा।

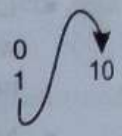


(b) $(20)_8$ (Two Zero Octal) का मतलब 8, 8 के दो पैकेट अर्थात decimal 16 होगा।



(c) $(23)_8$ (Two three Octal) का मतलब होगा। $2 \times 8 + 3 = 19$

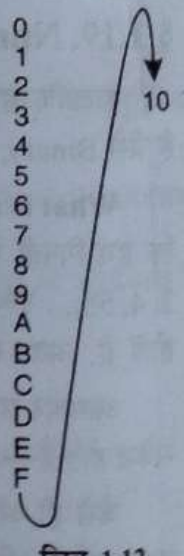
वाइनरी नम्बर सिस्टम—इसका बेस 2 होता है तथा केवल 1 या 0 का use किया जाता है। (चित्र 1.11)



चित्र 1.11

अत: बाइनरी नम्बर सिस्टम में 10 का अर्थ 2 होगा, दस नहीं होगा।

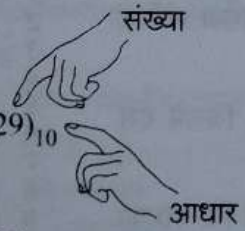
Hexadecimal Number System—इसका base 16 होता है, अत: 16 symbols use किये जाते हैं। 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (चित्र 1.12)



चित्र 1.12

यहाँ A का मतलब 10, B का मतलब 11, C का मतलब 12, D का मतलब 13, E का मतलब 14, F का मतलब 15 होता है। अत: $(16)_{10}$ को hexadecimal में 10 लिखा जायेगा।

अत: Number System अध्ययन करते समय किसी संख्या के साथ उसका base भी लिखना चाहिये जैसे—



(i) $(29)_{10}$

(ii) $(56)_8$

(iii) $(9A)_{10}$ इत्यादि।

Conversion of decimal number (base 10) to any other base (say base x).

(i) प्रश्न उठता है कि एक नम्बर सिस्टम से दूसरे नम्बर सिस्टम (मान base x) में कनवर्जन कैसे किया जाये। इसके लिये decimal नम्बर को x से लगातार divide कीजिये, तथा शेष (remainder) right side में लिखते जाइये। अब remainders को उल्टे क्रम में लिख दीजिये। ध्यान रहे कि यदि decimal number को hexadecimal में बदल रहे हैं तो 10, 11, 12, 13, 14 तथा 15 आने पर क्रमश: A, B, C, D, E तथा F लिखिये।

उदाहरण—Convert

(i) $(120)_{10} = ()_2$

(ii) $(120)_{10} = ()_8$

(iii) $(120)_{10} = ()_{16}$

हल—(i) $(120)_{10} = ()_2$

Divide by 2

2	120	0
2	60	0
2	30	0
2	15	1
2	7	1
2	3	1
2	1	1
2	0	0

अत: $(120)_{10} = (1111000)_2$

(ii) $(120)_{10} = ( )_8$

Divide by 8

8	120	0
8	15	7
8	1	1
	0	

अत: $(120)_{10} = (170)_3$

(iii) $(120)_{10} = ( )_{16}$

Divide by 16

16	120	8
16	7	7
	0	

अत: $(120)_{10} = (78)_{16}$

(ii) यदि किसी नम्बर में integer part के साथ साथ fraction part भी हो, तो integer part का conversion अलग तथा fraction part का अलग होगा।

Fractional decimal नम्बर को जिस सिस्टम में convert करना है उसकी बेस से fraction को multiply कीजिये। परिणाम का fraction (ध्यान दें, केवल fractional part) पुन: x से multiply कीजिये।

इस प्रकार प्राप्त results के integral parts को सीधे क्रम में लिख दीजिये (ध्यान दें कि hexadecimal system में 10, 11, 12, 13, 14 तथा 15 के लिये क्रमश: A, B, C, D, E तथा F लिखें।

उदाहरण—Convert

(i) $(0.625)_{10} = ( )_2$

(ii) $(0.60)_{10} = ( )_8$

(iii) $(0.9)_{10} = ( )_{16}$

(i) $(0.625)_{10}=(\ )_2$

Multiply by 2

$0.625\times 2=$ | 1.250

$0.250\times 2=$ | 0.500

$0.500\times 2=$ ↓ 1.000

अत: $(0.625)_{10}=(0.101)_2$

(ii) $(0.6)_{10}=(\ )_8$

Multiply by 8

$0.6\times 8=$ | 4.8

$0.8\times 8=$ | 6.4

$0.4\times 8=$ ↓ 3.2

अत: $(0.6)_{10}=(0.463)_8$

(iii) $(0.9)_{10}=(\ )_{16}$

Multiply by 16

$0.9\times 16=14.4 \Rightarrow$ | E.4

$0.4\times 16=6.4 \Rightarrow$ | 6.4

$0.4\times 16=6.4 \Rightarrow$ ↓ 6.4

अत: $(0.9)_{10}=(0.E66)_{16}$

अत: उक्त उदाहरणों से—

$(120.625)_{10}=(1111000.101)_2$,

$(120.6)=(170.463)_8$,

$(120.9)_{10}=(78.E66)_{16}$ इत्यादि।

यदि नम्बर किसी अन्य बेस में हो, व डैसीमल में ले जाना हो तो क्या करें?

इसका उत्तर अत्यंत simple है, प्रत्येक संख्या के प्रत्येक digit के साथ उसकी decimal के सापेक्ष प्लेस के अनुसार power जुड़ी होती है, decimal के left में पहली संख्या की power 0 उसके left की power 1 उससे left की power 2 इत्यादि। इसी प्रकार decimal के just right की डिजिट की power –1, उससे right वाली की power –2 इत्यादि। अत: बेस के अनुसार इन powers को digit के मान से multiply करके सभी products को जोड़ दीजिये।

उदाहरण—Convert

(i) $(1011.11)_2=(\ )_{10}$

(ii) $(63.2)_8=(\ )_{10}$

(iii) $(A5.4)_{16}=(\ )_{10}$

(i) $(1011.11)_2$

2^3 2^2 2^1 2^0 2^{-1} 2^{-2}

1 0 1 1 1 1 $)_{10}$

$(1011.11) = 1\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0+1\times 2^{-1}+1\times 2^{-2}$

$=8+0+2+1+0.5+0.25$

अत:

$(1011.11)_2=(11.75)_{10}$

(ii) $(63.2)_8 = (\ )_8$

$(63.2)_8 = 6 \times 8^1 + 3 \times 8^0 + 2 \times 8^{-1}$

$= 48 + 3 + 0.25$

अत:

$(63.2)_8 = (51.25)_{10}$

(iii) $(A\,5.4)_{16} = (\ )_{10}$

$(A\,5.4)_{16} = 10 \times 16^1 + 5 \times 16^0 + 4 \times 16^{-1}$

$= 160 + 5 + 0.25$

अत:

$(A\,5.4)_{16} = (160.25)_{10}$

अभ्यास प्रश्न

Now do the following conversion :

(i) $(28)_{10} = (\ )_2$ (ii) $(545)_{10} = (\ )_8$

(iii) $(895)_{10} = (\ )_{16}$ (iv) $(28.92)_{10} = (\ )_2$

बाइनरी, आक्टल व हैक्साडैसीमल संख्याओं का आपस में कनवर्ज़न—आपने देखा कि किस प्रकार किसी डैसीमल संख्या को किसी अन्य बेस में या किसी अन्य बेस के संख्या को डैसीमल में कनवर्ट किया जा सकता है। अब यदि आप किसी संख्या को किसी एक बेस से (डैसीमल के अलावा) ले जाना चाहते हैं तो एक तरीका तो यह है कि पहले उस संख्या को डैसीमल में ले जायें, फिर दूसरी बेस में ले जायें।

उदाहरण—

$(54)_8 = (\ )_{16}$

अत:

$(54)_8 = 4 \times 8^0 + 5 \times 8^1$

$= 4 + 40$

अत:

$(54)_8 = (44)_{10}$

अब

16	44	C
16	2	2
	0	

अत:

$(44)_{10} = (2C)_{16}$

अत:

$(54)_8 = (44)_{10} = (2C)_{16}$

इसी प्रकार

$(54)_8 = (\ )_2$

चूँकि $(54)_8 = (44)_{10}$

अत:

2	44	0
2	22	0
2	11	1
2	5	1
2	2	0
2	1	1
	0	

अत: $(54)_8 = (44)_{10} = (101100)_2$

लेकिन octal संख्या को बाइनरी या बाइनरी को octal में कनवर्ट करने हेतु एक direct विधि भी है—इसके लिए निम्न table याद कीजिये—

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

इसको याद करने हेतु पहले 0 तथा 1 को 4 बार लिखिये, अगली बार 001; को 2 बार लिखिये, फिर 00001111 एक बार लिखिये।

अब यदि $(54)_8$ का बाइनरी ज्ञात करना है तो 5 तथा 4 octal संख्याओं का बाइनरी (अर्थात 101 तथा 100) लिख दीजिये।

अत:

$(54)_8 = (101\ 100)_2$

तो हो गया डायरैक्ट कनवर्ज़न

इसी प्रकार

$(54.23)_8 = (101100.010011)_2$

किसी बाइनरी संख्या को ऑकटल में ले जाने हेतु तीन-तीन के groups बनाइये, decimal से left तथा decimal से right. यदि अन्तिम ग्रुप तीन से कम का हो तो संख्या के दोनों कोनों में 0 लगाकर तीन का पूरा कर लीजिये। अब सब बाइनरी संख्याओं का octal लिख दीजिये—

उदाहरणत:

$(1011011.11101)_2 = (\ )_8$

अब इसका octal लिखिये—

अत: $(001011011.111010)_2 = (133.72)_8$

यदि किसी बाइनरी संख्या को हैक्साडैसीमल या हैक्सडैसीमल संख्या को बाइनरी में कनवर्ट करना है तो निम्न तालिका याद कीजिये।

हैक्साडेसीमल	बाइनरी			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

Convert

$(3A.B)_{16} = (\ )_2$

अत: तालिका से

$(3A.B)_{16} = (00111010.1011)_2$

Convert

$(101111100.110010)_2 = (\ )_{16}$

यहाँ चार के ग्रुप complete कीजिये—

0001 0111 1100 · 1100 1000

अत:

$(101111100.110010)_2 = (17C.C8)_{16}$

Convert

$(23.4)_8 = (\ )_{16}$

Convert 23.4 binary

$(23.4)_8 = (010011.100)_2$

Now group into four

$(0001\ 0011\ .\ 1000)_2 = (13.8)_{16}$

Addition and Subtraction in Various number System : मान लीजिये आपको दो डैसीमल संख्याओं का योग करना है—

$$\begin{array}{r} {}^{1} \\ (59)_{10} \\ (36)_{10} \\ \hline (95)_{10} \end{array}$$

9 + 6 = 15
इसमें से 10 Minus किया, व 5 लिख दिया,
next position पर 1 carry चल गया

यहाँ आप यह सोचे कि 15 में से 10 ही minus क्यों किया गया ? क्योंकि संख्याओं का base 10 है। अब दो हैक्साडैसीमल संख्याओं को जोडिये—

Examples :

$$\begin{array}{r} (38)_{16} \\ +\ (54)_{16} \\ \hline (8C)_{16} \end{array}$$

$$\begin{array}{r} {}^{1} \\ (59)_{16} \\ +\ (39)_{16} \\ \hline (92)_{16} \end{array}$$

(नोट करें–9 + 9 = 18, 18 – 16 = 2
अतः 2 लिखा तथा 1 को carry में भेज दिया)

इसी प्रकार Subtraction Operations कीजिये—

$$\begin{array}{r} (94)_{10} \\ -\ (36)_{10} \\ \hline (48)_{10} \end{array}$$

यहाँ Borrow लेने पर
10 जोड़ना होगा

$$\begin{array}{r} (63)_{8} \\ -\ (16)_{8} \\ \hline (45)_{8} \end{array}$$

यहाँ Borrow लेने पर
8 जोड़ना होगा

$$\begin{array}{r} (83)_{16} \\ -\ (4C)_{16} \\ \hline (37)_{16} \end{array}$$

यहाँ Borrow लेने पर
16 जोड़ना होगा

अब दो octal संख्याओं को जोड़िये—

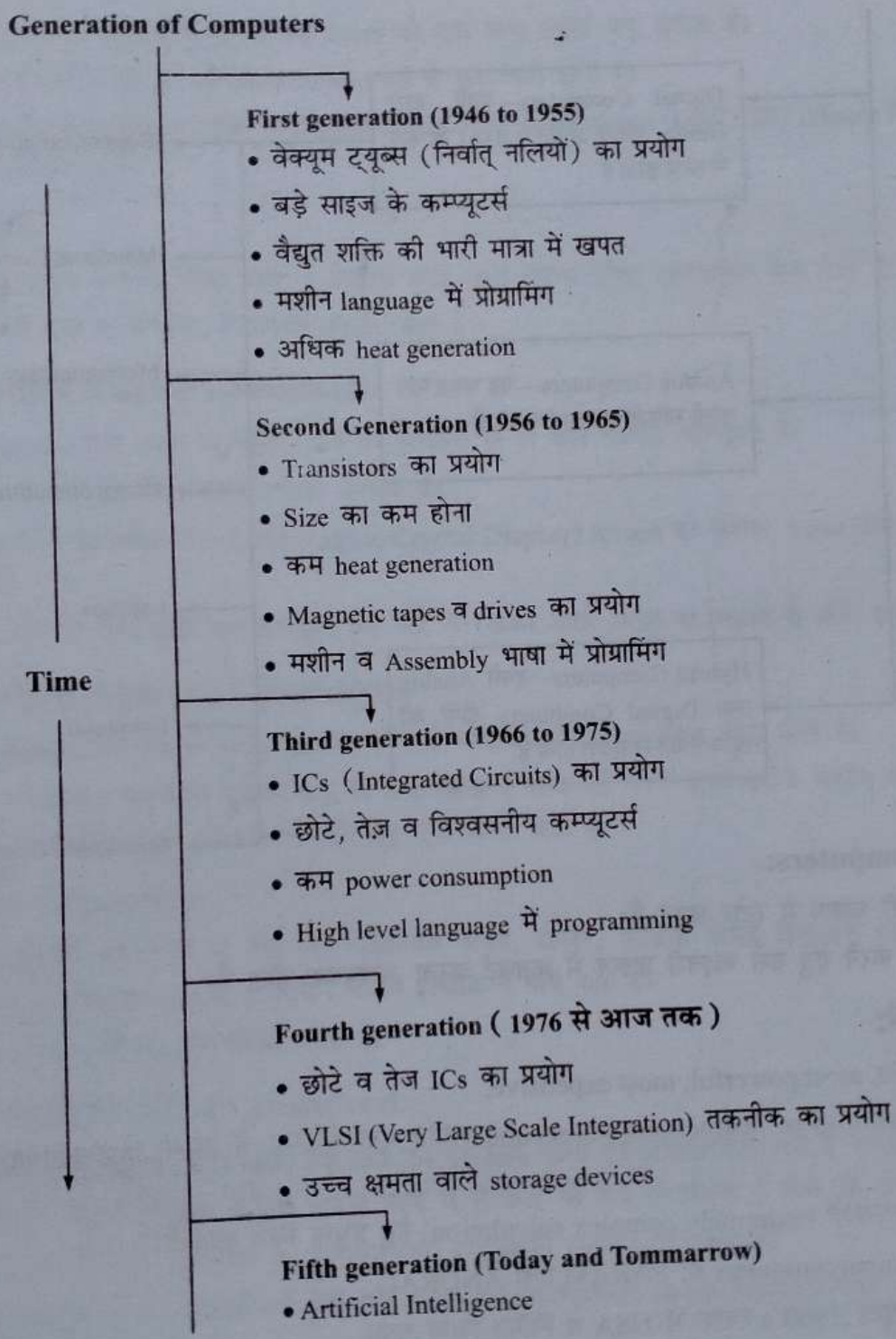
$$\begin{array}{r} (26)_{8} \\ (35)_{8} \\ \hline 63 \end{array}$$

6 + 5 = 11
इसमें से 8 minus कीजिये
Next Position पर 1 carry भेजिये

§ 1.20. कम्प्यूटर की विभिन्न जनरेशन्स (Generations of Computers) :

कम्प्यूटर्स को विभिन्न generations (पीढ़ियों) में वर्गीकृत किया गया है जिसका विवरण तालिका 1.1 में प्रदर्शित है—

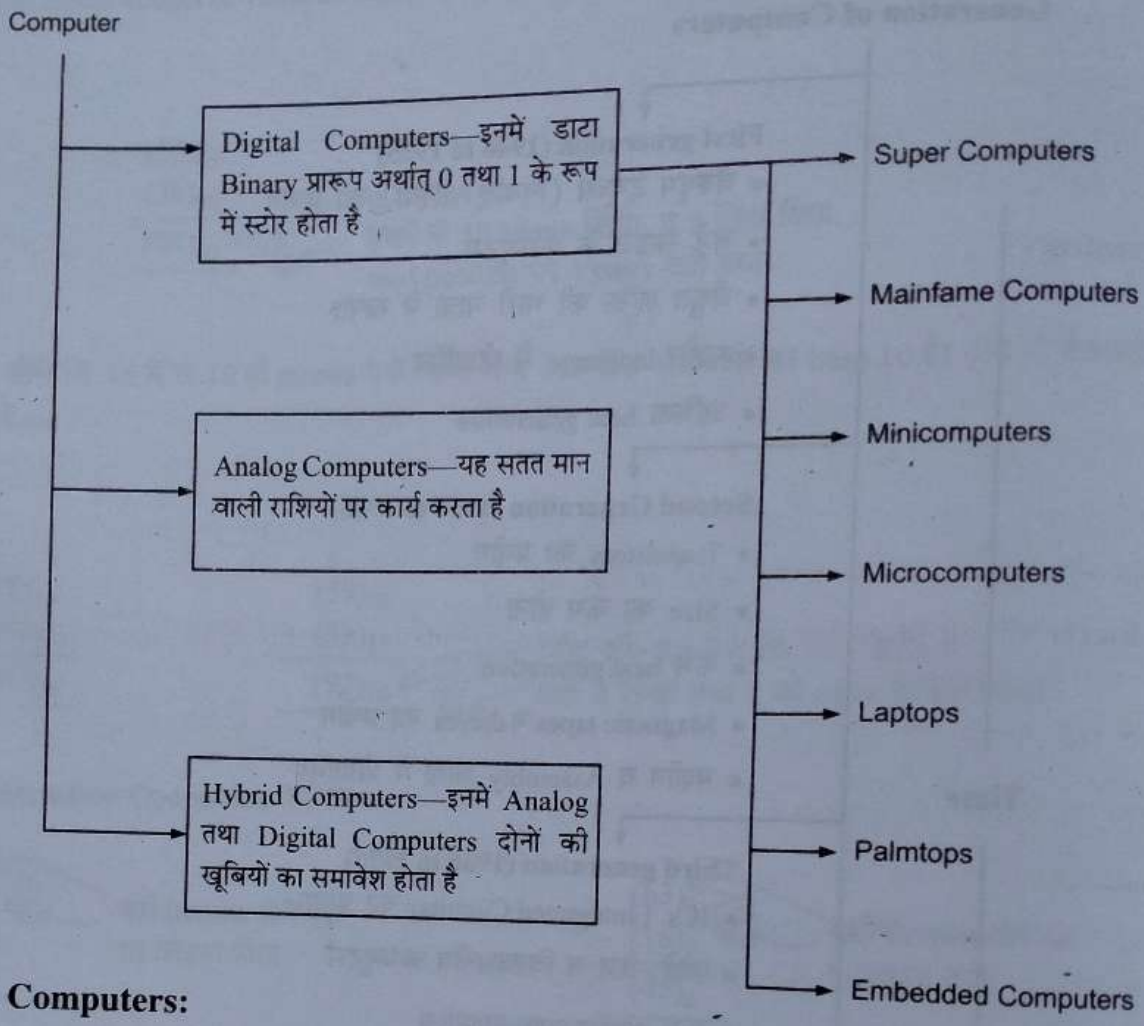
तालिका 1.1—Generation of Computers



§ 1.21. कम्प्यूटर्स का वर्गीकरण (Classification of Computers) :

कम्प्यूटर्स को विभिन्न आधारों पर वर्गीकृत किया जा सकता है जैसे कि आकार (size) के आधार पर, processing speed (कार्य करने के गति) के आधार पर, cost (मूल्य) के आधार पर इत्यादि। कम्प्यूटर्स का वर्गीकरण तालिका 1.2 में प्रदर्शित है—

तालिका 1.2



➢ Digital Computers:

- डेटा को बाइनरी प्रारूप में स्टोर करते हैं।
- डेटा को स्टोर करने हेतु उसे बाइनरी प्रारूप में कनवर्ट करना आवश्यक होता है।

➢ सुपर कम्प्यूटर्स:

- Largest, fastest, most powerful, most expensive.
- Can process billions of instructions per second अर्थात् एक सैकेण्ड में अरबों instructions को process कर सकते हैं।
- अत्यंत जटिल गणनाओं (extremely complex calculation) हेतु प्रयुक्त किये जाते हैं।
- भारत में निर्मित supercomputers हैं: PARAM तथा ANURAG
- पहला सुपर कम्प्यूटर 1960's दशक में USA में निर्मित किया गया।

➢ मेनफ्रेम कम्प्यूटर्स:

- यह Business industries, offices में use किये जाते हैं।
- यह बड़े, उच्च शक्ति के कम्प्यूटर्स होते हैं जो एक सैकेण्ड में अरबों instructions process कर सकते हैं।
- Multiprocessing क्षमता वाले होते हैं, कई processes को एक साथ कर सकते हैं।

➢ मिनी कम्प्यूटर्स:

- यह भी शक्तिशाली कम्प्यूटर होता है जो कई users को एक साथ सपोर्ट कर सकता है।
- Minicomputers की स्पीड mainframe computers से कुछ कम होती है।
- पहला minicomputer 1960's दशक में Digital Equipment Corporation (DEC) द्वारा release किया गया (DEC PDP-8 minicomputer)

➢ माइक्रोकम्प्यूटर:

- इनको माइक्रोकम्प्यूटर इसलिए कहा जाता है क्योंकि इनके द्वारा प्रयुक्त पॉवर अपेक्षाकृत कम होती है।
- इनको विद्यालयों, घरों व ऑफिस में उपयोग किया जाता है।

➢ लैपटॉप कम्प्यूटर्स (Laptop Computers):

- यह एक portable (एक स्थान से दूसरे स्थान पर आसानी से ले जाने योग्य) कम्प्यूटर है।
- आधुनिक लैपटॉप की प्रौसेसिंग क्षमता काफी अधिक है।
- इसमें tracker बॉल या माउस तथा LCD (Liquid Crystal Display) screen को क्रमश: Input तथा Output हेतु प्रयुक्त किया जाता है।
- लैप का अर्थ होता है गोद चूँकि इन कम्प्यूटर को गोद में रखकर कार्य किया जा सकता है अत: इन्हें लैपटॉप कहते हैं।

➢ पामटॉप कम्प्यूटर्स (Palmtop Computers):

- Palmtop computer को Pico computer भी कहते हैं क्योंकि इनका size बहुत छोटा होता है।
- इनको PDA (परसबल डिजिटल एसिस्टेन्ट) भी कहा जाता है तथा यह फोन डाइरेक्टरीज़ मेनटेन करने हेतु, कैलेण्डर्स, कैलकुलेटर्स, Appointments schedule करने हेतु यूज किया जाता है।

➢ Embedded Computers:

- ये घरेलू इलैक्ट्रॉनिक उपकरणों में जैसे कि माइक्रोवेव अवन, टोस्टर, वीडियो कैसेट रिकार्डस (VCR), Videogame Players, Airconditioners (A.C.), वाशिंग मशीन इत्यादि में पाये जाते हैं।
- इनको Automobiles में भी यूज़ किया जाता है।

➢ एनालॉग कम्प्यूटर्स (Analog Computers):

- यह सतत् मौलिक राशियों को स्टोर करने हेतु तथा उन पर कार्य करने हेतु प्रयोग किये जाते हैं अर्थात् ऐसी राशियाँ जिनका मान सतत् रूप से परिवर्तित होता है तथा अनन्त मानों में से कोई भी मान ले सकता है जैसे कि ताप, दाब, धारा, वोल्टेज स्पीड, त्वरण, वेग, इत्यादि (Works on continuous physical values quantities)
- इनको speedometer, oil refineries में ताप तथा flow measurement, traffic light control system इत्यादि में use किया जाता है।

➢ हाइब्रिड कम्प्यूटर (Hybrid Computer):

- यह Analog तथा Digital Computer दोनों की खूबियों का use करते हैं।
- इनको विशेष कार्यों हेतु use किया जाता है।
- इनको मौसम के हाल की जानकारी करने में (Weather forecasting), अस्पतालों में मरीज़ों के हार्टबीट, ब्लडप्रेशर मॉनीटरिंग इत्यादि हेतु use किया जाता है।

§ 1.22. इनपुट युक्तियाँ (Input Devices) :

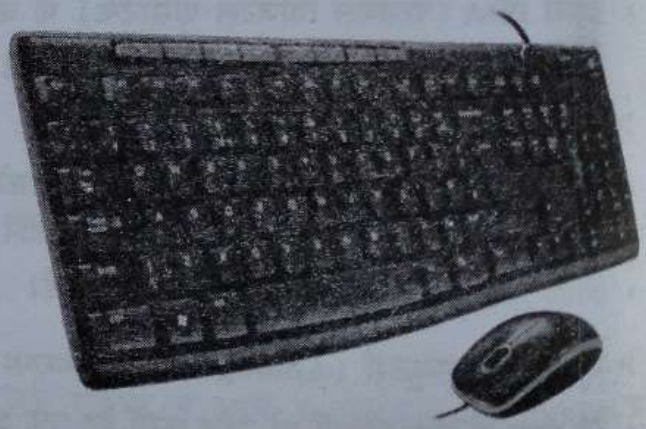
कम्प्यूटर में डेटा Input करने हेतु प्रयुक्त युक्तियाँ Input युक्तियाँ कहलाती है। विभिन्न Input युक्तियाँ निम्नवत है—

➢ Keyboard:

- यह टाइपराइटर के तरह होता है (चित्र 1.13)।
- जब यूज़र Keyboard की key को press करता है तो एक electronic signal भेजा जाता है जो bit pattern (1 तथा 0 का समूह) में convert हो जाता है।
- Keyboards में ASCII (America Standard Code for Information Interchange) का यूज़ किया जाता है।
- Keyboards में न्यूमैरिक, अल्फाबैटिकल तथा स्पेशल कैरेक्टर हेतु keys होती है।
- अधिकतर कम्प्यूटर Keyboards QWERTY Keyboards होते हैं जिनमें 8-bit का code तथा 256 करैक्टर्स का यूज़ किया जाता है। इनकी सबसे ऊपर वाली Keys में अक्षरों का क्रम "QWERTY" होता है।
- *Set of typewriter—like keys that enables you to enter data into a computer and other devices.*
- *Standard layout of letters, numbers, and punctuation is known as a QWERTY keyboard because the first six keys on the top row of letters spell QWERTY.*
- *Keys typically found on computer keyboards are:*
- **Alphanumeric keys:** *The letters and numbers on the keyboard.*
- **Punctuation keys:** *The comma, period, semicolon, and similiar keys.*
- **Special keys:** *This includes the function keys, control keys, arrow keys, caps lock key, and so on.*

➢ Mouse:

- माउस एक प्वाइंटिग डिवाइस है (चित्र 1.13) तथा Screen के Pointer को control करता है।
- यह एक palm size device है। (लगभग हथेली के साइज के बराबर)
- इसमें Ball का यूज़ होता है।
- यह कम्प्यूटर से केबिल द्वारा जोड़ा जाता है इसमें दो बटन तथा एक Scroll wheel होती है।
- जब Mouse को किसी सतह पर move किया जाता है तो Ball Roll करती है जिससे Pointer भी screen पर move करता है। और options को select किया जा सकता है।



चित्र 1.13—की-बोर्ड तथा माउस

- प्रोग्राम को select करने हेतु single click, प्रोग्राम को select तथा open करने हेतु double click तथा Window को move करने हेतु drag operation का यूज़ किया जाता है।
- *Optical mouse LED light का यूज़ करते हैं जबकि Laser mouse Laser तकनीक का यूज़ करते हैं।*
- *A pointing device that detects two-dimensional motion relative to a surface.*
- *Motion is translated into the motion of a pointer on a display, which allows a smooth control of the graphical user interface.*
- *Controls the motion of a pointer in two dimensions in a graphical user interface (GUI).*
- *Turns movements of the hand backward and forward, left and right into equivalent electronic signals that in turn are used to move the pointer.*

- *Relative movements of the mouse on the surface are applied to the position of the pointer on the screen, which signals the point where actions of the user take place, so that the hand movements are replicated by the pointer.*
- *Controls the movement of the cursor or pointer on a display screen.*
- *A small object you can roll along a hard, flat surface.*
- *Name derived from its shape, which looks a bit like a mouse, its connecting wire can be imagined to be the mouse's tail.*
- *As the mouse is moved , the pointer on the display screen moves in the same direction.*
- **Mechanical:** *Has a rubber or metal ball on its underside that can roll in all directions. Mechanical sensors within the mouse detect the direction the ball is rolling and move the screen pointer accordingly.*
- **Optomechanical:** *Same as a mechanical mouse, but uses optical sensors to detect motion of the ball.*
- **Optical:** *Uses a laser to detect the mouse's movement,have no mechanical moving parts, respond more quickly and precisely but they are more expensive.*

➢ Tracker Ball:

चित्र 1.14—ट्रैकर बॉल

- यह एक upside down mouse है (अर्थात् mouse के उल्टे आकार वाली युक्ति है) (चित्र 1.14)
- इसमें ball को rotate किया जाता है जबकि mouse एक जगह स्थिर रहता है।
- यह कम जगह लेता है।
- इसको video games, Laptops इत्यादि में यूज़ किया जाता है।
- *A pointing device consisting of a ball held by a socket containing sensors to detect a rotation of the ball about two axes-like an upside-down mouse with an exposed protruding ball.*
- *User rolls the ball with the thumb, fingers, or the palm of the hand to move a pointer.*

➢ Joystick:

चित्र 1.15—जॉयस्टिक

- यह एक pointing device है (चित्र 1.15) ।
- इसको video games में यूज़ किया जाता है।
- उद्योगों में इसको Robots को control करने हेतु यूज़ किया जाता है।
- *An input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling.*
- *Similar to a mouse, except that with a mouse the cursor stops moving as soon as you stop moving the mouse.*
- *The pointer continues moving in the direction the joystick is pointing.*
- *To stop the pointer, joystick must be returned to its upright position.*
- *Used mostly for computer games, but also for CAD/CAM systems and other applications.*

➢ Touch Screen:

- यह एक विशेष प्रकार की Screen होती है जो स्पर्श के प्रति संवेदनशील (Touch sensitive) होती है।
- इनको Screen पर Option Select करने हेतु यूज़ किया जाता है।

- *Touch Screen, ATM, Mobilephone, restaurents इत्यादि में यूज़ होती है।*
- *A computer display screen that is also an input device.*
- *Screens are sensitive to pressure; a user interacts with the computer by touching pictures or words on the screen.*
- *An input device normally layered on the top of an electronic visual display of an information processing system.*
- *Touchscreen technology makes it possible to interact with a computer system using direct touch of the electronic display instead of using a traditional keyboard and mouse.*
- *Types of touch screen technology are: Resistive: coated with a thin metallic electrically conductive and resistive layer that causes a change in the electrical current which is registered as a touch event and sent to the controller for processing. Surface wave: uses ultrasonic waves that pass over the touch screen panel. Capacitive: coated with a material that stores electrical charges.*

➢ Light Pen:

- यह इन्जीनियर तथा Architect (शिल्पकार) द्वारा यूज़ किया जाता है। इसमें एक Light sensitive (प्रकाश के प्रति संवेदनशील) cell का यूज़ होता है जिससे कम्प्यूटर Screen की position indicate की जाती है।
- इसको operate करने हेतु pen को screen से touch करना पड़ता है।
- इसको CAD (कम्प्यूटर एडेड डिज़ाइन) अनुप्रयोगों हेतु यूज़ किया जाता है।
- *An input device that utilizes a light-sensitive detector to select objects on a display screen.*
- *Similar to a mouse, except that with a light pen you can move the pointer and select objects on the display screen by directly pointing to the objects with the pen.*
- *A light sensitive input device used in conjunction with a computer's CRT display.*
- *Allows the user to point to displayed objects or draw on the screen in a similar way to a touchscreen but with greater positional accuracy.*

➢ Bar Code Reader:

- Bar code काली लाइनों की एक श्रेणी होती है (चित्र 1.16(a)) जिनकी चौड़ाई भिन्न-भिन्न होती है। यह कई उत्पादों पर Printed होता है।
- Bar code reader bar code की सूचना को decode करता है (चित्र 1.16(b))।
- यह एक लेज़र बीम का उत्सर्जन करता है जो कि bar code image से टकराकर वापस लौटती है।

(a) बार कोड रीडर

(b) बार कोड

चित्र 1.16

- Bar code का light detector इस सूचना की पहचान करता है।
- Bar code को Libraries, Books, उत्पादों, Speed post एवं अनेकों अनुप्रयोगों में यूज किया जाता है।
- *Barcode is an optical machine-readable representation of data relating to the object to which it is attached(a small image of lines (bars) and spaces that is affixed to retail store items, identification cards, and postal mail(speed posts and registerd posts etc.) to identify a particular product number or location). A barcode reader is an electronic device that can read and output printed barcodes to a computer.*
- *Consists of a light source, a lens and a light sensor translating optical impulses into electrical ones.*
- *Barcode readers contain decoder circuitry analyzing the barcode's image data provided by the sensor and sending the barcode's content to the scanner's output port.*
- *A hand-held or stationary input device used to capture and read information contained in a bar code .*
- *Consists of a scanner, a decoder and a cable used to connect the reader with a computer.*

➢ Scanner:

- इनको text या picture को scan करने हेतु यूज़ किया जाता है ताकि इनको computer में store किया जा सके।
- Scanner विभिन्न प्रकार के होते हैं जैसे कि Hand-held scanner, flat-bed scanner इत्यादि।
- *Scanner is a device that captures images from photographic prints, posters, magazine pages, and similar sources for computer editing and display.*
- *Come in hand-held, feed-in, and flatbed types and for scanning black-and-white only, or color.*
- *Very high resolution scanners are used for scanning for high-resolution printing*
- *Lower resolution scanners are sufficient for capturing images for computer display.*

Optical Character Reader (OCR):

- Optical character recognition data को input करने की एक विशेष विधि है जिसमें scanner के साथ-साथ एक विशेष software का use करता है जो कि scanned image को ASCII code में परिवर्तित कर देता है।
- इस text को word processing software द्वारा edit भी किया जा सकता है क्योंकि इसको picture की तरह treat नहीं किया जाता बल्कि text की तरह treat किया जाता है।
- OCR के लाभ हैं—data entry त्रुटियों का कम होना, peak loads को handle करना, Human readable बनाना इत्यादि।

OMR (Optical Mark Reader):

- OMR sheets के बारे में आपने काफी सुना होगा। आजकल अधिकतर बहुविकल्पीय प्रश्न वाली परीक्षाओं में इसको use किया जाता है तथा सही विकल्प वाले गोले को HB पैंसिल से ब्लैक कर दिया जाता है।
- Optical mark readers इन marks (काले निशान) को sense कर लेते हैं तथा इस प्रकार इनकी checking की जाती है।
- इसका लाभ है यह है कि बड़े पैमाने पर data की handling कम time, कम cost पर आसानी से हो जाती है।

Magnetic Ink Character Reader (MICR):

- MICR documents को directly पढ़ने की क्षमता रखता है जिनमें characters विशेष प्रकार की चुम्बकीय स्याही (special magnetic ink) (Iron oxide based ink) द्वारा लिखे गये होते हैं।
- यह scanning system cheques की processing हेतु बैंकिंग सिस्टम द्वारा 1950's में विकसित किया गया।
- इस सिस्टम में चैक में Bank, branch, account number इत्यादि magnetic ink में printed होते हैं।
- Costly, होता है। Very large scale applications हेतु ही उपयुक्त होता है।
- उच्च security

Microphone:

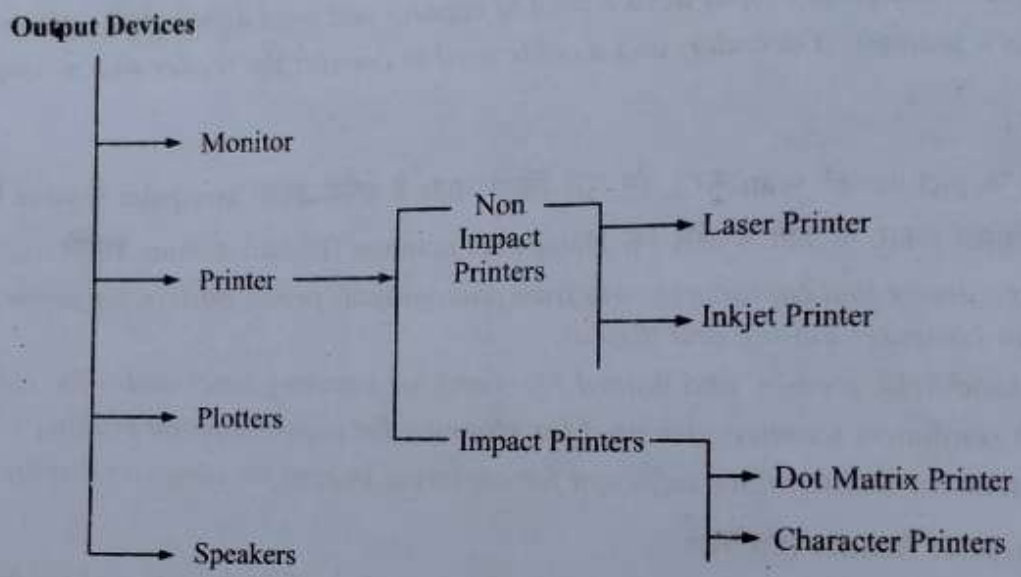
- यह ध्वनि सिगनल को वैद्युत सिगनल में कनवर्ट करता है।

डिजिटल कैमरा:

- यह पिक्चर्स को डिजिटल प्रारूप में कनवर्ट करता है।

§ 1.23. आउटपुट युक्तियाँ (Output Devices) :

कम्प्यूटर से आउटपुट प्राप्त करने हेतु विभिन्न आउटपुट युक्तियों का प्रयोग किया जाता है। विभिन्न आउटपुट युक्तियाँ निम्नवत् हैं—



➢ मॉनीटर:

- इसको VDU (Video Display Unit) भी कहा जाता है।
- इसकी सहायता से results को screen पर देखा जा सकता है।
- Monitors का size उसके diagonal (विकर्ण) द्वारा specify किया जाता है अर्थात् 21 inch screen का diagonal 21 inch लम्बाई का होगा।
- Screen की Resolutions screen पर pixels की संख्या द्वारा निर्धारित होती है। मॉनीटर का डिस्प्ले लाखों dots से मिलकर बना होता है, यह dots picture elements या pixels कहलाते हैं।
- Display हेतु CRT (Cathode Ray Tube) screen या LCD (Liquid Crystal Display) screen प्रयोग की जाती है।

➢ प्रिंटर :

प्रिंटर कम्प्यूटर द्वारा किसी document का printout लेने हेतु प्रयोगे किया जाता है। Printout को document की hard copy भी कहा जाता है। Printer दो प्रकार के होते हैं—Impact printers व non-impact printers. Impact printers paper पर print करते समय paper को स्पर्श करते है जबकि non-impact printer, पेपर को touch नहीं करते। Dot matrix printers (DMP) Daisy wheel printers व Character printers impact printer की श्रेणी में आते हैं जबकि inkjet व laser printers non-impact printers की श्रेणी में आते है।

डॉट मैटरिक्स प्रिंटर्स (DMPs) :

- इसमें pins की सहायता से ink से coat किये गये ribbon पर चोट की जाती है, जिससे ink ribbon से paper पर ट्रांसफर हो जाती है तथा character paper पर print हो जाता है।
- प्रिंट की carbon copies भी बनाई जा सकती है।
- कम कीमत
- Poor quality, very slow, very noisy, less resolution.
- कलर प्रिंट संभव नहीं है।

इंकजैट प्रिंटर्स (Inkjet Printers) :

- इसमें paper पर ink के droplets (छोटी बूँदें) का spray (छिड़काव) करके प्रिंट लिया जाता है।
- इनका path चुम्बकीय प्लेट्स द्वारा direct किया जा सकता है।
- जब पेपर को प्रिंटर में feed किया जाता है तो printer head आगे पीछे (back and forth) गति करता है जिससे पेज पर ऐसे हजारों droplets का spray हो जाता है। इन droplets का diameter मनुष्य के बालों (human hairs) से भी कम होता है।
- हल्का आकार, 300–600 dpi की image, office तथा home use हेतु प्रयुक्त।
- Slower than laser printers, more cartridge cost.

लेजर प्रिंटर्स (Laser Printers) :

- स्थिर वैद्युतिकी (static electricity) के सिद्धान्त पर आधारित।
- Laser की सहायता से पेपर पर positive charge उत्पन्न किया जाता है, तथा फिर paper पर टोनर (toner या ink powder) का spray किया जाता है। Ink केवल वहीं चिपकती है जहाँ positive charge होता है।
- उच्च स्पीड व बेहतरीन प्रिंट क्वालिटी (टैक्स्ट व ग्राफिक्स दोनों के लिये), उच्च resolution (600–1200 dpi)
- पेज के गीले होने welting की समस्या से मुक्त।
- महंगा, carbon copies उत्पन्न नहीं की जा सकती।
- *A printer (चित्र 1.17) is a device that accepts text and graphic output from a computer and transfers the information to paper, usually to standard size sheets of paper.*
- *An external device that communicates with another digital device to print what a user sees on a screen.*

चित्र 1.17—प्रिंटर

- *Use small pixels to transfer an image from the system to another surface.*
- *A peripheral which makes a persistent human readable representation of graphics or text on paper or similar physical media.*
- *Laser printer rapidly produces high quality text and graphics.*
- *Laser printers employ a xerographic printing process*
- *Inkjet printers operate by propelling variably sized droplets of liquid ink onto almost any sized page.*
- *They are the most common type of computer printer used by consumers. printer is a piece of haraware for a computer.*
- *Allows the user to print items on paper, such as letters and pictures.*

➢ प्लॉटर्स (Plotters) :

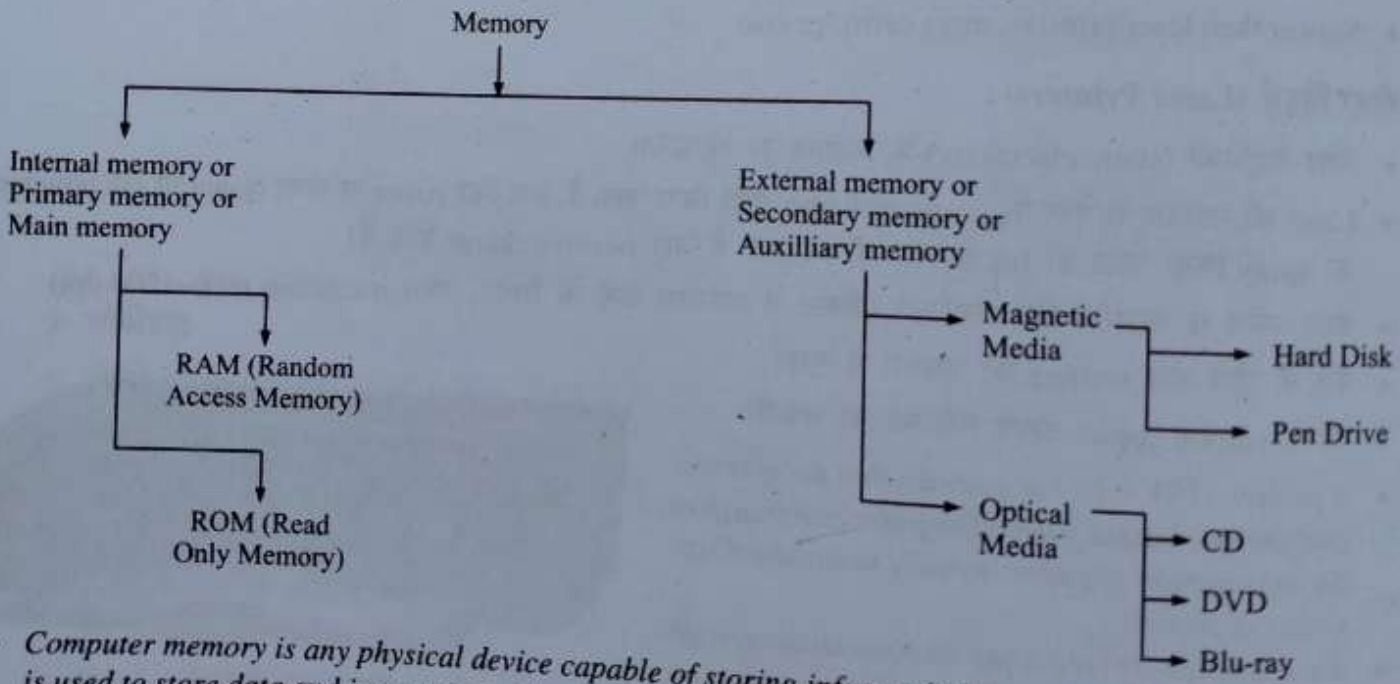
- प्लॉटर्स भी एक प्रकार के प्रिंटर्स होते हैं जो उच्च क्वालिटी की drawings, images तथा graphics की hard copy उत्पन्न करने हेतु प्रयोग किये जाते हैं।
- यह प्रिंटर्स की तुलना में महंगे होते हैं।
- Plotters कई प्रकार के होते हैं—ड्रम प्लॉटर (इसमें पेपर को rotating drum पर mount किया जाता है), फ्लैट बैड प्लॉटर (इनमें पेपर को horizontal surface पर fix किया जाता है), इत्यादि।

➢ स्पीकर्स (Speakers) :

- Speakers वैद्युत signals को ध्वनि signals में कनवर्ट करते हैं।
- कम्प्यूटर्स स्पीकर्स की सहायता से ध्वनि व संगीत play करते हैं।
- Stereo speakers में दो चैनल्स होते हैं—Left channel तथा Right channel, जिससे sound अधिक वास्तविक प्रतीत होता है।

§ 1.24. मैमोरी का वर्गीकरण (Classification of Memory) :

मैमोरी का वर्गीकरण निम्नवत् है—



- *Computer memory is any physical device capable of storing information temporarily or permanently. It is used to store data and instructions. Computer memory is the storage space in computer where data is to be processed and instructions required for processing are stored. The memory is divided into large number of small parts called cells. Memory is primarily of three types:*
  - ❑ *Cache Memory*
  - ❑ *Primary Memory/Main Memory*
  - ❑ *Secondary Memory*

Cache Memory: *Cache memory is a very high speed semiconductor memory which can speed up CPU. It acts as a buffer between the CPU and main memory. It is used to hold those parts of data and program which are most frequently used by CPU. The parts of data and programs are transferred from disk to cache memory by operating system, from where CPU can access them.*

- *Faster than main memory.*

- *Consumes less access time as compared to main memory.*
- *Stores the program that can be executed within a short period of time.*
- *Stores data for temporary use.*
- *Has limited capacity.*
- *Very expensive.*

Primary Memory (Main Memory): *Primary memory holds only those data and instructions on which computer is currently working. It has limited capacity and data is lost when power is switched off. It is generally made up of semiconductor device. It is divided into two subcategories RAM and ROM.*

- *Semiconductor memories.*
- *Usually volatile, Data lost in case power is switched off.*
- *Working memory of the computer,Faster than secondary memories.*

Secondary Memory: *It is slower than main memory. These are used for storing data/Information permanently. CPU directly does not access these memories instead they are accessed via input-output routines. Contents of secondary memories are first transferred to main memory, and then CPU can access it.*

- *Known as backup memory.*
- *Non-volatile, Data permanently stored even if power is switched off.*
- *Used for storage of data in a computer.*
- *Slower than primary memories.*

➢ Internal Memory:

- इसमें प्रोग्राम्स व डेटा स्टोर किये जाते हैं।
- यह built-in memory होती है, तथा यहाँ डेटा को manipulate किया जाता है।
- यह दो प्रकार की होती है—RAM तथा ROM

➢ RAM:

- Processing के समय data को temporarily (थोड़े समय के लिये) store करती है।
- किसी भी cell को randomly access किया जा सकता है।
- यह volatile memory होती है अर्थात् power के switch off होते ही data lost हो जाता है।

➢ ROM:

- यह processor के अंदर (चित्र के रूप में) होती है तथा इसका data user द्वारा चेंज नहीं किया जा सकता।
- कम्प्यूटर के निर्माण के समय ROM में प्रोग्राम्स स्टोर कर किये जाते हैं।
- ROM पर stored डेटा से यह निर्धारित होता है कि operating system कैसे load होगा।
- ROM एक non-volatile memory है अर्थात् power के off होने पर भी data lost नहीं होता।
- Computer को start करने हेतु निर्देश एक विशेष चिप पर होते हैं जिसको ROM-BIOS चिप कहा जाता है (ROM = Read Only Memory, BIOS = Basic input/output system)

Exernal Memory :

- इसको secondary memory या auxilliary memory भी कहा जाता है।
- यह बड़ी मात्रा में डेटा को permanently store करने के काम आती है तथा primary memory से कम cost की होती है।
- Hard disk, Pen drive, CD ROMs, DVD's, Memory sticks, Blu ray इसके उदाहरण हैं।

➢ हार्ड डिस्क (Hard Disk) :

- यह Desktop computer में होती है।
- इसमें कई सारे platters होते हैं तथा डेटा को platters के दोनों sides में store किया जाता है।
- डिस्क लगातार घूमती है तथा इनमें चुम्बकीय पदार्थ की कोटिंग होती है।
- यह traces तथा sectors में विभाजित होती है।
- Storage capacity 40GB से 150 GB तक होती है।
- *Hard disk is a data storage device used for storing and retrieving digital information using one or more rigid rapidly rotating disks called platters coated with magnetic material.*
- *Platters are paired with magnetic heads arranged on a moving actuator arm, which read and write data to the platter surfaces.*
- *A hard disk is part of hard disk drive that stores and provides relatively quick access to large amounts of data on an electromagnetically charged surface or set of surfaces.*
- *Modern computers typically come with a hard disk that contains gigabytes of storage.*

➢ CD-ROM :

- इसमें डेटा डिस्क पर डिजिटल form में स्टोर किया जाता है।
- इस डेटा को laser beam द्वारा read किया जाता है।
- इसकी capacity 650 MB तक होती है।
- डेटा को चेंज नहीं किया जा सकता।
- *CD-ROMs are stamped by the vendor, and then cannot be erased and filled with new data.*
- *To read a CD, a CD-ROM player is needed.*
- *Suitable for large storage capacity information like large software applications that support color, graphics, sound, and video.*

➢ DVD :

- यह एक optical storage device है जिसमें बड़ी मात्रा में डेटा स्टोर किया जा सकता है तथा इससे डेटा को fast access किया जा सकता है।
- इसमें विभिन्न कम्प्यूटर डेटा जैसे photons, videos, files, games इत्यादि store किये जा सकते हैं।

➢ Blu-ray (ब्लू-रे) :

- इसमें डेटा को read व write करने हेतु blue-violet laser का use किया जाता है (wavelength 405 nanometer), जिसके कारण इसका नाम Blue-ray पड़ा।
- इसकी स्टोरेज क्षमता CD व DVD की तुलना में बहुत अधिक होती है।
- Blu-ray डिस्क की शुरूआत के पश्चात् कई Industries के standards को पूरा करने हेतु कई companies ने मिलकर BDA (Blu-ray disc Association) की स्थापना की।
- Blu-ray तकनीक के नामकरण में Blue के स्थान पर Blu का use इसलिये किया जाता है क्योंकि BDA इसे trademark के रूप में register करना चाहती थी तथा रोजमर्रा के शब्दों को trademark के रूप में register नहीं कराया जा सकता।
- इसकी क्षमता 25 GB तक होती है।

- *Blu-ray is an optical disc format designed to display high definition video and store large amounts of data.*
- *Successor to DVD.*
- *A digital optical disc data storage format designed to supersede the DVD format capable of storing high-definition and ultra high-definition video resolution.*
- *An HD-DVD format that uses a 405 nanometer-wavelength blue-violet laser technology, in contrast to the 650nm-wavelength red laser technology used in traditional DVD formats.*
- *Rewritable Blu-ray disc, with a data transfer rate of 36 Mbps, can hold up to 27 GB of data on a single-sided single layer disc.*
- *The Blu-ray format was developed jointly by Sony, Samsung, Sharp, Matsushita, Pioneer and Philips, Mistubishi, Thomson, LG Electronics and Hitachi.*

➢ Pen Drive (पैन ड्राइव) :

- यह एक removable तथा rewritable memory है जो pen के आकार की होती है (चित्र 1.18 देखें)।
- इसमें एक small circuit board होता है जो कि plastic, rubber या metal case द्वारा सुरक्षित होता है।
- इसको access करने हेतु computer के USB Port (Universal Serial Port) में insert किया जाता है।
- Pen drives कई GB sizes में उपलब्ध है।
- *A pen drive, or a USB flash drive, is a portable data-storage device.*
- *now the most popular data-storage devices among consumers being small in size, lightweight and handy and can be easily carried from place to place.*
- *Currently available pen drives with storage capacities ranging from 8GB and 32GB can be used to store graphics-heavy documents, photos, music files and video clips.*

चित्र 1.18—पैन ड्राइव

यदि कोई register किसी memory का address store करता है तो उस resistor को memory pointer कहा जाता है।

§ 1.25. बूटिंग (Booting) :

जब कम्प्यूटर की पॉवर ON की जाती है तो वह user द्वारा use के लिये तैयार होने से पूर्व कई internal processes (आन्तरिक प्रक्रियाओं) से गुज़रती है। यह process boot process या PC की booting कहलाती है। वास्तव में "boot" शब्द bootstrap का संक्षिप्त रूप है तथा सम्भवत: इसका प्रयोग एक पुराने मुहावरे "Pull yourself up by your bootstraps" अर्थात् बिना किसी दूसरे की मदद के स्वयं को खड़ा करना (to improve your situation without any help from others) से जुड़ा है। Boot प्रक्रिया PC के Basic Input Output System (BIOS) द्वारा नियंत्रित की जाती है।

Basic Input Output System या BIOS एक software होता है जो कि एक flash memory chip पर stored होता है। PC में BIOS motherboard by embedded होता है। Boot process को नियंत्रित करने के साथ-साथ BIOS PC के Hardware घटकों से configuration interface भी प्रदान करता है। Boot process के समय निम्न घटनायें होती हैं—

- Power Button के ON होने पर PC की power supply सक्रिय हो जाती है तथा मदर बोर्ड तथा अन्य घटकों तक पॉवर पहुँच जाती है।
- PC एक power-on self test (POST) perform करता है। POST BIOS के अंदर एक छोटा सा कम्प्यूटर प्रोग्राम होता

है जो hardware failures को check करता है। POST signals के पश्चात् एक single beep यह व्यक्त करती है कि सब कुछ ठीक ठाक है (अर्थात् All is well) दूसरी beep sequence यह व्यक्त करती हैं कि hardware में कुछ गड़बड़ी है तथा PC repair विशेषज्ञ इन beeps की सहायता से यह पता लगा सकते हैं कि कौन सा घटक कार्य नहीं कर रहा है।

- PC द्वारा boot process का विवरण monitor पर display किया जाता है जैसे कि BIOS का निर्माता, Processor की विशिष्टतायें (specifications), instal किये जा चुके RAM की मात्रा, तथा detect की गई युक्तियाँ (devices).
- BIOS drive के पहले sector (boot disk) तक पहुँच बनाने का प्रयास करता है।
- BIOS यह सुनिश्चित करता है कि boot disk के पहले sector में boot loader (या bootstrap loader) है या नहीं तथा वह boot loader को memory में load करता है। Boot loader एक छोटा प्रोग्राम होता है जो कि PC के operating system को ढूंढने व launch करने हेतु designed होता है।
- जैसे ही बूट लोडर मैमोरी में आ जाता है, BIOS अपना कार्य बूट लोडर को सौंप देता है जो कि operating system को memory में load करना प्रारम्भ कर देता है।
- जब Boot loader अपना कार्य पूरा कर लेता है, तो वह PC का नियंत्रण operating system को सौंप देता है। अब OS user interaction हेतु तैयार हो जाता है तथा users अपनी commands देकर कार्य करना प्रारम्भ कर सकता है।

प्रश्नावली

1. (a) कम्प्यूटर क्या होता है?
   (b) डेटा तथा Information क्या होती है?
   (c) कम्प्यूटर का Block डायग्राम बनाइये व समझाइये।
   (d) हार्डवेयर व साफ्टवेयर में क्या अंतर होता है?
   (e) Memory कितने प्रकार की होती है?
   (f) विभिन्न Input व आउटपुट युक्तियाँ बताइये।
2. (a) What do you mean by algorithms and flowcharts ?
   (b) Explain the main advantages of flowcharts in computer programming.
   (c) Draw a flowchart to find the largest of three numbers.
3. (a) What do you mean by data and information?
   (b) Explain the different functional blocks of a computer.
4. What is operating system? What are its functions?
5. What are compilers, interpreters and assemblers?
6. Explain the various stager in program development?
7. What are errors? What are the different types of errors while writing a computer program?
8. What is algorithm? What are its main features?
9. What are flowcharts? What are its advantages?
10. Compare the important of algorithm and flowcharts in writing a computer program.

2 Chapter

ऑपरेटिंग सिस्टम का परिचय
(AN INTRODUCTION TO OPERATING SYSTEMS)

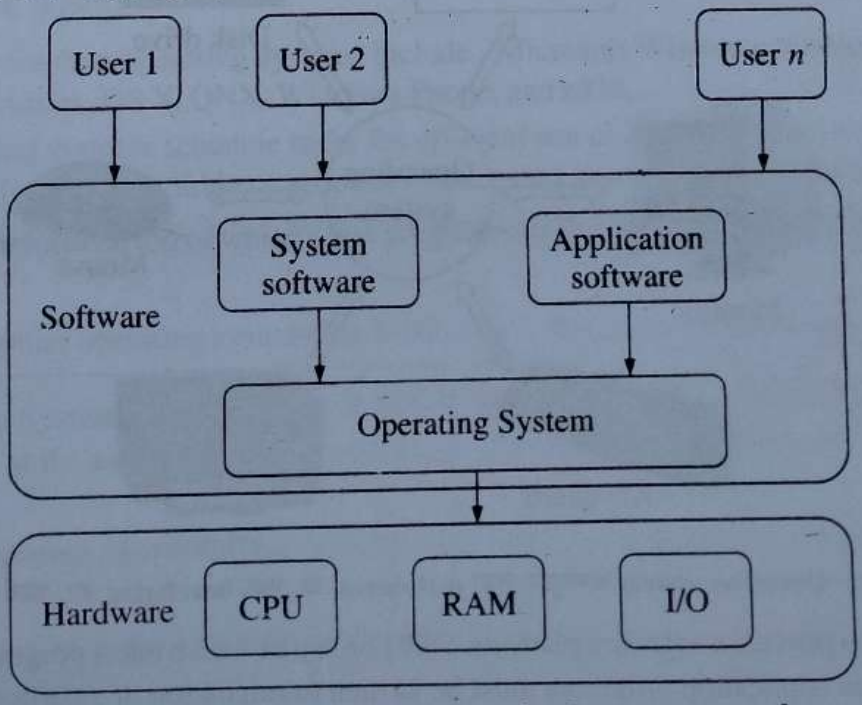
THINK ABOUT IT

Be careful what you water your dreams with. Water them with worry and fear and you will produce weeds that choke the life from your dreams. Water them with optimism and solutions and you will cultivate success. Always be on the lookout for ways to turn a problem into an opportunity for success. Always be on the lookout for ways to nurture your dream. **— Lao Tzu**

§ 2.1. ऑपरेटिंग सिस्टम (Operating System) :

एक ऑपरेटिंग सिस्टम user तथा कम्प्यूटर हार्डवेयर के बीच में मध्यस्थ का कार्य करता है। वह एक ऐसा वातावरण उत्पन्न करता है जिसमें यूजर दक्षता से एवं सुविधापूर्वक प्रोग्राम्स को execute करता है— तकनीकी शब्दों में कहा जाये तो ऑपरेटिंग सिस्टम एक ऐसा साफ्टवेयर होता है जो हार्डवेयर का प्रबन्धन करता है वह विभिन्न स्रोतों एवं सेवाओं जैसे—मेमोरी, प्रौसेसर, युक्तियाँ, सूचना के आवटंन का नियंत्रण करता है।

परिभाषा—एक ऑपरेटिंग सिस्टम वह प्रोग्राम होता है, जोकि यूज़र तथा कम्प्यूटर हार्डवेयर के बीच interface (दो वस्तुओं के बीच की सीमा) का कार्य करता है (चित्र 2.1) तथा सभी प्रकार के प्रोग्राम्स के execution (कार्य का सम्पादन) का नियन्त्रण करता है।



चित्र 2.1—ऑपरेटिंग सिस्टम User व Hardware के बीच मध्यस्थ का कार्य करता है

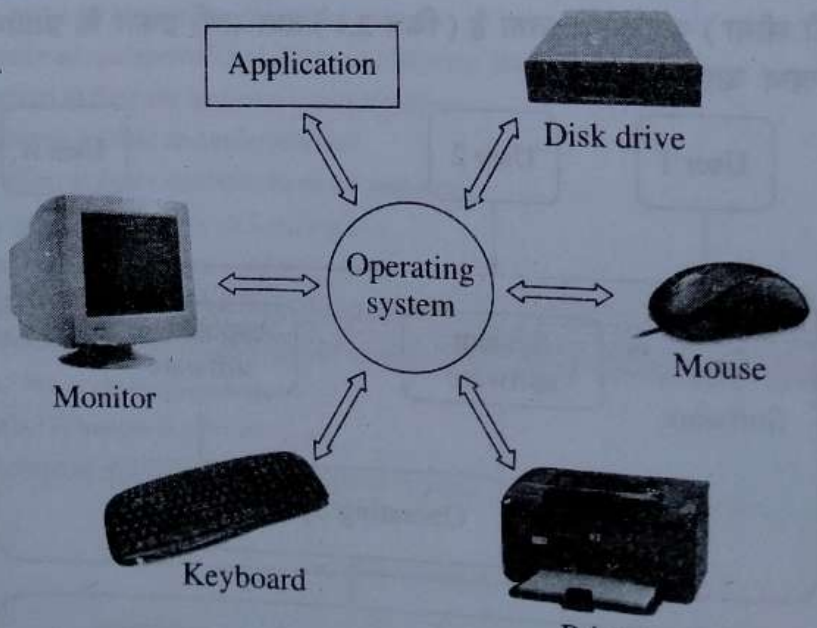
Computer के हार्डवेयर रिसोर्सस (hardware resources), जैसे—मेमोरी (memory), प्रौसेसर (processor) तथा इनपुट-आउटपुट डिवाइसेस (input-output devices) को व्यवस्थित करने के लिये बनाया गया सिस्टम सॉफ्टवेयर (system software) ही ऑपरेटिंग सिस्टम (operating system) होता है। यह व्यवस्थित रूप से designed सॉफ्टवेयर्स का समूह होता है जो कि आंकड़ों (data) एवं निर्देशों (commands) को नियंत्रित करता है। कम्प्यूटर के प्रत्येक रिसोर्स की स्थिति का लेखा-जोखा ऑपरेटिंग सिस्टम ही रखता है, ऑपरेटिंग सिस्टम ही निर्णय करता है कि किसका, कब और कितनी देर के लिए कम्प्यूटर रिसोर्स पर नियंत्रण होगा।

ऑपरेटिंग सिस्टम user तथा कम्प्यूटर के बीच एक माध्यम का कार्य करता है। इसके अलावा यह हार्डवेयर्स (hardwares) तथा सॉफ्टवेयर्स (softwares) के मध्य एक bridge या मध्यस्थ का कार्य भी करता है। ऑपरेटिंग सिस्टम के बिना computer का अपने आप में कोई अस्तित्व ही नहीं है। यदि ऑपरेटिंग सिस्टम न हो तो कम्प्यूटर अपने हार्डवेयर्स जैसे कि Keyboard, Monitor, CPU आदि के बीच कभी भी सम्बंध स्थापित नहीं कर पायेगा।

An operating system is software that manages (प्रबंधन करना) computer hardware and software resources and provides common services for computer programs. The operating system is an essential component (अनिवार्य घटक) of the system software in a computer system. Application programs usually require an operating system to function.

Operating system is a program that acts as an interface (मध्यस्थ, सीमा) between the user and the computer hardware and controls the execution (क्रियान्वयन) of all kinds of programs.

An operating System (OS) is an intermediary (मध्यस्थ) between users and computer hardware (acts as interface (see Fig. 2.2)). It provides users an environment (वातावरण) in which a user can execute programs conveniently (सुविधाजनक तरीके से, सुविधापूर्वक) and efficiently (दक्षतापूर्वक). It is a software which manages hardware. An operating System controls the allocation (आवंटन) of resources (स्रोत) and services (सेवायें) such as memory, processors, devices and information. The most popular operating system today is Microsoft's Windows operating system.



चित्र 2.2—Operating system कम्प्यूटर तथा peripherals के मध्य interfacing का कार्य करता है

Operating systems provide a software platform (मंच) on top of which other programs, called application programs, can run. The application programs must be written to run on top of a particular operating system.

The operating system of a computer is the first software that gets installed on the hard disk (हार्डवेयर डिस्क पर instal होने वाला सबसे पहला software), and it remains there even when the computer is turned off. The OS is also the first software that gets loaded into the computer's memory when it is turned on.

Without an OS, the computer would not even start up (OS के बिना कम्प्यूटर start नहीं हो पायेगा). The first task of the OS is to manage the starting up of the computer, also known as booting up (कम्प्यूटर को start करने अर्थात् बूट-अप करने की जिम्मेदारी OS की होती है). When this happens, the OS ensures (सुनिश्चित करना) all the various elements of the computer are working properly.

The operating system is the most important program that is on a computer. The operating system basically runs the computer and allows other programs to run as well (कम्प्यूटर पर run करता है तथा शेष प्रोग्राम्स को run होने की अनुमति प्रदान करता है). The operating system does all the basic things that a computer needs to do, such as recognizing inputs from input devices such as the mouse or the keyboard (इनपुट युक्तियों से आने वाले इनपुट्स को पहचानना), it keeps track of where all the files are on the computer (files को track करना), it allocates resources to the various programs that are running and it prevents unauthorized access to the computer (Resources का आवंटन करना, अनाधिकृत पहुँच पर रोक लगाना)।

Operating System

- A software that manages (प्रबंधन) computer hardware and software resources
- Provides common services for computer programs (सेवायें प्रदान करना)।
- An essential component of the system software in a computer system (अनिवार्य घटक)।
- Application programs usually require an operating system to function (Application प्रोग्राम्स हेतु आवश्यक)।
- An intermediary between users and computer hardware (मध्यस्थ)।
- Provides users an environment in which a user can execute programs conveniently and efficiently (सुविधाजनक वातावरण प्रदान करना)।
- controls the allocation of resources and services such as memory, processors, devices and information (Resources पर नियंत्रण कराना)।
- Examples of popular modern operating systems include , Microsoft Windows, Android, BlackBerry 10, Chrome OS, iOS, Linux, OS X, QNX, Windows Phone, and z/OS.
- Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing and other resources.
- provide a software platform on top of which other programs, called application programs, can run. (मंच तैयार करना)।
- For PCs, the most popular operating systems are DOS, OS/2, Windows and sometimes Linux.

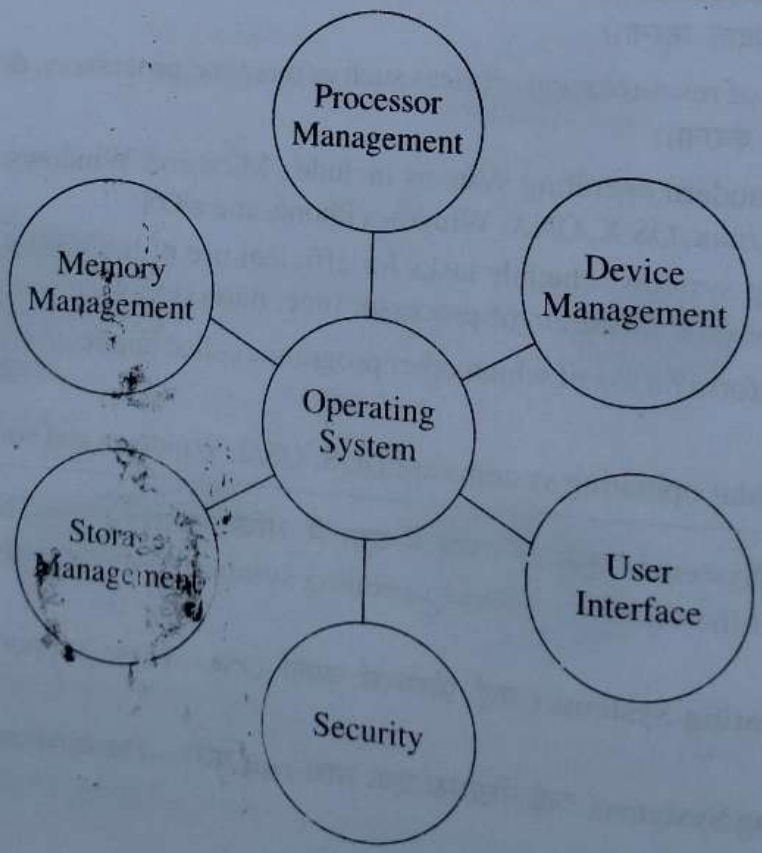
- **Multi-user Operating Systems (एक ही समय में एक से अधिक users हेतु)**—These Allows two or more users to run programs at the same time. Some operating systems permit hundreds or even thousands of concurrent users.
- **Multiprocessing Operating Systems (कई प्रौसेसर्स वाला OS)**—These Support running a program on more than one CPU.
- **Multitasking Operating Systems (कई प्रोग्राम्स एक साथ run करें)**—These Allow more than one program to run concurrently.

- **Multithreading Operating Systems (एक प्रोग्राम के कई भाग एक साथ run करें)**—These Allow different parts of a single program to run concurrently.
- **Real Time Operating Systems**—These respond to input instantly. General-purpose operating systems, such as DOS and UNIX, are not real-time.
- **Interactive Operating System**—An interactive operating system allows the user to directly interact with the user while one or more programs are running अर्थात् वह operating system जिसमें user सीधे computer से interact करने की अनुमति प्रदान करता है, Interactive operating system कहलाता है। इसके लिये user interface की help ली जाती है। यह User Interface Command Line Interface (CUI) या Graphical User Interface (GUI) होता है। अधिकांश operating systems interactive type होते हैं।

§ 2.2. ऑपरेटिंग सिस्टम के कार्य (Functions of Operating Systems) :

ऑपरेटिंग system के मुख्य कार्य निम्नवत् है—

- मैमोरी प्रबन्धन (Memory Management)
- प्रौसेसर प्रबन्धन (Processor Management)
- युक्ति प्रबन्धन (Device Management)
- सचिका प्रबन्ध (File Management)
- सुरक्षा (Security)
- सिस्टम के कार्य करने की क्षमता पर नियन्त्रण (Control Over System Performance)
- जॉब एकाउन्टिग (Job Accounting)
- त्रुटियों को डिटेक्ट करने में Help (Error Detecting Aids)
- साफ्टवेयर तथा यूज़र के मध्य समन्वय स्थापित करना (Coordination between other Software and Users)



मैमोरी प्रबन्धन—

इसका तात्पर्य प्राइमरी मेमोरी (प्राथमिक मेमोरी) तथा मुख्य मेमोरी (main memory) के प्रबन्धन से है। मेन मेमोरी बाइट्स या वर्ड्स का एक बड़ा व्यूह होता है (large array of words or bytes), जिसमें से प्रत्येक बाइट का या वर्ड का भिन्न address होता है।

Main मेमोरी से तीव्र स्टोरेज (fast storage) प्राप्त हो जाती है जिस पर कि CPU द्वारा सीधे पहुँच (access) बनाई जा सकती है।

मेमोरी प्रबन्धन हेतु ऑपरेटिंग सिस्टम निम्न कार्य करता है—

- प्राइमरी मेमोरी को ट्रैक करता है अर्थात् यह ध्यान रखता है कि उसके कौन से भाग का कौन प्रयोग कर रहा है एवं कौन-कौन से भाग प्रयोग में नहीं है।
- मल्टी प्रोग्रामिंग में ऑपरेटिंग सिस्टम यह निर्धारित करता है कि कौन सा प्रौसेस मेमोरी प्राप्त करेगा एवं कब और कितनी मेमोरी प्राप्त करेगा।
- जब कोई प्रौसेस मेमोरी request करता है तो ऑपरेटिंग सिस्टम मेमोरी का आंवटन करता है।
- जब प्रौसेस को मेमोरी की आवश्यकता नहीं होती या प्रौसेस समाप्त हो जाता है तो ऑपरेटिंग सिस्टम मेमोरी को deallocate (आवंटन को निरस्त करना या वापिस प्राप्त करना) करता है।

The memory manager (मैमोरी प्रबंधक) in an operating system coordinates (समन्वय स्थापित करना) the memories by tracking (पता लगाना, जानकारी करना) which one is available (उपलब्धता), which is to be allocated (आवंटन) or deallocated (आवंटन का निरस्तीकरण) and how to swap (आदान-प्रदान) between the main memory and secondary memories. The operating system tracks all memory used by each process so that when a process terminates (समाप्त होना), all memory used by that process will be available for other processes. Memory management means the management of Main Memory. Main memory provides a fast storage that can be access directly by the CPU. For a program to be executed, it must in the main memory.

Let us see what the operating System does for memory management.

- *Keeps tracks of primary memory i.e. which part of it are in use by whom and which part are not in use.*
- *In multiprogramming, OS decides which process will get memory when and how much.*
- *Allocates the memory whenever the process requests for it.*
- *De-allocates (takes back) the memory after the process no longer needs it or has been terminated.*

प्रौसेसर प्रबन्धन—

मल्टी प्रोग्रामिंग वातावरण में OS ये निर्धारण करता है कि कौन सा प्रौसेस प्रौसेसर को प्राप्त करेगा तथा कितने समय के लिए व कब प्राप्त करेगा। यह कार्य प्रौसेस शैड्यूलिंग (process scheduling) कहलाता है।

प्रौसेसर प्रबन्धन हेतु ऑपरेटिंग सिस्टम निम्न कार्य करता है—

- प्रौसेसर को ट्रैक करता है तथा प्रौसेस के स्टेट्स को ट्रैक करता है। इस कार्य हेतु प्रोग्राम को ट्रैफिक कन्ट्रोलर कहा जाता है।
- प्रौसेस हेतु प्रौसेसर (CPU) आवंटित करता है।
- जब प्रौसेसर की आवश्यकता नहीं होती तो प्रौसेसर को de-allocates (आवंटन को निरस्त करना या पुन: प्राप्त करना) करता है।

It deals with running multiple processes. मल्टी प्रोसेसिंग means एक समय में एक से अधिक कार्य के क्रियान्वयन के लिए सिस्टम पर एक से अधिक CPU रहते हैं। इस तकनीक को मल्टी प्रोसेसिंग कहते हैं। एक से अधिक प्रौसेसर उपलब्ध होने के कारण इनपुट आउटपुट एवं प्रोसेसिंग, तीनों कार्यों के मध्य समन्वय रहता है।

Most operating system allow a process to be assigned a priority (प्रायिक्ता निर्धारण) which affects its allocation of CPU time. The OS must allocate resources to processes, enable processes to share and exchange information, protect the resources of each process from other processes and enable synchronisation (समक्रमिकता) among processes. A multiprogramming or multitasking OS is a system executing many processes concurrently (एक साथ). Multiprogramming requires that the processor be allocated to each process for a period of time and de-allocated at a correct (उचित) time. If the processor is de-allocated during the execution of a process, it must be done in such a way that it can be restarted after some time as easily as possible. In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling.

Let us see what the operating System does for processor management.

- *Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller (यातायात नियंत्रक).*
- *Allocates the processor (CPU) to a process.*
- *De-allocates processor (takes the control back) when processor is no longer required (आवश्यकता न होने पर वापस लेना).*

युक्ति प्रबन्धन (Device Management)—

विभिन्न इनपुट व आउटपुट युक्तियां जैसे कि keyboard, mouse, printers, card readers का management भी operating system द्वारा किया जाता है। Operating system इन devices के दक्षतापूर्ण utilization की व्यवस्था करता है। आपरेटिंग सिस्टम विभिन्न युक्तियों के मध्य संचार (डिवाइस कम्यूनिकेशन) को उनके ड्राइवर्स द्वारा मैनेज (manage) करता है। इसके लिए ऑपरेटिंग सिस्टम निम्न कार्य करता है—

- सभी युक्तियों को ट्रैक करता है। इस कार्य को करने वाला प्रोग्राम I/O controller कहलाता है।
- यह decide (निर्णय लेना) करता है कि कौन सा डिवाइस कब और कितने समय के लिए प्रौसेस को दिया जायेग।
- एक दक्षता पूर्ण तरीके से डिवाइस का आवंटन करता है।
- जब कार्य पूर्ण हो जाता है तो डिवाइज को De-allocate करता है।

Device management means activating (सक्रिय करना) and controlling (नियंत्रण करना) the peripheral devices within the computer. In a desktop computer, the operating system interacts (पारस्परिक आदान प्रदान) with the device drivers for peripheral control. In very small embedded systems, the device management routines may be included within the OS.OS manages device communication via their respective drivers. Let us see what the operating System does for device management.

- *Keeps tracks of all devices. Program responsible for this task is known as the I/O controller or Input/output controller (इनपुट/आउटपुट नियंत्रक)*
- *Decides which process gets the device when and for how much time.*
- *Allocates the device in the efficient way.*
- *De-allocates devices (takes the control back).*

File Management—

A file is a collection of related information defined by its creator अर्थात् file सूचनाओं का संग्रह होता है जिसको कि इसके creator द्वारा define किया जाता है।

फाइल सिस्टम को सुविधा तथा आसानी के लिए directories में संगठित (organise) किया जाता है। इन directories में कई फाइल्स तथा दूसरी डायरैक्ट्रीज़ होती है।

फाइल मैनेजमेंट हेतु ऑपरेटिंग सिस्टम निम्न कार्य करता है—

- सूचना (information), location (स्थिति), प्रयोग (use), स्टेटस इत्यादि को ट्रैक करना। इन सुविधाओं को फाइल सिस्टम कहा जाता है।
- Files का creation तथा deletion
- Directories का creation तथा deletion
- Files को secondary storage को map करना
- Stable storage media पर files का back up प्रदान करना

Operating systems have different file systems that controls the creation (उत्पन्न करना), deletion (समाप्त करना), and access (पहुँच बनाना) of files of data and programs. The output of a program may need to be written into new files or input taken from some files. The operating systems provides this service. The user does not have to worry about secondary storage management. User gives a command for reading or writing to a file and sees the task get accomplished. Thus, the operating systems makes it easier for user programs to complete their task.

A file system is generally organized into directories for easy navigation and usage. These directories may contain files and other directories as well.

Let us see what the operating System does for file management.

- *Keeps track of information, location, uses, status etc. The collective facilities (सामूहिक सुविधायें) are often known as file system (फाइल सिस्टम)*
- *Creation and deletion of files and directories.*
- *Support of primitives for manipulating files and directories.*
- *Mapping of files onto secondary storage.*
- *Backup of files on stable storage media.*

अन्य कार्य (Other Activities)—

- Password की सहायता से सुरक्षा प्रदान करना ताकि प्रोग्राम तथा डाटा तक किसी प्रकार की अनाधिकृत पहुँच की सम्भावना न हो सके।
- किसी सेवा हेतु request तथा system की response के मध्य समय देरी (time delay) स्थापित करके system की performance control करता है।
- Job तथा users द्वारा use किये गये resources तथा उनके समय को ट्रैक करके job accounting का कार्य करना।
- विभिन्न त्रुटियों को detect करके error massage उत्पन्न करना और इस प्रकार error detection में सहायता करना (error detecting aids)
- Compilers, interpreters, assemblers तथा computer system के users द्वारा use किये जाने वाले अन्य software के मध्य सामन्जस्य स्थापित करना (Coordination between other softwares and users)।

- ***Security***—*Security means prevention from unauthorized access (अनधिकृत पहुँच से सुरक्षा प्रदान करना) while giving easy access to the authorized user. The operating system does this by means of password and similar other techniques, preventing unauthorized access to programs and data. There are many security threats to the computers, in particular various types of malicious software for example computer viruses, which can interfere with the normal operations of the computer. Computer viruses are malicious software programs which multiply themselves and interrupt the normal working of computer. Viruses can be very harmful and result in loss of data or system crashes. The OS of a computer has a number of built-in tools to protect against security threats, including the use of virus scanning utilities and setting up a firewall to block suspicious network activity. Another security feature is to control access to your computer by setting up a password. Without entering the correct password , one will not be able to get access to the software applications and files on your computer.*
- ***Controlling System Performance***—*The system performance can be controlled by recording delays between request for a service and response from the system.*
- ***Job Accounting***—*The operating system does job accounting by keeping track of time and resources used by various jobs and users.*
- ***Error Detecting Aids***—*An error is one part of the system may cause misfunctioning of the system. The operating system constantly monitors the system for detecting errors. Thereby preventing errors propagating to various part of the system and causing malfunctioning.*
- ***Coordination between Other Softwares and Users***—*Coordination (समन्वय) and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.*
- ***Hardware Services***—*In order to communicate (संचार) with the hardware devices, the OS uses drivers. Device drivers are program files that enable the OS to recognize the hardware device and its properties. Many of these drivers are built into the OS.*
- ***Networking***—*Operating systems are also capable of using the TCP/IP networking protocols. So that one system can appear on a network of the other and share resources (रिसोर्सेज को साझा करना) such as files, printers, and scanners.*

§ 2.3. ऑपरेटिंग सिस्टम का विकास (Evolution of Operating Systems) :

जब operating systems नहीं थे तो programmer को सीधे hardware से interact करना पड़ता था, प्रत्येक instruction तथा program को binary switches का use करके manually keyed-in करना पड़ता था। अगर कुछ गलत हो जाता था तो operator को फिर से वह प्रोग्राम लोड करना पड़ता था। यह कार्य बहुत समय लेता था, अत: एक automatic system की आवश्यकता महसूस होने लगी, जिससे कार्य सरल व तेज़ी से हो जायें और इसी कारण operating system विकसित किये गये। OS की आवश्यकता का मुख्य कारण यह था कि user को सेवायें प्रदान की जा सकें। अत: operating system के डिज़ाइन के मुख्य उद्देश्य थे—(i) Convenience अर्थात् सुविधा ताकि computer को user friendly (प्रयोग करने में सुगम सरल, easy for non-experts to use) बनाया जा सकें, (ii) Efficiency अर्थात् दक्षता ताकि computer के resources का बेहतर उपयोग हो सके तथा (iii) Ability to evolve अर्थात् विकसित होने की क्षमता अर्थात् नये functions के डाले जाने पर सेवा में कोई व्यवधान उत्पन्न न हो, अर्थात् नयी सेवायें आसानी से introduce की जा सकें।

§ 2.4. विभिन्न प्रकार के ऑपरेटिंग सिस्टम्स (Types of Operating Systems) :

ऑपरेटिंग system का प्रयोग कम्प्यूटर के पहली generation से प्रारम्भ हो गया था। समय के साथ कई प्रकार के ऑपरेटिंग सिस्टम्स विकसित हुए।

मुख्य प्रकार के ऑपरेटिंग सिस्टम्स निम्नवत् है—

1. बैच ऑपरेटिंग सिस्टम—

बैच ऑपरेटिंग सिस्टम के यूज़र कम्प्यूटर से सीधे interact (परस्पर कार्य करना) नहीं करते (Do not interact with the computer directly)। प्रत्येक यूज़र अपना जॉब ऑफलाइन डिवाइसेस (जैसे कि पंचकार्ड) द्वारा तैयार करता है तथा कम्प्यूटर ऑपरेटर को सबमिट कर देता है।

प्रौसेसिंग की स्पीड बढ़ाने के लिए एक ही प्रकार कि आवश्यकताओं वाले जॉब को बैच (Batch) बना दिया जाता है तथा एक ग्रुप के रूप में run किया जाता है। अत: प्रोग्रामर्स ऑपरेटर को अपना जॉब देते हैं और ऑपरेटर इन प्रोग्राम्स को sort (छाँटना) करता है तथा एक सी आवश्यकता वाले प्रोग्राम्स को batches में ग्रुप कर देता है।

Batch system की मुख्य समस्यायें निम्न है—

- यूज़र तथा job के मध्य interaction का अभाव (Lack of interaction between the user and job)
- कई बार CPU idle (फालतू या बेकार अर्थात् बिना किसी कार्य के) रहता है क्योंकि मैकेनिकल I/O devices की स्पीड CPU से काफी कम होती है (CPU is often idle, because the speed of the mechanical I/O devices is slower than CPU)।
- प्राथमिकता निर्धारित करने में समस्या (Difficult to decide the desired priority).

The users of batch operating system do not interact with the computer directly (कम्प्यूटर से सीधे interaction नहीं होता) (i.e., no direct interaction involved between user and the computer.). Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group (एक तरह के jobs का बैच तैयार करना). Thus, the programmers left their programs with the operator. The operator then sorts programs into batches with similar requirements.

Limitations—

- *Lack of interaction between the user and job.*
- *CPU is often idle, because the speeds of the mechanical I/O devices is comparatively much slower than CPU.*
- *Difficult to provide the desired priority.*

Simple Batch Systems—

- No direct interaction between user and the computer.
- User has to submit a job (written on cards or tape) to a computer operator.
- Computer operator places a batch of several jobs on an input device.
- Jobs are batched together by type of languages and requirement.
- Then a special program, the monitor, manages the execution of each program in the batch.
- The monitor is always in the main memory and available for execution.

Disadvantages of this type of system are: Zero interaction between user and computer and no mechanism to prioritize processes.

टाइम शैयरिंग ऑपरेटिंग सिस्टम (Time-sharing Operating Systems)—

Time-sharing वह तकनीक है जिसकी सहायता से विभिन्न टर्मिनल्स पर स्थित कई लोग एक ही समय में किसी कम्प्यूटर system को use कर सकते हैं। Time sharing systems attached terminals को main computer या

server से सीधे access प्रदान करते हैं। Server memory के प्रोग्राम्स को एक circular queue के रूप में treat करता है। Operating system प्रत्येक प्रोग्राम को एक fixed time allocate कर देता है (10 से 100 ms)। उस fixed time के लिये program execute होता है, तथा फिर उस time के finish होने के बाद अगला प्रोग्राम execute होता है। CPU इतनी तीव्र गति से कार्य करता है जिससे user को यह महसूस नहीं होता कि वह queue में है।

Time sharing is a technique which enables many people, located at various terminals to use the computer simultaneously. Time sharing operating system provides direct access to the attached terminals from the server. The server treats the programs in the memory in a circular queue and allocate a fixed time (about 10 to 100 ms) to each program. Each program is executed for that fixed time and then the CPU is passed to the next program in the queue. The CPU is so fast that user does not feel that he is in the queue.

Time sharing multitasking, मल्टी प्रोग्रमिंग का तार्किक विस्तार (logical extension) मात्र है। Time sharing का अर्थ है कि विभिन्न users के मध्य प्रौसेसर के टाइम का share (विभाजित) करना।

Multiprogrammed batch systems तथा time-sharing systems में मुख्य अन्तर यह है कि multiprogrammed batch systems का उद्देश्य processor के उपयोग को maximize करना होता है (maximize processor use) अर्थात् प्रौसेसर का ज्यादा से ज्यादा उपयोग किया जा सके जबकि Time-sharing system का मुख्य उद्देश्य response time को minimize करना होता है (minimize response time) जिससे कम से कम समय में (अर्थात् जल्दी से जल्दी) प्रौसेसर response प्राप्त की जा सके।

इसमें CPU द्वारा कई सारे कार्य करवाने हेतु CPU को एक जॉब से दूसरे जॉब के बीच स्विच किया जाता है तथा यह स्विचिंग अत्यन्त तीव्र गति से की जाती है। अत: user को immediate response (तुरन्त response) प्राप्त होतीं है।

उदाहरणत: एक Transaction processing में प्रौसेसर प्रत्येक यूज़र प्रोग्राम को एक छोटे समय (short burst or quantum of computation) के लिए करता है। अत: यदि N users है तो प्रत्येक user को time quantum (समय का एक छोटा सा हिस्सा) दिया जाता है। जब user आदेश (command) देता है तो तीव्र response प्राप्त हो जाती है।

Operating system इस छोटे समय (time quantum) के आवंटन हेतु CPU scheduling तथा multiprogramming का use करता है। जो computer system पहले batch systems के रूप में design किये गये थे, उनको भी time-sharing systems के रूप में modify किया गया है।

लाभ—Time sharing operating systems के मुख्य लाभ निम्नवत् हैं—

- तीव्र response प्रदान करना (quick response)।
- Software का duplication की आवश्यकता ना होना (Avoid duplication of software) अर्थात् कई software प्रतिलिपियों के बजाय एक ही software से काम चल जाता है।
- CPU के फालतू समय का कम होना (Reduces CPU idle time)

हानि या सीमायें—

- विश्वसनीयता की समस्या
- User के प्रोग्राम व डाटा की सुरक्षा की समस्या
- डाटा संचार की समस्या

Multiprogramming Batch Systems—एक ही समय पर दो से अधिक प्रक्रियाओं का प्रचालन होना मल्टी प्रोग्रामिंग कहलाता है। विशेष तकनीक के आधार पर CPU के द्वारा निर्णय लिया जाता है कि इन प्रोग्राम में से किस प्रोग्राम को चलाना है।

- *Operating system, picks and begins to execute one job from memory.*
- *Once this job needs an I/O operation, operating system switches to another job (CPU and OS always busy).*

- *Jobs in the memory are always less than the number of jobs on disk (Job Pool).*
- *If several jobs are ready to run at the same time, then system chooses which one to run (CPU Scheduling).*
- *In Non-multiprogrammed system, there are moments when CPU sits idle and does not do any work.*
- *In Multiprogramming system, CPU will never be idle and keeps on processing.*

In time sharing systems the main aim is on minimizing the response time, (अर्थात् यहाँ मुख्य उद्देश्य होता है response time को कम करना) while in multiprogramming the prime focus is to maximize the CPU usage. Time-Sharing Systems are similar to Multiprogramming batch systems.

- *A method or technique which enables many persons, located at different terminals, to use a particular computer system at the same time (अर्थात् कई लोग जो अलग-अलग टर्मिनल पर बैठे है, एक ही समय पर किसी कम्प्यूटर सिस्टम का प्रयोग कर सकें)।*
- *Can be thought as a logical extension of multiprogramming.*
- *Processor's time which is shared among multiple users simultaneously is called time-sharing.*
- *Main difference is that in case of Multiprogrammed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.*
- *Multiple jobs are executed by the CPU by switching between them, but the switching process occur so frequently. Thus, the user can receives an immediate response.*
- *Processor executes each user program in a short burst or quantum of computation. If many users are present, each user gets his time quantum. When the user submits the command, the response time is in few seconds at most.*
- *Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time.*

Advantages—

- *Quick response.*
- *Avoids duplication of software.*
- *Reduces CPU idle time.*

Disadvantages—

- *Not very reliable*
- *Insecurity and integrity problems of user programs and data.*
- *Data communication problems.*

Distributed Operating System—

Distributed operating system में users दूर वाले resources को भी उसी प्रकार use कर सकते हैं, जैसे कि local resources को करते हैं।

Distributed system में कई सारे यूज़र तथा कई सारे Real time applications (अनुप्रयोग) हेतु कई सारे केन्द्रीय प्रौसेसर का प्रयोग किया जाता है। इसमें विभिन्न प्रौसेसर जॉब्स को विभिन्न प्रौसेसर के मध्य बाँट दिया जाता है तथा जो प्रौसेसर जॉब ज्यादा अच्छी तरह से तथा दक्षतापूर्वक (efficiently) कर सकता है उसको वह जॉब आवंटित किया जाता है। यह सभी प्रौसेसर एक दूसरे से विभिन्न संचार लाइनों के माध्यम से सम्पर्क कर सकते हैं। जैसे—उच्च स्पीड वाली बस (high speed bus), टेलीफोन लाइन्स इत्यादि। इस प्रकार के सिस्टम loosely coupled systems या distributed systems कहलाते हैं।

Distributed systems के processes का आकार तथा कार्य अलग-अलग हो सकते हैं।

Distributed systems के लाभ—

- विभिन्न स्रोतों को share करने के कारण एक site का यूजर दूसरे site पर उपलब्ध स्रोतों का प्रयोग कर सकते है।
- यदि कोई site fail हो जाती है तो भी सिस्टम ऑपरेट कर सकता है।
- उपभोक्ताओं को बेहतर सेवा प्रदान की जा सकती है।
- Host computer पर load का कम होना।
- डाटा प्रौसेसिंग में कम समय लगना (Reduction of delays in data processing)।

The main aim behind developing distributed operating systems is the availability of powerful and inexpensive microprocessors and advances in communication technology. These advancements in technology have made it possible to design and develop distributed systems comprising of many computers that are inter connected by communication networks. The main benefit of distributed systems is its low price/performance ratio. Distributed systems use multiple central processors to serve multiple real time application and multiple users. Data processing jobs are distributed among the processors accordingly to which one can perform each job in the most efficient manner. The processors communicate with one another through various communication lines. These are called loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function and are referred to as sites, nodes, computers etc.

Advantages—

- *User at one site may be able to use the resources available at another using resource sharing facility.*
- *Speedup the exchange of data using E-mails.*
- *Remaining sites can potentially continue operating in case of failure of any site.*
- *Better customer services.*
- *Reduction of delays in data processing.*
- *Reduction of the load on the host computer.*

Network Operating System—

Network operating system में सभी computers एक network (जाल) के रूप में आपस में connected रहते हैं, जिससे कि प्रत्येक workstation का अपना computer होता है। Computers के Network के रूप में connect होने के कारण यह Printers, Hard disk drives programs इत्यादि share कर सकते हैं। User अपना work इस network के किसी भी computer से access कर सकते हैं।

Network operating system सर्वर पर काम करते हैं और सर्वर को डेटा को मैनेज करने users, groups, security, applications एवं अन्य networking function का सामर्थ्य प्रदान करते हैं।

नेटवर्क ऑपरेटिंग सिस्टम का मुख्य उद्देश्य कई कम्प्यूटर्स के मध्य फाइल्स तथा प्रिन्टर्स की पहुँच बनाना है जो कि किसी लोकल एरिया नेटवर्क (LAN), प्राइवेट नेटवर्क या अन्य नेटवर्क से जुड़े है।

नेटवर्क ऑपरेटिंग सिस्टम के मुख्य उदाहरण है—माइक्रोसाफ्ट विन्डो सर्वर 2003, माइक्रोसाफ्ट सर्वर 2008, यूनिक्स, लिनिक्स, मैक OS X, नावेल नेटवर्क तथा BSD

नेटवर्क ऑपरेटिंग सिस्टम के मुख्य लाभ है—

- हार्डवेयर शेयरिंग की सुविधा अर्थात् Printers, Scanners इत्यादि को शेयर करना संभव हो जाता है।
- सेन्ट्रलाइज्ड केन्द्रीकृत सर्वर की स्थिरता अधिक होती है।
- सिक्योरिटी अधिक होती है तथ सर्वर द्वारा नियान्त्रत होती है।

- नयी टेक्नालाजी से upgradation आसान हो जाता है।
- दूर स्थिर सर्वर तक पहुच बनायी जा सकती है।

किन्तु इसकी कुछ सीमायें भी है—

- सर्वर को खरीदने और run करने में काफी कॉस्ट (खर्च) आता है।
- सभी operations सेन्टर लोकेशन पर निर्भर हो जाने है।
- लगातार मेन्टिनेस (maintenance) तथा updation की ज़रूरत होती है।

Network operating system (NOS) is designed to support workstations, personal computer, that are connected on a local area network (LAN).Network Operating System runs on a server and and provides server the capability to manage data, users, groups, security, applications, and other networking functions. A network operating system includes special functions for connecting computers and devices into a local-area network (LAN). Some operating systems, such as UNIX and the Mac OS, have networking functions built in.The term network operating system is generally reserved for software that enhances a basic operating system by adding networking features. Its main purpose is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of NOS are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

Advantages—

- *Sharing of hardware becomes possible.*
- *Use of highly stable Centralized servers.*
- *Server managed security.*
- *Easily integration of upgrades to new technologies and hardwares into the system.*
- *Remote access to servers possible from different locations and types of systems.*

Disadvantages—

- *High server cost (buying and running a server).*
- *increased dependency on a central location for most operations.*
- *Requirement of regular maintenance and updation*

Real Time Operating System—

Real time systems remote locations से random enquiries को accept करके तुरन्त response प्रदान करते हैं। Operating system कम्प्यूटर को data का विश्लेषण करने (analyse) और उत्तर देने हेतु निर्देशित करता है।

Real time operating system वह डॉटा प्रौसेसिंग system होता है जिसमें input को प्रौसेस व रिस्पॉन्ड करने में लगने वाला समय इतना कम होता है कि वह इनवायरमेन्ट को कन्ट्रोल करता है।

Real time processing सदैव online होती है जबकि online system का real time होना आवश्यक नहीं है। इस processing में online processing की तुलना में response time काफी कम होता है।

Real time system तब उपयोग किये जाते हैं जब प्रौसेसर के कार्य में समय आवश्यकताएँ महत्वपूर्ण होती है जैसे कि वैज्ञानिक प्रयोग, मैडिकल Imaging systems, Industrial control system, Weapon systems, रोबोट्स तथा Home appliances controller, Air traffic कन्ट्रोल system इत्यादि।

Real time operating system दो प्रकार के होते हैं—

(i) Hard real time system

(ii) Soft real time system

Hard Real Time System—यह गारन्टी देते हैं कि महत्वपूर्ण कार्य समय से पूर्ण हो जाये। Hard real time system में secondary storage बहुत कम होती है तथा Data ROM में स्टोर होता है। इस प्रकार के system में वर्चुअल मेमोरी का अभाव होता है।

Soft real time system कम स्ट्रिक्टिव (strictive) होते हैं। इनमें क्रिटिकल Real time task को दूसरे task पर प्राथमिकता प्राप्त रहती है।

Hard Real Time व Soft Real Time Systems में अंतर–

Hard Real Time Operating Systems	Soft Real Time Operating Systems
Hard real time systems में मुख्य गणना का समय से पूरा हो जाना सबसे ज्यादा महत्व रखता है अर्थात् यदि समय से पूरी नहीं हो पाई तो फिर उसका महत्व नहीं रह जाता था तथा उससे सिस्टम को हानि भी पहुँच सकती है। Example: Missile launch system	Soft real time system में deadline (अर्थात् निर्धारित समय में कार्य को पूरा करना) उतना महत्वपूर्ण नहीं होता जितना कि Hard real time system में होता है किन्तु कार्य पूरा करना तथा Result देना अधिक महत्व रखता है Example: Personal Computer, Audio and Video system इत्यादि

रियल टाइम ऑपरेटिंग सिस्टम की प्रक्रिया बहुत ही तीव्र गति से होती है रियल टाइम ऑपरेटिंग सिस्टम का उपयोग तब किया जाता है जब कम्प्यूटर के द्वारा किसी कार्य विशेष का नियंत्रण किया जा रहा होता है। इस प्रकार के प्रयोग का परिणाम तुरंत प्राप्त होता है और इस परिणाम को अपनी गणना में तुरंत प्रयोग में लाया जाता है। आवश्यकता पड़ने पर नियंत्रित की जाने वाली प्रक्रिया को बदला जा सकता है। इस तकनीक के द्वारा कम्पयुटर का कार्य लगातार आंकड़े ग्रहण करना, उनकी गणना करना, मेमोरी में उन्हें व्यवस्थित करना तथा गणना के परिणाम के आधार पर निर्देश देना है।

A real time system accepts random enquiries from remote locations and provide instantaneous response. The operating system instructs the computer to analyse the data and send appropriate answer. They are dynamic in nature as they accept random input. Real time operating system is defined as an operating system known to give maximum time for each of the critical operations that it performs, like OS calls and interrupt handling. The Operating system which guarantees the maximum time for these operations are called hard real-time OS, while operating systems that can only guarantee a maximum of the time are called soft real-time OS. It is defined as a data processing system in which the time interval required to process and respond to inputs is very small that it controls the environment. Real time processing is always online whereas the vice versa is not true i.e. online system need not be real time. The time taken by the system to respond to an input and display of required updated information is known as response time. So in this method, response time is very less as compared to the online processing. RTOS are used to control machinery, scientific instruments, and industrial systems. In general, the user does not have much control over the functions performed by the RTOS. Real time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real time systems can be used as a control device in a dedicated application. Real-time operating system has well defined, fixed time constraints otherwise system will fail for example: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-appliance controllers, Air traffic control system etc.

Real-time operating systems are of two types—

***Hard Real Time Systems**—As the name suggests, a hard real-time system (also known as an immediate real-time system) is hardware or software that must operate within the confines of a stringent deadline. Hard real-time systems guarantee that critical tasks complete on time. The application may be considered to have failed if it does not complete its function within the allotted time span. In hard real time systems, secondary storage is limited or missing with data stored in ROM. In these systems virtual memory is almost never found. Examples of hard real-time systems include components of pacemakers, anti-lock brakes and*

aircraft control systems. Hard real-time processing fails if not completed within a specified deadline relative to an event; deadlines must always be met, regardless of system load.

***Soft Real Time Systems**—Soft real time systems are less restrictive. The task and its deadline is manageable. Critical real time task gets priority over other tasks and retains the priority until it completes. Soft realtime systems have limited utility than hard real-time systems. For example, Multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, camera and washing machine etc.*

Things to know—

- *Operating system provides services to both the users and to the programs. It provides programs, an environment to execute and provides users, services to execute the programs in a convenient manner.*
- *Some common services provided by operating systems are Program execution, I/O operations Communication, Error Detection, Resource Allocation, Protection, File System manipulation etc.*
- *Program execution includes handling many kinds of activities from user programs to system programs like printer spooler, name servers, file server etc. Each of these activities is encapsulated as a process.*
- *Program management includes loading a program into memory, executing the program, handling program's execution, providing a mechanism for process synchronization, providing a mechanism for process communication, providing a mechanism for deadlock handling etc.*
- *I/O subsystem comprised of I/O devices and their corresponding driver software. The duty of drivers is to hide the peculiarities of specific hardware devices from the user as the device driver knows the peculiarities of the specific device. Operating System manages the communication between user and device drivers. I/O operation means read or write operation with any file or any specific I/O device. Program may require any I/O device while running. Operating system provides the access to the required I/O device when required.*

§ 2.5. प्रचलित ऑपरेटिंग सिस्टम्स (Popular Operating Systems) :

MS Windows—Microsoft Windows (जिसे MS Windows भी कहा जाता है) एक GUI (Graphical User Interface) operating system है, जिसे microsoft द्वारा विकसित किया गया। इसमें आप icons (आइकन अर्थात् छोटे-छोटे ग्राफिकल चित्रों) पर क्लिक करके प्रोग्रामों को क्रियान्वित कर सकते हैं। इसमें Windows icons व menus की सहायता से commands दी जा सकती हैं। कम्प्यूटर को बूट करने से लेकर एप्लीकेशन साफ्टवेयर प्रयोग करने जैसे सभी कार्य विन्डोज़ द्वारा ही किये जाते हैं। इंटरनैट का प्रयोग भी विन्डोज़ माहौल (Windows environment) से सर्वाधिक है।

MS DOS—MS DOS अर्थात् Microsoft disk operating system. यह एक CUI (Character User Interface) operating system है। यह single tasking OS है। MS DOS की core में तीन files, IO.SYS, MSDOS.SYS तथा COMMAND.COM होती हैं।

UNIX—UNIX सिस्टम 1969 में minicomputers हेतु Ken Thompson, R. Canaday व D. Ritchie द्वारा AT and T की Bell Laboratories (USA) में विकसित किया गया। यह एक multiple process operating system है। यह workstation के बजाय server के रूप में ज्यादा उपयुक्त है तथा system की जानकारी न रखने वालों द्वारा use नहीं किया जा सकता। यह समझने में थोड़ा मुश्किल होता है। UNIX की प्रमुख विशेषता है—Networking UNIX की सहायता से विभिन्न equipments files को share कर सकते हैं।

LINUX—Linux या Boss linux (BOSS stands for Bharat operating system solutions) Unix के समान होता है। यह Open Office Suite का derivative है जो सामान्यत: use किये जाने वाले विभिन्न प्रकार के documents व open standard file types को support करता है। यह भारतीय भाषाओं हेतु design किया गया है (हालांकि installers व defaults

Apple Macintosh—यह केवल Apple (कम्पनी) द्वारा निर्मित hardware पर ही कार्य कर सकता है। यह UNIX आधारित है, किन्तु graphical interface भी प्रदान करता है। इसकी दो मुख्य files है—System file तथा finder. System file user interface को manage करती है। Operating system के विभिन्न कार्यों जैसे—डिस्क की फॉरमेटिंग करना, files को copy करना, file को delete करना, application programs को रन करना इत्यादि को यह दोनों फाइल्स मिलकर सम्पन्न करती हैं। Macintosh का मुख्य दोष यह है कि यह DOS तथा Windows (PC) अनुप्रयोगों के compatable (समरूप) नहीं है।

Solaris—यह operating system 1992 में Sun Microsystem द्वारा विकसित किया गया। यह Unix आधारित operating system है।

मोबाइल ऑपरेटिंग सिस्टम्स (Mobile OS)—मोबाइल. फोन्स के emerge (आविर्भाव) होने के बाद कई प्रकार के Mobile operating platforms विकसित हुए। Smart phones में प्रयुक्त OS PCs में प्रयुक्त OS से भिन्न होते हैं क्योंकि Smart phones छोटे होते हैं, इनकी लगातार तथा frequently (प्राय:) कनैक्ट करने की आवश्यकता होती है (Wireless phone network से या दूसरी devices से)। स्मार्टफोन विविध प्रकार के होते हैं तथा ऑपरेटिंग सिस्टम ऐसा होना चाहिए जो किसी Third Party की आवश्यकताओं के अनुकूल हो। चूँकि स्मार्टफोन में लिमिटेड मैमोरी होती है। अत: operating system compact (छोटा) होना चाहिए। स्मार्ट फोन OS कई प्रकार के होते हैं जैसे कि RTOS (Real Time Operating System, Single user single tasking operating system, Single user multitasking operating system तथा Multi-user operating system) किसी मोबाइल OS की आवश्यकताएँ PC आवश्यकताओं से भिन्न होती है जैसे कि सीमित संसाधन (Limited Resources). For example—

CPU—220–370 MHz (ARM 9/ARM 11), 64-128 MB RAM), no hard disk and no virtual memory, सीमित बैट्री लाइफ, Power loss की स्थिति में डाटा safety, विश्वसनीयता इत्यादि।

- Mobile operating system is an operating system for smartphones, tablets, PDAs, or other mobile devices.
- Software that allows smartphones, tablet PCs and other devices to run applications and programs.
- Typically starts up when a device powers on, presenting a screen with icons or tiles that present information and provide application access.
- Also manage cellular and wireless network connectivity, as well as phone access.
- Combines features of a personal computer operating system with other features useful for mobile or handheld use; a touchscreen, cellular, Bluetooth, Wi-Fi, GPS mobile navigation, camera, video camera, speech recognition, voice recorder, music player, near field communication and infrared blaster.
- Examples are Apple iOS, Google Android, Motion's BlackBerry OS, Nokia's Symbian, Hewlett-Packard's webOS and Microsoft's Windows Phone OS.
- Users can jailbreak or root some devices, however, which allows them to install another mobile OS or unlock restricted applications.
- A smartphone operating system is essentially the engine that smoothly runs your smartphone
- Manages both the hardware and the software to create an enjoyable user experience.
- Examples of smartphone operating systems include Android, BlackBerry and Windows इत्यादि।

स्मार्ट फोन के उदाहरण निम्नवत् है–

➢ Google Android

यह iPhone OS से मिलता-जुलता है, इसमें Accelerometer, Application store, Open GL for Graphics, GPS इत्यादि utilities होती है। इसमें जावा भाषा का प्रयोग किया जाता है।

Google Android की मुख्य विशेषताएँ निम्नवत् है—

- Developed by the open Handset Alliance.
- Based on Linux
- Free licensing
- यह linux पर आधारित है तथा Sony Ericsson, Motorola, LG, Samsung इत्यादि में use किया जाता है।

➢ Symbian OS

यह कम कीमत तथा कम मैमोरी वाले स्मार्टफोन में प्रयुक्त होता है तथा इसलिए इसमें Web browsing, GPS utilities, Graphics उतनी अधिक उपलब्ध नहीं हो पाती।

➢ RIM Black Berry

सारांश (Summary)

एक ऑपरेटिंग सिस्टम कम्प्यूटर के यूज़र तथा कम्प्यूटर हार्डवेयर के मध्य मध्यस्थ का कार्य करता है। ऑपरेटिंग सिस्टम का उद्देश्य यह होता है कि एक ऐसा वातावरण प्रदान किया जाए जिससे यूज़र प्रोग्राम्स को सुविधापूर्वक तथा दक्षतापूर्वक कार्यान्वित कर सके। अत: ऑपरेटिंग सिस्टम एक ऐसा साफ्टवेयर होता है जो कम्प्यूटर हार्डवेयर का प्रबन्धन करता है।

कम्प्यूटर को स्टार्ट करते समय जब उसकी पावर ऑन की जाती है तो उसे रन करने हेतु एक प्रारम्भिक प्रोग्राम की आवश्यकता होती है जिसको बूट स्ट्रैप प्रोग्राम कहा जाता है। यह प्रोग्राम ROM या EEPROM में स्टोर होता है (जिसको कि firmware भी कहा जाता है)। यह सिस्टम के डिवाइसेस को प्रारम्भ करता है तथा इस प्रोग्राम को पता होता है कि ऑपरेटिंग सिस्टम को कैसे Load करना है तथा सिस्टम का कार्य कैसे शुरू करवाना है। इसके लिए Bootstrap program ऑपरेटिंग सिस्टम कर्नल को लोकेट तथा मैमोरी में Load करता है। उसके बाद ऑपरेटिंग सिस्टम पहले प्रौसेस को Execute (कार्यान्वित) करता है तथा Interrupt सिगनल के आने का wait करता है जिससे आगे के कार्य किये जाते हैं।

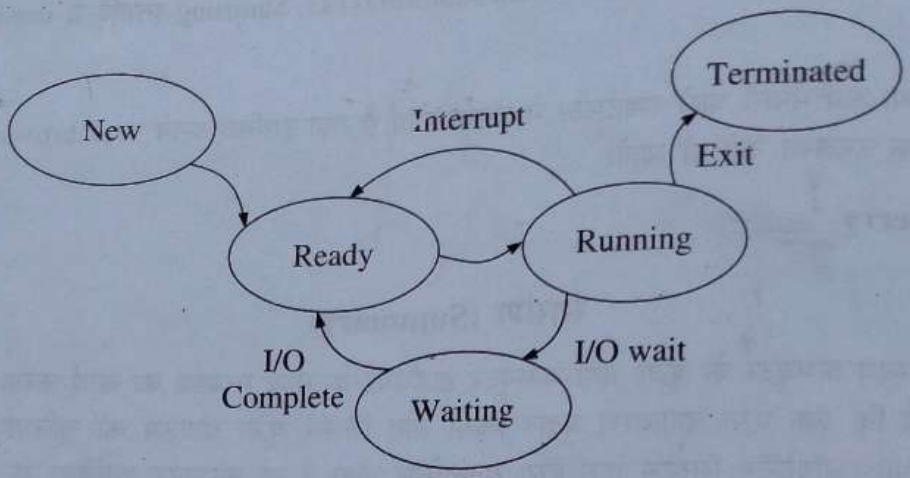
ऑपरेटिंग सिस्टम प्रोग्राम के Execution के लिए एक ऐसा वातावरण तैयार करता है जिसमें प्रोग्राम्स सुविधापूर्वक कार्यान्वित किये जा सकते हैं। इसके लिए ऑपरेटिंग सिस्टम विभिन्न प्रकार की सेवायें प्रदान करता है जैसे कि User Interface प्रदान करना, Program execution, I/O operation, File system manipulation, Communication, Error detection, Resource allocation, Accounting, Protection and Security इत्यादि।

यूज़र द्वारा ऑपरेटिंग सिस्टम से Interface करने की दो विधियाँ हैं—पहली तकनीक Command line interface या Command interpreter कहलाती है जिसमें यूज़र सीधे कमाण्ड्स को Keyboard द्वारा Enter करता है। दूसरी तकनीक GUI या Graphical User Interface कहलाती है जिसमें Desktop पर Icons होते हैं और माउस तथा Pointer की सहायता से कमाण्ड्स दिये जाते हैं।

जो प्रोग्राम execute हो रहा होता है उसे सामान्यत: प्रौसेस कहा जाता है। Process की Execution हेतु उसे Resources जैसे कि CPU time, Memory, files, I/O devices की आवश्यकता होती है। यह Resources प्रौसेस को Execution से पूर्व या Execution के समय Allocate किये जाते हैं। अधिकांश सिस्टम्स में Process को कार्य की इकाई (Unit of work) कहा जा सकता है तथा सिस्टम को प्रौसेसेस का समूह कहा जा सकता है। ऑपरेटिंग सिस्टम के प्रौसेसेस सिस्टम code execute करते हैं जबकि user processes user code execute करते हैं। यह दोनों processes एक साथ (concurrently) execute हो सकते हैं। Process तथा Job इन दोनों शब्दों को एक ही अर्थ में प्रयोग किया जाता है, एक Batch System jobs execute करता है जबकि एक Time Share System User Programs या Task execute करता है। हालांकि इन सबको प्रौसेस के नाम से ही जाना जाता है।

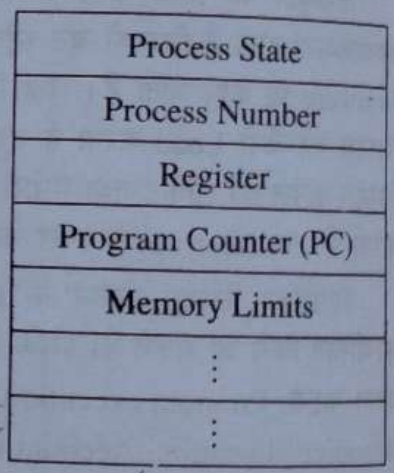
एक process में प्रोग्राम code के अतिरिक्त वर्तमान गतिविधि (जिसको कि प्रोग्राम counter तथा प्रौसेसर के रजिस्टर्स की वर्तमान स्थिति से व्यक्त किया जा सकता है), Stack, Data section इत्यादि हो सकते हैं।

जब Process execute होता है तो उसकी स्टेट (अवस्था) परिवर्तित होती है। स्टेट का तात्पर्य प्रौसेस के वर्तमान गतिविधि से है तथा किसी प्रौसेस की स्टेट New, Running, Waiting, Ready, Terminated में से कोई एक हो सकती है। (चित्र 2.4 देखें)



चित्र 2.4—Process states

ऑपरेटिंग सिस्टम में प्रत्येक प्रौसेस एक Process Control Block (PCB) द्वारा व्यक्त किया जाता है जिसको कि Task Control Block भी कहा जाता है (चित्र 2.5 देखें)। इसमें कई सूचनाएँ होती हैं जैसे कि Process state, Program counter, CPU registers, CPU scheduling information, Memory management information, Accounting information, I/O status information. अत: यह कहा जा सकता है कि प्रौसेस वह प्रोग्राम होता है जो Single Thread of Execution को Perform करता है।



चित्र 2.5—Process Control Block (PCB)

Process Scheduling—Multiprogramming का यह उद्देश्य होता है कि प्रत्येक समय में कोई ना कोई प्रौसेस Run करता रहे जिससे CPU का अधिकतम Utilization सम्भव हो सके। Time sharing का उद्देश्य यह होता है कि CPU को Process के बीच में तेजी से स्विच किया जाये जिससे Users Program से Interact कर सकें। इन उद्देश्यों को पूरा करने के लिए Process Scheduler अनेक उपलब्ध प्रौसेसेस में से किसी एक के CPU द्वारा Execution करने हेतु चयन करता है। Single Processor system में एक से अधिक Running process नहीं हो सकते तथा यदि प्रौसेसेस अधिक है तो दूसरों को CPU के free होने का इंतज़ार करना पड़ता है।

जैसे ही प्रौसेस system में enter करता है तो उसे एक Job Queue में डाल देते हैं इस Job Queue में system के सभी प्रौसेस होते हैं। वह प्रौसेसेस जो कि Main Memory में होते हैं तथा Ready (तैयार) होते हैं तथा Execute करने के लिए wait कर रहे होते हैं, Ready Queue में रखे जाते हैं और यह Queue एक Linked list में स्टोर होती है। System में कई अन्य Queues भी होते हैं। यदि किसी Process को CPU Allocate कर दिया जाता है तथा उसके बाद वह CPU को मुक्त कर देता है तथा यदि उसको किसी I/O डिवाइस की आवश्यकता होती है जो कि free नहीं है तो उसको डिवाइस Queue में रखा जाता है अर्थात् वह प्रौसेसेस जो कि किसी I/O डिवाइस के free होने का इंतजार कर रहे हैं, डिवाइस Queue कहलाते हैं। अत: प्रौसेसेस अपने जीवनकाल के दौरान कई शैड्यूलिंग Queues से गुजरता है तथा ऑपरेटिंग सिस्टम किसी ना किसी आधार पर इन प्रौसेसेस में से प्रौसेस को select करके उसे Resource आवंटित करता है। उपरोक्त का विस्तृत वर्णन आप आगे के अध्यायों में पढ़ेंगे।

प्रश्नावली

1. (a) Operating system से आप क्या समझते हैं?
   (b) Operating systems के विभिन्न कार्यों का वर्णन कीजिये।
   (c) ऑपरेटिंग सिस्टम के विकास पर टिप्पणी लिखिये।
   (d) ऑपरेटिंग सिस्टम के विभिन्न प्रकारों का वर्णन कीजिये।
   (e) निम्न को समझाइये—
   (i) Batch operating system (ii) Interactive operating system (iii) Time sharing operating system (iv) Real time system (v) Network operating system (vi) Distributed operating systems
2. Distributed व Time sharing OS में अंतर समझाइये।
3. Hard Real Time System व Soft Real Time System में अंतर समझाइये।
4. Process का तात्पर्य है—
   (a) Program in secondary memory
   (b) Program in execution
   (c) Program in disk
   (d) None of the above
5. Operating system किस प्रकार का software है—
   (a) Application software
   (b) Utility software
   (c) Operating software
   (d) System software
6. Fill in the blanks using the given words—
   server, database, e-commerce, e-mail, Internet service provider (ISP).
   (i) Computer that hosts various resources (including files, applications and databases) and places them at the disposal of all the devices connected to the network _____
   (ii) Group of data related to the same topic that is arranged in order and available for direct consultation by several users _____
   (iii) Sale or promotion of products and services over the Internet _____
   (iv) Service by which messages are exchanged between users of a computer network _____
   (v) Anyone can access the Internet from home through an _____

3 Chapter

फाइल सिस्टम (FILE SYSTEM)

THINK ABOUT IT

Build your own dreams, or someone else will hire you to build theirs. — **Farrah Gray**

In order to succeed, your desire for success should be greater than your fear of failure. — **Bill Cosby**

The difficulty lies not so much in developing new ideas as in escaping from old ones. — **John Maynard Keynes**

§ 3.1. परिचय (Introduction) :

यह तो आप जानते ही होंगे कि फाइल अपने creator (बनाने वाले) द्वारा परिभाषित सूचनाओं का समूह है अर्थात् A file is a collection of related information defined by its creator. कम्प्यूटर द्वारा files को disc पर store किया जाता है, जिससे इनकी long term storage (लम्बे समय तक संचय करना) प्राप्त की जा सकती है। Storage media के उदाहरण है—magnetic disc, optical disc इत्यादि। प्रत्येक media की अपनी अलग विशेषता होती है तथा इनके अभिलक्षण (speed, capacity data transfer rate, access methods) भिन्न होते हैं।

File systems को directories में संगठित किया जाता है। Directories के अंदर files या अन्य directories होती है। Operating system के file management संबंधी कार्य निम्नवत् है—

- Files को create तथा delete करना
- Directories को create तथा delete करना
- Files तथा directories के manipulation हेतु primitives को support करना
- Files को secondary storage पर map करना
- Files को stable storage media पर backup प्रदान करना

A file system is normally organized into directories for easy navigation and user friendly usage. These directories may contain files and other directories. The main responsibilities of an operating system with respect to file management are given below.

- *Provides an interface to the user to create/delete files.*
- *Provides an interface to the user to create/delete directories.*
- *Support of primitives to manipulate files and directories.*
- *Mapping of files to secondary storage.*
- *Back up of files on stable storage media.*

§ 3.2. फाइल (File) :

A file is a named collection of related information on a secondary storage अर्थात् सम्बंधित सूचनाओं के समूह को एक नाम देकर उसे store कर देना ही फाइल कहलाता है। यदि user की दृष्टि से देखा जाये तो file logical secondary storage का smallest allotment (सबसे छोटा आंवटन) है अर्थात् data को secondary storage पर तब तक नहीं लिखा जा सकता जब तक कि वह file में न हो। सामान्यत: फाइल प्रोग्राम्स (source form व object form में) तथा data दोनों को store करती हैं। data numeric, alphabetic, alphanumeric या binary हो सकता है। सामान्य तौर पर, file bits, bytes, lines व records का क्रम होती हैं, जिसका अर्थ file के creator (बनाने वाले) या user (इस्तेमाल करने वाले) द्वारा define किया जाता है। फाइल में विभिन्न प्रकार की सूचनायें अर्थात् source programs, object programs, executable programs, text, images, numeric and alphanumeric data, sound recordings, graphic images इत्यादि store किया जा सकता है। फाइल का एक निर्धारित स्ट्रक्चर (संरचना) होता है जो कि उसके type (प्रकार) पर निर्भर करता है। एक text file में characters का sequence होता है जो कि lines तथा pages के रूप में organised (व्यवस्थित) होता है। एक source file में subroutines तथा functions होते हैं तथा इसमें declarations तथा executable statements होतीं हैं। एक object file में bytes का sequence (क्रम) होता हैं जो कि blocks के रूप में organised होता है तथा system's linker द्वारा समझा जा सकता है। एक executable file में code sections का क्रम होता है जिसको कि loader द्वारा memory में लाकर execute किया जा सकता है।

A file is a user named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. Generally, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user. The file system structure is the basic level of organization in any operating system. The ways, how an operating system interacts with its users, applications, and security model are dependent upon the way it organizes files on storage devices. Providing a common file system structure makes sure (सुनिश्चित करता है) that users and programs are able to access and write files. A file system is the methods and data structures that an operating system uses to keep track of files on a disk or partition and the way the files are organized on the disk. File structure is a structure, which is according to a required format that operating system can understand—

- *A file has a certain well defined structure (संरचना) according to its type.*
- *A text file is a sequence of characters which is organized into lines.*
- *A source file is normally a sequence of procedures and functions.*
- *An object file is generally a sequence (क्रम) of bytes organized into blocks that are understandable by the machine.*
- *When operating system defines different file structures, it also contains the code to support these file structure. Note that Unix, MS-DOS support minimum number of file structure.*

§ 3.3. फाइल एट्रीब्यूट्स (File Attributes) :

Users की सुविधा के लिये file का नामकरण किया जाता है तथा यह नाम file name कहलाता है। यह name characters की string के रूप में होता है। File के मुख्य attributes (गुण) निम्नवत् है—

(i) **Name अर्थात् नाम**—File name अर्थात् file को जो नाम user रखता है वह फाइल नेम कहलाता है। User अपनी मर्ज़ी से इस नाम को रख सकता है, read कर सकता है (human readable) तथा फाइल की पहचान कर सकता है।

(ii) **Identifier**—यह एक unique संख्या के रूप में होता है जिसके द्वारा किसी file system के अंतर्गत file की पहचान की जाती है। User इस नाम को नहीं पढ़ सकता, अत: हम इसे non-human-readable file name कह सकते हैं।

(iii) **Type**—Type यानि प्रकार, चूंकि systems कई प्रकार की files को support करते हैं, अत:, file type की सूचना से यह पता लगाया जा सकता है कि file किस प्रकार की है।

(iv) **Location**—यह सूचना किसी device तथा उस device में file की location पर pointer के रूप में होती है।

(v) **Size**—File का आकार अर्थात् size (bytes, words या blocks में) तथा maximum allowed size इस attribute में अंतर्गत आता है।

(vi) **Time, date तथा user indentification**—इस attribute के अंतर्गत कई सूचनायें आती है जैसे कि file कब व किस समय बनी, उसको last बार कौन सी तिथि को व किस समय modify किया गया, अन्तिम बार कब व किस समय use किया गया, इत्यादि। यह data security, protection व उपयोग की monitoring के लिये useful होता है।

(vii) **Protection**—Protection का तात्पर्य सुरक्षा से है। Access control अर्थात् कौन-कौन इस फाइल तक पहुँच बना सकता है (अर्थात् file को read करने, write करने, execute करने इत्यादि का अधिकार किस-किस को प्राप्त है, यह सभी सूचनायें इस attribute के अंतर्गत आती है)।

फाइल संबंधी सभी सूचनायें directory structure (डायरैक्ट्री संरचना) में रखी जाती हैं तथा यह secondary storage में रहती हैं। सामान्यत:, directory entry के अंतर्गत file name तथा उसका identifier होता है, तथा यह identifier दूसरे file attributes को locate करता है।

The word attribute means qualities or characteristics. The main file attributes that different operating systems keep track are:

- **Name**—*This is the only information in human readable form. Some systems give special significance to names, and particularly extensions (.exe, .txt, etc.), and some do not. Some extensions may be of significance to the OS (.exe), and others only to certain applications (.jpg)*
- **Identifier**—*(e.g. inode number) is a unique tag, identifiers the file within file system and it is a number.*
- **Type**—*Text, executable, other binary, etc. (to support different types of files).*
- **Location**—*On the hard drive (Pointer to the device and location of file on the device).*
- **Size** *(in bytes, words or blocks)*
- **Protection** *(Access control information)*
- **Time & Date** *(Useful for protection, security and usage monitoring)*

§ 3.4. फाइल आपरेशन्स (File Operations) :

File एक abstract डेटा टाइप है। Operating system file संबंधी विभिन्न operations जैसे कि create a file (फाइल उत्पन्न करना), write a file (file पर write करना), read a file (file को read करना), reposition within a file, deleting a file (file को delete करना), Truncate a file (file की contents को erase करना किन्तु उसके attributes को बनाये रखना) इत्यादि हेतु system calls उत्पन्न करता है। आइये देखतें है कि उपरोक्त file operations करने के लिये operating system को क्या-क्या करना पड़ता है।

Creating a File—File को create (अर्थात् उत्पन्न) करने हेतु दो steps (पद) आवश्यक होते हैं—(a) file system में file के लिये space (स्थान) की तलाश करना तथा (b) directory में इस नयी file की entry करना।

Writing a File—File पर write करने के लिये एक सिस्टम call करनी होती है, जिसमें file का name व file पर write करने वाली information specified होती है। फाइल के नाम के अनुसार सिस्टम directory को search करके file की location find करता है। File की जिस लोकेशन पर अगला write किया जाना है, उस location पर system एक write pointer रखता है। जब भी कोई write operation perform किया जाता है, तो write pointer को update किया जाता है।

फाइल को Read करना (Reading a File)—File से read करने के लिये system call का use किया जाता है जिसमें फाइल का name specified होता है तथा यह भी specify किया जाता है कि file का अगला ब्लॉक memory में कहाँ रखा जाएगा। Directory को search किया जाता है तथा एक read pointer उस location की ओर point करता है जहाँ से अगली read होनी है। Read होने के बाद read pointer update होता है। चूंकि प्रौसेस सामान्यत: file को read या write कर रहा होता है, अत: एक

current file position pointer का यूज़ करके फाइल की current operation location को अपडेट रखा जा सकता है तथा read तथा write दोनों आपरेशन इस प्वांइटर को यूज़ कर सकते हैं जिससे स्पेस भी बचती है और जटिलता भी कम होती है।

Repositioning within a Files—Directory को file हेतु search किया जाता है तथा current file position pointer को दी गयी value पर reposition (दूसरे स्थान पर ले जाना) किया जाता है। इस operation को file seek operation कहा जाता है।

Deleting a File—फाइल को delete करने के लिए directory में file को search किया जाता है और उस फाइल स्पेस को मुक्त कर दिया जाता है ताकि उसको दूसरे यूज़र यूज़ कर सके। Directory entry को erase कर दिया जाता है।

Truncating a File—Truncating का तात्पर्य यह है कि file के contents को erase कर देना किन्तु उसके attributes को बचा के रख देना। अत: यूज़र को उसे फिर से उत्पन्न करने की आवश्यकता नहीं पड़ती। अत: इस function से सब attributes (file length को छोड़कर) unchange रहते हैं। File length शून्य पर reset हो जाती है तथा file के स्पेस को मुक्त कर दिया जाता है।

Main operations related to a file are:

- *Creating a file*
- *Writing a file*
- *Reading a file*
- *Deleting a file*
- *Truncating a file*
- *Repositioning a file*

उपरोक्त छ: operations के अलावा कई अन्य operations भी files पर किये जा सकते हैं जैसे कि appending करना (अर्थात् किसी file के अंत में new operation add करना), किसी फाइल को rename करना (अर्थात् फाइल का नाम बदलकर नया नाम रखना) इत्यादि।

§ 3.5. फाइल टाइप्स (File Types) :

सामान्यत: file name को दो भागों में विभक्त किया जाता है—name and extension. इस प्रकार user तथा operating system केवल नाम से ही पता लगा सकते हैं कि file कौन से प्रकार की है उदाहरणत: file name rajeev. doc में rajeev इस फाइल का नाम है तथा doc extension है जो कि यह बताता है कि यह file document (text) file है। इस प्रकार extension से फाइल के type का पता चल जाता है तथा यह भी पता चल जाता है कि फाइल पर कौन से operations किये जा सकते हैं, उदाहरणत: केवल .com, .exe व .bat extension वाली फाइल्स execute की जा सकती हैं। शेष प्रकार की फाइल्स का execution सम्भव नहीं है। .com व .exe binary executable files होती हैं। इसी प्रकार .bat extension वाली फाइल batch files होती हैं जिसमें ASCII format में operating systems हेतु command होती हैं।

विभिन्न फाइल Types (Extension के आधार पर)

Extension	Types of Files	Function
.exe, .com, .bin या कोई नहीं	Executable file	Read to run machine language program
.obj	Object file	Compiled, machine language, not linked
.bat	Batch file	Commands to command interpreter
.doc, tex	Text file	Contains documents
.jpg, .bmp	Image	Contain pictures
.pdf	Print or view	ASCII or binary file in a format for printing or viewing
.mpeg, .mp3	Multimedia	Containing audio and video information

File type means to the ability of the operating system to distinguish different types of file such as text files, source files, binary files etc. Operating system like MS-DOS and UNIX have the following types of files:

Ordinary Files (साधारण फाइल्स)

- These are the files that contain user information (यूज़र इनफॉरमेशन).
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the file.

Directory Files (डायरैक्ट्री फाइल्स)

- These files contain list of file names and other information related to these files.(फाइल्स के नामों की सूची)

Special Files (विशेष फाइल्स or Device फाइल्स)

- These represent physical device like disks, terminals, printers, networks, tape drive etc. They may be of following types:
- **Character Special Files**—In these files, data is handled character by character as in case of terminals or printers.
- **Block Special Files**—In these files, data is handled in blocks as in the case of disks and tapes.

§ 3.6. फाइल एक्सेस विधियाँ (File Access Methods) :

जैसा कि आप जानते हैं कि फाइल information को store करती हैं। जब इस information की आवश्यकता होती है तो file को access किया जाता है तथा computer memory में read किया जाता है अर्थात् files secondary storage में होती हैं तथा इनको इस्तेमाल करने के लिये इन्हे प्राइमरी मैमोरी में लाना पड़ता है। फाइल में stored सूचना को access करने (पहुँच बनाने) की कई विधियाँ होती है। कुछ सिस्टम्स file access करने हेतु केवल एक विधि प्रदान करते हैं जबकि कुछ सिस्टम्स में files को कई विधियों द्वारा access किया जा सकता है तथा सिस्टम design के समय यह निर्धारित किया जाता है कि किसी विशेष application हेतु file access की कौन सी विधि use की जायेगी।

फाइल एक्सैस की मुख्य विधियाँ हैं—Sequential access विधि, Direct या रैन्डम एक्सैस विधि तथा Indexed sequential access विधि

File access mechanism or methods means the way in which the records of a file may be accessed. Information is kept in files. Files reside on secondary storage. When this information is to be used, it has to be accessed and brought into primary main memory. Information in files could be accessed in different ways. It is normally dependent on an application that which method will be used. There are many ways to access files i.e.

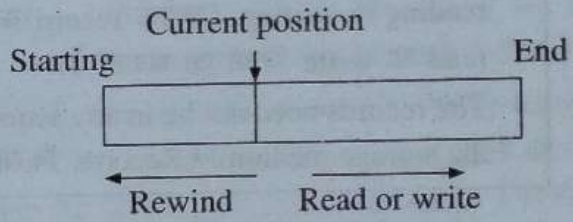
- *Sequential access,*
- *Direct/Random access,*
- *Indexed sequential access etc.*

सीक्वैंशल एक्सैस (Sequential Access):

Sequential access विधि file access की सबसे सरल विधि है। इस विधि द्वारा file में stored informations क्रमानुसार (in sequence, in order) access की जाती है (अर्थात् पहला record, फिर दूसरा रिकार्ड, फिर तीसरा record

... इत्यादि)। उदाहरणत:, एक read operation (read next) फाइल के आगे भाग को read करता है और फाइल प्वाइंटर को स्वत: advance (next क्रम पर point करना) कर देता है। इसी प्रकार write operation (write next) फाइल के end में append करता है, और नये written matter के end पर (अर्थात् file के नये end पर) advance हो जाता है।

In sequential access method, the records are accessed in some sequence or order (क्रम से). The information in the file is processed in order, one record after the other. This access method is the most primitive one. Information in a file is accessed sequentially one record after another. Sequential access is based on the tape model that is inherently a sequential access device. Sequential access is best suited where most of the records in a file are to be processed. For example, transaction files. Example: Compilers usually access files in this fashion.



चित्र 3.1—Sequentially accessed file

डायरैक्ट ऐक्सैस विधि (Direct Access Method)—

डायरैक्ट ऐक्सैस विधि (direct access method or relative access method) में फाइल से read करने या फाइल में write करने हेतु किसी क्रम की आवश्यकता नहीं होती। डिस्क ऐक्सैस विधि file के डिस्क मॉडल पर आधारित होती है क्योंकि डिस्क में किसी भी file block को randomly (अर्थात् बिना किसी क्रम के) access किया जा सकता है। डायरैक्ट ऐक्सैस हेतु file को हम blocks या records के numbered sequence के रूप में देख सकते हैं तथा बिना किसी क्रम के इन blocks से read या block में write कर सकते हैं उदाहरणत: पहले block 12 को read किया, फिर block 23 पर write किया, फिर block 10 से read किया इत्यादि यानि कोई निर्धारित क्रम की आवश्यकता नहीं होती।

यदि बड़ी मात्रा में information पर त्वरित पहुँच (immediate access) बनानी हो, तो direct access विधि useful रहती है। डेटाबेस में अक्सर डायरैक्ट ऐक्सैस विधि का use किया जाता है। माना कि कोई querry आती है तो उस querry (प्रश्न) से संबंधित blocks को access करके वांछित सूचना प्राप्त की जा सकती है।

डायरैक्ट ऐक्सैस विधि हेतु file operations में block number को parameter के रूप में use किया जाता है (अर्थात् read *n*, write *n* इत्यादि जहां *n* block number को इंगित करता है)

सामान्यत:, user द्वारा operating system को relative block number प्रदान किया जाता है जो कि फाइल की शुरुआत के सापेक्ष (relative to the beginning of the file) इंडैक्स को व्यक्त करता है।

अत: file का पहला relative block 0 होता है, अगला 1 होता है, इत्यादि। जबकि block का actual absolute disk address इससे भिन्न हो सकता है उदाहरणत: पहले block के लिये 2513, दूसरे block के लिये 1215 इत्यादि। Relative block address से operating सिस्टम यह निर्धारित कर सकता है कि file को कहां रखना चाहिये, जिससे user file system के उन हिस्सों तक पहुँच न बना सके तो उसकी file से संबंधित नहीं है।

Direct access is used in cases, when it is not necessary to process every record in a file. Information present in a record of a file is to be accessed only if some key value in that record is known. Direct access is based on the disk that is a direct access device and allows random access of any file block. Since a file is a collection of physical blocks, any block and hence the records in that block can be accessed. Databases are often of direct access type since they allow querry processing that involves immediate access to large amounts of information. All reservation systems use this method. Not all operating systems support direct access files. Sequential access of a direct access file is possible but direct access of a sequential file is not.

- Random access file organization provides accessing the records directly.(Records को directly access किया जा सकता है)
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing. (प्रत्येक record का अपना address होता है जिसकी सहायता से उस डायरैक्ट access करके read या write किया जा सकता है)
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium. (Records किसी भी क्रम में हो सकते हैं)

अन्य ऐक्सैस विधियाँ (Other Access Methods) :

डायरैक्ट ऐक्सैस विधि की मदद लेकर अन्य access विधियां बनायी जा सकती हैं—जैसे कि indexed sequential access। इन विधियों में file के लिये index बनायी जाती है। इस index में विभिन्न blocks के pointers होते हैं अर्थात् file में किसी record को ढूढ़ने हेतु पहले index search की जाती है तथा फिर pointer का use करके file को access किया जाता है और इस प्रकार वांछित record find किया जा सकता है।

- Combination of both the sequential access as well as direct access. (Sequential व डायेरैक्ट access का संयुक्त रूप)
- Access a file direct first and then sequentially from that point onwards. (पहले direct access, तत्पश्चात् sequential access)
- Involves maintaining an index (Index maintain की जाती है)
- Index is a pointer to a block (Index block की ओर pointer होती है)
- To access a record in a file, a direct access of the index is made. The information obtained from this access is used to access the file. For example, the direct access to a file will give the block address and within the block the record is accessed sequentially.The main advantage in this type of access is that both direct and sequential access of files is possible.This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

महत्वपूर्ण प्रश्न:

File access methods क्या हैं? प्रमुख file access methods का वर्णन कीजिये।

उत्तर—File से सूचना प्राप्त करने हेतु प्रयुक्त विधियों को file access methods कहा जाता है। प्रमुख file access methods निम्नवत् है—

(i) **Sequential Access:** यदि फाइल की सूचना क्रमवार process की जाये तो इसे sequential access कहते हैं।

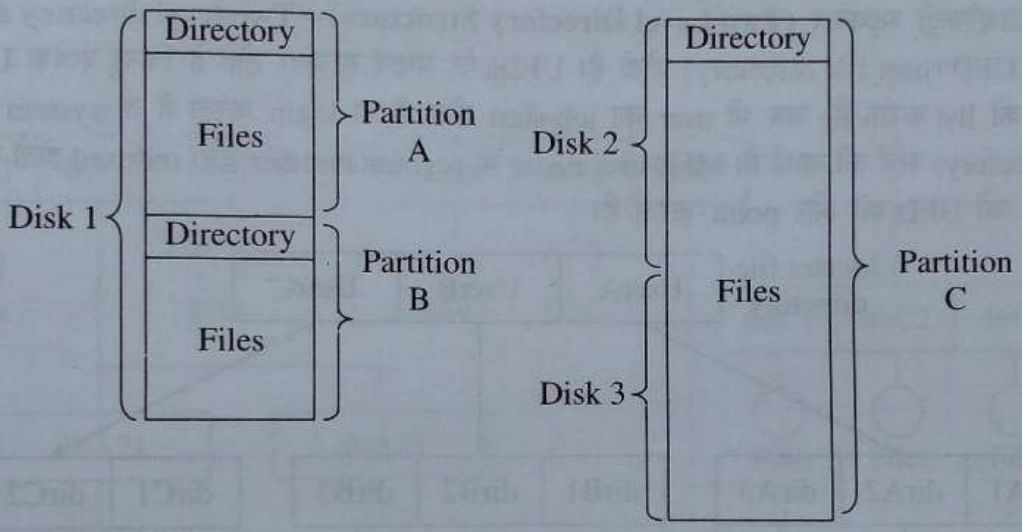
(ii) **Direct Access:** इसमें file fixed length logical records का बना हुआ माना जाता है तथा read/write बिना किसी क्रम के की जा सकती है।

(iii) **Indexed Access:** इसमें विभिन्न blocks के pointers होते हैं (जैसे कि book के back page पर index होती है)। File के किसी record को find करने हेतु पहले index की सर्च की जाती है तथा फिर pointer की सहायता से file को directly access किया जा सकता है तथा वांछित record तक पहुँचा जा सकता है।

§ 3.7. डायरैक्ट्री संरचना (Directory Structure) :

चूंकि computers में file system व्यापक (extensive) होता है तथा बड़ी मात्रा में डेटा स्टोर किया जाता है, अत: इस डेटा के उपयुक्त प्रबंधन हेतु फाइल्स को व्यवस्थित किया जाना आवश्यक है। इसके लिये directories का use किया जाता है।

किसी भी storage system (जैसे कि डिस्क) को file system हेतु use किया जा सकता है। यदि disk में कई file systems को store करना है तो उसके partitions (slices अर्थात् कई भाग) कर दिये जाते हैं और इन भागों में भिन्न-भिन्न file systems store किये जा सकते हैं। इन parts को combine करके बड़े स्ट्रक्चर्स बनाये जा सकते हैं जिनको volume कहा जाता है। प्रत्येक volume में file systems के साथ-साथ files के संबंध में सूचनायें भी होती हैं यह सूचनायें device directory (या volume table of contents) में entries के रूप में store की जाती है। अत: device directory (या संक्षेप में directory) उस volume से संबंधित सभी फाइल्स की सूचनाओं को record करती है (जैसे कि name, location, type, size इत्यादि) (चित्र 3.2)



चित्र 3.2—File System Organisation

अत: डायरैक्ट्रीज़ को एक प्रकार से symbol tables कहा जा सकता है जो file names को उनकी डायरैक्ट्री एन्ट्री में ट्रांसलेट करती है। अत:, डायरैक्ट्रीज़ को भी कई प्रकार से व्यवस्थित किया जा सकता है तथा कई प्रकार के operations किये जा सकते हैं जैसे कि insert entries, delete entries, किसी named entry को search (तलाश) करना, सभी entries को list करना आदि। अत: डायरैक्ट्रीज़ पर perform किये जाने वाले विभिन्न operations निम्नवत् हैं—

(i) फाइल उत्पन्न करना अर्थात् creating a files अर्थात् नई file create करके directory में add करना।

(ii) फाइल delete करना अर्थात् deleting a file अर्थात् जब फाइल की आवश्यकता न हो तो उसे directory से remove कर देना।

(iii) फाइल को सर्च करना अर्थात् searching a file, अर्थात् directory structure को search करने किसी विशेष नाम वाली file (या files) को find करना, या उन files को find करना जिनका नाम किसी विशेष pattern (प्रारूप) से मैच करता हो।

(iv) फाइल rename करना अर्थात् renaming a file अर्थात् फाइल का वर्तमान नाम चेंज करके उसका नया नाम रखना।

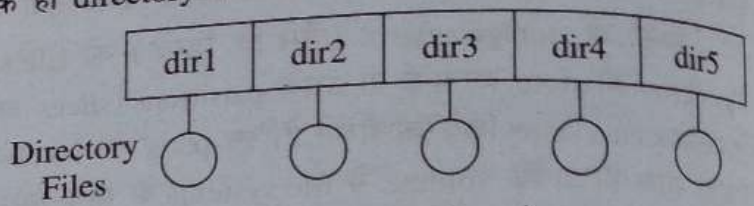
(v) डायरैक्ट्री को लिस्ट करना अर्थात् list a directory अर्थात् डायरैक्ट्री की फाइल्स को लिस्ट करना।

(vi) File system को traverse करना अर्थात् directory structure के अंतर्गत प्रत्येक डायरैक्ट्री व प्रत्येक फाइल को access करना।

विभिन्न प्रकार के डायरैक्ट्री स्ट्रक्चर—

(i) सिंगल लैवल डायरैक्ट्री (Single Level Directory)—Single level (अर्थात् एकल स्तर) सबसे simple directory structure है। इसमें सभी फाइल्स एक ही directory के अंतर्गत होती है।

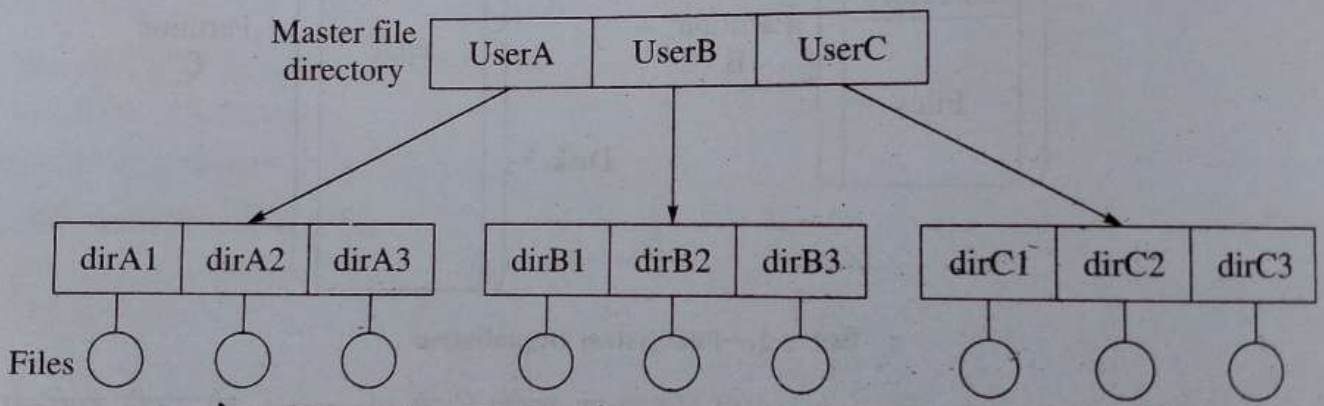
यद्यपि single level directory simple होती है किन्तु users की संख्या बढ़ने पर फाइल्स की संख्या बढ़ने पर या system में एक से अधिक users होने पर इसमें समस्यायें आती हैं। चूंकि सभी files एक ही directory में होती हैं अत: सबके नाम भी भिन्न होने चाहिये। यहां तक कि एक user के लिये भी इतने सारे file names को याद रखना अत्यंत मुश्किल है।



नोट—Directories तथा files के नाम user अपनी इच्छानुसार रख सकता है।

चित्र 3.3—सिंगल लैवल डायरैक्ट्री

टू-लैवल डायरैक्ट्री स्ट्रक्चर (Two Level Directory Structure)—Two level directory structure में प्रत्येक user की अपनी UFD (user file directory) होती है। UFDs के समान स्ट्रक्चर होते हैं किन्तु प्रत्येक UFD केवल single user की files को list करता है। जब भी user का job start होता है या login करता है तो system की MFD (अर्थात् master file directory) सर्च की जाती है। MFD user name या account number द्वारा indexed होती है तथा प्रत्येक entry उस विशेष user की UFD की ओर point करती है।



नोट—Directories तथा files के नाम user अपनी इच्छानुसार रख सकता है।

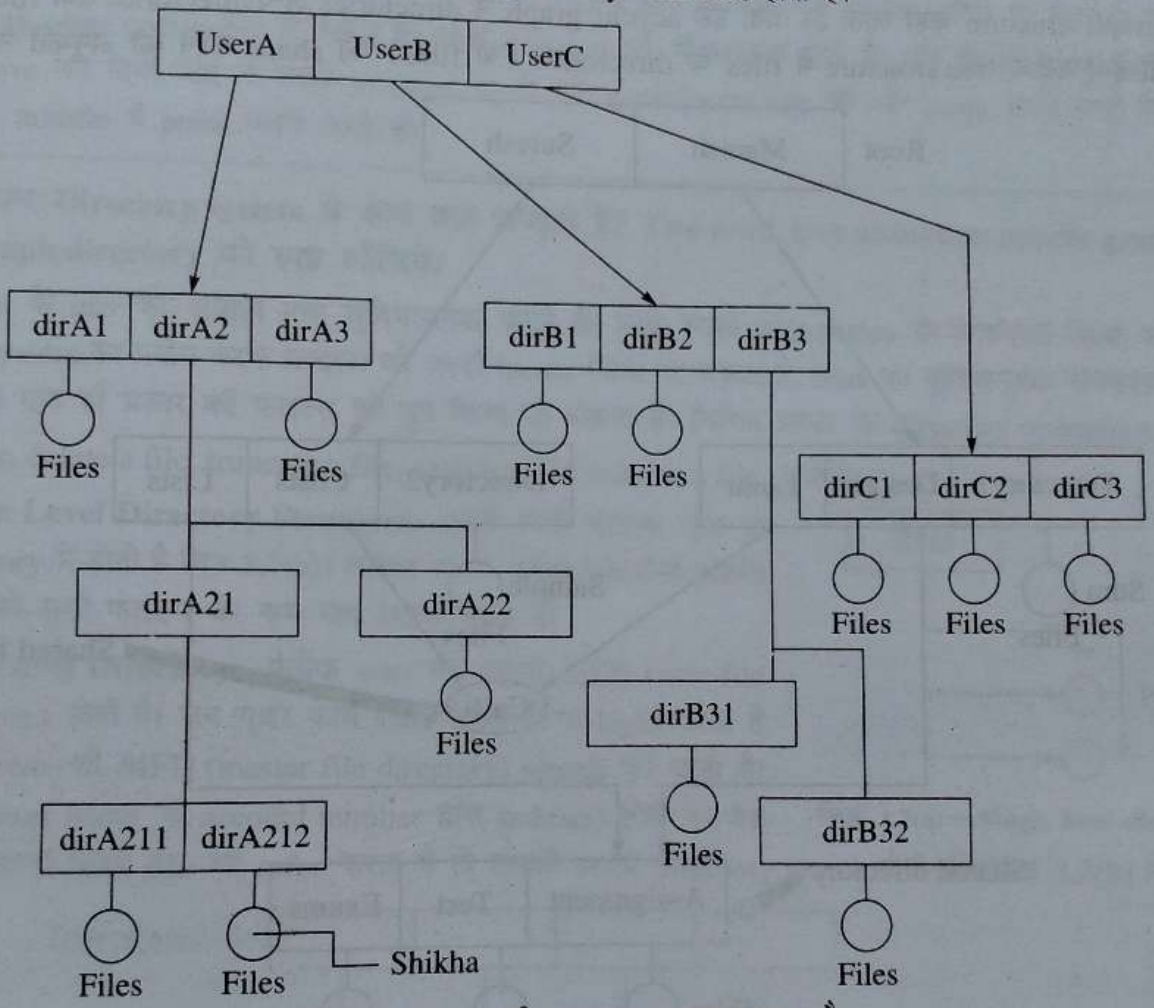
चित्र 3.4—Two level directory structure

जब user किसी file को refer करता है, तो केवल उसकी UFD ही सर्च की जाती है। अत:, विभिन्न users एक ही नाम की files create कर सकते हैं (जब तक की प्रत्येक UFD के अंदर के सभी file names भिन्न भिन्न हों)। अत: किसी user की file create करने हेतु operating system केवल उसी user की UFD search करके यह सुनिश्चित करता है कि उस नाम की दूसरी file तो exist नहीं करती। फाइल delete करने हेतु भी operating system अपनी सर्च केवल local UFD तक ही सीमित रखता है जिससे असावधानीवश उसी नाम वाली दूसरे user की file delete होने की संभावना नहीं रहती।

User की directory को create करने हेतु एक विशेष प्रोग्राम run किया जाता है जिसमें user name व account information होती है। यह प्रोग्राम नयी UFD create करता है और इसकी entry MFD में कर देता है। इस प्रोग्राम का execution केवल system administrators तक सीमित रहता है।

इस प्रकार हमने देखा कि Two level directory structure से name collision problem (अर्थात् एक ही नाम रखने वाली समस्या) समाप्त हो जाती है तथा users एक दूसरे से पृथक (isolated) रहते हैं। अत:, users का isolation users को independent (स्वतंत्र) कर देता है। किन्तु यदि users किसी task पर एक दूसरे को cooperate करना चाहते हैं तो यह संभव नहीं हो पाता।

Tree structure directory संरचना सबसे सामान्य directory संरचना है। एक two-level structure का विस्तार करके tree structure की संरचना की जा सकती है। इस प्रकार की संरचना में directory के अंदर subdirectories या files हो सकती है। अत: users अपनी subdirectories बनाकर files को organize कर सकता है। प्रत्येक tree की एक root directory होती है तथा system की प्रत्येक file का एक unique pathname होता है। सभी directories का same internal format (समान आन्तरिक प्रारूप) होता है। प्रत्येक directory में एक बिट यह define करने हेतु प्रयुक्त होती है कि entry file है या subdirectory (अर्थात् यदि यह बिट = 0 है तो entry file है तथा यदि यह bit = 1 है तो entry subdirectory है)। Directories को create या delete करने हेतु special system calls होती है।



नोट—Directories व Files के नाम user अपनी इच्छानुसार रख सकता है।

चित्र 3.5—Tree structure directory

सामान्य तौर पर, प्रत्येक process की एक current directory (वर्तमान डायरैक्ट्री) होती है। Current directory में अधिकांशत: वह files होती है जिसकी process को वर्तमान में आवश्यकता होती है। यदि किसी file को reference किया जाता है तो पहले उसे वर्तमान directory में search किया जाता है। यदि ऐसी फाइल को search करना है जो कि current directory में नहीं है तो या तो user को path name, specify करना पड़ता है या फिर current directory को change करके उस directory में move करना पड़ता है, जिसमे वह फाइल है।

Path name दो प्रकार के हो सकते हैं—Absolute path name तथा Relative path name.

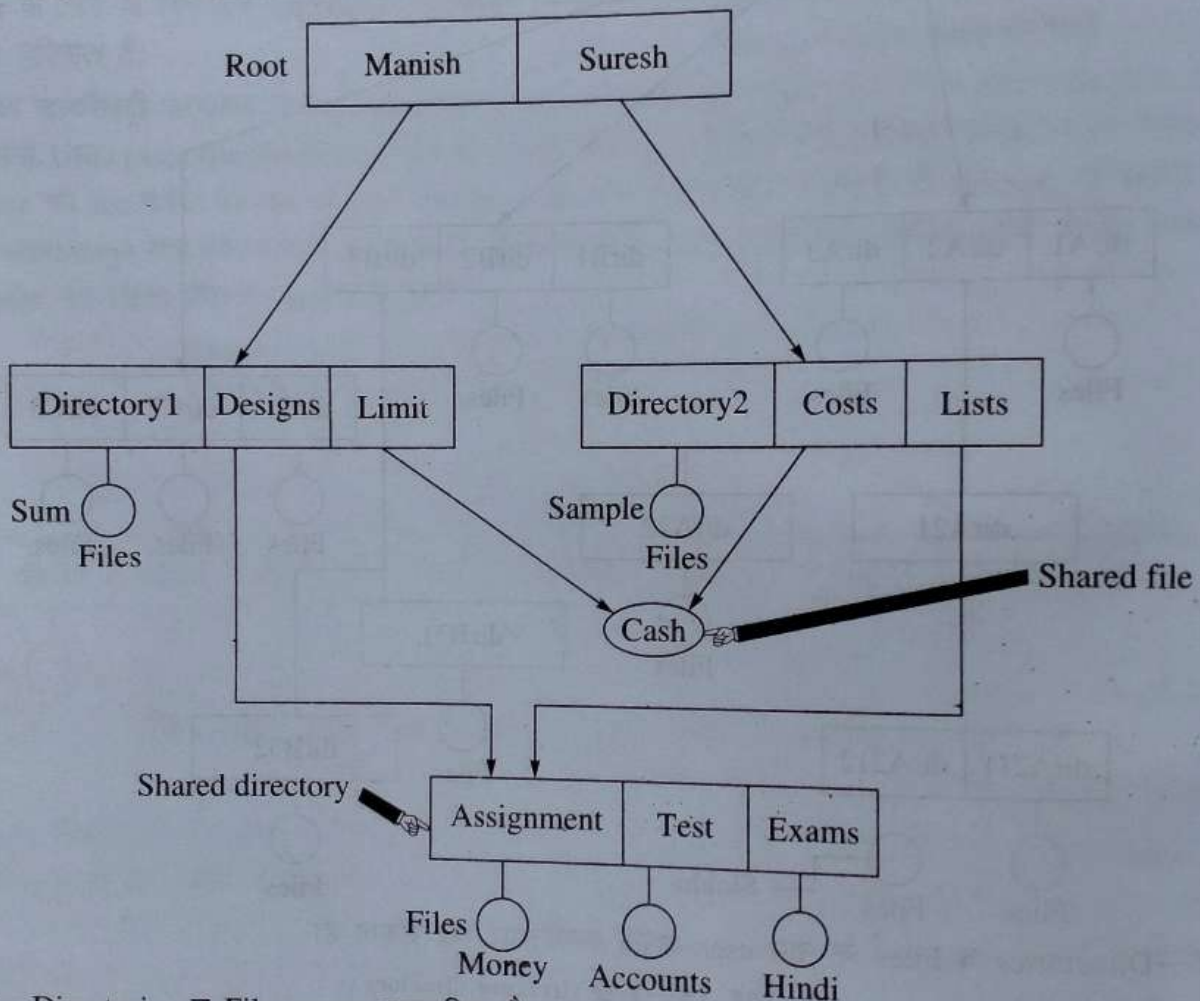
(i) Absolute path name root से शुरु होता है तथा path को follow करते हुए file तक पहुँचता है, तथा इस path name में root से file के मध्य की सभी sub directories शामिल होती हैं।

(ii) Relative path name current directory से वाँछित file तक का नाम विनिर्दिष्ट (specify) करता है।

उदाहरण—चित्र 3.5 को देखें। यदि वर्तमान directory dirA2 है तो relative path name—dirA21\dirA212\Shikha उसी file को refer करता है जिसको कि absolute path name—root\UserA\dirA2\dirA21\dirA212\Shikha refer करता है। कहने का तात्पर्य यह है कि absolute name root से start होकर directory तथा file तक लिखा जाता है जबकि relative path name में केवल current directory से आगे का path ही लिखा जाता है।

Acyclic-Graph Directories—

यदि दो प्रोग्रामर अलग-अलग directories पर काम करते हैं किन्तु किसी subdirectory को share करते हैं तो इसको acyclic graph structure कहा जाता है। अत: इस acyclic graph में directories अन्य directories तथा files को share कर सकती है (जबकि tree structure में files के directories (या files) को share करने की अनुमति नहीं होती है)



नोट—Directories व File names काल्पनिक है, user अपनी इच्छानुसार कोई भी नाम रख सकता है।

चित्र 3.6—Acyclic graph free structure

ध्यान दें कि shared file का अर्थ एक file की दो copies होना नहीं है क्योंकि दो copies होने की स्थिति में यदि यूज़र एक file में change करेगा तो वह changes दूसरे file में दिखाई नहीं देंगी। Shared file में केवल एक file अस्तित्व में रहती है और यदि एक व्यक्ति file में change करता है तो दूसरे को वह दिखाई देती हैं।

इस प्रकार के directories structure उस स्थिति में महत्वपूर्ण हो जाता है जब कई लोग एक टीम की तरह कार्य कर रहे होते हैं तथा कुछ files को share करना चाहते हैं। Single user की स्थिति में भी ऐसा सम्भव हो जाता है कि कुछ files अलग-अलग directories में रखने की आवश्यकता हो तथा ऐसी स्थिति में Acyclic graph structure use किया जाता है।

Shared file कई विधियों से implement की जा सकती हैं। एक विधि यह है कि एक नयी directory entry उत्पन्न की जाये जिसको कि link कहा जाता है। लिंक एक pointer होता है जो किसी directory या file की ओर point करता है।

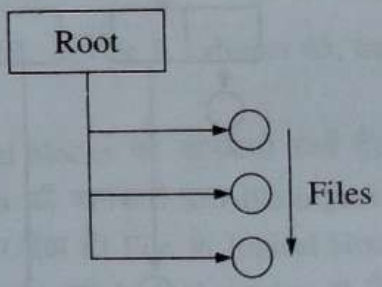
लिंक को absolute या relative path name द्वारा implement किया जा सकता है। जब file का reference किया जाता है तो directory search की जाती है। यदि directory entry एक link के रूप में marked होती है तो link information में file का real name शामिल कर दिया जाता है। Link को resolve करने हेतु pathname का यूज़ करके वास्तविक file को locate किया जाता है। Link की पहचान directory entry में उनके format द्वारा की जा सकती है। एक अन्य विधि द्वारा shared file को implement करने की एक अन्य विधि यह है कि उनसे सम्बन्धित सभी सूचना duplicate कर दी जाए किन्तु इसमें file के modify होने पर consistency का ध्यान रखना पड़ता है।

Acyclic graph structure tree structure से अधिक लचीला होता है किन्तु यह अधिक जटिल भी होता है। इसमें file के multiple absolute pathnames हो सकते हैं। इसमें deletion की भी समस्या होती है। यदि किसी के delete करने पर file को remove कर दिया जाए तो उसके pointers या तो एक nonexistent file की ओर point करने लगते हैं या फिर दूसरी file के middle में point करने लगते हैं।

महत्वपूर्ण प्रश्न: Directory system से आप क्या समझते हैं? Two level, tree structure, acyclic graph एवं general graph directory को स्पष्ट कीजिये।

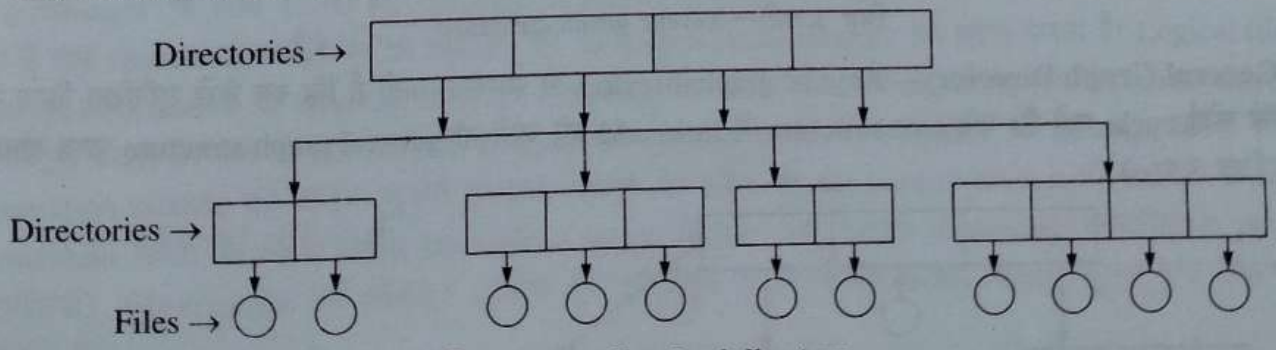
उत्तर—Files के use को आसान तथा सुविधाजनक बनाने के लिये इनको directories में व्यवस्थित किया जाता है। Directory system का प्रयोग करके फाइल्स को जल्दी locate किया जा सकता है, files का सुविधापूर्वक नामकरण किया जा सकता है। एक ही प्रकार की फाइल्स को ग्रुप किया जा सकता है। विभिन्न प्रकार के directory operations है— Create a file, delete a file, truncate a file, search a file, rename a file, इत्यादि।

(i) Single Level Directory Structure—इसमे सभी फाइल्स एक ही directory में होती है चित्र 3.7(a)। सबका अलग-अलग नाम होना चहिये। यूजर को सभी फाइल्स का नाम याद रखना पड़ता है।



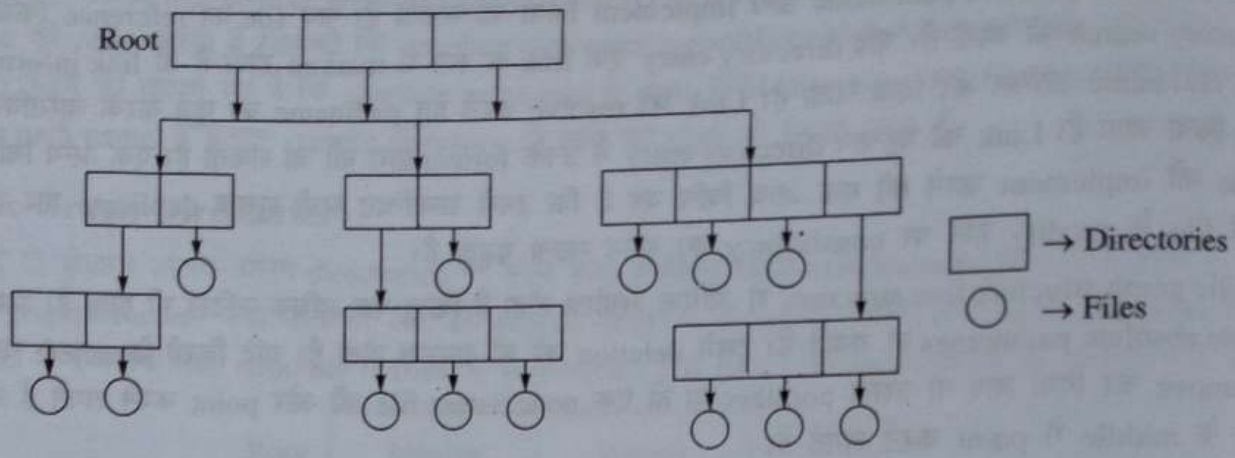
चित्र 3.7(a)—Single level directory

(ii) Two Level Directory—प्रत्येक user की अपनी UFD (user file directory) होती है। जब यूज़र कार्य start करता है या login करता है तो system की MFD (master file directory) search की जाती है। MFD user name या account number द्वारा indexed होती है। जब user अपनी किसी file को refer करता है तो उसकी अपनी directory search होती है (चित्र 3.7(b) देखें)



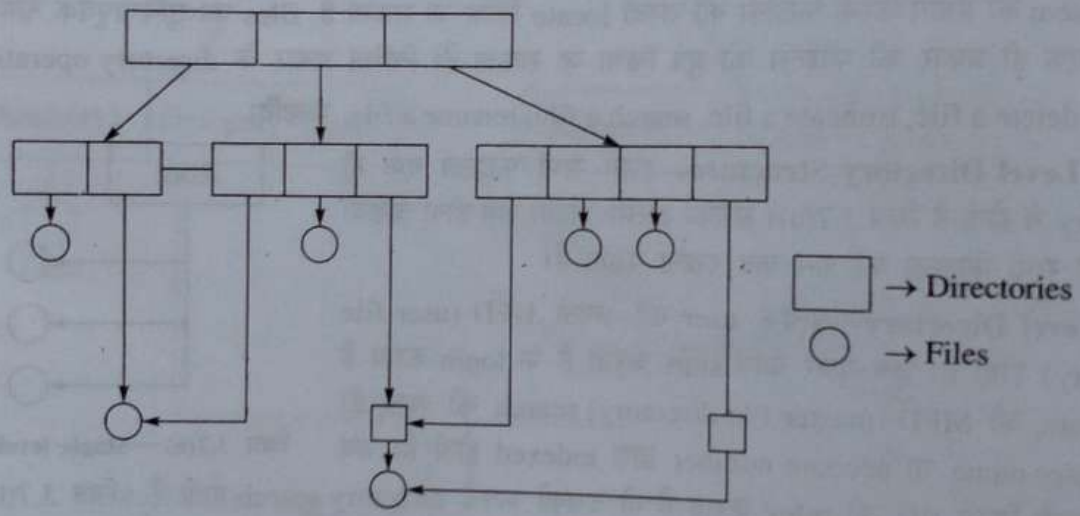
चित्र 3.7(b)—Two level directory

(iii) Tree Structured Directory—इस directory की subdirectories व files होती है (चित्र 3.7(c) देखें)। Directory entry में एक बिट entry को परिभाषित करती है। Special system calls द्वारा directories को create या delete किया जाता है। जब किसी file को reference किया जाता है तो current directory search होती है। Absolute path name root से शुरू होता है तथा उसमें रास्ते की सभी subdirectories के नाम होते हैं। Relative pathname current directory से शुरू होता है।



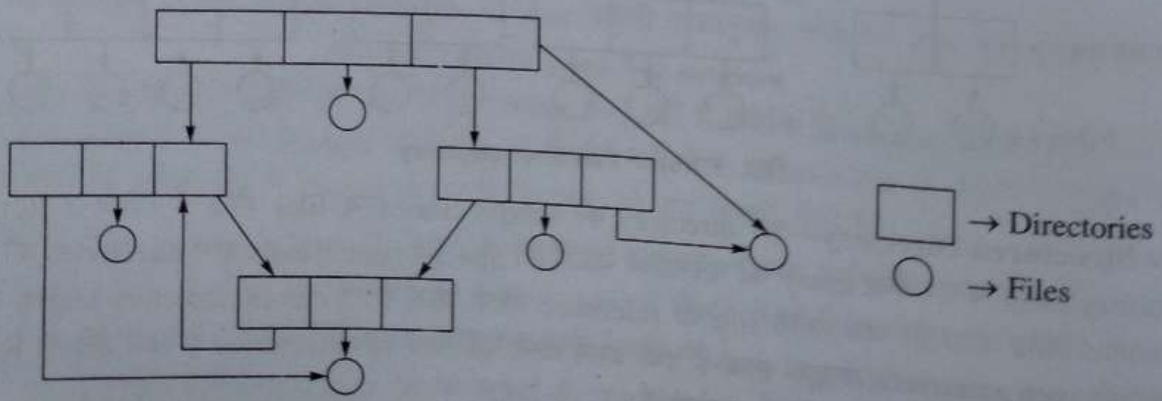
चित्र 3.7(c)—Tree structured directory

(iv) **Acyclic Graph Directory:** Acyclic graph directory structure tree structure की files तथा subdirectories को share करने की अनुमति (इजाज़त) देता है। इसमें shared file या directory की केवल एक copy stored रहती है। Files एक-या-अधिक path से access की जा सकती है (चित्र 3.7(d) देखें)।



चित्र 3.7(d)—Acyclic graph structure

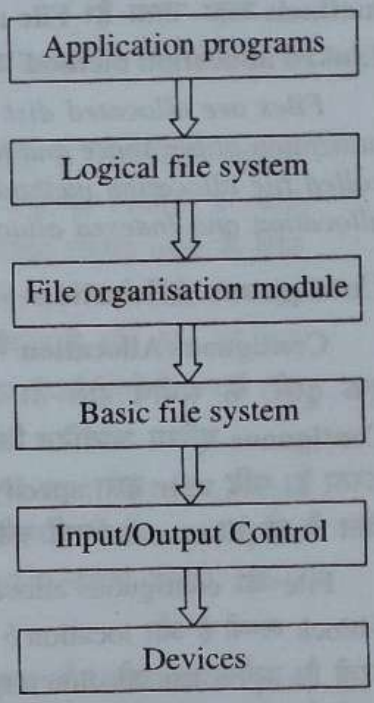
General Graph Directory—Acyclic graph directory में समस्या आती है कि यह कैसे सुनिश्चित किया जाये कि कोई cycle नहीं है। यदि tree structure में links add की जायें तो general graph structure प्राप्त होता है। (चित्र 3.7(e)).



चित्र 3.7(e)—General graph directory

§ 3.8. Layered File System :

हमने देखा कि फाइल सिस्टम द्वारा फाइल कन्टैन्ट्स (डेटा, प्रोग्राम इत्यादि) के online storage तथा access हेतु मैकेनिज़्म प्रदान की जाती है। फाइल सिस्टम secondary storage में stored होता है जैसे कि discs। Disc में data को rewrite (अर्थात् किसी block को read करके, modify करके, वहाँ दूसरा data store करना) किया जा सकता है तथा डिस्क में store information को directly access किया जा सकता है।



चित्र 3.8—Layered file system

File system कर levels (स्तरों) से composed (निर्मित) होता है। (चित्र 3.8)

सबसे निचला स्तर I/O control होता है जिसमें main memory तथा disk system के मध्य सूचना ट्रांसफर हेतु device drivers तथा interrupt handlers होते हैं। Device driver एक प्रकार का translator होता है। इसकी input पर high level commands होती है तथा output पर low level, hardware specific instructions होते हैं जो कि hardware controller द्वारा use किये जाते हैं। Hardware controller I/O devices को बाकी सिस्टम से interface करता है। Device driver सामान्यत: I/O controller की memory की विशेष locations पर विशेष bit pattern लिखकर controller को यह बताता है कि किस device location पर तथा क्या actions लेने हैं।

बेसिक फाइल सिस्टम device driver को generic commands issue करता है ताकि disk के physical blocks को read तथा write किया जा सके। प्रत्येक physical block अपने numeric (संख्यात्मक) डिस्क एड्रैस द्वारा पहचाना जाता है (उदाहरणत: device 1, cylinder 65, track 3, sector 9, इत्यादि)

File organisation module को files, उनके logical blocks तथा physical blocks की जानकारी होती है। प्रयुक्त file allocation का प्रकार (type of file allocation used) तथा file की location की जानकारी द्वारा file organisation module logical block addresses को physical block address में transfer कर देता है। File के logical blocks की क्रम संख्या होती है (0, 1, 2, 3, ... N)। चूंकि डेटा store करने वाले physical blocks इन logical number से मैच नहीं करता, अत: प्रत्येक block के location हेतु translation की आवश्यकता होती है। File organisation module में free space manager भी होता है, जो कि unallocated blocks (अर्थात् वह blocks जो कि allocated नहीं हैं) को track करता है तथा request प्राप्त होने पर इन blocks को file organisation module को प्रदान करता है। Logical file system metadata information को manage करता है। Metadata के अंतर्गत files के contents (अर्थात् actual data) को छोड़ कर सम्पूर्ण फाइल सिस्टम स्ट्रक्चर आता है। Logical file system directory structure को manage करता है ताकि file organisation module को वांछित सूचना उपलब्ध करायी जा सके। यह file control blocks की सहायता से file संरचना को maintain करता है। FCB अर्थात् file control block में file, तथा उसकी ownership (स्वामित्व), permissions (अनुमतियाँ), file contents की लोकेशन इत्यादि सूचनायें होती है। लॉजिकल फाइल सिस्टम सुरक्षा तथा बचाव का कार्य भी करता है।

File system implementation की परतों वाली संरचना को प्रयोग से code के duplication की संभावना नहीं रहती। Input/Output control तथा basic file system codes multiple file systems द्वारा use किये जा सकते हैं। फिर प्रत्येक फाइल सिस्टम का अपना logical file system तथा file organisation modules हो सकते हैं।

§ 3.9. फाइल एलोकेशन विधियाँ (File Allocation Methods) :

हमने देखा कि एक डिस्क पर कई फाइल्स स्टोर की जा सकती है अत: प्रश्न यह उठता है इन फाइल्स को किस प्रकार स्पेस allocate की जाये ताकि डिस्क स्पेस का बेहतर utilization (प्रयोग) हो सके तथा files तक जल्दी पहुँच (quick

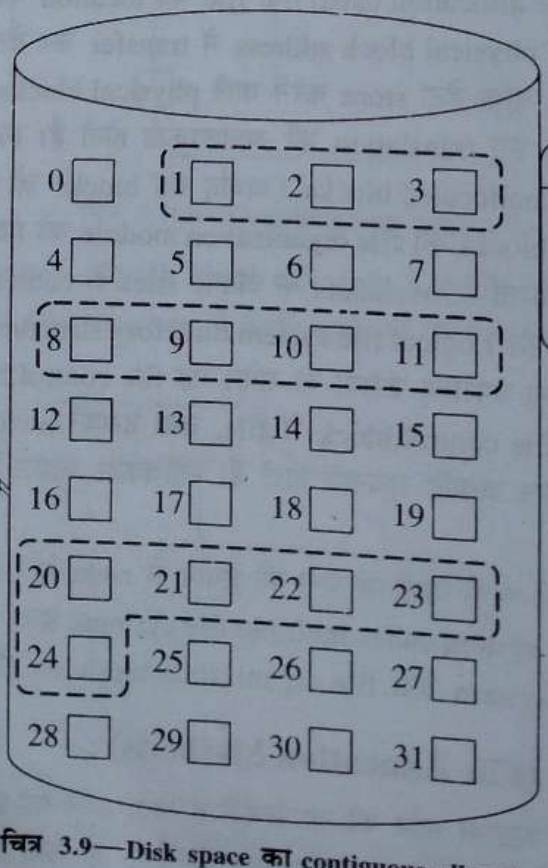
access) बनायी जा सके। **Files को डिस्क स्पेस allocate (आवंटित) करने हेतु प्रयुक्त विधियों को file allocation methods कहा जाता है। File allocation हेतु तीन मुख्य विधियाँ प्रचलित हैं–Contiguous allocation method, Linked allocation method तथा Indexed allocation method.**

Files are allocated disk spaces by operating system. The main idea behind allocation is effective utilization of file space and fast access of the files. The methods for allocating disk space to the files are called file allocation methods. The popular file allocation methods are: Contiguous allocation, Linked allocation and Indexed allocation.

Contiguous Allocation—

Contiguous Allocation में प्रत्येक फाइल डिस्क के Contiguous block में निवास करती है अर्थात् यह blocks एक दूसरे के समीप होते हैं। अत: Contiguous Allocation method में files को secondary storage हेतु Contiguous एरिया आवंटित किया जाता है। किसी फाइल को create करने से पहले user file के size को specify करता है। यदि user द्वारा specify किया गया size बड़ा होता है और उतना contiguous area disk पर उपलब्ध नहीं होता है तो file create नहीं की जा सकती है।

File का contiguous allocation पहले block के disk address और लम्बाई से व्यक्त किया जाता है। यदि file n-block लम्बी है और location b पर स्टार्ट करती है तो वह block—$b, b+1, b+2, b+3, \ldots b+n-1$ blocks occupy करती है। प्रत्येक file की directory entry starting block का address तथा इस file को allot किये गये area को व्यक्त करती है (चित्र 3.9 देखें)। Contiguous Allocation वाली file को access करना आसान होता है। Sequential access की स्थिति में file सिस्टम पिछले reference किये गये block का address याद रखना आवश्यक होता है और आवश्यकता पड़ने पर अगला block read किया जा सकता है। अत: block-b पर start होने वाली फाइल के block-i को directly access करने हेतु हम सीधे block-b + i को directly access कर सकते हैं। अत: Contiguous Allocation sequential तथा direct दोनो access द्वारा सपोर्ट किया जा सकता है।



देखें कि "assignment" नामक फाइल block 20 से start होती है तथा 24 पर समाप्त होती है (लम्बाई = 5 block)

Directory

File	Start	Length
Asha	1	3
Design	8	4
Assignment	20	5

चित्र 3.9—Disk space का contiguous allocation

Contiguous allocation में कुछ समस्यायें भी आती है जैसे कि नयी फाइल के लिए space (स्थान) की तलाश करना और इस कार्य के लिए रणनीति system पर निर्भर करती है।

Contiguous Allocation, Dynamic Storage Allocation के समान होती है अर्थात् free holes में से किस प्रकार से एक *n*-size वाली memory request को सन्तुष्ट किया जाए।

Contiguous Allocation से जुड़ी एक अन्य समस्या है कि ये कैसे निर्धारित किया जाए कि file के लिए कितनी space की आवश्यकता होगी। अत: जब file को create किया जाता है (अर्थात् उत्पन्न किया जाता है) तभी यह निर्धारित हो जाना चाहिए कि इसे कितनी space की आवश्यकता होगी तथा उतनी space उसे allocate की जानी चाहिए लेकिन user के लिए file को create करते समय यह जानकारी होना एक मुश्किल कार्य है कि उसके द्वारा create की जाने वाली file का size कितना होगा। यदि file को आवश्यकता से कम जगह मिली तो file का विस्तार नहीं किया जा सकता है और file को बड़ा नहीं किया जा सकता है (विशेष तौर पर best fit allocation विधि में file के दोनों ओर की जगह used होती है, अत: file को बढ़ाया नहीं जा सकता है) तब दो सम्भावनायें उत्पन्न होती हैं—

(i) User program को उपयुक्त message के द्वारा terminate कर दिया जाए और उसे बड़ी जगह देकर फिर से run किया जाए। चूँकि यह प्रक्रिया महंगी साबित हो सकती है इसलिए इस बात की पूरी सम्भावना है कि user space का over estimation करेगा (अर्थात् आवश्यकता से अधिक जगह बताएगा) जिसके कारण काफी space waste होगी।

(ii) एक बड़ा hole ढूंढ कर file को copy किया जाए और पुरानी space को मुक्त कर दिया जाए और यह प्रक्रिया repeat होती रहे। इस कार्य में काफी समय consume हो सकता है (काफी समय बरबाद हो सकता है)।

एक अन्य मुद्दा यह है कि यदि file के लिए space की जानकारी पहले से हो भी जाए तो भी प्रारम्भ में ही उसे इतनी सारी space आवंटित कर देना inefficient होता है क्योंकि यदि file धीरे-धीरे grow करती है तो बहुत समय तक काफी memory space unused (अप्रयुक्त) रहेगी और इस प्रकार file में काफी समय तक internal fragmentation बना रहेगा।

उपरोक्त समस्याओं के समाधान हेतु कुछ operating systems modified contiguous allocation scheme का प्रयोग करते है। इसमें शुरूआत में एक contiguous हिस्सा allocate किया जाता है तथा बाद में यदि यह हिस्सा पर्याप्त सिद्ध नहीं होता तो एक अन्य हिस्सा जिसको extent (विस्तार) कहा जाता है, add किया जाता है।

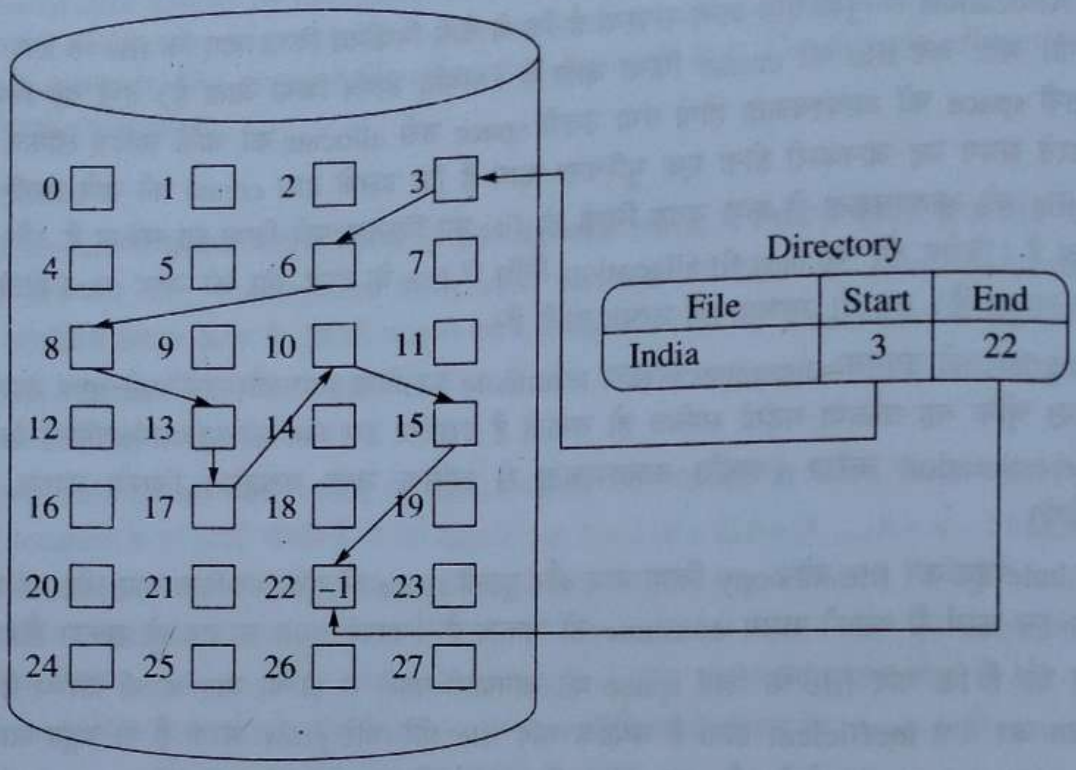
In contiguous allocation, each file occupies contiguous blocks (touching blocks, neighbouring blocks) on the disk. The location of a file is defined by the disk address of the first (starting) block, and its length. For example, if start block is 20 and the length is 5 blocks than the file will occupy block number 20 to 24. Both sequential access and direct access are supported by the contiguous allocation. The main limitation of contiguous allocation is difficulty to find free space for a new file. The other problem is that is requires that the user should be sure of the space required while creating a new file.

- Each file occupies a contiguous address space on disk (क्रम में allocated blocks)
- Assigned disk address is in linear order.
- Easy to implement(आसान)।
- External fragmentation is a major limitation with this type of allocation technique (बाहरी फ्रैगमैन्टेशन की समस्या)।

Linked Allocation—

Linked Allocation में प्रत्येक file डिस्क blocks की linked list होती है तथा disk block पूरी disk में कहीं भी scattered (बिखरे हुए) हो सकते हैं। Directory में file के पहले block व आखिरी block हेतु pointer होता है। उदाहरणत: कोई file जिसमे आठ block है (चित्र 3.16(a) देखें) block-3. पर प्रारम्भ हो सकती है फिर block-6. पर continue कर

सकती है और तत्पश्चात block 8, block-13, block-17, block-10, block-15, पर continue करके block-22 पर समाप्त हो सकती है (चित्र देखें) प्रत्येक block में अगले ब्लॉक हेतु pointer होता है यह pointers user को उपलब्ध नहीं होते हैं। अत: यदि प्रत्येक block 1024 bytes का है तथा प्रत्येक disk address (अर्थात् pointer) 8 bytes का है तो user को block size 1024-8 अर्थात् 1016 bytes का ही दिखेगा।



चित्र 3.10(a)—लिंक्ड एलोकेशन

एक नयी फाइल उत्पन्न करने हेतु डायरैक्ट्री में एक नयी एंट्री create कर दी जाती है। लिंक्ड एलोकेशन में प्रत्येक डायरैक्ट्री एंट्री में file के पहले डिस्क block का pointer होता है। Pointer को nil (the end-of-list pointer value) पर initialize किया जाता है (यह indicate करने के लिये कि file empty अर्थात् खाली है)। Size field को भी शून्य पर सैट कर किया जाता है। फाइल में write करने पर फ्री स्पेस मैनेजमेंट सिस्टम एक फ्री ब्लॉक की तलाश करता है और इस नये ब्लॉक पर write किया जाता है और इस ब्लॉक को फाइल के end से लिंक कर दिया जाता है। File को read करने हेतु एक-एक ब्लॉक को pointers का अनुसरण (follow) करते हुए read किया जाता है। अत: लिंक्ड allocation में external fragmentation नहीं होता है तथा free space लिस्ट का कोई भी ब्लॉक request को सन्तुष्ट करने हेतु यूज़ किया जाता है। इसमें file के create होते समय फाइल का size declare करने की आवश्यकता नहीं होती तथा जब तक free blocks उपलब्ध होते हैं तब तक file grow कर सकती है और इसलिए डिस्क स्पेस को compact करने की आवश्यकता नहीं होती है।

लिंक एलोकेशन की कुछ सीमाएँ भी होती है। पहली समस्या यह है कि इसको केवल sequential access हेतु use किया जा सकता है अर्थात् यदि फाइल का nवाँ ब्लॉक find करना है तो फाइल के शुरूआत से स्टार्ट करना पड़ेगा और प्वाइंटर को follow करते हुए ही n-वें ब्लॉक तक पहुँचा जा सकेगा।

प्वाइंटर के प्रत्येक access के लिए एक डिस्क read (और कभी-कभी डिस्क seak) ऑपरेशन की आवश्यकता होती है। अत: लिंक्ड एलोकेशन फाइल द्वारा डायरेक्ट ऐक्सैस inefficient होता है।

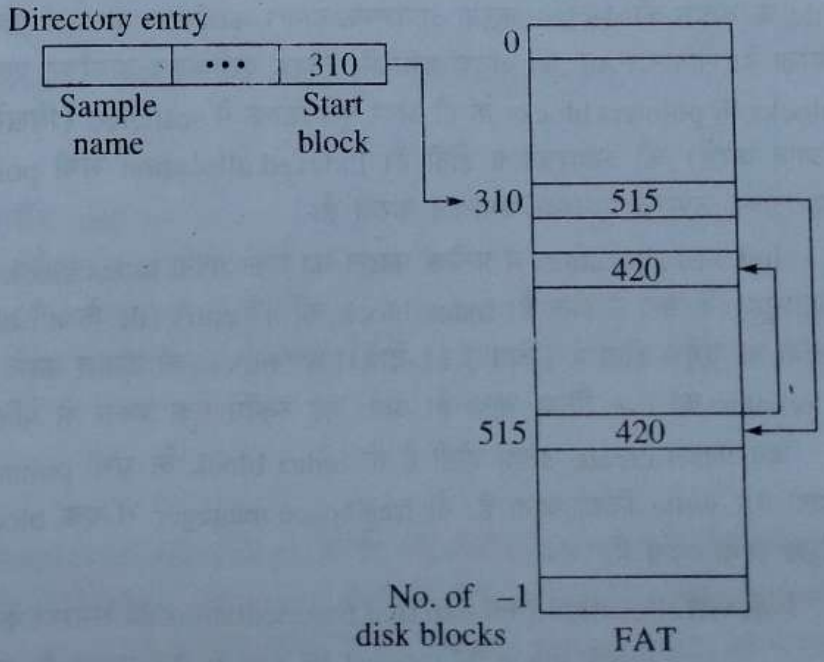
लिंक्ड एलोकेशन से जुड़ी दूसरी समस्या है, प्वाइंटर्स के लिए स्पेस की आवश्यकता। यदि 1024 वाट वाले ब्लॉक में प्वाइंटर को 8 बाइट की आवश्यकता है तो इसका अर्थ यह हुआ कि डिस्क का $\frac{8}{1024} \times 100$ अर्थात् 0.781 % प्वाइंटर स्टोर करने के लिए यूज़ करना पड़ता है। इसलिए फाइल्स कुछ ज्यादा space घेरती हैं। इस समस्या का एक समाधान यह है कि block

को ग्रुप में इकट्ठा किया जाए (जिनको क्लस्टर्स (clusters या गुच्छा) कहा जाता है) तथा block के बजाए clusters allocate किया जाये। उदाहरणत: किसी file system में आठ ब्लॉक का clusters define कर दिया जाए और disk में इन clusters units पर operate किया जाए। अत: इस स्थिति में file disk space में pointer द्वारा घेरी जाने वाली जगह का प्रतिशत कम हो जाएगा। इस विधि के प्रयोग से logical to physical block mapping सरल रहती है तथा डिस्क की thoughtput में सुधार आता है (क्योंकि कम डिस्क हेड seeks की आवश्यकता होती है) तथा block allocation तथा free list management हेतु कम जगह की आवश्यकता होती है। किन्तु इस विधि से internal fragmentation बढ़ जाता है क्योंकि एक आंशिक रूप से भरे हुए क्लस्टर में एक आंशिक रूप से भरे हुए ब्लॉक की तुलना में ज्यादा स्पेस waste होने की सम्भावना होती है।

लिंक्ड एलोकेशन से जुड़ी एक अन्य समस्या reliability अर्थात् विश्वसनीयता है। चूंकि इस विधि में सभी files प्वाइंटर्स की मदद से जुड़ी होती है तथा यह प्वाइंटर्स पूरी डिस्क पर scattered (बिखरे होते हैं)। अत: यदि एक भी प्वाइंटर लॉस या डैमेज (नष्ट) हो जाए तो पूरी फाइल लॉस्ट (गुम) हो जाएगी। ऑपरेटिंग सिस्टम सॉफ्टवेयर या डिस्क हार्डवेयर में कोई bug (त्रुटि) होने पर यदि कोई गलत प्वाइंटर pickup कर लिया गया तो इस block से आगे के block भी गलत pickup हो जाएंगें और फाइल free space list या किसी अन्य फाइल से लिंक हो जाएगी। इस समस्या का आंशिक समाधान होता है doubly linked list या फिर प्रत्येक block में फाइल नेम तथा रिलेटिव ब्लॉक नम्बर स्टोर करना किन्तु ये सब विधियाँ प्रत्येक फाइल स्टोर करने के ओवरहैड (overhead या खर्च) को बढ़ा देते हैं। लिंक्ड एलोकेशन का एक मिलता जुलता रूप है—फाइल एलोकेशन टेबल (file allocation table या FAT) का प्रयोग। यह एक सरल विधि है जिसको MS DOS या OS2 ऑपरेटिंग सिस्टम में प्रयोग किया गया। इस विधि में डिस्क की शुरूआत का एक हिस्सा टेबिल के लिए रखा जाता है। इस टेबिल में प्रत्येक डिस्क ब्लॉक के लिए एक एन्ट्री होती है और इसको block number द्वारा इंडैक्स किये जाता है। FAT को भी linked list की भांति प्रयोग किया जाता है। इसमें directory entry में file के पहले ब्लॉक का block number होता है।

Directory entry में file के पहले block का block number होता है। उस block number द्वारा index की गई table entry में file के अगले block का block number होता है। यह चेन (chain) अंतिम ब्लॉक तक continue करती है, जिसमें table entry के रूप में विशेष end-of-file value (जो कि file की समाप्ति को व्यक्त करती है) होती है। अप्रयुक्त blocks 0-table value व्यक्त किये जाते हैं। फाइल को एक नया ब्लॉक allocate करने हेतु पहली 0-valued table entry (अर्थात् पहले उपलब्ध अप्रयुक्त ब्लॉकों) की तलाश की जाती है तथा पिछली end-of-file value को इस ब्लॉक के address से replace (प्रतिस्थापित) कर दिया जाता है तथा इस नये ब्लॉक की 0-value को end of file value से replace कर दिया जाता है (अर्थात् अब यह नया ब्लॉक file का आखिरी ब्लॉक बन जाता है)। FAT संरचना का उदाहरण चित्र 3.10(b) में प्रदर्शित है।

FAT allocation scheme के परिणामस्वरूप भारी संख्या में disk head seeks उत्पन्न हो सकते हैं (जब तक कि FAT को cached न किया गया हो)। डिस्क हैड को FAT को read करने हेतु volume के start तक move करना होता है तथा वांछित block की location तलाश करनी पड़ती है, तथा फिर block की location की ओर move करना होता है। सबसे खराब स्थिति में, दोनों moves प्रत्येक block हेतु होती है। फायदा यह होता है कि random access time में सुधार आता है, क्योंकि FAT की information (सूचना) को read करके डिस्क हैड किसी भी block की location (स्थिति) को ढूँढ सकता है।



चित्र 3.10(b)—FAT (File Allocation Table)

In linked allocation, each file is a linked list of disk blocks—

- *The directory contains a pointer to the first and (optionally the last) block of the file. For example, a file of 8 blocks which starts at block 3, might continue at block 6, then block 8, block 13, block 17, block 10, block 15 and finally block 30 [see Fig. 3.10(a)]. Each block contains a pointer to the next block and the last block contains a NIL pointer. The value-1 may be used for NIL to differentiate it from block 0.*
- *With linked allocation, every directory entry has a pointer to the first disk block of the file. This pointer is initialized to nil (the end-of-list pointer value) to indicate an empty file. A write to a file removes the first free block and writes to that block. This new block is then linked to the end of the file. To read a file, the pointers are just followed from block to block.*
- *No problem external fragmentation. Any free block can be used to satisfy the request.*
- *No need to declare the size of a file when that file is created.*
- *File can grow continuously.*
- *Inefficient to support direct-access.*
- *Effective only for sequential-access files.*
- *To find the nth block of a file, it must start at the beginning of that file and follow the pointers until the nth block is reached.*
- *Each access to a pointer requires a disk read.*

- Each file carries a list of links to disk blocks (प्रत्येक file में disk block हेतु link list होती है)
- Directory contains link (pointer) to first block of a file (Directory में पहले block की link होती है)
- No external fragmentation (External fragmentation की समस्या से मुक्त)
- Effectively used in sequential access file (Sequential access में प्रयोग)
- Inefficient in case of direct access file (direct access file में inefficient)

इंडैक्सड एलोकेशन (Indexed Allocation)—

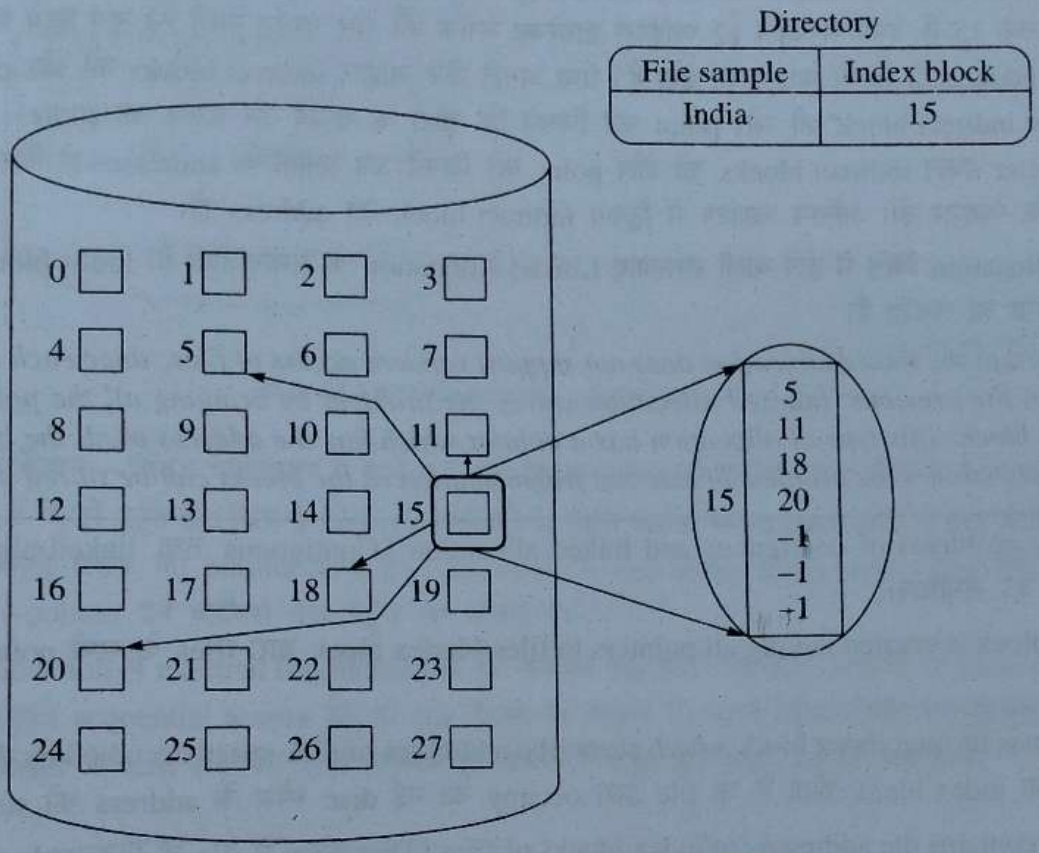
आपने देखा कि लिंक्ड एलोकेशन कन्टीजस एलोकेशन की file-size declaration समस्या (अर्थात् file उत्पन्न करते समय file के साइज को declare करना आवश्यक होना) का निदान करता है तथा external fragmentation समस्या का भी निदान करता है। लेकिन FAT की अनुपस्थिति में, लिंक्ड एलोकेशन डायरैक्ट एक्सैस को दक्षतापूर्वक सपोर्ट नहीं कर सकता क्योंकि blocks के pointers blocks के ही अंदर पूरी डिस्क में scattered (बिखरे हुए) होते हैं तथा उनको क्रमिक रूप से retrieved (प्राप्त करने) की आवश्यकता होती है। Indexed allocation सभी pointers को एक location (अर्थात index block) पर लाकर उपरोक्त समस्या समाधान करती है।

Indexed allocation में प्रत्येक फाइल का एक अपना index block होता है, जो key disk-block address की array (आव्यूह) के रूप में होता है। Index block की n^{th} entry file के n^{th} block की ओर point करती है। डायरैक्ट्री में इंडैक्स ब्लॉक का एड्रैस होता है (चित्र 3.11 देखें)। n^{th} block की तलाश करने तथा उसे read करने हेतु n^{th} index block entry में pointer को use किया जाता है। अत: यह स्कीम एक प्रकार से पेजिंग स्कीम से मिलती जुलती है।

जब फाइल create उत्पन्न होती है तो index block के सभी pointers nil पर सैट कर दिये जाते हैं। जब n^{th} block पहली बार write किया जाता है, तो free space manager से एक block प्राप्त करके उसका address n^{th} block entry में रख दिया जाता है।

Indexed allocation बिना external fragmentation की समस्या के डायरैक्ट एक्सैस को सपोर्ट करता है क्योंकि डिस्क की free block space की request को satisfy कर सकता है। हालांकि, indexed allocation में भी waste space

की समस्या होती है। इसके साथ-साथ indexed allocation का pointer overhead (खर्च) लिंक्ड एलोकेशन के pointer overhead से अधिक होता है। उदाहरणत:, यदि कोई ऐसी फाइल है जिसमें केवल दो या तीन blocks है लिंक्ड एलोकेशन की स्थिति में प्रत्येक block हेतु केवल एक pointer की जगह ही प्रयुक्त (खर्च) होती है किन्तु indexed allocation की स्थिति में मात्र एक या दो pointers के non-nil होने पर भी एक पूरा index block allocate करना होता है।



चित्र 3.11—Indexed allocation का उदाहरण

प्रश्न यह उठता है कि index block कितना बड़ा होना चाहिये। चूंकि प्रत्येक file हेतु index block होना आवश्यक है, अत: index block का size जितना कम हो, उतना ही अच्छा है। किन्तु यदि index block का size बहुत छोटा कर दिया जाये, तो वह बड़ी फाइलों हेतु पर्याप्त pointers को hold नहीं कर पायेगा तथा इस समस्या को deal करने हेतु कुछ न कुछ व्यवस्था करना आवश्यक हो जायेगा। इसके लिये कई mechanisms प्रयोग की जा सकती है जिनका वर्णन निम्नवत् है—

लिंक्ड स्कीम (Linked Scheme)—

एक index block सामान्यत: एक डिस्क का ब्लॉक (one disk block) होता है। अत:, यह स्वयं द्वारा ही directly read या write किया जा सकता है। बडी फाइलों हेतु कई इंडैक्स ब्लॉक्स को एक साथ लिंक किया जा सकता है। उदाहरणत:, एक इंडैक्स ब्लॉक में header का प्रयोग करके file name व पहले 100 disk block का address दिया जा सकता है। Index block के last word (next address) को छोटी फाइल के लिये nil किया जा सकता है तथा बड़ी फाइलों हेतु यह अगले index block का pointer हो सकता है।

मल्टीलैवल इंडैक्स (Multilevel Index)—

इस विधि में एक first level index block second level index block के सैट की ओर point करता है और second level block file block की ओर point करता है। किसी block को access करने के लिए operating system second level index block के तलाश हेतु first level index block का उपयोग करता है और फिर इस second block का उपयोग

करके वांछित ब्लॉक की तलाश करता है। इस विधि का विस्तार file के अधिकतम size के अनुसार तीसरे या चौथे लेवल तक किया जा सकता है।

संयुक्त स्कीम (Combined Scheme)—

एक अन्य विकल्प यह है कि इन्डैक्स ब्लॉक के पहले कुछ प्वाइंटर्स को फाइल के inode में रखा जाए। माना कि कुल प्वाइंटर्स की संख्या 15 है, इनमें से पहले 12 प्वाइंटर्स डायरेक्ट ब्लॉक की ओर प्वाइंट करने हेतु यूज़ किये जाएँ (अर्थात् इनमें उन ब्लॉक का address हो जिनमें फाइल का डेटा है) तथा अगले तीन प्वाइंटर indirect blocks की ओर point करें, पहला pointer single indirect block की ओर point करे जिसमें कि डाटा के बजाय उन ब्लॉक का address हो, जिनमें डाटा हो। दूसरा pointer डबल indirect blocks की ओर point करे जिनमें उस ब्लॉक के addresses हो जिसमें कि वास्तविक डाटा ब्लॉक्स के प्वाइंटर हों। अन्तिम प्वाइंटर में ट्रिपल indirect block का address हो।

Index Allocation विधि में होने वाली समस्याएं Linked Allocation के समान होती है। Index block को memory में cached किया जा सकता है।

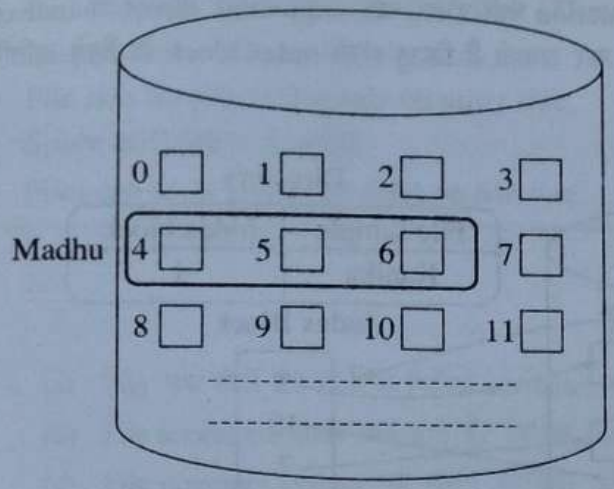
We know that the linked allocation does not support random access of files, since each block can only be found from the previous. Indexed allocation solves the problem by bringing all the pointers together into an index block. This type of allocation has a pointer which has the address of all the blocks of a file. This method also solves the problem of external fragmentation as the blocks can be stored in any location.

- Solve the problems of contiguous and linked allocation (Contiguous तथा linked allocation वाली समस्याओं का समाधान)।
- A index block is created having all pointers to files (Index block द्वारा files के सभी pointers को एकत्र करना)।
- Each file has its own index block which stores the addresses of disk space occupied by the file (प्रत्येक file का एक index block होता है जो file द्वारा occupy की गई disc स्पेस के address को store करता है)।
- Directory contains the addresses of index blocks of files (Directory में file के index block का address होता है)।

महत्वपूर्ण प्रश्न: फाइल एलोकेशन विधियों से आप क्या समझते हैं? फाइल एलोकेशन विधियों का वर्णन कीजिये।

उत्तर—चूंकि एक ही डिस्क में कई फाइल्स store होती है, अत: इन files को डिस्क allocate करने हेतु कई विधियों का प्रयोग किया जाता है ताकि disc की space का effective utilization हो जाये तथा आवश्यकता पड़ने पर file को जल्दी से जल्दी access किया जा सके। फाइल्स को डिस्क स्पेस आवंटित करने हेतु प्रयुक्त विधियां file allocation methods कहलाती है। File allocation methods निम्नवत् हैं—

(i) Contiguous Allocation Methods—Contiguous allocation में file को disk में contiguous blocks (लगातार, क्रम से) allocate कर दिये जाते हैं। Directory table में starting physical address तथा blocks की संख्या रहती है। (चित्र 3.12(a) देखें) यह विधि fast तथा simple होती है इसमें head movement की आवश्यकता नहीं होती (केवल एक ट्रैक movement पर्याप्त होता है) किन्तु इसमें user को file का size पहले से पता होना चाहिये। File का विस्तार संभव नहीं हो पाता। External तथा internal fragmentation की संभावना रहती है। File delete होने पर holes उत्पन्न हो जाते हैं।



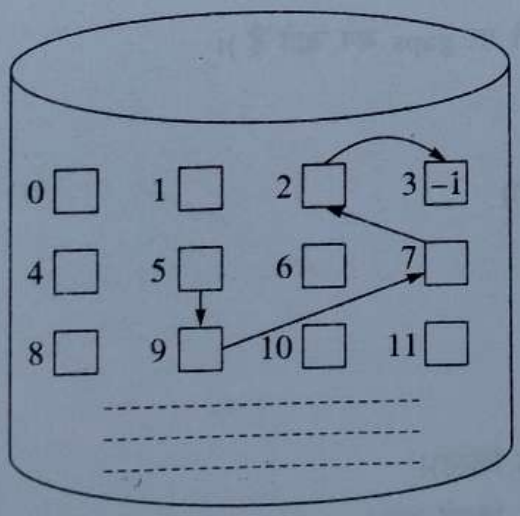
Directory

File	Start	Length
Madhu	4	3

चित्र 3.12(a)—Contiguous allocation

(ii) **लिंक्ड एलोकेशन**—लिंक्ड एलोकेशन में प्रत्येक फाइल डिस्क ब्लॉक की लिंक्ड लिस्ट होती है तथा disk block पूरी डिस्क में बिखरे रहते हैं (चित्र 3.12(b))। डायरैक्ट्री में पहले ब्लॉक का pointer होता है तथा प्रत्येक ब्लॉक में उससे अगले ब्लॉक का pointer होता है। Last block में end-of-the-list pointer value होती है। अत: pointer-by-pointer इन ब्लॉक्स तक पहुँच जा सकता है।

Linked allocation में external fragmentation की समस्या नहीं होती तथा file आसानी से grow कर सकती है। किन्तु इसको sequential access हेतु ही use किया जा सकता है। इसमें विश्वसनीयता की समस्या भी होती है। एक pointer के lost होने पर गलत लिंकिंग हो सकती है तथा पूरी file lost हो सकती है।



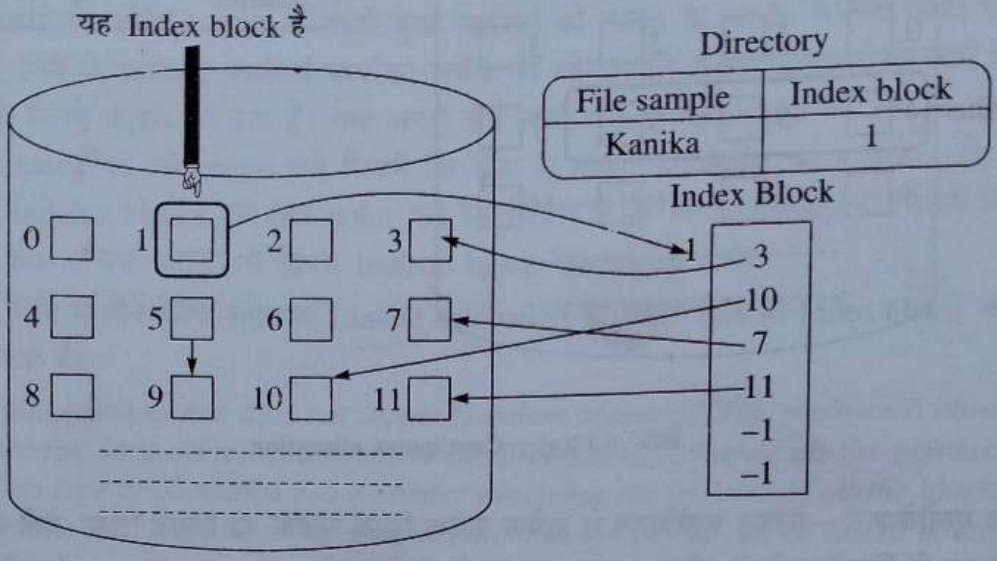
Directory

File	Start	End
Seema	5	3

चित्र 3.12(b)—लिंक्ड एलोकेशन

(iii) **इंडैक्सड एलोकेशन विधि (Indexed Allocation Method)**—Indexed allocation में प्रत्येक file का एक इंडैक्स ब्लॉक होता है जिसमे disk block के addresses का array (व्यूह) होता है। Index block की *n*th entry file के *n*th block की ओर point करती है। Directory में index block का address होता है (चित्र 3.12(c) देखें)।

Indexed addressing में external fragmentation नहीं होता, यह sequential, direct व indexed access को support करती है, file आसानी से grow कर सकती हैं किन्तु इसमें index block के लिये अतिरिक्त space की आवश्यकता होती है।



चित्र 3.12(c)—Indexed allocation

Important Points

Contiguous Allocation—

- All blocks of the file are consecutive (लगातार, नियमित रूप से एक के बाद एक, come in regular order) on disk (सभी block consecutive होते हैं)।
- Deleting files leaves gaps (फाइल को delete करने पर gaps बन जाते हैं)।
- Simple and efficient indexing (सरल इंडैक्सिंग)।
- Supports random access.
- Difficult to grow (फाइल का साइज़ बढ़ने में मुश्किल)
- Easy to map from logical to physical address.
- Compacting disk can be very slow.
- Need to preallocate which wastes space.

Linked Allocation—

- Linked list of file blocks (फाइल ब्लॉक्स की लिंक्ड लिस्ट)।
- Blocks can be scattered on disk (Blocks disk पर बिखरे हुये)।
- No limit on file size, it can grow easily (file size की सीमा नहीं)।
- File can grow easily.

Index Block Allocation—

- File block addresses are stored in an array which is stored in a index block (File block address array के रूप में disk ब्लॉक में stored)।

- Directory has a pointer to index block (Directory index block की ओर point करती है)।
- Easy random and sequential access.
- File size limitation depends on array size.
- Space utilization is good.
- Files can grow easily, no limit on file size.

प्रश्नावली

1. (a) File क्या होती है? File के विभिन्न attributes व types बताइये।
   (b) File access methods क्या होते हैं? विभिन्न file access methods का वर्णन कीजिये।
   (c) File directory system का वर्णन कीजिये। Single level, two level, tree structured, acyclic व general graph directory का वर्णन कीजिये।
   (d) File Allocation methcds क्या होते है? विभिन्न file allocation methods का वर्णन कीजिये।
2. Sequential एवं direct file access की तुलना कीजिये।
3. Contiguous, linked व indexed allocation के लाभ व सीमायें बताइये।

4 Chapter सीपीयू तथा डिस्क, ड्रम शैडयूलिंग (CPU AND DISK, DRUM SCHEDULING)

THINK ABOUT IT

The only thing that stands between you and your dream is the will to try and the belief that it is actually possible. — **Joel Brown**

Never be bullied into silence. Never allow yourself to be made a victim. Accept no one's definition of your life; define yourself. — **Harvey Fierstein**

A new idea is delicate. It can be killed by a sneer or a yawn; it can be stabbed to death by a quip and worried to death by a frown on the right man's brow. — **Charles Brower**

§ 4.1. Scheduling का शाब्दिक अर्थ क्या है?

Schedule (शैडयूल) शब्द का शब्दिक अर्थ होता है "निर्धारित कार्यक्रम" अर्थात् "planning of events" उदाहरणत:, दो व्यक्ति आपस में बातचीत करते हुए आपस में प्रश्न करते हैं "आज आपका क्या शैडयूल है" या फिर कोई व्यक्ति यह कहता है कि "आज तो मेरा बहुत busy schedule है" तो यहां schedule का अर्थ है" निर्धारित कार्यक्रम" या "समय सारणी"।

Scheduling की क्या आवश्यकता है?

एक सिंगल processor सिस्टम में एक समय में केवल एक ही प्रोसैस run कर सकता है। दूसरे सभी process को तब तक wait (इंतज़ार) करना पड़ता है जब तक कि CPU free नहीं हो जाता और तभी दूसरे process को CPU उपलब्ध हो पाता है।

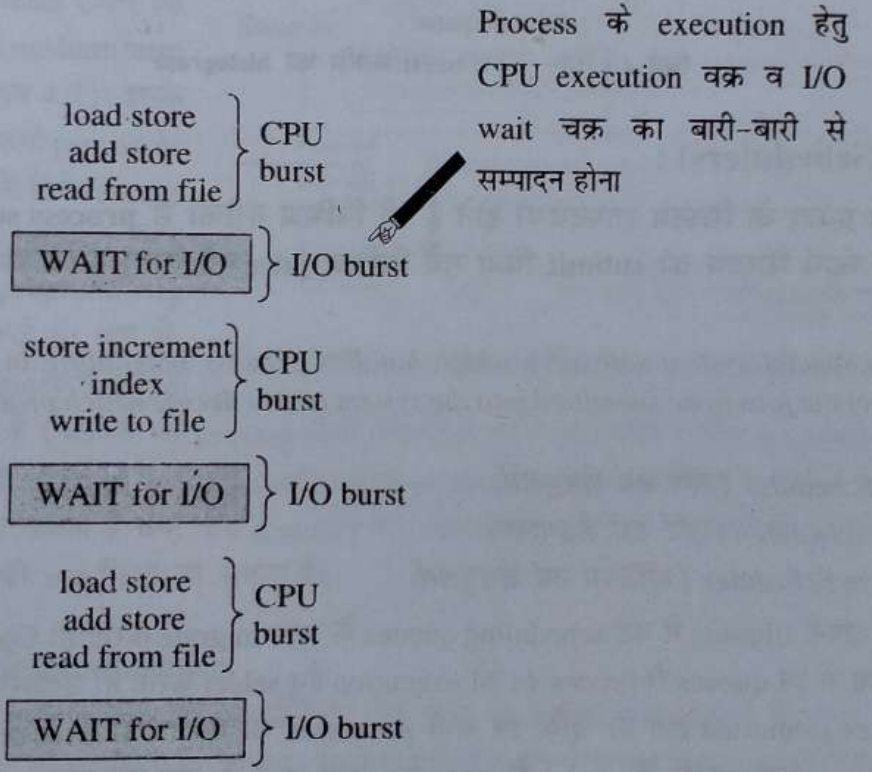
Multiprogramming systems में इस बात पर बल दिया जाता है कि प्रत्येक समय पर कोई न कोई process run करता रहे जिसके कि CPU का अधिकतम utilization हो सके, अर्थात् बेहतर तरीके से CPU का उपयोग हो सके। उदाहरणत: मान लीजिये किसी process को execute करने हेतु किसी I/O request के complete होने का इंतजार करना पड़ रहा है (अर्थात् input device से कोई सूचना प्राप्त करने या output device को सूचना भेजने में कुछ समय wait करना है) तो इस waiting time में CPU खाली रहेगा, CPU द्वारा कोई useful work नहीं किया जा सकेगा। मल्टीप्रोग्रामिंग का उद्देश्य होता है इस waiting time को utilize करना, इसका बेहतर ढंग से उपयोग करना। चूंकि memory में एक समय पर कई सारे process होते हैं, अत: इस waiting time में operating system उस process से CPU को लेकर किसी दूसरे process को दे देता है और यह प्रक्रिया चलती रहती है। जब भी कोई process wait करने वाली स्थिति में आता है तो दूसरा process CPU को take over करके उसका इस्तेमाल करता है।

अत:, CPU scheduling operating system का एक प्रमुख कार्य है। लगभग सभी computer resource प्रयोग से पहले schedule किये जाते हैं। विशेष तौर पर CPU की scheduling तो operating system design में सबसे ज्यादा महत्व रखती है।

In a single-processor system, only one process can run at a time and the others must wait until the CPU is free and can be rescheduled. Multiprogramming emphasizes (बल देता है) on some process running at all times, to maximize CPU utilization. A process is executed until it must wait, generally for the completion and some I/O request. The CPU then just sits idle and this waiting time is wasted; With multiprogramming, this time can be used productively. Several processes are kept in memory at one time. When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process. Every time one process has to wait, another process can take over use of the CPU.

किसी process के execution (अर्थात प्रक्रिया का क्रियान्वयन, कार्य का सम्पादन या निष्पादन करना, कार्य को पूरा करना) में दो चक्र होते हैं—

CPU execution तथा I/O wait तथा process बारी-बारी से इन दोनों states से कई बार complete होते हैं। Process का execution CPU burst से प्रारम्भ होता है (चित्र 4.1(a) देखें), फिर I/O burst होता है, फिर CPU burst होता है, तथा यह प्रक्रिया चलती रहती है, last में एक CPU burst की समाप्ति के साथ process complete हो जाता है तथा execution की समाप्ति हेतु system request भेजी जाती है।

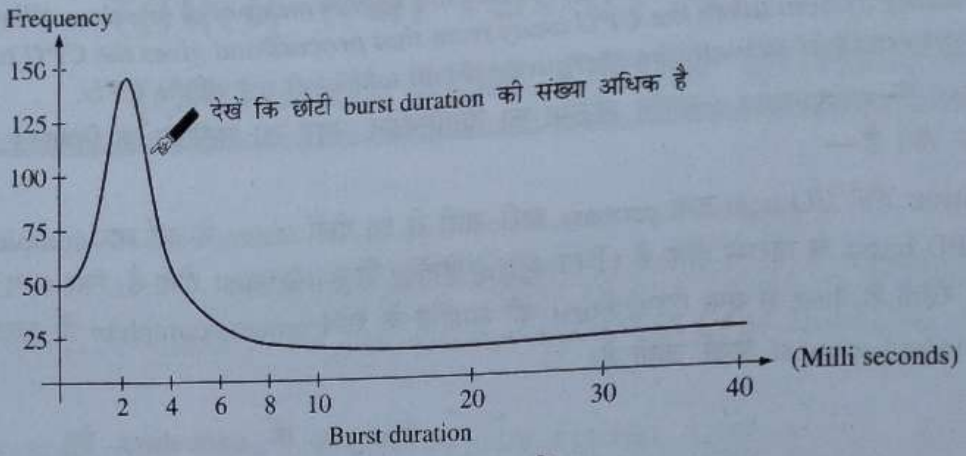


चित्र 4.1(a)—किसी process के execution हेतु CPU व I/O bursts का बारी-बारी से सम्पादन होना

CPU burst की अवधि विभिन्न processes व कम्प्यूटर्स के लिये भिन्न होती है किन्तु इनकी आवृति वक्र चित्र 4.1(b) में दर्शाये ग्राफ के समान होती है। इस curve के exponential (चरघातांकीय) या hyper exponential अभिलक्षण होते हैं तथा इसमें बड़ी संख्या में छोटी अवधि की CPU bursts तथा कम संख्या में बड़ी अवधि की CPU bursts होती हैं। एक I/O bound program में बहुत सारी short CPU bursts तथा एक CPU bound program में कम संख्या (few) में लम्बी अवधि वाली CPU bursts हो सकती है। इस प्रकार का वितरण (तथा इसकी जानकारी) CPU scheduling algorithm के चयन में महत्वपूर्ण भूमिका अदा कर सकता है।

Process execution consists of a cycle of CPU execution and I/O wait. Processes keep on alternating between these two states. Process execution begins with a CPU burst that is followed by an I/O burst, which ios followed by another CPU burst, then another I/O burst, and so on. Eventually, the final CPU burst ends with a system request to terminate execution.

The durations of CPU bursts vary greatly from process to process and from computer to computer, they tend to have a frequency curve as shown in figure. The curve is generally characterized as exponential or hyperexponential, with a large number of short CPU bursts and a small number of long CPU bursts. An I/O-bound program typically has many short CPU bursts. A CPU- bound generally has a few long CPU bursts.



चित्र 4.1 (b)—CPU burst अवधि का histogram

§ 4.2. शैडयूलर्स (Schedulers) :

शैडयूलर्स विशेष प्रकार के सिस्टम साफ्टवेयर होते हैं जो विभिन्न तरीकों से process scheduling को handle करते हैं। इनका मुख्य कार्य सिस्टम को submit किये गये विभिन्न jobs को चयन करने तथा run करने का निर्णय लेना होता है।

Schedulers are special system softwares which handles process scheduling in different ways. Their main work is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types:

- *Long Term Scheduler (लॉग टर्म शैडयूलर्स)*
- *Short Term Scheduler (शॉर्ट टर्म शैडयूलर्स)*
- *Medium Term Scheduler (मीडियम टर्म शैडयूलर्स)*

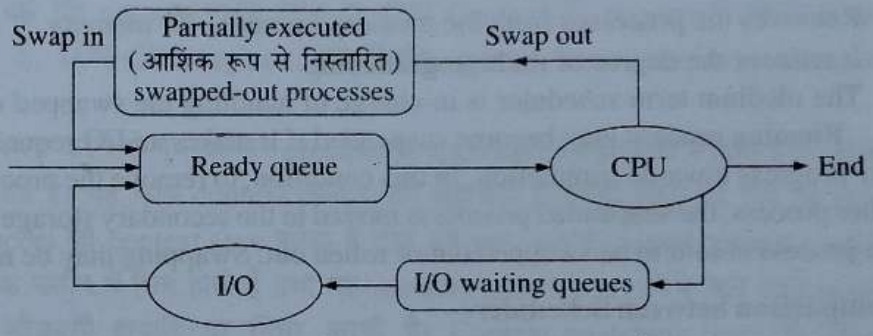
कोई भी process अपनी lifetime में कई scheduling queues के मध्य migrate करता है। Operating systems उपयुक्त schedulers की सहायता से इन queues से processes को execution हेतु select करता है। उदाहरणत: किसी batch system में बहुत सारे processes submitted होते हैं। चूँकि इन सभी processes को एकदम से execute करना सम्भव नहीं होता, अत:, इनको mass storage युक्ति (जैसे कि disk) में store कर दिया जाता है, तथा वह अपनी बारी आने की प्रतीक्षा करते हैं। Long term scheduler (या job scheduler) इस pool में से process को select करता है तथा execution हेतु memory में load करता है। Short term scheduler इन ready processes में से select करके इन्हें CPU का allocation (एक-एक करके) करता है। अत: इन दोनों schedulers में मुख्य अंतर frequency of execution का है। Short term scheduler को CPU allocation हेतु जल्दी-जल्दी नये प्रॉसेस का चयन करना होता है। सामान्यत:, short term scheduler प्रत्येक 100 ms के पश्चात् execute करता है। इतनी तीव्र गति की आवश्यकताओं के कारण short term scheduler को fast होना चाहिये।

Long term schedulers को उतनी तीव्र गति से कार्य नहीं करना पड़ता क्योंकि एक नये process व दूसरे नये process के उत्पन्न होने के मध्य कुछ मिनटों का समय भी लग सकता है Long term scheduler multiprogramming की degree को कन्ट्रोल करता है अर्थात् memory में कितने processes को स्थान देना है। यदि multiprogramming की डिग्री stable (स्थिर) है तो process creation (उत्पन्न होने) की औसत दर तथा system से निकास होने वाले processes की औसत

निकासी दर equal होनी चाहिये। चूँकि executions के बीच पर्याप्त समयान्तराल (time interval) होता है, अत: long term scheduler के पास यह निर्णय लेने के लिये पर्याप्त समय होता है कि execution हेतु किस process को select करना है।

यह भी महत्वपूर्ण है कि long term scheduler को ध्यानपूर्वक selection करना चाहिये। सामान्यत:, अधिकांश processes को I/O bound या CPU bound की श्रेणी में रखा जा सकता है। I/O bound process वह होता है जो कि computations (गणनाओं) के बजाये I/O करने में अधिक समय व्यतीत करता है। इसके विपरीत, CPU bound process गणनायें करने में अधिक time व्यतीत करता है, तथा उसके द्वारा उत्पन्न I/O requests बहुत कम होती हैं। अत: यह महत्वपूर्ण है कि long scheduler processes का चयन करने हेतु I/O bound तथा CPU bound processes दोनों के मध्य संतुलन बनाते हुये चयन करें। यदि सभी (या ज़्यादातर) चयनित processes I/O bound होंगे तो ready queue ज़्यादातर खाली रहेगी तथा short term scheduler के लिये करने को कुछ भी नहीं बचेगा। इसके विपरीत यदि सभी processes CPU bound होंगे तो I/O waiting queue ज़्यादातर खाली रहेगी, जिससे devices का उपयोग नहीं हो सकेगा तथा सिस्टम में असंतुलन उत्पन्न होगा। अत: सिस्टम से best performance प्राप्त करने हेतु यह जरुरी है कि long term scheduler CPU bound तथा I/O bound processes का combination बनाते हुये ही चयन करें।

कुछ operating systems (जैसे कि time sharing systems) में medium term scheduler भी होता है (चित्र 4.2)। इसके पीछे यह तर्क है कि कभी-कभी processes को memory से कुछ समय के लिये remove करना advantageous (फायदेमंद) सिद्ध होता है तथा इससे multiprogramming की डिग्री कम की जा सकती है। बाद में, process को फिर से memory में लाकर execute कराया जा सकता है (जहाँ से वह process पहले रोका गया था)। यह स्कीम स्वैपिंग (swapping) कहलाती है। प्रौसेस को swap-out तथा बाद में swap-in करने हेतु medium term scheduler का use किया जाता है। Swapping से process के mix में सुधार लाया जा सकता है तथा यदि memory की आवश्यकतायें अधिक हो जाती हैं तो कुछ memory को free करने हेतु भी swapping को use किया जा सकता है।



चित्र 4.2—Queuing में medium term schedulers को add करना

Long term scheduler or job scheduler:

- लांग टर्म शैडयूलर निर्णय लेता है कि प्रौसेसिंग हेतु कौन से प्रोग्राम्स को सिस्टम में प्रवेश देना है (Long term scheduler determines which programs are admitted to the system for processing).
- Process को पंक्ति में से select करना तथा execution हेतु memory में load करना (Selects processes from the queue and loads them into memory for execution).
- Process memory में CPU scheduling हेतु load होता है (Process loads into the memory for CPU scheduling).
- I/O bound तथा processor bound jobs के मध्य संतुलन स्थापित करना (Provides a balanced mix of jobs, such as I/O bound and processor bound).
- Controls the degree of multiprogramming (अर्थात मल्टी प्रोग्रामिंग की डिग्री का नियंत्रण करना). If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system (अर्थात यदि process उत्पन्न होने तथा process निकास होने की गति समान है, तो मल्टीप्रोग्रामिंग की डिग्री को स्थिर कहा जा सकता है)।

On some systems, the long term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When process changes the state from new to ready, then there is use of long term scheduler.

Short Term Scheduler or CPU Scheduler—

- सिस्टम performance को increase करना मुख्य उद्देश्य (Main purpose increasing system performance in accordance with the chosen set of criteria).
- It is the change of ready state to running state of the process.
- Ready process में से किसी एक का चयन करके उसे CPU allocate करना (Selects process among the processes that are ready to execute and allocates CPU to one of them).
- Also known as dispatcher (डिस्पैचर भी कहा जाता है)।
- Faster than long term scheduler (Long term scheduler से fast होता है)।

Medium Term Scheduler—

- Medium term scheduling is part of the swapping.
- Removes the processes from the memory (process को memory से remove करना).
- It reduces the degree of multiprogramming.
- The medium term scheduler is in-charge of handling the swapped out-processes.

Running process may become suspended if it makes an I/O request. Suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other process, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Comparison between Scheduler--

आधार (Criteria)	Long Term Scheduler	Short Term Scheduler	Medium Term Scheduler
Type (प्रकार)	It is a job scheduler (यह जॉब शैडयूलर होता है)	It is a CPU scheduler (यह CPU शैडयूलर होता है)	It is a process swapping scheduler. (यह प्रौसेस स्वैपिंग शैडयूलर होता है)
Speed (गति)	Lesser than short term scheduler	Fastest (सबसे fast)	Between both short and long term scheduler.
मल्टी प्रोग्रामिंग पर प्रभाव	It controls the degree of multiprogramming (नियंत्रित करता है)	It provides lesser control over degree of multiprogramming (कम नियंत्रण)	It reduces the degree of multiprogramming (कम कर देता है)
Time sharing सिस्टम में उपस्थिति	It is almost absent or minimal in time sharing system (अनुपस्थित या न्यूनतम)	It is also minimal in time sharing system (अनुपस्थित)	It is a part of Time sharing systems. (उपस्थित)
कार्य	It selects processes from pool and loads them into memory for execution (पूल से process का चयन करना, तथा execution हेतु मैमोरी में लोड करना)	It selects those processes which are ready to execute (Execution हेतु तैयार processes का चयन करना	It can re-introduce the process into memory and execution can be continued. (प्रौसेस को memory में पुनः लाना)

Long term scheduler determines which programs are admitted to the system for processing. It controls the degree of multiprogramming. Once admitted, a job becomes a process.

Medium term scheduling is part of the swapping function. This relates to processes that are in a blocked or suspended state. They are swapped out of real-memory until they are ready to execute. The swapping-in decision is based on memory-management criteria.

Short term scheduler, also known as a dispatcher executes most frequently, and makes the finest-grained decision of which process should execute next. This scheduler is invoked whenever an event occurs. It may lead to interruption of one process by preemption.

§ 4.3. कॉनटैक्स्ट स्विच (Context Switch) :

जब भी CPU को interrupt (व्यवधानित) किया जाता है, तो वह अपना वर्तमान कार्य छोड़कर Interrupt service subroutine को serve करता है। जब भी कोई Interrupt होती है तो System उस समय CPU पर run कर रहे process का current context save करता है तथा उसके बाद ही Interrupt को serve करता है ताकि processing के पश्चात् वापस लौटने पर process को पुन: शुरू करते समय उसे वह context वापस प्राप्त हो जायें। Context को process के PCB (Process Control Block) से व्यक्त किया जाता है, इसमें CPU registers की स्थिति, process state, memory management information इत्यादि होती है।

अत: CPU को दूसरे process हेतु स्विच करने से पूर्व वर्तमान process का state save किया जाता है तथा दूसरे process का state restore (पुन: प्राप्त) किया जाता है। यह कार्य context switch की सहायता से किया जाता है। जब भी context switching होती है, तो Kernel old process का context save करता है तथा नये process का saved context load करता है। Context switching की speed प्रत्येक मशीन में भिन्न होती है तथा यह memory speed, copy किये जाने वाले registers की संख्या, special instructions की मौजूदगी इत्यादि पर निर्भर करती है। Context switching time को सामान्यत: मिलीसैकेन्ड्स में व्यक्त किया जाता है।

§ 4.4. सीपीयू शैडयूलर (CPU Schedulers) :

जब भी CPU idle (खाली, कोई काम न होना) हो जाता है तो operating system ready queue (पंक्ति) में खड़े processes (जो कि execution के लिए CPU प्राप्त होने की प्रतीक्षा में हैं) में से किसी process को select करके उसको CPU आवंटित करता है। यह selection process (चयन प्रक्रिया) CPU scheduler (या short term scheduler) द्वारा सम्पन्न की जाती है। शैडयूलर का कार्य होता है— Memory में से execute होने के लिये तैयार processes में से एक को select करना तथा उसे CPU allocate (आवंटित) करना।

यहाँ ध्यान देने योग्य बात यह है कि यह जरूरी नहीं है कि ready queue (execution के लिये तैयार processes की पंक्ति) first in first out (FIFO) queue (अर्थात् पहले आओ, पहले पाओ के आधार पर बनी पंक्ति जैसे कि रेलवे टिकट खिड़की की पंक्ति होती है) आधारित पंक्ति ही हो। Ready queue को implement करने (कार्यान्वित करने) हेतु कई प्रकार की scheduling algorithms होती है, जिनका वर्णन इस अध्याय में आगे किया जायेगा। इन algorithms के आधार पर ready queue के सभी processes को line-up (पंक्ति में खड़ा करना) किया जाता है तथा उसी क्रम में वह अपनी बारी आने का (तथा CPU पर run होने का) इंतज़ार करते हैं। Queues में records सामान्यत: processes के process control blocks (PCBs) होते हैं।

Whenever the CPU becomes idle, the operating system selects one of the processes in the ready queue to be executed. The selection process is carried by the short-term scheduler. The scheduler selects a process from the processes in memory that are ready to execute and allocates the CPU to that process.

The ready queue is not necessarily a first-in first-out (FIFO) queue. A ready queue can be implemented as a FIFO queue, a priority queue, a tree, or even an unordered linked list. However, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally process control blocks (PCBs) of the processes.

Preemptive तथा Non-premptive Scheduling—

Non-preemptive scheduling वह होती है जिसमें process को CPU allocate कर देने के पश्चात् कार्य पूर्ण होने या process के wait state में जाने तक उससे CPU वापस नहीं लिया जाता है।

A scheduling is said to be non-preemptive if, once a process has been allocated the CPU, the CPU cannot be taken away from that process until completion or it enters a wait state.

Preemptive scheduling वह होती है जिसमें process से कार्य को बीच में ही व्यवधानित (Interrupt) करके CPU वापस प्राप्त किया जा सकता है।

A scheduling is said to be preemptive if, once a process has been given the CPU, it can be taken back in between by interrupting the process.

Non-preemptive Scheduling	Preemptive Scheduling
• Non-preemptive scheduling में process को CPU देने के पश्चात् उसे बीच में interrupt करके CPU वापस नहीं लिया जा सकता (Running task is executed till completion)।	• Preemptive scheduling में process से कार्य के बीच में ही CPU वापस लिया जा सकता है (Running task is interrupted for sometime and can be resumed later when priority task has finished its execution)।
• छोटे jobs को बड़े jobs के पूरा होने का अधिक समय तक इंतज़ार करना पड़ सकता है।	• इसमें छोटा job आने पर उसको CPU उपलब्ध कराया जा सकता है।
• Response time अधिक predictable होता है।	• Response time अपेक्षाकृत कम predictive होता है।
• Scheduler दो स्थितियों में CPU वापस लेता है— ◆ Process running state से waiting state में जाये। ◆ Process terminate ही जाये।	• यदि process running से ready state या waiting से ready state में switch करता है तो उसे CPU preempt कराया जा सकता है।

CPU scheduling से संबधित निर्णय लेने की आवश्यकता निम्न स्थितियों के उत्पन्न होने पर पड़ती है—

(i) जब कोई process running state से wait state में switch करता है (उदाहरण I/O request)।

(ii) जब कोई process terminate (समाप्त) हो जाता है।

(iii) जब कोई process running state से ready state में switch करता है (उदाहरण: interrupt)।

(iv) जब कोई process waiting state से ready state में switch करता है (उदाहरण: I/O की समाप्ति के पश्चात)।

उपरोक्त चार स्थितियों में से पहली व दूसरी स्थिति उत्पन्न होने पर निश्चित रूप से एक नया process select करके (ready queue में से) उसको CPU allot करना पड़ता है। दूसरी व तीसरी स्थिति उत्पन्न होने पर कुछ अन्य विकल्प भी सम्भव हैं।

जब उपरोक्त में से पहली व दूसरी स्थिति उत्पन्न होने पर scheduling की जाती है तो इसको non-preemptive scheduling (नॉन-प्रीएम्पटिव शैडयूलिंग) या cooperative (को-ऑपरेटिव या सहकारी) शैडयूलिंग कहा जाता है जबकि अन्य स्थितियों में की गई scheduling preemptive कहलाती है। Non-preemptive scheduling के अंतर्गत जब भी किसी process को CPU आवंटित किया जाता है तो CPU process के पास तब तक रहता है जब तक कि वह process के terminate होने पर या फिर wait state में switch होने पर CPU को release (मुक्त) नहीं कर देता। Windows के सभी versions में इस प्रकार की scheduling का use किया जाता है। Macintosh का **Mac OS X** operating system preemptive scheduling use करता है जबकि Macintosh के पिछले versions cooperative scheduling use करते थे।

कुछ hardwares पर केवल cooperative scheduling ही use की जा सकती है क्योंकि इसमें preemptive scheduling की भाँति कुछ special hardware (जैसे कि timer) की आवश्यकता नहीं होती।

Preempted scheduling से जुड़ी मुख्य समस्या shared डाटा access में आती है (अर्थात् ऐसा data जो दो process के मध्य साझा हो, तक पहुँच बनाना) उदाहरणत: मान लीजिये दो process एक common data को share करते हैं अब यदि उनमें से एक process द्वारा data को update किया जा रहा है तथा उसी दौरान उसको preempt किया जाता है ताकि दूसरा process run कर सके। अब यदि यह दूसरा process उसी डाटा को read करने की कोशिश करता है जो कि inconsistent state (अस्थिर अवस्था) में है तो ऐसी स्थिति में shared data तक पहुँच बनाने हेतु सामंजस्य स्थापित करने हेतु कुछ अन्य मैकेनिज़्म की आवश्यकता पड़ती है। इसका वर्णन आगे किया जायेगा।

डिस्पैचर—

CPU शैडयूलिंग प्रक्रिया में एक महत्वपूर्ण घटक है, डिस्पैचर। डिस्पैचर एक module होता है जो कि short term scheduler द्वारा select किये गये process को CPU का control प्रदान करता है।

"The dispatcher is a module that gives control of the CPU to the process selected by the short term scheduler."

उपरोक्त कार्य में निम्न प्रक्रियायें शामिल होती हैं—

- Switching context (Context को स्विच करना)।
- Switching to user mode (यूज़र मोड पर स्विच करना)।
- Jumping to proper location in the user program to restart the program (उस program को restart (फिर शुरू करना) करने हेतु उपयुक्त location पर जम्प करना)।

डिस्पैचर को fast होना चाहिये क्योंकि प्रत्येक process switch के समय इसको invoke (पुकारना, बुलाना) किया जाता है। डिस्पैचर द्वारा एक process को stop करने तथा दूसरे process की running start करने में लगने वाला समय dispatch latency (डिस्पैच लैटेन्सी) कहा जाता है।

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. It performs the function of switching context, switching to user mode and jumping to the proper location in the user program to restart that program.

The dispatcher should be as fast as possible, since it is called during every process switch. The time it takes for the dispatcher to stop one process and start another running is known as the ***dispatch latency****.*

§ 4.5. शैडयूलिंग क्राइटेरिया (Scheduling Criteria अर्थात् शैडयूलिंग मापदंड) :

विभिन्न शैडयूलिंग algorithms के भिन्न अभिलक्षण होते हैं तथा किसी शैडयूलिंग algorithm का चुनाव कुछ processes के पक्ष में हो सकता है (दूसरे processes की तुलना में)। अत:, भिन्न स्थितियों में भिन्न algorithm को choose किया जाता है।

CPU scheduling algorithms की तुलना हेतु कई मापदंड होते हैं। अत:, कोई algorithm किसी मापदंड पर बेहतर साबित हो सकती है जबकि दूसरी algorithm किसी दूसरी मापदंड पर बेहतर साबित हो सकती है। यह ठीक ऐसा ही है जैसे कि लम्बी दूरी की यात्रा हेतु रेल या aeroplane एक बेहतर विकल्प हो सकता है जबकि छोटी दूरी की यात्रा हेतु बस या कार बेहतर साबित हो सकती है। इसी प्रकार यदि cost को मापदंड बनाया जाये तो train aeroplane की तुलना में बेहतर विकल्प है जबकि यदि समय को मापदंड बनाया जाये तो aeroplane train के तुलना में बेहतर विकल्प है। अत: दो वस्तुओं की तुलना में कौन सा बेहतर है, यह इस बात पर निर्भर करता है कि आप उन दोनों की तुलना करने हेतु क्या मापदंड use कर रहे हैं। इसी प्रकार विभिन्न शैडयूलिंग algorithms की तुलना करने हेतु भी विभिन्न मापदंड होते हैं, जिनका विवरण निम्नवत् है—

- **CPU Utilization : Utilization का अर्थ है, किसी वस्तु का बेहतर तरीके से उपयोग। CPU का utilization तभी सम्भव है जब CPU को अधिक समय के लिये busy (व्यस्त) रखा जाये। CPU utilization की थ्यौरेटिकल रेंज**

0 से 100% हो सकती है। सामान्यत: (real systems में) इसकी रेंज lightly loaded systems (हल्के भार वाले system) के लिये 40% से लेकर heavily loaded systems (अधिक भार वाले सिस्टम्स) के लिए 90% तक हो सकती है।

The more efficiently we can use a system, the more its utilization is—

CPU utilization means that the CPU should be kept as busy as possible. Theoretically, CPU utilization can range from 0 to 100 percent. Practically, in a real system, it should range from 40 percent (in a lightly loaded system) to 90 percent (in a heavily used system)

Throughput : एक निश्चित समय में कम्प्यूटर द्वारा किया जाने वाला कार्य throughput कहलाता है। जब CPU processes को execute करता है, तो यह कार्य सम्पन्न होते हैं। अत:, per unit time में सम्पन्न कार्यों की संख्या को throughput कहा जाता है। लम्बे कार्यों हेतु throughput को प्रति घंटे में सम्पन्न किये गये कार्यों की संख्या के रूप में जबकि छोटे कार्यों हेतु प्रति सैकेन्ड में सम्पन्न किये गये कार्यों की संख्या के रूप में व्यक्त किया जा सकता है। उदाहरणत: यदि CPU द्वारा एक सैकेन्ड में 12 कार्य सम्पन्न किये जाते हैं तो उसकी throughput 12 processes per second होगी।

The number of processes that are completed per time unit, called throughput. For long processes, this rate may be process per hour while for shorter transactions, it may be measured in terms of processes per second.

Turnaround Time—

किसी process को complete होने में कितना समय लगता है, इसको turnaround time द्वारा व्यक्त किया जाता है। अर्थात् process के submission (प्रस्तुत करने) के बाद उसके completion (पूर्ण होने) में लगने वाला समय turnaround time कहलाता है।

"The interval from the time of submission of a process to the time of completion is called turn around time. Turnaround time is the sum of periods spent in waiting to get into the memory, waiting in the ready queue, execution on the CPU and doing I/O."

अत: turnaround time memory में प्रवेश के लिये इंतज़ार में लगने वाले समय, ready queue में इंतज़ार में लगने वाले समय, CPU execution में लगने वाले समय तथा I/O करने में लगने वाले समय का योग होता है।

Waiting Time—

CPU scheduling algorithm न केवल process के execution तथा I/O में लगने वाले समय को प्रभावित करती है बल्कि वह process द्वारा ready queue में इंतज़ार में लगने वाले समय को भी प्रभावित करती हैं। Ready queue में इंतज़ार करने में लगने वाली अवधियों (समयों) का योग waiting time कहलाता है।

"The sum of periods spent in waiting in the ready queue is called the waiting time."

Response Time—

कई बार process से यह अपेक्षा होती है कि उसके कुछ results जल्दी प्राप्त हो जाये तथा शेष results बाद में compute होते रहें तथा जल्दी प्राप्त होने वाले results user को उपलब्ध कराये जा सकें। अत:, ऐसी स्थितियों में turnaround time के बजाय response time criteria महत्वपूर्ण हो जाता है जो कि request की submission के पश्चात् पहली response (प्रतिक्रिया या output) प्राप्त होने के मध्य के समय को व्यक्त करता है।

"Request के submission के पश्चात् पहली response प्राप्त होने में लगने वाला समय response time कहलाता है। अत:, response time request submission के पश्चात् लगने वाला वह समय है जिसके बाद पहली response प्राप्त होती है न कि final output।"

"Response time is the time taken to produce the first response after the submission of the request. Hence, it is the time it takes to start responding, and not the time it takes to output the response."

"Response time means, the time, the system takes to start responding not the time it takes to output the response. The turnaround times is typically limited by the speed of the output device."

अत:, scheduling का मुख्य उद्देश्य होता है CPU utilization व throughput को maximize (अधिकतम) करना, तथा turnout time, waiting time व response time को minimize (न्यूनतम) करना।

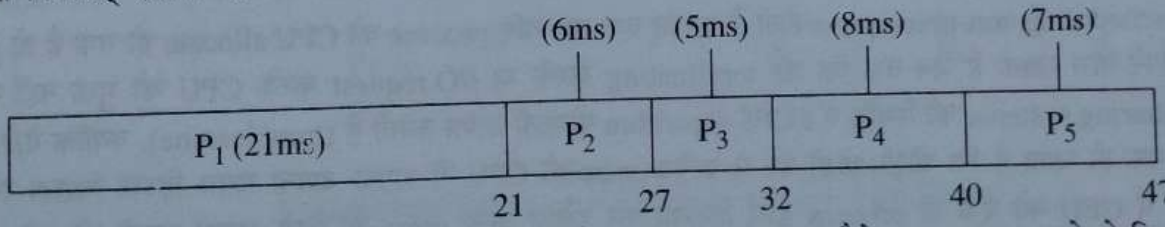
§ 4.6. शैडयूलिंग एल्गोरिद्मस (Scheduling Algorithms) :

फर्स्ट-कम, फर्स्ट सर्व्ड शैडयूलिंग (First Come-First Served Scheduling or FCFS scheduling) : First come first served का अर्थ होता है—पहले आओ, पहले पाओ। **यह सबसे सरल शैडयूलिंग एल्गोरिद्म है। FCFS algorithm के अनुसार सबसे पहले request करने वाले process को सबसे पहले CPU allocate किया जाता है, उसके अगली request को उसके बाद तथा यही क्रम चलता रहा है।** यह ठीक ऐसे ही है जैसे कि रेलवे टिकट खिड़की पर टिकट मिलते हैं, जो लाइन में सबसे पहले होता है उसे सबसे पहले सर्व किया जाता है और पंक्ति आगे बढ़ती रहती है। FCFS scheduling को एक FIFO queue (first in, first out queue) द्वारा easily manage किया जा सकता है तथा पहले Input होने वाला process, पूर्ण होकर पहले बाहर जाता है तथा क्रमवार executions होते रहते हैं। जिस प्रकार रेलवे खिड़की पर यदि कोई नया व्यक्ति आता है तो उसे पंक्ति में सबसे पीछे जाकर खड़ा होकर अपनी बारी आने का इंतजार करना होता है, इसी प्रकार FCFS scheduling में भी यदि कोई नया process ready queue में प्रवेश करता है तो उसका PCB (Process Control Block) queue की tail (सबसे पीछे का भाग, पूँछ) से link कर दिया जाता है। इस प्रकार आगे बढ़ते-बढ़ते वह queue के head (सबसे आगे का भाग) तक पहुँचता है। जब भी CPU free होता है तो queue के head पर जो भी होता है, उसे CPU allocate (आवंटित या विनिहित करना) किया जाता है। Running process को queue से remove (हटाना) कर दिया जाता है। FCFS के लिये code लिखना व समझना आसान होता है।

FCFS scheduling में average waiting time (औसत प्रतीक्षा समय) काफी अधिक होता है। गाना कि पाँच process execute होने हैं जिनका burst time निम्नवत् है—

Process	Burst time
P_1	21
P_2	6
P_3	5
P_4	8
P_5	7

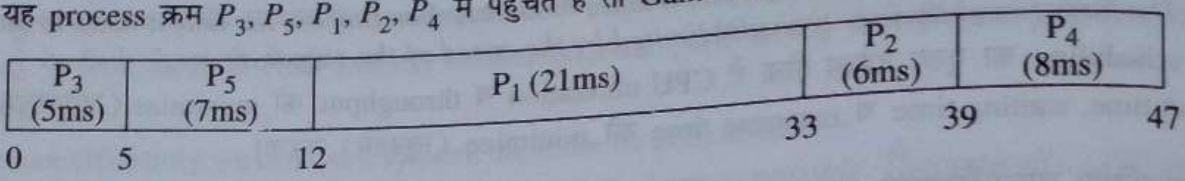
यदि यह process P_1, P_2, P_3, P_4, P_5 क्रम में arrive करते हैं (पहुँचते हैं) तो इनके execute होने के क्रम को Gantt chart द्वारा निम्नवत् दर्शाया जा सकता है—



आइये, इन process के waiting time की गणना करते हैं अर्थात् यदि यह सब प्रौसेस time 0 पर आये तो किसको कितना इंतज़ार करना पड़ा। P_1 सबसे पहले पहुँचा, इसका waiting time 0 है, P_2 को 21 ms इंतज़ार करना पड़ा, P_3 को 27 ms, P_4 को 32 ms तथा P_5 को 40 ms प्रतीक्षा करनी पड़ी। अत:

$$\text{Average waiting time} = \frac{0+21+27+32+40}{5} = \frac{130}{5} = 26.5 \text{ ms}$$

यदि यह process क्रम P_3, P_5, P_1, P_2, P_4 में पहुँचते हैं तो Gantt chart निम्नवत होगा—



अत:, इस स्थिति में

$$\text{Average waiting time} = \frac{0+5+12+33+39}{5} = \frac{89}{5} = 17.8 \text{ ms}$$

यानि इस स्थिति में waiting time पिछले case की तुलना में काफी कम है (क्योंकि इसमें बड़े प्रौसेस का arrival late हुआ है)। अत: FCFS policy में waiting time न्यूनतम नहीं होता तथा यदि processes के CPU burst time में काफी अंतर है तो उनका क्रम बदलने पर average waiting time भी चेंज हो जाता है। यदि अधिक burst time वाला process queue में आगे खड़ा है तो average waiting time अधिक हो जाता है।

FCFS scheduling की कमी समझने के लिये एक उदाहरण लेते हैं—माना कि एक situation है जहाँ एक CPU-bound process है तथा कई I/O bound process हैं। आप जानते होंगें कि CPU bound process वह होते हैं जिसमें I/O requests कम होती हैं तथा अधिकांश CPU आधारित computation होती हैं जबकि I/O bound processes I/O करने में अधिक समय व्यतीत करते हैं तथा इन्हें CPU की कम आवश्यकता होती है। ऐसी स्थिति में CPU bound process को CPU मिल जायेगा तथा वह CPU को hold कर लेगा। जबकि शेष process इस समय में अपना I/O कार्य पूरा करके CPU प्राप्त करने के लिये ready queue में खड़े हो जायेंगे। अब आप देखें कि जब तक यह process ready queue में खड़े अपनी बारी आने की प्रतीक्षा कर रहे होंगे, तब तक I/O devices idle (अर्थात् खाली) रहेंगे, जब तक CPU bound process CPU को release नहीं करता। जब CPU bound process अपनी CPU burst finish करके I/O device पर move करेगा, सभी I/O bound processes अपनी CPU burst (जो कि बहुत छोटी अवधि की होगी) को बहुत जल्दी finish कर लेंगे तथा इन्हें फिर से I/O queues में जाकर लगना होगा। इस स्थिति में CPU idle हो जायेगा।

अब CPU process वापस ready queue में जायेगा तथा फिर से इसको CPU allocate हो जायेगा। अत: फिर से I/O processes को ready queue में इंतजार करना होगा तब तक कि CPU bound process पूरा नहीं हो जाता। अत: उपरोक्त स्थिति में एक convoy* effect उत्पन्न हो जायेगा जहाँ सभी process एक बड़े process द्वारा CPU के free किये जाने का इंतजार कर रहे हैं। *(यह ठीक ऐसे ही है जैसे कि किसी स्पीड पोस्ट काउन्टर में सबसे आगे खड़ा व्यक्ति 50 पत्र लेकर खड़ा है तथा उसके पीछे 10 व्यक्ति केवल एक (या दो) पत्र लेकर अपनी बारी आने की प्रतीक्षा कर रहे हैं, किन्तु उनकी बारी तब आयेगी जब आगे वाले व्यक्ति के 50 letters post नहीं हो जाते।)*

अत: इस प्रकार की स्थिति उत्पन्न होने पर CPU तथा अन्य युक्तियों के utilization पर प्रतिकूल प्रभाव पड़ता है तथा ऐसी स्थितियों में छोटे processes को प्राथमिकता देने पर utilization में सुधार लाया जा सकता है।

FCFS scheduling non-preemptive होती है अर्थात् एक बार यदि process को CPU allocate हो गया है तो process CPU को अपने पास रखता है जब तक कि वह terminating करके या I/O request करके CPU को मुक्त नहीं कर देता। अत:, time sharing systems की स्थिति में FCFS algorithm परेशानी उत्पन्न करती है (troublesome), क्योंकि ऐसे सिस्टम्स में यह आवश्यक हो जाता है कि थोड़ी-थोड़ी देर में प्रत्येक uses को CPU में अपना-अपना समय-हिस्सा मिलता रहे। अत:, ऐसे सिस्टम्स में CPU को एक ही process द्वारा हिलका कर रखना अन्य users के लिये अच्छी खासी परेशानी का सबब बन सकता है।

In the first-come, first-served (FCFS) scheduling algorithm, the process that requests the CPU first is allocated the CPU first and this implementation is managed with a FIFO queue. When a process enters the

---

*Convoy: accompany or escort, पहरे से साथ ले जाना (जैसे कि किसी बड़े व्यक्ति की सुरक्षा के लिये एक काफिला उसके साथ चलता है)

ready queue, its PCB is linked onto the tail (the end, the last part) of the queue. When the CPU is free, it is allocated to the process at the head (the front side (अगला सिरा)) of the queue. Hence, the running process is removed from the queue. The code for FCFS scheduling is simple to write and easily understandable.

In a dynamic situation, where we may have one CPU-bound process and many I/O-bound process, the CPU-bound process will get and hold the CPU. During this period, all the other processes finish their I/O and will move into the ready queue, waiting for their chance to get the CPU. While the processes wait in the ready queue, the I/O devices become free. Now, the CPU-bound process finishes its CPU burst and moves to an I/O device. All the I/O-bound processes, which have short CPU bursts, execute quickly and move back to the I/O queues. This results in a situation where the CPU becomes free. The CPU-bound process will then move back to the ready queue and be allocated the CPU. Once more, all the I/O processes end up waiting in the ready queue until the CPU-bound process is done. This gives rise to a sort of convoy effect as all the other processes wait for the one big process to release CPU resulting in lower CPU and device utilization than might be possible if the shorter process were allowed to go first.*

The FCFS scheduling is non-preemptive. Once the CPU has been allocated to a process, that process keeps the charge of CPU until it releases the CPU, either by terminating or by requesting I/O. Hence, this algorithm can prove a nuisance for time-sharing systems, not allowing each user to get its share of the CPU at regular intervals.

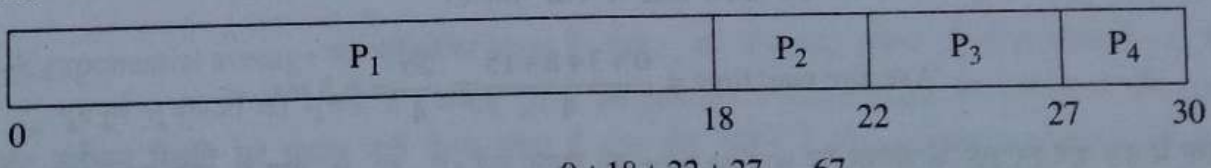
महत्वपूर्ण बिन्दु

- Jobs are executed on first come, first serve basis. (Jobs "पहले-आओ पहले पाओ" के आधार पर execute किये जाते हैं)
- Easy to understand and implement (समझने व implement करने में आसान)
- Poor in performance as average wait time is high (अधिक average wait time)

प्रश्न : FCFS scheduling के आधार पर निम्न का average wait time ज्ञात करें।

Process	Burst Time (ms)
P_1	18
P_2	4
P_3	5
P_4	3

हल : Gantt चार्ट



$$\text{Average wait time} = \frac{0+18+22+27}{4} = \frac{67}{4} = 16.75 \text{ ms}$$

अभ्यास प्रश्न

(i) यदि 6 process, $P_1, P_2, P_3, P_4, P_5, P_6$ इसी क्रम में आते हैं, तथा इनका burst time क्रमश: 20 ms, 10 ms, 5 ms, 2 ms, 11 ms तथा 23 ms है तो FCFS scheduling के आधार पर average wait time की गणना करें।

(ii) यदि उपरोक्त प्रश्न में process के आने का क्रम P_6, P_1, P_5, P_2, P_3, तथा P_4 है तो FCFS के आधार पर average wait time की गणना करें।

(iii) यदि उपरोक्त processes के आने का क्रम P_4, P_3, P_2, P_5 व P_2, P_1 है तो FCFS के आधार पर average waiting time की गणना करें।

(iv) कौन से क्रम में average wait time minimum होगा?

(v) कौन से क्रम में average wait time maximum होगा?

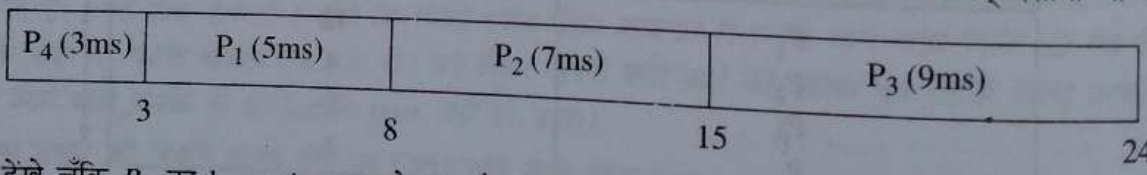
शॉर्टेस्ट-जॉब-फर्स्ट शैडयूलिंग (Shortest-Job-first Scheduling or SJF Scheduling)—

Shortest-job-first अर्थात् सबसे छोटा-सबसे पहले। **SJF algorithm में किसी process को CPU allot करने से पहले यह देखा जाता है कि उस process की अगली CPU burst कितनी अवधि की है तथा जिस process की CPU burst सबसे छोटी होती है उसको CPU allocate किया जाता है अर्थात् CPU burst समय की अवधि के बढ़ते हुये क्रम के अनुसार बारी-बारी से processes को CPU का allocation किया जाता है।** यदि दो processes की next CPU burst same होती है तो tie को ब्रेक करने हेतु FCFS scheduling का नियम अपनाया जाता है अर्थात् पहले आने वाले process को प्राथमिकता दी जाती है। चूंकि SJF scheduling में allocation process CPU की अगली burst की length पर निर्भर करता है अत: इसको shortest-next-CPU-burst algorithm भी कहा जा सकता है।

आइये, Shortest-Job-First (SJF) algorithm को समझने के लिये एक उदाहरण लेते हैं। मान लीजिये कि processes का CPU burst time निम्नवत् है—

Process	Burst time (ms)
P_1	5
P_2	7
P_3	9
P_4	3

SJF scheduling के अंतर्गत इन process की scheduling को Gantt chart द्वारा निम्नवत् दर्शाया जा सकता है—



आप देखे चूँकि P_4 का burst time सबसे कम है अत: इसको scheduling में सबसे पहले रखा गया है तथा P_3 का burst time सबसे अधिक होने के कारण इसे सबसे last में रखा गया है।

$$\text{Average wait time} = \frac{0+3+8+15}{4} = \frac{26}{4} = 6.5 \text{ ms}$$

उपरोक्त में हम यदि FCFS के आधार पर wating time की गणना करें (P_1, P_2, P_3 तथा P_4 क्रम मानते हुये) तो average wating time का मान $\frac{0+6+14+21}{4} = 10.25$ ms प्राप्त होगा। अत: हम देखते हैं कि SJF scheduling से processes का average wait time घट जाता है क्योंकि छोटे process को बड़े process से आगे खड़ा करने पर छोटे process के waiting time में आने वाली कमी बड़े process के waiting time में होने वाली बढ़ोत्तरी से अधिक होती है। आप इस बात को निम्न तालिका से भी समझ सकते हैं—

Process	Waiting Time as per FCFS	Wating Time as per SJF	कमी/अधिकता
P_1	0	3	+3
P_2	6	8	+2
P_3	14	15	+1
P_4	21	0	−21

अत: उपरोक्त तालिका से आप देख सकते हैं कि P_4 का waiting time 21 ms कम हो गया जबकि P_1, P_2 व P_3 के waiting time में मात्र क्रमश: 3, 2 व 1 ms की बढ़ोत्तरी हुई। अत: average waiting time में $\frac{-21+3+2+1}{4} = \frac{-15}{4}$ = − 3.75 ms की कमी आ गयी तथा वह 10.25 ms (FCFS) से घटकर 10.25 − 3.75 = 6.5 ms (SJF) हो गया।

हमने देखा कि SJF algorithm में CPU का allocation इस बात पर निर्भर करता है कि process की next CPU request का burst time कितना है। अब प्रश्न यह उठता है कि अगली CPU request के burst time का पता कैसे लगाया जाये कि वह कितनी अवधि की होगी। इसके लिये long term scheduling में submission के समय user द्वारा specify की गई process time limit की length को use किया जा सकता है। अत:, users को submission के समय process time limit की accurance गणना हेतु प्रेरित किया जाता है क्योंकि इसकी कम value होने पर fast response मिलेगी। यदि user ने इस value को बहुत कम specify कर दिया तथा उतने समय में कार्य पूर्ण नहीं हो पाया, तो time-limit-exceeded error के कारण re-submission करना पड़ेगा। अत: SJF scheduling को long term scheduling में अक्सर use किया जाता है।

Short term scheduling में SJF algorithm को implement नहीं किया जा सकता क्योंकि CPU की अगली burst का पता लगाना संभव नहीं है। इसका एक तरीका यह है कि next CPU burst के अनुमानित मान की गणना की जाये तथा यह अपेक्षा की जाये कि अगली burst का मान पिछली burst के लगभग बराबर होगा। अत:, अनुमान लगाकर अगली burst की गणना की जा सकती है तथा उस आधार पर सबसे छोटे burst वाले process को CPU allocate किया जा सकता है।

Next Burst का अनुमान लगाने की विधि (Method to Predict the Next Burst)—

अगली CPU burst का अनुमान लगाने हेतु पिछले CPU bursts की lengths का exponential average (चरघातांकीय औसत) किया जाता है माना कि nth CPU burst की लम्बाई t_n है, तथा अगली CPU burst की अनुमानित लम्बाई τ_{n+1} है, तो τ_{n+1} के लिये सूत्र

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n \qquad 0 \le \alpha \le 1$$

यह सूत्र exponential average को परिभाषित करता है। यहाँ t_n का मान ताज़ा सूचना को व्यक्त करता है (t_n का वर्तमान मान) तथा τ_n भूतपूर्व सूचनाओं का सूचक है (अर्थात् पूर्व में इस process के burst timer का संकेतक)। यदि $\alpha = 0$, $\tau_{n+1} = \tau_n$, अर्थात् वर्तमान स्थिति का प्रभाव नहीं लिया जाता है (या कह सकते हैं कि current conditions को transients (क्षणिकायें) माना जाता है), तथा यदि $\alpha = 1$ तो $\tau_{n+1} = t_n$ अर्थात् जो most recent (अभी-अभी घटित, सबसे नवीन) को प्राथमिकता दी जाती है तथा पुरानी स्थितियाँ irrelevant तथा old (अप्रांसगित, असंगत तथा पुरानी) मान ली जाती है। अधिकतर स्थितियों में α का मान 0.5 लिया जाता है अर्थात् वर्तमान स्थिति व पुरानी स्थिति को बराबर का महत्व देते हुये next burst का अनुमान लगाया जाता है। Initial (प्रारम्भिक) τ_0 का मान constant या overall system average लिया जाता है।

Exponential average के behavior को समझने के लिये हम τ_{n+1} के सूत्र को expand कर सकते हैं—

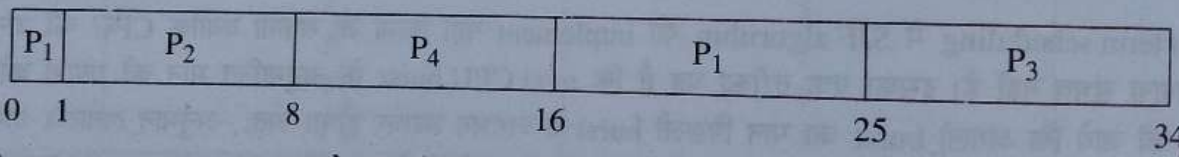
$$\tau_{n+1} = \alpha t_n + (1-\alpha)\alpha t_{n-1} + \ldots + (1-\alpha)^j \alpha t_{n-j} + \ldots + (1-\alpha)^{n+1} \tau_0$$

चूँकि दोनों α तथा $1-\alpha$ का मान 1 या 1 से कम रहता है, अत: प्रत्येक successive term (next term) का weight (भार) predecessor (अर्थात् पिछली term) से कम होता है।

SJF algorithm preemptive या non-preemptive हो सकती है। चुनाव की आवश्यकता तब पड़ती है जब ready queue में एक नया process प्रवेश करता है जबकि कोई पिछला process पहले ही execute हो रहा होता है। ऐसा संभव है कि नये process का CPU burst पहले से execute हो रहे process के अवशेष समय से कम हो। ऐसी स्थिति उत्पन्न होने पर preemptive SJF algorithm currently running process को preempt कर देगी तथा इस नये process को मौका प्रदान करेगी जबकि एक non-preemptive SJF algorithm पहले currently running process को अपना काम finish करने का मौका देगी। Preemptive SJF को shortest-remaining-time-first (अर्थात् shortest-अवशेष-समय-प्राथमिकता भी कहा जाता है)। उदाहरणत:; मान लीजिये कि चार processes का burst time निम्नवत है—

Process	Arrival Time	Burst Time (ms)
P_1	0	10
P_2	1	7
P_3	2	9
P_4	3	8

यदि process ready queue में तालिका में दिये गये समय के अनुसार प्रवेश करते हैं तो उनकी preemptive SJF schedule निम्न Gantt chart के अनुसार होगी—



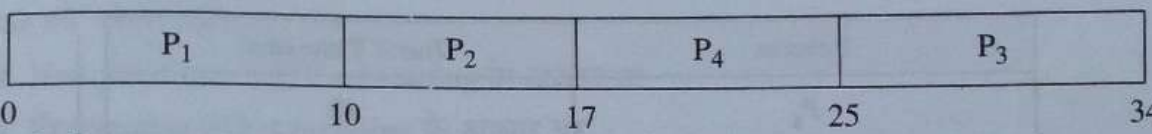
आइये इस Gantt chart का समझने की कोशिश करते हैं। समय 0 पर P_1 आया तथा इसको CPU allocate हो गया क्योंकि वह सबसे पहले आया तथा उस समय कोई अन्य दावेदार नहीं था। समय $t = 1$ ms पर P_2 प्रकट हुआ। यहाँ आप देखें कि P_1 का शेष $10 - 1 = 9$ ms का burst बचा है जबकि P_2 का burst समय 7 ms अर्थात् इस अवशेष समय से कम है, अत: P_1 को preempt करके P_2 को CPU allocate कर दिया गया। $t = 2$ ms पर P_3 प्रकट हुआ, पर इसका burst 9 ms है जबकि P_2 का अवशेष समय 7 ms – 1 ms = 6 ms है, अत: P_2 ही execute होता रहेगा, इसी प्रकार $t = 3$ ms पर P_4 आया, इसका burst भी P_2 के अवशेप समय से अधिक है, अत: P_2 ही execute होता रहेगा। P_2 के $t = 8$ ms पर complete होने के पश्चात् तीन process बचे, P_1, P_3, व P_4 तथा इनके remaining burst क्रमश: $10 - 1 = 9$ ms, 9 ms व 8 ms है। चूँकि इन तीनों में से P_4 का burst सबसे कम है, अत: अब P_4 को मौका मिलेगा। $t = 16$ ms पर P_4 के complete होने के पश्चात् P_1 तथा P_3 बचे तथा दोनो के अवशेष समय 9 ms है किन्तु चूँकि P_1 पहले आया था इसलिये यहाँ Tie को break करने के लिये FCFS नियम लगेगा तथा P_1 को पहले अवसर दिया जायेगा तथा $t = 25$ ms पर अंतत: P_3 की बारी आयेगी।

उपरोक्त उदाहरण में average wating time के मान की गणना निम्नवत् की जा सकती है—

$$\text{Average wating time} = \frac{(16-1)+(1-1)+(25-2)+(8-3)}{4}$$

$$= \frac{15+0+23+5}{34} = \frac{43}{4} = 10.75 \text{ ms}$$

यदि इसके स्थान पर non-preemptive scheduling use की जाती तो average Gantt chart निम्नवत् होता—



इस स्थिति में—

$$\text{Average waiting time} = \frac{0+(10-1+(17-2)+(25-3)}{4}$$

$$= \frac{9+15+22}{4} = \frac{46}{4} = 11.50 \text{ ms}$$

The shortest-job-first or SJF algorithm associates with each process the length of the process's next CPU burst. The CPU assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are of equal time length, FCFS scheduling criteria is used as a tie-breaker.

The SJF scheduling algorithm is provably optimal, because it gives the minimum average waiting time. Moving a short process ahead in the queue by giving priority to the shorter one, decreases the waiting time of the shorter process more than it increases the waiting time of the long process. Hence, the average waiting time decreases.

The difficulty which arises here is in knowing the length of the next CPU request. For long-term scheduling in the length the process time limit that a user specifies when he submits the job can be used. Thus, users are inspired to estimate the process time limit more accurately and since a lower value may mean faster response.

Although the SJF algorithm is optimal, it cannot be implemented at the level of short-term CPU scheduling. We may not know the length of next CPU burst, but we may be able to predict its value.

The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts. We define—

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n, \qquad 0 \le \alpha \le 1$$

where, t_n = *length of nth CPU burst*

τ_{n+1} = *predicted value of next CPU burst*

τ_n = *denotes past history*

This formula defines an exponential average.

If $\alpha = 0$, *then* $\tau_{n+1} = \tau_n$, *and recent history has no effect; if* $\alpha = 1$, *then* $\tau_{n+1} = t_n$, *and only the most recent CPU burst matters commonly,* $\alpha = 1/2$ *giving equal importance to recent history and past history. The intial* τ_0 *can be defined as a constant or as an overall system average.*

The SJF algorithm can be made either preemptive or non-preemptive. When a new process arrives at the ready queue while a previous process is still executing and if the next CPU burst of the newly arrived process is shorter than what is left of the currently executing process, than, a preemptive SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will not take the CPU back and allow the currently running process to finish its CPU burst.

Shortest Jobs-First (SJF) Scheduling :

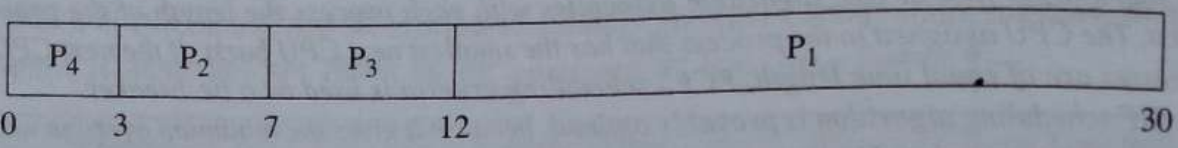
- Best approach to minimize waiting time (Waiting time को न्यूनतम करने हेतु best approach.)
- Actual time taken by the process should known to processor in advance (Process द्वारा लिया जाने वाला समय प्रौसेसर को पहले से पता होना चाहिये)
- Impossible to implement.

प्रश्न—SJF शैडयूलिंग के आधार पर निम्न का average wait time ज्ञात करें।

Process	Burst Time (ms)
P_1	18
P_2	4
P_3	5
P_4	3

हल—In shortest job first scheduling, the shortest process is executed first.

उपरोक्त का Gantt Chart—



$$\text{Average waiting time} = \frac{0+3+7+12}{4} = \frac{22}{4} = 5.5 \text{ ms}$$

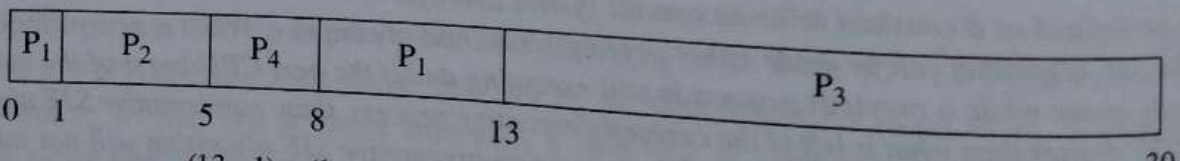
प्रश्न—What is preemptive SJF?

उत्तर—In preemptive shortest job first scheduling, jobs are put into ready queue as they arrive, but as a process with short burst time arrives, the existing process is preempted अर्थात् preemptive scheduling में यदि नये process का burst time currently running process के अवशेष burst time से कम है, तो CPU currently running process से लेकर नये process को allocate कर दिया जाता है।

प्रश्न—Preemptive SJF के आधार पर निम्न का average wait time ज्ञात करें—

Process	Burst Time (ms)	Arrival Time (ms)
P_1	18	0
P_2	4	1
P_3	5	2
P_4	3	3

हल—Gantt chart



$$\text{Average wait time} = \frac{(13-1)+(1-1)+(8-2)+(5-3)}{4} = \frac{20}{4} = 5 \text{ ms}$$

विचार प्रश्न:

(i) **FCFS, preemptive SJF व non-preemptive SJF में से सबसे अधिक time किसका होता है? सबसे कम wait time किसका होता है?**

(ii) **चार प्रौसेस P_1, P_2, P_3, P_4 के burst समय क्रमशः 20 ms, 10 ms, 2 ms तथा 5 ms हैं। SJF scheduling के आधार पर इनके average wait time की गणना करें।**

(iii) पाँच प्रौसेस P_1, P_2, P_3, P_4, P_5 का burst time व arrival time निम्नवत् दिया गया है। इनके average wait time की गणना करें–

(a) Non-preemptive SJF scheduling के आधार पर

(b) Preemptive SJF scheduling के आधार पर

(c) FCFS scheduling के आधार पर

Process	Burst Time	Arrival Time
P_1	15 ms	0
P_2	11 ms	1
P_3	5 ms	2
P_4	2 ms	3
P_5	8 ms	4

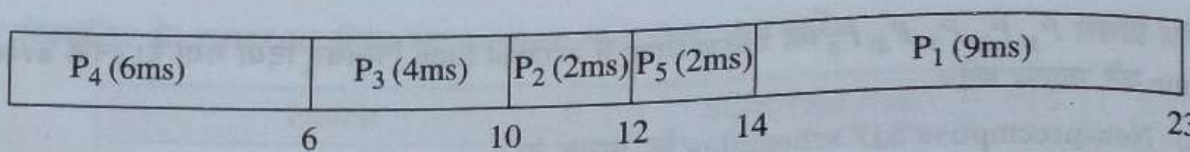
प्रायरटी शैडयूलिंग (Priority Scheduling)—

Priority का अर्थ होता है प्राथमिकता, अर्थात् **यदि शैडयूलिंग प्राथमिकता के आधार पर की जायें तो इसे priority scheduling कहा जाता है। इस प्रकार की scheduling में प्रत्येक process के साथ एक priority attach कर दी जाती है तथा उच्चतम प्राथमिकता (highest priority) वाले process को CPU allocate किया जाता है। यदि दो processes की priority same है तो FCFS (first come first served) क्रम में allocation किया जाता है।** Priority निर्धारित करने हेतु numbers का use किया जाता है तथा इनकी range भी निर्धारित होती है (0 से 7, 0 से 15, 0 से 31, 0 से 63, 0 से 127 इत्यादि (2 की घात में)), अर्थात् चूंकि कम्प्यूटर आधारित गणनायें binary में होती हैं, अत: उपरोक्त ranges भी उसी अनुसार है, जैसे कि 0 से 7 में 8 संख्यायें होती है ($2^3 = 8$) इत्यादि। यह भी आवश्यक नहीं है कि 0 की priorty सबसे कम मानी जायेगी या सबसे अधिक अर्थात कुछ कम्प्यूटर्स low number को lower priority व्यक्त करने हेतु प्रयुक्त करते हैं, जबकि कुछ कम्प्यूटर्स lower number को higher priority use करने के लिए use करते हैं। उदाहरणत: यदि किसी कम्प्यूटर में lower number lower priority के लिये use किया जाता है तो 2 वाले की priority 1 से अधिक होगी जबकि यदि lower number higher priority को indicate करता है तो 1 वाले की priority 2 से अधिक होगी।

उदाहरण के तौर पर, मान लीजिये कि 5 process P_1, P_2, P_3, P_4, P_5 टाइम 0 पर arrive करते हैं तथा उनकी priority तथा burst time निम्नवत् हैं—

Process	Burst time (ms)	Priority
P_1	9	5
P_2	2	3
P_3	4	2
P_4	6	1
P_5	2	4

यहां हम यह assume कर रहे हैं कि lower संख्या higher priority को indicate कर रही है अर्थात् 1 priority सर्वाधिक है तथा 2 priority 1 के बाद है इत्यादि। अत:, उपरोक्त प्रोसैस priority के अनुसार execute होगें अर्थात् सबसे पहले P_4 execute होगा, क्योंकि इसकी priority सर्वाधिक है (1), फिर P_3 execute, तत्पश्चात् क्रमानुसार P_2, P_5 तथा P_1 execute होंगे। अत:, इस scheduling को हमे निम्न Gantt chart द्वारा व्यक्त कर सकते हैं—



अत: उपरोक्त के अनुसार—

$$\text{Average waiting time} = \frac{0+6+10+12+14}{5} = \frac{42}{5} = 8.4 \text{ ms}$$

Priorites को internally या externally define किया जा सकता है। Internally defined priorites की स्थिति में priority की गणना करने हेतु कई measurable quantities (वह राशियां जिनका मापन करना संभव है) को पैमाना बनाया जाता है जैसे कि time limits, memory requirements, number of open files, average I/O burst तथा average CPU burst का अनुपात इत्यादि। External (बाहरी) priorities set करने हेतु operating system के बाहर के criteria use किये जाते हैं जैसे कि प्रौसेस का महत्व (importance of process), computer use करने के लिये किये जाने वाले funds (भुगतान), कार्य को sponsor (प्रायोजित) करने वाले विभाग इत्यादि।

Priority scheduling preemptive या non-preemptive हो सकती है। जब कोई process ready queue में प्रवेश करता है तो उसकी priority पहले से run कर रहे process से compare की जाती है। यदि नये आगन्तुक (अर्थात् जिस process ने ready queue में अभी-अभी प्रवेश किया है) की priority run कर रहे process से अधिक है तो preemptive priority scheduling की स्थिति में running process को preempt करके CPU का allocation इस नये किन्तु अधिक priority वाले process को दे दिया जाता है जबकि non-preemptive priority scheduling की स्थिति में new process को ready queue के head पर डाल दिया जाता है तथा current process complete हो जाने के बाद ही उसको CPU allocate हो पाता है।

Priority scheduling में होने वाली एक मुख्य समस्या indefinite blocking या starvation* है अर्थात् low priority processes को एक लम्बे समय तक अपनी बारी आने का इंतजार करना पड़ता है तथा heavily loaded systems में higher priority processes में लगातार आने पर यह भी संभव हो सकता है कि lower priority process का इंतज़ार कभी खत्म ही न हो तथा उन्हें अनंत काल तक इंतज़ार ही करना पड़े।

Indefinite blocking की समस्या का हल है ageing। Ageing वह तकनीक है जिसके द्वारा लम्बे समय तक wait करने वाले processes की priority धीरे-धीरे बड़ा दी जाती है ताकि उनकी भी बारी आ जाये तथा वह भी CPU का स्वाद चख सकें। उदाहरणत:, एक 0–127 प्राइरटी सिस्टम में यदि किसी process की priority 127 है तथा प्रत्येक 10 मिनट में उसकी priority एक कदम आगे बढ़ा दी जाती है तो 21 घण्टे 20 मिनट बाद $\left(\because \frac{1280}{60} = 21\frac{1}{3}\right)$ वह priority 0 पर पहुँच जायेगा तथा execute हो जायेगा और indefinite blocking या starvation से बच जायेगा।

ध्यान दें कि SJF algorithm भी priority scheduling algorithm का ही special case है। चूंकि SJF algorithm में process को CPU allocate करने हेतु उसका burst time देखा जाता है, तथा जिसका burst time सबसे कम होता है उसी को CPU allocate किया जाता है अत: हम कह सकते हैं कि SJF algortihm में burst time ही priority indicate करता है तथा priority P next burst time का inverse है, यानि जिसका next CPU burst न्यूनतम है उसकी priority सर्वाधिक है।

In priority scheduling, a priority (प्राथमिकता) is associated (सम्बद्ध की जाती है) with each process, and the CPU is allocated to the process with the highest priority (सर्वोच्च प्राथमिकता) i.e., every time the CPU is available, it is alloted to the process having the highest priority among those waiting in the ready queue. Equal-priority processes are decided in FCFS order.

* Starvation = भुखमरी

Priorities are generally indicated by some fixed range of numbers, such as 0 to 7 or 0 to 1023 etc. Some systems use low numbers to represent low priority; others use low numbers for high priority. Priorities can be defined internally or externally as per requirements. Internally defined priorities use some measurable quantities to computer the priority e.g., time limits, memory requirements, the number of open files, the ratio of average I/O burst to average CPU burst have been used in computing priorities etc. External priorities can be set by criteria outside the operating system e.g., importance of the process, the type and amount of funds being paid for computer use, the department sponsoring the work, and some other factors which may differ as per need.

Priority scheduling may either preemptive or non-preemptive. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process while non-preemptive priority scheduling will put the new process at the start of the ready queue.

The main problem with priority scheduling algorithms is indefinite blocking, or starvation. A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low-priority processes waiting indefinitely. Particularly, in a heavily loaded computer system, a steady stream of higher-priority processes.

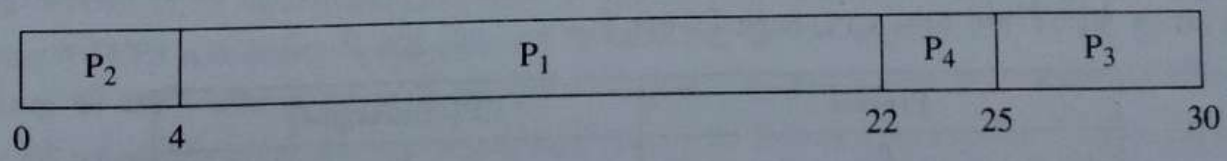
A solution to this problem is aging which is a technique of gradually increasing the priority of processes that wait in the system for a long time.

- Priority is assigned for each process (प्रत्येक process से प्राथमिकता सम्बद्ध की जाती है)
- Process with highest priority is executed first and so on (सर्वोच्च प्राथमिकता वाला process सबसे पहले execute किया जाता है तथा यह क्रम चलता रहता है)
- Processes with same priority are executed in FCFS manner (समान प्राथमिकता वाले processes का निर्णय First Come-First Served (FCFS) के आधार पर लिया जाता है)
- Priority can be decided based on memory requirements, time requirements or any other resource requirement (internally) or funds, importance, sponsor (externally).

प्रश्न—Priority scheduling के आधार पर निम्न का average wait time ज्ञात करें।

Process	Burst Time	Priority
P_1	18	2
P_2	4	1
P_3	5	4
P_4	3	3

हल—Gantt chart



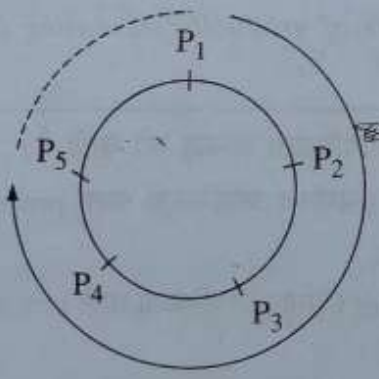
$$\text{Average wait time} = \frac{0+4+22+25}{4} = \frac{51}{4} = 12.75 \text{ ms}$$

अभ्यास प्रश्न

पाँच processes जिनका burst time क्रमशः 15, 5, 3, 12 तथा 9 ms है तथा जिनकी प्राथमिकता क्रमशः 2, 3, 5, 1, 4 है, के Priority Scheduling के आधार पर Average wait time की गणना करें। उपरोक्त waiting time की तुलना SJF algorithm के आधार पर प्राप्त waiting time से करें।

राउन्ड रॉबिन शैडयूलिंग (Round Robin Scheduling Algorithm)—

राउन्ड रॉबिन शैडयूलिंग एल्गोरिद्म विशेषतया time sharing systems हेतु design की गई है। यह FCFS के समान है किन्तु इसमें processes के मध्य preemption की व्यवस्था की जाती है। समय की एक छोटी इकाई, जिसको time quantum (या time slice) कहा जाता है, परिभाषित की गई है जिसका मान लगभग 10 से 100 मिलीसैकेन्ड रखा जाता है। Ready queue के एक circular queue के रूप में treat किया जाता है। CPU scheduler queue के सभी सदस्यों को बारी-बारी से अधिकतम 1 quantum का समय allocate करता है। यदि process इतने समय में complete नहीं हो पाता तो उस process को preempt करके queue में next member को CPU allocate कर दिया जाता है। यह process तब तक चलता रहता है तब तक process complete नहीं हो जाता।



सबको अधिकतम 1 quantum समय दिया जाता है, यदि उतने समय मे कार्य पूर्ण नहीं हो पाता तो उस process को preempt करके CPU queue के अगले सदस्य को allocate कर दिया जाता है।

चित्र 4.3—Round Robin Scheduling

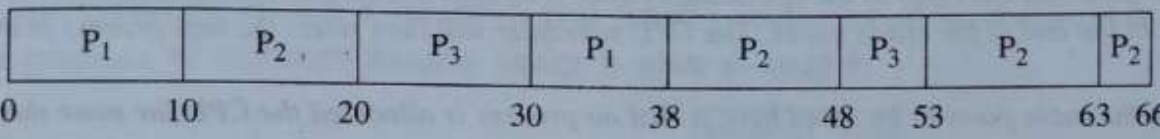
RR शैडयूलिंग (Round Robin Scheduling) को implement करके हम ready queue को FIFO (First in First Out) queue के रूप में treat करते हैं। CPU scheduler ready queue के पहले process को CPU allocate करता है तथा 1 time quantum पश्चात् interrupt करने हेतु timer को set कर दिया जाता है।

अतः, दो स्थितियाँ उत्पन्न हो सकती हैं। यदि process का CPU burst 1 time quantum से कम है (उदाहरणतः, यदि time quantum 10 ms है तथा उस process की CPU burst मात्र 6 ms है) तो process स्वयं ही CPU को release कर देगा और scheduler queue में खड़े अगले process को CPU allocate कर देगा। यदि process का CPU burst एक time quantum से अधिक है (उदाहरणतः यदि time quantum 10 ms है तथा उस process का CPU burst 18 ms है) तो timer 10 ms बाद interrupt उत्पन्न करेगा, context switch execute होगा तथा process को queue की tail पर डाल दिया जायेगा तथा queue के next member को CPU allocate कर दिया जायेगा।

RR scheduling का average waiting time अपेक्षाकृत अधिक होता है। उदाहरणतः मान लीजिये निम्न तीन process time 0 पर arrive करते हैं तथा इनकी CPU burst निम्नवत् है—

Process	CPU Burst (ms)
P_1	18
P_2	33
P_3	15

यदि time quantum 10 ms का है तो P_1 को शुरू में 10 ms मिलेंगे। फिर उसको queue के अंत में रख दिया जायेगा तथा CPU P_2 को allocate कर दिया जायेगा। 10 ms के बाद CPU P_3 को allocate कर दिया जायेगा। अब फिर से P_1 की बारी आयेगी तथा चूँकि अब P_1 का केवल 8 ms का burst time अवशेष है, अत: वह 8 ms बाद स्वयं ही CPU को release कर देगा। यह क्रम चलता रहेगा जब तक सभी processes पूरे नहीं हो जाते। Scheduling का Gantt chart निम्नवत् रहेगा—



उपरोक्त स्थिति में—

Waiting time of P_1 = 10 + 10 = 20 ms
Waiting time of P_2 = 10 + 18 + 5 = 33 ms
Waiting time of P_3 = 20 + 18 = 38 ms

अत:, Average waiting time $= \frac{20+33+38}{3} = \frac{91}{3} = 30.33$

RR scheduling में किसी भी process को एक बार में 1 time quantum से अधिक समय के लिये CPU allocate नहीं किया जाता (केवल उस स्थिति को छोड़कर जब वह queue में अकेला खड़ा है)। यदि process की CPU burst 1 time quantum से अधिक है तो process को preempt करके ready queue में सबसे पीछे खड़ा कर दिया जाता है। अत: RR algorithm preemptive algorithm है।

यदि ready queue में n-process हैं तथा time quantum q है तो प्रत्येक process को CPU time का कम से कम $\frac{1}{n}$ वाँ हिस्सा प्राप्त होता है (अधिकतम q time units के टुकड़ों में)। प्रत्येक process को अगले time quantum की प्राप्ति हेतु अधिकतम $(n-1)q$ समय तक इंतजार करना होता है उदाहरणत: यदि 4 processes है तथा time quantum 20 ms है तो प्रत्येक process को प्रत्येक 80 ms में कम से कम 20 ms मिलेंगे तथा अगले time quantum हेतु अधिकतम (4 – 1) 20 = 60 ms इंतजार करना होगा।

RR algorithm की performance time quantum के size पर निर्भर करती है। बहुत अधिक size का time constant होने पर RR algorithm लगभग FCFS के समान हो जाती है (यदि time quantum सबसे अधिक CPU burst से भी बड़ा हो)। बहुत कम size का time quantum (माना 1 ms) होने पर इसको processor sharing कहा जाता है तथा ऐसा प्रतीत होता है (theoretically) जैसे कि प्रत्येक process का अपना एक स्वतंत्र processor है जो कि real processor की $\frac{1}{n}$ स्पीड से run कर रहा है।

यहाँ एक बात और ध्यान देने योग्य है कि RR scheduling की performance context switching में लगने वाले समय द्वारा भी प्रभावित होती है। यदि context switch time time, quantum का 10% है तो CPU time का 10% context switching में खर्च हो जायेगा।

Turnaround time भी time quantum पर निर्भर करता है। यदि अधिकांश process अपने कार्य को एक ही time quantum में निबटा लें, तो turnaround time बेहतर हो जाता है। लेकिन time quantum बहुत अधिक कर देने पर RR scheduling व FCFS scheduling में कोई अंतर नहीं रह जाता। मोटे तौर पर यदि देखा जाये, तो यदि 75 या 80 प्रतिशत् CPU bursts की अवधि time quantum से कम होने पर system performance को बेहतर बनाया जा सकता है।

The RR scheduling algorithm is designed particularly for time-sharing systems in which preemption is added to switch between processes in a FCFS manner. A small unit of time, called a time quantum or time slice (generally from 10 to 100 milliseconds is defined). The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to a maximum of 1 time quantum.

New processes are added to the tail (अन्तिम सिरा) of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

If the process has a CPU burst of less than 1 time quantum. The process itself releases the CPU and scheduler proceeds to the next process in the ready queue. Otherwise, after 1 time quantum, the timer goes off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the end of the ready queue. The CPU scheduler will then select the next process in the ready queue.

Hence, the main point to be noted here is that no process is allocated the CPU for more than 1 time quantum in a row (unless it is the only runnable process). After that process is preempted (वापस प्राप्त करना) and is put back at the tail of the ready queue. The RR scheduling algorithm is thus preemptive.

If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units. Each process must wait no longer than $(n-1) \times q$ time units until its next time quantum.

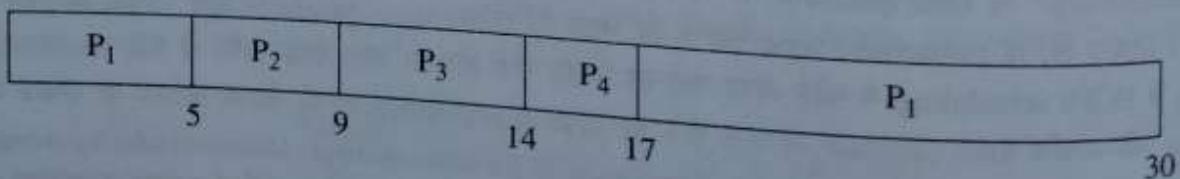
The performance of the RR algorithm depends heavily on the size of the time quantum. If the time quantum is extremely large, the RR policy is almost the same as the FCFS policy. If the time quantum is extremely small, the RR approach is called processor sharing and theoretically creates the impression that each of n processes has its own processor running at 1/n the speed of the real processor.

- A fixed time is allotted to each process, called quantum, for execution (प्रत्येक process को processing हेतु एक निश्चित time quantum allot किया जाता है)
- Once a process is executed for given time period that process is preempted and other process executes for given time period (Time पूरा होने पर उससे CPU वापस लेकर next process को दिया जाता है तथा यह प्रक्रिया एक circular queue के रूप में चलती रहती है)।
- Context switching is used to save states of preempted processes (Preempted process के state save करने हेतु context switch का use किया जाता है)

प्रश्न—राउन्ड रॉबिन शैडयूलिंग के आधार पर निम्न का average wait time ज्ञात करें (time quantum = 5 ms)

Process	Burst Time (ms)
P_1	18
P_2	4
P_3	5
P_4	3

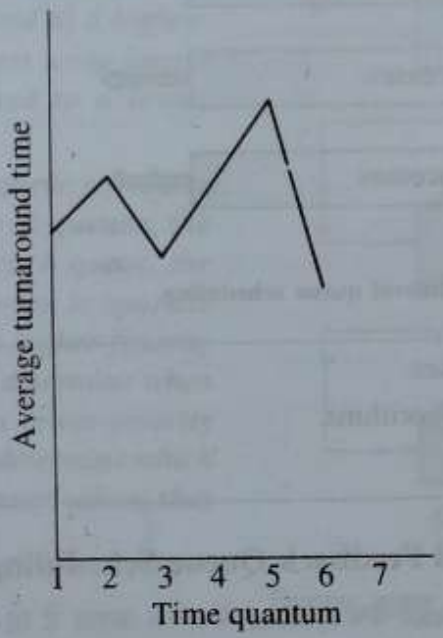
हल—Gantt chart



$$\text{Average wait time} = \frac{(17-5)+5+9+14}{4} = \frac{40}{4} = 10 \text{ ms}$$

मल्टीलैवल क्यू शैडयूलिंग (Multilevel Queue Scheduling)—

कई बार ऐसा होता है कि process को ग्रुप्स में वर्गीकृत (classify) किया जा सकता है। उदाहरणत: foreground processes (batch processes) तथा background process (interactive processes)। इन दो प्रकार के processes की responses time आवश्यकतायें भिन्न होती हैं तथा इसलिये इनकी शैडयूलिंग आवश्यकतायें भी भिन्न होती है। इसके साथ-साथ foreground processes की प्राथमिकतायें बैकग्राउन्ड प्रौसेसेज़ से अधिक हो सकती हैं।



Process	Time
P_1	6
P_2	3
P_3	1
P_4	7

चित्र 4.4—The way in which turnaround time varies with the time quantum

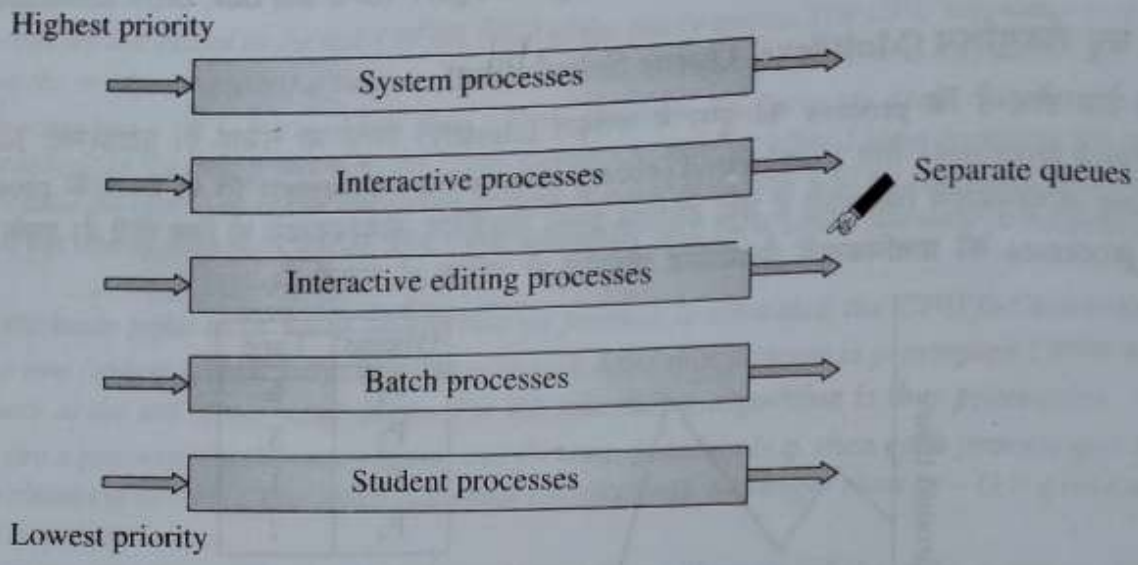
एक मल्टीलैवल क्यू शैडयूलिंग एल्गोरिद्म में रैडी क्यू को कई अलग-अलग पंक्तियों में विभाजित कर दिया जाता है। प्रत्येक क्यू को processes permanently assign (कार्य निर्धारित करना, कार्य सौंपना) कर दिये जाते हैं जो कि process की किसी property (विशेषता या गुण) पर निर्भर करते हैं (जैसे कि memory size, process priority, process size इत्यादि)। प्रत्येक क्यू की अपनी अलग शैडयूलिंग एल्गोरिद्म होती है। इसके साथ-साथ क्यूज की शैडयूलिंग भी किसी algorithm के आधार पर की जाती है जो कि सामान्यत: fixed priority preemptive scheduling होती है।

उपरोक्त के उदाहरण हेतु five queues को consider करते हैं (चित्र 4.5) आप देंखे कि system processes वाली queue की priority सर्वाधिक है तथा प्रत्येक queue की priority निर्धारित है। अत:, जब तक उच्च priority वाली क्यूज के process complete नहीं हो जाते, उस से निम्न priority वाली queues का कोई process run नहीं कर सकता। यदि प्रक्रिया को थोड़ा लचीला बनाना है, तो क्यूज़ को time slice दिये जा सकते हैं जिसका कि queue अपने विभिन्न processes के मध्य विभाजित कर सकती है।

When processes can be easily classified into different groups, multilevel queue scheduling can be used. e.g., a common partition is made between foreground processes and background processes.

A multilevel queue scheduling algorithm divides the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property (विशेषता या गुण) of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm (i.e., FCFS, SJF, priority or RR as the case may be)

There may be scheduling among the queues to which is generally implemented as fixed-priority preemptive scheduling.



चित्र 4.5—Multilevel queue scheduling

- Multiple queues are maintained for processes.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

मल्टीलैवल फीडबैक क्यू शैडयूलिंग (Multilevel Feedback Queue Scheduling)—

सामान्यत: multilevel queue scheduling में process जब system में प्रवेश करता है तो उसे queue permanently assign कर दी जाती है तथा उसके पश्चात् वह अपनी queue change नहीं कर सकता। इस प्रक्रिया में scheduling overhead कम होता है किन्तु लचीलेपन का अभाव होता है।

Multilevel feedback queue scheduling algorithm process को queue change करने की सुविधा प्रदान करती है। सामान्यत: प्रौसेस को CPU burst के अभिलक्षणों के अनुसार अलग-अलग किया जाता है। यदि कोई process अधिक CPU time खर्च करता है, तो उसको low priority वाली queue में move कर दिया जाता है। I/O bound तथा interactive processes को higher priority वाली क्यूज़ में move किया जा सकता है। इसी प्रकार, यदि किसी प्रौसेस को low priority वाली क्यू में अधिक समय तक इंतजार करना पड़ रहा है तो उसे उच्च priority वाली क्यू में move किया जा सकता है। इस प्रकार की ageing से starvation की समस्या उत्पन्न नहीं होती।

सामान्यत:, एक multiple feedback queue schedular को निम्न parameters द्वारा परिभाषित किया जा सकता है—

(i) Number of queue अर्थात् पंक्तियों की संख्या।

(ii) Scheduling algorithm for each queue अर्थात् प्रत्येक पंक्ति के लिये use की जाने वाली शैडयूलिंग एल्गोरिद्म।

(iii) Method to determine the upgradation (promotion) of a process अर्थात् किस विधि से किसी प्रौसेस का अपग्रेडेशन करके उसे उच्च प्राथमिकता वाली पंक्ति में भेजा जायेगा।

(iv) Method to determine the demotion of a process अर्थात् किसी विधि से किसी प्रौसेस का डिमोशन करके उसे निम्न प्राथमिकता वाली पंक्ति में भेजा जायेगा

(v) Method to determine which queue a process will enter अर्थात् जब कोई प्रौसेस सेवा चाहता है तथा वह सिस्टम में प्रवेश करता है तो किस विधि से यह निर्धारित किया जायेगा कि उसे किस क्यू में भेजा जाये।

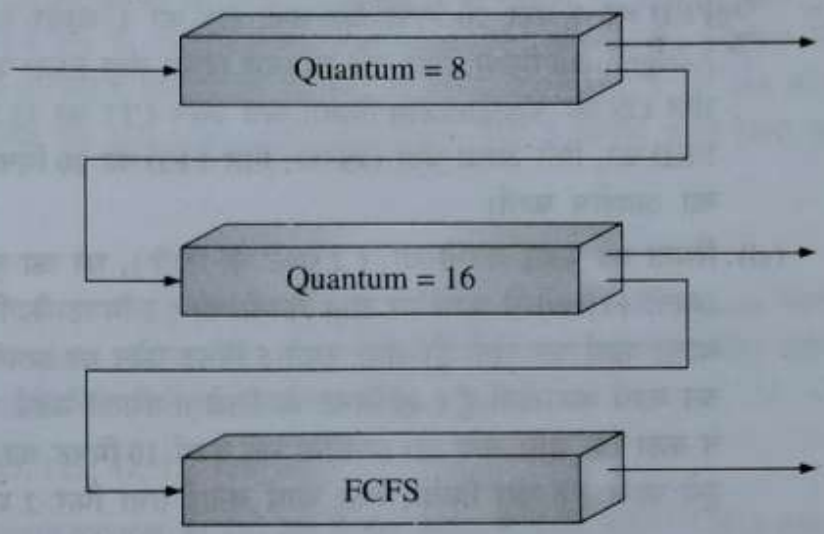
हालांकि multilevel feedback queue scheduling algorithm एक general algorithm है तथा design की आवश्यकताओं के अनुसार इसे configure किया जा सकता है किन्तु यह अत्यंत जटिल एल्गोरिद्म भी है।

Best scheduler को define करने हेतु सभी parameters की value select करने के लिए किसी न किसी विधि की आवश्यकता तो पड़ेगी ही।

When the multilevel queue scheduling algorithm is used, processes are permanently assigned to a queue when they enter the system, and can not move from one queue to the other.

However, multilevel feedback-queue scheduling algorithm, allows a process to move between queues. A process that waits too long in a lower-priority queue may be moved to a higher-priority queue and a process using larger CPU time can be demoted to a lower priority queue.

A multilevel feedback-queue scheduler is defined by the number of queues, the scheduling algorithm for each queue, the method used to determine when to upgrade or promote a process to a higher-priority queue, the method used to determine when the demote a process to a lower-priority queue, the method used to determine which queue a process will enter when that process needs service, etc.



चित्र 4.6—Multilevel feedback queues

विचार प्रश्न

निम्नलिखित दिये गये कथन किसी न किसी scheduling algorithm से analogy (समानता) रखते हैं: आप इन कथनों को ध्यानपूर्वक पढ़ें तथा बतायें कि यह किस scheduling algorithm से समानता (analogy) व्यक्त करते हैं अर्थात् FCFS, SJF, Preemptive SJF, Non-premptive SJF, Priority, Round Robin इत्यादि।

(i) साधना को उसकी बहन ने कहा कि मुझे होमवर्क करने में help कर दो, उसके पापा ने उसे कहा कि मुझे एक कप चाय बना दो तथा उसकी मम्मी ने कहा कि मुझे Kitchen में थोड़ी help कर दो। साधना ने अपने पापा के कार्य को प्राथमिकता देते हुये पहले उन्हें चाय बनाकर दी, फिर अपनी मम्मी के बताये कार्य को प्राथमिकता देते हुए उन्हें Kitchen में help करायी तथा फिर अपनी बहन को होमवर्क करने में help करायी।

(ii) रवि को उसकी मम्मी ने कहा कि बाजार से जाकर कुछ सामान ले आओ, उसके Friends ने कहा कि आओ हमारे साथ आकर Cricket खेलो तथा उसकी बहन ने कहा कि मुझसे Maths का एक प्रश्न हल नहीं हो पा रहा, इसे हल करके समझा दो। चूँकि Maths के प्रश्न हल करने का कार्य मात्र पाँच मिनट का था, अतः उसने सबसे पहले प्रश्न को हल किया, उसके बाद वह बाज़ार से सामान लेकर आया (जिसमें लगभग 20 मिनट का समय लगा) तथा फिर अपने दोस्तों के साथ क्रिकेट खेलने चला गया (लगभग 1 घण्टे के लिये)।

(iii) एक पुस्तक विक्रेता के पास दो ग्राहक आये। पहले ग्राहक ने चार नोटबुक, दो पैन व एक फोल्डर माँगा। कुछ देर बाद दूसरा ग्राहक आया जिसने प्रथम वर्ष की FED की पुस्तक माँगी। पुस्तक विक्रेता ने जो ग्राहक पहले आया था उसे पहले सर्व किया तथा उसे स्टेशनरी दी। उसके बाद दूसरे ग्राहक को (जो बाद में आया था, FED की पुस्तक लाकर दी)।

(iv) शालिनी को अपना operating system का एसाएनमैंट करना था (लगभग एक घण्टे का) तथा वह अपने लैपटॉप पर मूवी भी देखना चाहती थी (लगभग दो घण्टे की) उसने सोचा कि चूँकि Assignment करने में कम टाइम की आवश्यकता थी, इसलिये उसने सोचा कि पहले assignment कर लिया जाये तथा फिर

मूवी देखी जाये। अभी उसे assignment करते हुये 20 मिनट ही हुये थे कि उसकी मम्मी ने उसे चाय बनाने को कहा। चूँकि चाय बनाने में केवल 5 मिनट का वक्त लगना था किन्तु assignment complete करने में अभी 40 minute का समय अवशेष था, अत: वह assignment करना छोड़ कर पहले चाय बनाने लगी। चाय बनाने के पश्चात् 40 मिनट का अवशेष assignment पूरा किया, फिर वह मूवी देखने बैठ गयी।

(v) अशोक को तीन assignments करने थे—CIT, OS तथा FED. CIT का assignment 45 minute का था, FED का 1 घंटा 20 मिनट का तथा OS का 1 घण्टा 05 मिनट का। उसने पहले आधा घंटा CIT का Assignment किया, फिर उसे छोड़कर आधा घंटा FED का Assignment किया, फिर उसे छोड़कर आधा घंटा OS का Assignment किया। अब उसने CIT का 15 मिनट का अवशेष कार्य किया, फिर आधा घंटा FED का, फिर आधा घंटा OS का, फिर FED का 20 मिनट का अवशेष कार्य तथा फिर OS का 05 मिनट का अवशेष कार्य।

(vi) स्मिता को पढ़ाई करनी थी, (2 घण्टे के लिये), घर का कुछ कार्य करना था (40 मिनट के लिये) तथा अपनी Friend से फोन पर बात करनी थी (5 मिनट के लिये)। उसने सोचा कि पहले सबसे कम समय वाला कार्य कर लेती हूँ। अत: उसने 5 मिनट फोन पर अपनी friend से बात की। अब उसने सोचा कि घर का कार्य कर लेती हूँ (40 मिनट के लिये)। उसको कार्य करते हुये 20 मिनट ही हुये थे कि उसकी मम्मी ने कहा कि चाय बना दो। हालांकि यह कार्य 10 मिनट का था किन्तु उसने घर का कार्य बीच में न छोड़ते हुये पहले वह पूरा किया, फिर चाय बनाई तथा फिर 2 घण्टे पढ़ाई की।

§ 4.7. डिस्क शैडयूलिंग (Disk Scheduling) :

अब तक इस अध्याय में CPU scheduling के संबंध में चर्चा की गई। आइये, अब यह देखते हैं कि डिस्क की scheduling किस प्रकार की जाती है।

हम जानते हैं कि फाइल system सामान्यत: secondary memory में निवास करता है जिसको भारी मात्रा में storage डाटा में permanently स्टोर करने हेतु design किया जाता है (जैसे कि disk)। Physical disk को कई partitions में segment किया जाता है जिससे media use को कन्ट्रोल किया जा सके तथा एक ही स्पीड पर मल्टीपल तथा varying file system सम्भव हो सके। File system को logical file system संरचना पर mount किया जाता है ताकि वह यूज़ हेतु उपलब्ध हो सके।

File system अक्सर layered (परतों वाली) या modular (कई भागों वाली) संरचना में implement की जाती है। इसमें से निचले स्तर या परतें स्टोरेज युक्तियों की physical properties से सम्बन्धित होती है जबकि ऊपर वाली परतें सांकेतिक फाइल नेम तथा फाइल के लॉजिकल प्रापर्टीज़ से सम्बन्धित होती है तथा बीच वाले लेवल logical file concept को physical file properties से मैप करते हैं।

डिस्क drives एकल विमीय व्यूह (single dimensional arrays) के logical blocks के रूप में address किये जाते हैं। यह लॉजिकल ब्लाक्स ट्रांसफर का सबसे छोटा यूनिट होते हैं। Logical blocks की यह one dimensional array डिस्क के sector पर sequentially map की जाती है। Sector zero सबसे बाहर वाले cylinder के पहले ट्रैक का पहला सेक्टर होता है। मैपिंग इस सेक्टर से स्टार्ट होकर बाहर वाले सिलिण्डर से अन्दर की ओर चलती है।

Operating system का एक दायित्व यह भी होता है कि हार्डवेयर को दक्षतापूर्वक उपयोग किया जा सके। अत: disk drive को उपयोग करते समय fast access time व अधिक bandwidth वाँछनीय होती है। Access time के दो भाग होते हैं—

Seek time तथा rotational latency time—

Seek time वह समय होता है जो डिस्क arm heads को वाँछित सेक्टर वाले सिलिण्डर की ओर move करने में लगता है। Rotational latency वह अतिरिक्त time होता है जो डिस्क वाँछित सेक्टर को disk head पर move करने में लेती है। Disk bandwidth की गणना हेतु transfer की गयी bytes की कुल संख्या को पहली request तथा अन्तिम transfer के बीच लगने

वाले कुल समय से भाग करके की जा सकती है। अत: Disk को I/O request का एक अच्छे क्रम में सर्व करके access time व bandwidth दोनों में सुधार किया जा सकता है। जब किसी प्रौसेस को डिस्क से I/O की आवश्यकता होती है तो वह operating system को एक system call issue करता है। इस request में कई प्रकार की सूचनाएं होती है, जैसे कि operation input है या output, ट्रांसफर का disk address क्या है, memory address क्या है तथा sector number क्या है?

यदि वांछित डिस्क ड्राइव और कन्ट्रोलर उपलब्ध होते है तो request तुरन्त सर्विस की जा सकती है। किन्तु यदि इन दोनों में से कोई busy होता है तो आने वाली नई request उस drive की pending request की queue के डाल दी जाती है। अत: एक multiprograming system में कई प्रौसेस होते हैं। इसलिए इसकी disk queue में कई pending request हो सकती हैं। अत: जब कोई request पूरी होती है तो ऑपरेटिंग सिस्टम को इस बात का चुनाव करना होता है कि अब कौन सी pending request को सर्विस दी जाए। इसके लिए कई disk scheduling algorithms होती है जिनका वर्णन किया जा रहा है।

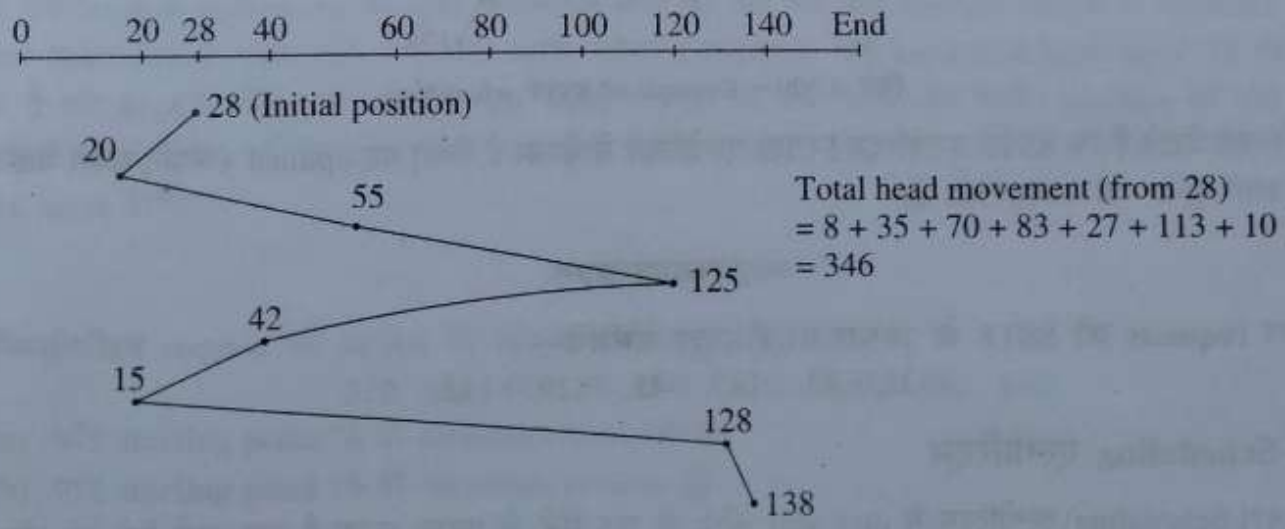
FCFS Scheduling—

FCFS अर्थात् first come first served अर्थात् पहले आओ पहले पाओ। अत: उस algorithm में जो request पहले आती है उसे पहले सर्व किया जाता है हालांकि ये request पक्षपातरहित (fair) होती है किन्तु यह सबसे तेज़ सर्विस प्रदान नहीं कर पाती है उदाहरणत: माना कि disk queue में निम्न सिलिण्डर पर ब्लाक हेतु request है—

20, 55, 125, 42, 15, 128, 38.

FCFS algorithm के अनुसार operating system request को इसी क्रम में सर्व करेगा। माना कि प्रारम्भ में disk dead cylinder संख्या 28 पर स्थित है तो यह यहाँ से पहली request की ओर अर्थात् सिलिण्डर 20 पर move कर जाएगा। उसके बाद सिलिण्डर 55 पर और इसी क्रम में क्रमश: सिलिण्डर 125, 42, 15, 128, 38 पर move करेगा और इस प्रकार total head movement 346 सिलिण्डर्स का होगा। इसको चित्र 4.7(a) में दिखाया गया है।

FCFS Algorithm में होने वाली समस्या को उपरोक्त उदाहरण की सहायता से समझा जा सकता है। आप देखें कि उपरोक्त में डिस्क हैड को सिलिण्डर संख्या 55 से 125 पर जाना पड़ा तथा वापस फिर सिलिण्डर संख्या 42 पर आना पड़ा। यदि सिलिण्डर संख्या 55 व 42 को एक साथ सर्व किया जाता तो total head movement कम हो जाता तथा system की performance बेहतर बनाई जा सकती।



चित्र 4.7(a)—FCFS disk scheduling का उदाहरण

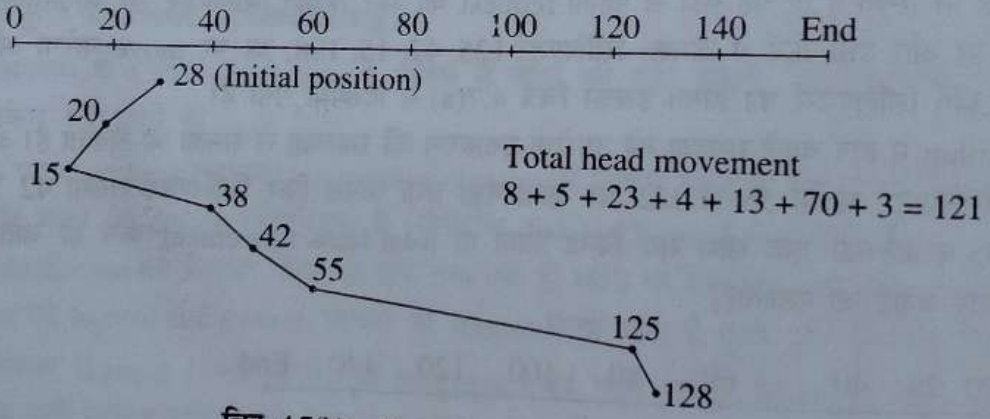
अभ्यास प्रश्न

निम्न requests को FCFS के आधार पर schedule कीजिये—

21, 31, 51, 121, 53, 122, 60, 130

SSTF Sheduling—

SSTF अर्थात् shortest-seek time first algorithm। इसका अर्थ यह है कि head को अपनी वर्तमान position से सबसे पास वाली (निकटतम) position वाली request की ओर move किया जाए। अत: SSTF algorithm उस request को select करती है जिसका seek time इसकी वर्तमान head position से न्यूनतम होता है। चूंकि seek time head द्वारा traversed किये गये सिलिण्डर की संख्या पर निर्भर करता है, अत: SSTF एल्गारिदम उस pending request को select करती है जिसकी position वर्तमान head position से निकटतम होती है। SSTF शैड्यूलिंग shortest job first अर्थात् SJF शैड्यूलिंग का एक रूप है और SJF शैड्यूलिंग की भांति इसमें भी कुछ request का starvation (अर्थात् सर्व न होना) हो सकता है। उदाहरणत: मान लीजिए queue में सिलिण्डर 20 हेतु request है तथा उसके बाद 155 हेतु request है अब मान लीजिए कि जब वह सिलिण्डर 20 की request को सर्व कर रहा है तभी सिलिण्डर 32 हेतु request आ जाती है तो यह request सर्व हो जाएगी। चूंकि सिलिण्डर 32, 20 के अधिक निकट है तथा 155 वाली request को wait करना पड़ेगा। अब यदि इसी बीच सिलिण्डर 26 हेतु request आ जाती है तो यह भी सर्व हो जाएगी और 155 वाली request हो और इंतज़ार करना पड़ता है। कहने का तात्पर्य यह है कि यदि वर्तमान disk position के आस-पास वाली position की request लगातार आती रही तो दूर वाली request एक लम्बे इंतजार के बाद तड़प-तड़प के मर जाएगी। इस स्थिति की सम्भावना तब अधिक होती है जब pending request वाली पंक्ति लम्बी होती चली जाती है।



चित्र 4.7(b)—Example of SSTF scheduling

अत: हम देखते हैं कि SSTF एल्गोरिद्म FCFS एल्गोरिद्म से बेहतर है किन्तु यह optimal (अर्थात् सबसे बढ़िया) नहीं है तथा उसमें सुधार की आवश्यकता है।

अभ्यास प्रश्न

निम्न requests को SSTF के आधार पर शैडयूल कीजिये—

33, 43, 143, 45, 120, 132, 5

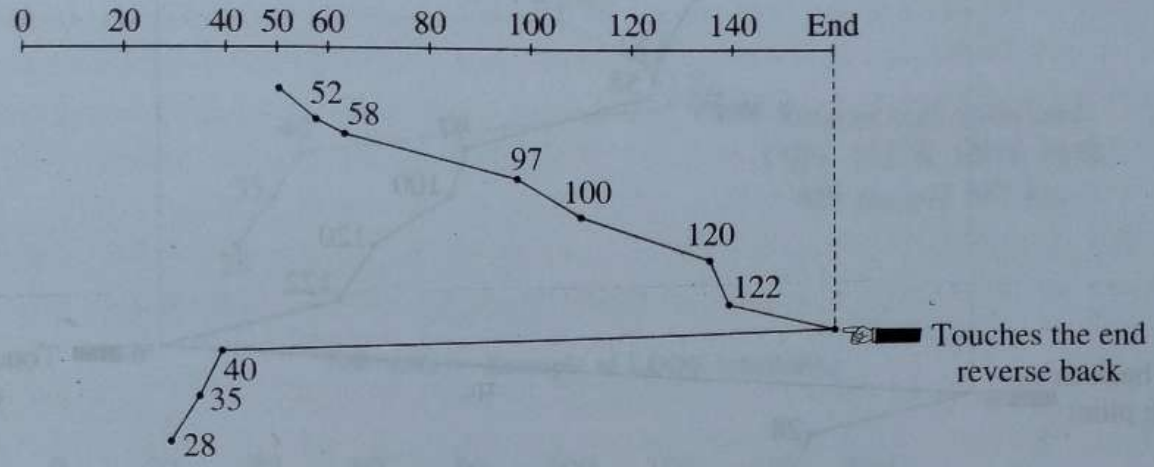
SCAN Scheduling एल्गोरिद्म

SCAN Scheduling एल्गोरिद्म में disk arm, disk के एक सिरे से प्रारम्भ करता है तथा दूसरे सिरे की ओर move करता है तथा requests को service करता है। Disk के दूसरे सिरे पर पहुँचने के पश्चात् पुन: अपना direction reverse करता है तथा पुन: सभी request service करते हुये प्रारम्भिक सिरे तक पहुँचता है तथा इस प्रकार servicing की प्रक्रिया चलती रहती है। Head लगातार disk के start व end के मध्य (back and forth) गति करता है। SCAN algorithm को

elevator algorithm भी कहा जाता है क्योंकि इसमें disk arm किसी building में लगे elevator (लिफ्ट) की तरह व्यवहार करता है तथा ground floor से top floor की ओर सभी request को service करते हुये चलता है तथा top floor पर पहुँचने के पश्चात् वापस ग्राउन्ड फ्लोर की ओर सभी request को service करते हुये चलता है।

उपरोक्त को समझने हेतु एक उदाहरण लेते हैं, माना कि queue निम्नवत् है—

52, 58, 26, 100, 97, 120, 35, 40, 122 तथा head 50 से start करता है।



चित्र 4.7(c)—Example of SCAN scheduling

माना कि head 50 से start करके forward दिशा में (end की ओर) चलता है। आप देखें कि उपरोक्त queue में 50 से अधिक वाले request है: 52, 58, 97, 100, 120 तथा 122 तो 50 से आगे चलते हुये वह पहले 52, फिर 58, फिर 97, फिर 100, फिर 100 तथा फिर 122 वाली request serve करता हुआ end तक जायेगा। अब end से zero तक चलने में यह शेष requests अर्थात् क्रम से 40, 35 तथा 28 को serve करेगा (चित्र 4.7(c) देखें)।

SCAN Scheduling की मुख्य विशेषता यह है कि यदि head के मूवमैंट वाली direction में (head के front में) कोई request queue में arrive करती है तो वह तुरन्त serve हो जाती है (उदाहरणत: यदि head front movement कर रहा है तथा 60 position पर है, तथा उसी समय 61 या 62 position की request आती है तो वह तुरन्त serve हो जायेगी)। लेकिन यदि head के movement की दिशा के विपरीत दिशा की ओर की कोई request आती है तो उसे head को दूसरे सिरे तक जाकर फिर से वापस आने का इंतजार करना पड़ेगा। (उदाहरणत: यदि head front movement कर रहा है, तथा 60 पर है और head के पीछे वाले कोई request उदाहरणत: 59, 58 (या उससे कम वाली) position की कोई request आती है तो head अपना पूरा चक्कर लगाकर तथा दूसरे सिरे तक जाकर जब उल्टी डायरैक्शन में वापस आयेगा तभी यह request serve होगी।

अभ्यास प्रश्न

निम्नलिखित request को SCAN के आधार पर शैडयूल कीजिये–

32, 35, 45, 68, 50, 132, 135, 85, 145

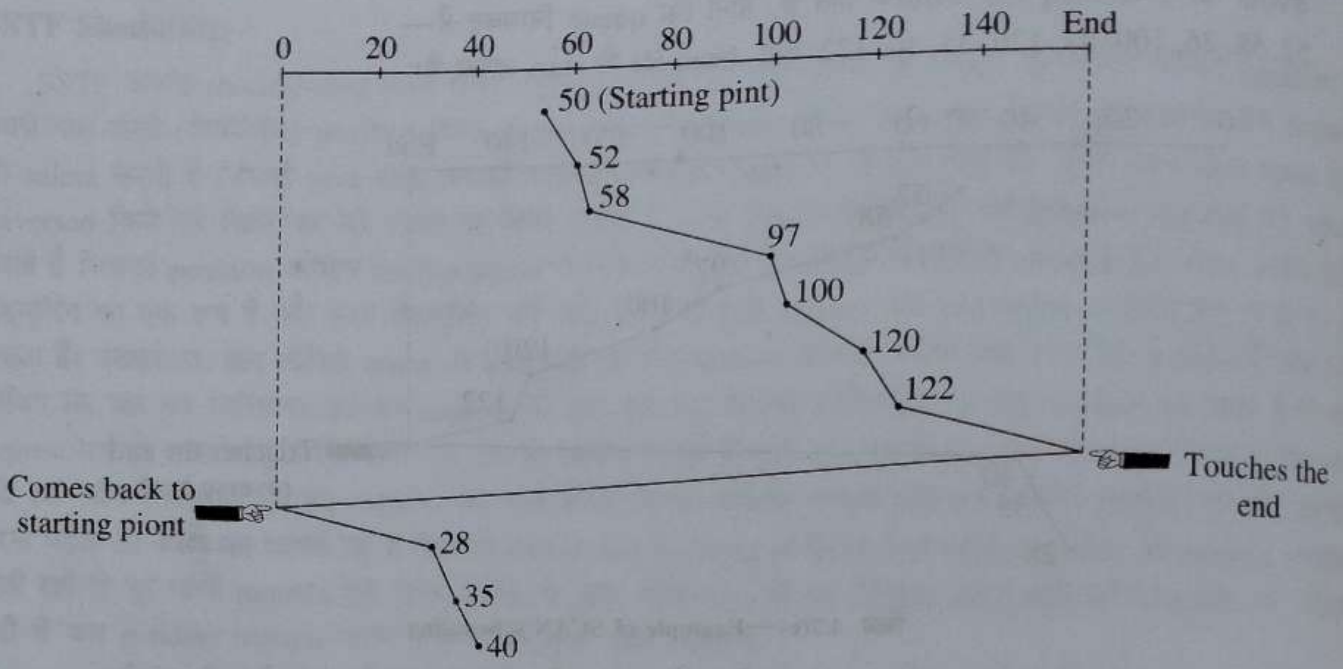
(a) यदि starting point 70 हो direction forward हो

(b) यदि starting point 70 हो direction reverse हो

C-Scan या Circular Scan Scheduling—

SCAN disk scheduling में हमने देखा कि head एक दिशा में चलता है तथा सिरे तक पहुँचकर वापस लौटता है। यदि हम लगभग एक समान वितरण वाली requests की कल्पना करें तो यह स्पष्ट है जब भी head किसी सिरे को छूकर दूसरी ओर चलेगा तो head में front में बहुत कम requests होगी (क्योंकि यह area अभी-अभी serve हुआ है) तथा डिस्क

के दूसरे छोर पर request का घनत्व बहुत अधिक होगा (क्योंकि इस area की request बहुत देर से serve होने के इंतज़ार में खड़ी हैं)। तो क्यों न पहले इन्हीं requests को serve कर दिया जाये? C-SCAN या circular scan algorithm इसी सिद्धान्त पर आधारित है।



चित्र 4.7(d)—Example of C-SCAN scheduling

C-Scan scheduling scan scheduling का एक ही रूप है जिसमें disc head प्रारम्भ से दूसरे सिरे तक की request serve करता है तथा फिर तुरन्त start पर (दूसरे सिरे पर) पहुँच जाता है (अर्थात् return journey में काई request serve किये बिना ही) तथा फिर से starting से request serve करना प्रारम्भ करता है। (चित्र 4.8(d) देखें) अत: इस प्रकार का design समान wait time प्रदान करता है। C-scan scheduling द्वारा cylinders को एक circular list की तरह treat किया जाता है जहाँ डिस्क head अंतिम सिरे से तुरन्त प्रारम्भ पर आ जाता है।

अभ्यास प्रश्न

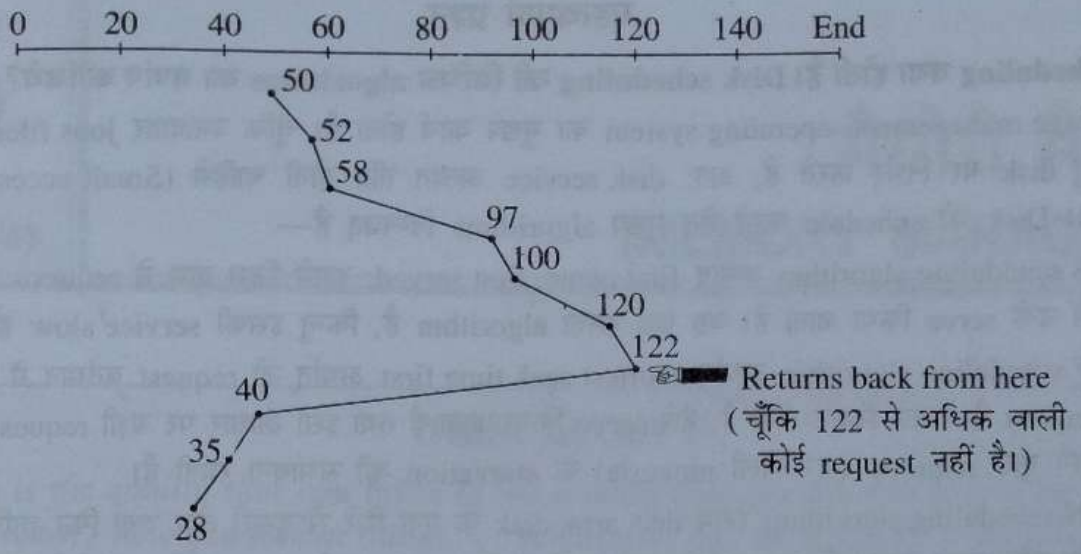
निम्नलिखित request को C-Scan के आधार पर शैडयूल कीजिये—

28, 32, 35, 45, 68, 50, 132, 135, 85, 145

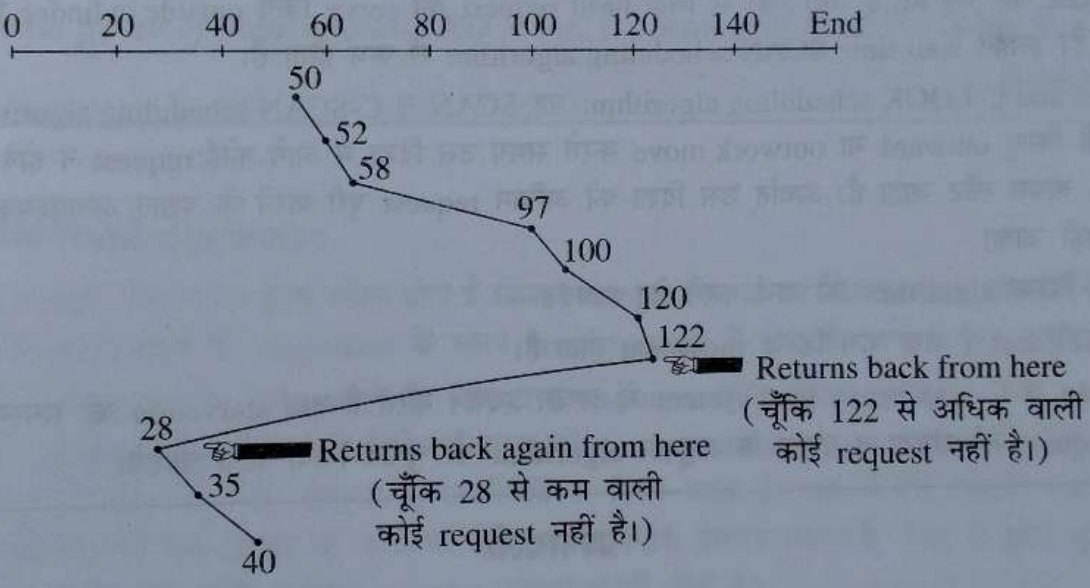
(a) यदि starting point 70 हो direction forward हो

(b) यदि starting point 70 हो direction reverse हो

आपने देखा कि SCAN व C-SCAN Scheduling में disk arm, disk के दूसरे छोर तक जाता है, भले ही request हो या न हो। इसका व्यवहारिक रूप हैं, LOOK व C-LOOK scheduling। इसमें disk arm केवल छोर की अंतिम request तक जाता है तथा उससे आगे कोई request न होने पर वापस लौट जाता है। इनको LOOK तथा (तथा C-LOOK) scheduling इसलिए कहा जाता है क्योंकि यह किसी दिशा में आगे बढ़ने से पहले यह देखती हैं (अर्थात् look करती है) कि आगे कोई request है भी या नहीं। अत: आगे कोई request न होने पर disk arm अनावश्यक रूप से आगे नहीं बढ़ता। (चित्र 4.8(e) देखें)



चित्र 4.8(e)—Example of LOOK scheduling



चित्र 4.8(f)—Example of C-LOOK scheduling

अभ्यास प्रश्न

निम्नलिखित request को C-Look के आधार पर शैडयूल कीजिये–

28, 32, 35, 45, 68, 50, 132, 135, 85, 145

(a) यदि starting point 70 हो direction forward हो

(b) यदि starting point 70 हो direction reverse हो

महत्वपूर्ण प्रश्न

• **Disk scheduling क्या होती है। Disk scheduling की विभिन्न algorithms का वर्णन कीजिये?**

उत्तर—Storage management, operating system का मुख्य कार्य होता है। चूंकि ज्यादातर jobs files के input व आउटपुट हेतु disk पर निर्भर करते हैं, अत: disk service अत्यंत तीव्र होनी चाहिये (Small access time, high bandwidth)। Disk को schedule करने हेतु मुख्य algorithms निम्नवत् है—

(i) FCFS scheduling algorithm अर्थात् first come, first served: इसमें जिस क्रम में requests आती है, उसी क्रम में उन्हें serve किया जाता है। यह एक सरल algorithm है, किन्तु इसकी service slow होती है।

(ii) SSTF scheduling algorithm अर्थात् shortest seek time first अर्थात् जो request वर्तमान में serve की जा रही request के सबसे निकट होती है, उसे serve किया जाता है तथा इसी आधार पर सभी requests serve होती है। इसमें कुछ requests (दूर वाली requests) के starvation की संभावना रहती है।

(iii) SCAN scheduling algorithm: इसमें disk arm, disk के एक सिरे से दूसरी ओर तथा फिर वापिस दूसरी छोर की ओर गति करता कहता है तथा मार्ग में आने वाली सभी requests को serve करता रहता है। इसको elevator algorithm भी कहते हैं।

(iv) Circular SCAN या C-SCAN scheduling algorithm: इसमें disk arm inwards move करके innermost cylinder तक पहुँचता है तथा वहां से बिना किसी request को serve किये outside cylinder पर jump कर जाता है। इसका wait time SCAN scheduling algorithm से कम होता है।

(v) LOOk and C-LOOK scheduling algorithm: यह SCAN व C-SCAN scheduling algorithm के समान होती है किन्तु outward या outwork move करते समय उस दिशा में आगे कोई request न होने की स्थिति में drum वापस लौट जाता है। अर्थात् उस दिशा की अन्तिम request पूरी करने के पश्चात् अनावश्यक रूप से सिरे तक नहीं जाता।

नोट—डिस्क algorithm को सर्च करने हेतु मापदंड—

- SSTF fast है तथा कम डिस्क movment होता है।
- Scan व C-scan heavy load system में अच्छा प्रदर्शन करते हैं तथा starvation की समस्या नहीं होती
- Request की संख्या व प्रकार के अनुसार algorithm का चुनाव किया जाना चाहिये।

प्रश्नावली

1. (a) CPU scheduling से आप क्या समझते हैं?
   (b) CPU scheduling की मुख्य एल्गोरिद्म्स पर चर्चा कीजिये
   (c) डिस्क scheduling की मुख्य algorithms पर चर्चा कीजिये।
2. Round Robin scheduling व priority scheduling में अंतर समझाइये।
3. Long term, medium term व short term scheduler में अंतर समझाइये।
4. निम्न कथन किस scheduling से सम्बन्धित हैं—
   (a) पहले आओ, पहले पाओ _____
   (b) टाइम क्वांटम _____
   (c) सबसे छोटा, सबसे पहले _____
   (d) सबसे कम अवशेष समय वाला सबसे पहले _____
   (e) सबसे अधिक प्राथमिकता वाला सबसे पहले _____
5. Scheduling की क्या आवश्यकता है।

5 Chapter

मैमोरी मैनेजमेंट
(MEMORY MANAGEMENT)

THINK ABOUT IT

Creativity is the quality that you bring to the activity that you are doing. It is an attitude, an inner approach - how you look at things . . . Whatsoever you do, if you do it joyfully, if you do it lovingly, if your act of doing is not purely economical, then it is creative. **— Osho**

Success is not the key to happiness. Happiness is the key to success. If you love what you are doing, you will be successful. **— Albert Schweitzer**

I am thankful for all of those who said NO to me. It's because of them I'm doing it myself. **— Albert Einstein**

§ 5.1. परिचय (Introduction) :

किसी भी कम्प्यूटर सिस्टम का मुख्य उद्देश्य होता है प्रोग्राम्स execute करना। यह प्रोग्राम्स तथा इनसे संबंधित डेटा (जिसको कि यह प्रोग्राम्स access करते हैं) execution के समय main memory में होता है (कम से कम आंशिक रूप से)।

CPU के utilization को बेहतर बनाने के लिये तथा response स्पीड बेहतर बनाने के लिये कम्प्यूटर बहुत सारे processes को memory में रखता है। इसके लिये कई प्रकार की memory management schemes (मैमोरी प्रबंधन स्कीम्स) होती हैं तथा इनमें से कौन सी अधिक उपयुक्त है, यह स्थिति पर निर्भर करता है। अत: मैमोरी मैनेजमेंट स्कीम का चयन कई कारकों पर निर्भर करता है तथा सिस्टम का हार्डवेयर डिज़ाइन भी मैमोरी प्रबंधन स्कीम के चयन में मुख्य भूमिका अदा करता है क्योंकि प्रत्येक स्कीम की अपनी हार्डवेयर support आवश्यकतायें होती है।

पिछले अध्याय में हमने देखा कि किस प्रकार कई processes CPU को एक साथ share करते हैं। CPU scheduling के फलस्वरूप CPU के utilization तथा कम्प्यूटर की स्पीड, दोनों में सुधार लाया जा सकता है। इस performance में सुधार लाने के लिये यह भी आवश्यक है कि कई सारे processes मैमोरी को share कर सकें अर्थात् मैमोरी में स्थान पा सकें।

Memory Management—

- *Main Memory refers to a physical memory that is the internal memory to the computer.*
- *Also known as RAM (Random access memory).*
- *Computer able to change only data that is in main memory.*
- *Every program we execute and every file we access must be copied from a storage device into main memory.*
- *All the programs are loaded in the main memory for execution. Sometimes complete program is loaded into the memory, but some times a certain portion or subroutine of the program is loaded into the main*

memory only when it is called by the program, this mechanism is called Dynamic Loading, resulting in improvement of the performance.

- *When one program is dependent on some other program, then, rather than loading all the dependent programs, CPU links the dependent programs to the main executing program when its required. This mechanism is known as Dynamic Linking.*

Memory Management
- handles or manages primary memory.
- keeps track of each and every memory location whether it is allocated to some process or it is free.
- checks how much memory is to be allocated to processes.
- decides which process will get memory at what time.
- decides allocation strategy.
- tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Operating system computer की memory का प्रबंधन करता है तथा यह सुनिश्चित करता है कि processes के पास कार्य करने हेतु पर्याप्त मैमोरी उपलब्ध है। इसके साथ-साथ सिस्टम में उपलब्ध विभिन्न प्रकार की मैमोरी को properly use किया जाना सुनिश्चित करता है।

जैसे ही कम्प्यूटर बूट-अप होता है, operating system memory में स्वत: (अपने आप) लोड हो जाता है जबकि application programs memory तभी execute हो सकते हैं जब उससे संबंधित instruction व डेटा मैमोरी मे लोड होते हैं। किसी program द्वारा किसी अन्य प्रोग्राम को allocate की गई मैमोरी तक पहुँच बनाने की अनुमति नहीं मिलनी चाहिये अन्यथा वह प्रोग्राम दूसरे प्रोग्राम के डेटा को corrupt कर सकता है। ध्यान रखें कि मैमोरी एक scarce resource (अर्थात् दुर्लभ resource) है तथा बिना आवश्यकता के किसी प्रोग्राम को मैमोरी allocate कर देने से दूसरे processes के execution में बाधा उत्पन्न हो सकती है। अत: मल्टी यूज़र तथा मल्टी टास्किंग वातावरण में efficient memory management एक अनिवार्य अंग है। Memory management से संबंधित मुख्य गतिविधियाँ है—

- यह track करना कि वर्तमान में मैमोरी का कौन सा हिस्सा किस के द्वारा use किया जा रहा है।
- यह decide करना कि memory उपलब्ध होने पर कौन सा process memory में load किया जाये।
- Allocation technique का निर्णय लेना।
- Memory को allocate तथा deallocate करना।

§ 5.2. मैमोरी मैनेजमेंट (Memory Management) :

हम जानते हैं कि मैमोरी में bytes तथा binary words के समूह स्टोरड् होते हैं तथा प्रत्येक word का एक address होता है। CPU इन address की सहायता से instructions को memory से fetch करता है। यह instruction decode होने के पश्चात् आवश्यकतानुसार अन्य memory read cycles भी हो सकती है। जब यह instruction execute हो जाता है तो results को memory में store कर दिया जाता है।

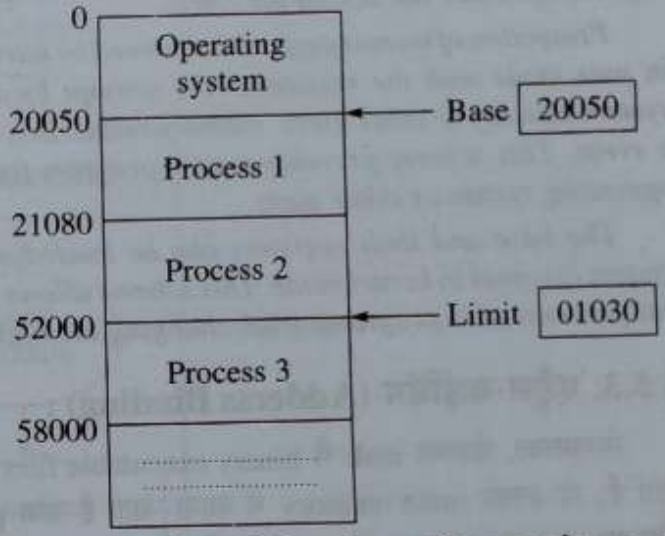
Memory management is the functionality of an operating system which handles or manages primary memory. It keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed and updates the status accordingly.

यहाँ मुख्य बात यह है कि CPU की पहुँच केवल main memory तथा registers (जो कि processor के अंदर निर्मित हैं) तक ही होती है। अत: जो भी instruction execute किया जाना है या जो भी डेटा use किया जाना है, उसे main memory में ही होना चाहिये, तभी CPU उस पर पहुँच बना पायेगा। यदि ऐसा नहीं है, तो यह instruction या डेटा main memory

में move किया जाता है ताकि CPU उस पर operate कर सके। इसके लिये memory read cycles perform की जाती हैं जो data को secondary storage devices से main memory (registers) में लेकर आती हैं। अत:, physical memory से डेटा न केवल तीव्र गति से आना चाहिये बल्कि सुरक्षित रूप में भी आना चाहिये ताकि एक यूज़र का डेटा दूसरे user से सुरक्षित रह सके। यह सुरक्षा हार्डवेयर द्वारा प्रदान की जाती है तथा इसकी कई विधियाँ है।

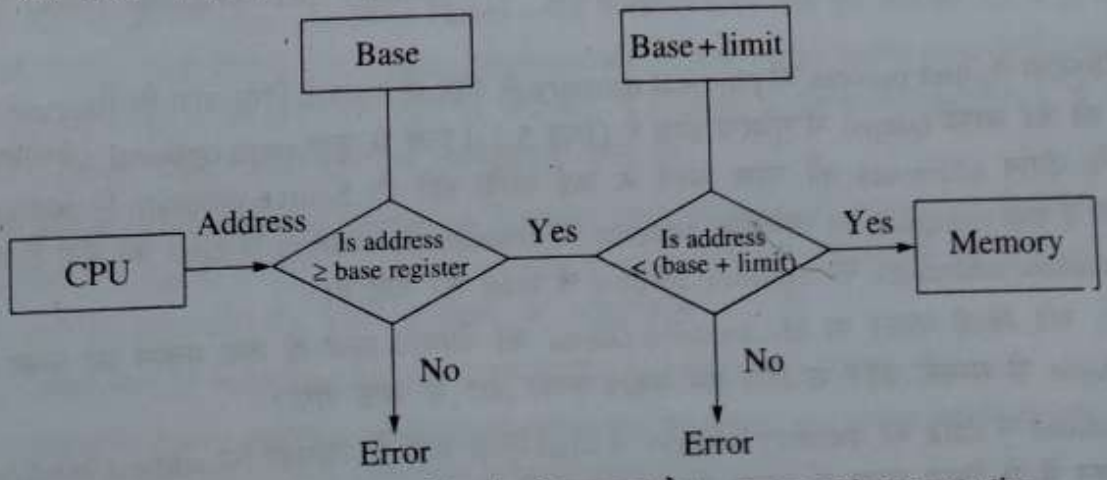
इसके लिये यह सुनिश्चित किया जाता है कि प्रत्येक process का अपना एक separate memory space हो। अत:, process द्वारा use किये जाने वाले legal address की range निर्धारित कर दी जाती है। अत:, process उस range के अंदर के addresses को ही use कर सकेगा, उसके बाहर के नहीं। इसके लिये दो रजिस्टर्स की सहायता ली जाती है जिनको base registers तथा limit रजिस्टर्स कहा जाता है। Base register सबसे छोटा legal (वैध) physical memory address store करता है जबकि limit register range का size store करता है। उदाहरणत: यदि base register में 20050 stored है तथा limit register में 01030 stored है तो इसका तात्पर्य यह हुआ कि program 20050 से 20050 + 01030 = 21080 तक की address range को use कर सकता है। *(चित्र 5.1(a) देखें)*

(यह ठीक ऐसा ही है जैसे कि किसी होटल में manager किसी पार्टी को यदि base address 200 प्रदान कर दे व limit address 25 प्रदान कर दे तो वह पार्टी room number 200 से 225 तक के सभी rooms को use कर सकती है)



चित्र 5.1(a)—बेस व लिमिट रजिस्टर की सहायता से logical address space define करना

अत:, सुरक्षा प्रदान करने हेतु CPU hardware user mode में उत्पन्न address की तुलना इन रजिस्टर्स से करता है, तथा यदि कोई प्रोग्राम अपनी range की बाहर की memory (अर्थात् operating system द्वारा या किसी अन्य user द्वारा use की जा रही memmoy) को access करने का प्रयास करता है तो trap (interrupt या error signal) उत्पन्न हो जाता है (चित्र 5.1 (b) देखें)। अत: इस hardware द्वारा प्रदान की गई सुरक्षा के कारण user program गलती से या जानबूझकर (accidentally or deliberately) operating system या दूसरे users की memory तक पहुँच नहीं बना पाता तथा उनके codes या data structures के साथ किसी भी प्रकार की छेड़खानी नहीं कर पाता।



चित्र 5.1(b)—Base व limit रजिस्टर्स की सहायता से hardware address protection

Base तथा limit registers केवल operating system द्वारा विशेष instuctions use करके store किये जा सकते हैं, अर्थात् user के पास base तथा limit registers में कुछ भी store करने का अधिकार नहीं होता। यह विशेष instructions Kernel mode में execute किये जा सकते है तथा चूंकि Kernel mode में केवल operating system execute करता है,

अत: केवल operating system ही base register व limit register को load कर सकता है। अत: उपरोक्त विधि केवल operating system को ही इन registers की contents को change करने का विशेषाधिकार देती है, अन्य users को नहीं।

Operating system के पास Kernel mode में execute करने पर operting system तथा user मैमोरी तक असीमित रूप से पहुँच बनाने (unrestricted access) का अधिकार होता है। अत:, operating system user memory में user program load कर सकता है, error की स्थिति में इन प्रोग्राम्स को dump कर सकता है। System calls के parameters को access तथा modify कर सकता है, इत्यादि।

To ensure that each process has a separate memory space, protection is provided by using two registers, usually a base and a limit. The base register holds the smallest legal physical memory address; the limit register specifies the size of the range.

Protection of memory space is achieved by having the CPU hardware compare every address generated in user mode with the registers. Any attempt by a program executing in user mode to access operating system memory or other users' memory results in a trap to the operating system, which treats this attempt as a error. This scheme prevents a user program from modifying the code or data structures of either the operating system or other users.

The base and limit registers can be loaded only by the operating system, since only the operating system executes in kernel mode. This scheme allows the operating system to change the value of the registers but prevents user programs from changing the register's contents.

§ 5.3. एड्रैस बाइंडिंग (Adderss Binding) :

सामान्यत:, प्रोग्राम्स disk में binary executable files के रूप में stored होते हैं। जब इन प्रोग्राम्स को execute करना होता है, तो इनको main memory में लाया जाता है तथा process के रूप में स्थापित किया जाता है। इस आधार पर कि कौन सा memory management प्रयोग में लाया जा रहा है, process को execution के दौरान कई बार disk व memory के मध्य move भी करना पड़ता सकता है। डिस्क में stored वह process जो कि execution हेतु memory में जाने का इंतज़ार कर रहे है, input queue (इनपुट पंक्ति) कहलाते हैं।

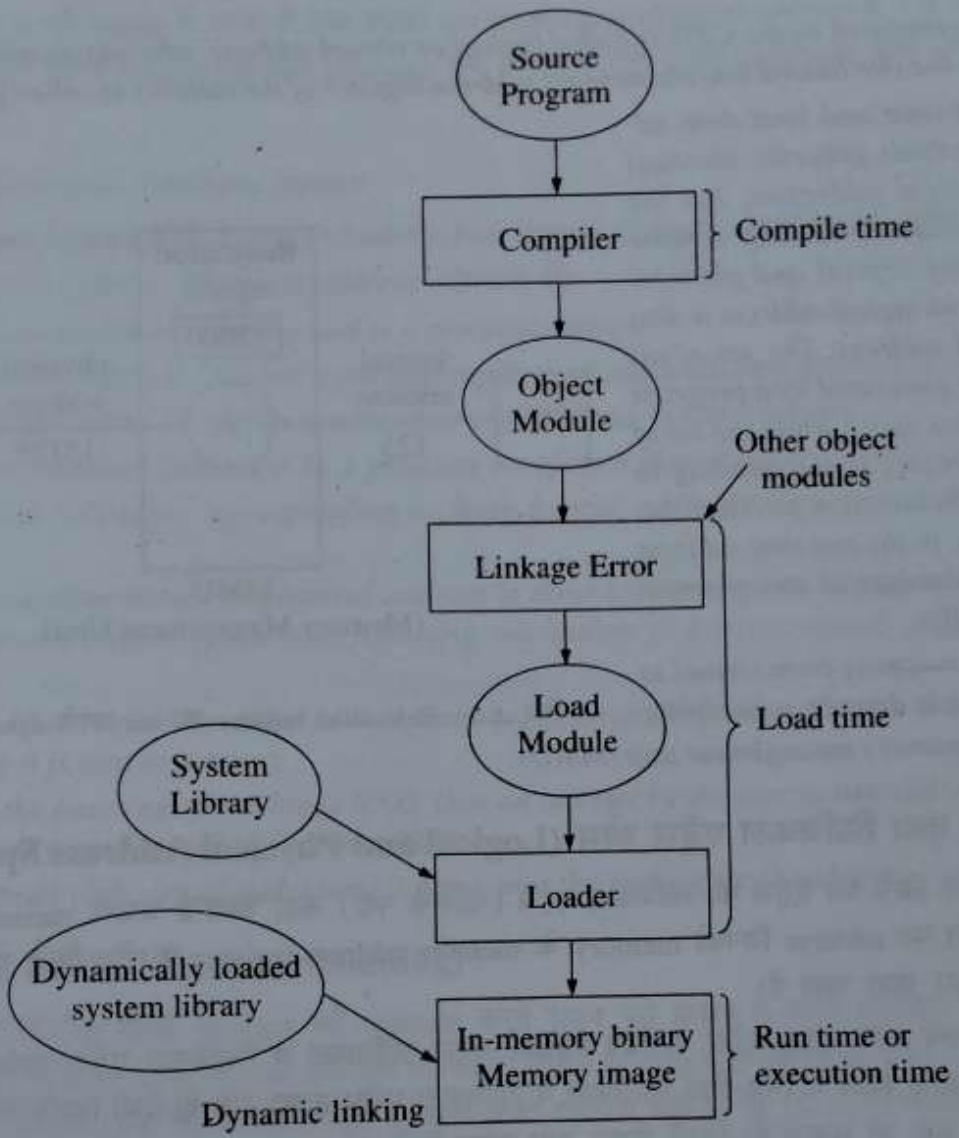
सामान्य प्रक्रिया के अंतर्गत input queue में से किसी process का चयन करके उसको memory में load कर दिया जाता है। जब process execute होता है तो वह memory से instruction या डेटा execute करता है। कार्य पूर्ण होने पर process terminate हो जाता है तथा उसके द्वारा प्रयोग में लायी जा रही memory space available (रिक्त) घोषित कर दी जाती है।

अधिकतर सिस्टम्स में, user process को physical memory में कहीं भी स्थान दे दिया जाता है। Execute होने से पहले user program को कई चरणों (steps) से गुजरना होता है (चित्र 5.2)(इनमें से कुछ steps optional (वैकल्पिक) भी होते हैं) इन steps के दौरान addresses को व्यक्त करने के कई तरीके होते हैं। Source program में address सांकेतिक (symbolic) होता है तथा compiler इन सांकेतिक address को relocatable address से bind कर देता है तथा linkage editor इन relocatable addresses को absolute address से bind कर देता है।

(उदाहरणत:, यदि किसी मकान पर Mr Amitabh Gupta की नेमप्लेट लगी हो तथा मकान का नम्बर 105 हो तो Mr Amitabh Gupta से सम्पर्क करने के लिये हमें मकान नम्बर 105 में जाना होगा)

अत: instructions व data को memory address से bind किया जाता है, जिसको कि address binding कहा जाता है। यह प्रक्रिया निम्न में से किसी चरण में सम्पन्न की जा सकती है—

(a) कम्पाइल करते समय (During Compiling Time)—यदि compile time पर (अर्थात् कम्पाइल करते वक्त) यह मालूम है कि process memory में कहा reside करेगा (अर्थात् उसको कहां स्थान मिलेगा) तो compilation के दौरान ही adsolute code उत्पन्न किया जा सकता है। किन्तु यदि किसी समय यह location change हो जाती है, तो इस code को फिर से compile करना पड़ेगा।



चित्र 5.2—User program की multistep processing

(b) **लोडिंग करते समय (During Loading Time)**—यदि कम्पाइल करते वक्त इस बात का पता न हो कि process को मैमोरी में कहां जगह मिलेगी तो compiler absolute code के बजाय relocatable code जनरेट करता है। अत: final binding loading के समय की जाती है। यदि बाद में starting address change होता है तो केवल user code को reload (पुन: load) करने की आवश्यकता होती है।

(c) **एक्सीक्यूशन के समय (During Execution Time or Run Time)**—यदि execution के समय process को एक memory segment से दूसरे memory segment में move करने की आवश्यकता हो तो binding run time पर की जाती है। इस स्कीम हेतु विशेष हार्डवेयर की आवश्यकता होती है।

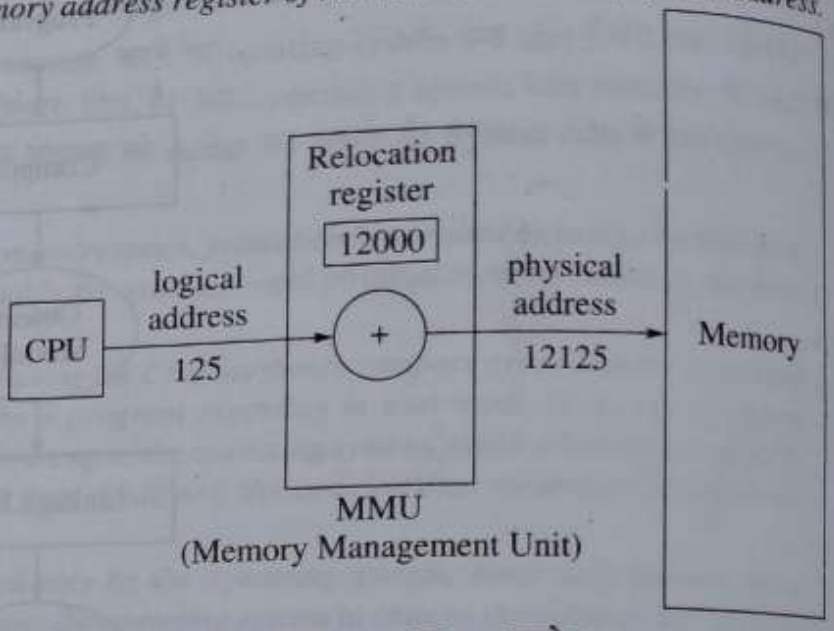
Instructions and data to memory addresses can be done in following ways:

- ***During Compile Time**—When it is known at compile time where the process will reside in memory, compile time binding is used to generate the absolute code.*
- ***During Load Time**—When it is not known at compile time where the process will reside in memory, then the compiler generates relocatable code.*
- ***During Run Time**—If the process can be moved during its execution from one memory segment to another, then binding must be delayed to be done at run time.*

An address generated by the CPU is called logical or virtual address, whereas an address seen by the memory unit i.e., the one loaded into the memory address register of the memory is called physical address.

The compile-time and load-time address-binding methods generate identical logical and physical addresses, but the execution-time address-binding scheme results in differing logical and physical addresses. Here the logical address is also known as virtual address. The set of all logical addresses generated by a program is a logical address space while the set of all physical addresses corresponding to these logical addresses is a physical address space. Thus, in the run time address binding scheme, the logical and physical address spaces differ.

The run time mapping from virtual to physical addresses is done by a hardware device called the memory management unit (MMU).



चित्र 5.3—Relocation register का use करके dynamic relocation

§ 5.4. लॉजिकल तथा फिजिकल एड्रैस स्पेस (Logical and Physical Address Space) :

CPU द्वारा जनरेट किये गये एड्रैस को लॉजिकल एड्रैस (तार्किक पता) कहा जाता है जबकि memory द्वारा देखे जाने वाले address (अर्थात् वह adderss जिनको memory के memory address register में लोड किया गया है), physical address (भौतिक पता) कहा जाता है।

कम्पाइल टाइम तथा लोड टाइम एड्रैस बाइंडिंग विधियाँ समान लॉजिकल व फिज़िकल एड्रैस उत्पन्न करती हैं जबकि executing time binding विधि में भिन्न-भिन्न लॉजिकल व फिजिकल एड्रैस उत्पन्न होते हैं। ऐसी स्थिति में logical address को virtual address कहा जा सकता है। किसी प्रोग्राम द्वारा उत्पन्न किये गये समस्त logical addresses का समूह logical address space कहलाता है जबकि किसी प्रोग्राम द्वारा उत्पन्न किये गये सम्सत physical addresses का समूह physical address space कहलाता है। अत:, execution time binding विधि में logical address space व physical address space भिन्न होते हैं जबकि compile time binding scheme व load time binding scheme में यह identical (एक समान) होते हैं।

अत: run-time (अर्थात् execution time) binding विधि में virtual address की physical address से mapping करने हेतु MMU (memory management unit) नामक हार्डवेयर की आवश्यकता पड़ती है। इस मैपिंग को करने हेतु कई विधियाँ होती है जिनका वर्णन आगे किया जायेगा।

उपरोक्त मैपिंग (अर्थात् MMU scheme) को समझने के लिये पिछले खंड में वर्णित base-register को एक बार फिर से consider करते हैं। अब base register relocation register कहलाता है। जब भी user process memory में भेजा जाता है, तो उसके द्वारा उत्पन्न address को प्रत्येक बार उसको relocation register में stored value से add किया जाता है। (चित्र 5.3 देखें) उदाहरणत: यदि base में 12000 है तो user द्वारा address 0 को address करने पर उसे dynamically location 12000 पर relocate कर दिया जाता है। इसी प्रकार user द्वारा adder 125 को address करने पर उसे dynamically location 12125 पर relocate (mapped) कर दिया जाता है।

अत: इस स्थिति में user program को वास्तविक address (real physical address) दिखाई नहीं देता। Memory management hardware logical address को physical address में convert करता है। अत:, हम देखते हैं कि logical

address 0 से max तक की range में रहता है तथा इसका corresponding physical address R + 0 से R + max की range में रहता है। User केवल logical address generate कर सकता है तथा इनको hardware द्वारा physical address से map किया जाता है।

Logical versus Physical Address Space

- *Address generated by the CPU is a logical address or Virtual address. (अर्थात् CPU द्वारा generate किया गया address logical address या virtual address कहलाता है।)*
- *Address actually available on memory unit is a physical address.*
- *Virtual and physical addresses are the same in compile-time and load-time address-binding schemes.*
- *Virtual and physical addresses differ in execution-time address-binding scheme.*
- *Set of all logical addresses generated by a program is referred to as a logical address space.*
- *Set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.*
- *Run-time mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.*
- *The value in the base register is added to every address generated by a user process which is treated as offset at the time it is sent to memory.*
- *For example, if the base register value is 9000, then an attempt by the user to use address location 200 will be dynamically reallocated to location 9200.*
- *User program deals with virtual addresses; it never sees the real physical addresses.*

§ 5.5. डायनामिक लोडिंग (Dynamic Loading) :

अभी तक हमने देखा कि किसी process को execute करते समय पूरा प्रोग्राम व उससे संबंधित सम्पूर्ण डेटा मैमोरी में उपलब्ध हो। उपरोक्त विधि में प्रौसेस का size उपलब्ध फिजिकल मैमोरी के size द्वारा सीमित हो जाता है अर्थात् यदि प्रोग्राम व उससे संबंधित डेटा अधिक size का है तथा मैमोरी में उतनी जगह उपलब्ध नहीं हो पा रही है, तो समस्या उत्पन्न हो सकती है। मैमोरी space के बेहतर utilization हेतु dynamic loading का प्रयोग किया जाता है।

Dynamic loading में routine तब तक load नहीं होती है, जब तक कि उसे call नहीं किया जाता। सभी routines को डिस्क में relocatable load format में रखा जाता है। Main program को memory में load करके execute किया जाता है। जब कोई routine किसी अन्य routine को call करती है, तो calling routine पहले यह check करती है कि यह दूसरी routine load हुई है या नहीं। यदि नहीं, तो relocatable linking register की सहायता से वांछित subroutine को memory में load किया जाता है तथा इस change को दर्शाने हेतु program की address tables (पता तालिकाओं) को update किया जाता है। अब कन्ट्रोल इस नयी loaded routine को pass कर दिया जाता है।

In case of dynamic loading, a routine is not loaded until it is called. All routines are kept on disk in a relocatable load format. The main program is loaded into memory and is executed. Whenever a routine needs to call another routine, the calling routine first checks to see whether the other routine has been loaded or not. If not, the relocatable linking loader is called to load the desired routine into memory and to update the program's address tables to reflect this change. After that, control is passed to the newly loaded routine.

डायनैमिक लोडिंग का मुख्य लाभ यह है कि unused routine (अर्थात अप्रयुक्त रूटीन) को load करने की आवश्यकता नहीं होती। विशेष तौर पर उन स्थितियों में जहाँ infrequently होने वाली घटनाओं (अर्थात कभी-कभार होने वाली घटनाओं, या ऐसी घटनाओं जिनके होने के chance बहुत कम होते हैं) हेतु बहुत बड़े codes आवश्यक होते हैं (जैसे कि error routine

में dynamic loading काफी उपयोगी सिद्ध होती है) क्योंकि इन स्थितियों में total program size बहुत बड़ा होने के बावजूद use किया जाने वाला अंश (portion that is used and loaded) काफी कम होता है।

यहाँ एक बात उल्लेखनीय है कि dynamic loading हेतु operating system से किसी special support की आवश्यकता नहीं होती। यह user की जिम्मेदारी है कि वह program को इस प्रकार से design करे जिससे dynamic loading का लाभ उठाया जा सके। हालांकि, operating systems library routines की सहायता प्रदान करके user को dynamic loading implement करने हेतु सहायता प्रदान कर सकते हैं।

Dynamic Loading—

- A routine of a program is not loaded until it is called by the program.
- All routines are kept on disk in a re-locatable load format.
- Main program is loaded into memory and is executed.
- Other routines methods or modules are loaded on request.
- Better memory space utilization and unused routines are never loaded.

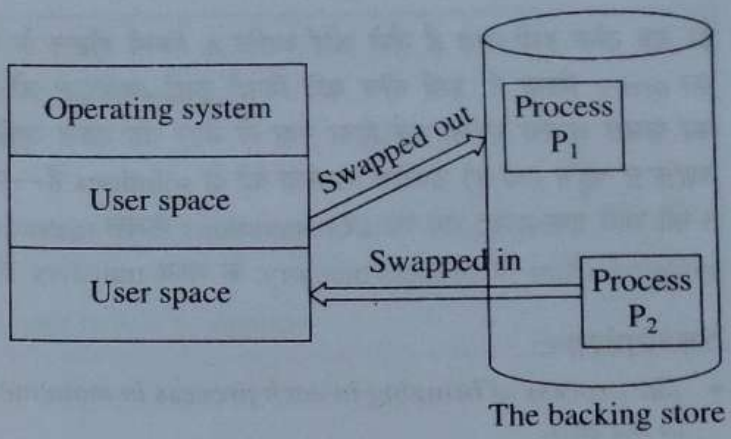
Dynamic Linking—

- Linking is the process of collecting and combining various modules of code and data into a executable file that can be loaded into memory and executed.
- Operating system can link system level libraries to a program.
- When it combines the libraries at load time, the linking is called static linking and when this linking is done at the time of execution, it is called as dynamic linking.
- In static linking, libraries linked at compile time, so program code size becomes bigger
- In dynamic linking libraries linked at execution time so program code size remains smaller.

§ 5.6. स्वैपिंग (Swapping) :

किसी भी process को execution के समय मैमोरी में होना चाहिये। Processes को सामान्यत:, memory तथा backing store के मध्य swap किया जा सकता है। उदाहरणत: किसी process (यदि फिलहाल, उसकी execute होने की बारी नहीं है) को temporary रूप से backing store में swap किया जा सकता है तथा पुन: उसकी बारी आने पर फिर से main memory में लाया जा सकता है। उदाहरण के तौर पर, यदि हम Round Robin scheduling की बात करें, तो यह हम जानते हैं कि इस प्रकार की scheduling में प्रत्येक process को एक निर्धारित time quantum के लिये CPU allocate किया जाता है तथा time quantum expire (समाप्त) होने के पश्चात् यदि process का execution पूर्ण न हुआ हो, तो इस process को queue की tail पर पहुँचा दिया जाता है। अत:, time quantum expire होने के पश्चात् मैमोरी मैनेजर इस process को (जिसका time quantum अभी-अभी समाप्त हुआ है) swap out कर देगा तथा मुक्त हुई memory space (freed memory space) में किसी अन्य process को swap in कर देगा (चित्र 5.4 देखें)। इस दौरान (in the meantime), CPU शैडयूलर मैमोरी में मौजूद किसी अन्य process (next one in the queue) को CPU का time slice allocate कर देगा। इसी क्रम में जब अगला process finish होगा तो इसको भी किसी अन्य process से swap कर दिया जायेगा। अत:, संक्षेप में कहा जाये तो तात्पर्य यह है कि जब भी कोई process finish होगा तथा उसका next turn आने में अभी काफी समय है तो उसके द्वारा occupied (घेरी गयी) memory space को एक ऐसे process से swap (अदला-बदली) कर दी जायेगी, जिसकी बारी पहले आने वाली है। अत:, मैमोरी शैडयूलर processes को तेज़ी से मैमोरी तथा backing store के मध्य swap करता है जिससे वह program, जो execute होने के लिये ready है, मैमोरी में उपलब्ध हो पाते हैं, तथा जब भी CPU शैडयूल CPU को reshedule करना चाहता है, तो वह उपलब्ध हो जाते हैं। इस स्थिति में quantum time का मान पर्याप्त होना चाहिये ताकि swaps के मध्य उपयुक्त मात्रा में computing सम्भव हो सके।

Swapping का एक मिलता जुलता रूप होता है। Roll out–Roll in। यह policy priority based scheduling algorithms हेतु use की जा सकती है। यदि कोई उच्च प्राथमिकता वाला process (higher priority process) आता है तो memory मैनेजर memory में स्थित किसी कम प्राथमिकता वाले process (lower priority process) से इस higher priority वाले process की अदला बदली कर सकता है, जिससे यह higher priority वाला process load होकर execute हो सके। जब यह higher priority process complete हो जाता है तो इसे lower priority process से वापस swap back किया जा सकता है। Swapping की यह विधि Roll-out–Roll-in कहलाती है।



चित्र 5.4—दो processes की swapping

सामान्यत:, swapped out process (जिसको कि memory से बाहर swap किया गया है) मैमोरी मे वापस उसी memory space में swap in किया जा सकता है जहां से उसे swap out किया गया था। यह सीमा address binding के लिये प्रयुक्त विधि पर निर्भर करती है अर्थात् यदि binding assembly या load time पर की जाती है तो process को वापस swap in करने पर कोई दूसरी memory space में move करना असुविधाजनक एवं मुश्किल हो सकता है। यदि execution time binding का प्रयोग किया जा रहा है, तो process को किसी दूसरी memory space पर swap-in किया जा सकता है क्योंकि इस विधि में physical address execution time के समय compute किया जाता है।

चूंकि swapping का तात्पर्य process को memory व backing store के मध्य अदला बदली से है, अत: swapping में backing store की आवश्यकता होती है जो कि सामान्यत: fast disk होती है। Backing store में पर्याप्त स्थान होना चाहिये ताकि सभी users हेतु सभी memory images की copies इसमें store की जा सकें तथा इन memory images का direct access भी प्राप्त किया जा सके।

सिस्टम एक ready queue भी maintain करता है जिनमें वह सभी process शामिल होते है जिनकी memory images backing store में होती हैं या फिर memory में, तथा run होने के लिए ready होते हैं (ready to run)। जब भी CPU scheduler किसी process को execute करने का निर्णय लेता है, तो वह dispatcher को call करता है। डिस्पैचर यह चैक करता है कि queue का next process memory में उपलब्ध है या नहीं। यदि नहीं है, तथा memory में कोई free क्षेत्र भी नहीं है तो dispatcher मैमोरी के किसी process की swapping वांछित process (जिसको run किया जाना है) से कर देता है। फिर वह registers को reload करता है तथा selected process को control transfer करता है।

Swapping में लगने वाला context switch time काफी अधिक होता है। बेहतर CPU utilization हेतु प्रत्येक process का execution time swapping हेतु लगने वाले समय (swap time) की तुलना में काफी अधिक होना चाहिये।

कई अन्य कारक भी swapping को प्रभावित करते हैं, उदाहरणत: यदि हम memory में स्थित किसी process को swap करना चाहते हैं तो वह पूरी तरह फुर्सत में होना चाहिये (completely idle)। यहाँ यह ध्यान रखा जाना चाहिये कि कोई pending I/O न हो।

यह संभव है कि जिस process को swap किया जा रहा है वह किसी I/O operation के लिये wait कर रहा हो। यदि I/O user memory को I/O buffers हेतु asynchronously access कर रहा है, तो process को swap नहीं किया जा सकता। माना कि I/O operation queue में है क्योंकि device busy है। यदि ऐसी स्थिति में किसी process P_A को swap out तथा किसी दूसरे process P_B को swap in कर दिया जाता है तो I/O उस मैमोरी को इस्तेमाल करने लगेगा जो कि अब P_B की

है। यह ठीक इसी तरह है जैसे कोई व्यक्ति A किसी होटल के कमरा संख्य 325 में ठहरा है तथा उसने वेटर को खाना लाने का order किया है, इसी बीच यदि किसी दूसरे व्यक्ति B जो कि किसी दूसरे कमरे में है (माना कमरा नम्बर 411) से A का कमरा अदला बदला कर दिया गया तो वेटर वह खाना व्यक्ति B को सर्व कर जायेगा (क्योंकि अब कमरा नम्बर 325 में व्यक्ति B पहुँच गया है) उपरोक्त समस्या की दो solutions हैं—पहला तो यह कि pending I/O वाले process को swapping न की जाये तथा दूसरा यह कि I/O operations केवल operating system buffers में execute किये जायें। तब operating system buffers व process memory के मध्य transfers तभी होंगे जब process swapped in हो जायेगा।

Swapping—

- *The process of bringing in each process in main memory, running it for a while and then putting it back to the disk.*
- *A mechanism in which a process can be swapped temporarily out of main memory to a backing store, and then brought back into memory for continued execution.*
- *Process needs to be in memory for execution.*
- *Sometimes there is not enough main memory to hold all the currently active processes in a timesharing system.*
- *Excess process are kept on disk and brought in to run dynamically.*
- *Backing store is a usually a hard disk drive or any other secondary storage which fast in access and large enough to accommodate copies of all memory images for all users.*
- *It must be capable of providing direct access to these memory images.*

स्वैपिंग—

Swapping वह तकनीक है जिसमें किसी process को temporarily (थोड़े समय के लिये) memory से backing store को swap किया जाता है तथा बाद में execution हेतु पुन: memory में लाया जाता है।

बैकिंग स्टोर सामान्यत: एक हार्ड डिस्क ड्राइव या कोई अन्य सैकेन्ड्री स्टोरेज़ होती है। इसका access fast होना चाहिये तथा इसमें पर्याप्त space (स्थान) होना चाहिये ताकि सभी users हेतु सभी memory images की copies को स्थान उपलब्ध कराया जा सके इसमें इन सभी memory images हेतु direct access उपलब्ध कराने की क्षमता होनी चाहिये।

§ 5.7. Contiguous Memory Allocation :

मुख्य मैमोरी (main memory) में operating system तथा विभिन्न user processes को accomodate (स्थान देना) किया जाता है। अत: यह आवश्यक है कि main memory को efficient तरीके से allocate किया जाये।

सामान्यत:, memory को दो partitions (हिस्सों) में विभाजित किया जाता है—एक तो operating system हेतु तथा दूसरा user process हेतु। अत: operating system को low memory या high memory में place किया जा सकता है। यह ठीक ऐसा ही है जैसे कि किसी व्यक्ति को बहुमंजिला इमारत में ऊपर की floors आवंटित की जायें या नीचे की। Interrupt vector की location इस निर्णय पर महत्वपूर्ण प्रभाव डालती है कि operating system को higher memory location पर रखा जाता है या lower memory locations पर। चूंकि interrupt vector अक्सर low memory में होता है, अत: programmers अधिकतर operating system को भी low memory में ही रखना पसंद करते हैं।

Contiguous memory allocation में प्रत्येक process को एक single contiguous section of memory (अर्थात memory का एक contiguous section) allocate कर दिया जाता है।

Contiguous Memory Allocation

- Each process is contained in a single contiguous block of memory.
- Memory divided into several fixed size partitions.
- Each partition contains exactly one process.
- When a partition becomes free, a process is selected from the input queue and loaded into it.
- Free blocks of memory are known as holes.
- The set of holes is searched to determine which hole is best to allocate.

§ 5.8. Memory Mapping तथा Protection :

मैमोरी को सुरक्षा प्रदान करने हेतु (जिससे कि कोई अनाधिकृत यूज़र उस तक पहुँच न बना सके तथा एक user दूसरे user की memory तक पहुँच कर उसमें कोई अनाधिकृत फेरबदल न कर सके), एक relocation register तथा limit register की सहायता ली जाती है। Relocation register में smallest physical address store होता है जबकि limit register में logical address की range stored होती है। अत: प्रत्येक logical address का मान limit register से कम होना चाहिये अन्यथा वह address illegal माना जायेगा तथा error उत्पन्न हो जायेगा (क्योंकि limit register से अधिक value का logical address किसी अन्य यूज़र की address space में होगा, अत: इस user को उस memory में पहुँच बनाने की अनुमति नहीं है) यदि logical address की value limit register से कम है तो address legal है तथा इस स्थिति में MMU इस logical address को phyical address से dynamically map कर देगा। (logic address में relocatable register के contents को add करके)

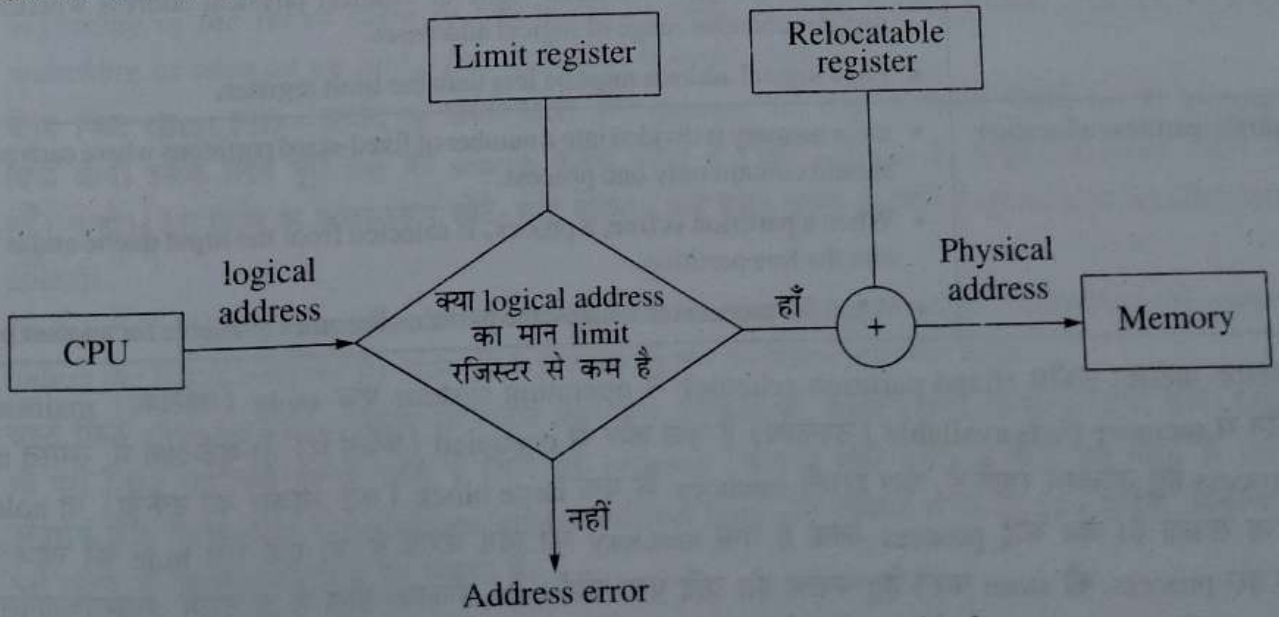
Example—माना Relocatable register = 200030
Limit register = 64200

अत:, user का address space = 200030 से 264230 तक

अब यदि logical address = 84356 तो यह limit register (64200) से अधिक है, अत: addressing error उत्पन्न होगा।

यदि logical address = 23333 तो यह limit register (64200) से कम है, अत: valid है, अत: physical address = [relocatable register] + [logical address] = 200030 + 23333 = 233363 (physical address).

उपरोक्त विवरण को एक फ्लोचार्ट द्वारा निम्नवत् दर्शाया जा सकता है (चित्र 5.5 देखें)।



चित्र 5.5—Memory की mapping तथा सुरक्षा हेतु फ्लोचार्ट

अत: relocation register स्कीम से operating system का size भी dynamically change किया जा सकता है। उदाहरणत: यदि कोई ऐसा code या buffer space है जो कि फिलहाल प्रयोग में नहीं है तो उस space को किसी दूसरे उपयोग में लाया जा सकता है। इस प्रकार का code transient OS code कहलाता है।

Memory Protection—

- A phenomenon by which we control memory access rights on a computer.
- Aimed to prevent a process from accessing memory that has not been allocated to it.
- Prevents a bug within a process from affecting other processes, or the operating system itself, and instead results in a segmentation fault or storage violation exception being sent to the disturbing process, generally killing of process.

प्रश्न यह उठता है कि इनपुट क्यू में प्रतीक्षा करने वाले processes को मैमोरी कैसे एलोकेट की जाती है।

The question arises that how to allocate available memory to the processes that are in the input queue and waiting to be brought into the memory?

Fixed Size Multiple Partition Method

Memory allocate करने की contiguous memory allocation विधि में प्रत्येक process को memory का एक contiguous section allocate किया जाता है। Memory allocate करने की एक सरल विधि यह है कि memory को कई fixed size partitions (समान आकार के कई भाग) में विभाजित कर दिया जाये तथा प्रत्येक partition एक process को आवंटित कर दिया जाये। अत: ऐसी स्थिति में मल्टीप्रोग्रामिंग की डिग्री partitions की संख्या द्वारा सीमित हो जाती है अर्थात् जितने partitions हैं, उतने ही process एक साथ run कर सकते हैं इस प्रकार के multipartition-method में जब भी कोई partition free होता है, तो input queue से process select तथा उस free partition पर load कर दिया जाता है। जब यह process terminate होता है, तो वह अगले process के लिये उपलब्ध हो जाता है। यह विधि प्रारम्भ में IBM OS/360 ऑपरेटिंग सिस्टम द्वारा use किया गया, किन्तु अब प्रचलन में नहीं है।

Memory Allocation	Description
Single-partition allocation	• Relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. • Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. • Each logical address must be less than the limit register.
Multiple-partition allocation	• main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. • When a partition is free, a process is selected from the input queue and is loaded into the free partition. • When the process terminates, the partition becomes available for another process.

फिक्स्ड पार्टीशन स्कीम (fixed partition scheme) में operating system एक table (तालिका) maintain करता है कि कौन से memory parts available (उपलब्ध) हैं तथा कौन से occupied (प्रयोग में) हैं। शुरूआत में, समस्त memory users process हेतु उपलब्ध रहती है, तथा इसको memory के एक large block (बड़े आकार का टुकड़ा) या hole के रूप में देखा जा सकता है। जब कोई process आता है तथा memory की माँग करता है, तो एक ऐसे hole की खोज की जाती है जो कि इस process को store करने हेतु पर्याप्त हो। यदि ऐसा कोई hole उपलब्ध होता है तो उसमें आवश्यकतानुसार मैमोरी allocate (आंवटित) कर दी जाती है तथा शेष मैमोरी भविष्य की आवश्यकताओं हेतु उपलब्ध रखी जाती है।

जब process system में प्रवेश करते हैं, तो उनको input queue (निवेश पंक्ति) में रखा जाता है। OS किसी scheduling algorithm के अनुसार input queue में खड़े process का क्रम निर्धारित कर सकता है। Processes को memory तब तक allocate की जाती है जब तक कि पंक्ति में अगले क्रम में लगे process की मैमोरी आवश्यकताओं को पूरा करने की गुंजाइश न हो अर्थात् memory का कोई ऐसा block (अर्थात् hole) उपलब्ध न हो जो कि उस process की मैमोरी आवश्यकताओं को संतुष्ट करने हेतु पर्याप्त हो। ऐसी स्थिति उत्पन्न होने पर ऑपरेटिंग सिस्टम या तो किसी बड़े hole के free होने का इंतज़ार करता है, या फिर input queue में खड़े अगले सदस्यों को देखता है कि किसी छोटी मैमोरी आवश्यकता वाले प्रौसेस की जरूरत पूरी की जा सके।

सामान्य तौर पर, किसी समय में मैमोरी की पूरी लम्बाई में बिखरें विभिन्न आकार के मैमोरी होल्स का समूह होता है। जब कोई प्रौसेस आता है तथा मैमोरी की demand (माँग) करता है तो सिस्टम ऐसे hole की तलाश करने में जूट जाता है जो कि इस demand को पूरा करने हेतु पर्याप्त हो। यदि होल का आकार बहुत ज्यादा बड़ा है तो इसको दो भागों में split (विभाजित) कर दिया जाता है, तथा इसमें से एक भाग को process को allocate कर दिया जाता है तथा दूसरा भाग होल्स के समूह को वापस कर दिया जाता है। जब कोई process टर्मिनेट (समाप्त) होता है तो वह अपने द्वारा प्रयुक्त memory block को release (अवमुक्त) कर देता है तथा यह block भी holes के set में चला जाता है। यदि यह नया होल किसी पुराने होल का पड़ोसी होता है तो इन दोनों होल्स को merge करके (मिलाकर) एक बड़ा होल बना दिया जाता है। इस समय सिस्टम यह भी check करता है कि क्या कोई ऐसा प्रौसेस तो नहीं है, जो memory के free होने का इंतज़ार कर रहा है तथा क्या यह नया बड़ा block इन इंतज़ार कर रहे process में से किसी की मैमोरी आवश्यकताओं को संतुष्ट करने हेतु पर्याप्त है?

उपरोक्त विधि डायनैमिक स्टोरेज एलोकेशन (dynamic storage-allocation) के लिये प्रयुक्त की जाने वाली कई विधियों में से एक है, जो कि यह निर्धारित करने के लिये प्रयुक्त की जाती है कि *n* size की मैमोरी request को उपलब्ध free holes की list से कैसे संतुष्ट किया जाये। अत: उपलब्ध होल्स में से फ्री होल का चयन करने हेतु कई विधियाँ प्रचलित है जैसे कि first fit, best fit, worst fit strategies (तरीके) आदि।

(i) **फर्स्टफिट (First Fit)**—अर्थात् जो सबसे पहला ऐसा होल उपलब्ध हो जाये जो कि memory की माँग को पूरा करने हेतु पर्याप्त हो, उसे allocate कर दिया जाये। सर्चिंग की शुरूआत तालिका के प्रांरभ से की जा सकती है या फिर वहां से जहां से पिछला first-fit search समाप्त हुआ था। जैसे ही कोई ऐसा होल उपलब्ध होता है जो वांछित आवश्यकता पूर्ण करने हेतु पर्याप्त हो, searching process stop कर दिया जाता है।

Under this schemes, we allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or where the previous first-fit search ended and can be stopped searching as soon as we find a free hole that is large enough.

(ii) **बैस्ट फिट (Best Fit)**—अर्थात् जो सबसे छोटा होल demand को संतुष्ट करने हेतु पर्याप्त हो, वह allocate कर दिया जाये। इसके लिये पूरी list को search करना पडता है (यदि list size के अनुसार ordered (क्रम में लगी हुई) न हो)। इस विधि के फलस्वरूप छोटे-छोटे होल्स (बचे हुये) उत्पन्न हो जाते हैं (produces smallest leftover holes)।

Under this scheme, we allocate the smallest hole that is big enough, searching the entire list, unless the list is ordered by size. This produces the smallest leftover hole.

(iii) **वर्स्ट फिट (Worst Fit)**—अर्थात् जो सबसे बड़ा होल उपलब्ध हो, पहले उसको आवंटित किया जाये। इसके लिये भी पूरी लिस्ट को सर्च करना पडता है (यदि वह ordered (क्रम में लगी हुयी) न हो)। इस विधि से सबसे बड़ा अवशेष होल उत्पन्न होता है (produces largest leftover hole) जो भविष्य में किसी अन्य मैमोरी आवश्यकता को पूरी करने में लाभदायक सिद्ध हो सकता है।

Under this scheme we allocate the largest hole searching the entire list, if not (sorted by size) this produces the largest leftover hole; which may be useful for future memory demands.

Simulations के अनुसार first fit व best fit विधियाँ time व storage utilization worst fit की तुलना में बेहतर सिद्ध हुई है।

§ 5.9. फ्रैगमैंटेशन (Fragmentation) :

मैमोरी एलोकेशन की फर्स्ट फिट व बैस्ट फिट memory allocation विधियों से external fragmentation की समस्या उत्पन्न होती है। जैसे-जैसे processes मैमोरी में लोड तथा remove होते हैं, मैमोरी में उपलब्ध free space छोटे-छोटे टुकड़ों में बंटने लगती है। External fragmentation तब होती है जबकि मैमोरी में किसी request को पुरा संतुष्ट करने हेतु space की उपलब्धता तो पर्याप्त होती है किन्तु यह उपलब्ध space contiguous नहीं होती अर्थात् बड़ी संख्या में छोटे-छोटे होल्स के रूप में fragmentation होती है। कई स्थितियों में प्रत्येक दो processes के मध्य memory के free blocks तो होते हैं किन्तु यह wasted होते हैं तथा इतने बड़े नहीं होते कि किसी process को allocate किये जा सकें। यदि यह सारे छोटे-छोटे blocks एक बड़े free block के रूप में हो तो कई processes द्वारा इनका लाभ उठाया जा सकता तथा कई अन्य processes को run किया जा सकता किन्तु external fragmentation के कारण यह सम्भव नहीं हो पाता।

"First fit व best fit memory allocation strategies के कारण memory में छोटे-छोटे अप्रयुक्त टुकड़ो का उत्पन्न हो जाना external fragmentation कहलाता है।"

"The fragmentation of memory into small pieces (which may be of no use) as a result of first fit and best fit memory allocation strategies is called external fragmentation".

अत: हम देखते हैं कि चाहे first fit strategy use की जाये या best fit, external fragmentaiton की समस्या को पूरी तरह से समाप्त नहीं किया जा सकता। आंकड़े बताते हैं कि प्रत्येक N allocated blocks में लगभग $0.5\,N$ blocks के बराबर मैमोरी fragmentation के रूप में waste (नष्ट) हो जाती है। अत:, एक तिहाई मैमोरी (one third on the memory) अप्रयुक्त (unused) रह जाती है। इसको 50 प्रतिशत नियम (50 percent rule) कहा जाता है।

"External fragmentation के फलस्वरूप 50 percent memory का unusable रह जाना 50 percent rule कहलाता है।

"As a result of external fragmentation 50 percent of the memory become unusable. This is known as 50 percent rule".

Fragmentation—

- *Occurs in a dynamic memory allocation system when most of the free blocks are too small to satisfy any request.*
- *Inability to use the available memory.*
- *Processes are loaded and removed from the memory.*
- *Hence, free holes exists to satisfy a request but is non contiguous i.e. the memory is fragmented into large no. of small holes (known as External Fragmentation).*
- *Sometimes, physical memory is broken into fixed size blocks and memory is allocated in unit of block sizes. The memory allocated to a space may be slightly larger than the requested memory. The difference between allocated and required memory is known as Internal fragmentation i.e. the memory that is internal to a partition but of no use.*
- *As processes are loaded and removed from memory, the free memory space is broken into small little pieces.*
- *If after sometime, processes can not be allocated to memory blocks considering their small size and memory blocks remains unused, known as Fragmentation.*

Memory fragmentation external या internal हो सकती है माना कि किसी multiple partition allocation scheme में एक 16332 bytes वाला एक hole है। माना की अगला प्रौसेस 16328 बाइट्स की मांग करता है यदि हम इसको 16328 बाइट्स एलोकेट कर देते हैं तो 4 बाइट्स का एक hole उत्पन्न हो जाएगा। इस hole को ट्रैक करने हेतु over heads इस hole से भी अधिक होगें। अत: इस समस्या को दूर करने हेतु मैमोरी को fixed size के block में बांट दिया जाता है और block size के आधार पर मैमोरी को units के रूप में एलोकेट किया जाता है। हालांकि इस विधि द्वारा प्रौसेस को allocated memory, requested memory (जिसकी मांग की गई है) से थोड़ी बड़ी हो सकती है। अत: इन दोनों के मध्य अन्तर को internal fragmentation कहा जाता है। वह मैमोरी जो कि partition के अन्दर होती है किन्तु अप्रयुक्त होती है, Internal fragmentation कहलाती है।

The memory which is internal to a partition but is not being used is called internal fragmentation. External fragmentation की समस्या का एक समाधान होता है compaction. इसका उद्देश्य है memory की contents को इस प्रकार shuffle करना (मिलाना या फेटना) होता है जिससे सारी free memory एक साथ एक block में आ जाए हालांकि यह हमेशा सम्भव नहीं हो पाता। यदि relocation static है तथा assembly या load time पर किया गया है तो compaction नहीं किया जा सकता है। Compaction केवल तभी किया जा सकता है जब relocation dynamic है तथा execution time पर किया गया है। यदि addresses को dynamically relocate किया जाए तो relocation में केवल प्रोग्राम तथा data को move करने की तथा फिर base रजिस्टर को change करने की आवश्यकता होती है। जब compaction सम्भव होता है तो उसके मूल्य की भी गणना करनी चाहिए। सबसे simple compaction algorithm में सभी प्रोसेस को मैमोरी के एक छोर की तरफ move किया जाता है जिससे सभी holes एक दिशा में move करते हैं जिससे उपलब्ध मैमोरी का एक बड़ा hole उपलब्ध हो जाता है। हालांकि ये स्कीम महंगी होती है। External fragmentation समस्या का एक अन्य समाधान यह होता है कि processes के logic address स्पेस को non contiguous कर दिया जाए जिससे प्रोसेस को तब physical memory एलोकेट कि जाए जब वह मौजूद हो। इसके दो तरीके हैं—paging तथा segmentation।

Internal Fragmentation vs External Fragmentation—

1. **External Fragmentation**—*Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so cannot be used.*
2. **Internal Fragmentation**—*Memory block assigned to process is bigger. Some portion of memory is left unused as cannot be used by another process.*

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

§ 5.10. पेजिंग (Paging) :

पेजिंग एक मैमोरी मैनेजमैंट स्कीम है जिससे process की physical address space non contiguous रखी जा सकती है। Paging विधि के प्रयोग से varying size (अलग-अलग आकार के) के memory chunks बैकिंग स्टोर में फिट करने की समस्या नहीं होती।

- *A memory management mechanism that allows the physical address space of a process to be non-contiguous.*
- *Solution to fragmentation problem.*
- *Physical memory is divided into blocks of equal size called Pages.*
- *Pages belonging to a certain process are loaded into available memory frames.*
- *External fragmentation avoided by using paging technique.*

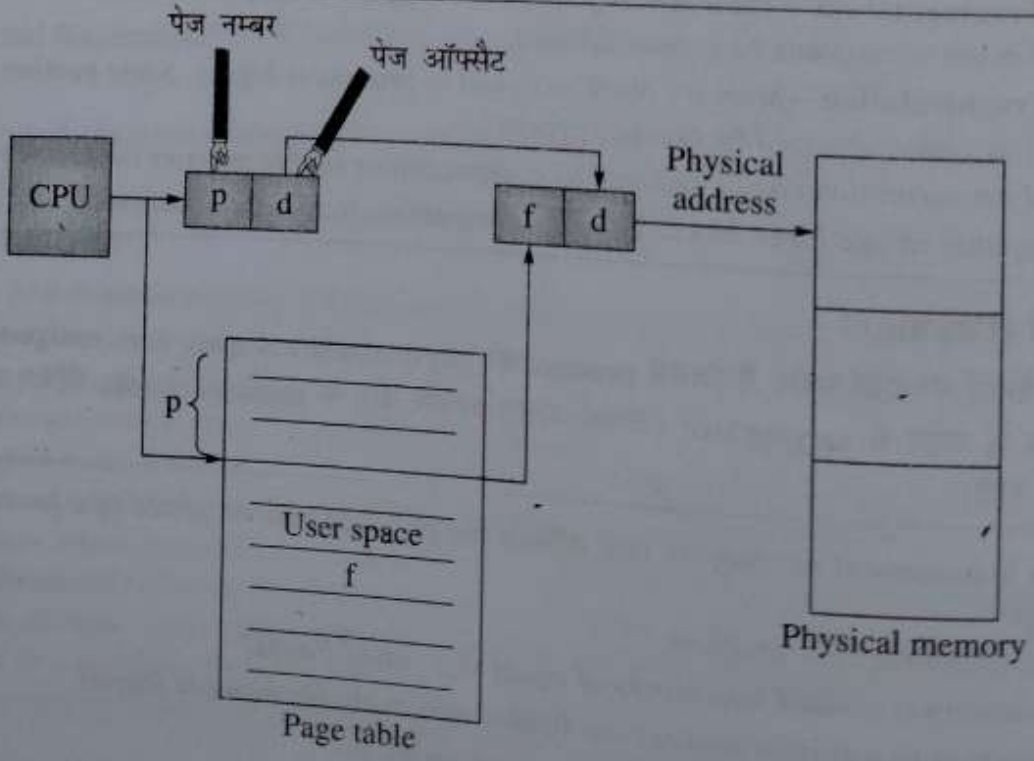
- *Technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.*
- *Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available.*
- *Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.*

पेजिंग स्कीम लागू करने हेतु physical memory को fixed-size block में विभाजित किया जाता है जिनको frames कहा जाता है तथा logical memory को छोटे blocks में विभाजित किया जाता है जिन्हें pages कहा जाता है। जब process को execute करना होता है तो उसके pages, backing store से उपलब्ध memory frames में load किये जाते हैं। Backing store को fixed-size blocks में divide किया जाता है जिसका size memory frames के समान होता है।

Paging हेतु hardware support चित्र 5.6 (a) में प्रदर्शित है। CPU द्वारा generated (उत्पन्न) प्रत्येक address को दो भागों में विभाजित किया जाता है— Page number (p) तथा Page offset (d). Page number page table हेतु index के रूप में use किया जाता है। Page table में physical memory के प्रत्येक page हेतु base address होता है। अत: physical address की गणना करने हेतु इन base address को page offset से combine किया जाता है तथा यही physical memory address, memory unit में भेजा जाता है। मैमोरी का पेजिंग मॉडल चित्र 5.6 (b) में प्रदर्शित है।

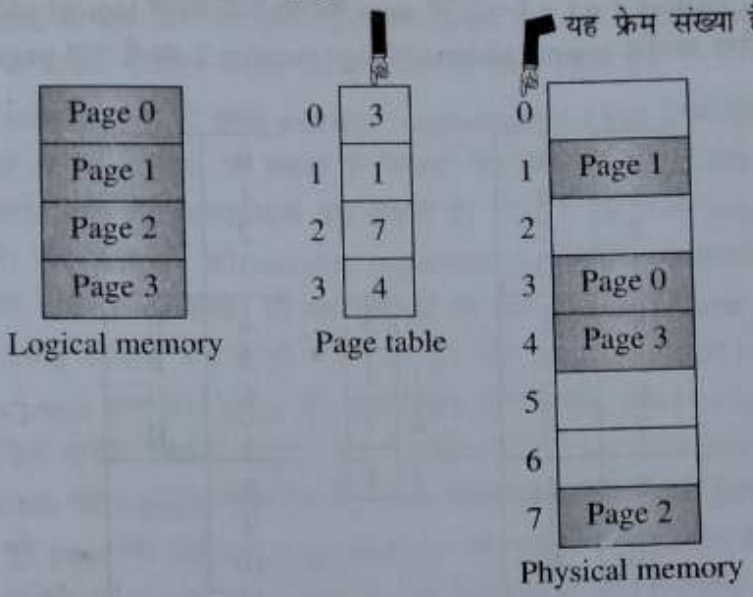
Address generated by CUP is divided into

- **Page number (p):** Page number is used as an index into a page table which contains base address of each page in physical memory.
- **Page offset (d):** Page offset is combined with base address to define the physical memory address.



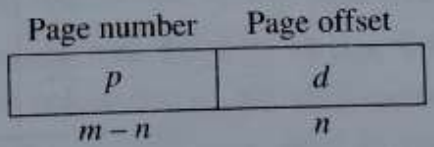
चित्र 5.6 (a)

इस table से स्पष्ट है कि page 0 फिजिकल मैमोरी फ्रेम 3 मे स्थित है



चित्र 5.6 (b)

पेज साइज़ (फ्रेम साइज़) हार्डवेयर के अनुसार परिभाषित होता है। पेज का आकार (size) सामान्यत: 2 की घात (in power of two) होता है (जैसे कि 512 bytes, 1024 bytes, 1 MB, 4 MB इत्यादि) तथा यह कम्प्यूटर के architecture (संरचना) पर निर्भर करता है। दो की घात का लाभ यह होता है कि logical address से page number तथा page offset में translation (अनुवाद, change करना) आसान हो जाता है। यदि logical address space का size 2^m है तथा page size 2^n addressing units (अर्थात् bytes या words) है, तो logical address की higher $(m - n)$ bits page number को designate करती हैं तथा n low order bits page offset को designate करती हैं। अत: logic address निम्नवत् होती है—



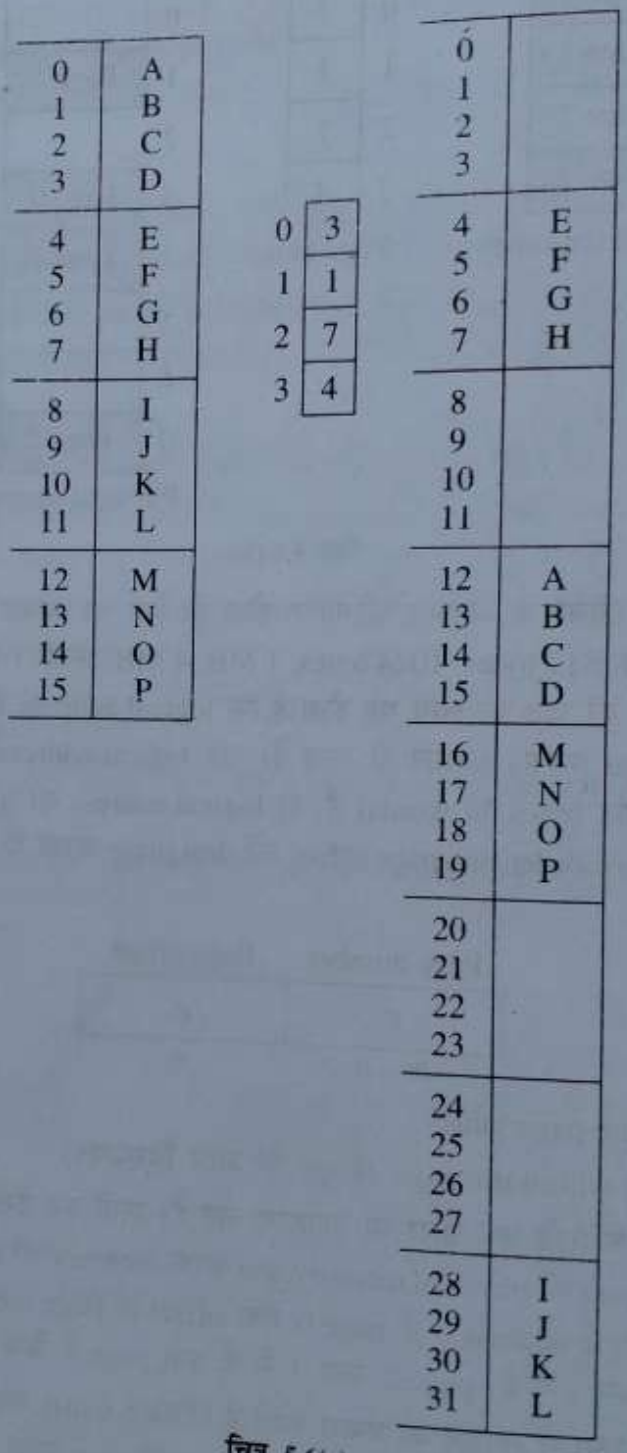
जहां p = index into the page table

d = displacment within the page. (Page के अंदर विस्थापन)

आइये, उपरोक्त को समझने के लिये एक छोटा सा उदाहरण लेते हैं। इसमें हम देखेंगे कि किस प्रकार एक 4 bytes का page size तथा 16 bytes (4 pages) की physical memory use करके memory की user's view को physical memory में map किया जा सकता है। Logic address 0 है page 0 तथा offset 0. Page table पर indexing करने पर देखें कि page 0 फ्रेम 3 में है, page 1 फ्रेम 1 में है, page 2 फ्रेम 7 में है तथा page 3 फ्रेम 4 में है।

आइये, इसको थोड़ा और विस्तार से समझने का प्रयास करते है (चित्र 5.6 (c)) आप देखे कि यह एक छोटा सा उदाहरण लिया गया है केवल समझने के उद्देश्य से तथा वास्तविक memories का size बहुत बड़ा होता है। इस उदाहरण में page size 4 byte व physical memory 32 bytes (अर्थात 8 pages की) ली गई है। अब हम यहां यह समझने का प्रयास करेंगे कि किस प्रकार user view को physical memory से map किया जाता है। Page 0 का logical address 0 है, offset भी 0 है। Page table में index करने पर हमे पता चलता है कि यह page physical memory के तीसरे फ्रेम में है। अत: mapping करने पर physical address होगा $3 \times 4 + 0 = 12$, और आप देखें कि page 0 physical memory में address 12 से प्रारम्भ हो रहा है। इसी प्रकार logical address 3 (जहां पर D stored है) का physical address calculate किया

जा सकता है अर्थात् $3 \times 4 + 3 = 15$ और आप देखें कि physical memory address 15 पर D stored है। इसी प्रकार logical address 5 physical address $4 \times 1 + 1 = 5$ पर map करेगा। इसी प्रकार logical address 9 physical address $4 \times 7 + 1 = 29$ पर map करेगा क्योंकि logical address 9 page number 2 पर है तथा page 2 फ्रेम 7 में स्थित है तथा इसका offset 1 है।



चित्र 5.6(c)

Page Table—

- *Data structure used by a virtual memory system in a computer operating system to store the mapping between virtual address and physical addresses.*
- *Virtual address is also known as Logical address and is generated by the CPU.*
- *Physical address is the address that actually exists on memory.*

आप देखें कि paging भी एक प्रकार का dynamic relocation होता है। चूंकि प्रत्येक logic address पेजिंग हार्डवेयर द्वारा किसी physical address से बंध जाता है, अत: पेजिंग का प्रयोग बेस (या relocation) registers की table प्रयोग करने के समान है, जिसमें से प्रत्येक रजिस्टर मैमोरी के एक फ्रेम के लिये प्रयुक्त हो।

जब हम पेजिंग स्कीम प्रयोग करते हैं जो इसमे external fragmentation (बाह्य फ्रैगमैन्टेशन) नहीं होता। कोई भी free फ्रेम किसी भी process को allocate किया जा सकता है जिसको उस फ्रेम की आवश्यकता हो। हालांकि, कुछ internal fragmentation (अंत: फ्रैगमैन्टेशन) की आवश्यकता पड़ सकती है। ध्यान दे कि फ्रेम्स units के रूप में (इकाई के रूप में) allocate किये जाते हैं। यदि process की memory requirements (मैमोरी आवश्यकतायें) page boundaries को concide नहीं करती (अर्थात् समरूप नहीं होती) तो यह संभव है कि last allocated frame (अंतिम एलोकेटेड फ्रेम) पूरी तरह से भरा हुआ न हो (अर्थात् आंशिक रूप से ही भरा हुआ हो) यदि page size 1024 bytes है तो 40,444 bytes के process को 39 complete pages तथा 508 bytes की आवश्यकता होगी (चूंकि 40444 ÷ 1024 = 39, अवशेष 508) अत: इसको 40 फ्रेम allocate किये जायेंगे, जिसके कारण 1024 – 508 = 516 bytes का internal fragmentation हो जायेगा। सबसे विकट स्थिति में process को n pages तथा एक byte की आवश्यकता होगी, अत: इसको $n + 1$ फ्रेम allocate करने पड़ेगें, जिसके लगभग एक पूरे page का internal fragmentation हो जायेगा (उदाहरणत: यदि page size 1024 bytes है तो 23553 byte के process को 23 pages plus one bytes की आवश्यकता होगी (क्योंकि 23553 ÷ 1024 = 23 अवशेष 1), अत: ऐसी स्थिति में 24 फ्रेम allocate करने पड़ेंगे 1024 – 1 = 1023 bytes यानि लगभग एक पूरे फ्रेम का इंटरनल फ्रैगमैंटेशन हो जायेगा।

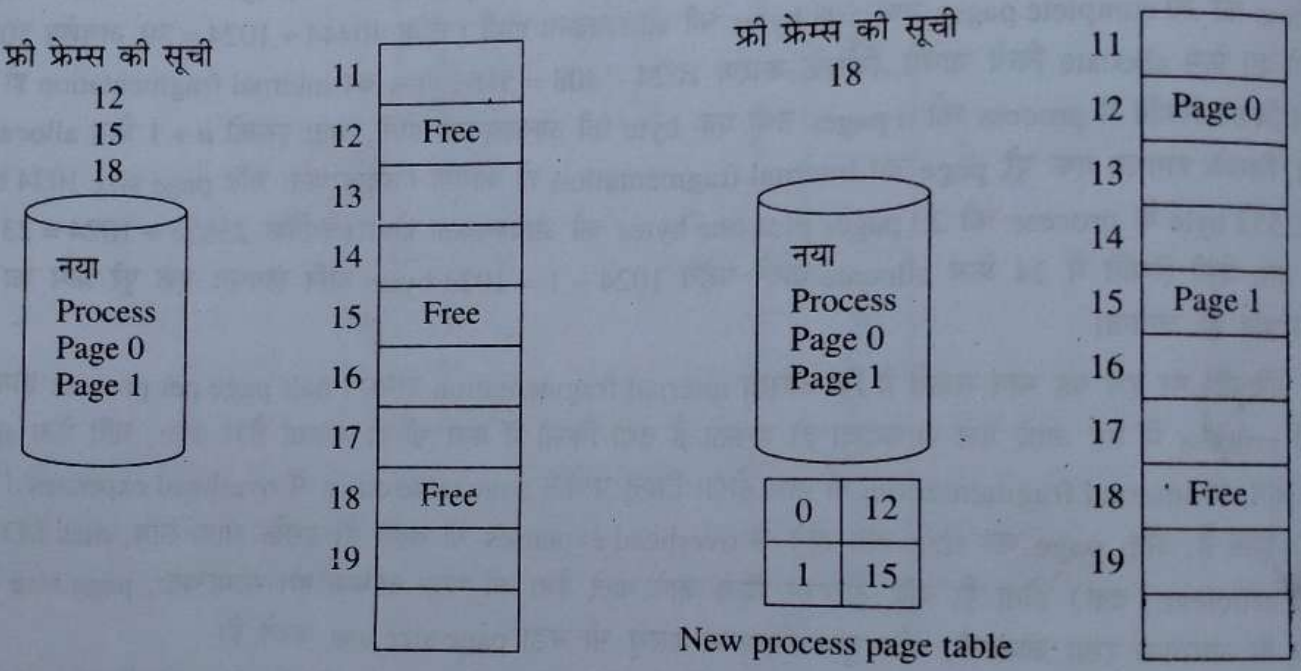
मोटे तौर पर हम यह मान सकते हैं कि औसत internal fragmentation लगभग half page per process होगा (चूंकि किसी process में यह आधे पेज से ज्यादा हो सकता है तथा किसी में कम भी हो सकता है)। अत:, यदि फ्रेम size छोटा रखा जाये, तो internal fragmentation भी कम होगा। किन्तु प्रत्येक page table entry में overhead expenses (अतिरिक्त खर्च) होता है, अत: page का size कम होने से overhead expenses भी बढ़ते हैं। इसके साथ-साथ, disk I/O भी तभी ज्यादा efficient (दक्ष) होता है, यदि ट्रांसफर किये जाने वाले डेटा की मात्रा अधिक हो। सामान्यत:, page size 4 kB या 8 kB के आसपास रखा जाता है, और कुछ सिस्टम्स इससे भी बड़ा page size use करते हैं।

सामान्यत:, प्रत्येक page table entry 4 byte लम्बी होती है पर इस size में भी परिवर्तन हो सकते हैं। एक 32-bit page table entry 2^{32} physical page frames में से किसी एक की ओर इशारा (point) करती है। अत:, यदि frame size 8 kB है तो इस स्थिति में एक 4-byte entries वाला system 2^{45} bytes अर्थात् 32 Terrabyte physical memory को address कर सकता है। ($\therefore$ 8 kB = 2^{13} bytes, $2^{32+13} = 2^{45}$)

जब कोई process execute होने के लिये system में प्रवेश करता है तो उसके size (जो कि pages में व्यक्त किया जाता है) को examine (निरीक्षण) किया जाता है। Process के प्रत्येक page को एक frame की आवश्यकता होती है। अत: यदि process में n pages होते हैं; तो memory में कम से कम n frames उपलब्ध होने चाहिये। यदि n frames उपलब्ध हैं तो यह इस process को allocate कर दिये जाते हैं। Process का पहला page किसी एक allocated फ्रेम में load कर दिया जाता है तथा फ्रेम नम्बर को पेज तालिका (page table) में डाल दिया जाता है (चित्र देखें) अब अगला page किसी दूसरे फ्रेम में load किया जाता है तथा इसका फ्रेम नम्बर भी page table में डाल दिया जाता है। इसी प्रकार सभी pages को frame allocate हो जाते हैं तथा उनकी details (विवरण) page table में डाल दिया जाता है। अत: page table में यह details होते हैं कि किसी page number को memory में कौन सा frame allocated किया गया है।

Paging की एक महत्वपूर्ण विशेषता यह है कि इस user's view of memory तथा actual physical memory के मध्य स्पष्ट separation प्रदान करती है। User program memory को एक single space के रूप मे (अर्थात अविभाजित या undivided) देखता है, (जिसमें केवल उसका ही program है) जबकि वास्तव में user program पूरी physical memory

में टुकड़ो में बिखरा होता है। (Scattered throughout the physical memory in form of frames) तथा इस physical memory में user के program के दूसरे users के program भी होते हैं। अत:, user's view of memory तथा actual physical memory के बीच सामंजस्य स्थापित करने का कार्य address translation hardware द्वारा किया जाता है। इसके द्वारा logical address को physical address में translate किया जाता है। यह map user से hidden (छिपा) रहता है तथा इसको operating system द्वारा नियंत्रित (control) किया जाता है। ध्यान दें कि user process द्वारा memory तक पहुँच नहीं बनायी जा सकती जिसका स्वामित्व (ownership) उसके पास नहीं है अर्थात अपनी page table के बाहर की memory को वह न ही address कर सकता है, न ही उससे कोई छेड़खानी कर सकता है। (page table में केवल वही memories होती है, जहां process के अपने pages stored हैं। अत:, कोई भी user किसी दूसरे user की memory location तक नहीं पहुँच सकता)।



(a) Allocation से पूर्व

(b) Allocation के पश्चात्

चित्र 5.7 Page allocation

चूंकि operating system physical memory का management (प्रबंधन) करता है, अत: उसे physical memory के allocations की जानकारी होनी चाहिये अर्थात् कुल कितने frames उपलब्ध हैं, कितने फ्रेम्स allocated हैं, कितने फ्रेम्स फ्री हैं तथा allocate किये जा सकते हैं, इत्यादि। यह सूचना frame table (फ्रेम टेबिल) नामक डेटा स्ट्रक्चर में रखी जाती है। Frame table में प्रत्येक physical page frame के लिये एक entry होती है जो यह बताती है कि अमुक physical page frame आवंटित (allocated) है या free है तथा यदि allocated है तो किस process के किस page के लिये आवंटित है।

इसके साथ-साथ operating system को यह भी ध्यान रखना होता है कि चूंकि users process user space में operate करते है, अत: सभी logical addresses को physical addresses उत्पन्न करने हेतु map किया जाना आवश्यक है। यदि कोई user system call उत्पन्न करता है (उदाहरणत: I/O उत्पन्न करने हेतु) और parameter के रूप में कोई address प्रदान करता है (उदाहरणत: कोई बफर), तो address ठीक प्रकार से map होकर सही physical address generate कर दिया जाये। Operating system प्रत्येक process के लिये page table की copy maintain करता है। जैसे कि instruction counter तथा register contents की copy maintain करता है। यदि operating system का logical address को physical

address को manually map किया जाना है तो यह copy logical address को physical address में translate करने के लिए प्रयोग की जा सकती है। यह CPU dispatcher द्वारा भी use किया जाता है, तथा इसकी सहायता से CPU dispatcher hardware page table को define कर सकता है, जब किसी process को CPU allocate किया जाता है। अत:, paging से context switch time बढ़ जाता है।

हार्डवेयर सपोर्ट (Hardware Support)—

प्रत्येक ऑपरेटिंग सिस्टम में page tables को store करने की अपनी विधि (methods) होती है। अधिकांश में, प्रत्येक process हेतु page table allocate की जाती है। पेज table की ओर point करने हेतु pointer का use किया जाता है जो कि process control block की दूसरी register values (जैसे instruction counter) को store करता है।

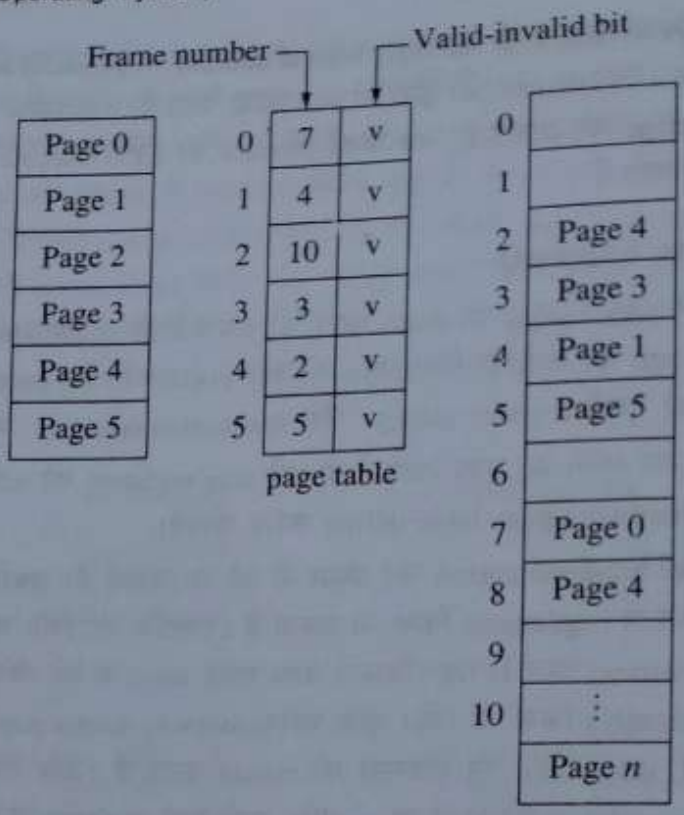
यदि डिस्पैचर को प्रौसेस स्टार्ट करने हेतु कहा जाता है तो उसे user registers को reload करना चाहिये और stored user page table से correct hardware page table define करना चाहिये।

Page table की hardware implementation कई प्रकार से की जा सकती है। सबसे सरल स्थिति में, page tables को dedicated registers के रूप में implement किया जा सकता है (हालांकि यह विधि तभी संतोषजनक होती है यदि पेज table छोटी हो (256 या 512 entries) वाली)। यह रजिस्टर्स उच्च स्पीड logic से बने होने चाहिये ताकि paging address translation दक्षतापूर्वक (efficiently) किया जा सके। चूंकि प्रत्येक memory access paging map के द्वारा होता है, अत: दक्षता महत्वपूर्ण पहलू है। CPU dispatcher इन रजिस्टर्स को reload करता है (ठीक वैसे ही, जैसे यह दूसरे registers reload करता है)। Page table registers को load या modify करने वाले registers केवल operating system ही use ही कर सकता है, चूंकि memory map को change करने का अधिकार operating system के पास होता है।

चूँकि उपरोक्त विधि छोटी page table हेतु संतोषजनक होती है, तथा आधुनिक कम्प्यूटर्स में page tables लम्बी होती हैं, अत: इनमें registers की सहायता से page table का implementation व्यवहारिक नहीं हैं। अत:, ऐसी स्थिति में page table को memory में रखा जाता है तथा एक PTBR अर्थात् page table base register page table की ओर point करता है। अत: page table में changes करने हेतु केवल इस PTBR register में change करना पड़ता है, जिससे context switching time कम हो जाता है। इस विधि से जुड़ी मुख्य समस्या है, user मैमोरी location को access करने में लगने वाला समय।

Paged वातावरण में मैमोरी प्रोटैक्शन (मैमोरी सुरक्षा) हेतु प्रत्येक फ्रेम के साथ प्रोटैक्शन बिट्स सम्बद्ध की जाती है। सामान्यत:, यह बिट्स page table में रखी जाती हैं। एक बिट यह define करने हेतु use की जाती है कि page read-write है या read only है। मैमोरी का प्रत्येक reference page table की सहायता से सही frame number find करता है। जब फिजिकल एड्रैस compute हो रहा होता है, तो protection bits को check कर के यह सुनिश्चित किया जा सकता है कि read only page पर कोई write operation perform न हो। यदि read only page पर write operation perform करने की कोशिश की जाती है तो hardware trap (error signal) उत्पन्न हो जाता है। इसी प्रकार अन्य protections जैसे कि read only, read write, execute only protection हेतु भी hardware create किया जा सकता है।

Page table की प्रत्येक entry के साथ एक अन्य अतिरिक्त बिट भी attached होती है जिसको valid-invalid bit कहा जाता है। जब यह bit "valid" पर set की जाती है, तो संबंधित page process की logical address space में होता है, तथा यह एक valid (legal या वैध) page होता है। जब यह bit invalid पर set होती है तो page process की logical address पर नहीं होता है अर्थात् invalid (illegal या अवैध) होता है। अत:, valid-invalid bit द्वारा illegal addresses trap किये जाते हैं। Operating इस बिट को प्रत्येक page हेतु set करता है ताकि प्रत्येक page तक पहुँच को allow (अनुमति प्रदान करना) या disallow (अनुमति प्रदान नहीं करना) किया जा सके।



चित्र 5.8—Page table में vatid/invalid bit

Shared Pages—

Paging का एक लाभ यह है कि common modes को share (साझा) किया जा सकता है (possibility of sharing common modes)। विशेषकर, time sharing environment में यह महत्वपूर्ण हो जाता है। यदि code reentrant code होता है तो इसको share किया जा सकता है। Reentrant code का तात्पर्य non-self-modifying code से है अर्थात् जो code execution के समय change नहीं होता, वह code reentrant कहा जाता है। अत:, इस code को दो या अधिक processes एक साथ execute कर सकते हैं। प्रत्येक process की execution हेतु registers व data storage की एक अपनी copy होती है जबकि data दोनों processes का भिन्न होता है। फिज़िकल मैमोरी में editor की केवल एक copy रखने की आवश्यकता होती है। प्रत्येक user page table editor की same physical copy को map की जाती है, जबकि data page भिन्न-भिन्न frames को map किये जाते हैं। इस प्रकार memory की saving की जा सकती है।

इसी प्रकार compilers, window systems, data base systems, run time libraries इत्यादि को भी share किया जा सकता है।

Paging is a memory-management scheme that allows the physical address space of a process to be noncontiguous. It avoids the problem of fitting memory chunks of varying sizes onto the backing store.

पेजिंग वह तकनीक है जिसमें physical memory को same sizes के blocks में बाँट दिया जाता है तथा यह blocks pages कहलाते हैं। पेज का size 2 की power (घात) में होता है, (512 bytes से 8192 bytes तक) जब कोई process execute होना होता है तो उससे संबंधित pages को उपलब्ध memory frames में load किया जाता है। Paging तकनीक से external fragmentation की समस्या उत्पन्न नहीं होती।

Process का logical space non-contiguous हो सकता है तथा free मैमोरी फ्रेम की उपलब्धता होने पर process को physical memory allocate की जा सकती है। Operating system सभी free frames को track (जानकारी रखना)

करता रहता है। किसी *n*-page size के प्रोग्राम को run करने हेतु operating system को *n* free frames की आवश्यकता होती है।

CPU द्वारा generate किया गया address दो पार्ट्स में विभाजित होती है—(i) Page number (P) जिसको कि page table में index के रूप में use किया जाता है (जिसमें physical memory के प्रत्येक पेज का base address होता है) तथा (ii) Page offset (d) जिसको कि base address से combine करके physical memory address define किया जाता है।

§ 5.11. सैगमैन्टेशन (Segmentation) :

सैगमैंटेशन एक मैमोरी मैनेजमेंट स्कीम है जो कि user view of memory को सपोर्ट करती है। Logical address space segments का समूह (collection of segments) होती है। प्रत्येक सैगमैंट का नाम व लम्बाई (name and length) होती है। अत: addresses द्वारा segment का नाम तथा segment के अंदर offset (अर्थात् displacement अर्थात् starting point से दूरी) specify किया जाता है। अत:, user द्वारा address को दो भागों में specify किया जाता है—segment name तथा offset (जबकि पेजिंग में user केवल single address specify करता है जो कि hardware द्वारा page number व offset में विभाजित किया जाता है, किन्तु यह प्रोग्रामर को दिखाई नहीं दे पाता)।

सरलीकरण हेतु (for simplification), segments की numbering की जाती है और segment name के बजाय segment numbers से refer किये जाते हैं। अत: logic address निम्न जोड़े का बना होता है—

<सैगमैंट-संख्या, विस्थापन>

अर्थात् < segment-number, offset>

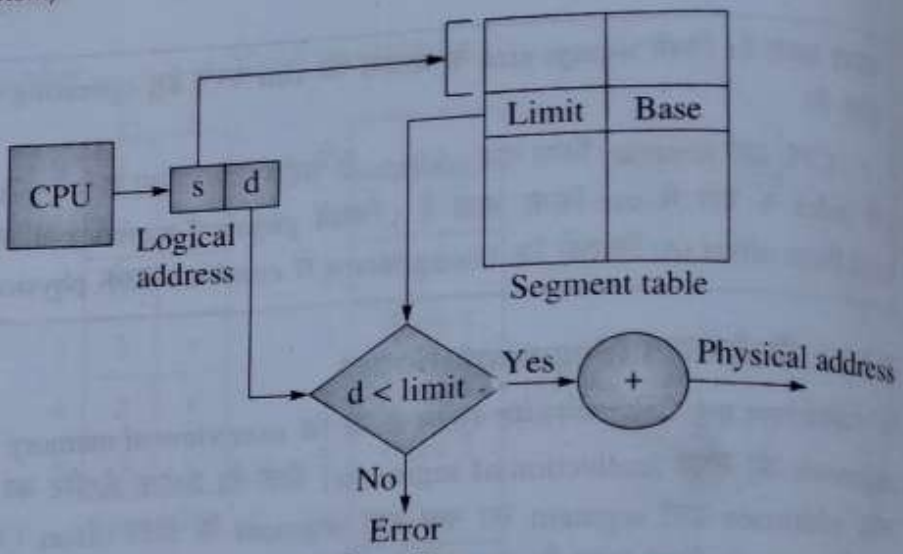
सामान्यत:, user program compile किया जाता है तथा compiler स्वत: (automatically) segments बना लेता है (जो कि input program को reflect करते हैं), उदाहरणत: एक compiler निम्न के लिये अलग-अलग segments का निर्माण करता है— code, global variables, The heap (from which the memory is allocated), the stacks used by each thread and the standard *C* library.

Segmentation—

- *Memory management scheme that supports the user-view of memory.*
- *Allows breaking of the virtual address space of a single process into segments that may be placed in non-contiguous areas of physical memory.*
- *A technique to break memory into logical pieces where each piece represents a group of related information.*
- *For example, data segments or code segment for each process, data segment for operating system and so on. Segmentation can be implemented using or without using paging.*
- *Segment are having varying sizes and thus eliminates internal fragmentation.*
- *External fragmentation still exists but to lesser extent.*

हालांकि उपरोक्त विधि से user program के objects को two dimensional address (द्विविमीय पता) से refer कर सकता है, किन्तु वास्तव physical memory one dimensional sequences of bytes (बाइट्स का एक विमीय क्रम) ही होती है। अत:, एक ऐसी implementation की आवश्यकता होती है जो user defined two dimensional address को one dimensional physical address से map कर सके। यह mapping एक segment table की सहायता से की जा सकती है। Segment table की प्रत्येक entry में segment base व segment limit होती है। Segment base में starting physical address होता है जहां segment memory मे होती है (अर्थात् segment base memory में segment का starting physical address specify करती है) जबकि segment limit segment की लम्बाई specify करती है।

Segment table का प्रयोग चित्र 5.9 (a) में प्रदर्शित है। एक logical address के दो भाग होते हैं— segment number *S* तथा उस segment में offset d. Segment number segment table को index के रूप में use किया जाता है। Logical address की offset *d* का मान 0 तथा segment limit के बीच होना चाहिये। यदि ऐसा नहीं है, (अर्थात् offset illegal है) तो error उत्पन्न हो जाता है। किन्तु यदि *d* का मान limit से कम है, इसको segment की base मे *d* add करके desired byte की physical memory का address produce किया जाता है।



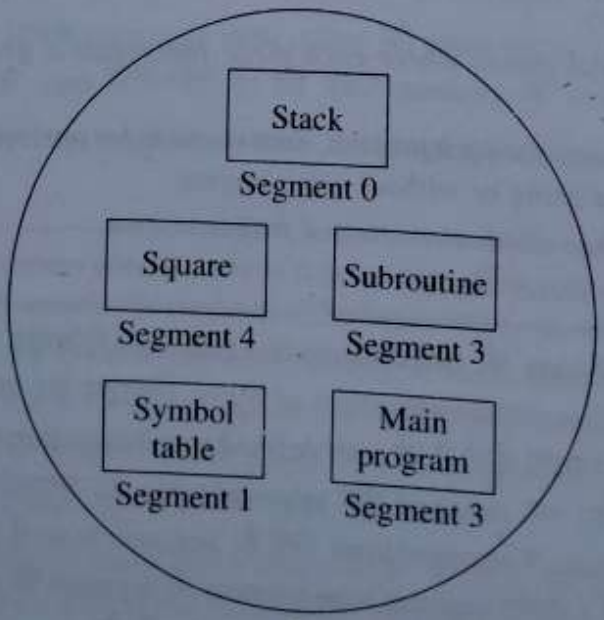
चित्र 5.9(a)—सैगमैंट हार्डवेयर (Use of Segment Table)

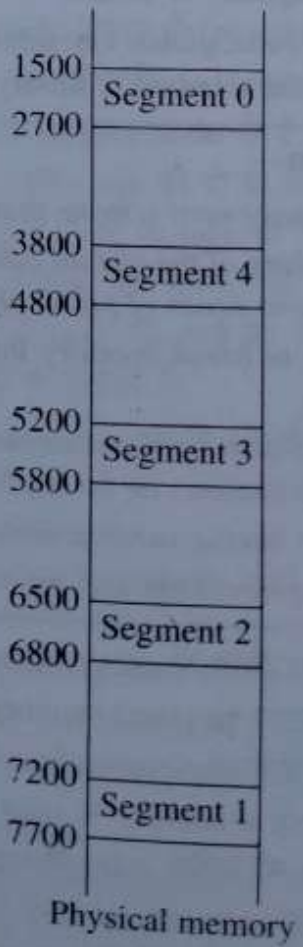
Address generated by CPU is divided into—

- **Segment Number (s)**—Segment number is used as an index into a segment table which contains base address of each segment in physical memory and a limit of segment.
- **Segment Offset or Displacement (d)**—Segment offset is first checked against limit and then is combined with base address to define the physical memory address.

एक example के रूप में चित्र 5.9 (b) को consider कीजिये। इसमें पाँच segments प्रदर्शित है, 0, 1, 2, 3, तथा 4। यह segments physical memory में stored हैं (चित्र देखें)। Segment table में प्रत्येक segment के लिये अलग-अलग

	Base	Limit
0	1500	1200
1	7200	500
2	6500	300
3	5200	600
4	3800	1000

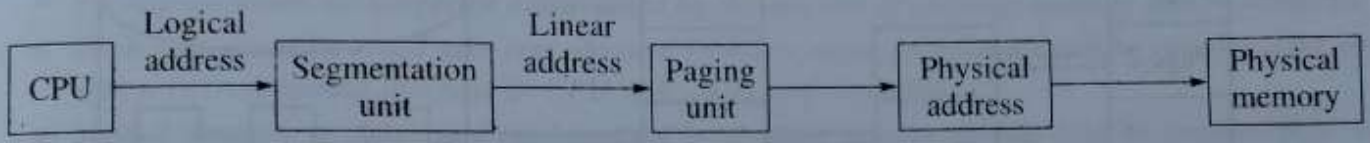




चित्र 5.9(b)—सैगमैंटेशन का उदाहरण

entry है जिसमें segment का starting address (अर्थात् base address) तथा segment की लम्बाई (limit) stored है। उदाहरणत: सैगमैंट संख्या-2 6500 से प्रारम्भ होती है तथा इसकी लम्बाई 300 bytes है। अत: यदि segment 2 के 165वीं byte को reference करने पर location 6500 + 165 = 6665 पर mapping की जाती है। इसी प्रकार segment 4 की 72वीं bytes की reference locaaion 3800 + 72 = 3872 पर map की जाती है। ऐसे ही यदि segment 3 की 728वीं byte reference की जायेगी तो operating system trap हो जायेगा (error signal उत्पन्न हो जायेगा) क्योंकि यह segment केवल 600 byte लम्बी है तथा reference इससे अधिक लम्बाई का है, जो कि illegal या invalid है।

Paging व segmentation, दोनों के लाभ भी हैं व सीमायें भी। Intel Pentium pure segmentation व segmentation with paging को support करता है। Pentium systems में CPU logical addresses generate करता है तथा segmentation unit प्रत्येक logical address के लिये linear address generate करती है। यह linear address paging unit को दिया जाता है जो कि main memory का physical address generated करती है। अत: segmentation व paging units MMU (memory management unit) का निर्माण करते हैं (चित्र 5.9 (c))।



चित्र 5.9(c)—Pentium में logical से physical address translation

Segmentation with Paging—

- *Combines the advantages of paging and segmentation.*
- *Known as 'Page the Elements'. Each segment in this scheme is divided into pages and each segment is maintained in a page table.*
- *Logical address is divided into following 3 parts :*
  - *Segment numbers (S)*
  - *Page number (P)*
  - *The displacement or offset number (D)*

सैगमैंटेशन—

सैगमैंटेशन वह तकनीक है जिसकी सहायता से मैमोरी को logical भागों मे विभाजित किया जाता है तथा प्रत्येक भाग संबंधित सूचना के समूह को व्यक्त करता है अर्थात् segmentation is a technique to break memory into logical pieces or parts where each piece represents a group of related information. उदाहरणत: प्रत्येक process की code segment या data segment, operating system की code segment इत्यादि। Segmentation को paging का use करने या paging का use किये बिना, अर्थात् दोनों ही तरीके से implement किया जा सकता है।

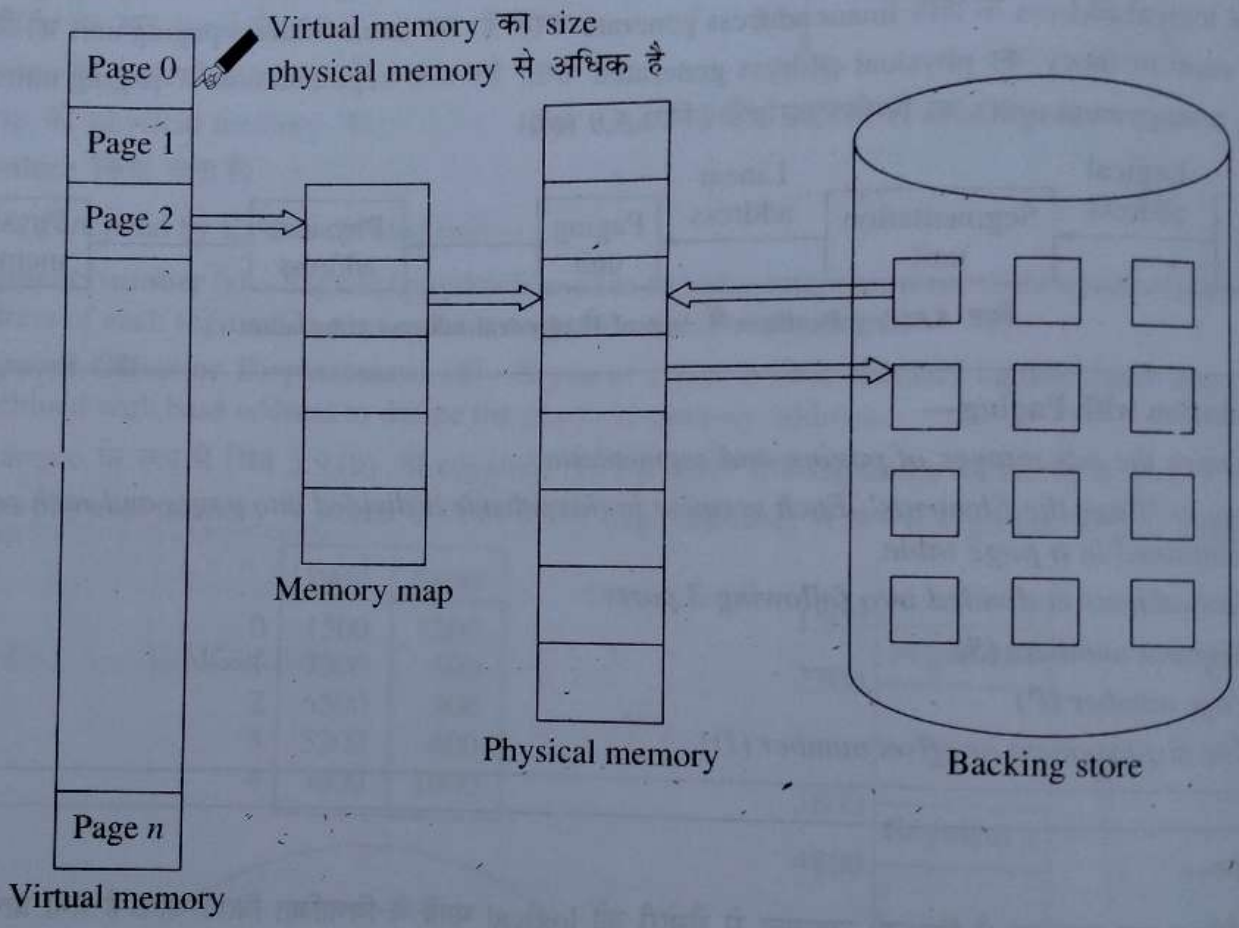
Segments के साइज़ अलग-अलग हो सकते हैं तथा इस प्रकार इसमें internal fragmentation नहीं होता। External fragmentation की संभावना रहती है किन्तु यह बहुत कम होता है।

CPU द्वारा generate किया गया address दो भागों में विभाजित होता है—(i) Segment number(s), जिसको कि segment table में index के रूप में use किया जाता है (जिसमें physical memory की प्रत्येक segment का base address होता है तथा segment की limit होती है) तथा (ii) Segment offset जिसको कि लिमिट के विरुद्ध चैक किया जाता है तथा limit से कम पाये जाने पर बेस address से combine करके physical memory address define किया जाता है।

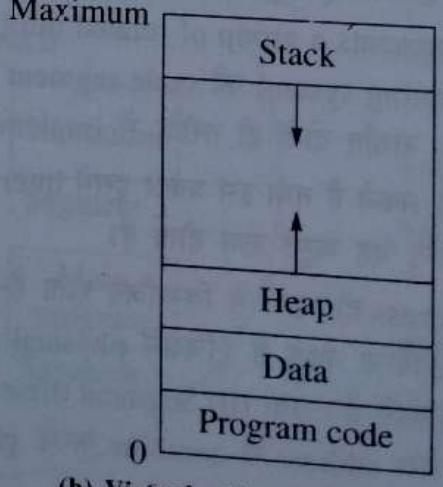
§ 5.12. वर्चुअल मैमोरी (Virtual Memory) :

अब तक जो मैनेजमेंट स्कीम्स अपने पढ़ी, उन सभी का मुख्य उद्देश्य यही है कि किस प्रकार multiprogramming वातावरण हेतु एक साथ कई process को memory में जगह प्रदान की जाये। किन्तु इन सभी स्कीम्स में process के execution से पूर्व सम्पूर्ण process को memory में होना वांछनीय है अर्थात् यदि memory में पूरा process नहीं है, केवल आंशिक रूप में है, तो वह execute नहीं किया जा सकता।

वर्चुअल मैमोरी वह तकनीक है जिसकी सहायता से process को तब भी execute किया जा सकता है। जब वह completely (पूर्णरूपेण) memory में न हो। अतः इस तकनीक का मुख्य लाभ यह है कि प्रोग्राम्स का size physical memory से अधिक रखा जा सकता है। (चित्र 5.10)



(a)



(b) Virtual address space

चित्र 5.10

Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.

This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire program is not required to be loaded fully in main memory.

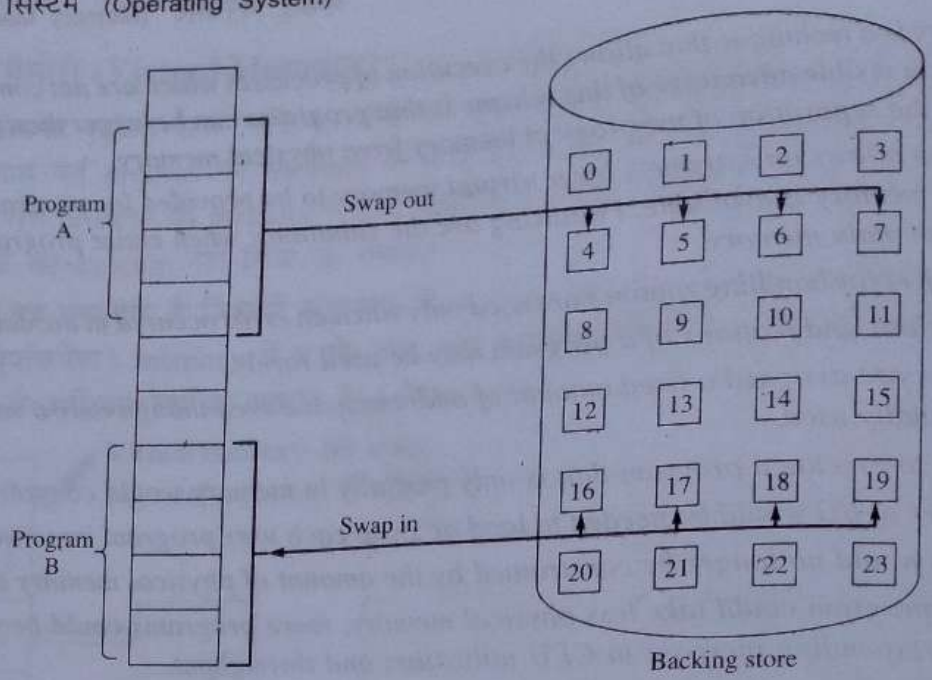
- *User written error handling routines are used only when an error occured in the data or computation.*
- *Certain options and features of a program may be used rarely.*
- *Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.*
- *The ability to execute a program that is only partially in memory would counter many benefits.*
- *Less number of I/O would be needed to load or swap each user program into memory.*
- *A program would no longer be constrained by the amount of physical memory that is available.*
- *Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.*
- *Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.*
- *Avoids reading into memory pages that will not be used in anyway, decreasing the swap time and the amount of physical memory needed.*
- *Hardware support is required to distinguish between those pages that are in memory and those pages that are on the disk using the valid-invalid bit scheme.*
- *Where valid and invalid pages can be checked by checking the bit. Marking a page will have no effect if the process never attempts to access the page.*
- *While the process executes and accesses pages that are memory resident, execution proceeds normally.*
- *Result of the operating system's failure to bring the desired page into memory.*

§ 5.13. डिमांड पेजिंग (Demand Paging) :

डिमांड पेजिंग वह मैमोरी मैनेजमेंट तकनीक है जो कि virtual memory systems में use की जाती है। इस तकनीक के द्वारा page को मैमोरी में तभी load किया जाता हैं जब प्रोग्राम execution के समय उनकी demand (माँग) होती है, तथा access न किये जाने वाले pages को physical memory में load नहीं किया जाता।

"Demand paging is a memory management technique, in which pages are only loaded in the memory in the case when they are demanded during program execution, the pages which are not accessed (not demanded) are never loaded into physical memory".

डिमांड पेजिंग paging system with swapping के समान होती है (चित्र 5.11) जहां process secondary memory (सामान्यत: disk) में रखे जाते हैं। जब किसी process को execute करना होता है, तो उसको memory में swap किया जाता है। लेकिन memory में पूरा process swap करने के बजाय lazy swapper (अर्थात् सुस्त स्वैपर) का use किया जाता है। एक lazy swapper तब तक page को memory में swap नहीं करता जब तक कि page की आवश्यकता नहीं होती। चूंकि अब हम process को pages के समूह के रूप में देखते हैं (न कि एक contiguous address space के रूप में)। अत: swapper शब्द के बजाय यहां "pager" शब्द प्रयोग करना अधिक उपयुक्त है क्योंकि swapper पूरे process को manipulate करता है जबकि pager केवल individual pages (pages of a process) से संबंध रखता है। अत:, demand paging में swapper के बजाय pager शब्द का प्रयोग किया गया है।



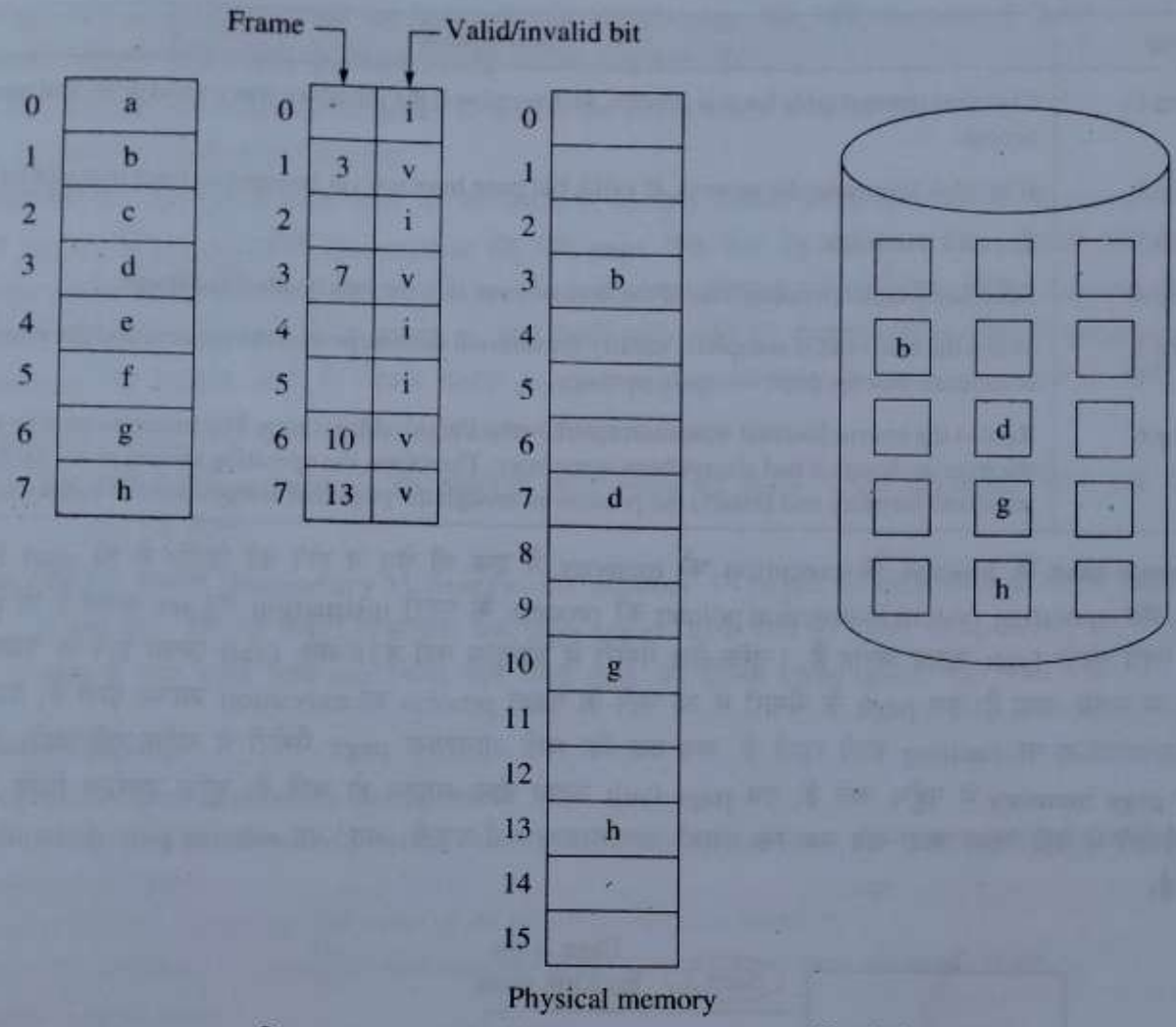
चित्र 5.11—Paged memory को contiguous disk space में transfer करना

जब किसी process को swap-in करना होता है, तो pager यह guess (अनुमान) करता है कि process को फिर से swap out करने से पूर्व कौन-कौन से pages को use किया जायेगा (अर्थात् कौन-कौन से pages के use की आवश्यकता पड़ेगी)। अत:, पूरा process swap करने के बजाय pager केवल जरूरी pages को ही memory में लेता है। अत:, pager उन memory pages को read नहीं करता जो गैर ज़रूरी है तथा जिनका प्रयोग फिलहाल नहीं किया जाना है, अत: swap time भी कम हो जाता है तथा आवश्यक physical मैमोरी की मात्रा भी घट जाती है।

इस स्कीम का प्रयोग करते समय किसी ऐसे हार्डवेयर सपोर्ट (hardware support) की आवश्यकता होती हैं जो कि मैमोरी तथा डिस्क पर स्थित pages के मध्य विभेद (distinguish, to make difference) कर सके। इसके लिये valid invalid bit scheme का प्रयोग किया जा सकता हैं। इसमें यदि bit को valid पर set किया जाता है तो इसका अभिप्राय होता है कि संबंधित page memory में है तथा legal है, जबकि इस bit को invalid पर set करने का तात्पर्य होता है कि या तो page valid ही नहीं है (अर्थात् इस process के logical address space में नहीं है) या फिर valid होने के बाबजूद इस समय मैमोरी में उपलब्ध नहीं है (अर्थात् वर्तमान में वह डिस्क में है) जो पेज मैमोरी में लाया जाता है, उसकी page table entry को set कर दिया जाता है जबकि जो page memory में नहीं होता उसे page entry table में या तो invalid mark कर दिया जाता है या फिर page entry table में उस page का disk address रहता है। इसको चित्र 5.12 में दर्शाया गया है।

यहाँ ध्यान देने की बात यह है कि यदि किसी invalid page को process access नहीं करता (अर्थात् process execution के दौरान किसी invalid mark किये गये page की अवश्यकता नहीं पड़ती) तो इस page का invalid mark करने पर execution प्रभावित नहीं होता अर्थात् normal execution चलता रहेगा। अत: यदि ठीक प्रकार से अनुमान लगाकर केवल उन्हीं pages को memory में लाया जाये जिनकी कि process execution के समय आवश्यकता होगी तो process ठीक वैसे ही run करेगा जैसे कि सभी pages अर्थात् पूरा process ही मैमोरी में हो अर्थात् यदि process execution के time केवल memory resident (मैमोरी में स्थित) pages को access करता रहे, तो execution process normally चलता रहेगा।

किन्तु यदि process एक ऐसे page को access करने की कोशिश करता है जो कि मैमोरी में लाया ही नहीं गया है तो क्या होगा? ऐसी स्थिति में page fault trap उत्पन्न हो जायेगा अर्थात् यदि process किसी ऐसे page को access करने का प्रयास करता है जो कि मैमोरी में उपलब्ध नहीं है (अर्थात् invalid page है), तो page fault trap उत्पन्न हो जाता है (trying to access an invalid page results in a page fault trap)। अत:, page table से address translate करते समय यदि paging hardware को यह पता चलता है कि invalid bit set है (अर्थात् process एक invalid page तक पहुँच बनाने



चित्र 5.12—Page table (कुछ pages main memory में नहीं हैं)

का प्रयास कर रहा है) तो page fault trap उत्पन्न होगा तथा operating system trap हो जायेगा। अत: संक्षेप में यह कहा जा सकता है कि यदि operating system वांछित page को आवश्यकता के समय मैमोरी में उपलब्ध करने में असफल हो जाता है, तो परिणामत: trap उत्पन्न हो जाता है।

Page fault की handling करने हेतु (निबटने हेतु) निम्न procedure (चित्र 5.13) (विधि) है—

(i) Pocess की एक internal table (जो कि सामान्यत: PCB process control block में रखी जाती है) की सहायता से यह चैक किया जाता है कि reference valid memory access है या invalid।

(ii) यदि reference invalid है तो process को टर्मिनेट कर दिया जाता है। यदि reference valid है किन्तु अभी तक मैमोरी में उपलब्ध नहीं है तो उसको page किया जाता है (अर्थात् डिस्क से मैमोरी में लाया जाता है)।

(iii) एक free frame की तलाश की जाती है (free frame list में से)।

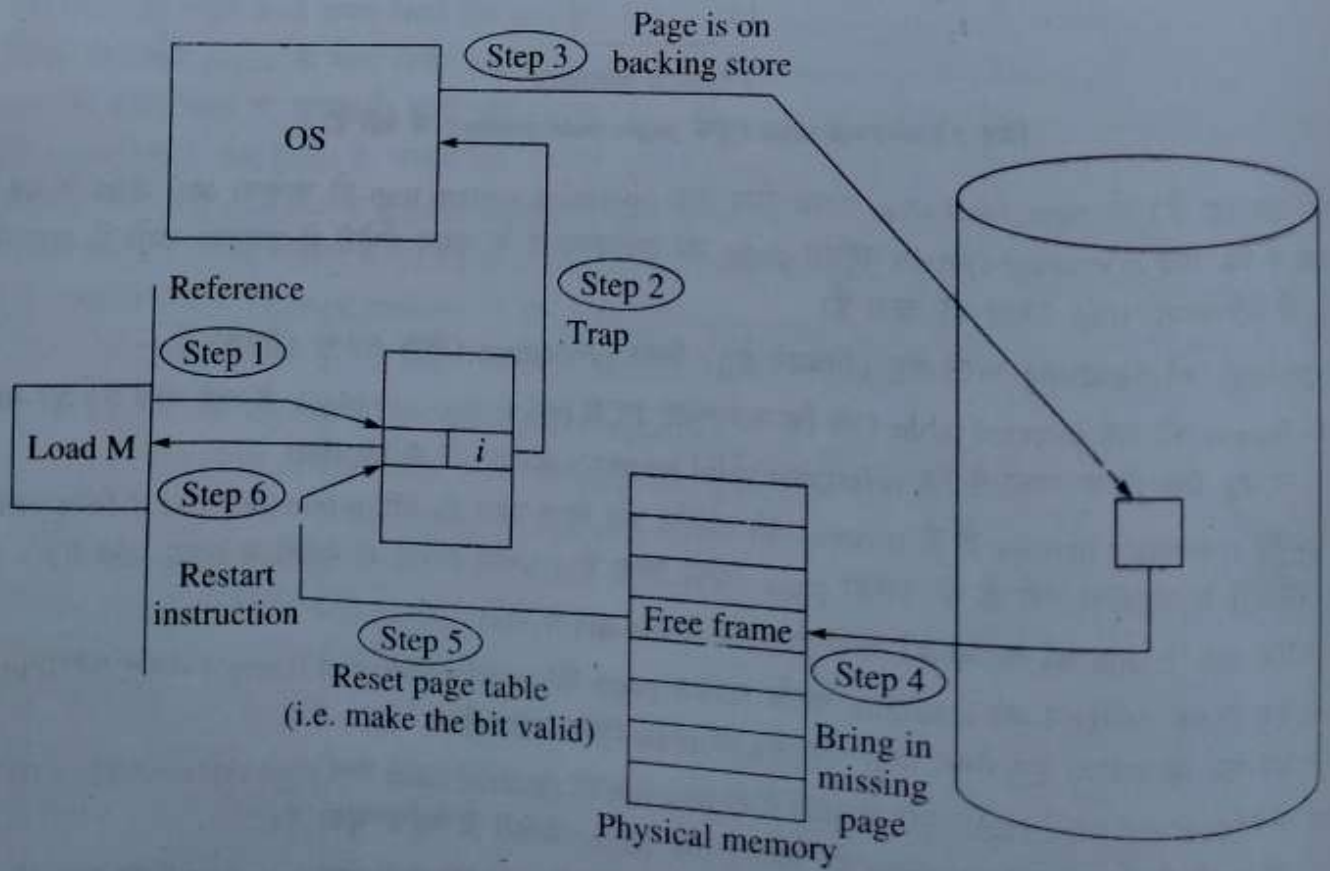
(iv) एक डिस्क ऑपरेशन को schedule करके वांछित page को newly allocated frame (अर्थात् वह frame जो इस पेज के स्वागत हेतु तैयार रखा गया है) में read किया जाता है।

(v) जब डिस्क रीड (disc read) पूरी हो जाती है तो process की internal table तथा page table modify (संशोधित) की जाती है जिससे यह indicate हो जाता है कि page, मैमोरी में पहुँच चुका है।

(vi) अब instruction को restart (पुन: प्रारम्भ) किया जाता है (जो कि trap द्वारा interrupt किया गया था)। अब चूंकि वांछित page को मैमोरी में लाया जा चुका है, अत: process अब उस page को access कर सकता है।

Step	Description
Step 1	Check an internal table for this process, to determine if the reference was a valid or invalid memory access.
Step 2	If invalid, terminate the process. If valid, but page have not yet brought in, page in the latter.
Step 3	Locate a free frame.
Step 4	Schedule a disk operation to read the desired page into the newly allocated frame.
Step 5	When the disk read is complete, modify the internal table kept with the process and the page table to indicate that the page is now in memory.
Step 6	Restart the instruction that was interrupted by the illegal address trap. The process can now access the page as though it had always been in memory. Therefore, the operating system reads the desired page into memory and restarts the process as though the page had always been in memory.

Extreme case में, process के execution को memory में एक भी पेज न होने की स्थिति में भी start किया जा सकता है। जब operating system instruction pointer को process के पहले instruction हेतु set करता है तो process page के लिये तुरन्त fault उत्पन्न करता है, (चूंकि पेज मैमोरी में उपलब्ध नहीं है)। अत: fault उत्पन्न होने के पश्चात् page memory में लाया जाता है। इस page के मैमारी में आ जाने के पश्चात् process का execution प्रारम्भ होता है, तथा प्रत्येक पेज की आवश्यकता पर faulting होती रहती है, जब तक कि सभी आवश्यक page मैमोरी में पहुँच नहीं जाते। जब सभी आवश्यक page memory में पहुँच जाते हैं, तब page fault उत्पन्न होना समाप्त हो जाते हैं। चूंकि उपरोक्त विधि में page तब तक मैमोरी में नहीं लाया जाता जब तक कि उसकी आवश्यकता नहीं पड़ती, अत: यह scheme pure demand paging कहलाती है।



चित्र 5.13

"Memory management की वह स्कीम जिसके अंतर्गत page को तभी memory में प्रवेश किया जाता है, जब उसकी आवश्यकता पड़ती है, प्योर डिमांड पेजिंग कहलाती है।"

"The memory management scheme, in which the page is not brought into the memory, until its need arises, is called pure demand paging."

सैद्धान्तिक रूप से, ऐसी संभावना व्यक्त की जा सकती है, कि कुछ प्रोग्राम्स प्रत्येक instruction के execution के समय कई नये pages को access करेंगे (instruction हेतु एक page तथा डेटा हेतु कई अन्य पेज), जिससे प्रति instruction कई page faults उत्पन्न होंगे, तथा सिस्टम की performance पर प्रतिकूल प्रभाव पड़ेगा। किन्तु, running process के विश्लेषण से यह निष्कर्ष निकलता है कि वास्तव में, ऐसी स्थिति उत्पन्न होने की संभावना अत्यंत क्षीण है। programs locality of references का व्यवहार करते हैं जिसके कारण demand paging से बेहतर performance प्राप्त हो जाती है।

Demand Paging हेतु प्रयुक्त हार्डवेयर पेजिंग व स्वैपिंग के समान होता है—

(i) **पेज टेबिल (Page Table)**—इस टेबिल में किसी entry को valid या invalid mark किया जाता है, (valid-invalid bit द्वारा)

(ii) **सैकेन्ड्री मैमारी (Secondary Memory)**—यह memory उन pages को hold करती है जो कि main memory में नहीं होते। सैकेन्ड्री मैमोरी सामान्यत: एक high-speed डिस्क होती है। इसको swap device (स्वैप युक्ति) कहा जाता है तथा इसके लिये use किया जाने वाला disk का हिस्सा swap space कहा जाता है।

Demand Paging—

- *A method of vartual memory management.*
- *In a system that use demand paging, the operating system copies a disk page into physical memory in case of page fault.*
- *Process begins execution sith none of its pages in physical memory.*
- *Pages should only be brotught into memory if the execution process demands them.*
- *Called lazy loading.*
- *In pure swapping,where all memory for a process is swapped from secondary storage to main memory during the process startup.*

To achieve this process a page table implementaion is used. The page table maps logical memory to physical memory. The page table uses a bitwise operator to mark whether a page is valid or invalid. A valid page currently resides in main memory. An invalid page is currently resides in secondary memory. When a process tries to access a page, the following steps are taken:

- *Attempt to access page.*
- *If page is valid then continue processing.*
- *If page is invalid a page-fault trap occurs.*
- *Check if the memory reference is a valid reference to a location secondary memory. If not, the process is terminated (illegal memory access). Otherwise, we have to page in the required page.*
- *Schedule disk operation to read the desired page into main memory.*
- *Restart the instruction.*

Advantages

- *Only loads pages that are demanded by the executing process.*
- *Suitable in multi-programming, as there is more space in main memory, more process can be loaded reducing context switching time which utilizes large amount of resources.*

- *Large virtual memory.*
- *More efficient use of memory.*
- *No limit to degree of multi-programming.*
- *Less loading latency occurs at program startup.*
- *Reduce the bill of material (BOM) cost in smart phones.*

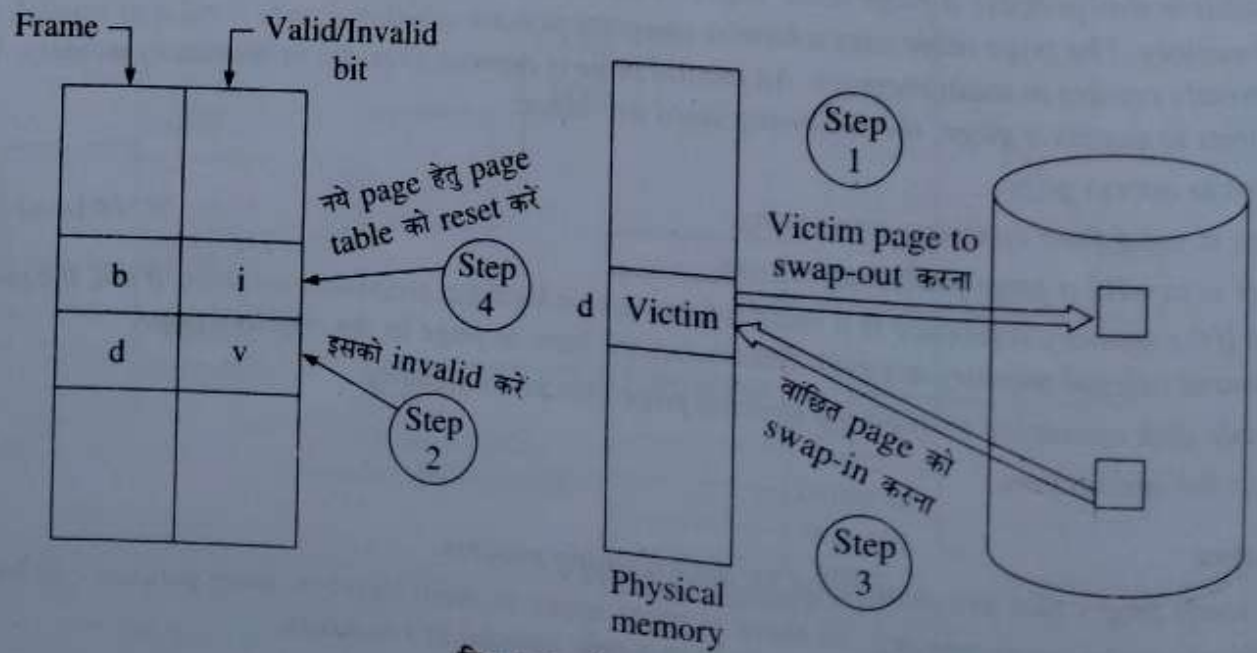
Disadvantages

- *Individual programs face extra latency when they access a page for the first time.*
- *Programs running on low-cost, low-power embedded systems may not have a memory management unit that supports the page replacement strategy.*
- *Memory management with page replacement algorithms becomes slightly more complex.*
- *Possible security risks, including vulnerability to timing attacks.*
- *Thrashing which may occur due to repeated page faults.*
- *Number of tables and amount of process overhead for handling page interrupts are more than simple paged management strategies.*

§ 5.14. पेज रिप्लेसमैंट (Page Replacement) :

अभी तक किये गये discussion में हम यह मानते आये है कि कोई भी page faults ज्यादा से ज्यादा एक बार हो सकता है (जब उसे पहली बार referenced किया गया है) यह स्थिति हर बार नहीं होती। यदि 20 पेजों वाला कोई प्रोसेस केवल उनमें से आधे को यूज़ करता है तो demand paging उनमें से 10 पेजों को (जो कि कभी use नहीं होने) लोड करने हेतु आवश्यक I/O को बचाती है, इस प्रकार मल्टी प्रोग्रामिंग की डिग्री दोगुनी हो जाती है।

Page replacement का तातपर्य यह है कि यदि कोई फ्रेम free नहीं है तो एक ऐसा फ्रेम ढूंढ़ा जाए जो वर्तमान में उपयोग नहीं किया जा रहा है और उसको free कर दिया जाए। किसी फ्रेम को free करने हेतु उसकी contents को swap space पर लिख दिया जाता है तथा page table तथा अन्य tables को change कर दिया जाता है जिससे यह पता चल जाता है कि page अब memory में नहीं है (चित्र 5.14 में देखें) अब इस मुक्त हुए फ्रेम को उस page को hold करने हेतु use किया जा सकता है जिसके कारण process fault हुआ था। इसके लिए page fault routine को निम्नवत् modify किया जा सकता है—



चित्र 5.14—पेज रिप्लेसमैंट

1. वांछित page की डिस्क में लोकेशन ज्ञात करें।
2. एक free frame की तलाश करें।
   (a) अगर कोई फ्री फ्रेम है तो उसका प्रयोग करें।
   (b) यदि कोई free फ्रेम नहीं है तो किसी page replacement algorithm का उपयोग करके एक victim फ्रेम (अर्थात् शिकार फ्रेम) करने का चुनाव करें।
   (c) इस victim फ्रेम को डिस्क पर भेजें तथा उसी अनुसार पेज टेबल तथा अन्य टेबल चेन्ज करें।
3. वांछित पेज को नये फ्री हुए फ्रेम पर read करें तथा उसी अनुसार पेज तथा फ्रेम टेबिलस को चेन्ज करें।
4. User प्रोसेस को फिर से स्टार्ट करें।

Page Replacement Algorithm—

- *Techniques using which Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated.*
- *Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.*
- *When the page that was selected for replacement and was paged out, is referenced again then it has to read in from disk, and this requires for I/O completion.*
- *This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm*
- *Looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself.*
- *Evaluation of an algorithm by running it on a particular string of memory reference and computing the number of page faults.*

ध्यान दें कि यदि कोई फ्रेम फ्री नहीं है तो दो पेज ट्रान्सफर की आवश्यकता होगी (एक 0 तथा दूसरा 1) इस स्थिति के कारण page fault सर्विस time दो गुना हो जाएगा और access time भी बढ़ जाएगा।

इस overhead (खर्च) को कम करने हेतु एक modify bit (या डर्टी बिट) का प्रयोग किया जाता है। जब यह स्कीम यूज़ की जाती है तो प्रत्येक पेज या फ्रेम के साथ हार्डवेयर में एक मोडिफाई बिट सम्बन्धित होती है। जब page पर कोई वर्ड या bit लिखी जाती है तो हार्डवेयर page की modify bit को set कर देता है जिससे यह पता चल जाता है कि page modify किया गया है। जब किसी page को replacement के लिए select किया जाता है तो उसकी modify bit को examine (निरीक्षण) किया जाता है। यदि बिट set होती है तो इससे ये पता चल जाता है कि disk में रीड किये जाने के बाद page modify हुआ है इस स्थिति में disk को पेज में लिखना आवश्यक हो जाता है। किन्तु यदि modify bit सेट नहीं है तो इसका मतलब यह हुआ कि memory में read होने के बाद page modify नहीं हुआ है अर्थात् उसमें कोई overwriting नहीं हुई है तो ऐसी स्थिति में उसे वापस डिस्क में लिखने की आवश्यकता नहीं है क्योंकि उसकी copy पहले से ही डिस्क में मौजूद है। यह तकनीक read only pages पर भी यूज़ की जा सकती है क्योंकि इनको modify नहीं किया जा सकता है। उपरोक्त विधि page fault का service time काफी कम कर देती है क्योंकि यदि page modify नहीं हुआ है तो I/O time आधा हो जाता है।

Page रिप्लेसमेन्ट की सहायता से एक छोटी (physical) फिजीकल मैमोरी होने पर भी यूज़र को बड़ी मात्रा में logical मैमोरी प्रदान की जा सकती है।

डिमान्ड पेजिंग के बिना यूज़र address तथा physical address अलग-अलग हो सकते हैं। डिमान्ड पेजिंग के बाद लॉजिकल address का size फिजिकल मैमोरी द्वारा सीमित नहीं होता। उदाहरणत: यदि यूज़र का प्रोसेस 30 page का है तो demand paging की सहायता से उसे 15 फ्रेम में किया जा सकता है तथा page replacement algorithm की सहायता से आवश्यकता पड़ने पर free frame की तलाश की जा सकती है। यदि page modify हुआ है तो disk पर copy कर दिया जाता है यदि बाद में इस page की फिर आवश्यकता पड़ती है तो page fault उत्पन्न होता है तथा फिर उस page को memory में लाना पड़ता है।

अत: demand paging को उत्पन्न करने हेतु दो मुख्य बातें ध्यान में लेनी पड़ती है अर्थात् उचित frame-allocation algorithm तथा page replacement algorithm विकसित करनी पड़ती है। यदि memory में कई सारे प्रौसेस हैं तो यह डिसाइड करना पड़ता है कि प्रत्येक प्रौसेस को कितने फ्रेम ऐलोकेट करते है इसके साथ-साथ जब page replacement की आवश्यकता पड़ती है तो उस फ्रेम का चुनाव करना पड़ता है अत: इसके लिए उपयुक्त algorithm का design एक महत्वपूर्ण कार्य है तथा demand paging की विधियों में थोड़ा सुधार लाने पर भी सिस्टम की परफॉरमेन्स में काफी सुधार आता है। सामान्य तौर जिस रिप्लेसमेन्ट algorithm की page fault rate कम होगी वह अच्छा परफॉरमेन्स दे सकती है।

Page replacement algorithm कई प्रकार की होती हैं तथा प्रत्येक operating system की अपनी अलग algorithm होती है। किसी algorithm का मूल्यांकन (evaluation) करने के लिए उसे memory references की string पर run किया जाता है तथा page faults की संख्या की गणना की जाती है। इस memory reference string को reference string कहा जाता है तथा इनको random number generator की सहायता से कृत्रिम रूप से (artificial) उत्पन्न किया जा सकता है। Page fault की संख्या की गणना के लिए हमें page फ्रेम की संख्या की जानकारी भी आवश्यक होती है। जैसे-जैसे उपलब्ध फ्रेम की संख्या बढ़ती है page fault की संख्या कम हो जाती है हालांकि physical memory बढ़ने पर फ्रेम्स की संख्या बढ़ती है।

Page Replacement Algorithms—

- *Decide which memory pages to page out (swap out, write to disk) when a page of memory needs to be allocated.*
- *Paging happens when a page fault occurs and a free page cannot be used to satisfy the allocation, either because there are none, or because the number of free pages is lower than some threshold.*
- *When the page that was selected for replacement and paged out is referenced again it has to be paged in (read in from disk), and this involves waiting for I/O completion.*
- *This determines the quality of the page replacement algorithm: the less time waiting for page-ins, the better the algorithm.*
- *A page replacement algorithm looks at the limited information about accesses to the pages provided by hardware, and tries to guess which pages should be replaced to minimize the total number of page misses, while balancing this with the costs (primary storage and processor time) of the algorithm itself.*

Local vs. Global Replacement—

Replacement algorithms can be *local* or *global*.

- *When a process incurs a page fault, a local page replacement algorithm selects for replacement some page that belongs to that same process (or a group of processes sharing a memory partition). A global replacement algorithm is free to select any page in memory.*
- *Local page replacement assumes some form of memory partitioning that determines how many pages are to be assigned to a given process or a group of processes. Most popular forms of partitioning are fixed partitioning and balanced set algorithms based on the working set model.*
- *The advantage of local page replacement is its scalability: each process can handle its page faults independently without contending for some shared global data structure.*

Precleaning—

- *Most replacement algorithms simply return the target page as their result.*
- *This means that if target page is dirty (that is, contains data that have to be written to the stable storage before page can be reclaimed), I/O has to be initiated to send that page to the stable storage (to clean the page).*
- *To deal with this situation, various precleaning policies are implemented.*
- *Precleaning is the mechanism that starts I/O on dirty pages that are (likely) to be replaced soon.*
- *The idea is that by the time the precleaned page is actually selected for the replacement, the I/O will complete and the page will be clean.*
- *Precleaning assumes that it is possible to identify pages that will be replaced next. Precleaning that is too eager can waste I/O bandwidth by writing pages that manage to get re-dirtied before being selected for replacement.*

Anticipatory Paging—

- *Some systems use demand paging-waiting until a page is actually requested before loading it into RAM.*
- *Other systems attempt to reduce latency by guessing which pages not in RAM are likely to be needed soon, and pre-loading such pages into RAM, before that page is requested. (This is often in combination with pre-cleaning, which guesses which pages currently in RAM are not likely to be needed soon, and pre-writing them out to storage).*
- *When a page fault occurs, "anticipatory paging" systems will not only bring in the referenced page, but also the next few consecutive pages (analogous to a prefetch input queue in a CPU).*
- *The swap prefetch mechanism goes even further in loading pages (even if they are not consecutive) that are likely to be needed soon.*

§ 5.15. विभिन्न पेज रिप्लेसमैंट एल्गोरिद्म्स (Page Replacement Algorithms) :

FIFO Page Replacement Algorithm—FIFO page replacement algorithm सबसे सरलतम page replacement algorithm है। FIFO अर्थात् First In First Out अर्थात् जो सबसे पहले आएगा वो सबसे पहले जाएगा अत: FIFO replacement algorithm में प्रत्येक page के साथ वह समय सम्बन्धित कर दिया जाता है जबकि वह मैमोरी में लाया गया था जो page सबसे पुराना होता है उसको replace कर दिया जाता है। ध्यान दें कि इस algorithm को implement करने के लिए सभी process का time record करना आवश्यक नहीं है। इसके बजाय एक FIFO QUEUE में memory के सभी page को hold किया जा सकता है। जब कोई page replace करना होता है तो queue के head वाला page replace कर दिया जाता है। जब कोई नया page memory में लाया जाता है तो उसको queue की tail पर Insert किया जाता है।

Page replacement algorithm समझने व प्रोग्राम करने में आसान होती है किन्तु इसकी परफॉरमेन्स हमेशा अच्छी नहीं होती क्योंकि यह जरूरी नहीं की सबसे पुराना page अब प्रयोग में न हो, ऐसा भी सम्भव है कि वह पेज अधिक use होने वाला page हो, यदि ऐसा हुआ तो page fault की संख्या बढ़ सकती है।

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

आइये, FIFO page-replacement algorithm को समझने हेतु उदाहरण के तौर पर निम्न reference string लेते हैं—

5, 2, 0, 1, 2, 1, 6, 0, 5, 2, 0, 5, 1, 0, 1, 6, 5, 6

माना कि memory में तीन फ्रेम्स है— पहले तीन faults से page faults उत्पन्न होने पर पहले तीन page निम्नवत् आ जायेंगे—

(5, 2, 0)

अगला reference (अर्थात् 1) page fault उत्पन्न करेगा, जिससे पेज 5 replace होगा (क्योंकि वह memory में सबसे पहले आया था)।

(1, 2, 0)

अगला reference अर्थात् 2 page fault उत्पन्न नहीं करेगा, क्योंकि page 2 memory में उप्लब्ध है।

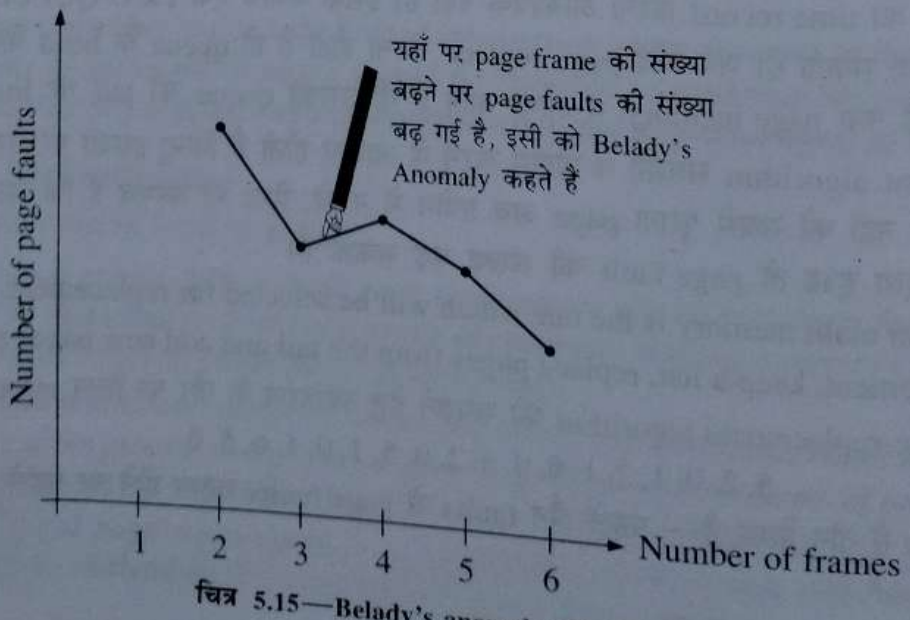
उपरोक्त scheme को निम्न तालिका द्वारा समझा जा सकता है—

Reference	Page Fault (Yes, No)	Pages Availble in Memory	Oldest Page
5	Yes	5	5
2	Yes	5, 2	5
0	Yes	5, 2, 0	5
1	Yes	1, 2, 0	2
2	No	1, 2, 0	2
1	No	1, 2, 0	2
6	Yes	1, 6, 0	0
0	No	1, 6, 0	0
5	Yes	1, 6, 5	1
2	Yes	1, 6, 5	6
0	Yes	2, 0, 5	5
5	No	2, 0, 5	5
1	Yes	2, 0, 1	2
0	No	2, 0, 1	2

तालिका में आप देखें कि प्रत्येक बार जब page fault उत्पन्न होता है तो सबसे पुराने पेज को हटाकर नया page लाया जाता है। कुल 9 page faults उत्पन्न हुये है।

FIFO replacements से संबंधित एक अन्य समस्या है Belady's anomaly।

सामान्यत:, यदि page frames की संख्या बढ़ाई जाये तो page faults की संख्या कम हो जानी चाहिये। किन्तु, कुछ strings के लिये page frames बढ़ाने पर पेज फॉल्ट्स की संख्या बढ़ जाती है। इस phenomenon को Belady's anomaly कहा जाता है (चित्र 5.15 देखें)।



चित्र 5.15—Belady's anomaly का उदाहरण

"Page frames की संख्या बढ़ाने पर कुछ memory process patterns के लिये page faults की संख्या बढ़ जाने की विसंगति को Belady's anomaly कहा जाता है।"

"The phenomenon where increasing the number of page frame results in the increase in number of page faults for a given memory access patterns is called Belady's anomaly. It is seen only in FIFO page replacement policy."

Optimal Page Replacement—

Optimal page replacement की page fault rate अन्य algorithm से कम रहती है तथा इसमें Belady's Anamoly के समस्या भी नहीं होती अर्थात् जैसे-जैसे frames की संख्या बढ़ेगी, page faults की संख्या भी हमेशा कम ही होगी, कभी बढ़ेगी नहीं। इस algorithm को OPT या MIN कहा जाता है तथा यह निम्नवत् होती है—

उस पेज को replace करें जो कि सबसे लम्बे समय तक use नहीं होगा।

उपरोक्त एल्गोरिद्मस यह गारन्टी प्रदान करती है कि page fault rate न्यूनतम होगा किन्तु इसको implement करना कठिन होता है क्योंकि इसमें reference string की future knowledge (आगे कौन से page reference किये जाते हैं, की जानकारी) आवश्यक होती है। अत: इसको मुख्यत: तुलनात्मक अध्ययन हेतु use किया जाता है।

- *An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.*
- *Replace the page that will not be used for the longest period of time . Use the time when a page is to be used.*

LRU Page Replacement—

चूंकि optimal algorithm feasible (व्यवहार्य) नहीं है, किन्तु optimal algorithm का approximation (सन्निकटत: होना) possible है। आप यदि FIFO तथा OPT algorithms की तुलना करें तो आप पायेंगे कि एक यह देखती है कि page कितनी देर पहले memory में लाया गया जबकि दूसरी यह देखती है कि page को मैमोरी में कितनी देर बाद पुन: लाने की आवश्यकता पड़ेगी। LRU का सिद्धान्त है कि ऐसा पेज देखा जाये जो कि सबसे ज्यादा देर तक memory में use नहीं किया गया है अर्थात् past को future के approximation के रूप में use किया जाये।

"LRU अर्थात् least recently used page replacement algorithm में उस page को replace किया जाता है जो कि सबसे अधिक period के लिये use नहीं किया गया है।"

"LRU page replacement algorithm replaces that page from the memory which has been not used for the longest period of time"

हालांकि LRU page replacement algorithm अच्छी algorithm है, परन्तु इसको implement करने में अतिरिक्त हार्डवेयर की आवश्यकता होती है क्योंकि इसमें last use किये गये फ्रेम के क्रम को निर्धारित करना आवश्यक हो जाता है। यह निम्नवत् किया जा सकता है—

काउन्टर्स द्वारा (Using Counters)—

इसमें प्रत्येक page table entry में time of field use नामक field attach कर दी जाती है तथा CPU से logical clock या counter add कर दिया जाता है। प्रत्येक memory reference पर clock को increment (एक बढ़ाना) कर दिया जाता है। जब भी page को reference किया जाता है तो clock register की contents को उस page की page table entry की time of use field में copy कर दिया जाता है। इस प्रकार प्रत्येक पेज को अंतिम time of use या time of last reference उपलब्ध रहता है। अत: जिसकी time value न्यूनतम होती है (अर्थात् जो पेज सबसे ज्यादा समय तक memory में बैठा है तथा use नहीं हुआ है) को replace कर दिया जाता है। इस स्कीम में page table को सर्च करना आवश्यक हो

जाता है (LRU page की तलाश हेतु) एवं प्रत्येक memory access हेतु memory write की आवश्यकता भी पड़ती है (page table में time of use field write करने हेतु)। Page tables change करने के time भी maintain करने पड़ते है (CPU scheduling के कारण)। Clock के overflow का भी ध्यान रखता पड़ता है।

Stack—

LRU implement करने की एक अन्य विधि है—page numbers का stack रखना। जब भी page reference किया जाता है तो उसे stack से remove करके सबसे top पर रख दिया जाता है। इस प्रकार सबसे नवीन used किया गया page (most recently used) stack के top पर रहता है तथा सबसे पूर्व में use किया गया page (least recently used) stack के bottom पर रहता है।

ध्यान दें कि optimal replacement algorithm तथा LRU algorithms stack algorithms कहलाती है। Optimal replacement की LRU replacement में भी Belady's Anamoly की समस्या उत्पन्न नहीं होती। Stack algorithm वह algorithm होती है जिसके लिये यह दर्शाया जा सकता है कि n frames हेतु memory में pages का समूह $(n + 1)$ frames हेतु memory में pages के समूह का उपसमूह होता है।

"An stack algorithm is an algorithm for which it can be shown that the set of pages in memory for n frames is always a set of pages that would be in memory units n + 1 frames."

LRU replacement हेतु memory में pages का समूह *n*-most recently used pages होता है। यदि frames की संख्या बढ़ा दी जाये, तो फिर भी यह *n* pages most recently referenced pages ही रहेंगे तथा memory में ही होंगे।

काउन्टिंग आधारित पेज रिप्लेसमैंट—

पेज रिप्लेसमैंट हेतु कई अन्य algorithms भी use की जाती है जिनमें प्रत्येक page को references की संख्या को counter द्वारा count किया जाता है। इस प्रकार की page replacement algorithms के उदाहरण है— LFU (least frequently used) page replacement algorithm या MFU (most frequently used) page replacement algorithms.

Least frequently used page replacement algorithm (अर्थात् LFU page replacement algorithm) में सबसे small count वाले page (अर्थात् अब तक जिसका प्रयोग सबसे कम हुआ है) को replace किया जाता है। इसके पीछे यह वजह है कि सबसे actively (सक्रियता से या अधिक) use किये जाने वाले पेज का count भी सबसे ज्यादा होगा। हालांकि ऐसा भी संभव है कि किसी process के शुरूआती भाग में कोई पेज ज्यादा use हुआ हो (तथा उसकी count बहुत अधिक हो) किन्तु बाद में यदि उसका प्रयोग नहीं भी है तो भी वह मैमोरी में बना रहेगा। इसका solution यह है कि निश्चित समय अंतराल बाद काउन्ट्स को right shift करके काउन्ट को कम किया जाये।

Least Frequently Used (LFU) Algorithm

- Page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

MFU अर्थात् most frequently used page replacement algorithm सबसे अधिक used page को हटा दिया जाता है। इसके पीछे यह सोच है कि जिस page की count सबसे कम है, शायद आगे उसकी जरूरत पड़ेगी, इसलिये उसे memory में ही रहने दिया जाये।

उपरोक्त दोनों विधियां (MFU तथा LFU) का implementation मंहगा होता है तथा यह OPT को approximate भी नहीं कर पाती, अत: यह कम प्रचलन में हैं।

Most Frequently Used (MFU) Algorithm

- *This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.*

Page Buffering Algorithm

- *To get process start quickly, keep a pool of free frames.*
- *On page fault, select a page to be replaced.*
- *Write new page in the frame of free pool, mark the page table and restart the process.*
- *Now write the dirty page out of disk and place the frame holding replaced page in free pool.*

प्रश्नावली

1. (a) Memory management से आप क्या समझते हैं? इसके क्या उद्देश्य हैं।
   (b) निम्न को समझाइये–
   (i) Swapping (ii) Paging (iii) Segmentation
   (iv) Demand paging (v) Multiple partitions
   (c) Page replacement क्या है। विभिन्न page replacement policies को समझाइये।
2. Page fault कब होता है? Page fault उत्पन्न होने पर operating system क्या-क्या कार्य करता है?
3. Multiple partition को उदाहरण सहित समझायें।

6 Chapter

डैडलॉक (DEADLOCK)

THINK ABOUT IT

It's not about how much you do, but how much love you put into what you do that counts.

— Mother Teresa

Every day is an opportunity to be creative - the canvas is your mind, the brushes and colours are your thoughts and feelings, the panorama is your story, the complete picture is a work of art called, 'my life'. Be careful what you put on the canvas of your mind today - it matters.

— Innerspace

§ 6.1. परिचय (Introduction) :

किसी मल्टीप्रोग्रामिंग वातावरण में कई processes सीमित resources को प्राप्त करने की कोशिश में रहते हैं अर्थात् चूंकि resources की संख्या सीमित होती है, अत: जब कोई process किसी resource हेतु request करता है जबकि यदि वह resource उस समय उपलब्ध न हो तथा किसी दूसरे process द्वारा use किया जा रहा हो तो process wait state में चला जाता है। कभी-कभी ऐसी स्थिति भी उत्पन्न हो जाती है कि waiting state से बाहर आना ही संभव नहीं हो पाता क्योंकि जिस resource के लिये वह request कर रहा है वह किसी अन्य waiting process द्वारा hold किया हुआ है। यह स्थिति Deadlock कहलाती है।

"A set of processes is in a deadlock state, if every process in the set is waiting for an event that can only be caused by some other process in the same set."

Deadlock is when two or more tasks never make progress(blocked process) because each is waiting for some resource held by another process. यानि एक ऐसी स्थिति जहां दो (या अधिक) process कभी आगे नहीं बढ़ पाते क्योंकि प्रत्येक किसी दूसरे द्वारा hold किये गये resource के free होने का इंतज़ार कर रहा होता है।

Deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process. (अर्थात् Blocked processes का समूह जिनमें से प्रत्येक ने कुछ resources को hold कर रखा है तथा दूसरे process द्वारा acquire किये गये resource के free होने का कभी न समाप्त होने वाला इंतज़ार कर रहा है।)

A deadlock is a situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.

- Situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.
- Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.
- Situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.

§ 6.2. अनुरोध, प्रयोग तथा अवमुक्त करना (Request, Use and Release) :

किसी सिस्टम में रिसोंसेज़ (resources) की संख्या सीमित होती है तथा यह resources कई processes के मध्य वितरित होते हैं। Resources को कई भागों में partition (विभाजित करना) किया जाता है। Resources के types के उदाहरण है—memory space, CPU cycles, files, तथा I/O devices (printers, scanners, DVD drivers इत्यादि)। एक ही प्रकार के resources को उस resource का instance कहा जाता है। उदाहरणत: यदि system में दो CPU है, (माना CPU 1 तथा CPU 2) तो कहा जायेगा कि CPU resource type के दो instance हैं—CPU 1 तथा CPU 2. इसी प्रकार यदि तीन प्रिंटर्स (PR1, PR2 तथा PR3) attached हैं, तो कहा जायेगा कि resource type printer के तीन instance है—PR1, PR2 तथा PR3.

अब यदि कोई process किसी resource type के instance की request करता है तो उस रिसोर्स टाइप के किसी भी instance का allocation करके उस request को संतुष्ट किया जा सकता है। यदि ऐसा नहीं है (अर्थात् दो instances में असमानता है) तो इसका अर्थ यह हुआ कि resource types classes को ठीक प्रकार से परिभाषित नहीं किया गया है। उदाहरणत: माना कि एक सिस्टम में दो printers हैं। अब यदि किसी process को इन दोनों में से कोई भी printer allot कर देने पर कोई फर्क नहीं पड़ता तो इनको एक ही resource class में रखा जा सकता है। किन्तु यदि एक प्रिन्टर किसी इमारत की चौथी मंजिल पर रखा है तथा दूसरा छठी मंजिल पर है तो चौथी मंजिल वाले व्यक्ति के लिये दोनो प्रिंटर्स एक से नहीं प्रतीत होंगे क्योंकि छठी मंजिल पर रखें प्रिंटर को use करने के लिये उसे छठी मंजिल पर जाना होगा।

किसी सामान्य operation में जब कोई process किसी resource को utilize करना चाहता है तो उसे निम्न चक्र से गुज़रना होता है—

Request—Request अर्थात् अनुरोध, जब भी process को किसी resource की आवश्यकता होती है तो वह उसके लिये request प्रेषित करता है। यदि वह resource उपलब्ध होता है तो तुरन्त उस process को allocate हो जाता है। किन्तु यदि request तुरन्त पूरी नहीं की जा सकती (उदाहरणत:, यदि कोई दूसरा process उस resource को use कर रहा है) तो requesting process को wait करना पड़ता है, जब तक कि वह resource उप्लब्ध नहीं हो जाता।

Use—Use अर्थात् प्रयोग, जब process को resource उप्लब्ध हो जाता है तो वह उसका आवश्यकतानुसार प्रयोग करता है (उदाहरणत: यदि resource printer है तो process को जो भी print करना होता है, print कर लेता है)।

Release—Release अर्थात् मुक्त करना। Resource का प्रयोग करने के बाद process उस resource को release कर देता है।

जब भी कोई process या thread किसी Kernel-managed resource का use करता है तो operating system यह check करता है कि process ने resource को request किया है तथा उसे resource allocate हो गया है। एक system table यह record करती है कि कोई resource free है या allocated है, तथा जब भी resource allocate होता है तो system table में यह record किया जाता है कि resource किस process को allocate हुआ है। यदि process द्वारा किसी ऐसे resource की माँग की जाती है जो currently किसी अन्य process को allocated है तो इस process को उस resource के लिये wait करने वाले processess की queue में डाल दिया जाता है।

A process must request a resource before using it and must release the resource after using it. The number of resources requested may not exceed the total number of resources available in the system.

If the request is made and the resource is available, it is allocated to the process. The process uses it and releases it. If the resource is not available, the process has to wait in a queue.

§ 6.3. डैडलॉक (Deadlock) :

Processes का कोई set deadlock अवस्था में तब कहा जाता है जब set का प्रत्येक process एक ऐसे event के होने के लिये इंतज़ार में खड़ा है जो कि उस set के किसी दूसरे process द्वारा ही किया जाना संभव है (यहाँ

event का तात्पर्य resource को अधिग्रहीत करने (aquisition) तथा मुक्त करने (release) से है)। Resource physical हो सकते हैं (उदाहरणत:, printers, tape drives, memory space तथा CPU cycles) या logical हो सकते हैं (उदाहरणत: files, semaphores तथा monitors)।

A set of processes is in a deadlock state when every process in the set is waiting for an event (e.g., resource aquisition and release) that can be caused only by another process in the set. The resources can be physical e.g., printers, tape drives, memory space, and CPU cycles or logical e.g., files, semaphores, and monitors.

डैडलॉक को समझने के लिये एक सिस्टम का उदाहरण ले सकते हैं जिसमें तीन DVD drives हैं तथा तीन processes को यह allocated है। अब यदि तीनों processes को एक अन्य DVD drive की आवश्यकता पड़ती है तो यह तीनो process deadlock अवस्था में आ जायेंगे क्योंकि इनमें से प्रत्येक दूसरी DVD drive के मुक्त होने की घटना (event) की प्रतीक्षा करता रहेगा।

इसी प्रकार एक अन्य उदाहरण लेते हैं— माना कि एक सिस्टम में एक प्रिंटर तथा एक CD-RW drive है तथा process P_1 ने प्रिंटर को होल्ड कर रखा है और process P_2 ने CD-RW drive को होल्ड कर रखा है। अब यदि CD-RW drive की request करता है तथा P_2 printer की request करता है तो deadlock की स्थिति उत्पन्न हो जायेगी।

अत: यदि कोई प्रोग्रामर Multi-threaded applications develope कर रहा है तो उसे उपरोक्त problem का विशेष ध्यान रखना चाहिये। Multi-threaded programs में deadlock उत्पन्न होने की संभावनायें रहती है क्योंकि multiple threads shared resources को प्राप्त करने हेतु आपस में compete (मुकाबला) कर सकते हैं। डैडलॉक में process का execution कभी finish नहीं हो सकता तथा सिस्टम resources tied up हो जाते हैं (अर्थात process से free नहीं हो पाते) जिससे दूसरे jobs भी start नहीं हो पाते। अगले खण्ड में हम देखेंगे कि वह कौन-कौन सी conditions हैं जो कि deadlock उत्पन्न कर देती हैं।

§ 6.4. डैडलॉक हेतु आवश्यक शर्तें (Necessary Conditions for Deadlock) :

यदि किसी सिस्टम में नीचे दी गई चार शर्तें एक साथ पूरी हो जाती हैं तो डेडलॉक की स्थिति उत्पन्न हो सकती है (Coffman's condition for deadlock)—

(i) Mutual Exclusion—कम से कम एक resource non-shareable mode में hold होना चाहिये अर्थात् इस resource का एक समय में केवल एक process द्वारा ही use किया जाना सम्भव हो। अब यदि दूसरा process इस resource की request करता है तो उसको इस resource के release (मुक्त) होने तक प्रतीक्षा करनी होगी।

(ii) Hold and Wait—Process द्वारा कम से कम किसी एक resource को hold करके रखा गया हो तथा कुछ अतिरिक्त resources के अधिग्रहण (acquisition) हेतु प्रतीक्षारत हो जो कि इस समय दूसरे processes द्वारा hold किये गये हो।

(iii) No Preemption—Resources को preempt करना सम्भव न हो अर्थात् process द्वारा hold किये गये resource को तभी स्वत: मुक्त करना संभव हो जब process द्वारा task (कार्य) पूरा कर लिया जाये अर्थात् कार्य पूर्ण होने से पहले process resource को स्वत: मुक्त करने की स्थिति में नहीं हो।

(iv) Circular Wait—Circular wait की स्थिति हो अर्थात् waiting processes का एक set (समूह) ($P_0, P_1, P_2 ..., P_{n-1}, P_n$) हो जिसमें P_0, P_1 द्वारा hold किये गये रिसोर्स को मुक्त किये जाने का इंतज़ार करता हो, P_1, P_2 द्वारा hold किये गये resource को मुक्त किये जाने का इंतज़ार करता है, $P_{n-1}; P_n$ द्वारा hold किये गये resource के मुक्त किये जाने का इंतज़ार करता है तथा P_n, P_0 द्वारा hold किये गये resources के मुक्त किये जाने का इंतज़ार करता हो। साधारण शब्दों में समझा जाये तो हम कह सकते हैं कि तीन process $A, B,$ व C हैं जिसमें A को B के द्वारा hold resource के free होने का इंतज़ार है तभी उसका कार्य पूरा होगा, B को C के द्वारा hold resource के free होने का इंतज़ार है तभी उसका कार्य पूरा होगा तथा C को A द्वारा hold resource के free होने का इंतज़ार है तभी उसका कार्य पूरा होगा। ऐसी स्थिति को circular wait स्थिति कहा जाता है।

विचार प्रश्न

चित्र 6.1 में Traffic deadlock प्रदर्शित है। इस चित्र के आधार पर बतायें कि Road के प्रत्येक section पर—

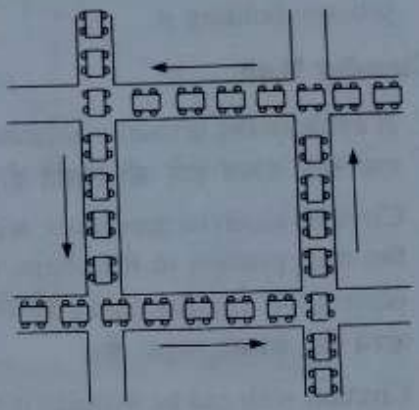
(i) Mututal exclusion की शर्त apply करती है या नहीं?

(ii) Hold-and-wait की शर्त apply करती है या नहीं?

(iii) No-preemption की शर्त apply करती है या नहीं?

(iv) Circular wait की शर्त apply करती है या नहीं?

(v) इस deadlock को break करने हेतु कोई तरीका बताइये।



चित्र 6.1—Example of a traffic deadlock

Coffman identified four conditions that must hold simultaneously for there to be a deadlock—

Mutual Exclusion

- Resources involved must be unshareable (non-shareable) (मतलब resources की साझेदारी संभव न हो); otherwise, the processes would not be prevented from using the resource when necessary.
- Resources shared such as read-only files do not lead to deadlocks (साझेदारी वाले resources से डैडलॉक उत्पन्न नहीं होता) but resources, such as printers and tape drives, requires exclusive access by a single process.
- At least one resource (thread) must be held in a non-shareable mode, that is, only one process at a time claims exclusive control of the resource (कोई न कोई ऐसा resource हो, जिसकी साझेदारी न की जा सके)।
- If another process requests that resource, the requesting process must be delayed until the resource has been released (यदि कोई process इस resource की request करे, तो उस process को इस resource के free होने का इंतज़ार करना पड़े)।

Hold and Wait

- Processes must hold the resources they have already been allocated while waiting for other (requested) resources (दूसरे resource का इंतज़ार करते समय process स्वयं को allocate किये गये रिसोर्स को hold करके रखे, अर्थात उसे मुक्त न करें)।
- If the process had to release its resources when a new resource or resources were requested, deadlock could not occur because the process would not prevent others from using resources that it controlled (free कर देने से deadlock उत्पन्न नहीं होगा)।
- In this situation processes must be prevented from holding one or more resources while simultaneously waiting for one or more others. Requesting process hold already, resources while waiting for requested resources.
- A process must exist that is holding a resource already allocated to it while waiting for additional resource that are currently being held by other processes.

No Preemption

- Processes must not have resources taken away while that resource is being used (Resource पर कब्जा जमाने के बाद process के complete होने तक resource उससे वापस न लिया जाये)।
- Otherwise, deadlock could not occur since the operating system could simply take enough resources from running processes to enable any process to complete.
- Preemption of process resource allocations can avoid the condition of deadlocks, whereever possible.

- Resources already allocated to a process cannot be preempted.
- Resources cannot be removed from the processes are used to complétion or released voluntarily by the process holding it.

Circular Wait

- A cycle in the resource allocation graph is necessary for deadlock to occur (Resource allocation ग्राफ में एक चक्र उत्पन्न होने की स्थिति हो)।
- Circular chain of processes, with each process holding resources which are currently being requested by the next process in the chain, cannot exist. If it does, the cycle theorem indicated that deadlock could occur (एक circular chain की स्थिति हो जहां पहला दूसरे का, दूसरा तीसरे का, तथा आखिरी पहले का resource फ्री करने हेतु इंतज़ार करता हो)।
- Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing (or decreasing) order.
- Processes in the system form a circular list or chain where each process in the list is waiting for a resource held by the next process in the list.

(i) **Mutual Exclusion**—At least one resource must be help in a non-sharable mode i.e. only one process at a time can use this resource. If another process requests that resource, the requesting process must wait until the resource has been released by the process already holding it.

(ii) **Hold and Wait**—A process must be holding at least one resource and waiting to acquire some additional resources that are currently being held by other processes.

(iii) **No Preemption**—Resources cannot be preempted i.e., a resource can be released only itself by the process holding it, after it has completed its task.

(iv) **Circular Wait**—A set $\{P_0, P_1, ..., P_n\}$ of waiting processes must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2, P_2 is waiting for a resource held by P_3, and similarly P_{n-1} is waiting for a resource held by P_n, and P_n is waiting for a resource help by P_0.

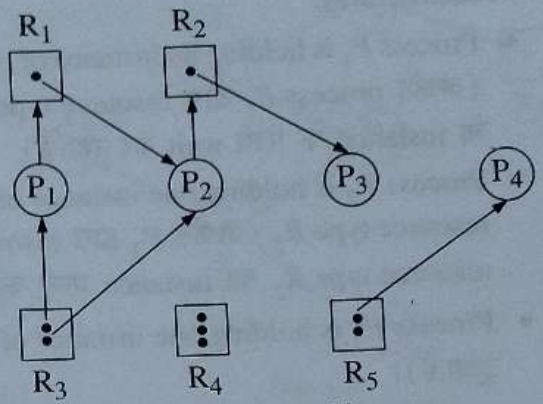
Deadlock हेतु चार शर्तें–

- Resources साझा करने की स्थिति न हो (Mutual Exclusion Condition)
- Resources को Hold करके दूसरे Resources की request की जा रही हो (Hold and Wait Condition)
- Hold किये गये Resources को मुक्त न किया जा रहा हो (No preemption)
- Waiting की chain हो जहाँ प्रत्येक process किसी दूसरे process के resources मुक्त करने का इंतज़ार करता है (Circular Wait Condition)

§ 6.5. रिसोर्स एलोकेशन ग्राफ (Resource Allocation Graph) :

डैडलॉक का वर्णन एक विशेष प्रकार के दैशिक ग्राफ (directed graphs) की सहायता से भी किया जा सकता है। यह ग्राफ System Resource Allocation Graphs कहलाते हैं। इन ग्राफ में vertices की समूह (V) तथा edges का समूह (E) होता है। vertices के समूह को दो प्रकार की नोड्स में विभाजित किया जाता है— $P = \{P_1, P_2 ..., P_{n-1}, P_n\}$ जो कि system के सभी सक्रिय processes (active processes) को व्यक्त करता है तथा $R = \{R_1, R_2, ... R_{n-1}, R_n\}$ जो कि सिस्टम में उपस्थित सभी प्रकार के resources का समूह होता है (चित्र 6.2 देखें)।

अब इस ग्राफ में यदि कोई edge process P_i से resource R_j के ओर directed है तो उसको $P_i \rightarrow R_j$ द्वारा व्यक्त किया जाता है तथा यह इस बात का संकेत है कि process P_i द्वारा resource R_j के instance की request भेजी गई है तथा वह process (P_i) उस resource (R_j) को प्राप्त करने हेतु इंतज़ार (wait) कर रहा है। इस प्रकार की directed edge (process से resource की ओर) को request edge कहा जाता है।



चित्र 6.2—रिसोर्स एलोकेशन ग्राफ

इसी प्रकार यदि कोई edge resources R_j से process P_i की ओर directed है तो उसको $R_j \rightarrow P_i$ द्वारा व्यक्त किया जाता है। यह इस बात का संकेत है कि resource R_j का कोई instance process P_i को allocate किया गया है तथा process P_i इसको use कर रहा है। इस प्रकार की directed edge (resource से process की ओर) को assignment edge कहा जाता है।

चित्र 6.2 में देखें कि प्रत्येक process को circle (वृत्त) द्वारा दर्शाया जाता है जबकि प्रत्येक resource type को rectangle (आयत) द्वारा दर्शाया जाता है। चूंकि resource के कई instances हो सकते हैं अत: इन instances को rectangle के अंदर dot द्वारा दर्शाया जाता है उदाहरण: मान लीजिये resource type "printer" के पाँच instance PR1, PR2, PR3, PR4 तथा PR5 हैं तो इन्हें rectangle के अंदर पाँच dots show करके दर्शाया जायेगा। आप ध्यान से देखें कि request edge केवल rectangle की ओर point करती है (अर्थात् rectangle के अंदर किसी dot की ओर point नहीं करती) जबकि assignment edge rectangle के अंदर किसी dot से प्रारम्भ होती है। ऐसा होना भी चाहिये क्योंकि जब कोई process किसी resource की माँग करता है तो उसे यह मतलब नहीं होता कि उसे उस resource का कौन सा instance प्राप्त होता है, अर्थात यदि वह प्रिंटर की माँग करता हैं तो उसे चाहें PR1 प्राप्त हो, PR2, या PR3, उसके लिये एक ही बात है। किन्तु allocation में यह स्पष्ट होना चाहिये कि resource type का कौन सा instance process को allot किया गया है अर्थात् process को PR1 दिया गया है, PR2 दिया गया है या फिर PR3, PR4 या PR5 दिया गया है।

जब भी कोई process P_i किसी resource type R_j के instance की request करता है तो resource allocation graph में एक request edge add कर दी जाती है (जो कि P_i से R_j की ओर इशारा करती है।) जब यह request पूरी हो जाती है तथा उस resource type का कोई instance उस process को allocate हो जाता है तो request edge assignment edge में transform हो जाती है (बदल जाती है) (जो कि R_j के किसी instance से P_i की ओर इशारा करती है तथा यह संकेत देती है कि R_j का यह instance फिलहाल P_i के कब्जे में है)। जब process अपना काम पूरा कर लेता है और उसे उस resource की आवश्यकता नहीं रह जाती, तो वह resource को free (मुक्त) या release कर देता है तथा assignment edge delete हो जाती है।

चित्र 6.2 में प्रदर्शित resource-allocation graph निम्न स्थिति को show करता है—

- P, R तथा E के समूह

  $P = \{P_1, P_2, P_3, P_4\}$

  $R = \{R_1, R_2, R_3, R_4, R_5\}$

  $E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_2, R_1 \rightarrow P_2, R_2 \rightarrow P_3, R_3 \rightarrow P_1, R_3 \rightarrow P_2, R_5 \rightarrow P_4\}$

➢ **Resource instances:**

- Resource type R_1 has one instance
- Resource type R_2 has one instance
- Resource type R_3 has two instances
- Resource type R_4 has three instances
- Resource type R_5 has two instances

➢ **Process States:**

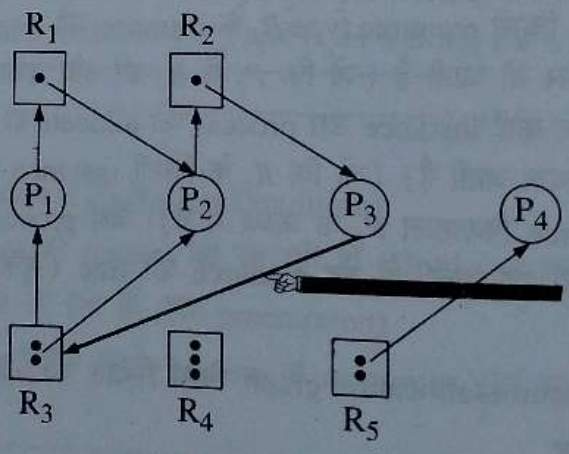
- Process P_1 is holding an instance of resource type R_3 and is waiting for an instance of resource type R_1 (अर्थात् process P_1 द्वारा resource type R_3 का एक instance hold किया हुआ है तथा वह resource type R_1 के instance के लिये wait कर रहा है)
- Process P_2 is holding one instance each of rescurce type R_1 and R_3 and is waiting for an instance of resource type R_2 (अर्थात् P_2 द्वारा resource type R_1 तथा R_3 के एक-एक instance hold किये गये हैं तथा वह resource type R_2 का instance प्राप्त करने हेतु wait कर रहा है)।
- Process P_3 is holding one instance of R_2 (Process P_2 ने resource type R_2 का एक instance hold किया हुआ है)।
- Process P_4 is holding one instance of resource type R_5 (Process P_4 ने resource type R_5 का एक instance hold किया हुआ है)।

अत:, Resource-allocation graph द्वारा system के processes, resources, edges, resource instances व process states की जानकारी प्राप्त की जा सकती है। अब बात करते हैं deadlock की। यदि resource-allocation graph में कोई cycle (चक्र) नहीं है तो इसका तात्पर्य यह है कि सिस्टम का कोई भी process deadlock अवस्था (state) में नहीं है। किन्तु यदि इस graph में कोई cycle दिखाई दे रही है, तो deadlock उत्पन्न होने की स्थिति बन सकती है।

यदि प्रत्येक resource type का केवल एक-एक instance उपलब्ध है, तो cycle होने पर deadlock की स्थिति उत्पन्न हो जायेगी यदि cycle में resource types का एक ही समूह है तथा उनमें से प्रत्येक का एक ही instance है, तो deadlock की स्थिति उत्पन्न हो जायेगी तथा cycle में involved प्रत्येक process deadlocked हो जायेगा। ऐसी स्थिति में graph में cycle का होना deadlock उत्पन्न होने की आवश्यक व पर्याप्त शर्त (necessary and sufficient condition) है।

यदि प्रत्येक resource के कई instances हैं तो cycle होने पर भी यह ज़रूरी नहीं है कि deadlock की स्थिति उत्पन्न हो जायेगी। ऐसी स्थिति में graph में cycle का होना deadlock की आवश्यक किन्तु पर्याप्त शर्त नहीं है (Necessary but not a sufficient condition for existence of deadlock)।

आइये, इस concept को समझने के लिये चित्र 6.2 में प्रदर्शित resource-allocation graph को एक बार फिर consider करते हैं माना कि process P_3 resource type R_3 के instance की request करता है। किन्तु R_3 का कोई भी instance available नहीं है, तो graph में एक request edge $P_3 \rightarrow R_3$ add कर दी जाती है (चित्र 6.3 देखें)। आप देखें कि system में दो cycles exist करती है—



P_3 द्वारा resource type R_3 की request भेजने के कारण यह request edge add की गई है

चित्र 6.3—डैडलॉक युक्त रिसोर्स एलोकेशन ग्राफ

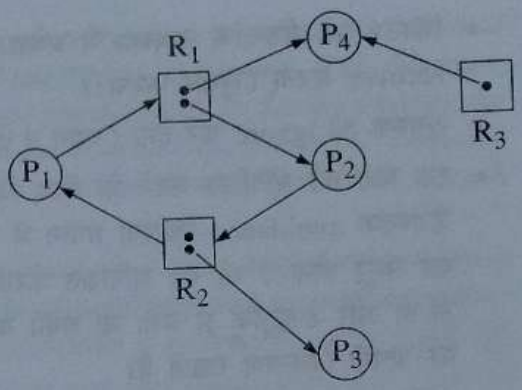
(i) $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_2 \rightarrow P_3 \rightarrow R_3 \rightarrow P_1$

(ii) $P_2 \rightarrow R_2 \rightarrow P_3 \rightarrow R_3 \rightarrow P_2$

इससे स्पष्ट है कि—

अत:, process P_1, P_2 तथा P_3 डैडलॉक अवस्था में हैं। P_2; R_2 के लिए wait कर रहा है जो कि P_3 के पास है। P_3; R_3 के लिए wait कर रहा है जो कि P_1 तथा P_2 के पास है। P_1; R_1 के लिए wait कर रहा है जो कि P_2 के पास है।

अब चित्र 6.4 में प्रदर्शित resource allocation ग्राफ को देखें। इसमें भी cycle $P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_2 \rightarrow P_1$ exist करती है किंतु यहाँ डैडलॉक स्थिति नहीं है क्योंकि यदि P_3; R_2 को release कर देगा तो R_2; P_2 को प्राप्त हो जायेगा तथा cycle break हो जायेगी।



चित्र 6.4

अत:, संक्षेप में कहा जाये तो उपरोक्त वर्णन से निम्न बातें स्पष्ट होती हैं—

(i) यदि resource-allocation graph में cycle नहीं है तो deadlock की स्थिति नहीं है

(ii) यदि resource-allocation graph cycle exist करती है तथा प्रत्येक resource type का केवल एक-एक instance ही उपलब्ध है, तो डैडलॉक निश्चित रूप से exist करता है।

(iii) यदि resource-allocation ग्राफ में cycle exist करती है किन्तु प्रत्येक resource टाइप के एक से अधिक instance हैं, तो deadlock की स्थिति हो भी सकती है या नहीं भी, अर्थात् हो सकता है सिस्टम deadlock स्थिति में हो, या न भी हो।

A system resource-allocation graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two types of nodes: $P = \{P_1, P_2, P_3 ..., P_{n-1}, P_n\}$, the set consisting of all the active processes in the system, and $R = \{R_1, R_2, R_3 ..., R_{n-1}, R_m\}$, the set consisting of all resource types in the system.

A directed edge from process P_i to resource type R_j; $P_i \rightarrow R_j$ (called request edge) signifies that process P_i has requested an instance of resource type R_j and is currently waiting for that resource. A directed edge from resource type R_j to process P_i; $R_j \rightarrow P_i$ (called assignment edge) signifies that an instance of resource type R_j has been allocated to process P_i.

Each process P_i is represented as a circle and each resource type R_j as a rectangle. Since resource type R_j may have more than one instance, each such instance is represented as a dot within the rectangle. A request edge points to only the rectangle R_j, whereas an assignment edge must also clarify the instance allocated by one of the dots in the rectangle.

When process P_i requests an instance of resource type R_j, a request edge is inserted in the graph and when this request is fulfilled, the request edge is instantaneously transformed to an assignment edge. When the process no longer needs access to the resource, it releases the resource; and hence the assignment edge is deleted.

Now to check whether there is a deadlock, we check if there exists any cycle in the graph. If the graph contains no cycles, then no process in the system is deadlocked. If the graph does contain a cycle, then a deadlock may exist.

If each resource type has only one instance, then a cycle implies that a deadlock has occurred. If the cycle involves only a set of resource types, each of which has only a single instance, then a deadlock has occurred. Each process involved in the cycle is deadlocked. In this situation, a cycle in the graph is both a necessary and a sufficient condition for the existence of deadlock.

If each resource type has many instances, then a cycle dose not necessarily means that a deadlock has occurred. Hence, for this situation, a cycle in the graph is a necessary but not a sufficient condition for the existence of deadlock.

§ 6.6. डैडलॉक को हैंडल करने की विधियाँ (Methods for Handling Deadlock) :

Deadlock की समस्या से निबटने की विधियाँ निम्नवत् हैं—

- Deadlock को avoid या prevent (बचाव) हेतु protocols (नियम, शिष्टाचार) का पालन ताकि सिस्टम कभी डैडलॉक अवस्था में प्रवेश ना करे।

- सिस्टम को डैडलॉक अवस्था में प्रवेश करने की अनुमति देना, उसको detect करना (पता लगाना) तथा उसको recover करना (सुधार करना)।
- समस्या को ignore कर देना (ध्यान न देना) और ऐसे व्यवहार करना जैसे कि डैडलॉक कभी सिस्टम में होगें ही नहीं।
- इस बात को सुनिश्चित करने के लिए कि डैडलॉक की स्थिति सिस्टम में उत्पन्न ना हो, डैडलॉक prevention या डैडलॉक avoidance विधियां प्रयोग में लायी जाती है। डैडलॉक prevention या डैडलॉक से बचाव, उन विधियों का समूह होता है जो यह सुनिश्चित करती है कि डैडलॉक की आवश्यक शर्तों में से कम से कम एक स्थिति उत्पन्न ना हो और डेडलॉक से बचा जा सके। यह विधियाँ डैडलॉक से बचाव हेतु processes द्वारा की जाने वाली रिक्वेस्ट पर अपनी नियन्त्रण रखती है।
- Deadlock avoidance विधियाँ (अर्थात् सिस्टम को डैडलॉक से दूर रखने वाली विधियाँ) यह सुनिश्चित करती हैं कि operating system को कुछ advance additional information (अग्रिम अतिरिक्त सूचनायें) प्रदान की जायें जिससे यह पता चल सके कि कोई process अपनी lifetime के दौरान कौन-कौन से resources को request व use करेगा। इस अतिरिक्त सूचना के आधार पर ऑपरेटिंग सिस्टम निर्णय ले सकेगा कि किसी request के लिये process को wait करना चाहिये या नहीं। यह निर्णय लेने के लिये कि current request को satisfy किया जाना है या delay (देरी) किया जाना है, system को प्रत्येक process द्वारा currently available resources (मौजूदा समय में उपलब्ध resources), Resources currently allocated to each process (मौजूदा समय में विभिन्न process को आवंटित resources) तथा future request and release (भविष्य में प्राप्त होने वाली request तथा release संबधी जानकारी) आदि सभी सूचनायें consider (विचार करना) की जाती हैं।
- यदि कोई सिस्टम न तो deadlock prevention algorithm का use करता है तथा न ही deadlock avoidance scheme का use करता है तो deadlock उत्पन्न हो सकता है। ऐसी स्थिति में सिस्टम एक ऐसी algorithm का use कर सकता है जो सिस्टम की अवस्था का निरीक्षण करके यह जाँच करती रहे कि system में deadlock तो उत्पन्न नहीं हो गया है तथा यदि deadlock उत्पन्न हो गया हो तो एक ऐसी algorithm का use करें जो कि उसको deadlock अवस्था से बाहर ला सकें (Deadlock Recovery Algorithm)।
- यदि system न तो deadlock avoidance व prevention हेतु कोई scheme use करता है न ही deadlock detection व recovery की कोई व्यवस्था करता है (अर्थात् न तो deadlock से दूर रहने व बचाव का प्रयास करना है न ही यह चैक करने हेतु कोई व्यवस्था रखना है कि डैडलॉक उत्पन्न तो नहीं हुआ है तथा उसके होने की स्थिति में उसे दूर करने हेतु कोई मैकेनिज़्म की व्यवस्था रखता है) तो ऐसे में एक ऐसी स्थिति उत्पन्न हो सकती है कि system deadlock अवस्था में पहुँच जाये तथा यह भी पता न लगाया जा सके कि यह क्यों व कैसे उत्पन्न हुआ है। यह स्थिति (अर्थात् undetected deadlock की स्थिति) सिस्टम के perfomance (निष्पादन) को deteriorate (खराब) कर सकती है क्योंकि resources उन processes द्वारा hold हो जाते हैं जो run नहीं कर पाते और उसके बाद दूसरे processes भी उन resources को request करने पर deadlock अवस्था में प्रवेश कर जाते हैं। अंततः, system कार्य करना बंद कर देता है (stop functioning) तथा ऐसे में उसे फिर से manually start (manually restart) करने के अलावा कोई चारा (विकल्प) शेष नहीं रह जाता।

हालांकि deadlock समस्या से निबटने हेतु यह कोई viable approach (व्यवहार्य तरीका) नहीं है किन्तु फिर भी कई operating systems इसी को use करते हैं। कई systems में deadlock बहुत कम होते हैं, ऐसे में यह विधि deadlock avoidance, prevention, detection व recovery से कम खर्चीली सिद्ध होती है।

Deadlock prevention and deadlock avoidance schemes make sure that a situation of deadlock does not arise in a system. Whereas deadlock prevention provides a set of methods for ensuring that at least one of the necessary conditions cannot-hold by limiting, how requests for resources can be made. We discuss these methods in section.

The deadlock avoidance schemes require that the operating system by given in advance additional information concerning which resources a process will request and use during its lifetime. With this prior knowledge, it can decide for each request whether or not the process should wait. To decide whether the current request can be satisfied or must be delayed, the system must consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process.

If a system does not uses either a deadlock-prevention or a deadlock-voidance algorithm, then a deadlock situation may arise. If this happens, the system can provide an algorithm that examines the state of the system to determine whether a deadlock has occurred and an algorithm to recover from the deadlock.

If a system neither makes sure that a deadlock will never occur nor provides mechanisms for deadlock detection and recovery, then there are chances that the system enters a deadlocked state yet has no way of recognizing what has happened. Resulting in deterioration of the system's performance. Hence, the system will stop functioning and will have to be restarted manually.

§ 6.7. डैडलॉक से बचाव (Deadlock Prevention) :

हमने देखा कि deadlock उत्पन्न होने की चार शर्तें हैं—Mutual exclusion, Hold and wait, No preemption, तथा Circular wait। आइये एक बार फिर से उपरोक्त स्थितियों पर विस्तार से चर्चा करते हैं—

Mutual Exclusion—Non-shareable resources (अर्थात् वह resources जो share न किये जा सकें) हेतु mutual exclusion की शर्त लागू होती है। उदाहरणत: कोई प्रिंटर एक साथ कई processes द्वारा share नहीं किया जा सकता। Shareable resources (अर्थात् वह resources जिनको share किया जा सकता है) पर mutual exclusion की शर्त लागू नहीं होती, इसलिये यह deadlock में involve (शामिल) नहीं हो सकते। Shareable resources का उदाहरण है—read only files, तथा यदि एक से अधिक process किसी read only file पर पहुँच बनाने का प्रयास करते हैं, तो उनको एक साथ उस file को access (पहुँच बनाना) प्रदान की जा सकती है। किसी shareable resource के लिये process को कभी प्रतीक्षा करने की आवश्यकता नहीं पड़ती। किन्तु mutual exclusion condition को समाप्त नहीं किया जा सकता क्योंकि कई resources unshareable होते हैं।

Hold and Wait—Hold and wait का तात्पर्य है कि process ने एक resource hold कर रखा है तथा दूसरे को प्राप्त करने हेतु उसने request send की हुई है तथा request पूरी होने का इंतज़ार कर रहा है। अत: यह सुनिश्चित करने के लिये कि सिस्टम में hold and wait स्थिति उत्पन्न न हो, इस बात को सुनिश्चित करना चाहिये कि जब कोई process किसी resource हेतु request करे, उस समय उसके द्वारा कोई दूसरा resource hold न किया गया हो। एक protocol यह use की जा सकती है कि process द्वारा प्रारम्भ होते समय ही सभी resource request कर लिये जायें तथा process को वह resources allocate कर दिये जायें। अत:, इसको implement करने हेतु resource request करने हेतु system calls अन्य system calls से पहले प्राप्त हो जानी चाहिये। एक वैकल्पिक protocol यह use की जा सकती है कि कोई process तभी किसी resource हेतु request भेज सके जबकि वह कोई अन्य resource hold न कर रहा हो। Process कुछ resources की request करे और उनको use करे तथा अब यदि उसे कोई नयी resource requert भेजनी हो तो उससे पहले वह allocated सभी resources को release कर दें तथा उनको release करने के पश्चात् ही उसे नयी request भेजने का अधिकार दिया जाये।

उपरोक्त दोनों protocols में अंतर समझने हेतु एक उदाहरण लेते हैं। मान लीजिये एक process को DVD drive से disk पर एक file में data copy करना है, file को sort करना है तथा results को print करना है। पहली protocol के अनुसार उसको प्रारम्भ में ही DVD drive, disk file तथा printer की request करनी होगी जबकि दूसरी protocol के अनुसार वह प्रारम्भ में DVD drive व disk file की request भेज सकता है तथा इतना काम होने पर वह DVD drive व disk file को release करने के बाद वह फिर disk drive व printer की request भेज सकता है तथा disk file से file को printer को print कराकर वह अपना कार्य समाप्त कर सकता है।

उपरोक्त दोनों protocols में पहला दोष यह है कि resource utilization low हो जायेगा क्योंकि process को allocated resources काफी समय तक unused रहेंगे (जैसे कि प्रथम प्रोटोकॉल में आप देखे कि उसे प्रिंटर की आवश्यकता तो last में होगी किन्तु वह पूरे execution के दौरान प्रिंटर को hold करके रखेगा)। ऐसे में, यदि यह संभव हो कि प्रिंटर को बाकी execution के बाद request किया जाये तो printer का unused time कम हो जायेगा। इसी प्रकार यदि यह सुनिश्चित हो जाये कि disk file को release करने के बाद उसे पुन: प्राप्त करने पर disk file का data सुरक्षित रहेगा, तभी disk file को release किया जा सकता है।

दूसरा दोष यह है कि starvation की संभावना बनती है। यदि किसी process को कई resources चाहिये तो उसको काफी समय तक wait करना पड़ सकता है, क्योंकि ऐसा हो सकता है कि उसके द्वारा request किये गये resources में से कम से कम एक (या अधिक) resource सदैव किसी अन्य process को ही allocated रहें।

No Preemption—

Non preemption का अर्थ है कि यदि process को CPU allocate कर दिया जाता है तो process उस CPU को तब तक अपने पास रखता है जब तक कि process terminate न हो जाये या फिर waiting state में न चला जाये। Deadlock की तीसरी आवश्यक शर्त यही है कि यदि allot किये गये resources का preemption सम्भव न हो, तो deadlock उत्पन्न हो सकता है। अत: deadlock से बचने के लिये यह protocol निर्धारित की जा सकती है कि यदि process ने किसी resource को hold करके रखा हो तथा वह किसी दूसरे resource की request करे जिसे कि तत्काल (immediately) allocate करना संभव न हो (अर्थात् process को wait करना पड़े) तो उस process द्वारा hold किये गये मौजूदा resources को भी preempt कर दिया जाये (अर्थात् वापस ले लिया जाये) तथा यह preempted resources भी process की request list (list of resources for which the process is waiting अर्थात् resources की list जिसके लिये process wait कर रहा हो) में add कर दिये जायें। अत: अब process restart (फिर से start) तभी हो सकेगा जब अपने पुराने resources (जो उसके पास पहले से थे तथा जिनको preempt किया गया) तथा नये resources (जिसकी उसने request भेजी थी) एक साथ प्राप्त करेगा।

एक दूसरा विकल्प यह है कि जब कोई process किन्हीं resources की request भेजता है तो यह चैक किया जाये कि वह resources उपलब्ध हैं या नहीं। यदि हाँ, तो उन्हें allocate कर दिया जाये। यदि नहीं, तो यह चैक किया जाये कि वह किसी ऐसे process को तो allocated नहीं हैं जो कि किसी अन्य process से कुछ अतिरिक्त resources प्राप्त करने हेतु wait कर रहा हो। यदि हाँ, तो उस waiting process से यह resources preempt करके requesting process को दिये जा सकते हैं। यदि resources ऐसे process के पास है, जो कि wait अवस्था में नहीं है, तो फिर requesting process थोड़ा इंतज़ार कर सकता है। इस waiting period में उसके held resourecs को preempt किया जा सकता है, यदि किसी दूसरे process को इनकी जरूरत है। अत:, अब यह process restart तभी हो सकेगा जब इसको requesting resources (जिसके लिये यह request कर रहा था) तथा preempted resources (जो कि waiting period में इससे preempt कर लिये गये थे) दोनों एक साथ इसे प्राप्त हो जाये।

यह protocol ऐसे resources पर लागू की जा सकती है जिनका state आसानी से save किया जा सके तथा बाद में फिर से restore (पुन: प्राप्त) किया जा सके (जैसे कि CPU registers या memory space)। प्रिंटर्स या टेप ड्राइव्स पर उपरोक्त protocol लागू नहीं किया जा सकता।

सर्कुलर वेट (Circular Wait)—

Circular wait यानि एक ऐसा चक्र जहां पहला, दूसरे का इंतज़ार करता हो, दूसरा, तीसरे का तथा तीसरा, पहले का, यानि कि कभी न खत्म होने वाना इंतज़ार; तथा deadlock की चौथी शर्त भी यही है, circular wait। Circular wait की स्थिति न आने पाये, इसका एक उपाय यह है कि सभी resources types की ordering (एक क्रम निर्धारित कर देना) कर दी जाये तथा प्रत्येक process इसी बढ़ते हुये क्रम में resources की माँग (request) करें।

उदाहरण के तौर पर, मान लीजिये कि $R = \{R_1, R_2, R_3, ..., R_n\}$ resource type का set (समूह) है। इसमें से प्रत्येक resource type को एक संख्या आवंटित कर दी जाये जिससे यह पता चल जाये कि उनका क्रम क्या है। अत:, एक one-to-one function (फलन) परिभाषित कर लिया जाये (अर्थात् $F : R \to N$) यहाँ N कोई प्राकृतिक संख्या (natural number) है, तथा जिसे उस resource की संख्या कहा जा सकता है।

उदाहरण:

$$F(\text{DVD}) = 2$$
$$F(\text{disc drive}) = 6$$
$$F(\text{printer}) = 11 \text{ इत्यादि।}$$

अब deadlock से बचाव हेतु protocol निर्धारित किया जा सकता है, जिसके अनुसार प्रत्येक process केवल resource क्रम संख्या के बढ़ते क्रम के अनुसार ही resource की माँग कर सकता है, अर्थात् कोई process प्रारम्भ में resource type R_i के कितने भी instances की request भेज सकता है। उसके पश्चात् process resource type, R_j के instances की request तभी भेज सकेगा यदि $F(R_j) > F(R_i)$ अर्थात् यदि R_j से जुड़ी क्रम संख्या का मान R_i से जुड़ी क्रम संख्या से अधिक है। अत:, यदि एक resource के कई instances की आवश्यकता है तो process को उसकी single request issue करनी होगी। जैसे कि उपरोक्त प्रदर्शित क्रम संख्याओं को लें तो यदि process को DVD व disc drive की एक साथ आवश्यकता है तो पहले DVD की माँग करनी होगी तथा उसके बाद disc drive की (क्योंकि DVD की क्रम संख्या 2 है व disc drive की 6 है)। अर्थात् इसका उल्टा संभव नहीं हो पायेगा, यदि वह पहले disc drive की request कर देता है (जिससे जुड़ी क्रम संख्या 6 है) तो वह उससे पहले वाली क्रम संख्या वाले किसी भी resource के लिये request नहीं कर पायेगा।

एक अन्य विकल्प यह भी संभव है कि कोई process किसी resource R_j के instance के लिये तभी request कर सकता है यदि उसने इससे ऊपर की क्रम संख्या वाले सभी resources को release कर दिया हो $F(R_i) \geq F(R_j)$।

यदि इन दोनों प्रोटोकॉल्स को use किया जाये तो circular wait condition उत्पन्न नहीं होगी।

(i) *The ME (mutual exclusion) condition must hold for non-sharable resources. Sharable resources, do not require mutually exclusive access and thus cannot be involved in a deadlock. A process never needs to wait for a sharable resource. Actually, we cannot prevent deadlocks by denying the mutual-exclusion condition, because some resources are always non-sharable.*

(ii) *To make sure that the hold-and-wait condition never occurs it must be taken care of that, whenever a process requests a resource, it does not hold any other resources (a) each process to request and be allocated all its resources before it begins execution or (b) process request resources only when it has none. A process may request some resources and use them. It must release all the resources that is it currently allocated, to make any fresh request.*

The disadvantage of above is poor resource utilization since resources may be allocated but unused for a long period.

Also, starvation is possible because process that needs several popular resources may have to wait indefinitely, because at least one of the resources that it needs may always be allocated to some other process for its execution.

(iii) *To ensure preemption, if a process is holding some resources and requests another resource that cannot be immediately allocated to it then all resources currently being held are preempted and these resources are added to the list of resources for which the process is waiting.*

Another option is that if a process requests some resources, we first check whether they are available. If yes, we allocate them. If no we check whether they are allocated to some other process that is waiting for additional resources. If yes, we preempt the desired resources from the waiting process and allocate them to the requesting process. If not the requesting process must wait.

(iv) The last condition for deadlocks is the circular-wait condition. To break this vicious circle, is to impose a total numbering of all resource types and to require that each process requests resources in an increasing order of numbering.

A process can initially request any number of instances of a resource type— R_x. After that, the process can request instances of resource type R_x if and only if $F(R_x) > F(R_y)$. If several instances of the same resource type are needed, a single request for all of them must be issued. Another option is that a process request an instance of R_y it has released any resources R_x such that $F(R_x) \geq F(R_y)$.

If these two protocols are used, then the circular-wait condition cannot hold. We can demonstrate this fact by assuming that a circular wait exists (proof by contradiction). Let the set of processes involved in the circular wait be.

§ 6.8. डैडलॉक से दूर रहने की विधियाँ (Deadlock Avoidance) :

पिछले खण्ड में हमने देखा कि deadlock से बचाव (deadlock prevention) हेतु requests करने पर कोई न कोई प्रतिबंध लगाकर यह प्रयास किया जाता है कि deadlock की चार शर्तों में से कोई एक पूरी न होने दी जाये ताकि deadlock न हो पाये। इन विधियों का मुख्य दोष यह है कि इससे device utilization तथा system througput पर प्रतिकूल प्रभाव पड़ता है।

Deadlock को avoid करने का एक अन्य वैकल्पिक तरीका यह है कि resources की requests से संबंधित कुछ अतिरिक्त सूचनायें (additional information about how resources are requested), उदाहरणत: यदि किसी सिस्टम में एक DVD तथा एक प्रिंटर है तथा यदि system के पास यह सूचना है कि process P_1 पहले DVD तथा फिर प्रिंटर की request करेगा तथा process P_2 पहले प्रिंटर तथा फिर DVD की request करेगा तो इस request व release के क्रम की जानकारी होने पर system प्रत्येक request पर यह decide कर सकता है कि process को waiting में रखना चाहिये या नहीं तथा इस प्रकार भविष्य में किसी deadlock होने की स्थिति से बचा जा सकता है। अत:, किसी भी request से संबंधित निर्णय (decision) लेते समय system मौजूदा उपलब्ध resources (currently available resources), मौजुदा आवंटित resources (currently allocated resources) तथा प्रत्येक process द्वारा भविष्य में होने वाली requests व releases (the future requests and releases for each process) पर विचार करके ही निर्णय लेगा।

इस संबंध में प्रयुक्त विभिन्न algorithms में अतिरिक्त सूचना की मात्रा एवं प्रकार में भिन्नता रहती है (differ in amount and type of additional information required)। एक algorithm model के अनुसार, प्रत्येक process यह घोषणा करे कि उसे अधिकतम प्रत्येक प्रकार के कितने resources की आवश्यकता होगी। यदि यह सूचना पूर्व में ही उपलब्ध हो जायेगी, तो फिर एक ऐसी algorithm design की जा सकती है जो यह सुनिश्चित कर देगी कि system कभी deadlock अवस्था में नही पहुँचेगा। यह algorithm deadlock avoidance algorithm कहलाती है। अत: हम कह सकते हैं कि deadlock avoidance algorithm निरंतर resource allocation अवस्था का निरीक्षण करती है ताकि यह सुनिश्चित हो जाये कि circular wait स्थिति कभी उत्पन्न नहीं होगी। Resource allocation state को define करने हेतु उपलब्ध व allocated resources तथा processes की अधिकतम demand पर विचार किया जाता है।

Deadlock prevention schemes prevent deadlocks by limiting the conditions how requests can be made. The conditions ensure that at least one of the necessary conditions for deadlock cannot occur. However, this leads to low device utilization and reduced system throughput.

Deadlock avoidance schemes require prior additional information about how resources are to be requested. The complete sequence of requests and releases for each process, the system can decide for each request whether or not the process should wait in order to avoid a possible future deadlock. Each request requires that in making this decision the system consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process.

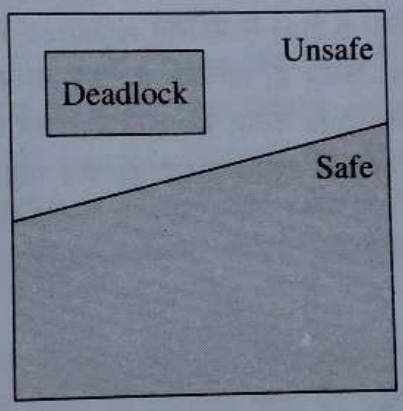
सुरक्षित अवस्था एल्गोरिद्म (Safe State Algorithm)—

किसी state को safe तभी कहा जा सकता है यदि सिस्टम प्रत्येक process को (अपने अधिकतम तक) resources allocated कर सके और इसके बाद भी deadlock को avoid कर सकें।

"A state is safe if the system is able to allocate resources to each of the process (up to its maximum) in some order and can still avoid a deadlock.

एक सिस्टम safe state में तभी कहा जा सकता है जब एक safe sequence exist करता हो। Process का एक क्रम $\langle P_1, P_2, P_3 \ldots P_{n-1}, P_n \rangle$ वर्तमान allocation state के लिये safe sequence होगा यदि प्रत्येक P_i द्वारा की गयी resource request को वर्तमान उपलब्ध resources तथा सभी P_j द्वारा hold की गई resources (जहां $j < i$) द्वारा मिलकर पूरा किया जा सकता हो। इस स्थिति में यदि P_i की आवश्यकतानुसार सभी resiurces उपलब्ध नहीं भी है तो भी P_i wait कर सकता है जब तक सभी P_j अपना कार्य पूरा न कर लें। जब यह सभी P_j अपना कार्य पूरा कर लेंगे तो P_i को उसकी जरूरत के अनुसार सभी resources उपलब्ध हो जायेंगे तथा वह भी अपना कार्य पूरा कर सकेगा तथा सभी resources को release करके terminate हो सकेगा। जब P_i terminate हो जायेगा तो P_{i+1} को उसकी आवश्यकतानुसार resource प्राप्त हो जायेंगे तथा वह भी अपना कार्य पूरा कर लेगा तथा यह क्रम चलता रहेगा। यदि इस प्रकार का क्रम exist नहीं करता तो system state का unsafe (असुरक्षित) कहा जाता है।

हालांकि सभी unsafe states का deadlock होना जरूरी नहीं है किन्तु unsafe states, deadlock की ओर जा सकते हैं। जब तक state safe है, operating system, unsafe व deadlock states से बचा रहता है।



चित्र 6.5—Safe, unsafe, तथा deadlock state spaces

A state is said to be safe if the system can allocate resources to each process (up to its maximum) in some order and still can avoid a deadlock. In other words, a system is in a safe state only if there exists a safe sequence. A sequence of processes $\langle P_1, P_2, P_3, \ldots, P_{n-1}, P_n \rangle$ *is said to be a safe sequence for the current allocation state if, for each* P_i, *the resource requests that* P_i *can still make can be satisfied by the current available resources and the resources held by all* P_j, *with* $j < i$. *Hence, if the resources that* P_i *needs are not immediately available, then* P_i *can wait until all* P_j *have finished its work. Then* P_i *can obtain its needed resources, complete its task, release its resources, and terminate. When* P_i *terminates,* P_{i+1} *can carry on. If no such sequence exists, then the system state is said to be unsafe.*

A safe state is not a deadlocked state and a deadlocked state is an unsafe state. But, not all unsafe states are deadlocks. It may be possible that an unsafe state may lead to a deadlock. As long as the state is safe, the operating system can avoid deadlock. In an unsafe state, the operating system cannot prevent processes form requesting resources such that a deadlock occurs.

रिसोर्स-एलोकेशन-ग्राफ एल्गोरिद्म (Resource Allocation Graph Algorithm)—

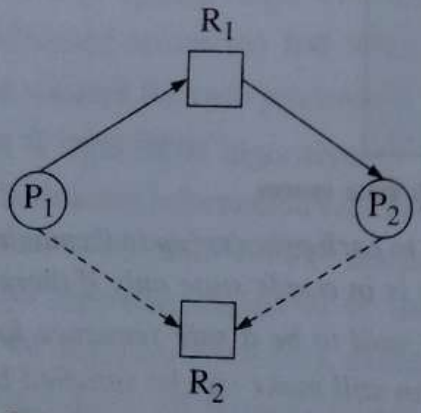
यदि एक ऐसा resource allocation system उपलब्ध है जिसमें प्रत्येक resource type का केवल एक instance है तो resource allocation graph मे थोड़ा सा रूपान्तरण (variant) करके उसे deadlock avoidance के लिये use किया

जा सकता है। इसके लिये resource allocation graph में request तथा assignment edges (जिनको पहले ही discuss किया जा चुका है) के साथ-साथ एक नयी edge जिसको claim edge कहा जाता है, use की जाती है। यहाँ claim edge, $P_i \rightarrow R_j$ यह व्यक्त करती है कि process P_i भविष्य में किसी समय resource R_j के लिये request कर सकता है। इस edge की दिशा request edge की तरह ही दिखाई जाती है किन्तु इसको solid line के बजाय dashed line से दर्शाया जाता है। जब process P_i resource R_j हेतु request करता है तो claim edge, request edge में convert हो जाती है। इसी प्रकार, यदि process P_i, resource R_j को release कर देता है तो assignment edge पुन: claim edge में convert हो जाती है। ध्यान दें कि resources का claim process के प्रारम्भ होने से पहले ही प्राप्त हो जाना चाहिये अर्थात् जब process P_i execute होना शुरू करें, तो सभी claim edges पहले से ही resource allocation graph में मौजूद रहनी चाहिये। यदि बाद में claim edge add करने की अनुमति तभी दी जानी चाहियें, यदि इस process से associated सभी edges, claim edges हैं, अन्यथा नयी claim edge जोड़ने की अनुमति नहीं दी जानी चाहिये।

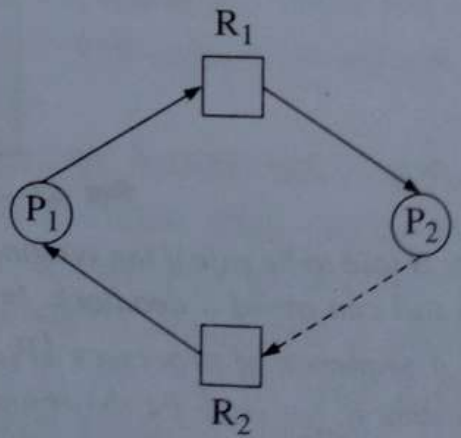
मान लीजिये कि कोई Process P_i, Resource R_j हेतु request करता है। यह request तभी प्रदान की जा सकती है यदि request edge $P_i \rightarrow R_j$ को assignment edge $R_j \rightarrow P_i$ में convert करने पर resource-allocation graph में कोई cycle (चक्र) न बनती हो। इसके लिये cycle detection algorithm use की जा सकती है। ध्यान रखें कि यदि graph में n processes है, तो graph में cycle detection algorithm के लिये n^2 operation की आवश्यकता होती है।

यदि कोई cycle exist नहीं करती तो resource का allocation system को safe state में रखेगा। किन्तु cycle होने पर यह allocation system को unsafe state में ले जायेगा। अत: P_i का अपनी request पूरी होने के लिये wait करना पड़ेगा।

उपरोक्त algorithm को समझने हेतु चित्र 6.6(a) में प्रदर्शित resource-allocation graph को consider करते हैं। माना कि process P_1 resource R_2 की request करता है, हालांकि R_2 इस समय free है, पर फिर भी इसको P_1 को allocate नहीं किया जा सकता क्योंकि इससे ग्राफ में cycle उत्पन्न हो जाती है, चित्र 6.6 (b) देखें। इस प्रकार की cycle का उत्पन्न होना system के unsafe state की ओर जाने का इशारा करता है। अब यदि P_1 भी R_2 के लिये request करने लगेगा, तो deadlock उत्पन्न हो जायेगा।



चित्र 6.6(a)—Deadlock avoidance हेतु resource allocation graph



चित्र 6.6(b)—Resource allocation graph में unsafe state

If we have a resource-allocation system with only one instance of each resource type, to analyse deadlock avoidance, a new edge, called a claim edge is added. A claim edge $P_i \rightarrow R_j$ indicates that process P_i may request resource R_j at some time in the future. This edge is similar to a request edge in direction but is represented by a dashed line. When process P_i requests resource R_j, the claim edge $P_i \rightarrow R_j$ is converted to a request edge. In the same way, when a resource R_j is released by P_i, the assignment edge $R_j \rightarrow P_i$ is reconverted to a claim edge $P_i \rightarrow R_j$ again before process P_i starts executing, all its claim edges must already appear in the resource-allocation graph. Alternately, a claim edge $P_i \rightarrow R_j$ to be added to the graph only if all the edges associated with process P_i are claim edges.

Now how to avoid a deadlock? Suppose that process P_i requests resource R_j. The request can be granted only if converting this request edge $P_i \rightarrow R_j$ to an assignment edge $R_j \rightarrow P_i$ does not result in the formation of a cycle in the resource-allocation graph. An algorithm for detecting a cycle. In a graph containing 10 processes requires an order of n operations.

If no cycle exists, then the allocation of the resource will put the system in a safe state. If a cycle is found, then the allocation will drive the system in an unsafe state. Hence, process P_i will have to wait for its requests to be satisfied.

§ 6.9. बैंकर्स एल्गोरिद्म (Banker's Algorithm) :

हमने पिछले खण्ड़ में देखा कि resource-allocation algorithm केवल तभी मान्य है यदि प्रत्येक resource का एक ही instance उप्लब्ध हो। किन्तु यदि resources के multiple instances है (एक से अधिक instances) तो ऐसे resource-allocation system हेतु resource-allocation graph algorithm मान्य नहीं है। अत:, ऐसी स्थितियों में जिस algorithm का use किया जा सकता है, उसको Banker's algorithm कहा जाता है। Banker's algorithm हालांकि resource-allocation graph algorithm से कम efficient (दक्ष) होती है, उन system के लिये भी use की जा सकती है जिसमें resources के multiple instances होते हैं। प्रश्न यह उठता है कि आखिर इस algorithm को Banker's algorithm क्यों कहा जाता है। इसका नाम Banker's algorithm इसलिये है क्योंकि इसको banking system में यह सुनिश्चित करने के लिये लागू किया जा सकता है कि बैंक अपना कैश कभी भी इस तरह से allocate न कर दें जिससे वह अपने सभी customers (उपभोक्ताओं) की आवश्यकताओं को satify न कर सकें।

जब कोई process system में enter करता है तो उसे यह declare करना होता है कि उसे प्रत्येक resource के अधिकतम कितने instances की आवश्यकता होगी। यह संख्या सिस्टम के कुल उपलब्ध resources की संख्या से अधिक नहीं होनी चाहिये। जब user resources के किसी समूह की माँग (request) करता है, तो system यह पता लगाता है कि इस माँग को पूरा करने (अर्थात् उन resources को allocate करने) पर system safe state में रहेगा अथवा नहीं। यदि हां, तो resource allocate कर दिये जाते हैं, यदि नहीं, तो process को wait करना होता है जब तक कि दूसरे process पर्याप्त resources को release न कर दें।

Banker's Algorithm को implement (कार्यान्वित) करने हेतु कई data structures maintain करना होता है। यह data structures, resource-allocation graph के state को encode करते हैं। माना कि systems में processes की संख्या n है तथा resources की संख्या m है। ऐसी स्थिति में निम्न data structures की आवश्यकता होगी—

- **Available**—Available अर्थात् उपलब्ध। एक m लम्बाई का वैक्टर प्रत्येक type के उपलब्ध resources को indicate करता है। यदि Available $[j] = k$ तो इसका तात्पर्य है कि j टाइप के k instances उपलब्ध है। उदाहरण के लिये यदि Available [5] = 3 तो इसका तात्पर्य यह है कि Resources R_5 (यहाँ R_5 किसी भी resource की क्रम संख्या है जो कि system द्वारा दी गई है) के तीन instances उप्लब्ध है अर्थात् माना कि R_5 का अर्थ किसी सिस्टम में DVD है तो Available [5] = 3 का अर्थ हुआ कि system में मौजूदा 3 DVD उपलब्ध हैं।
- **Max**—Max अर्थात् maximum या अधिकतम। एक $n \times m$ matrix प्रत्येक process की अधिकतम demand को परिभाषित करती है।

  यदि Max $[i][j] = k$ तो इसका तात्पर्य है कि process p_i, resource type R_j को अधिकतम k instance की demand करेगा उदाहरणत: यदि Max [2][5] = 3 का तात्पर्य है कि process P_2, resource type R_5 के अधिकतम तीन instance की demand करेगा।
- **Allocation**—Allocation अर्थात् आवंटन, निर्धारण या विनिहित करना। एक $n \times m$ matrix यह बताती है कि currently प्रत्येक process को प्रत्येक resource के कितने instances allocated है। यदि allocation $[i][j] = k$

तो इसका अर्थ यह है कि process P_i को इस समय Resource R_j के k instance allocated हैं। उदाहरणत: यदि Allocation [2][5] = 1 तो इसका तात्पर्य यह है कि process P_2 इस समय resource type R_5 का इस समय एक instance allocated है।

- **Need**—Need अर्थात् आवश्यकता या जरूरत। एक $n \times m$ matrix यह बताती है कि प्रत्येक process की अवशेष resource आवश्यकतायें (remaining resource need) क्या हैं अर्थात् प्रत्येक process को प्रत्येक resource type के अभी और कितने instances की आवश्यकता है। यदि Need $[i][j] = k$ तो इसका तात्पर्य है कि process P_i को अपना कार्य पूरा करने हेतु resource type R_j के अभी k instances की आवश्यकता अवशेष है। उदाहरणत: यदि Need [2][5] = 2 तो इसका तात्पर्य है कि process P_2 को अपना कार्य पूरा करने के लिये resource R_5 के 2 instances की आवश्यकता और पड़ेगी। यहाँ नोट करने की बात यह है कि Need $[i][j]$ = Max $[i][j]$ – Allocation $[i][j]$ अर्थात् यदि Max [2][5] = 3, Allocation [2][5] = 1 तो Need [2][5] = 3 – 1 = 2 अर्थात् यदि process P_2 को resource type R_5 के अधिकतम 3 instance की आवश्यकता है तथा उसे वर्तमान में 1 instance allocated है तो उसे इस कार्य को पूर्ण करने के लिये अभी इस resource type के दो instance की आवश्यकता अवशेष (बाकी) है।

इन डाटा स्ट्रक्चर में समय के साथ size तथा value का परिवर्तन होता रहता है। Banker's algorithm को समझने से पूर्व हम कुछ notations (संकेत या चिन्हान्कन) समझने का प्रयास करते हैं। माना कि X तथा Y, n लम्बाई के वैक्टर है। यदि i के सभी मानों अर्थात् $i = 1, 2, 3, \ldots n-1, n$ के लिये $X[i] \leq Y[i]$ तभी हम कह सकते हैं कि $X \leq Y$ अर्थात् $X \leq Y$ तभी कहा जा सकता है, जब i के सभी मानों के लिये $X[i]$ का मान $Y[i]$ से less than या equal to हो। उदाहरणत: यदि $X = (2, 5, 4, 9)$ तथा $Y = (1, 4, 2, 3,)$ तो $Y < X$ इसी प्रकार यदि $X = (3, 5, 4, 8)$ तथा $Y = (5, 6, 8, 9)$ तो $X < Y$ तथा यदि $X = (2, 3, 5, 9)$ तथा $X = (1, 2, 6, 8)$ तो $X \neq Y$।

अब यदि matrices, Allocation तथा Need की प्रत्येक row को vector treat किया जाये तथा उन्हे Allocation$_i$ तथा Need$_i$ कहा जाये तो Vector allocation$_i$, process P_i को मौजूदा आवंटित रिसोंसेज (currently allocated resources) को व्यक्त करता है जबकि vector Need$_i$ उन अतिरिक्त resources को व्यक्त करता है जिनकी P_i का अपना कार्य पूरा करने हेतु आवश्यकता अवशेष है।

When a new process enters the system, it must declare the maximum number of instances of each resource type that it will need. However, this number may not exceed the total number of resources in the system. When a user requests a set of resources, the system must determine whether the allocation of these resources will leave the system in a safe state. If yes, the resources are allocated; if no, the process must wait until some other process releases enough resources.

Let n be the number of processes in the system and m be the number of resource types. The data structures which must be maintained to implement the banker's algorithm.

- ***Available***—*A vector of length m indicates the number of available resources of each type. If Available [j] = k, there are k instances of resource type R_j available.*
- ***Max***—*An n × m matrix defines the maximum demand of each process. If Max [i][j] = k, then process P_i may request at most k instances of resource type R_j.*
- ***Allocation***—*An n × m matrix defines the number of resources of each type currently allocated to each process. If Allocation [i][j] = k, then process P_i is currently allocated k instances of resource type R_j.*
- ***Need***—*An n × m matrix indicates the remaining resource need of each process. If Need [i][j] = k, then process P_i may need k more instances of resource type R_j to complete its task Need [i][j] = Max [i][j] – Allocation [i][j].*

सेफटी एल्गोरिद्म (Safety Algorithm)—

यह ज्ञात करने के लिये कि सिस्टम safe state में है अथवा नहीं, निम्न एल्गोरिद्म का use किया जा सकता है—

(i) माना की Work तथा Finish क्रमश: m तथा n लम्बाई के vector है।

(ii) माना कि प्रारम्भ में Work = Available
तथा Finish [i] = False (असत्य) ($i = 0, 1, 2, \ldots n-1$) के लिए

(iii) i का वह मान ज्ञात करें, जिसके लिये निम्न दोनों कथन सत्य हो—
(a) Finish [i] = False
(b) $Need_i \leq Work$
यदि i का ऐसा कोई मान नहीं है, तो पद संख्या 5 पर जायें

(iv) Work = Work + Allocation
Finish [i] = true
पद 3 पर जायें

(v) यदि Finish [i] = = true (सभी i के लिये) तो सिस्टम सुरक्षित अवस्था में है (यहाँ दो = = चिन्ह equal की condition चैक करने हेतु हैं, अर्थात् यदि Finish [i] true के तुल्य है तो)।

यह ज्ञात करने के लिये कि कोई state safe है या नहीं, उपरोक्त algorithm में $m \times n^2$ operations की आवश्यकता होती है—

(i) Let *Work* and *Finish* be vectors of length m and n, respectively.

(ii) *Work* = *Available* and *Finish* [i] = *false* for $i = 0, 1, \ldots, n-1$.

(iii) Find a value of i such that both these hold.
(a) *Finish* [i] = = false
(b) *Need* $i \leq$ *Work*

(iv) If no such i exists, go to step (v)
Finish [i] = *true*
Go to step (iii)

(v) If *Finish* [i] = = *true* for all i, then it can be said that the system is in a safe state.

This algorithm will require of $m \times n^2$ operations to determine if a state is safe.

रिसोर्स रिक्वैस्ट एल्गोरिद्म (Resource Request Algorithm)—

यह ज्ञात करने के लिये कि रिक्वैस्ट को सुरक्षित रूप से grant (प्रदान) किया जा सकता है या नहीं, निम्न एल्गोरिद्म को use किया जा सकता है—

माना कि Process P_i का request vector request i है। यदि Request i [j] = = k, तो Process P_i को Resource R_j के k instances की आवश्यकता है। जब Process P_i द्वारा resources की request भेजी जाती है तो निम्न actions perform किये जाते हैं—

(i) यदि $Request_i \leq Need_i$, तो पद 2 पर जायें अन्यथा error condition उत्पन्न करें क्योंकि process अपने maximum claim से अधिक की माँग कर रहा है।

(ii) यदि $Request_i \leq Available$ तो पद 3 पर जायें। नहीं तो P_i को wait करना होगा क्योंकि resources अभी उपलब्ध नहीं हैं।

(iii) यह मानते हुये कि Process P_i को requested resources states निम्नवत् modify करते हुये allocate कर दिये गये हैं—

$$Available = Available - Request_i$$
$$Allocation_i = Allocation_i + Request_i$$
$$Need_i = Need_i - Request_i$$

यदि उपरोक्त परिणामी (Resultant) resource allocation state safe है तो transaction complete कर लिया जाता है तथा Process P_i को resource allocate कर दिये जाते हैं। यदि नया (परिणामी) state unsafe है तो P_i $Request_i$ की wait करता है तथा पुराना resource allocation ही बना रहने दिया जाता है।

Let $Request_i$ be the request vector for process P_i. If $Request_i[j] == k$, then this means that process P_i needs k instances of resource type R_j. When a request for resources is made by process P_i, the actions are taken are:

(i) If $Request_i \leq Need_i$, go to step (ii). In other case, generate an error condition, as the process has illegally exceeded its maximum claim.

(ii) If $Request_i \leq Available$, go to step (iii). Otherwise, P_i must wait, as the resources are not available.

(iii) Have the system assume to have allocated the requested resources to process P_i by modifying the state as given below:

$$Available = Available - Request_i;$$
$$Allocation_i = Allocation_i + Request_i;$$
$$Need_i = Need_i - Request_i;$$

If the resulting state is safe, the transaction is allowed and process P_i is allocated its resources. In other case, if the new state is unsafe, then P_i must keep waiting for $Request_i$, and the previous resource–allocation state is restored back.

बैंकर्स एल्गोरिद्म Edsger Dijkstra द्वारा विकसित resource–allocation व Deadlock avoidance algorithm है जो कि system की safety को चैक करने हेतु use की जाती है। इसके लिये यह algorithm resources के अधिकतम संभव आवंटन को simulate करती है तथा S-state check करके deadlock की possible स्थिति को चैक करती है जिसके आधार पर यह निर्णय लिया जा सकता है कि Resource का आवंटन किया जाना चाहिये या नहीं। इसके लिये विभिन्न data structures जैसे कि Available, Max, Allocation तथा Need का use किया जाता है।

Program in C for Banker's algorithm:

```
include<stdio.h>
void main()
{
int max[25], allocation[25], need[25], seq[25], available, work, i,
j=0, k, m, n, e;
printf("Please enter number of processes.....");
scanf("%d",&n);
printf("Please enter maximum allocated resources to the process.....");
for(i=0;i<n;i++)
{
printf("maximum of p %d is",i);
scanf("%d",&max[i]);
printf("Allocation of p %d is",i);
scanf("%d",&allocation[i]);
need[i]=max[i]-allocation[i];
printf("need of p %d is : %d",i,need[i]);
}
```

```
printf("Enter available resources");
scanf("%d",&available);
printf("Max allocation need \n");
for(i=0;i<n;i++)
printf("%d %d %d",max[i],allocation[i],need[i]);
m=k=n;
e=m+1;
while(m>0)
{
if(k==e)
m=0 ;
else
{
k=e;
e=0;
for(i=0;i<n;i++)
if((need[i]>0)&&(need[i]<=available))
{
need[i]=0;
work=allocation[i]+available;
available=work;
seq[j]=i;
m--;
j++;
}
else
e++;
}
}
if(k==e)
printf(" unsafe sequence");
else
{
printf("safe sequence");
printf("The sequence of process is :");
for(i=0;i<n;i++)
printf("p %d\t",seq[i]);
}
}
```

§ 6.10. डैडलॉक डिटैक्शन (Deadlock Detection) :

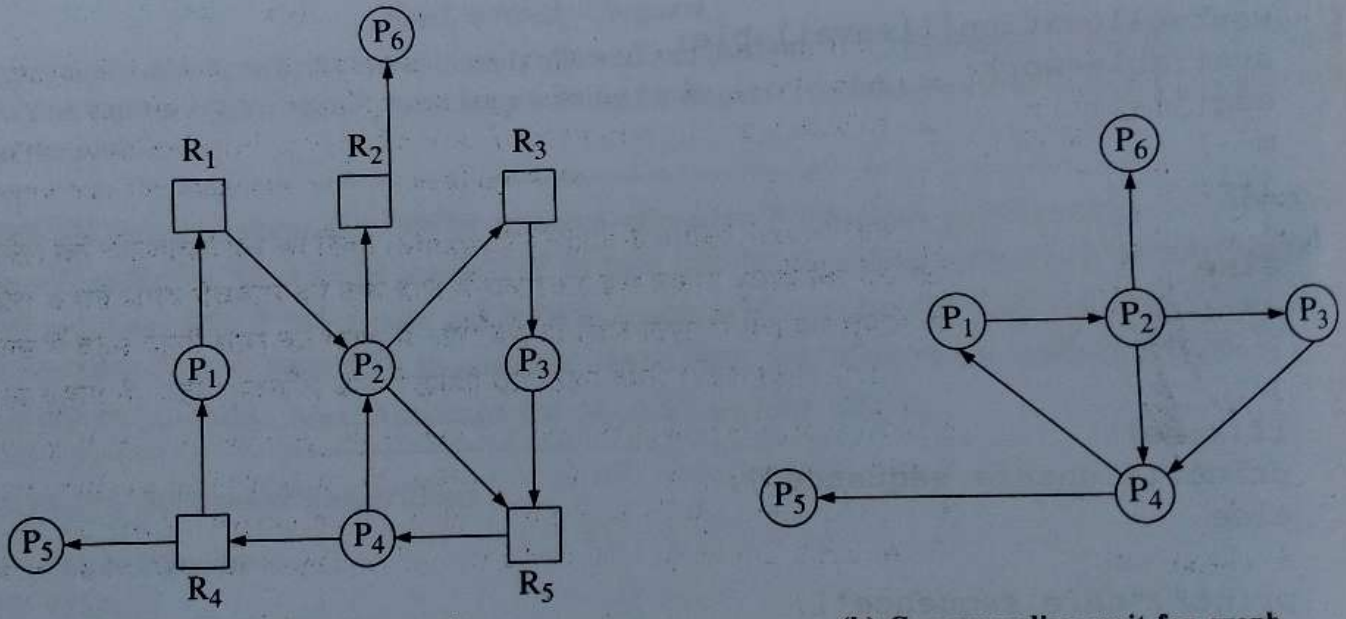
यदि सिस्टम deadlock prevention या deadlock detection avoidance algorithm use नहीं करता, तो deadlock स्थिति उत्पन्न हो सकनी है। ऐसी स्थिति में सिस्टम के पास एक ऐसी algorithm होनी चाहिये जो system के स्टेट का निरीक्षण (examination) करके यह चैक करे कि कही डैडलॉक तो उत्पन्न नहीं हो गया, और यदि हाँ, तो एक ऐसी algorithm जो डैडलॉक से रिकवरी प्रदान कर सके।

If a system does not use either a deadlock-prevention or a deadlock-avoidance algorithm, then the system must provide an algorithm to examines the state of the system to determine whether a deadlock has occurred, and if yes, then an algorithm to recover from the deadlock.

However, a detection and recovery scheme requires overhead that includes not only the run-time costs of maintaining the information and executing the deadlock detection algorithm, but also the potential losses encountered in recovering from a deadlock.

यदि प्रत्येक resource का single instance है (If each resource type has a single instance): यदि प्रत्येक resources का single instances है तो deadlock detection हेतु algorithm wait-for graph का प्रयोग करके प्राप्त की जा सकती है। Wait-for graph resource-allocation graph का ही मिलता जुलता रूप (variant) है। यदि resource allocation graph (चित्र 6.7(a)) की सभी resource nodes हटा दी जायें तथा संगत edges को collapse (समाप्त) कर दिया जाये तो wait-for-graph (चित्र 6.7(b)) प्राप्त होता है।

Wait-for-graph में यदि कोई edge P_i से P_j की ओर directed है तो इसका अर्थ यह होता है कि process P_i; process P_j द्वारा किसी resource के release किये जाने की प्रतीक्षा कर रहा है जिसकी P_i को आवश्यकता है। Wait-for graph में $P_i \rightarrow P_j$ edge केवल तभी हो सकती है यदि इसके संगत resource allocation graph में दो edges $P_i \rightarrow R_x$ तथा $R_x \rightarrow P_j$ है (जहाँ R_x कोई resource है)(चित्र 6.7 (b) देखें)।



(a) Resource allocation graph **(b) Corresponding wait-for graph**

चित्र 6.7

यहाँ भी deadlock तभी उत्पन्न होगा यदि wait-for graph में कोई cycle exist करती है। Deadlocks की detection हेतु system को wait-for graph maintain करना होगा तथा कुछ अंतराल बाद (periodically) यह check करने हेतु algorithm invoke (पुकारना) करते रहना होगा जो कि graph में cycle को search करती रहे। यदि graph में n vertices हैं, तो इस graph में cycles की search हेतु इस algorithm को n^2 operations की आवश्यकता पड़ेगी।

If all resources have only a single instance, then we can use a deadlock detection algorithm using wait-for graph. By removing the resource nodes and collapsing the appropriate edges of the resource-allocation graph.

A deadlock exists in the system if and only if the wait-for graph contains a cycle. In such a case, an algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph for example, if the number of vertices are 9, 81 operations are to be performed.

यदि resource type के कई instances हो (deadlock delectation algorithm for multiple instances of a resource type): यदि प्रत्येक resource type के कई instances उपलब्ध हैं, तो wait-for-graph स्कीम लागू नहीं हो पाता। ऐसी स्थिति में प्रयुक्त deadlock detection algorithm कई time varying data structures का use करती है (Banker's algorithm की भांति)

- Available
- Allocation
- Request

उपरोक्त डाटा स्ट्रक्चर को consider करते हुए multiple instances वाले resource types हेतु deadlock detection algorithm निम्नवत् है—

(i) माना कि Work तथा Finish क्रमश: m व n लम्बाई के vectors है

(ii) प्रारम्भ में Work = Available.

यदि i के सभी मानों के लिये $\text{Allocation}_i \neq 0$,

तो Finish $[i]$ = false

अन्यथा Finish $[i]$ = true

(iii) Index i का काई ऐसा मान ज्ञात करें कि निम्न दोनो true हो—

Finish $[i]$ = = false

$\text{Request}_i \leq \text{Work}$

यदि i का ऐसा कोई मान नहीं है तो पद 5 पर जायें।

(iv) $\text{Work} = \text{Work} + \text{Allocation}_i$

Finish $[i]$ = true

पद 3 पर जायें।

(v) यदि Finish $[i]$ = = false (किसी i के लिये $(0 \leq i \leq n)$) तो सिस्टम deadlock state में है। यदि Finish $[i]$ = = false, तो process i deadlock में है।

उपरोक्त algorithm में deadlock को detect करने हेतु $m \times n^2$ ऑपरेशन्स की आवश्यकता होती है।

(i) Let *Work* and *Finish* be vectors of length m and n, respectively.

(ii) Initialize $Work = Available$. For $i = 0, 1, \ldots n-1$, if $Allocation_i \neq 0$, then $Finish\,[i] = false$; otherwise, $Finish\,[i] = true$.

(iii) Find an index i such that both

(a) $Finish\,[i] = = false$

(b) $Request_i \leq Work$

(iv) If no such i exists, go to step (v)

$Work = Work + Allocation_i$

$Finish[i] = true$

Go to step (ii).

(v) If $Finish[i] = = false$, for some i, $0 \leq i < n$, then the system is in a deadlock state. Additionally if $Finish\,[i] = = false$, then process P_i is in a deadlock state.

§ 6.11. डैडलॉक रिकवरी (Deadlock Recovery) :

यदि detection algorithm से यह पता चलता है कि सिस्टम deadlock अवस्था में है तो उससे बाहर आने हेतु को विकल्प होते हैं जैसे कि operator को सूचित किया जाये तथा वह manually इससे deal करे, या फिर circular wait brea की जाये, या कुछ resources को preempt कर दिया जाये।

- Process termination अर्थात् प्रौसेस को abort (लक्ष्य पूरा होने से पहले समाप्त करना) कर देना, अत: सभ deadlocked processes को abort करके या फिर एक-एक करके processes को abort करके, जब तक कि cycle eliminate नहीं हो जाती, Deadlock से recovery प्राप्त की जा सकती है।
- Preemption को use करके deadlock से recovery प्राप्त करने हेतु victim का selection roll back (preempte process को फिर से कब शुरू करना है) starvation न हो, इत्यादि का ध्यान रखा जाना चाहिये।

कुछ महत्वपूर्ण प्रश्नोत्तर

प्रश्न–1. डैडलॉक को परिभाषित करें?

उत्तर—डैडलॉक blocked processes का समूह है जिसमें प्रत्येक प्रौसेस कुछ रिसोर्सेज़ को hold करके दूसरे प्रौसेसेज़ द्वारा होल्ड किये गये रिसोर्सेज़ के free होने का कभी खत्म न होने वाला इंतज़ार कर रहा होता है।

प्रश्न–2. डैडलॉक की आवश्यक शर्तें बतायें।

उत्तर—Conditions for deadlock : **Four conditions** must hold simultaneously for there to be a deadlock.

- **Mutual Exclusion**—At least one resource (thread) must be held in a non-shareable mode i.e. it could not be shared between the two (साझेदारी संभव न हो)
- **Hold and Wait**—Requesting process hold already, resources while waiting for requested resources. There must exist a process that is holding a resource already allocated to it while waiting for additional resource that are currently being held by other processes (Wait करते समय allocated resource को hold करना)
- **No Preemption**—Resources already allocated to a process cannot be preempted (made free) (काम समाप्त हुये बिना रिसोर्स को फ्री न करना).
- **Circular Wait**—The processes in the system form a circular list or chain where each process in the list is waiting for a resource held by the next process in the list (चक्र की स्थिति)

प्रश्न-3. Kernel I/O subsystem की विभिन्न सेवाओं का उल्लेख कीजिये।

उत्तर—कर्नल I/O सबसिस्टम—किसी computer के I/O (अर्थात् इनपुट-आउटपुट) से संबंधित मूल तत्व हैं—Buses, Device controllers तथा Devices. डेटा को devices तथा main memory के मध्य move करने का कार्य CPU द्वारा programmed I/O के रूप में किया जाता है या फिर DMA controller के द्वारा। डिवाइस को कन्ट्रोल करने वाला Kernel module device driver कहलाता है।

Kernel द्वारा I/O से संबंधित कई विशेष सेवायें प्रदान की जाती हैं जैसे कि Scheduling, Buffering, Caching, Spooling, Device reservation, Error handling.

- **Scheduling**—I/O request की scheduling अर्थात् इनको execute करने का क्रम निर्धारित करना। एक अच्छी scheduling system की overall performance को बेहतर बनाती है, विभिन्न processes के मध्य device की sharing सही तरीके से करने में मदद करती है तथा average waiting time को कम करती है।
- **Buffering**—Buffer एक memory area होता है जो कि डेटा को उस समय स्टोर करता है जब वह दो डिवाइसेज़ के मध्य या फिर किसी device तथा application के मध्य ट्रांसफर किया जाता है। बफरिंग करने के कई कारण हैं—

(i) डेटा स्ट्रीम को उत्पन्न (produce) करने वाले तथा उपभोग (consume) करने वाले के मध्य स्पीड मिसमैच की समस्या का निदान करना (ii) दो विभिन्न data transfer size वाली युक्तियों के मध्य सामंजस्य स्थापित करना (iii) Application I/O हेतु copy semantics को support करना इत्यादि।

Caching (कैशिंग)—Cache एक fast memory क्षेत्र होता है जो डेटा की copies को hold करता है। Cached copy को access करना origina copy को access करने की तुलना में अधिक efficient होता है। Cache व Buffer में अंतर यह होता है कि Buffer केवल data item की exi ting copy store करता है जबकि cache उस item की अन्य copy faster storage पर hold करता है।

Spooling and Device Reservation (स्पूलिंग तथा डिवाइस रिजर्वेशन)—स्पूल एक प्रकार का बफर होता है जो कि ऐसे डिवाइस हेतु output को hold करता है जो कि interleaved (मिले जुले) data streams को स्वीकार नहीं कर सकता (जैसे कि प्रिन्टर)। हालांकि प्रिन्टर केवल एक समय में एक जॉब को सर्व कर सकता है लेकिन यदि कई जॉब्स प्रिन्टर से अपनी फाइल को एक साथ प्रिन्ट कराना चाहते हैं तो ऑपरेटिंग सिस्टम इस समस्या को पूलिंग द्वारा साल्व करता है।

Error Handling and I/O Protection—Operating system protected memory का use करके hardware तथा application errors से सुरक्षा प्रदान करता है।

प्रश्नावली

1. (a) डैडलॉक से आप क्या समझते हैं?
   (b) डैडलॉक हेतु चार शर्तों का उल्लेख कीजिये।
   (c) डैडलॉक को Handle करने की विधियाँ बताइये।
   (d) निम्न को समझाइये—
      (i) Deadlock Prevention
      (ii) Deadlock Avoidance
      (iii) Banker's Algorithm
      (iv) Deadlock Detection and Recovery
2. Resource–allocation graph क्या होता है?
3. Hold and Wait तथा No preemption में अंतर समझाइये।
4. निम्न में से Deadlock avoidance algorithm है—
   (a) Round Robin algorithm
   (b) Elevator's algorithm
   (c) Banker's algorithm
   (d) Weaver's algorithm
5. किसी system को safe state में कब कहा जा सकता है?

7 Chapter

मैमोरीज़ (MEMORIES)

THINK ABOUT IT

The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore, all progress depends on the unreasonable man.

— George Bernard Shaw

Great minds discuss ideas. Average minds discuss events. Small minds discuss people.

— Eleanor Roosevelt

Truly creative people care a little about what they have done, and a lot about what they are doing. Their driving focus is the life force that surges in them now.

— Alan Cohen

§ 7.1. परिचय (Introduction) :

डिजिटल सिस्टम्स का एनालॉग सिस्टम्स की तुलना में एक मुख्य लाभ यह है कि डिजिटल सिस्टम्स में डिजिटल सूचनाओं को आसानी से स्टोर किया जा सकता है। डिजिटल सिस्टम्स की इसी मैमोरी क्षमता के कारण डिजिटल सिस्टम्स अनेक अनुप्रयोगों में लाभदायक सिद्ध होते हैं।

डिजिटल मैमोरी में हज़ारों registers होते हैं तथा प्रत्येक रजिस्टर एक बाइनरी वर्ड (binary word is a collection of bits) को स्टोर कर सकता है। नयी जनरेशन के कम्प्यूटर्स कोर मैमोरी के स्थान पर सेमी कन्डक्टर मैमोरी प्रयोग करते हैं क्योंकि यह सस्ती व कार्य करने में आसान होती है।

"The memory of a computer is where the program and data are stored before the calculations begin. The memory is therefore one of the most active parts of the computer. The memory is equivalent to thousands of register, each storing a binary word."

§ 7.2. मैमोरी सम्बन्धी शब्दावली (Memory Terminology) :

मैमोरी सैल (Memory Cell)—वह युक्ति या वैद्युत परिपथ जो एक बिट (0 या 1) (bit means binary digit 0 or 1) को स्टोर कर सकता है, मैमोरी सैल कहलाता है। मैमोरी सैल के उदाहरण हैं—फ्लिप फ्लॉप, आवेशित संधारित्र, मैगनेटिक टेप पर एक single spot इत्यादि।

"A device or electrical circuit used to store a single bit is called a memory cell for example a flip flop, a charged capacitor and a single spot on a magnetic tape or disk."

मैमोरी वर्ड (Memory Word)—बिट्स का समूह जो कि किसी instruction या डाटा को व्यक्त करता है, मैमोरी वर्ड कहलाता है। उदाहरणत: यदि किसी रजिस्टर में आठ फ्लिप फ्लॉप हैं, तो वह 8-बिट वर्ड को स्टोर करता है। सामान्यत:, कम्प्यूटर्स में वर्ड साइज़ 4 से 64 बिट की range में होता है।

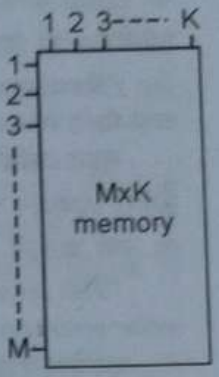
"A group of bits in a memory that represents instruction or data of same type is called a memory word."

बाइट (byte)—8-बिट का समूह बाइट कहलाता है (जबकि 4-bits को nibble कहा जाता है)। वर्ड साइज को बाइट में भी व्यक्त किया जा सकता है उदाहरणत: 16 बिट के वर्ड साइज को दो-बाइट कहा जा सकता है इत्यादि।

"A group of 8-bits is termed as byte."

मैमोरी क्षमता या डैन्सिटी (Memory Capacity or Density)—मैमोरी की क्षमता अर्थात् मैमोरी में कितनी बिट्स स्टोर की जा सकती हैं। मान लीजिये कि किसी मैमोरी में 8-बिट वाले 512 वर्ड स्टोर किये जा सकते हैं तो उस मैमोरी की क्षमता 512×8 बिट्स लिखी जाती है। यहाँ पहला अंक मैमोरी में स्टोर किये जाने वाले words की संख्या तथा दूसरा अंक word size (अर्थात् प्रत्येक वर्ड में बिट्स की संख्या) को दर्शाता है। बड़ी साइज की मैमोरी 'किलो' या 'मैगा' में व्यक्त की जाती है जहाँ $1\text{ k} = 2^{10} = 1024$, तथा $1\ M = 2^{20} = 1,048,576$। अत: यदि मैमोरी का साइज 2048×8 है तो उसे $2\text{ k} \times 8$ लिखा जा सकता है।

चित्र (7.1) में प्रदर्शित मैमोरी में $M \times K$ क्षमता वाली मैमोरी का ब्लॉक डायग्राम प्रदर्शित है।



चित्र 7.1—M × K क्षमता की मैमोरी

"By memory capacity, we mean that how can we specify the number of bits that can be stored in the memory."

एड्रैस (Adderss)—मैमोरी में स्टोर किये गये प्रत्येक word का एक unique address होता है। एड्रैस को बाइनरी में व्यक्त किया जाता है जबकि एड्रैस साइज़ बड़ा होने पर ऑक्टल या हैक्साडैसीमल में भी व्यक्त किया जाता है। चित्र 7.2 में देखें कि 8 वर्ड स्टोर करने हेतु प्रत्येक लोकेशन का 3 बिट address होगा। स्पष्ट है कि यदि मैमोरी में 2^n वर्ड स्टोर किये जा सकते हैं, तो प्रत्येक address n बिट का होगा।

Address	Words
0 0 0	Word 0
0 0 1	Word 1
0 1 0	Word 2
0 1 1	Word 3
1 0 0	Word 4
1 0 1	Word 5
1 1 0	Word 6
1 1 1	Word 7

चित्र 7.2—प्रत्येक वर्ड लोकेशन का Unique address

यदि मैमोरी में स्टोर्ड words की संख्या M है, तो address bits की संख्या n वह smallest number है, जिससे $2^n \geq M$। एड्रैस बिट्स की संख्या के बराबर ही address बस की चौड़ाई होती है। तालिका 7.1 से एड्रैस बस का साइज ज्ञात किया जा सकता है (ध्यान रखें कि K = Kilo, M = Mega, G = Giga, T = Terra) उदाहरणत: यदि मैमोरी का साइज़ $4K \times 8$ है तो मैमोरी में stored words की संख्या $4K$ होगी। चूँकि $4K = 4 \times 1024 = 2^{12}$ अत: address बस की चौड़ाई 12 होगी अर्थात् 12 एड्रैस लाइन्स होगी।

तालिका 7.1—मैमोरी का आकार ज्ञात होने पर ऐड्रैस बस की चौड़ाई ज्ञात करने के लिये तालिका

$2^1 = 2$	$2^{11} = 2\text{ K}$	$2^{21} = 2\text{ M}$	$2^{31} = 2\text{ G}$
$2^2 = 4$	$2^{12} = 4\text{ K}$	$2^{22} = 4\text{ M}$	$2^{32} = 4\text{ G}$
$2^3 = 8$	$2^{13} = 8\text{ K}$	$2^{23} = 8\text{ M}$	$2^{33} = 8\text{ G}$
$2^4 = 16$	$2^{14} = 16\text{ K}$	$2^{24} = 16\text{ M}$	$2^{34} = 16\text{ G}$
$2^5 = 32$	$2^{15} = 32\text{ K}$	$2^{25} = 32\text{ M}$	$2^{35} = 32\text{ G}$
$2^6 = 64$	$2^{16} = 64\text{ K}$	$2^{26} = 64\text{ M}$	$2^{36} = 64\text{ G}$
$2^7 = 128$	$2^{17} = 128\text{ K}$	$2^{27} = 128\text{ M}$	$2^{37} = 128\text{ G}$
$2^8 = 256$	$2^{18} = 256\text{ K}$	$2^{28} = 256\text{ M}$	$2^{38} = 256\text{ G}$
$2^9 = 512$	$2^{19} = 512\text{ K}$	$2^{29} = 512\text{ M}$	$2^{39} = 512\text{ G}$
$2^{10} = 1024$	$2^{20} = 1024\text{ K}$	$2^{30} = 1024\text{ M}$	$2^{40} = 1024\text{ G}$
$1\text{ K} = 2^{10}$	$1\text{ M} = 1024\text{ K}$	$1\text{ G} = 2^{10}\text{ M}$	$1\text{ T} = 2^{10}\text{ G}$
$= 1024$	$= 2^{10}\text{ K}$	$= 1024\text{ M}$	$= 1024\text{ G}$

"Address is a number that identifies the location of word in the memory. Each word stored in the memory has a unique binary address."

रीड ऑपरेशन (Read Operation)—वह ऑपरेशन जिसके द्वारा किसी मैमोरी लोकेशन पर स्टोर किये गये बाइनरी वर्ड को sense करके किसी अन्य डिवाइस को ट्रांसफर किया जाता है। मैमोरी रीड ऑपरेशन कहलाता है। रीड ऑपरेशन को fetch ऑपरेशन भी कहा जाता है।

"Read operation is the operation by which the binary word stored in a specific memory location is sensed and then transferred to another device."

राइट ऑपरेशन (Write Operation)—मैमोरी की किसी लोकेशन पर नये बाइनरी वर्ड को प्लेस करना अर्थात् किसी मैमोरी लोकेशन पर बाइनरी वर्ड को स्टोर करना राइट ऑपरेशन कहलाता है। उल्लेखनीय है कि जब भी किसी मैमोरी लोकेशन पर नया बाइनरी वर्ड राइट किया जाता है, तो वह उस लोकेशन पर पहले से स्टोर्ड बाइनरी वर्ड को replace कर देता है।

"The operation where a new word is placed or stored into a particular memory location is called a write operation."

एक्सेस टाइम (Access Time)—यह मैमोरी की ऑपरेटिंग स्पीड का माप होता है। मैमोरी के रीड ऑपरेशन में लगने वाला समय उसका एक्सेस टाइम t_{acc} कहलाता है। अर्थात् मैमोरी को एड्रैस इनपुट प्राप्त होने तथा मैमोरी आउटपुट पर डाटा प्राप्त होने के मध्य लगने वाला समय मैमोरी का एक्सेस टाइम कहलाता है।

"It is a measure of operating speed of the memory. It is amount of time required to perform a read operation. It is the time between memory receiving a new address input and data becoming available at the memory output."

वॉलेटाइल तथा नॉनवॉलेटाइल मैमोरीज (Volatile and non Volatile Memories)—वह मैमोरी जिसमें डाटा स्टोर करने हेतु वैद्युत पावर की आवश्यकता होती है, वॉलेटाइल मैमोरी कहलाती है। यदि वैद्युत पॉवर remove कर दी जाये, तो वॉलेटाइल मैमोरी में स्टोर डाटा lost हो जाता है। जो मैमोरीज़ बिना वैद्युत पॉवर के भी डाटा स्टोर कर सकती है नॉन वॉलेटाइल मैमोरीज़ कहलाती है। कई सेमी कन्डक्टर मैमोरीज वॉलेटाइल की होती हैं। सभी मैगनेटिक मैमोरीज नॉन वॉलेटाइल होती हैं।

"Non volatile memories can store information without electrical power. All magnetic memories are non-volatile."

रैन्डम एक्सेस तथा सीक्वैंशल एक्सैस मैमोरीज़ (Random Access and Sequential Access Memories (RAM and SAM))—रैन्डम एक्सैस मैमोरी (RAM) में सभी मैमोरी लोकेशन का एक्सैस टाइम एक समान होता है अर्थात् वह मैमोरी जिसमें किसी मैमोरी वर्ड से डाटा read या write करने में लगने वाला समय मैमोरी वर्ड की physical location पर निर्भर नहीं होता, RAM कहलाती है। अधिकतर सेमी कन्डक्टर मैमोरीज़ RAM होती हैं।

सीक्वैंशल एक्सैस मैमोरीज़ में एक्सैस टाइम नियत नहीं होता बल्कि एड्रैस लोकेशन पर निर्भर होता है। अतः, किसी एड्रैस लोकेशन को locate करने हेतु क्रमवार एक-एक करके address की sequencing की जाती है तथा desired address तक पहुँचा जाता है स्पष्ट है कि desired address तक reach करने (अर्थात् sequencing) की प्रक्रिया में प्रत्येक address का access time भिन्न होगा। SAM के उदाहरण हैं—मैगनेटिक टेप, डिस्क, मैगनेटिक बबल मैमोरी इत्यादि। स्पष्ट है कि audio tape cassette SAM का उदाहरण है जिसमें किसी गीत तक पहुँचने हेतु टेप कोfast forward या rewind करना पड़ता है। इसी प्रकार RAM की तुलना आप Jukebox से कर सकते हैं जिसमें किसी गीत को सिलैक्ट करने हेतु सही कोड पंच कर दिया जाता है तथा किसी भी song को सिलैक्ट करने में same time लगता है।

"RAM is a type of memory in which the access time is same for any address in the memory. SAM is a type of memory in which the access time is not constant but varies depending on address location."

रीड/राइट मैमोरी तथा रीड ओनली मैमोरी (Read/Write Memory and Read Only Memory (RWM and ROM))—वह मैमोरी जिसमें read व write, दोनों ऑपरेशन किये जा सकते हैं, Read/Write Memory कहलाती है।

रीड-ओनली मैमोरी (ROM) में सिर्फ रीड ऑपरेशन किया जा सकता है। ROM में डाटा को एक बार स्टोर कर दिया जाता है (फैक्टरी में या programmer) तथा फिर इस डाटा को चेंज नहीं किया जा सकता, केवल read किया जा सकता है। सभी ROMs नॉन-वॉलेटाइल होती हैं तथा पॉवर सप्लाई remove करने पर भी डाटा स्टोर कर सकती हैं।

"Any memory that can be read from or writte into with equal ease is called a Read/Write Memory. In a ROM, the data is written into only once, and this operation is perfo- n.ed at the factory (or by programmer). Thereafter, the infc mation can only be read from the memory."

स्टैटिक तथा डायनैमिक मैमोरी (Static and Dynamic Memory)—वह सेमी कन्डक्टर मैमोरीज़ जिनमें पॉवर सप्लाई करने पर डाटा Permanently stored रहता है तथा उनमें डाटा को मैमोरी में periodically rewrite करने की आवश्यकता नहीं होती, स्टेटिक मैमोरी कहलाती है।

कुछ ऐसी सेमी कन्डक्टर मैमोरी होती है जिनमें पॉवर सप्लाई करने के पश्चात् भी डाटा permanently stored नहीं रहता अर्थात् डाटा को memory में periodically rewrite करना पड़ता है। ऐसी memories मे capacitors पर stored वोल्टेज मे leakage होते है जिसको compensate करना पड़ता है। इस प्रक्रिया को refreshing कहते हैं। जिन मैमोरी युक्तियों में refreshing आवश्यक होती है डायनैमिक मैमोरी कहलाती हैं।

डायनैमिक रैम मैमोरी MOS तथा CMOS ट्रान्जिस्टर गेटों द्वारा बनती है तथा इसमें बिट का संग्रह एक आवेश (charge) के रूप में होता है। यह आवेश MOS ट्रान्जिस्टर के सोर्स तथा गेट के बीच बनने वाले संधारित्र (capacitor) पर स्टोर किया जाता है। इस मैमोरी की कमी यह होती है कि आवेश (Bit information) लीक (leak) हो जाने के कारण संग्रहीत डाटा को प्रत्येक कुछ सैकण्डों (लगभग 2 मिली सैकेण्ड) पश्चात् rewrite करना आवश्यक होता है। इसे डायनैमिक मैमोरी की refreshing कहते हैं तथा इसके लिये अतिरिक्त परिपथ की आवश्यकता होती है। इस कारण सिस्टम की लागत (cost) बढ़ जाती है। परन्तु डायनैमिक रैम मैमोरी का लाभ यह है कि इसकी पैकिंग डैन्सिटी (Packing density) अधिक होती है अर्थात् कम से कम चिप क्षेत्र में अधिक बिटों को संग्रह किया जा सकता है तथा इसमें कम पॉवर खर्च होती है। इसीलिये यह मैमोरी सस्ती होती है। सिस्टम मैमोरी साइज़ कम से कम 8KB होने पर डायनैमिक रैम मैमोरी प्रयोग करना सस्ता पड़ता है।स्टैटिक रैम मैमोरी में re-freshing की आवश्यकता नहीं होती किन्तु इसमें अधिक पॉवर खर्च होती है तथा इसकी पैकिंग डैन्सिटी भी कम होती है अत: यह मैमोरी डायनैमिक रैम के अपेक्षाकृत अधिक कीमत वाली होती हैं। SRAM को बाइपोलर तथा MOS दोनों प्रकार के ट्रान्जिस्टर्स से बनाया जा सकता है जबकि DRAM को केवल MOS ट्रान्जिस्टर्स से बनाया जाता है। बाइपोलर मैमोरी चिप्स का मुख्य लाभ यह है कि इनके द्वारा अधिक स्पीड प्राप्त होती है।

आजकल बाज़ारों में iRAM भी उपलब्ध हो रही है। iRAM अर्थात् Integrated (एकीकृत) RAM। इस मैमोरी के अन्तर्गत एक डायनैमिक रैम, इसके कन्ट्रोल तथा रिफ्रैश सर्किट को एक ही IC चिप पर एकीकृत किया जाता है। इस मैमोरी का आविष्कार होने के पहले डायनैमिक रैम के लिये कन्ट्रोल सर्किट तथा रिफ्रैश सर्किट अतिरिक्त IC चिप द्वारा डायनैमिक रैम कन्ट्रोलर के रूप में प्रयोग किये जाते थे। किन्तु अब VLSI तकनीक के आधार पर iRAM प्राप्त करना सुविधाजनक हो गया है।

"Semiconductor memory devices in which the stored data will remain permanentty stored as long as the power is applied, without the need for periodically rewriting the data into the memory, are called static memories.

Semiconductor memory devices in which the stored data will not remain permanently stored, even with power applied, unless the data are periodically rewritten (re-freshed) into the memory are called dynamic memories."

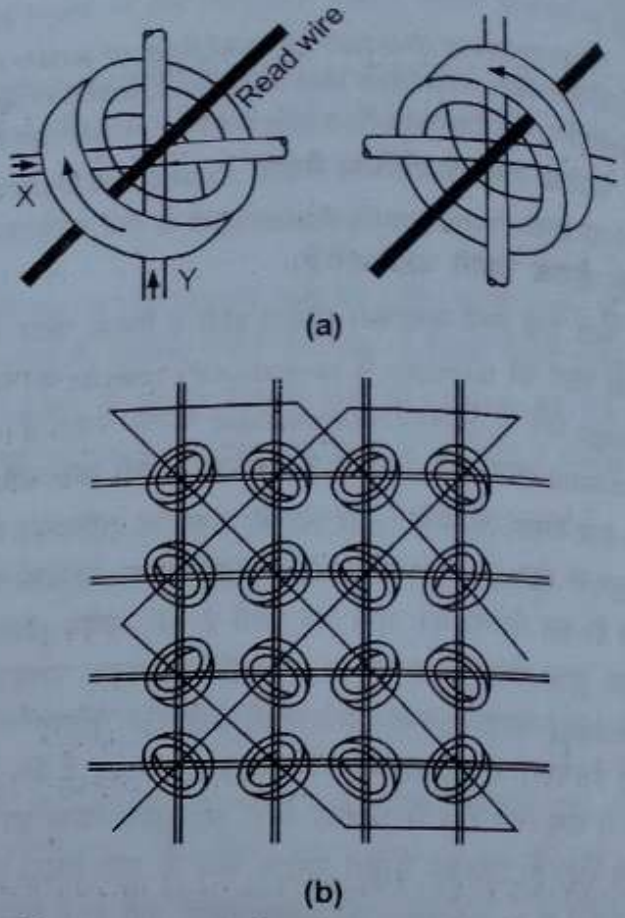
इन्टरनल एंड ऑक्सिलरी मैमोरी (Internal and Auxilliary Memory)—इन्टरनल मैमोरी को कम्प्यूटर की मेन मैमोरी या वर्किंग मैमोरी भी कहा जाता है। यह उन इन्स्ट्रक्शन व डाटा को स्टोर करती है, जिन पर CPU currently work कर रहा होता है। यह सैमीकन्डक्टर मैमोरी होती है तथा कम्प्यूटर की उच्चतम स्पीड वाली मैमोरी होती है।

ऑक्सिलरी मैमोरी को mass storage भी कहा जाता है। इस मैमोरी की स्पीड इंटरनल मैमोरी से कम होती है तथा यह भारी मात्रा में डाटा (massive amount of data) स्टोर कर सकती है।

"Internal memory stores instructions and data CPU is currently working on. Auxiliary memory stores massive amount of information external to internal memory."

सैमीकन्डक्टर व नॉन-सैमीकन्डक्टर मैमोरी (Semi conductor and Non-semi conductor memories)—अर्द्धचालक मैमोरी अर्द्धचालक पदार्थों की बनी होती है जबकि नॉन-अर्द्धचालक मैमोरी चुम्बकीय मैमोरी होती है। चुम्बकीय मैमोरी के उदाहरण हैं—फ्लॉपी डिस्क, कैसेट, हार्ड डिस्क, चुम्बकीय क्रोड, चुम्बकीय टेप, चुम्बकीय ड्रम इत्यादि। कम्प्यूटर्स में प्रयुक्त चुम्बकीय मैमोरी चुम्बकीय छल्लों (magnetic ferrites) के रूप में होती है। चित्र (7.3a) में चुम्बकीय कोर तथा चित्र (7.3b) में कोर विन्यास (core configuration) प्रदर्शित है।

"Semi conductor memory is made of semiconductor materials while non-semiconductor memory is made of magnetic materials. The examples of magnetic memory are Floppy disc, Cassette, Hard disk, Magnetic tapes, Magnetic core and Magnetic drums."



चित्र 7.3—(a) चुम्बकीय कोर (b) चुम्बकीय मैमोरी कोर विन्यास

डिस्ट्रक्टिव तथा नॉन-डिस्ट्रक्टिव मैमोरी (Destructive and Non-destructive memory)—डिस्ट्रक्टिव मैमोरी वह होती है जिसमें एक बार डाटा read करने के पश्चात् डाटा नष्ट हो जाता है अर्थात् destructive memory में प्रत्येक read ऑपरेशन के बाद write operation करना पड़ता है। उदाहरणत: मैगनेटिक कोर मैमोरी। नॉन डिस्ट्रक्टिव मैमोरी में read-out क्रिया के पश्चात् भी डाटा मैमोरी में सुरक्षित (save) रहता है अर्थात् नॉन डिस्ट्रक्टिव मैमोरी में डाटा को रीड करने के पश्चात् उस लोकेशन का डाटा पुन: use किया जा सकता है उदाहरण—फ्लिप फ्लॉप।

स्पष्ट है कि किसी मैमोरी लोकेशन से डाटा read करते समय यदि डाटा नष्ट हो जाये तो इसे destructive read out कहा जाता है तथा यदि read करने के पश्चात् भी डाटा उस लोकेशन पर सुरक्षित रहे (अर्थात् सिर्फ copy क्रिया हो) तो इसे नॉन डिस्ट्रक्टिव read out कहते हैं।

"During read operation, if data is lost from the memory, than the read out is called a destructive read out. Memories with destructive read out are called destructive memories.

Memories having a non destructive read out i.e., data remains intact during a readout are termed non destructive memories."

प्रैक्टिस प्रश्न

(i) एक मैमोरी चिप का साइज 4 K × 8 है। इस मैमोरी ने कितने words स्टोर किये जा सकते हैं? इस मैमोरी में कुल कितनी बिट्स स्टोर की जा सकती है? इस मैमोरी में address bits की संख्या बताइये।

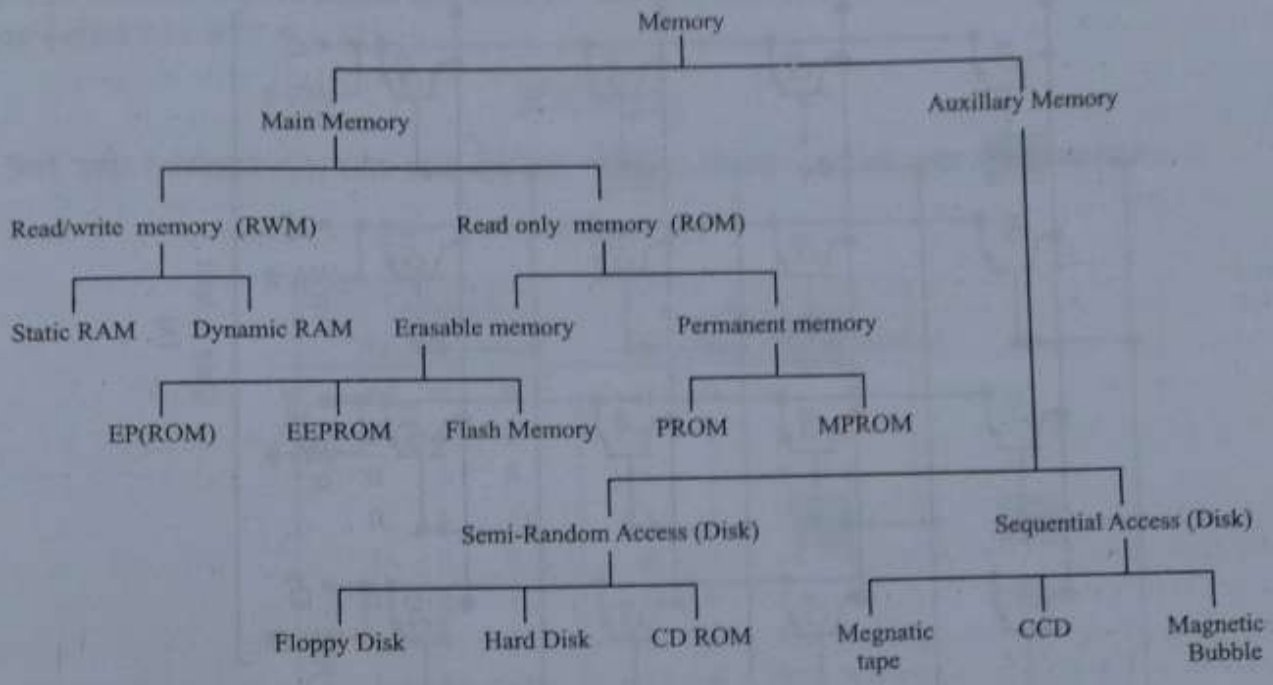
(ii) कौन-सी मैमोरी ज्यादा बिट्स स्टोर कर सकती है–5 M × 4 या 100K × 16 ?

(iii) यदि किसी मैमोरी का साइज़ 64 K × 8 है तो इसके लिये address bus व data bus का साइज़ बताइये।

§ 7.3. मैमोरी वर्गीकरण (Memory Classification) :

मैमोरी का वर्गीकरण तालिका 7.2 में प्रदर्शित है।

तालिका 7.2

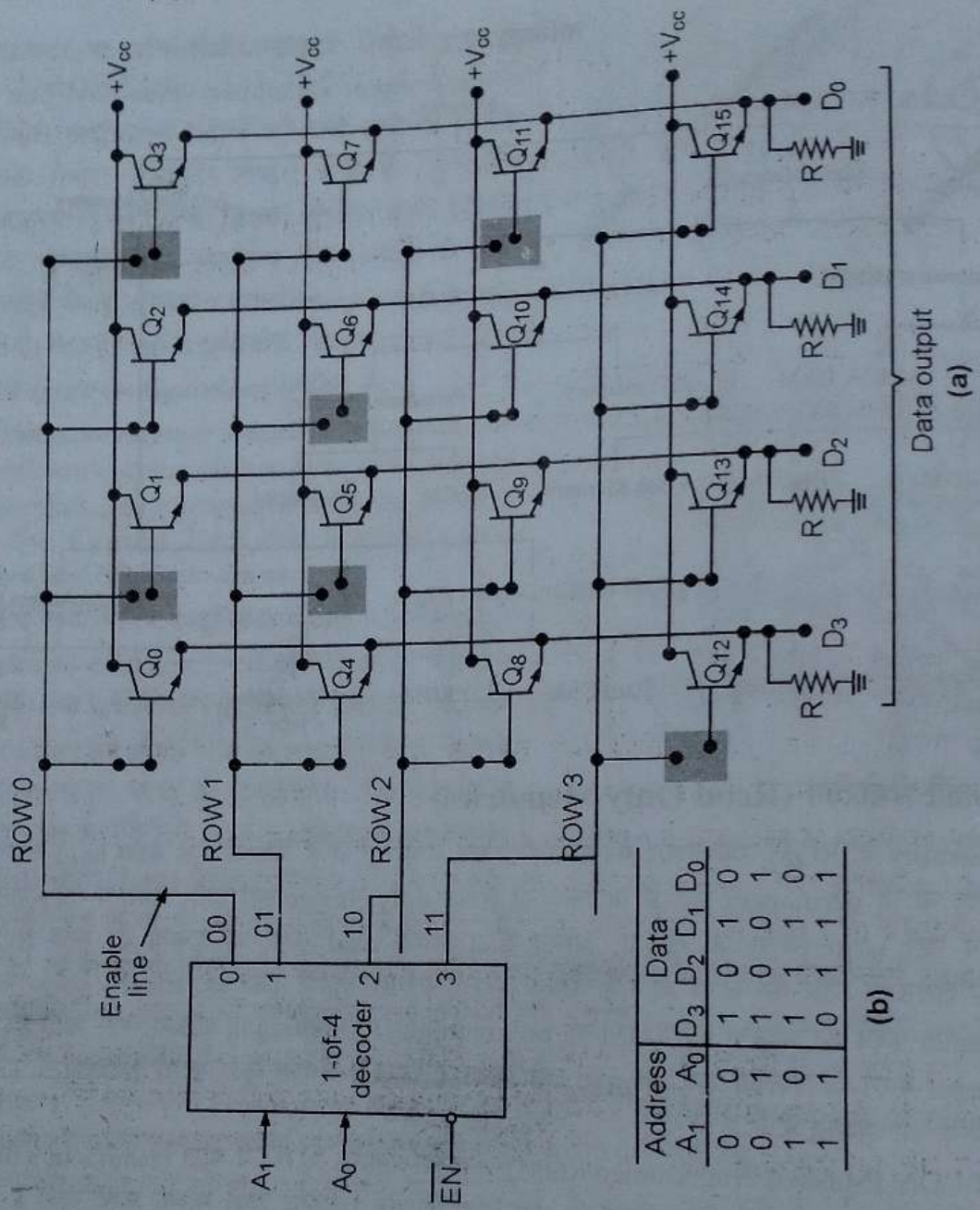


§ 7.3. रीड-ओनली मैमोरीज (Read Only Memories) :

Read only memory या ROM एक प्रकार की सैमीकन्डक्टर मैमोरी होती है जिसको ऐसे डाटा hold करने हेतु डिज़ाइन किया जाता है जो कि या तो permanent होते हैं या फिर जो frequently change नहीं होते। सामान्य ऑपरेशन में, कोई भी नया डाटा ROM में नहीं write किया जा सकता, अर्थात् डाटा केवल read किया जा सकता है। कुछ ROMs में डाटा manufacturing के दौरान ही स्टोर कर दिया जाता है जबकि कुछ दूसरी ROMs में डाटा electrically enter किया जा सकता है। ROM में डाटा enter करने की प्रक्रिया को ROM की programming या burning in प्रक्रिया कहा जाता है। कुछ ROMs में एक बार डाटा enter करने के पश्चात् उसे change नहीं किया जा सकता जबकि कुछ दूसरी ROMs में डाटा को erase तथा reprogram किया जा सकता है।

मास्क प्रोग्राम्ड ROM (Masked Programmed ROM)—मास्क प्रोग्राम्ड ROM में डाटा manufacture द्वारा customer की specification के अनुसार स्टोर कर दिया जाता है। एक फोटोग्राफिक निगेटिव जिसे मास्क कहा जाता है द्वारा चिप पर कनैक्शन को कन्ट्रोल किया जाता है। स्पष्ट है कि अलग-अलग सूचना स्टोर करने हेतु अलग-अलग मास्क चाहिये। चूँकि यह मास्क महंगे होते हैं, अत: MROM तभी उपयुक्त रहते हैं जब आपको same ROM की large quantity आवश्यक हो जैसे कोई mathematical table, CRT displays हेतु कैरेक्टर कोड इत्यादि। इन ROMs का मुख्य दोष यह है कि एक बार डाटा स्टोर होने के पश्चात् उसे चेंज नहीं किया जा सकता अर्थात् यह ROM reprogrammed नहीं की जा सकती।

चित्र 7.4 में एक छोटे से बाइपोलर ROM का परिपथ प्रस्तुत है। इसमें 16 मैमोरी सैल हैं, तथा चार-चार सैल को चार rows में arrange किया गया है। प्रत्येक सैल एक npn बाइपोलर ट्रांजिस्टर है तथा इनको कॉमन कलैक्टर कानफिगुरेशन में कनैक्ट किया गया है (Input बेस पर तथा आउटपुट एमीटर पर)। प्रत्येक row एक चार बिट का रजिस्टर बनाती है। देखें कि कुछ ट्रांजिस्टर की बेस को उस row की enable लाइन से कनैक्ट किया गया है जबकि कुछ ट्रांजिस्टर के बेस को उस row की enable लाइन से कनैक्ट नहीं किया गया। यदि मैमोरी सैल में 1 स्टोर करना है तो कनैक्शन किया जायेगा, तथा 0 स्टोर करना है तो कनैक्शन नहीं किया जायेगा ।



Address		Data			
A_1	A_0	D_3	D_2	D_1	D_0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	1	1	1

(b)

चित्र 7.4-(a) बाइपोलर MROM की संरचना, जहाँ प्रत्येक मैमोरी सल हेतु एक बाइपोलर ट्रांजिस्टर प्रयुक्त किया गया हैं (b) प्रत्येक एड्रैस पर स्टोर्ड बाइनरी वर्ड

स्पष्ट है कि row 0 में 1010, row 1 में 1001, row 2 में 1110 तथा row 3 में 0111 स्टोर किया गया हैं (चित्र 7.4b)।

1–of–4 डिकोडर द्वारा एड्रैस इनपुट A_1A_0 को डिकोड किया जाता है तथा जिस row का डाटा read किया जाना है उसे सिलैक्ट किया जाता है उदाहरणत: यदि row 2 का डाटा read किया जाना है तो $A_1 = 1$ तथा $A_0 = 0$ किया जायेगा। इससे row 2 लाइन high हो जायेगी तथा Q_8, Q_9, Q_{10} ऑन हो जायेगें (Q_{11} ऑन नहीं होगा क्योंकि उसकी बेस पर कनैक्शन नहीं है)। अत: डाटा आउटपुट D_3, D_2, D_1 हाई तथा D_0 low प्राप्त होगी। अर्थात् आउटपुट पर 1110 प्राप्त होगा। इसी प्रकार किसी भी row को सिलैक्ट करके उसे read किया जा सकता है।

ध्यान दें कि इस ROM की किसी भी row को read करने हेतु $\overline{EN}$ सिगनल low होना आवश्यक है। यदि $\overline{EN} = 1$ होगा तो डिकोडर की सभी आउटपुट inactive हो जायेंगी तथा सभी ट्रांजिस्टर ऑफ रहेंगे। यह बात भी ध्यान देने योग्य है कि एक समय में केवल एक ही row के ट्रांजिस्टर्स activate होंगे।

बाइपोलर MROM निम्न क्षमता वाली मैमोरीज़ हेतु उपलब्ध हैं। 74187 एक मुख्य बाइपोलर MROM IC है जो कि 256×4 क्षमता की मैमोरी है। इसका access time 40 ns है। 7488 A भी एक बाइपोलर MROM IC है जिसको access time 45 ns तथा कैपेसिटी 32×8 है।

प्रैक्टिस प्रश्न

(i) निम्न डाटा (तालिका 7.3) स्टोर करने हेतु एक MROM डिज़ाइन कीजिये तथा परिपथ बनाइये।

तालिका 7.3

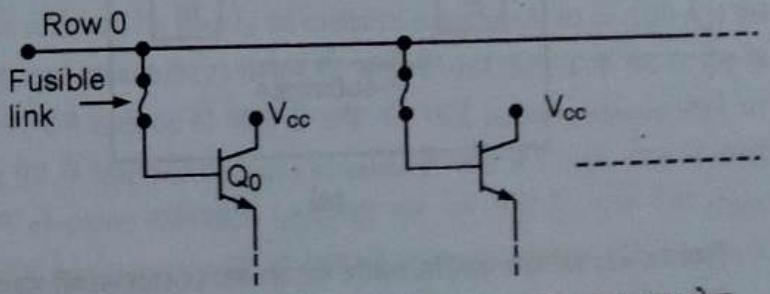
Address			Data			
A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	1	1
1	0	1	1	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	0	1

(ii) चित्र (7.3) में प्रदर्शित MROM को mathematical फंक्शन $y = x^2 + 3$ स्टोर करने हेतु डिज़ाइन कीजिये। इनपुट x दो बिट A_1A_0 की बाइनरी इनपुट है। इस MROM का चित्र बनाइये।

"The mask programmed ROM has its storage location written into (programmed) by manufacturer according to customers specification. A photographic negative called mask is used to control the interconnection on the chip. The major disadvantage of this type of ROM is that they cannot be reprogrammed in the event of design change requiring a modification of stored data."

प्रोग्रामेबल ROMs (Programmable ROMs or PROMs)—मास्क प्रोग्रामेबल ROM महंगा होता है तथा high volume applications हेतु ही उपयुक्त रहता है। Lower volume applications हेतु fusible-link PROMs का विकास किया गया जो कि user programmable होते हैं अर्थात् जिनको manufacturing के दौरान प्रोग्राम नहीं किया जाता बल्कि user के द्वारा प्रोग्राम किया जाता है। एक बार प्रोग्राम होने के पश्चात् PROM एक MROM की तरह ही हो जाता है जिसको पुन: प्रोग्राम नहीं किया जा सकता। इसलिये PROM को one time programmable ROM भी कहा जाता है।

Fusible-link PROM की संरचना MROM की भांति ही होती है। किन्तु जबकि MROM में बेस कनैक्शन्स को manufacture द्वारा ही close या open कर दिया जाता है, जबकि fusible link PROM में प्रत्येक कनैक्शन thin fusible link का बना होता है (चित्र 7.5)। जिसको user अपनी इच्छा से या तो वैसे ही छोड़ सकता है या blow कर सकता है।



चित्र 7.5-PROM में प्रत्येक कनैक्शन fusible link का बना होता

अत: fusible link PROM में user selectively कुछ links को blow करके इच्छानुसार डाटा को स्टोर कर सकता है। जिस सैल में 1 स्टोर करना होता है, वहाँ का fuse link blow नहीं किया जाता तथा जहाँ 0 स्टोर करना होता है वहाँ का fuse blow कर दिया जाता है। अत: PROM की किसी location में डाटा स्टोर करने हेतु address इनपुट पर address apply किया जाता है तथ डाटा पिनों पर वांछित डाटा को प्लेस किया जाता है तथा IC पर स्पेशल प्रोग्रामिंग पिन होती है जिस पर high

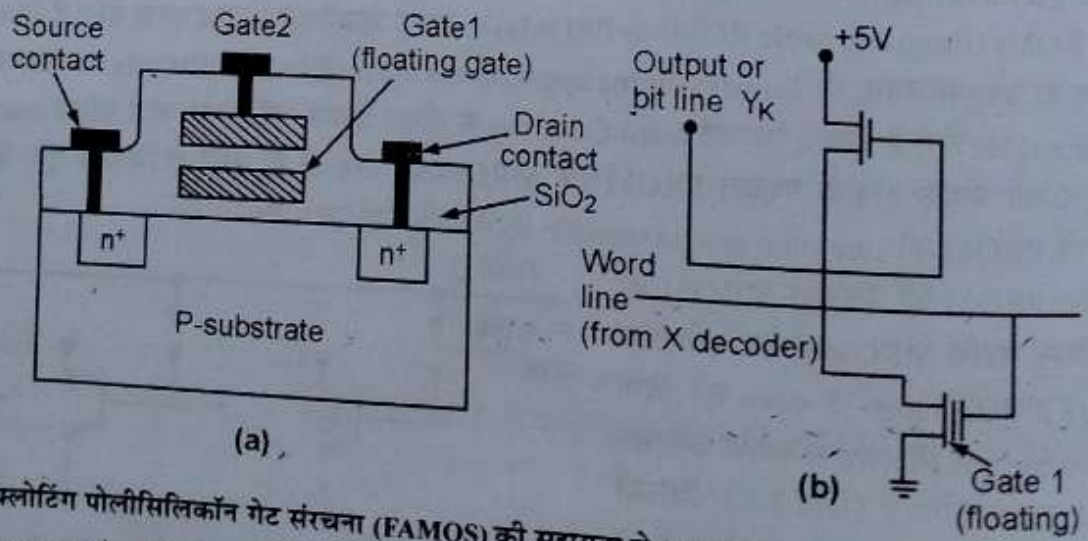
voltage pulse (10–30V) दे दी जाती है। इससे selected fusible links जहाँ 0 स्टोर होना होता है, में large current flow हो जाता है, जिससे fuse burn हो जाती है व लिंक open हो जाता है। एक बार सभी लोकेशन्स के प्रोग्राम हो जाने के पश्चात् डाटा को बार-बार address इनपुट apply करके read किया जा सकता है। पॉवर के remove होने पर भी डाटा स्टोर रहता है।

74186 बाइपोलर PROM IC का उदाहरण है, जिसकी क्षमता 64 × 8 तथा access time 50 ns है। TBP285166 भी PROM IC है जिसकी क्षमता 2K × 8 है।

"PROMs are user programmable ROMs, and have fusible links. They are not programmed during manufacturing process but are cystom-programmed by the user. Once programmed, the PROM cannot be erased and the data cannot be changed. Hence, they are one-time programmable ROMS. To store the data on the PROMS, the user can selectively blow off the fuse links. The process of programming PROM in done by a special appratus called a PROM programmer."

इरेसेबिल प्रोग्रामेबल ROM (Erasable Programmable ROM or EPROM)— EPROM वह ROM है जो user द्वारा प्रोग्राम की जा सकती है तथा जितनी बार user चाहे, वह इसको erase कर पुन: प्रोग्राम (reprogram) भी कर सकता है। एक बार प्रोग्राम हो जाने के बाद EPROM एक non-volatile memory की भाँति व्यवहार करती है। EPROM की प्रोग्रामिंग हेतु इस पर 10–25 V की रेंज में वोल्टेज apply की जाती है (लगभग 50 ms प्रति address लोकेशन के लिये)। चिप की सम्पूर्ण प्रोग्रामिंग हेतु कुछ मिनट का समय लगता है।

EPROM में स्टोरेज सैल MOS ट्रांजिस्टर्स होते हैं जिनके दो पोलीसिलिकॉन के गेट होते हैं। इनमें से एक गेट के electrical connection नहीं होते अर्थात् यह गेट फ्लोटिंग होता है (चित्र 7.6(a))। इस डबल गेट NMOS ट्रांजिस्टर को FAMOS (Floating-gate Avalanche-Injection Metal Oxide Semiconductor) भी कहा जाता है। नार्मल अवस्था में प्रत्येक ट्रांजिस्टर ऑफ रहता है तथा प्रत्येक सैल में 1 स्टोर रहता है। ट्रांजिस्टर को ऑन करने हेतु high voltage programming pulse सप्लाई की जाती है जो फ्लोटिंग क्षेत्र में उच्च-ऊर्जा वाले इलेक्ट्रॉन inject करती है। पल्स के टर्मिनेट होने के पश्चात् यह इलैक्ट्रॉन इस क्षेत्र में trapped हो जाते हैं क्योंकि कोई डिस्चार्ज पथ नहीं होता। अत: ट्रांजिस्टर permanently on हो जाता है तथा सैल में 0 स्टोर हो जाता है। प्रोग्रामिंग प्रौसेस में EPROM के address व data pins को use कर यह select किया जा सकता है कि कौन से मैमोरी सैलों को 0 व कौन से मैमोरी सैलों को 1 पर प्रोग्राम करना है। EPROM सैल का परिपथ चित्र 7.6(b) में प्रदर्शित है।

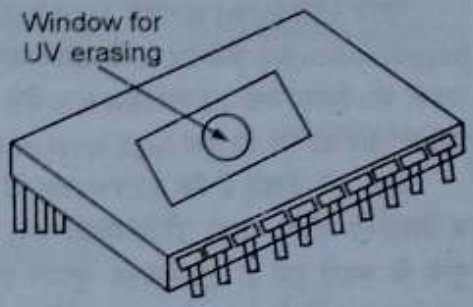


चित्र 7.6-(a) फ्लोटिंग पोलीसिलिकॉन गेट संरचना (FAMOS) की सहायता से EPROM प्राप्त करना (b) EPROM सैल का परिपथ

एक बार EPROM सैल के प्रोग्राम हो जाने के पश्चात् यह उस अवस्था में ही रहता है। पॉवर को हटाने पर भी सैल की अवस्था चेंज नहीं होती। SiO_2 की (superb) insulating क्षमता फ्लोटिंग गेट में फंसे चार्ज को कई सालों तक trap करके रख सकती है (ऐसा अनुमानित है कि storage 125°C होने पर भी 10 वर्ष बाद 70% चार्ज maintained रहेगा)।

EPROM को erase करने हेतु इसे पराबैंगनी प्रकाश (ultraviolet light) को expose किया जाता है। यह UV किरणें EPROM IC पर लगी window (चित्र 7.6(c)) पर डाली जाती हैं। इस UV light द्वारा फ्लोटिंग गेट व सिलिकॉन सब्सट्रेट के मध्य

फोटोधारा प्रवाहित होती है जो फ्लोटिंग गेट में स्टोर्ड चार्ज को remove कर देती है तथा ट्रांजिस्टर को ऑफ कर देती है, जिससे सैल पुन: लॉजिक 1 अवस्था में आ जाता है। Erasing हेतु लगभग 15 से 20 मिनट तक UV light apply करने की आवश्यकता होती है। दुर्भाग्यवश, ऐसी कोई विधि नहीं है जिससे कुछ सिलैक्टेड सैल erase किये जा सकें अर्थात् UV light सभी सैलों को एक साथ erase कर देती है तथा erased EPROM में सभी सैल पर 1 स्टोर हो जाता है। एक बार erase करने के पश्चात् EPROM को पुन: प्रोग्राम किया जा सकता है।



चित्र 7.6 (c)-EPROM IC में UV प्रकाश डालने हेतु लगी Window

Intel 2732 EPROM IC का उदाहरण है। यह 4K × 8 NMOS EPROM है जो +5V पर कार्य करता है (बतायें कि इसमें कितनी एड्रैस इनपुट्स व कितनी डाटा इनपुट्स होगी?)

आपने देखा कि EPROM में डाटा erase व reprogram किया जा सकता है किन्तु उक्त विश्लेषण के अध्ययन करने के बाद मुझे आपको EPROM के मुख्य दोष गिनाने की आवश्यकता नहीं होनी चाहिये—

(i) erase व reprogram करने हेतु इसको सर्किट से हटाना आवश्यक है अर्थात on-circuit erasing सम्भव नहीं है।

(ii) erase operation द्वारा पूरी चिप erase हो जाती है अर्थात् कुछ लोकेशन को select करके erase करना सम्भव नहीं है।

(iii) erase व reprogramming प्रोसेस 20 मिनट से अधिक लेता है अर्थात् erase व reprogramming समय अधिक है।

"A EPROM can be programmed by the user and it can also be erased and reprogrammed as often as desired. The process of programming an EPROM involves the application of special voltage levels for specified amount of time. The programming is done in a separate apparatus and takes several minutes.

The storage cells in an EPROM are MOS tranistors with a floating silicon gate. In normal state, each transistor is off hence storing a 1. A transistor can be turned on by application of high voltage programming pulse that injects high energy electrons in the floating gate. The electrons remained trapped in this region once the pulse is terminated, since there is no discharge path. This keeps the transitor permanently on thereby storing a 0.

The EPROM cell can be erased by exposing it to ultraviolet light applied through a window on the chip. This erasing time takes 15 to 20 minutes of exposure to UV rays. All the chip is erased at the same time."

इलैक्ट्रिकली इरेसेबिल PROM (Electrically Erasable PROM or EEPROM or E^2 PROM)—EPROM के उक्त दोषों से निजात पाने हेतु EEPROM का विकास किया गया। EEPROM की संरचना EPROM के समान होती है (अर्थात् फ्लोटिंग गेट संरचना) किन्तु इसमें गेट 1 (फ्लोरिंग) तथा सिलिकॉन (सब्सट्रेट) के मध्य ऑक्साइड की thickness बहुत कम कर दी जाती है (the oxide thickness between gate 1 and silicon is reduced by nearly an order of magnitude to ≈ 100 Å)। इस modification से E^2 PROM में वैद्युत erasibility (electrical erasability) सम्भव हो जाती है। MOSFET के गेट व ड्रेन के मध्य उच्च वोल्टेज (21 V) एप्लाई करने पर फ्लोटिंग गेट में चार्ज induce हो जाता है तथा यह चार्ज power remove करने पर भी स्टोर रहता है। वोल्टेज को reverse करने पर floating गेट में trapped charges remove हो जाते हैं (SiO_2 परत के पतले हो जाने से quantum-mechanical tunnelling के कारण electron ऑक्साइड layer को पार कर जाते हैं) तथा सैल erase हो जाता है। चूँकि यह charge transport mechanism बहुत low currents पर हो जाती है, अत: EEPROM की erasing व programming in-circuit ही की जा सकती है।

EEPROM का अन्य गुण यह है कि इसमें individual bytes को electrically erase व rewrite किया जा सकता है। Write operation के वक्त internal circuitry उस address पर पहले से stored data को automatically erase कर देती है। इस byte erasability के कारण EEPROM में stored data में changes करना आसान होता है। इसके अतिरिक्त EEPROM को program करने में (EPROM की तुलना में) समय भी कम लगता है। EEPROM location पर write करने में केवल 5 ms (लगभग) का समय लगता है जबकि EPROM में 50 ms लगते हैं।

पुराने EEPROM IC जैसे Intel 2816 में मैमोरी चिप के अलावा external circuitry लगाना आवश्यक होता था। यह support circuitry जैसे 21 V प्रोग्रामिंग वोल्टेज हेतु तथा erase व प्रोग्रामिंग हेतु टाइमिंग तथा sequence कन्ट्रोलिंग हेतु लगानी पड़ती थी। किन्तु नयी EEPROM ICs जैसे Intel 2864 में यह विशेषता होती है कि सभी सपोर्ट circuitry मैमोरी के IC में ही इंटीग्रेट कर दी गई है। इस high level of Integration के कारण EEPROM IC को प्रयोग अत्यन्त सरल हो गया है।

अत: हमें देखते हैं कि EEPROM के मुख्य गुण हैं—ऑन सर्किट इरेज़ेबिटिली (on-circuit erasability), कम इरेज़िंग व प्रोग्रामिंग समय, बाइट इरेज़ेबिलिटी तथा हाई लैवल ऑफ इंटीग्रेशन। किन्तु यह गुण हमें मुफ्त में प्राप्त नहीं हो जाते। इन गुणों के बदले हमें दो मुख्य दण्ड भुगतने होते हैं—डैन्सिटी तथा cost। EEPROM का मैमोरी सैल जटिल होता है तथा on-chip support circuitry के कारण पैकिंग डैन्सिटी में EEPROM परिपथ EPROM से काफी पीछे छूट जाता है। आपको यह जानकर आश्चर्य नहीं होना चाहिये कि 1 M bit EEPROM, 1 M bit EPROM के तुलना में दोगुना सिलिकॉन क्षेत्र occupy करता है। अत:, ऑपरेशन में श्रेष्ठ होने के बावजूद EEPROM उन अनुप्रयोगों में EPROM को replace नहीं कर पाया है जहाँ डैन्सिटी व cost मुख्य factors हैं।

"EPROM has an important characteristics namely its electrical erasability. Since it can be erased at very low currents, the erasing and programming of an EEPROM can be done in-circuit. Another advantage of EEROM is the ability to electrically erase and rewrite individual bytes."

फ्लैश मैमोरी (Flash Memory)—हमने देखा कि EPROM non volatile है, उच्च डैन्सिटी व कम कीमत प्रति बिट वाली मैमोरी है। किन्तु इसको इरेज़ व पुन: प्रोग्राम करने हेतु परिपथ से remove करना आवश्यक होता है। EEPROM non-volatile, कम एक्सैस टाइम, त्वरित in-circuit erasure, तथा individual byte reprogramming सुविधायें प्रदान करती है किन्तु इनकी डैन्सिटी कम व कीमत EPROM के तुलना में बहुत अधिक होती है। ऐसे में सैमीकन्डक्टर इंजीनियर्स के लिये यह चुनौती ही थी कि एक ऐसी मैमोरी का विकास किया जाये जो कि EEPROM के on-circuit erasure गुण को बनाये रखते हुए डैन्सिटी व cost में EPROM के नज़दीक हों। इस चुनौती का उत्तर फ्लैश मैमोरी के रूप में प्राप्त हुआ।

फ्लैश मैमोरी सैल एक साधारण EPROM सैल की भाँति होता है, किन्तु यह थोड़ा बड़ा होता है। इसकी गेट-ऑक्साइड परत पतली होती है जिससे वैद्युत इरेज़िबिलिटी सम्भव हो जाती है किन्तु इसका EEPROM से अधिक डैन्सिटी में बनाया जा सकता है। इसकी cost भी EEPROM से कम होती है। इसका erase तथा write टाइम भी कम होता है। फ्लैश चिप में bulk operation (chip के सभी cells को erase करना) UV EPROM से कम समय लेता है। (जहाँ UV EPROM में लगभग 20 मिनट लगते हैं वहीं फ्लैश मैमोरी में कुछ सौ मिलीसैकेण्ड लगते हैं)। नयी फ्लैश मैमोरी में sector erasing सुविधा उपलब्ध है जिसमें सम्पूर्ण मैमोरी न erase करके कुछ sectors (512 बाइट का एक सैक्टर) एक समय में erase किये जा सकते हैं। अत: यदि memory का कुछ भाग update करना होता है तो सम्बन्धित sectors erase करने से काम चल जाता है, सम्पूर्ण मैमोरी की erase करके reprogram नहीं करना पड़ता।

IC 28F256A एक CMOS फ्लैश मैमोरी IC है। इसकी कैपेसिटी 32 K × 8 होती है (आप बतायें कि इसकी एड्रैस बस व डाटा बस का साइज क्या होगा?)।

विभिन्न ROMs की तुलना (Comparision of Different ROMs)—विभिन्न ROMs के तुलना तालिका 7.4 में की गई है—

तालिका 7.4–विभिन्न ROMs की तुलना

ROM का प्रकार (Type of ROM)	मुख्य विशेषतायें (Main Features)	टाइप नम्बर (Type Number)
MROM	(i) निर्माता द्वारा प्रोग्राम की गई ROM (ii) high volume operation के लिये उपयुक्त (iii) बेसिक मैमोरी सैल हेतु बाइपोलर ट्रांजिस्टर या MOSFET का प्रयोग (iv) री-प्रोग्रामिंग सम्भव नहीं	74187, 7488A, TMS47256

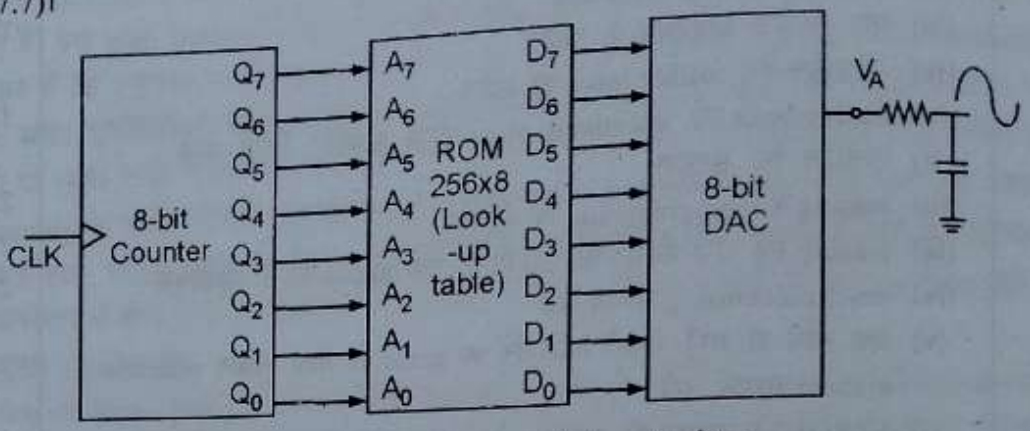
ROM का प्रकार (Type of ROM)	मुख्य विशेषतायें (Main Features)	टाइप नम्बर (Type Number)
PROM	(i) user द्वारा प्रोग्राम की जाने वाली ROM. (ii) re-programming सम्भव नहीं (iii) मूल संरचना MROM के समान (iv) प्रोग्रामिंग हेतु fusible links का प्रयोग (v) fusible links को selectively blow करके प्रोग्रामिंग किया जाना	74186, TBB285166, TMS27PC256
EPROM	(i) फ्लोटिंग गेट संरचना (ii) erasing व reprogramming की सुविधा (iii) erasing हेतु UV प्रकाश को 20 मिनट तक apply करना आवश्यक (iv) on-circuit erasing सम्भव नहीं (v) एक साथ ही सभी मैमोरी लोकेशन का erase हो जाना अर्थात् selective erasure सम्भव नहीं	2732 273512 27C512
EEPROM	(i) EPROM के समान ही फ्लोटिंग गेट संरचना किन्तु पतली आक्साइड परत (ii) ऑन-सर्किट erasibility सम्भव (iii) electric erasibility सम्भव (iv) individual byte erasibility सम्भव (v) support circuitry का मैमोरी चिप में ही इंटीग्रेशन सम्भव (vi) कम erase तथा प्रोग्रामिंग समय (vii) अधिक cost (viii) low डैन्सिटी	2864
Flash Memory	(i) संरचना EPROM के समान (ii) पतली गेट आक्साइड परत के कारण electrical erasability सम्भव (iii) EEPROM से अधिक डैन्सिटी (iv) EEPROM से कम cost. (v) कम erasure समय (vi) सैक्टर erase की सुविधा	28F256A

ROM के अनुप्रयोग (ROM Application)—फर्मवेयर (Firmware)—ROM का एक मुख्य अनुप्रयोग उन डाटा व प्रोग्राम कोड्स का स्टोरेज है, जो कि माइक्रोप्रोसैसर सिस्टम की पॉवर ऑन करते ही उपलब्ध हो जायें। इन डाटा व प्रोग्राम कोड्स को फर्मवेयर (firmware) कहा जाता है क्योंकि यह हार्डवेयर (ROM चिप्स) में स्टोर्ड रहते हैं तथा नार्मल सिस्टम ऑपरेशन के दौरान चेंज नहीं होते।

बूटस्ट्रैप मैमोरी (Bootstrap Memory)—कई मिनीकम्प्यूटर व अधिकतर बड़े कम्प्यूटर्स के ऑपरेटिंग सिस्टम प्रोग्राम ROM में स्टोर न करके external mass memory (जैसे मैगनेटिक डिस्क) में स्टोर किये जाते हैं। ROM में एक अपेक्षाकृत छोटा प्रोग्राम स्टोर किया जाता है जिसको बूटस्ट्रैप प्रोग्राम कहा जाता है। जब कम्प्यूटर ऑन किया जाता है तो वह बूटस्ट्रैप प्रोग्राम के इंस्ट्रक्शन को execute करता है। इन instructions से CPU सिस्टम हार्डवेयर को initialize करता है। यह बूटस्ट्रैप प्रोग्राम डिस्क से इन्टरनल मेन मैमोरी में ऑपरेटिंग सिस्टम प्रोग्राम को लोड करता है। अब कम्प्यूटर ऑपरेटिंग सिस्टम प्रोग्राम को execute करता है तथा इस प्रकार कम्प्यूटर user की commands को respond करने हेतु तैयार होता है। इस startup प्रक्रिया को सिस्टम की बूटिंग (booting up the system) कहा जाता है।

डाटा टेबिल्स (Data Tables)—ROMs का प्रयोग डाटा की तालिकायें स्टोर करने हेतु किया जाता है जैसे trignometric tables, code conversion tables इत्यादि। डाटा कनवर्टर परिपथ में एक प्रकार के कोड में डाटा इनपुट किया जाता है व दूसरे कोड में आउटपुट प्राप्त किया जाता है।

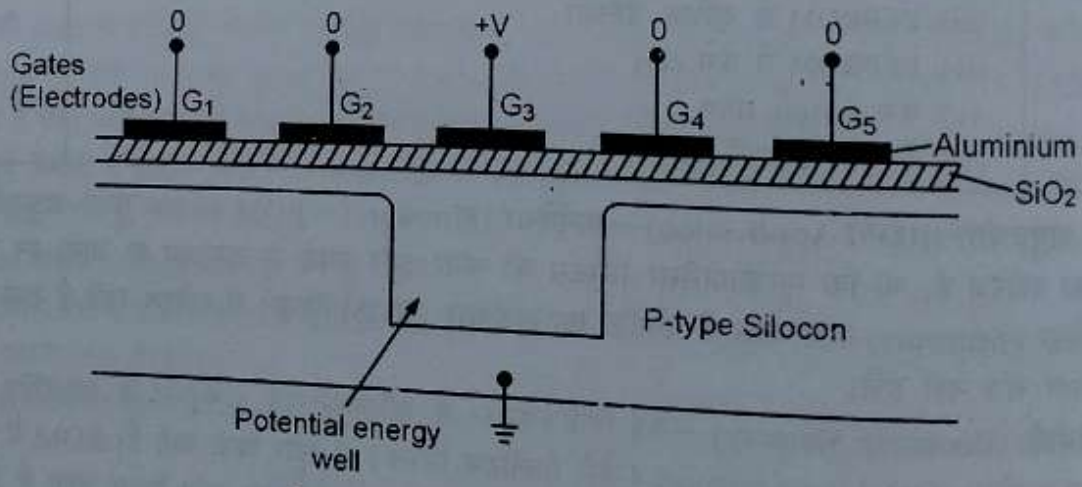
फंक्शन जनरेटर (Function Generator)—फंक्शन जनरेटर वह परिपथ होता है जो कि sine waves, triangle waves, square waves इत्यादि उत्पन्न करता है। इसके लिये एक 8-बिट काउन्टर को ROM चिप की एड्रैस लाइन को दिया जाता है। इस ROM लुक-अप टेबिल की आउटपुट डिजिटल होती है जिसको DAC में देकर आउटपुट पर एनालॉग तरंग प्राप्त की जा सकती है (चित्र 7.7)।



चित्र 7.7-ROM की सहायता से फंक्शन जनरेशन

§ 7.4. चार्ज कपल्ड डिवासेज (Charge Coupled Device or CCD) :

एक ऐसा MOSFET जिसमें एक लम्बा चैनल बना हो तथा उसके सोर्स व ड्रेन के मध्य कई (लगभग 1000) closely spaced इलैट्रोड्स हो, तो वह सीरियल मैमोरी या शिफ्ट रजिस्टर की भाँति फंक्शन कर सकता है। प्रत्येक गेट इलेक्ट्रोड तथा सब्सट्रैट के मध्य MOS कैपेसिटर बन जाता है (चित्र 7.8) जिसमें चार्ज स्टोर किया जा सकता है। उदाहरणतः, यदि सोर्स को लॉजिक 1 एप्लाई किया जाये, तो सोर्स के nearest capacitor में चार्ज स्टोर हो जाता है (गेट G_1 तथा सब्सट्रैट के मध्य), जबकि गेट G_1 पर उपयुक्त वोल्टेज एप्लाई की गई हो। अब यदि इस वोल्टेज को G_1 से हटाकर तुरन्त G_2 पर लगा दिया जाये, तो चार्ज पैकेट G_2 पर move कर जायेगा।



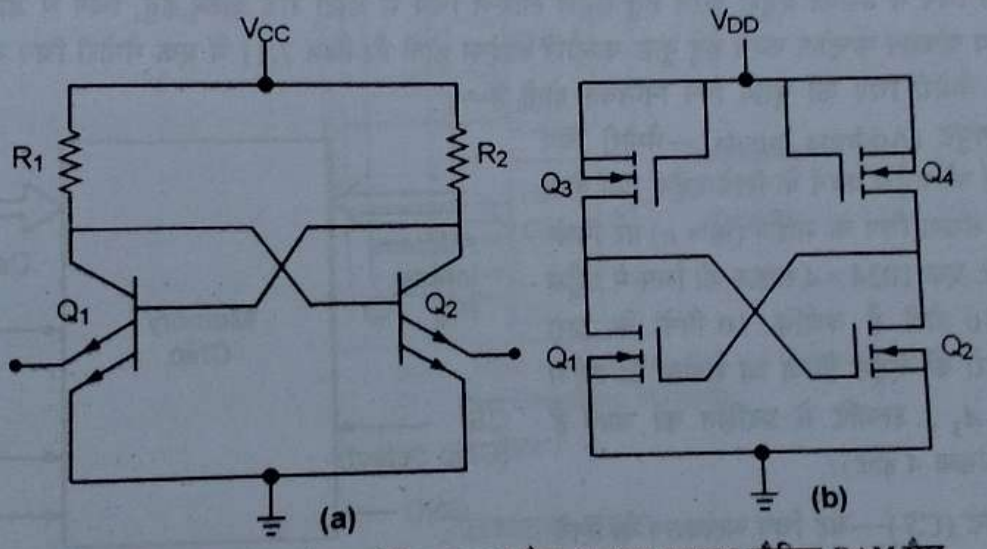
चित्र 7.8-n चैनल CCD की सरलतम संरचना

इस प्रक्रिया की पुनरावृत्ति (repetition) से चार्ज एक कैपेसिटर से दूसरे कैपेसिटर को ट्रांसफर होता जायेगा, इसलिये इस युक्ति को charge couple device (CCD) कहा जाता है। CCD से अत्यन्त high density के शिफ्ट रजिस्टर व सीरियल मैमोरीज बनायी जा सकती है। हालांकि डिजिटल सिस्टम्स में सीरियल मैमोरीज का use सीमित है, किन्तु CCD के कई महत्वपूर्ण अनुप्रयोग हैं—जैसे इमेज प्रोसैसिंग (Image processing) में, डिजिटल सिगनल प्रौसैसिंग (digital signal processing or DSP) में, जहाँ CCD का उच्च डैन्सिटी सीरियल स्वभाव एक बहुमूल्य विशेषता है। इमेज प्रौसैसिंग तथा डिजिटल सिगनल प्रौसेसिंग आधुनिक कन्ट्रोल युक्तियों (विशेषकर रोबोटिक्स (robotics)) के महत्त्वपूर्ण क्षेत्र हैं।

§ 7.5. रैन्डम एक्सेस मैमोरी (Random Access Memory or RAM) :

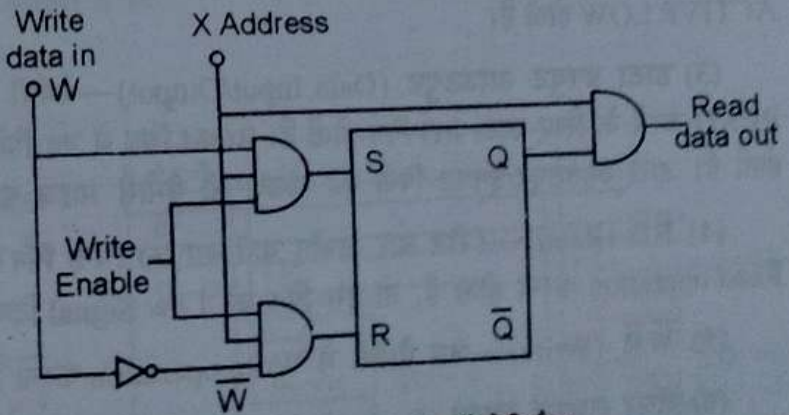
किसी डिजिटल सिस्टम के ऑपरेशन हेतु डाटा को स्टोर करना तथा आवश्यकता पड़ने पर retrieve (read) करना आवश्यक होता है। सेमीकन्डक्टर मैमोरी में स्टोरेज सैलों के आव्यूह (matrices or arrays) होते हैं जिसमें प्रत्येक सैल एक बिट डाटा स्टोर कर सकता है। इस प्रकार की मैमोरी में डाटा randomly मैमोरी में write या read किया जा सकता है। अत: इस मैमोरी को रैन्डम एक्सैस मैमोरी या RAM कहा जाता है। RAM में सभी मैमोरी लोकेशन का एक्सैस टाइम समान होता है। चूँकि इन मैमोरी से डाटा read तथा write किया जा सकता है अत: इनको read write memory (RWM) भी कहा जाता है तथा इस प्रकार यह ROM से भिन्न होती हैं जिनमें डाटा केवल रीड किया जा सकता है। RAM को बनाने हेतु MOS तथा बाइपोलर तकनीक, दोनों का प्रयोग किया जा सकता है। यहाँ यह उल्लेख करना महत्त्वपूर्ण है कि RAM volatile होती है, अर्थात् पॉवर सप्लाई के फेल होने पर RAM में stored सूचना lost हो जाती है।

मूल रम सैल (Basic RAM Cell)—चित्र 7.9 में मूल बाइपोलर स्टेटिक RAM सैल व मूल NMOS स्टैटिक RAM सैल प्रदर्शित है। बाइपोलर सैल में दो बाइपोलर ट्रांजिस्टर व दो प्रतिरोधों का प्रयोग होता है जबकि NMOS सैल में चार N-चैनल MOSFET का प्रयोग होता है। बाइपोलर सैल को अधिक चिप क्षेत्रफल की आवश्यकता होती है क्योंकि इसमें अलग से प्रतिरोध फैब्रीकेट करने पड़ते हैं। MOS सैल में MOSFET ही प्रतिरोध की भाँति व्यवहार करते हैं।



चित्र 7.9-(a) मूल बाइपोलर स्टैटिक RAM सैल (b) मूल NMOS स्टैटिक RAM सैल

RAM ऑपरेशन (RAM Operation)—यह समझने के लिये कि RAM किस प्रकार ऑपरेट करता है, चित्र 7.10 में प्रदर्शित 1-बिट S-R फ्लिप फ्लॉप देखें। इसमें डाटा इनपुट व डाटा आउटपुट लाइन प्रदर्शित हैं। हम देखते हैं कि सैल से डाटा रीड या राइट दोनों के लिये एड्रैस लाइन $X = 1$ होना आवश्यक है। यदि Write करना है तो write enable लाइन भी 1 होनी चाहिये। अब यदि write data input 1 होगा तो $S = 1, R = 0$, अत: $Q = 1$ हो जायेगा अर्थात् सैल में 1 write हो जायेगा। यदि write data input 0 होगा तो $S = 0$, $R = 1$, अत: $Q = 0$ हो जायेगा। डाटा रीड आउट लाइन भी तभी active होगी जब $X = 1$ होगा, अत: जो भी डाटा stored होगा वह read out लाइन पर प्राप्त हो जायेगा।



चित्र 7.10-एक-बिट मैमोरी सैल

मूल RAM संरचना (Basic RAM Organisation)—मान लीजिये कि हम 1024 × 8 RAM बनाना चाहते हैं। इस सिस्टम में 10 एड्रैस लाइन्स, 8 डाटा इनपुट लाइन्स व 8 डाटा आउटपुट लाइन्स की आवश्यकता होगी। इसमें कुल 1024 × 8 = 8192 सैलों की आवश्यकता होगी। इनमें 8 सैलों को एक horizontal line में arrange किया जायेगा तथा यह सभी 8 सैल एक ही एड्रैस लाइन से excite होंगे। इस प्रकार की 1024 लाइन्स होगी, जिसमें से प्रत्येक का भिन्न एड्रैस होगा। एक 10 to 1024 लाइन डिकोडर की सहायता से एड्रैस को डिकोड किया जायेगा। इस प्रकार की एड्रैसिंग one dimensional या लीनियर एड्रैसिंग कहलाती है।

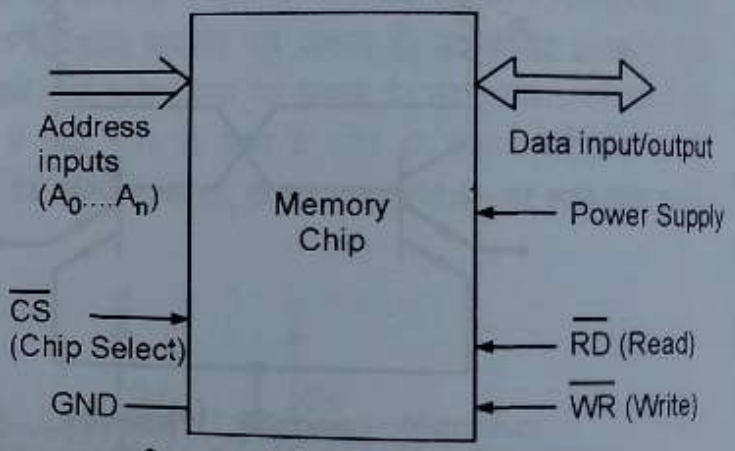
प्रैक्टिस प्रश्न

एक 1024-बिट RAM में 8 बिट के 128 वर्ड स्टोर करने हैं। यदि linear selection use किया जाये तो सिस्टम का ब्लॉक डायग्राम बनाइये। नोट–प्रत्येक 1 बिट सैल को rectangle से प्रदर्शित कीजिये जिसमें तीन टर्मिनल हों- X : address input, W : write input तथा R : read input।

§ 7.6. मैमोरी चिप की पिन संरचना (Pin Structure of a Memory Chip) :

प्रत्येक मैमोरी चिप में उसको एड्रैस करने हेतु एड्रैस लाइन्स चिप से डाटा रीड करने हेतु, चिप में डाटा write करने हेतु डाटा लाइन व अन्य फंक्शन कन्ट्रोल करने हेतु कुछ कन्ट्रोल लाइन्स होती है। चित्र 7.11 में एक मैमोरी चिप का लॉजिक डायग्राम प्रदर्शित है। किसी मैमोरी चिप की मुख्य पिनें निम्नवत् होती हैं—

(1) एड्रैस इनपुट (Address Inputs)—मैमोरी चिप में विभिन्न रजिस्टरों को एड्रैस करने के लिये एड्रैस पिनें होती हैं। एड्रैस पिनों की संख्या चिप के साइज़ ($m \times n$) पर निर्भर करती है। उदाहरणत: एक 1024 × 4 साइज़ की चिप में एड्रैस पिनों की संख्या 10 होती है क्योंकि 10 पिनों के द्वारा $2^{10} = 1024$ रजिस्टरों को एड्रैस किया जा सकता है। एड्रैस लाइन A_0, A_1, A_2, A_3 ... इत्यादि से प्रदर्शित की जाती हैं (डाटा लाइन्स की संख्या 4 होगी)।



चित्र 7.11-मैमोरी चिप का लॉजिक डायग्राम

(2) चिप सेलेक्ट (CS)—यह चिप सलेक्शन के लिये प्रयुक्त की जाती है। चिप सलेक्शन के लिए CS लाइन ACTIVE LOW होती है।

(3) डाटा इनपुट आउटपुट (Data Input/Output)—मैमोरी चिप में WRITE करने के लिये अथवा मैमोरी चिप से READ करने के लिए डाटा I/O पिन होती हैं। ROM चिप में यह पिनें केवल READ फंक्शन अर्थात् आउटपुट के लिये प्रयुक्त होती हैं। डाटा आउटपुट/इनपुट पिनों की संख्या भी मैमोरी साइज पर निर्भर करती है।

(4) $\overline{RD}$ (Read)—(रीड बार अर्थात् यहाँ बार का चिन्ह पिन का active low होना indicate करता है) जब मैमोरी से Read operation करना होता है, तो इस पिन को Low Signal दिया जाता है।

(5) $\overline{WR}$ (Write)—जब मैमोरी में Write operation करना होता है, तो पिन को Low दिया जाता है।

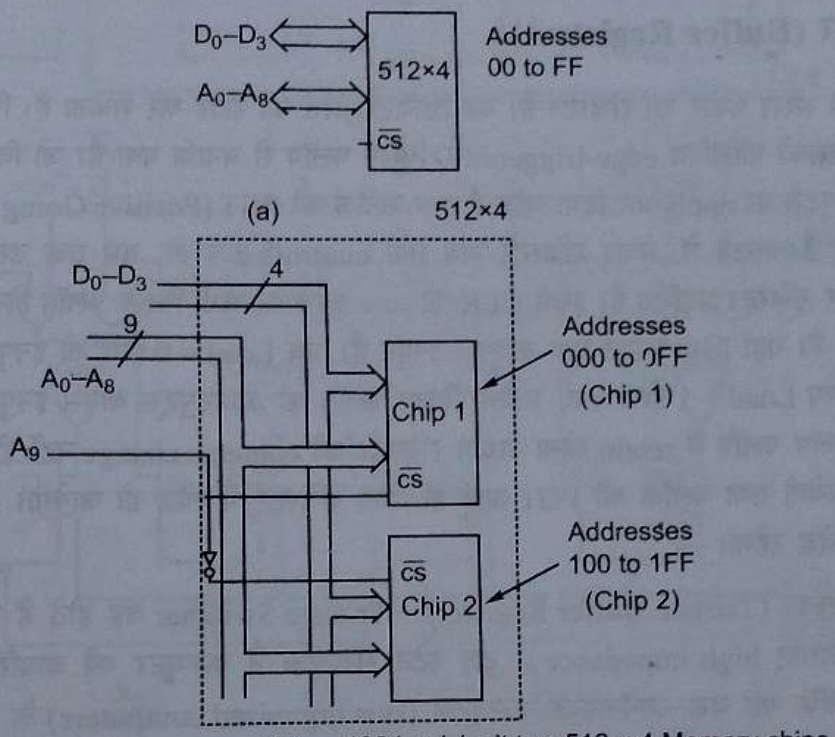
(6) पॉवर सप्लाई लाइन (Power Supply Input)—सभी मैमोरी चिप में सप्लाई के लिये एक या अधिक पिन होती हैं। इस पिन को V_{DD} अथवा V_{CC} द्वारा प्रदर्शित किया जाता है। TTL लॉजिक पर आधारित चिप में +5 V, DC सप्लाई की आवश्यकता होती है।

(7) ग्राउन्ड (GND)—प्रत्येक चिप में एक या अधिक पिन ग्राउन्ड (ground) की जाती हैं।

§ 7.7. Extension of Memory in Terms of Depth :

मान लीजिये आपके पास एक memory chip है जिसका size है 512 × 4. इसका अर्थ यह हुआ कि इसकी 9 address lines तथा 4 data lines हैं। यदि आप इस प्रकार की कई इस चिप्स को ऐसे connect करते हैं जिससे word size तो same रहे किन्तु Address बढ़ जाये तो इसे Extension of memory इन terms of depth कहा जायेगा।

512 × 4 की दो चिप्स use करने पर memory क्षमता 1024 ×4 हो जायेगी, चार चिप्स use करने पर 2096 ×4 इत्यादि।



(a) 1024 × 4 (unit two 512 × 4 Memory chips i.e. Extension of depth

चित्र 7.12

अभ्यास प्रश्न

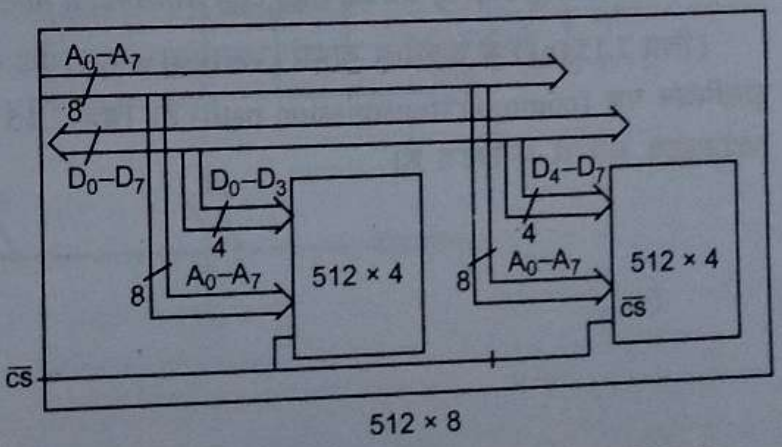
यदि 512 × 4 चिप्स से 2048 × 4 memory बनानी है तो

(i) सभी chips की data bus common करें।

(ii) तक सभी Address line common करें।

(iii) 2 Additional address lines को 2 × 4 decoder द्वारा चार lines में बदलें व प्रत्येक के Chip select () को एक एक लाइन दें। उक्त का logic परिपथ बनायें।

Extension of memory in terms of word size— उदाहरणत: यदि 512 × 4 memory chips से 512 × 8 memory बनानी है तो दो 512 × 4 chips लीजिये। दोनों की Address bus common कर दीजिये। चार data lines पहली चिप को दीजिये। चार data lines दूसरी चिप को दीजिये। दोनों की CS पिन common कर दीजिये।



चित्र 7.13–Extension of memory in terms of Depth

विचार प्रश्न

यदि 2K × 4 memory की चिप्स use करके निम्न memory size प्राप्त करना हो तो कितनी चिप्स की आवश्यकता होगी—

(i) 2K × 8 (ii) 2K × 12 (iii) 4K × 4 (iv) 8K × 4
(v) 8K × 8 (vi) 12K × 8 (vii) 10K × 10 (viii) 5K × 5

§ 7.8. बफर रजिस्टर (Buffer Registers) :

बफर रजिस्टर सबसे सरल प्रकार का रजिस्टर है। यह डिजिटल वर्ड को स्टोर कर सकता है। चित्र 7.14 में बफर रजिस्टर का परिपथ प्रदर्शित है, जिसको पॉज़िटिव edge-triggered *D* फ्लिप फ्लॉप से बनाया गया है। जो बिट्स फ्लिप फ्लॉप पर लोड करनी होती हैं, उन्हें *X*-इनपुट्स पर apply कर दिया जाता है तथा क्लॉक की PGT (Positive Going Trigger) पर बिट्स फ्लिप फ्लॉप पर लोड हो जाती हैं।वास्तव में, बफर रजिस्टर जब तक controlled न हो, तब तक उसका कोई उपयोग नहीं है। चित्र 8.2 में कन्ट्रोल्ड बफर रजिस्टर प्रदर्शित है। इसमें CLR को low करने पर सभी फ्लिप फ्लॉप क्लियर हो जाती है। तत्पश्चात् क्लियर को high कर देते हैं। यहाँ load इनपुट एक कन्ट्रोल इनपुट है। जब Load = 0 होगा तो इनपुट बिट्स फ्लिप फ्लॉप तक नहीं पहुँच पायेंगी। इस समय Load = 1 होगा अत: प्रत्येक फ्लिप फ्लॉप की आउटपुट्स वापस इनपुट की फीडबैक हो जायेंगी। अर्थात पिछला डाटा ही फ्लिप फ्लॉप में retain रहेगा अर्थात रजिस्टर की contents change नहीं होंगी। LOAD = 1 करने पर *X* इनपुट्स *D* पर पहुँच जायेगी तथा क्लॉक की PGT आते ही डाटा रजिस्टर में लोड हो जायेगा। अब लोड के low होने पर भी डाटा रजिस्टर में ही लोड रहेगा।

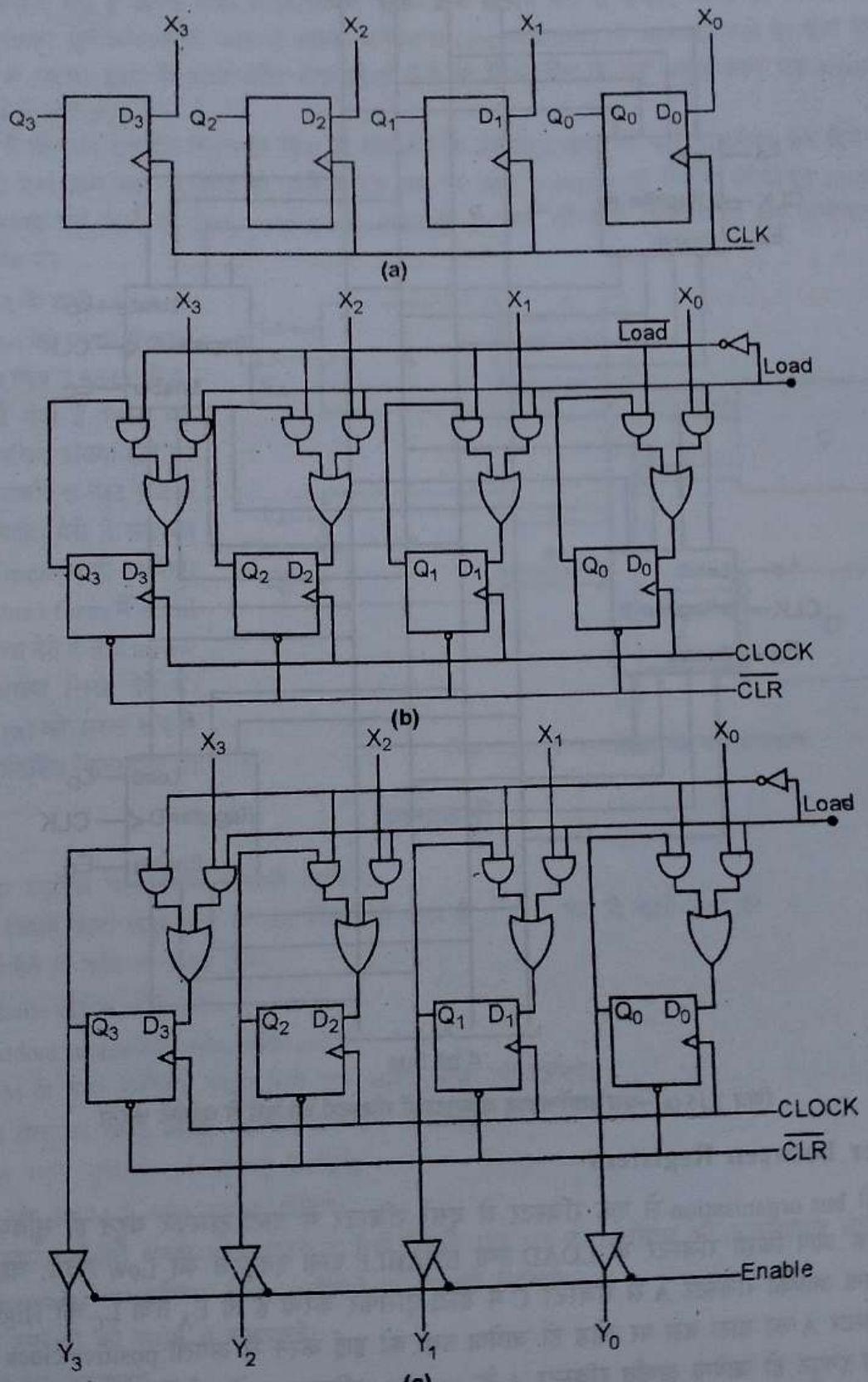
ट्राई-स्टेट बफर रजिस्टर (Tristate Buffer Register)—Tristate Switcher वह होते हैं जिसमें तीन स्टेट्स होते हैं, high, low तथा Hi-Z (अर्थात् high impedance)। ट्राई स्टेट रजिस्टर्स ने कम्प्यूटर की वायरिंग व डिजाइन को अत्यन्त simplify कर दिया है क्योंकि यह बस-आर्गेनाइज्ड कम्प्यूटर्स (Bus organised computers) के लिये आदर्श है।

थ्री-स्टेट स्विचों का मुख्य अनुप्रयोग रजिस्टर्स के टू-स्टेट आउटपुट को थ्री-स्टेट आउटपुट में कनवर्ट करना होता है। चित्र 7.14 (c) को थ्री-स्टेट रजिस्टर इसलिये कहा जाता है क्योंकि आउटपुट लाइन्स पर थ्री-स्टेट स्विच लगे हैं। जब Enable = 0 होगा, तो आउटपुट फ्लोट करेगी (अर्थात् Hi-Z अवस्था में रहेगी। जब Enable = 1 होगा तो आउटपुट लाइन्स पर प्राप्त हो जायेंगी।

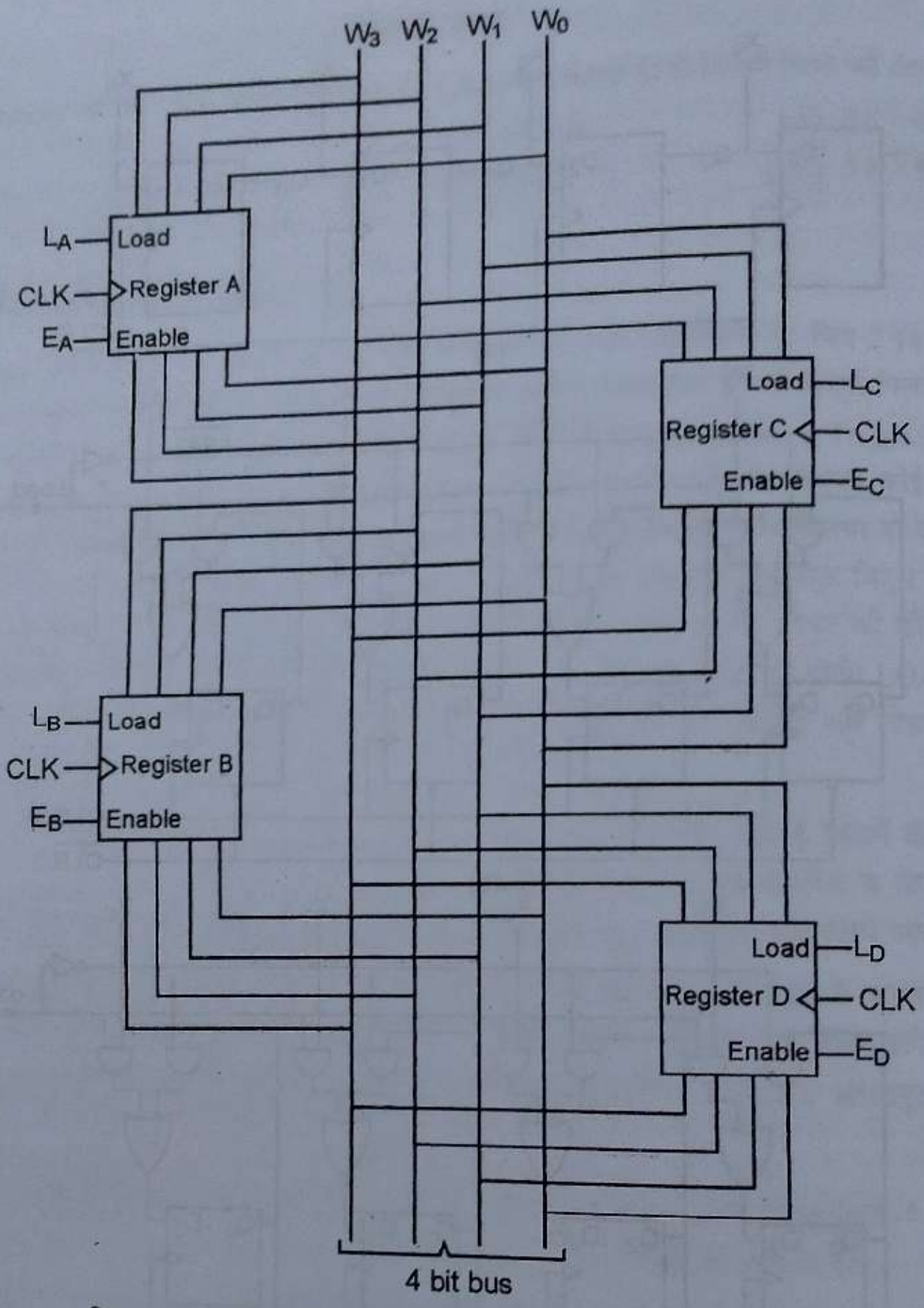
बस ऑर्गेनाइज्ड कम्प्यूटर्स में थ्री-स्टेट रजिस्टर्स का प्रयोग (Use of Three-State Registers in Bus Organized Computers)—बस (bus) वार्य का ग्रुप होता है जिसमें बाइनरी वर्ड ट्रांसमिट किया जा सकता है।

"A bus is a group of wires, that can transmit a binary word."

(चित्र 7.15(a)) में ऊर्ध्वाधर वायर्स (Vertical wires) एक 4-लाइन बस है जो कई थ्री-स्टेट रजिस्टर्स के लिये एक कॉमन ट्रांसमिशन पथ (common transmission path) है। चित्र 7.15 (a) में ध्यान से देखें कि प्रत्येक रजिस्टर के इनपुट्स तथा आउटपुट्स बस से कनैक्टेड है।



चित्र 7.14—(a) बफर रजिस्टर (b) कन्ट्रोल्ड बफर रजिस्टर (c) थ्री-स्टेट बफर रजिस्टर



चित्र 7.15 (a)–बस आर्गेनाइज्ड कम्प्यूटर में रजिस्टर्स को बस से कनैक्ट करना

Data Transfer between Registers

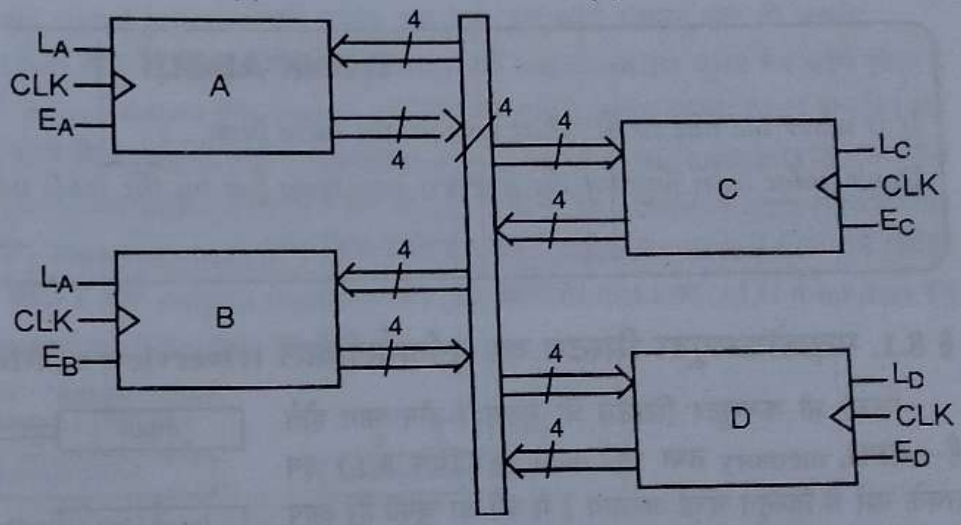
इस प्रकार की bus organisation से एक रजिस्टर से दूसरे रजिस्टर में डाटा ट्रांसफर बहुत ही सुविधाजनक ढंग से हो सकता है। जब तक आप किसी रजिस्टर के LOAD तथा ENABLE दोनों इनपुट्स को Low रखेंगे, वह रजिस्टर बस से isolated रहेगा। माना आपको रजिस्टर A से रजिस्टर C में डाटा ट्रांसफर करना है तो E_A तथा L_C को High कर दीजिये। को high करने से रजिस्टर A का डाटा बस पर लोड हो जायेगा तथा को हाई करने से अगली positive clock edge पर बस का डाटा रजिस्टर C में इनपुट हो जायेगा अर्थात् रजिस्टर A के contents रजिस्टर C में पहुँच जायेंगे। इसी प्रकार B रजिस्टर का डाटा D रजिस्टर में ट्रांसफर करने हेतु E_B व L_D को हाई करना पड़ेगा।

आप बस आरगेंनाइज्ड कम्प्यूटर में डाटा ट्रांसफर की सरलता का अंदाजा इसी बात से लगा सकते हैं कि प्रत्येक रजिस्टर को अलग-अलग एक दूसरे से कनैक्ट करने की आवश्यकता नहीं है बल्कि सभी रजिस्टर्स को केवल एक कॉमन बस से कनैक्ट

करने की आवश्यकता नहीं है बल्कि सभी रजिस्टर्स को केवल एक कॉमन बस से कनैक्ट करने की आवश्यकता है। इससे न केवल डाटा ट्रांसफर सुविधाजनक हो जाता है बल्कि कनैक्शन्स (connections) भी सरल हो जाते हैं। ऐसा केवल ट्राइस्टेट स्विच की वजह से सम्भव हुआ है। *(यह ठीक ऐसा ही है जैसे कि किसी देश के कई प्रमुख शहर एक common राजमार्ग द्वारा connect किये जाते हैं)।*

उल्लेखनीय है कि यदि ट्राइस्टेट स्विच के बिना ही रजिस्टर्स के आउटपुट डायरेक्ट बस से कनैक्ट कर दिये जाते तो सभी रजिस्टर्स के डाटा एक साथ बस पर लोड हो जाते अर्थात् बस पर डाटा contents की स्थिति अनिश्चित (unpredictable) हो जाती। यह समस्या बस कन्टैन्शन (bus contention) कहलाती है। अत: थ्री-स्टेट रजिस्टर्स से बस कन्टैन्शन की समस्या का भी निदान होता है।

माइक्रोप्रोसैसर के सभी registers एक common bus के माध्यम से डाटा transfer करते हैं। चित्र 7.15 (a) में 4-बिट बस दिखायी गयी है किन्तु यह idea बिट्स की अधिक संख्या होने पर भी लागू होता है अर्थात् 8-बिट बस में 8 वायर्स होंगी इत्यादि। ऐसे में बस का बनाना मुश्किल (messy) हो जायेगा। ऐसे में बस को short form में solid arrow line से दिखा देते हैं तथा arrow पर बिट्स की संख्या लिख देते हैं। अत:, चित्र 7.15 (a) को सरल रूप में चित्र 7.15 (b) में प्रदर्शित किया गया है।



चित्र 7.15 (b)–सरलीकृत का बस डायग्राम

प्रश्नावली

1. अर्ध-चालक स्मृतियों पर संक्षिप्त टिप्पणी लिखिये।
2. वह मैमोरी जिसमें डाटा पावर जाने के बाद विनष्ट हो जाता है ..... नाम से जानी जाती है।
3. (i) निम्न मैमोरी में कौन-से अन्तर हैं—
   (a) Volatile और Non Volatile
   (b) Random access और Sequential access
   (ii) EPROM के कार्य लिखिये। इसके लिये एक आदर्श नम्बर भी दीजिये।
4. मूल RAM सैल का खण्ड आरेख खींचिये तथा इसे समझाइये।
5. अर्ध-चालक स्मृति युक्तियों पर टिप्पणी लिखिये।
6. (a) RAM तथा ROM में अन्तर स्पष्ट कीजिये।
   (b) 2048 बाइट्स मैमोरी क्षमता प्रदान करने के लिये कितने 128 × 8 RAM चिप्स की आवश्यकता होती है?
7. गतिकी (Dynamic) तथा स्थैतिक (Static) मैमोरी पर टिप्पणी लिखिये।
8. अर्द्धचालक स्मृतियों को संक्षेप में समझाइये।
9. निम्नलिखित को समझाइये—
   (i) Volatile and Non-volatile memories.
   (ii) Random Access and Sequential Access memories.
10. अर्द्धचालक मैमोरीज़ क्या होती हैं? उदाहरण सहित समझाइये।

8 Chapter

माइक्रोप्रोसैसर्स (MICROPROCESSORS)

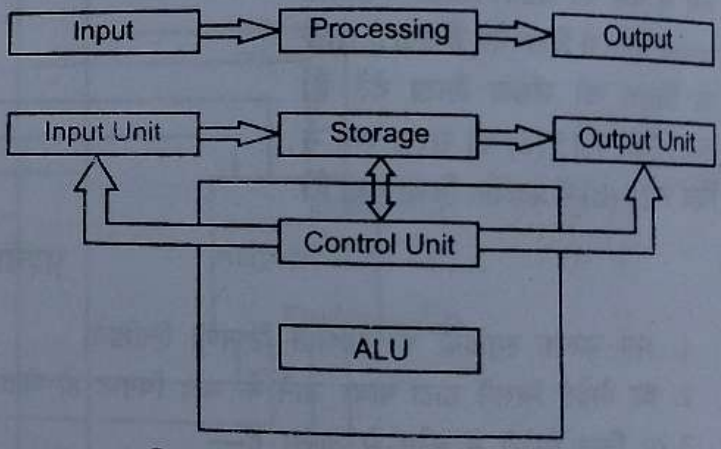
THINK ABOUT IT

It is never too late to be what you might have been. — George Eliot

Don't judge each day by the harvest you reap but by the seeds you plant. — Robert Louis Stevenson

§ 8.1. माइक्रोकम्प्यूटर सिस्टम का पुर्नअवलोकल (Overview of Microcomputer System)

किसी भी कम्प्यूटर सिस्टम के मुख्यत: तीन भाग होते हैं—CPU, memory तथा I/O devices (चित्र 8.1) इन सबके बारे में विस्तृत चर्चा अध्याय 1 में की जा चुकी है। आप यह भी जानते होंगे कि I/O devices के रूप में keyboards, floppy drive (FDD), hard disk drive (HDD), Tape drive, VDU, Printer, Plotter इत्यादि का use किया जाता है। आप सबको यह भी भली भाँति पता होगा कि CPU किसी भी कम्प्यूटर सिस्टम का दिमाग होता है तथा कम्प्यूटर द्वारा किये जाने वाले सभी कार्य CPU द्वारा ही नियन्त्रित होते हैं।



चित्र 8.1—Computer Block diagram

"यदि CPU को किसी IC पर Integrate कर दिया जाता है, तो ऐसा CPU माइक्रोप्रोसेसर (Microprocessor या μP) कहलाता है।"

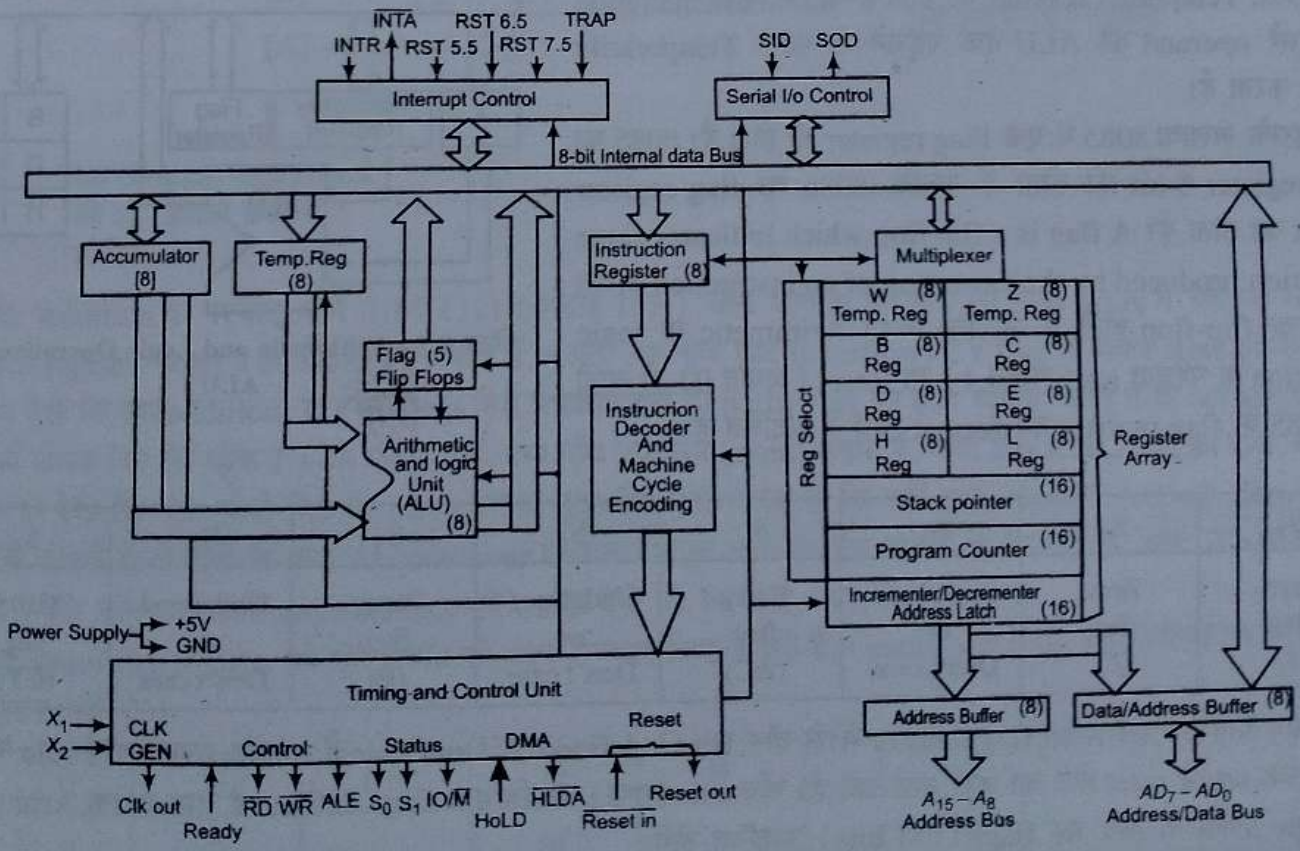
"A microprocessor is a multi purpose programmable logic device that reads binary instruction (called program) from memory, accepts binary data as input and processes data according to these instructions and provides result as output."

अत: CPU को एक छोटी सी चिप में एकीकृत करने की इस तकनीकी के बाद कम्प्यूटर का विकास बहुत तेजी से हुआ। 1980 के दशक में INTEL नामक कम्पनी ने 8080 नामक माइक्रोप्रोसैसर develope किया (यहाँ 8080 इस माइक्रोप्रोसैसर चिप IC का नम्बर है) तथा फिर इसमें कुछ सुधार करके 8085 बनाया गया। इसके बाद तकनीकी के विकास के साथ-साथ 8086, 80286, 80386, 80486, Pentuum, Pentium IV इत्यादि माइक्रोप्रोसैसर विकसित किये गये। वास्तव में, आधुनिक कम्प्यूटर्स में use किये जाने वाले चिप्स तकनीक व कार्यशैली में 8085 से काफी आगे बढ़ चुके हैं किन्तु फिर भी माइक्रोप्रोसैसर की पढ़ाई की शुरूआत 8085 से इसलिये कराई जाती है क्योंकि यह सबसे Basic (तथा सरल) माइक्रोप्रोसैसर है तथा आगे के सभी माइक्रोप्रोसैर्स इसी को आधार मानते हुये विकसित किये गये। अत: एक बार 8085 का अध्ययन कर लेने के पश्चात् आगे के माइक्रोप्रोसेर्स का अध्ययन काफी आसान हो जाता है।

§ 8.2. 8085 Microprocessor

आइये, जानने की कोशिश करें कि एक माइक्रोप्रोसेसर के अन्दर क्या-क्या होता है, अर्थात उसकी संरचना (Architecture) कैसा होता है चित्र 8.2 में 8085 माइक्रोप्रोसेसर का architecture प्रदर्शित है। माइक्रोप्रोसेसर में एक टाइमिंग तथा कन्ट्रोल यूनिट (Timing and control unit) होती है जो कि computer द्वारा किये जाने वाले operations को कन्ट्रोल करने हेतु कई प्रकार के control signals generate करती है। यह memory में डाटा के flow को control करती है तथा I/O devices को भी control करती है। अत:, control unit की स्थिति किसी company के chairman की भाँति है जिसके आदेशों के उपरान्त ही उस company के कार्य सम्पन्न होते हैं। इसी प्रकार, memory तथा I/O devices से data (bus के जरिये) read करना, write करना, program को byte by byte memory से CPU तक bus के द्वारा लाना, उसका executions, result का storage, इन सभी कार्यों के लिये जब तक control unit signal generate नहीं करता, तब तक यह कार्य सम्पन्न नहीं हो सकते।

माइक्रोप्रोसेसर का दूसरा important part होता है ALU यानि Arithmetic and logic unit इस यूनिट का कार्य होता है गणनायें करना। Arithmetic गणनायें जैसे Add, Subtract तथा logical गणनायें जैसे AND, OR, XOR, NOT इत्यादि। यह तो आपने Digital Electronic may पढ़ा होगा कि computer द्वारा Arithmetic operations भी logic circuits द्वारा ही किये जाते हैं (जैसे half adders, full adders, full subtractors इत्यादि)। अत: ALU में सभी Arithmetic व logic operations (जिनके लिये कोई ALU design होता है) logical circuits द्वारा किये जाते हैं। अत:, control unit द्वारा दिये गये निर्देशों के अनुसार ALU को binary input दी जाती हैं तथा outputs (results) प्राप्त की जाती हैं। 8085 का ALU 8-bit data हेतु designed होता है, अत: इसके सभी operations 8-bit data पर होते हैं।



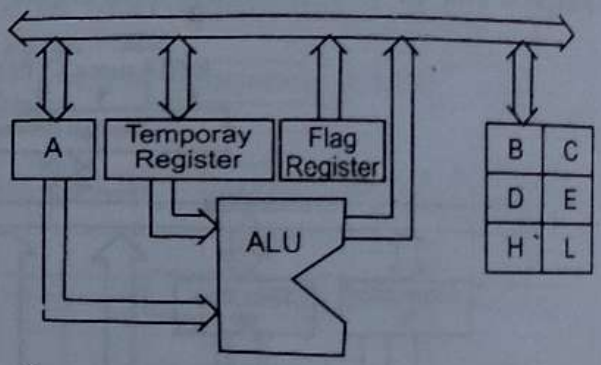
चित्र 8.2—8085 Architecture

माइक्रोप्रोसेसर में तथा ALU के अलावा Accumulators, General Purpose registers तथा कुछ special purpose registers भी होते हैं। Registers का कार्य binary 8-data को store करना होता है। For example—यदि कोई register 8-bit का है तो इसका मतलब यह हुआ कि वह 8 bits (one byte binary word) store कर सकता है। आप पूछेंगे कि CPU के अन्दर इन registers कि क्या आवश्यकता है, जबकि data store करने के लिये तो कम्प्यूटर में पर्याप्त memory होती है।

इस प्रश्न का उत्तर मैं आपको एक उदाहरण द्वारा देता हूँ। आपके घर की kitchen का जो राशन आता है आप वह सब store-room में store करते हैं लेकिन इसके बावजूद kitchen में भी कुछ डिब्बे रखने जरूरी होते हैं। यदि आपको चाय बनानी है तो आप हर बार दो चम्मच चीनी लेने हेतु store room में तो नहीं जायेंगे न। इसी प्रकार microprocessor में भी कुछ general purpose registers होते हैं तथा जिस data पर operations होने होते हैं, memory से लाकर उन्हें इन registers में store कर लिया जाता है। इससे microprocessor की speed fast हो जाती है क्योंकि मैमोरी से डाटा लाने ले जाने के लिये memory read/write cycles करनी पड़ती है तथा यह time consuming होती है। यदि हम 8085 का उदाहरण लें तो इसमें B, C, D, E, H तथा L नामक General purpose registers होते हैं तथा यह सभी 8-bit registers होते हैं। अत:, यह 8-bit data को store कर सकते हैं। यदि 16-bit data पर operation करना हो तो इसके लिये 8085 में registers pairs का प्रावधान है, अर्थात BC, DE, तथा HL को जोड़े में (in pairs) use किया जा सकता है। इसके अलावा 8085 में Accumulator register (या Register A) भी होता है। इस register को 8085 में विशेष दर्जा प्राप्त है। जब भी ALU द्वारा कोई arithmetic operation या logic operation किया जाना होता है तो उस operation हेतु आवश्यक operands में से एक हमेशा A register में होता है तथा A register से ALU को दिया जाता है। यदि operation हेतु दो operands की आवश्यकता है तो दूसरा operands यदि किसी अन्य register में है तो वहाँ से वह operand internal bus द्वारा (bus की चर्चा आगे की जायेगी) Temporary register पर पहुँचता है तथा Temporary register (इसकी चर्चा भी आगे की जायेगी) से ALU में पहुँचता है। अत:, यदि किसी operation के दो operands हैं तो ALU में दिये जाने से पहले उनमें से एक A register में होगा तथा दूसरा Temporary register में (चित्र 8.3) देखें।

अत: Temporary register वह होता है जो कि किसी operation के दूसरे operand को ALU तक पहुँचने से पहले Temporarily store करता है।

इसके अलावा 8085 में एक Flag register भी होता है। 8085 का flag register 8-bit का होता है जबकि 8086 का flag register 16-bit का होता है। A flag is a flip flop which indicates some condition produced by the execution of an instruction अर्थात flag एक flip-flop होती है जो कि किसी Arithmetic या logic operation के पश्चात set (अर्थात 1) या Reset (अर्थात 0) हो जाती है। 8085 के flag register का format चित्र में प्रदर्शित है।



चित्र 8.3—Arithmetic and Logic Operations in ALU

S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
Sign flag (S)	Zero flag (Z)	Undefined or Don't care	Carry flag (AC)	Undefined or Don't care	Parity flag (P)	Undefined or Don't care	Carry flag (CY)

आप देखें कि इस 8-bit flag register में से तीन Bit Undefined या Dont care हैं, अर्थात उनका कोई role नहीं है, तथा उनके set या reset होने का कोई अर्थ नहीं है। पाँच bits (तथा) पर विभिन्न flags defined हैं तथा प्रत्येक Arithmetic या logic गणना के बाद यह flags (तथा bits) प्रभावित होंगे।

(i) S_0 पर carry flag का status जायेगा। यदि किसी Arithmetic addition के उपरान्त Accumulator register overflow (अर्थात पर carry उत्पन्न हुआ) तो CY = 1 हो जायेगा तथा underflow (अर्थात पर carry उत्पन्न नहीं हुआ) तो CY = 0 हो जायेगा।

(ii) S_2 पर Parity flag का status जायेगा। यदि किसी गणना के परिणाम के बाद accumulator में Bits की संख्या odd आई तो P = 0 तथा even आई तो P = 1।

(iii) S_4 पर Auxilliary carry flag का status जायेगा। यदि चौथे बिट (अर्थात पहले निबल) पर carry आता है तो AC flag set हो जाता है।

(iv) S_6 पर zero flag का status जाता है। यदि किसी गणना के पश्चात Accumular की सभी बिट्स शून्य हो जायें, तो zero flag set (अर्थात 1) हो जाता है, अन्यथा zero flag reset (अर्थात zero) रहता है।

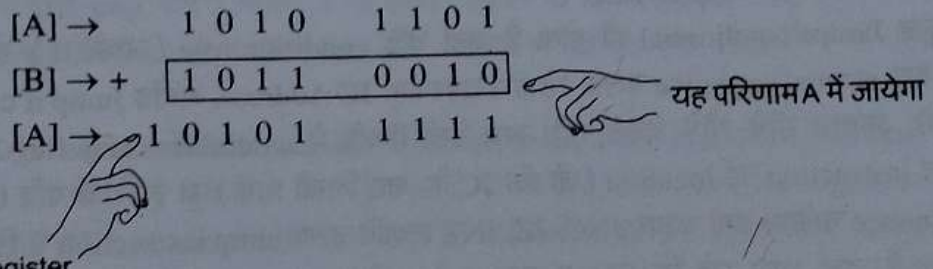
(v) S_7 पर sign flag का status जाता है। चूंकि signed notation की स्थिति में leftmost bit (MSB) sign को इंगित करती है, अत: Accumulator की MSB का मान 1 होने पर sign flag set व zero होने पर sign flag reset हो जाता है।

आइये, उदाहरण की सहायता से समझते हैं—

माना A = 1010 1101

B = 1011 0010

अब माना कि ADD B Instruction execute हुआ। ADD B Instruction का मतलब है कि B register की contents को A register के साथ जोड़ो तथा परिणाम A register में डाल दो। अब जब ALU A तथा B को जोड़ेगा तो निम्न परिणाम प्राप्त होंगे—



अत: Addition के पश्चात् A में 0101 1111 (अर्थात 1FH) चला जायेगा। परिणाम का नौवां bit A में नहीं जा सकता (क्योंकि A register केवल 8 bit store कर सकता है)। अत: यह नौवां बिट (overflow) CY flag में जाकर store हो जायेगा। अब आप देखें कि इस additions से दूसरे flags कैसे प्रभावित होते हैं। parity flag का हाल जानना चाहते हैं तो accumulator में 1's की संख्या गिन लें। चूँकि 1's की संख्या 6 है, तथा पाँच एक सम (even) संख्या है, अत: P = 1. AC flag का हाल जानना चाहते हैं तो देखें कि क्या चौथी बिट पर carry उत्पन्न हुआ था। पता चलता है कि नहीं हुआ था, अत: AC = 0. Zero flag की स्थिति जाननी है तो देखें कि क्या Accumulator के सभी बिट्स zero हैं। पता चलता है कि नहीं है, अत: Z = 0. Sign flag की स्थिति जानने के लिये Accumulator की MSB देखें जो कि 0 है, अत: S = 0.

अत: status flag register की पूरी स्थिति को आप निम्नवत् लिख सकते हैं, (undefined flags चाहे 0 लिखें या 1, उससे कोई मतलब नहीं है)।

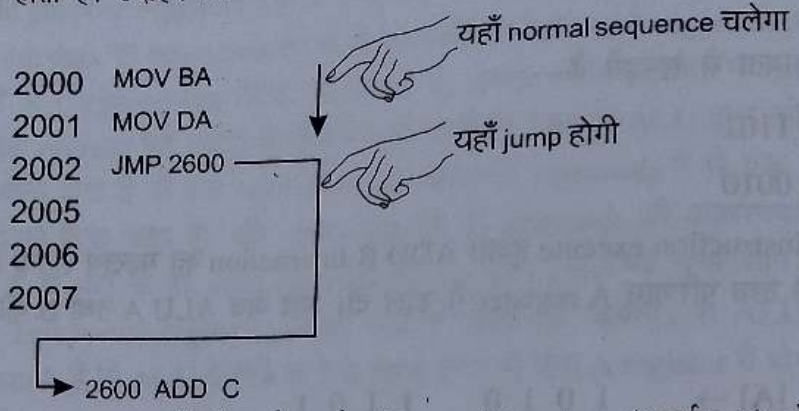
0	0	0	0	0	1	0	1
S	Z		AC		P		CY

अत: flag register की स्थिति—

Flag register = 0000 0101

ध्यान दें कि यहाँ इस विशेष गणना के (example) के बाद flag register की यह स्थिति है, तथा प्रत्येक गणना के उपरान्त स्थिति भिन्न होगी।

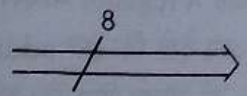
प्रश्न यह उठता है कि flag register का use क्या है। Flag register के विभिन्न flags Branch control instructions के दौरान program को निर्णय लेने में सहायता करते हैं। जैसे कि C में if—else statements होती हैं ऐसे ही Assembly language programming में jump तथा call instruction होती है। जब भी कोई program execute होता है तो सामान्य स्थिति में वह क्रमवार (sequence में) होता है तथा एक instruction execute हो जाने के पश्चात् उससे अगली memory location में stored instruction execute होता है। लेकिन जब भी प्रोग्राम में कोई JUMP statement आती है तो next instruction का execution अगली मैमोरी location से न होकर JUMP के आगे लिखे memory Address पर store पर लिखी गई instruction से होता है। उदाहरणत:



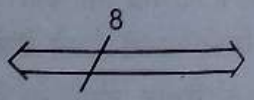
8085 में कुछ Jumps conditional भी होता है जहाँ यदि condition true (अर्थात 1) होगी तो Jump होगा अन्यथा Jump नहीं होगा तथा normal sequence चलता रहेगा उदाहरणत: JC Address, अर्थात Jump if carry अर्थात यदि carry flag 1 है तो जम्प करो, अन्यथा सीधे-सीधे चलते रहो। अब ऐसी स्थिति में assembler carry flag को जाकर चैक करेगा, यदि Cy = 1 तो अगला instruction नई location (जो कि JC के बाद लिखी होगी) से आयेगा। यदि Cy = 0 तो Jump नहीं होगी, तथा normal sequence चलेगा। इसी प्रकार JNC, JZ, JNZ इत्यादि अन्य jump instruction में निर्णय लेने हेतु flag register का use किया जाता है। यहाँ ध्यान रखें कि CY, P, S व Z flag ही decision making में सहायता करते हैं। जबकि AC flag का decision making में कोई role नहीं होता। इसका role BCD addition के समय आता है। इसकी चर्चा आगे की जायेगी।

Buses—A bus is an interconnection of wires to transfer data (and addresses) from one device to others. A set of parallel lines to send address and data to several devices is called a bus.

आपने देखा कि microprocessor को कार्य करते समय data का बार-बार transfer करना पड़ता है (जैसे कि एक register से दूसरे register को, register's से ALU को, ALU से registers को) इत्यादि। इसी प्रकार memory से भी data transfer करना होता है। इन सब के लिये wires का parallel connection होता है जिन्हें बस कहते हैं। उदाहरणत: यदि 8-bit के A register से 8-bit के B register को data भेजना है, तो आपको 8-parallel lines चाहिये, अत: 8-bit bus की जरूरत होगी। चूँकि 8085 के रजिस्टर्स 8-bit data को store तथा transfer करते हैं, अत: इसकी internal bus में 8-lines होती है। Bus को show करते समय 8 lines न show करके एक मोटे तीर द्वारा show किया जाता है (चित्र 8.4 देखें)



(a) 8-line unidirectional bus



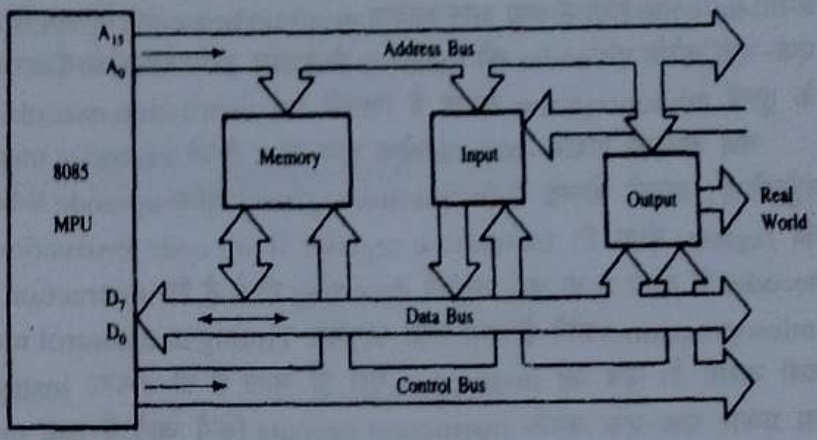
(b) 8-line bidirectional bus

चित्र 8.4—Symbol of a bus

जो bus microprocessor के अंदर data transfer का कार्य करती है, वह उसकी internal bus कहलाती है। जो bus memory तथा अन्य devices को address तथा data carry करती है, वह external bus कहलाती हैं। 8085 का Bus diagram चित्र 8.5 में प्रदर्शित है।

यह तो आप जानते ही होंगे कि 8085 माइक्रोप्रोसैसर 64 KB (अर्थात 64 K × 8) मैमोरी को address कर सकता है। चूँकि $64\ K = 2^{16}$ ($64\ K = 64 \times 1024 = 2^6 \times 2^{10} = 2^{6+10}$) होता है अत: इतनी मैमोरी को address करने हेतु उसे 16 bit का address generate करना होता है तथा इस address को भेजने हेतु इसे 16-bit line (16-bit bus) की जरूरत होती है। 8085 को data भेजने हेतु 8-bit bus की जरूरत होती है। इसके लिये 8085 में दो 8-bit buses होती हैं, जिनको क्रमश:

Address bus तथा Address-data bus कहा जाता है। 8085 से 16-bit address को lower 8-bits Address-data bus पर जाते हैं तथा upper 8-bits Address bus पर जाते हैं। 8-bits डाटा को लाने, तथा ले जाने का कार्य Address-data bus करती है। अत:, आपने देखा कि Address-data bus दोहरी भूमिका निभाती है, पहले तो वह address के lower bits को लेकर जाती है, तथा फिर data को लाने/या ले जाने (अर्थात memory से read या write operation हेतु) का कार्य भी करती है। यह प्रक्रिया Multiplexing कहलाती है। अत:, हम कह सकते हैं कि the Address-data bus is multiplexed. प्रश्न यह उठता है कि फिर यह कैसे पता चल पाता है कि किसी विशेष समय पर Address-data bus पर Address है या data। तो इसके लिये 8085 पर एक विशेष पिन होती है, ALE pin (पिन नम्बर) (ALE अर्थात Address/atch eneble)। इस पिन पर कन्ट्रोल·unit यदि 1 भेजता है तो इसका तात्पर्य है कि Address/Data Bus पर Address भेजा रहा है तथा यदि ALE = 0, तो Address data bus द्वारा data carry किया जा रहा है। अत: ALE पिन को demultiplexing pin भी कहा जाता है।



चित्र 8.5—Bus Diagram of 8085 microprocessor

विचार प्रश्न : Why is Address bus unidirectional while Address-Data bus Bidirectional?

अब तक आपने एक बात और नोट कर चुके होंगे कि चूँकि 8085 के सभी general purpose registers तथा internal/external data buses, तथा ALU सब 8-bit data पर ही कार्य करते हैं, तथा इसी कारणवश 8085 को 8-bit microprocessor कहा जाता है।

प्रोग्राम काउन्टर (Program Counter)—यदि आपने दिल्ली के मैट्रो ट्रेन की यात्रा की होगी तो आपने सुना होगा ट्रेन जैसे ही स्टेशन छोड़ती है, तभी घोषणा होती है कि अगला स्टेशन क्या है? इससे मिलता जुलता काम हो Program Counter का होता है। जब microprocessor यह द्वारा कोई प्रोग्राम को run कराना होता है तो हम इस प्रोग्राम को memory में store कर देते हैं। अब हमें प्रोग्राम का starting address processor को बताना पड़ता है कि इस address से हम प्रोग्राम store किया है, अत: इसे run करो। जैसे ही हम starting address processor को देते हैं, तो यह address Program Counter नामक register में चला जाता है तथा यहाँ से पहला instruction fetch होकर microprocessor में आता है। अब चूँकि अगला instruction अगली memory location से आना होता है तो Program counter की contents auto-increment (अर्थात स्वत: 1 add हो जाना) हो जाती है। संक्षेप में बात यह है कि Program Counter Stores the address of the next instruction to be fetched from the memory. अर्थात Program Counter एक 16-bit का register होता है जो कि उस address को store करता है जहाँ से processor को अगला instruction fetch करके लाना है। एक प्रश्न उठता है कि Program Counter 16-bit का ही क्यों होता है? क्योंकि इसको address store करना होता है तथा 8085 द्वारा address की जाने वाली memory का address 16-bit का होता है।

Instruction Decoder—इंस्ट्रक्शन डिकोडर क्या करता है, यह समझने से पहले यह जानना जरूरी है कि op-code क्या होता है। देखिये, आप जब भी माइक्रोप्रोसैसर में कोई प्रोग्राम लिखते हैं तो Assembly language में लिखते हैं। Assembly language में प्रोग्राम लिखने के लिये अंग्रेजी जैसे शब्द (English like words) होते हैं जिनको निमोनिक्स (Mnemonics) कहा जाता है जैसे MOV B, A, ADD C, SUBB इत्यादि तथा प्रत्येक Mnemonic का कुछ-न-कुछ अर्थ होता है। अत:, आपका जो प्रोग्राम होता है वह Mnemonics के रूप में होता है। लेकिन जब आप इस प्रोग्राम को memory में स्टोर करते हैं, तो Mnemonics के रूप में न करके hexadecimal codes के रूप में store करते हो। अत:, microprocessor के प्रत्येक instruction के संगत एक hexadecimal code होता है जो कि microprocessor की manual में दिया होता है तथा इन

codes को operation code या short में, op-code कहा जाता है। 8085 microprocessor के प्रत्येक instruction का भी 8-bit op-code होता है तथा आप आपना program hex-code के रूप में memory में store कर देते हो तथा microprocessor एक-एक करके इन codes को memory से लाकर इनके संगत कार्य करता रहता है। अत:, microprocessor प्रत्येक op-code के तुल्य microprograms करता है जिससे वह instruction execute हो जाता है।

जब आपका प्रोग्राम execute होना शुरू होता है तो op codes memory से fetch होकर सबसे पहले जिस register में पहुँचते हैं, उसको बोलते हैं Instruction register. चूँकि op-code 8-bit के होते हैं, अत: Instruction register भी 8-bit का register होता है। Instruction register से op code Instruction decoder में जाता है जब op-code Instruction decoder में जाता है तो यहाँ उसकी decoding होती है कि instruction को पूरा करने के लिये कौन कौन से छोटे-छोटे कार्य (microprograms) होने हैं तथा उसी अनुसार Timing and control unit उन कार्यों को सम्पन्न करने हेतु control signals जारी करता है। जब यह instruction पूरा हो जाता है तो अगला instruction मंगाया जाता है (fetch किया जाता है) तथा इस प्रकार एक-एक करके instruction execute किये जाते हैं तथा program complete हो जाता है।

यहाँ एक बात और समझ लें कि यह जरूरी नहीं कि op-code आने पर भी instruction execute हो जाये, क्योंकि 8085 में कुछ 2 bytes या 3 bytes वाले instruction भी होते हैं। यदि दो-बाइट साइज का instruction है तो इसका मतलब यह हुआ कि op-code के साथ-साथ इस instruction में 1 byte की सूचना और जानी है। अत:, Instruction decode op-code देखकर यह पता लगा लेगा कि भाई, इस op-code की बची हुयी सूचना भी लेकर आनी है। एक memory-read cycle और चलाओ, तथा उस सूचना को भी लेकर आओ, और जब यह byte आ जायेगी, तभी instruction execute हो सकेगा।

तो कहने का मतलब यह है कि Instruction decoder द्वारा op-code को decode करने के बाद आगे जो भी होना होगा, control unit उसी हिसाब से signal issue करेगा।

Interrupts—Interrupt का अर्थ होता है व्यवधान या रुकावट। जब कोई माइक्रोप्रोसेसर अपना normal कार्य कर रहा होता है (अर्थात main program कर रहा होता है) तो उसके बीच में यदि उससे कोई अन्य छोटा प्रोग्राम करने हेतु उसके main program का बाधित किया जाता है, तो इसको Interrupt कहते हैं।

यह ठीक ऐसे ही है जैसे कि आप घर में बैठकर कोई पुस्तक पढ़ रहे हैं (main program)। तभी दूध वाले ने घंटी बजायी (Interrupt Signal)। आपने अपनी पुस्तक पर book mark लगाया (Saving data into stack)। दूध लेने गये (serving the subroutine) तथा दूध लेने के बाद पुन: अपनी किताब से वहीं से पढ़ने लगे, जहाँ पर छोड़कर गये थे (return to the main program)।

Interrupt means to allow normal program execution to be interrupted by some external signal or by a special instruction in a program. In response to an interrupt, the microprocessor stops executing its current program and calls a program which services the interrupt. This program which services the interrupt is called Interrupt Service subroutine.

अर्थात main program को किसी सिगनल द्वारा (जो कि microprocessor की किसी विशेष pin पर apply किया जाता है) या फिर किसी instruction द्वारा (जो कि main program के अंदर लिखा होगा) बाधित कर देना Interrupt कहलाता है। यदि Microprocessor की पिन पर signal देकर उसे interrupt किया जाता है तो इसे hardware Interrupt कहते हैं। 8085 में hardware interrupt के उदाहरण हैं—RST 7.5, RST 6.5, RST 5.5 तथा TRAP। यदि किसी instruction द्वारा microprocessor को interrupt किया जाता हैं तो इसे software interrupt कहते हैं। 8083 में software interrupts के उदाहरण हैं—RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, तथा RST 7। जाहिर है कि इन सभी interrupts के op-code होते हैं, जो कि आपको program में रखने पड़ेंगे।

जब भी microprocess interrupt होता है (चाहे वह hardware हो या software) तो वह कुछ समय के लिये अपना normal कार्य suspend करता है तथा अपना data तथा return address memory के एक special भाग (called stack) में save करता है (जिसको कि PUSH operation कहते हैं) तथा फिर interrupt service subroutine नामक program को करने चला जाता है। जब यह कार्य पूरा हो जाता है तो वह पुन: वहीं पर लौट आता है जहाँ पर उसके अपने मुख्य कार्य को

छोड़ा था तथा stack से data retrieve अर्थात पुन: प्राप्त करता है (जिसको POP operation कहते हैं) तथा पुन: अपना main program करना शुरू कर देता है। 8085 के प्रत्येक interrupt हेतु microprocessor को अपना program execution किस memory address पर transfer करना है, यह पूर्व निर्धारित है (तालिका Table 8.1 देखें).

Table 8.1 (a) Vector address for hardware interrupts.

Interrupt	Vector Address
RST 75	60
RST 65	34
RST 55	2C
RST 45 or TRAP	24

Table 8.1 (b) Vector address for software interrupts.

Interrupt	Vector Address
RST 0	0000
RST 1	0008
RST 2	0010
RST 3	0018
RST 4	0020
RST 5	0028
RST 6	0030
RST 7	0038

Vector address की गणना का सूत्र यह है कि आप उस interrupt को 8 से गुणा कर दीजिये उदाहरण $7.5 \times 8 = 60$ तथा 60 का hexadecimal होता है 3C. अत: RST 7.5 का vector address है 003 C तथा RST 7.5 पर signal प्राप्त होने पर program का transfer memory location 003 C पर हो जायेगा।

चूँकि प्रत्येक interrupt signal एक particular memory location (या vector address) पर program का transfer कर देता है, अत: यह vectored interrupts कहलाते हैं।

Interrupts के संबंध में एक अन्य शब्दावली हैं—maskable तथा non-maskable interrupts. यदि interrupt signal प्राप्त होने पर उसे कुछ समय के लिये pending में डाला जा सकता हो (या mask किया जा सकता हो) तो वह Interrupt maskable interrupt कहलाता है। यदि Interrupt सिगनल आने पर उसकी तुरन्त सेवा करना अनिवार्य हो तथा किसी भी तरह से उसे रोक पाना सम्भव न हो (उसकी masking न की जा सकती हो) तो ऐसा interrupt non-maskable interrupt कहलाता है। 8085 में TRAP non-maskable interrupt होता है। बाकी सभी interrupts (RST 7.5, RST 6.5, RST 5.5) maskable होते हैं तथा SIM नामक instruction द्वारा mask किये जा सकते हैं।

SIM Instruction द्वारा Interrupts को mask करना—SIM Instrrupt द्वारा Interrupt को MASK करने हेतु एक पूर्व निर्धारित Bit pattern है जिसको आपको याद करना होगा

D_8	D_7	D_5	D_4	D_3	D_2	D_1	D_0
Serial Output Data	Serial Data Enable	Don't Care	Reset 7.5	Mask Set Enable (MSE)	Mask 7.5	Mask 6.5	Mask 5.5

अब आपको जो भी Interrupt Mask करनी है, उसका bit high कर दीजिये तथा साथ में MSE (Mask Set Enable) high कर दीजिये। Reset 7.5 RST 7.5 को Reset करने का एक अतिरिक्त control है तथा RST 7.5 = 1 करने पर भी RST 7.5 disable हो जायेगी।

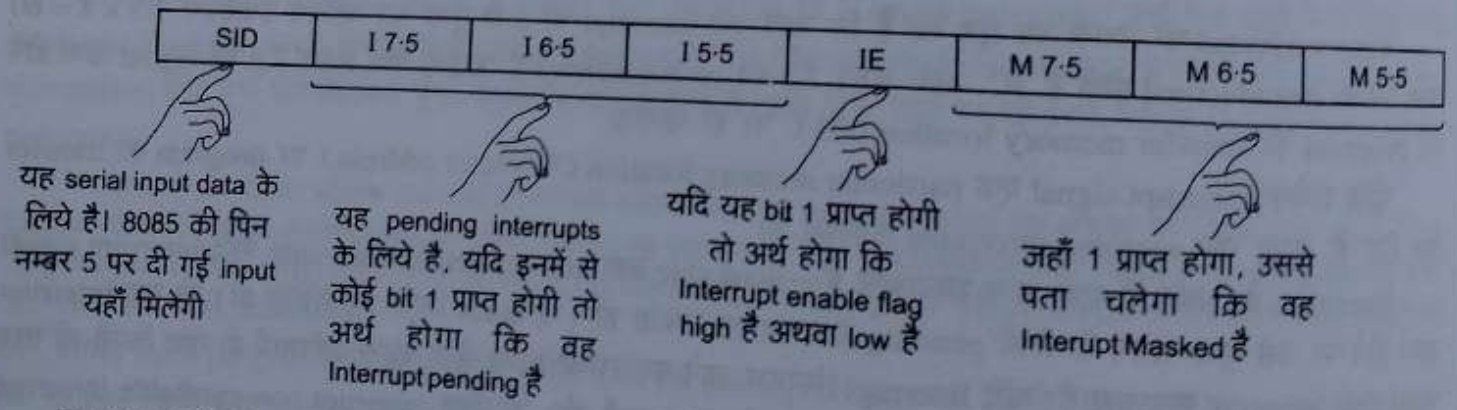
उदाहरणत: आपको RST 5.5 व RST 7.5 को MASK करना है तो निम्नवत् होगा—

तो बिट pattern आय 0000 1101 अर्थात 0DH. अब इसको Accumulator में move कराने हेतु instruction होता है MVI A, 8-bit data अत: आप लिखेंगे—

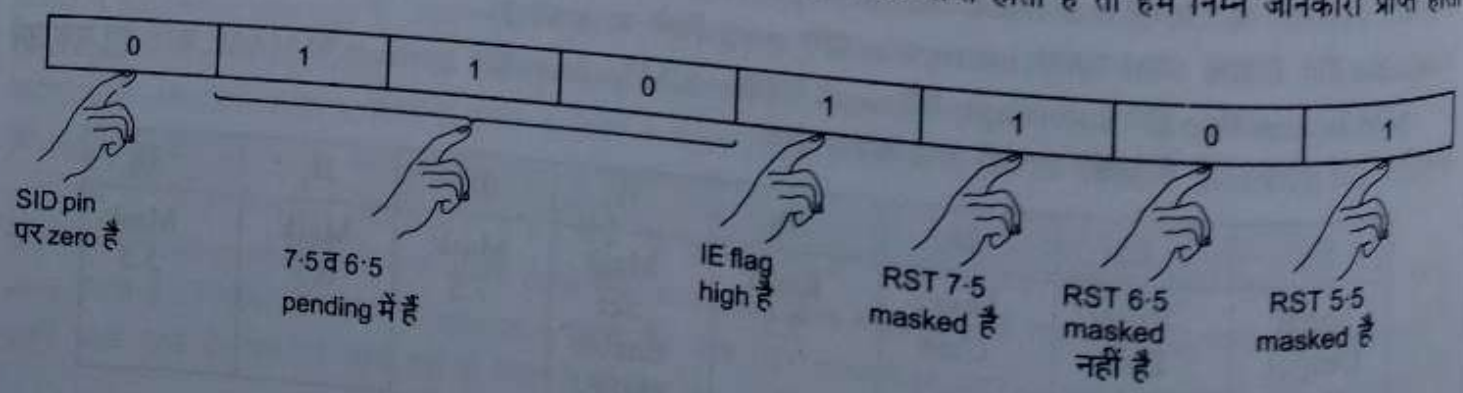
```
MVI A 0DH
SIM
```

RST 7.5 व RST 5.5 Mask हो जायेंगे। ध्यान रहे कि किसी भी Interrupt को Mask करने हेतु MSF को high रखना जरूरी है। यह भी ध्यान रखें कि TRAP को mask नहीं किया जा सकता अर्थात TRAP non-maskable है। SIM के Bit pattern की दो MSB's SOD तथा SDE का कार्य Serial Data Transfer हेतु किया जाता है। यदि कोई data (1 या 0) Serially transmit करना है तो SDE को high करके SIM की MSB पर वह data भेजा जाता है तो वह 8085 की SOD पिन (pin no 4) पर उपलब्ध हो जाता है। Masking करते समय इन दोनों का कोई role नहीं होता अत: SDE को zero ही रखा जाता है।

RIM—RIM एक Instruction है जिसके execution के बाद accumulator पर एक bit pattern उत्पन्न हो जाता है जिससे यह पता लगाया जा सकता है कि कौन-कौन सी Interrupts masked हैं, कौन-कौन सी pending में है, इत्यादि। इसका bit pattern निम्नवत् है।



उदाहरणत: RIM execute करने के बाद आपको Accumulator में 6D प्राप्त होता है तो हमें निम्न जानकारी प्राप्त होती है—

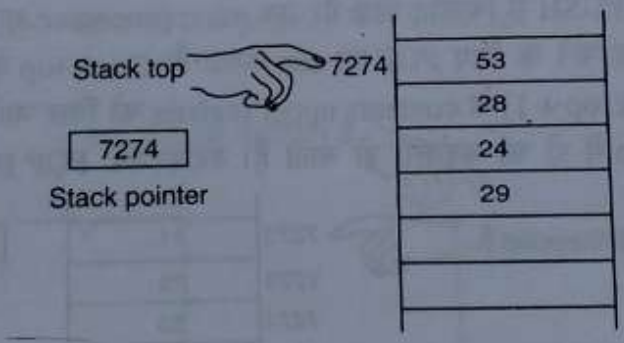


इसके अलावा Interrupts हेतु हो अन्य Instructions भी होती है, EI अर्थात Enable Interrupts (जो कि सभी Interrupts को Enable कर देती है तथा DI (Disable Interrupts) जो कि सभी Interrupts को disable कर देती है (Except TRAP)

STACK—An stack is a segment of memory set aside to store addresses and data while a sub program executes, अर्थात stack मैमोरी का वह क्षेत्र है जो कि किसी subroutine के समय address व data को store करके रखने हेतु सुरक्षित रखा जाता है। जब microprocessor अपना main program कर रहा होता है, तथा यदि बीच में उसे कोई Interrupt signal प्राप्त होता है तो वह अपना data तथा address आदि save करने के बाद ही sub routine को serve करने जाता है। यदि वह ऐसा नहीं करेगा तो subroutine का execution करते समय registers की contents change हो जायेगी व उसका main प्रोग्राम का data lost हो जायेगा। अत:, वह Interrupt Service subroutine को सेवायें देने से पहले अपना डाटा stack में छोड़ जाता है, (जिसको PUSH operation कहा जाता है) तथा subroutine को serve करने के बाद stack से data वापस प्राप्त करता है (retrieve करता है) (जिसको POP operation कहा जाता है)।

Stack के बारे में खास बात यह है कि LIFO mode में कार्य करता है अर्थात Last Input First Out. यानि जो data सबसे last में stack में जायेगा, वह सबसे पहले वापस मिलेगा। ठीक ऐसे ही जैसे कि किसी पार्टी की table पर plates रखी होती है, जो प्लेट सबसे last में रखी गयी होती है (सबसे ऊपर वाली) वही सबसे पहले उठती है। Stack की जो सबसे last filled location होती है, उसको stack top कहा जाता है। Stack top का address store करने के लिये 8085 में एक विशेष 16-bit का register होता है जिसको stack pointer कहा जाता है।

आइये समझें कि किस प्रकार 8085 up stack में PUSH द्वारा data save करता है व POP द्वारा data वापस प्राप्त करता है (नीचे देखें)—



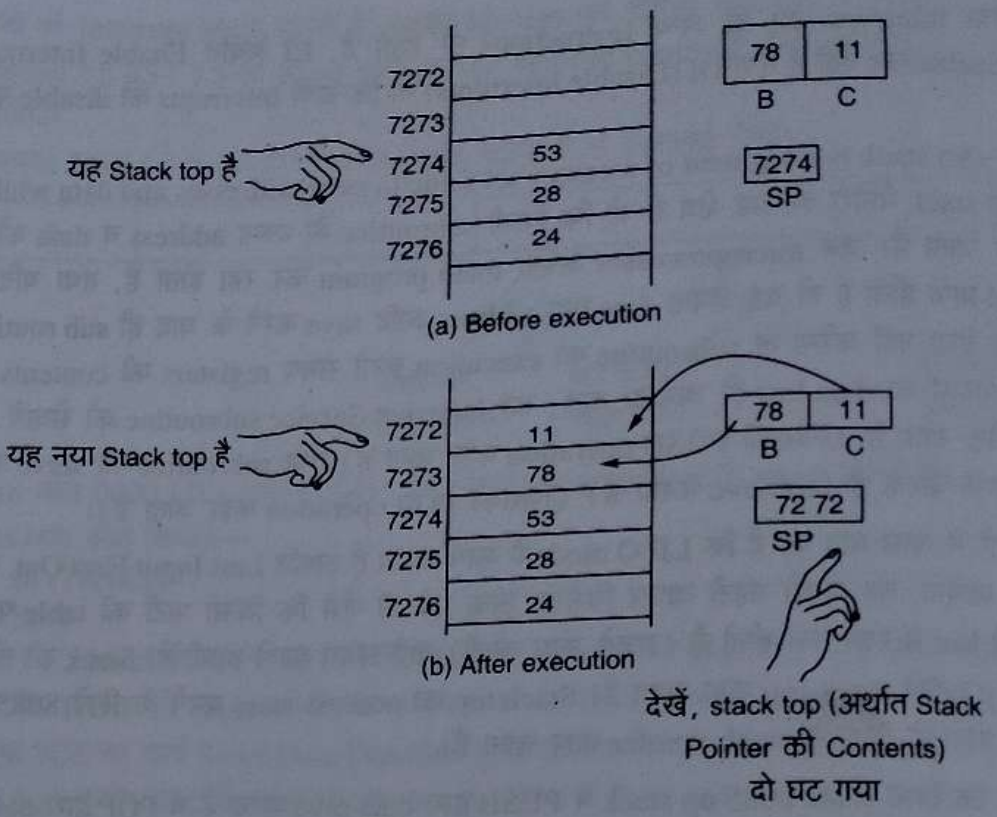
मान लीजिये कि stack की last filled address 7274 है। इसका मतलब कि stack यहाँ तक filled है, अत: stack pointer में 7274 store होगा। अब यदि आपको register B तथा C के contents को stack में store करना है तो उसके लिये 8085 में instruction होता है PUSH rp यहाँ rp पर मतलब register pair है तथा आप rp की जगह B, D, या H लिख सकते हैं। तो यदि आप BC register pair की contents को stack में PUSH करना चाहते हैं तो आपको instruction लिखना होगा।

PUSH B

अब यदि B = 78 तथा C = 11 (नोट : यह सब assumed data है)

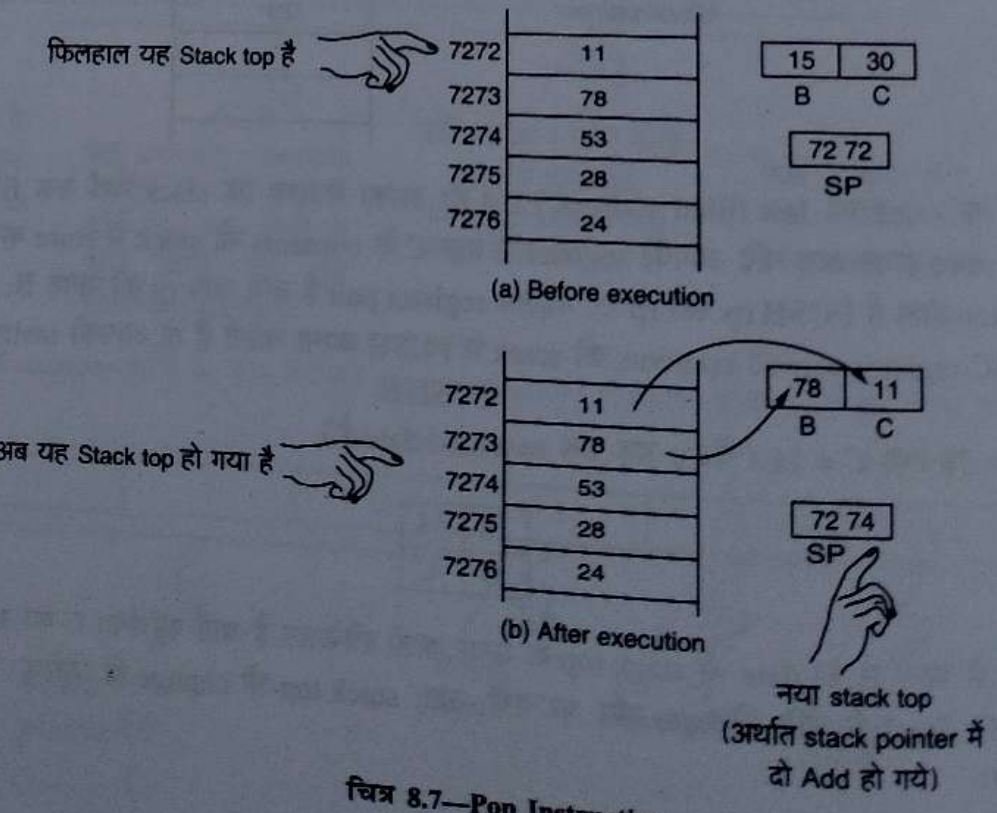
78	11
B	C

तो PUSH B से पहले B का data जो stack top के ऊपर वाली लोकेशन है वहाँ पहुँचेगा। C का डाटा उससे ऊपर वाली location में पहुँचेगा। Stack में चूँकि दो byte और भर गयी, अत: stack top भी change हो जायेगा (stack pointer में दो minus हो जायेगा)।



चित्र 8.6—Push Instruction

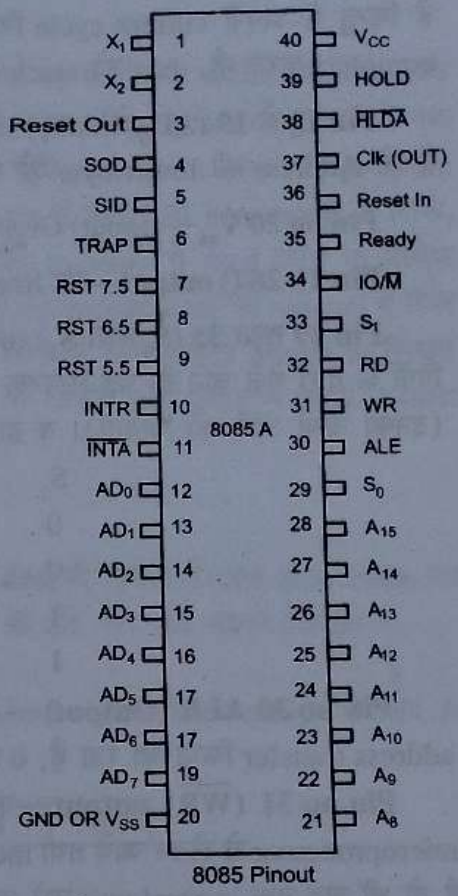
POP rp—POP का काम PUSH से विपरीत होता है। जब microprocessor अपनी subroutine की सेवा करके वापस आता है, तो stack से डाटा वापस पाने के लिए POP का use करता है। Stack top के contents register pair के lower register को मिल जाते हैं। [Stack top + 1] के contents upper register को मिल जाते हैं, तथा चूँकि stack top के दो डिब्बे खाली हो जाते हैं अत: stack top में दो की बढ़ोत्तरी हो जाती है। उदाहरणत: POP B के लिये देखें—



चित्र 8.7—Pop Instruction

§ 8.3. 8085 का पिन डायग्राम :

जैसा कि आप जानते होंगे कि 8085 40 पिनों वाला DIP IC है। इसका पिन डायग्राम चित्र 8.8 में प्रदर्शित है। आइये एक-एक करके इसकी पिनों का मुआयना करते हैं—



चित्र 8.8—8085 Pin Diagram

Pin 1 and 2 (Input Pins)—जैसे कि प्रत्येक घर में, office में, दुकान में घड़ी होती है जो कि पूरी दिनचर्या को control करती है। ऐसे ही clock 8085 को भी उपलब्ध करायी जाती है जो उसे 3MHz की pulses प्रदान करती है यह एक crystal clock होती है जो pin 1 तथा 2 के मध्य apply की जाती है (चित्र 8.9)।

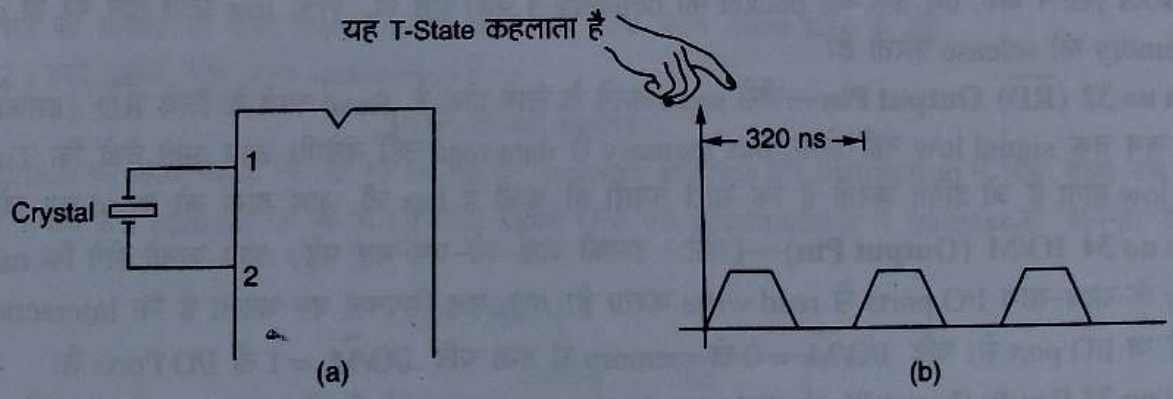
Pin 3 Reset Out (Output Pin)—यह pin Microprocessor को Reset किये जाने की सूचना देती है।

Pin 4 (SOD) Output Pin—इस पर मैंने पहले भी चर्चा की थी। SIM द्वारा serial data output कराने हेतु इस pin का use किया जाता है।

Pin 5 (SID) Input Pin—जो serial data आप यहाँ Input करायेंगे RIM के पश्चात् वह Accumulator की MSB पर मिलेगा।

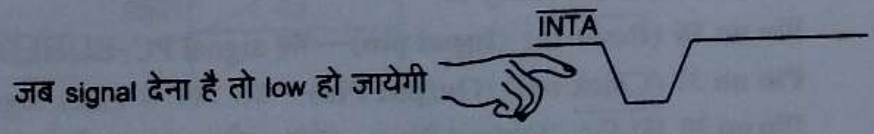
Pin 6, 7, 8, 9 (TRAP, RST 7.5, RST 6.5, RST 5.5) Input Pins—यह पिन microprocessor की hardware interrupt pins है जिससे उसको interrupt किया जा सकता है।

Pin no 10 INTR (Input Pin)—यह सबसे low priority वाली Interrupt Pin है।



चित्र 8.9

Pin no 11 $\overline{\text{INTA}}$ —(Output Pin) $\overline{\text{INTA}}$ (अर्थात आई-एन-टी-ए बार) का मतलब Interrupt Acknowledge, और इसके ऊपर लगा bar यह बता रहा है कि यह Active low है। Means कि normally तो high रहेगी, तथा जब signal देना होगा तो low हो जायेगी (चित्र 8.10 देखें)।



चित्र 8.10—Active low INTA

जब भी Microprocessor को कोई Interrupt signal प्राप्त होता है तो वह एक low $\overline{\text{INTA}}$ signal वापस भेजता है। यह low $\overline{\text{INTA}}$ signal एक तरह से microprocessor द्वारा Interrupt signal की सहर्ष प्राप्ति को स्वीकार करना इंगित करता है। अत: low $\overline{\text{INTA}}$ signal भेजकर microprocessor यह संदेश देना चाहता है कि महोदय, मुझे interrupt सहर्ष स्वीकार

है किन्तु मैं अपनी current cycle निपटा लूँ फिर आपकी सेवा करता हूँ। फिर microprocessor अपनी current cycle complete करता है, data को stack में save करता है, तथा निकल पड़ता है interrupt service subroutine के लिये।

Pin 12 से 19 (AD_0 से AD_7) (Bidirectional)—यह lines multiplexed Address data bus से connected रहती है जो Address की lower byte को ले जाने का तथा data को लाने ले जाने का दोहरा काम करती है।

Pin no 20 V_{ss}—(Input) Ground reference line

Pin 21-28 () output—यह lines Address bus से connect की जाती है जो address की upper byte को ले जाती है।

Pin 29 तथा 33 (S_0 तथा S_1) (output)—Timing तथा control unit द्वारा जारी दो महत्वपूर्ण सिगनल तथा इन दोनों पिनों के द्वारा भेजे जाते हैं। यह सिगनल status signal कहलाते हैं तथा बताते हैं कि read cycle चल रही है या write cycle (इनकी चर्चा आगे की जायेगी)। व द्वारा इंगित किये जाने वाले operations निम्नवत् हैं—

S_1	S_0	Operation
0	0	Halt
0	1	Write
1	0	Read
1	1	Opcode fetch

Pin no 30 ALE (Output)—ALE अर्थात Address Latch Enable. यदि यह 1 है, तो Address Data Bus पर address transfer किया जा रहा है, 0 होने पर data.

Pin no 31 ($\overline{WR}$) output—जब memory write cycle चलती है तो Bus का काम होता है कि 8-bit data को microprocessor से लेकर जाये तथा memory में write करके आये। जब Bus data को लेकर memory के दरवाजे पर पहुँचती है तो भी जब तक control unit उसे आदेश नहीं करेगी, वह data को write नहीं करेगी। और control unit $\overline{WR}$ signal को low करके यह आदेश देती है। यह बिल्कुल ऐसी ही है जैसे कोई व्यक्ति डाक लेकर पते पर तो खड़ा हो लेकिन जब तक उसका Boss yes न करे, तब तक वह packet की delivery न करे। ऐसे ही, $\overline{WR}$ low प्राप्त होने पर ही data bus अपना data memory को release करती है।

Pin no 32 ($\overline{RD}$) Output Pin—जैसे write करने के लिये होता है, Read करने के लिये $\overline{RD}$ (इसको रीड-बार पढ़ें) होता है। जब तक signal low नहीं होगा, bus memory से data read नहीं करेगी। आप आगे देखें कि Timing cycle के state में low होता है जो इंगित करती है कि सारी तैयारी हो चुकी है bus जी, अब डाटा को Read कर लो।

Pin no 34 IO/$\overline{M}$ (Output Pin)—(नोट : इसको आई-ओ-एम-बार पढ़ें) आप जानते होंगे कि microprocessor memory के साथ-साथ I/O ports से read write करता है। अत: यह सिगनल यह बताता है कि interaction memory से हो रहा है या I/O port से। यदि I/O/$\overline{M}$ = 0 तो memory से तथा यदि I/O/$\overline{M}$ = 1 तो I/O Ports से।

Pin no 35 Ready (Input Pin)—यह signal microprocessor को किसी peripheral device से प्राप्त होता है जो यह बताता है कि device ready है।

Pin no 36 ($\overline{Reset\text{-}in}$) (Input pin)—यह signal PC, EI, HLDA को reset कर देता है।

Pin no 37 (Clock out) (Output Pin)—इस पिन से clock signal प्राप्त किया जा सकता है।

Pin no 38 $\overline{HLDA}$ (Output Pin)—(नोट : इसे एच-एल-डी-ऐ-बार पढ़ें)—DMA transfer में जब microprocessor को hold request भेजी जाती है तो requesting device उसे स्वीकार करते हुये $\overline{HLDA}$ भेजता है तथा कहता है कि current cycle पूरी कर लूँ, फिर आपको buses का control देता हूँ।

Pin no 39 (Hold) (Input Pin)—इस पिन के द्वारा microprocessor को DMA request भेजी जाती है।

Pin no 40 (+ V_{CC}) (Input)—यह V_{CC} पिन है तथा इससे + 5V सप्लाई को IC से connect किया जाता है।

§ 8.4. Instruction Set of 8085

किसी भी microprocessor के अध्ययन करते समय सबसे महत्वपूर्ण है कि आप यह जानने कि कोशिश करें कि उसके द्वारा कौन-कौन से instructions किये जाते हैं क्योंकि after all microprocessor को बनाया ही programming के लिये गया है। यदि आप instructions नहीं सीखेंगे तो फिर भला programming कैसे आयेगी। चूँकि microprocessors के program assembly language में लिखे जाते हैं, अत: जब भी आप प्रोग्राम लिखें तो program में कोई ऐसा Mnemonic न लिखें, जो कि invalid हो। ध्यान रखें कि आप जब इस प्रोग्राम को microprocessor की memory में feed करेंगे तो प्रत्येक Mnemonic के संगत op-code आपको लिखना पड़ेगा तथा यह op-code आपको उस microprocessor की manual में मिल जायेंगे लेकिन यदि आपने instruction गलत लिखा है तो उसका op-code ही नहीं मिलेगा। अक्सर छात्र programs लिखते समय तरह-तरह की गलतियाँ करते हैं, अत: Instruction set पर चर्चा करने से पहले मैं आपको कुछ उदाहरण लेकर समझाता हूँ कि गलत instruction से मेरा क्या मतलब है—

उदाहरण 1 : 8085 की assembly language में एक instruction होता है

MVI r, 8-bit data

यहाँ r कोई भी 8-bit register हो सकता है तथा 8-bit data कोई भी binary data हो सकता है। इस instruction का काम यह है कि इस 8-bit के data को register r में transfer कर देगा। उदाहरण के तौर पर, यदि आपने लिखा—

MVI A, 36H.

जहाँ 36 hexadecimal संख्या है। अब चूँकि 36 H $=(00110110)_2$ अत: इस instruction के execution के बाद A के contents निम्नवत् हो जायेंगे—

0	0	1	1	0	1	1	0

अब मान लीजिये आपने अपने प्रोग्राम में एक instruction लिखा—

MVI A, 3824 H

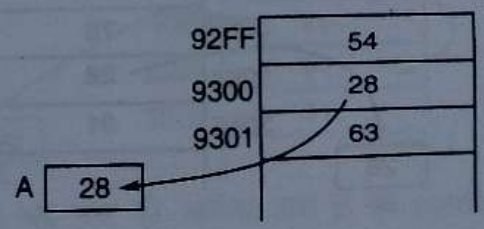
बताइये, हो गई न गलती। भला 8-bit के accumulator में 16-bit का डाटा कैसे जा सकता है। बड़े बर्तन का सामान छोटे बर्तन में नहीं जा सकता न। डाटा transfer तभी होगा जब दोनों बर्तन same size के हों।

उदाहरण 2 : इसी प्रकार एक अन्य instruction है—

LDA 16-bit address.

इस instruction का कार्य यह है कि जो भी 16-bit का memory address इस instruction में दिया होगा उस address की contents (अर्थात उस address पर जो 8-bit डाटा store होगा वह accumulator में transfer हो जायेगा। उदाहरणत: आपने लिखा—

LDA 9300 H



तो 9300 में stored 28 A में चला जायेगा।

अब मान लीजिये आपने गलती से लिख दिया

LDA 26 H

तो यह फिर से गलत instruction हो गया, क्योंकि LDA के बाद आपको address की दो bytes लिखनी है, एक नहीं।

उदाहरण 3 : 8085 में एक instruction होता है—

MOV r_2, r_1

यहाँ 8-bit r_1 का data 8-bit, r_2 में ट्रांसफर होता है।

उदाहरणत:

MOV B, A

यहाँ A register का data B में ट्रांसफर होगा। अब मान लिया आपने गलती की और लिख दिया

MOV B, 27

यह भी गलत है क्योंकि इस instruction के संगत कोई op-code आपको 8085 की manual में नहीं मिलेगा।

ऐसे ही कई प्रकार की गलतियाँ हो जाती है, क्योंकि जिनसे बचने के लिये instruction set को सावधानी पूर्वक पढ़ना चाहिये अर्थात कहाँ 8-bit data लिखना है, कहाँ 16-bit address लिखना है। किस instruction का क्या फंक्शन है इत्यादि।

8085 के instruction set को पाँच ग्रुप्स में बाँटा जा सकता है—

Data Transfer Group

इन instruction की सहायता से data को register के बीच या memory से register के बीच ट्रांसफर किया जाता है। Data transfer group (आंकड़ा स्थानान्तरण समूह) के instruction निम्नवत् है—

(i) MOV r_2, r_1

यहाँ r_1 व r_2 के जगह आप कोई भी 8-bit general purpose register लिख सकते हैं; जैसे—MOV A, B या MOV C, B या MOV B, H इत्यादि। यहाँ जो रजिस्टर बाद में लिखा जायेगा (r_1) वह source है तथा वहाँ का data r_2 (जो destination है) में पहुँचेगा। अत: हम इस instruction को पढ़ेंगे MOV r_1 to r_2। उदाहरणत: MOV C, B को पढ़ेंगे MOVE B to C इत्यादि।

उदाहरण

MOV B, C

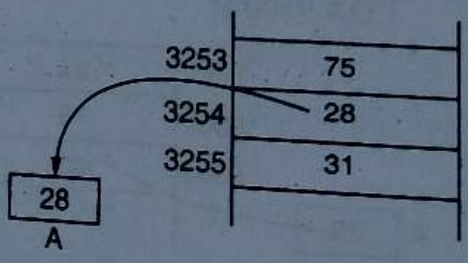
माना इन्स्ट्रक्शन execute होने से पहले B =22 C =33

Instruction execute होने के बाद

C का डाटा B में चला गया, अत: B change हो गया। C पर कोई प्रभाव नहीं पड़ा।

(ii) LDA 16-bit address—इस instruction में आप जो 16-bit address लिखेंगे उस memory address पर जो 8-bit का डाटा होगा वह accumulator में ट्रांसफर हो जायेगा। उदाहरणत:

LDA 3254 H अर्थात Load the accumulator with the contents of memory location 3254 H. अब 3254 H में जो भी stored होगा (चित्र देखें) वह accumulator में transfer हो जायेगा।



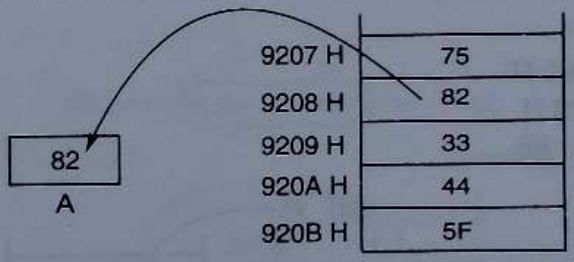
(iii) MOV r, M अर्थात move the contents of memory to register r देखने में यह instruction बहुत simple लगता है। आप कहेंगे कि यह memory की contents को register r में transfer कर देगा। पर मामला इतना सरल नहीं है जनाब। मामला पेचीदा है। Memory में कोई एक location नहीं है, पूरी 65536 locations हैं तथा प्रत्येक का 16-bit का unique address है। तो प्रश्न यह है कि LDA instruction में तो आपने LDA के आगे 16-bit का address लिख दिया था, अत: microprocessor को पता ढूँढने की परेशानी नहीं हुई। पर यहाँ तो आपने 16-bit का address लिखा ही नहीं है तो फिर कैसे ढूँढेगा microprocessor कि 65536 bytes में से कौन-सी बाइट को register r में पहुँचाये।

तो साहब, इस instruction (तथा आगे जितनी भी instruction जिनमें M अक्षर लिखा होगा (memory referenced instructions)) के लिये 8085 में यह व्यवस्था है कि ऐसी स्थिति में वह address को HL pair से उठायेगा। HL pair में जो भी 16-bit stored होंगे, उनको address मानते हुये उस address से डाटा उठा कर register r में पहुँचाया जायेगा। इसी कारण HL pair को memory pointer का नाम दिया गया है क्योंकि सभी memory reference instructions की स्थिति में memory का address यहीं से उठाया जाता है। यहाँ एक बात यह भी समझ लें कि इस प्रकार की instruction use करने से पहले आपको HL pair में address load करना पड़ता है जिसके लिये LXIH 16-bit data नामक instruction होता है। उदाहरणत: LXIH 9500 से HL pair में 9500 पहुँच जायेगा। दूसरी बात यह समझ लें कि जब भी किसी instruction हेतु memory का address किसी register pair से उठाना पड़ता है तो इसे indirect addressing कहते हैं। जबकि यदि address सीधे instruction के अंदर दिया होता है (जैसे कि LDA में) तो इसे direct addressing कहते हैं। ऐसे ही, यदि data register to register transfer होता है (जैसे MOV A, B) तो इसे register addressing कहते हैं। यह सब addressing modes है जिनके विषय में आप आगे पड़ेंगे। किन्तु यहाँ मैं इसका जिक्र इसलिये कर रहा हूँ ताकि अब आगे जो भी instruction आये पढ़ें, तो प्रत्येक के लिये सोचें कि इसमें addressing mode कौन-सी है।

आइये, एक उदाहरण लेते हैं। मान लीजिये कि आपने निम्न दो instruction execute कराये—

LXI H 9208H
MOV A, M

तो इसमें निम्न प्रभाव होगा—



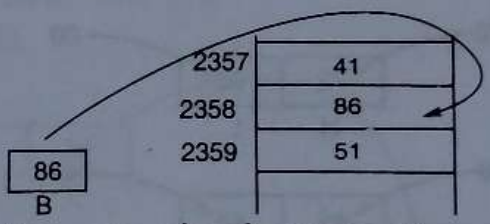
यहाँ A में जो पहले stored होगा वह नष्ट हो जायेगा व 9208 H की contents A में पहुँच जायेगी।

(iv) MOV M, r

यहाँ register r का data उस memory location में जायेगा जिसका address HL pair में होगा। उदाहरणत:

LXI H 2358 H
MOV M, B

तो इन दो इन्सट्रक्शन्स का प्रभाव निम्नवत् होगा—



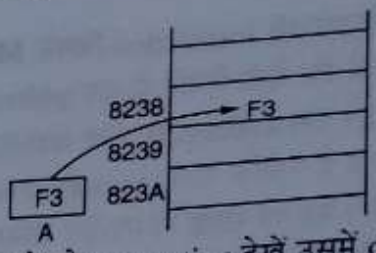
यहाँ 2358 H में जो पहले stored होगा वह नष्ट हो जायेगा तथा B की contents 2358 H में पहुँच जायेगी।

(v) STA 16-bit address अर्थात store the contents of accumulator at the memory location whose 16-bit address is given in the instruction अर्थात Accumulator की contents को उस memory location पर store कर दो जिसका address instruction में दिया गया है। ध्यान दें कि चूँकि यह address instruction के साथ देना जरुरी है, अत: यह direct addressing है।

उदाहरण

STA 8238 H

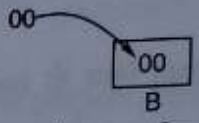
Now suppose A = F3 This F3 will reach 8238H.



(vi) MVI r, 8-bit data—अब तक आपने जो instruction देखें उसमें data register to register transfer हो रहा था या फिर memory to register. कई बार ऐसा होता है कि user (programmer) अपना स्वयं का data किसी register में store कराना चाहता है। उदाहरणत: मान लीजिये कि आप चाहते हैं कि register B में 00 H transfer करा दें। अत: 8-bit data को instruction के साथ लिख दिया जाता है। Instruction के execution के बाद यह data register में पहुँच जाता है। उदाहरणत:

MVI B, 00 H

अत: B में 00 पहुँच जायेगा।



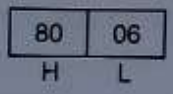
इस प्रकार की addressing जहाँ data instruction के साथ दिया जाता है, Immediate addressing कहलाती है।

(vii) MVI M, 8-bit data—इस Instruction द्वारा आप 8-bit data को वांछित memory location में पहुँचा सकते हैं। मान लीजिये आप memory location 8006 H में 28 H store करना चाहते हैं तो HL pair initialize कीजिये व उक्त instruction का use कीजिये अर्थात—

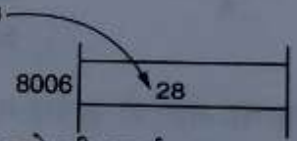
LXI H 8006 H
MVI M, 28 H

इन Instructions के निम्न प्रभाव होंगे—

(i)



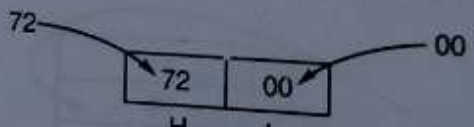
(ii)



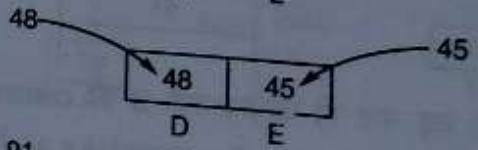
(viii) LXI rp 16-bit data—इस instruction पर मैं पहले ही चर्चा कर चुका हूँ। जब किसी register pair को initialize करना होता है, तो LXI instruction का use कर सकते हैं। HL को initialize करने हेतु की जगह H लिखना होगा। BC को initialize करने हेतु की जगह B लिखना होगा तथा DE को initialize करने हेतु की जगह DE लिखना होगा। बताइये, इस Instruction की addressing mode क्या है?

उदाहरण—

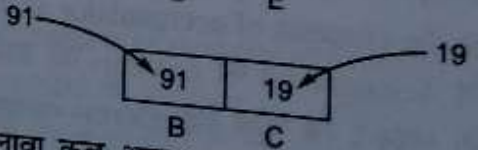
(a) LXI H, 7200



(b) LXI D, 4845



(c) LXI B, 9119



डाटा transfer group के इसके अलावा कुछ अन्य instructions भी हैं, जो छात्र कम use करते हैं तथा अक्सर भूल जाते हैं, अत: कृपया इनको भी महत्व दें, व ध्यान से समझें—

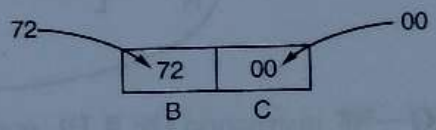
(ix) LDAX rp—यहाँ जो register pair लिखा होगा उस में जो address store होगा, उस address की contents accumulator में पहुँचेंगी। यहाँ इस instruction में यह restriction है कि rp की जगह B या D ही use किया जा सकता है

H मान्य नहीं है अर्थात LDAX H या STAX H invalid instruction हैं तथा op-code table में आपको इनके op-code नहीं मिलेंगे—

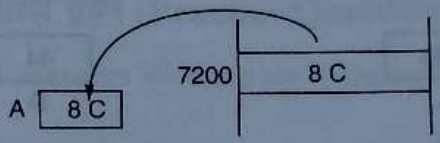
उदाहरण—

LXI B 7200
LDAX B

Now LXI B 7200 will move 72 into B and 00 into C



Now LDAX B will treat this as address, and move its contents to A.

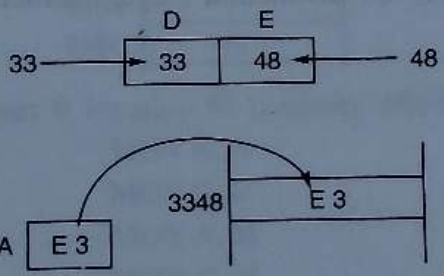


(x) STAX rp—यहाँ दिये गये rp (BC के लिये B, DE के लिये D, HL मान्य नहीं है) में stored address पर Accumulator की contents पहुँचायी जायेगी—

उदाहरण—

LXI D 3348 H
STAX D

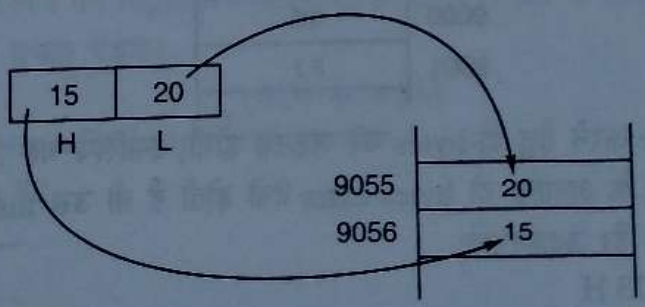
Now :



(xi) SHLD 16-bit address—यह instruction HL pair का data दिये गये memory address में transfer करता है। किन्तु यहाँ समस्या यह उत्पन्न होगी कि एक memory location तो केवल 8-bit store कर सकती है जबकि HL pair में तो 16-bit का data होगा। तो इसमें यह होता है कि L का data तो instruction में दिये गये address पर transfer होता है और H का डाटा उससे अगली location पर ट्रांसफर होता है।

उदाहरणत : माना H में 15 है, L में 20 है, आपने दिया

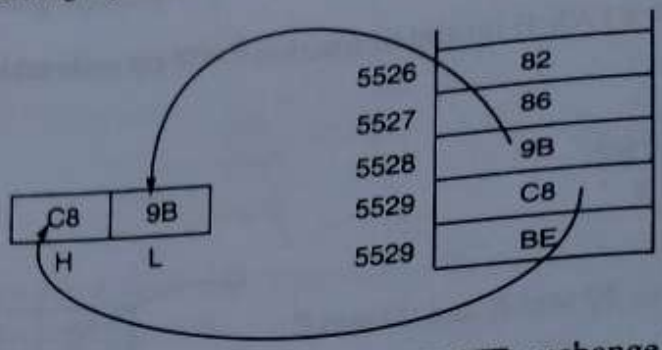
SHLD 9055 H.



तो L की contents (20) जायेंगी 9055 H में व H की contents (15) जायेंगी 9056 में।

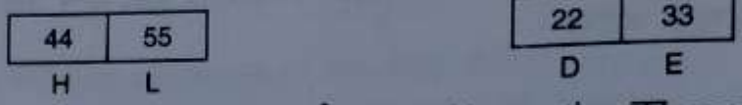
(xii) LHLD address—इस instruction द्वारा दिये गये address का data. HL pair में load होता है। तो जो address mention की गयी होगी उसकी contents L में आयेगी, उससे अगली location की contents H में आयेगी उदाहरणत:

LHLD 5528 H.



(xiii) XCHG—यह instruction DE व HL register pairs का डाटा exchange करता है। माना XCHG execute होने से पहले

अत: XCHG execute होने के पश्चात्

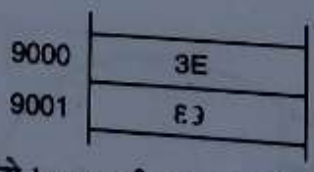


इससे पहले कि आप अन्य ग्रुप्स के instructions पढ़े, size of instruction का concept भी समझ लें। यदि किसी instruction में केवल Mnemonic होता है तो program लिखते time केवल उसका op-code feed करना होता है। चूँकि op-code 8-bit या 1 byte का होता है, अत: ऐसे instruction 1 byte instruction कहलाते हैं।

उदाहरणत:

MOV r_2, r_1
MOV *r, M*
MOV *M, r*
LDAX *rp*
STAX *rp*
XCHG इत्यादि।

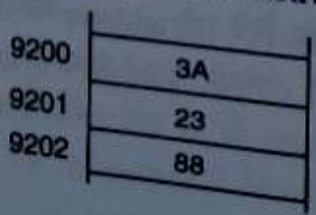
यदि instruction में Mnemonic के साथ 8-bit का data भी होता है तो memory में store करते समय इस instruction को दो bytes की जरूरत होगी उदाहरणत: MVI A 89 H का उदाहरण लें अब MVI A का op-code 3E होता है। यदि आप यह instruction memory में store कर रहें हैं तो आपको 3E के साथ-साथ 89 को भी store करना होगा (उससे अगली location पर)



तो ऐसे instructions को store करने हेतु दो-bytes की जरूरत होगी, इसलिये यह 2-byte instructions कहलाते हैं। यदि किसी instruction में op-code के अलावा दो bytes extra देनी होती है तो उस instruction को memory में store करने हेतु 3-bytes की जरूरत पड़ती है। उदाहरणत:

LDA 8823 H

LDA का op-code होता है 3A. अत: आप memory में इस instruction को निम्नवत् store करेंगे—



ध्यान दें कि हमेशा ऐसी स्थिति में lower byte पहले store की जाती है, फिर upper byte. यहाँ आप देखें कि इस instruction को store करने में 3-bytes खर्च होती है, अत; यह 3-byte instructions कहलाते हैं। 3 byte instructions के अन्य उदाहरण निम्नवत् हैं—

LDA address (16-bit)
STA address (16-bit)
LHLD address (16-bit)
SHLD address (16-bit)
LXI *rp*, data (16-bit) इत्यादि।

अभ्यास प्रश्न (Practice Questions)

माना 8085 microprocessor के कुछ registers की contents व memory location की contents निम्नवत् हैं—

A = 45 B = 46 C = 88
D = 95 E = 82 H = 91
L = 88

9188	22
9189	28
918A	92
918B	11
918C	F2

निम्न Instruction के बाद इन registers व location की contents बताइये—

MOV B, A
MOV D, C
MOV A, M
MOV C, M
LDA 918B

Arithmetic group—Binary संख्याओं पर Arithmetic operation करने हेतु इन instruction का use किया जाता है। यदि आपने Data transfer group ध्यान से पढ़ लिया है, तो इन instructions को समझने में आपको ज्यादा दिक्कत नहीं आयेगी। यहाँ ध्यान रखें कि जब भी किसी Arithmetic operation में दो operends use होते हैं, तो एक operend स्वत: A में ही माना जाता है।

(i) ADD *r*—यह instruction दिये गये register (A, B, C, D, E, H या L) के contents A से जोड़ेगा व result A में store करेगा। Flag register पर भी प्रभाव पड़ेगा।

$$[A] \leftarrow [r] + [A]$$

means that contents of A and r are added and result goes to A.

उदाहरण—माना A = 18, B = 82

अब माना अपने use किया है—

ADD B

[A] ← [A] + [B]

A	=	0001	1000
B	=	1000	0010
		1001	1010

So A में 1001 1010 अर्थात 9 A चला जायेगा।

Flag register को देखें—

Cy = 0 (कोई carry नहीं)
P = 1 (1 की संख्या 4 (even))
AC = 0 (Nibble पर carry नहीं)
Z = 0 (Accumulator में non-zero value)
S = 1

1	0	0	0	0	1	0	0

अत: इस instruction के execution के बाद—

A = 9A
Flag register = 84

A तथा flag register मिलकर PSW या program status word कहलाते हैं।

PSW = | FR | Acc |

अत: PSW = 84 9A

(ii) ADD M—इसमें memory के contents A से जुड़ेगी, result A में जायेगा। Memory का address HL pair से लिया जायेगा।

$$[A] \leftarrow [A] + [M_{HL}]$$

(iii) ADI 8-bit data—Instruction के साथ दिया गया 8-bit data A से जुड़ेगा, result A में जायेगा।

$$[A] \leftarrow [A] + \text{8-bit data}$$

(iv) यदि addition में carry flag के contents (जो कि उसमें पहले से हों) भी जुड़ते हैं, तो उसे Add with carry कहते हैं। 8085 में Add with carry हेतु निम्न इन्स्ट्रक्शन्स हैं—

ADC r
means
$[A] \leftarrow [A] + [r] + [CY]$
ADC M
means
$[A] \leftarrow [A] + [M_{HL}] + [CY]$
ADI 8-bit data
means
$[A] \leftarrow [A] + \text{data} + [CY]$

(v) इसी प्रकार substraction हेतु instructions हैं—

SUB r
means
$[A] \leftarrow [A] - [r]$
SUB M
means
$[A] \leftarrow [A] - [M_{HL}]$
SUI 8-bit data
means
$[A] \leftarrow [A] - \text{data}$

ध्यान रखें कि substraction instructions में यदि result negative होता है (अर्थात छोटी संख्या में से बड़ी संख्या minus की जाती है) तो carry flag set हो जाता है अर्थात्
for positive result CY = 0
For negative result CY = 1.

(vi) यदि substraction में carry flag को भी consider किया जाये, तो उसे Substraction with Borrow कहा जाता है। 8085 में इसके लिये निम्न instruction है—

SBB r
means
$[A] \leftarrow [A] - [r] - [CY]$
SBB M
means
$[A] \leftarrow [A] - [M_{HL}] - [CY]$
SBI 8-bit data
means
$[A] \leftarrow [A] - \text{data} - [CY]$

(vii) किसी register या memory location को increment (1 add करना) या decrement करने हेतु 8085 में निम्न instruction है—

INR r
means
$[r] \leftarrow [r] + 1$

उदाहरण—

ADD C
यदि C में 55 है, तो 56 हो जायेगा।
INR M
means
$[M_{HL}] \leftarrow [M_{HL}] + 1$

यदि HL pair में 4000 है तथा 4000 memory location में 48 है तो 49 हो जायेगा।
इसी प्रकार

DCR r
means
$[r] \leftarrow [r] - 1$
DCR M
means
$[M_{HL}] \leftarrow [M_{HL}] - 1$

(viii) यदि आपको register pair को increment करना है या decrement करना है, तो उसके लिये निम्न instructions हैं—

INX rp
means
$[rp] \leftarrow [rp] + 1$
DCX rp
$[rp] \leftarrow [rp] - 1$

उदाहरण—

माना H = 56, L = 48
अब INR H से H में 57 हो जायेगा।
INR L से L में 49 हो जायेगा।
लेकिन INX H में HL pair को एक संख्या मानते हुये increment करेगा।
यानि 5648 + 1 = 5649
अत: H = 56 L = 49

(ix) DAD rp—यह instruction 16 bit addition के लिये use किया जाता है। HL के contents दिये गये register pair से जुड़ेंगे व result HL pair में जायेगा।

$$[HL] \leftarrow [HL] + [rp]$$

माना

HL = 2000
DE = 1298

अब

DAD D होने पर HL = 3298.

(x) DAA—यह instruction कुछ अलग ही कार्य करता है तथा अपने आप में अनूठा है, DAA का full form है Decimal Adjust Accumulator. आमतौर पर, जब आप कोई mathematical operation करते हैं तो यह मानकर चलते हैं कि दोनों संख्यायें Binary हैं। किन्तु यदि दोनों संख्यायें BCD हैं (Binary Coded Decimal) तो क्या होगा। Result गलत आयेगा (यदि sum 9 से ज्यादा है तो)

उदाहरण 1—

BCD 1	23
BCD 2	46
Result	69

यहाँ परिणाम सही आया, क्योंकि result 9 से ज्यादा नहीं है।

उदाहरण 2—

BCD 1 = 69
BCD 2 = 22
Result = 8B

यहाँ परिणाम गलत आया (क्योंकि BCD में 69 + 22 = 91 आना चाहिये) क्योंकि B नौ से ज्यादा है इसे ठीक करने के लिये B में 0110 (यानि 6) add करना होगा।

8 B	=	1000	1011
Add 6	+		0110
Result		1001	0001
अर्थात		91	

अब परिणाम ठीक हो गया ।

उदाहरण 3—

BCD 1	45
BCD 2	89
Result	CE

फिर से परिणाम गलत आया (क्योंकि BCD में 45 + 89 = 134 आना चाहिये), अत: इसको भी ठीक करने हेतु दोनों जगह 6 add करना होगा।

CE	=	1100	1110
Add 6 to Both	=	0110	0110
Result	=	10011	0100
अर्थात		1 3	4

तो परिणाम ठीक हो गया।

प्रश्न यह उठता है कि आपको यह correction खुद करना होगा। जी नहीं, यह correction आपको DAA करके देंगे जब भी कोई BCD operation करें, तो उसके बाद DAA लिख दें, correction खुद हो जायेगा।

उदाहरण

ADD B
DAA

इत्यादि।

Logical group—Logical group में AND operation (bit wise), OR operation (bit wise), XOR operation (bit wire) होते हैं। इसके अतिरिक्त कुछ compare, rotate व complement operation भी होते हैं।

(i) AND operation—यदि दोनों बिट्स high हों तो उनको AND high होता है, अन्यथा low होता है अर्थात्

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

मान लीजिये

A =	0101	1101	= 5D
B =	1011	1011	= BB
A and B =	0001	1001	= 19

8085 AND operation करने हेतु निम्न instruction हैं—

ANA r

means

[A] ← [A] AND [r]

ANA M

means

[A] ← AND [M_{HL}]

ANI 8-bit data

means

[A] ← [A] AND 8-bit data

(ii) OR operation—यदि दो bits में से कोई एक high हो, तो उन दोनों का OR high होता है, अर्थात

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

उदाहरण,

A =	1001	1101	= 9D
B =	1000	1001	= 89
A OR B =	1001	1101	= 9D

8085 में OR operations हेतु instructions निम्नवत् हैं—

ORA r

means

[A] ← [A] OR [r]

ORA M

means

[A] ← [A] OR [M_{HL}]

ORI 8-bit data

means

[A] ← [A] OR data

(iii) XOR operations—यदि input bits में 1 की संख्या odd हो तो उनका XOR high होता है, यदि input bits की संख्या even हो तो उनका XOR low होता है अर्थात

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

उदाहरण,

A =	1010	1100	= AC
B =	0100	0100	= 44
A XOR B =	1110	1000	= E8

8085 में XOR operations हेतु निम्न instructions हैं—

XRA r

[A] ← [A] XOR [r]

XRA M

[A] ← XOR [M_{HL}]

XRI 8-bit data

[A] ← [A] XOR data

Compare instructions—Compare instruction किसी register की contents को change नहीं करते, केवल carry flag को change करते हैं। यह दिये गये contents की तुलना Accumulator की contents से करते हैं। यदि A की contents बड़ी पायी जाती है तो Cy = 0, यदि छोटी पायी जाती है तो Cy = 1. वास्तव में यह register contents को घटाकर देखते हैं, किन्तु contents को change नहीं करते।

Compare हेतु instructions निम्नवत् हैं—

CMP r

means

if [A] > [r] then CY = 0

if [A] < [r] then CY = 1

CMP M

if [A] > [M_{HL}] then CY = 0

if [A] > [M_{HL}] then CY = 1

CPI 8-bit data

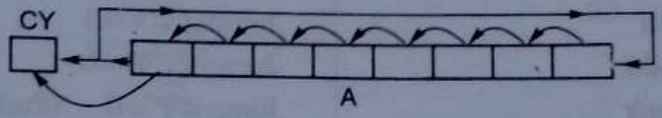
means

if [A] > data then CY = 0

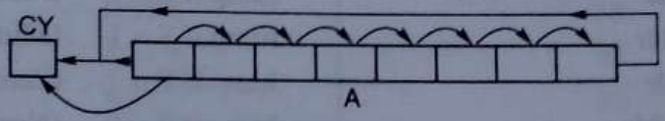
if [A] < data then CY = 1

Rotate instructions—यह instructions Accumulator को rotate करते हैं। यदि rotation carry के साथ होता है तो Rotate with carry, तथा यदि carry के बिना होता है तो rotate without carry कहा जाता है—

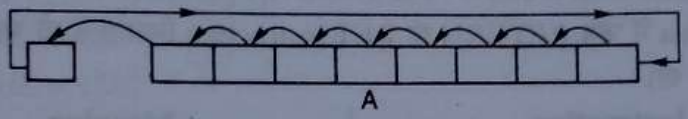
RLC (Rotate Accumulator left without carry)



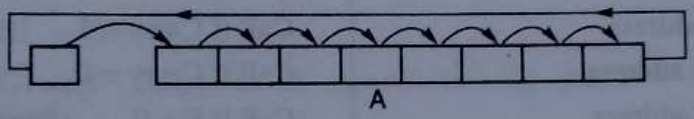
RRC (Rotate Accumulator Right without carry)



RAL (Rotate Accumulator left with carry)



RAR (Rotate Accumulator right with carry)



Other Instructions

CMA—यह accumulator के contents को complement (1 का 0 तथा 0 का 1) कर देता है (complements the accumulator)

CMC (Complement the carry flag)—Carry flag के contents को complement कर देता है।

STC (Set the carry flag)—Carry flag को set (अर्थात 1) कर देता है।

यहाँ एक बात और समझ लें। इस ग्रुप में आपने कई ऐसे instructions देखें जहाँ डाटा केवल Accumulator में होता है तथा उस डाटा पर कोई न कोई operation होता है। जैसे—RAL, RAR, RRC, RLC या CMA. ऐसे instruction में Implied (या Implicit Addressing) होती है अर्थात जहाँ डाटा कहीं से मँगाना नहीं है, वह तो accumulator में already मौजूद है।

Branch Control Group—Branch control group के instructions तब use होते हैं जब program का normal sequence change करना होता है। यदि बिना किसी condition के sequence change होता है तो unconditional branching कहा जाता है। यदि कोई condition पूरी होने पर conditional होती है तो इसे conditional branch कहते हैं।

उदाहरण,

JMP Address

यह unconditional jump है। यहाँ आपको 16-bit का address specify करना होगा। जहाँ आपको jump करानी है।

उदाहरणत: JMP 9400.

इसी JC Address अर्थात Jump if carry अर्थात यदि carry flag 1 होगा तभी jump होगी। समस्त conditional branching statements flags से condition को check करने के बाद ही branch करती है।

विभिन्न JUMP statements निम्नवत् है—

Instruction	Meaning
JMP addr	Jump unconditionally
JC addr	Jump if CY = 1
JNC addr	Jump if CY = 0
JPO addr	Jump if P = 0 (Parity odd)
JPE addr	Jump if P = 1 (Parity even)
JP addr	Jump if S = 0 (Plus)
JM addr	Jump if S = 1 (Minus)
JZ addr	Jump if Z = 1 (Zero)
JNZ addr	Jump if Z = 0 (Non zero)

CALL Instruction—Call Instruction तब use होती है जब program को sub-program या sub-routine करने जाता है तथा sub-routine के last में जब RETURN या RET instruction मिलता है तो वह वहीं वापस लौट आता है, जहां से वह program छोड़कर गया था।

Instruction	Meaning
CALL address	Call unconditionally
CC address	Call if Carry = 1
CNC address	Call if Carry = 0
CPO address	Call if P = 0 (Parity odd)
CPE address	Call if P = 1 (Parity even)
CP address	Call if S = 0 (Plus)
CM address	Call if S = 1 (Minus)
CZ address	Call if Z = 1 (Zero)
CNZ address	Call if Z = 0 (Non zero)

RETURN—Return instruction का मतलब कि sub-routine से वापस main program में लौट जाना। यहां address देने की आवश्यकता नहीं होती क्योंकि sub-routine में आने से पहले stack में return address save करके ही आया जाता है।

RETURN भी conditional या unconditional होते हैं—

Instruction	Meaning
RET	Return unconditionells
RC	Return if Cy = 1 (carry)
RNC	Return if Cy = 0 (No carry)
RPO	Return if P = 1 (Odd parity)
RPE	Return if P = 0 (Even parity)
RP	Return if S = 1 (Plus)
RM	Return if S = 0 (Minus)
RZ	Return if Z = 1 (Zero)
RNZ	Return if Z = 0 (Non zero)

RST या Software interrupts—यह instruction भी program को interrupt करती है तथा यहाँ address जहाँ jump करना है, पूर्व निर्धारित होता है—

Software Interrupts	Vector Address (Hexadecimal)
RST 0	00
RST 1	8
RST 2	10
RST 3	18
RST 4	20
RST 5	28
RST 6	30
RST 7	38

PCHL—यह instruction program counter तथा HL pair के contents को exchange करता है—
माना, यदि instruction से पूर्व

PC = 4559
HL = 1282

तो instruction के बाद

PC = 1282
HL = 4559

Machine Control Group—Programs को control करने हेतु कुछ अन्य instructions हैं जिनको machine control group में रखा गया है—

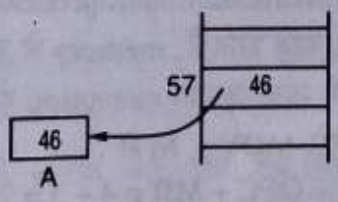
(i) HLT या Halt यानि stop अर्थात बस, अब रुक जाओ, प्रोग्राम समाप्त हो चुका है। यह instructions हर प्रोग्राम के last में लिखा जाता है।

(ii) NOP means कुछ मत करो, खाली बैठो अर्थात No operation. जब कभी program में waiting states (delay) डालने हों, तो इसका use किया जाता है।

(iii) IN 8-bit address तथा OUT 8-bit address.

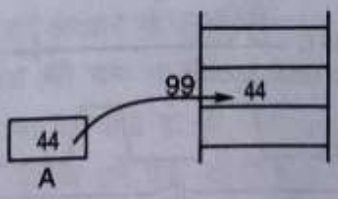
यह instruction accumulator is I/O port में डाटा लाने व ले जाने हेतु होते हैं। I/O ports का address 8 bit का होता है, अत: यहाँ 8 bit address दिया जाता है। उदाहरणत: IN instruction दिये गये address 57 से डाटा accumulator में लाता है।

For example: IN 57



Out instruction दिये गये port address पर Accumulator का डाटा पहुँचाता है।

For example : OUT 99 H.



PUSH व POP—इसका use stack में डाटा push करने (sub-routine में जाने से पहले) व POP करने हेतु (sub-routine से लौटने के पश्चात्) किया जाता है। इन पर चर्चा पहले ही की जा चुकी है।

SIM व RIM—Interrupt की masking हेतु SIM व interrupts की स्थिति ज्ञात करने हेतु RIM का use किया जाता है। इनपर भी चर्चा पहले की जा चुकी है।

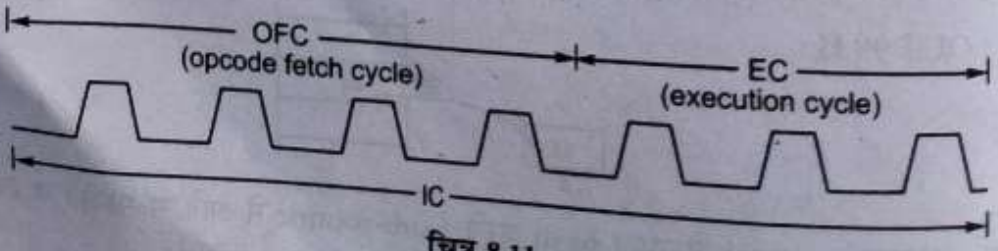
§ 8.5. Timing Diagrams

Timing diagram ठीक वैसे ही होते हैं जैसे कि आपके college का time-table होता है उदाहरणत: पहला पीरियड Visual Basic का होगा, दूसरा पीरियड Communication network का होगा, तीसरा पीरियड C programming का होगा इत्यादि इत्यादि। ऐसे ही, microprocessor में जो operations होते हैं, उनका भी निर्धारण रहता है कि पहले T-state में क्या-क्या operations होंगे, दूसरे T-states में क्या-क्या operations होंगे, तीसरे T-state में क्या-क्या operations होंगे। जब भी आप कोई program लिखते हैं तो उस program को आप मैमोरी में feed कर देते हैं। फिर आप microprocessor को उस program का starting address देते हैं कि हमने इस address से अपना प्रोग्राम store किया है, आप इस प्रोग्राम को run करो। उदाहरणत: माना कि आपने अपना प्रोग्राम memory location 2000 H से store किया है। तो जब आप इस program को run करोगे तो program counter में 2000 H load हो जायेगा और पहले instruction का op-code 2000 H से आयेगा। Op-code का memory से microprocessor में आना op-code fetch कहलाता है तथा इसके लिये जो जो घटनायें घटती हैं, उन्हें op-code fetch cycle कहते हैं। उदाहरणत: यदि आपका पहला instruction था MOV B, A यानि A register का डाटा B register में जाना है यानि माइक्रोप्रोसैसर के अंदर ही अंदर execution होना है। इस प्रकार के instruction op-code fetch cycle में ही execute हो जाने हैं। चूँकि 8085 की op-code cycle को complete होने में 4 T-states लगते हैं, अत: यह instruction तो 4 T-states में ही execute हो लेगा। लेकिन सभी instruction ऐसे नहीं होते। अब उदाहरण लें LDA 2800 का। यह instruction तीन बाइट का है, तो जब इसका op-code fetch होकर आयेगा तो instruction decoder पता लगा लेगा कि यह तो तीन बाइट का instruction है और अभी तो एक ही बाइट आयी है (op-code वाली) तो इसका मतलब दो बाइट और आनी है, तो फिर दो memory read cycles और होगी तब जाकर पूरा instruction आयेगा चूँकि memory read cycle में 3 T-states लगते हैं, अत: इस instruction को memory में आने में ही 4 + 3 + 3 = 10 T-state लगेंगे। अब आप देखें कि इस instruction का काम क्या है? LDA Addr instruction का काम यह है कि जो memory address instruction में दिया हुआ है उस address के contents memory से लेकर आओ तथा Accumulator में store करो। अब यह करने के लिये फिर से एक बार memory read करनी पड़ेगी तब जाकर यह instruction execute होगा यानि इसके execution हेतु एक memory read cycle और चाहिये। यानि इस instruction को पूरा होने में 4 + 3 + 3 + 3 अर्थात 13 T-states लगेंगे। तो मेरे कहने का मतलब यह है कि जब तक आपको properly यह मालूम नहीं होगा कि कोई instruction करता क्या है, आप यह नहीं पता लगा पायेंगे कि वह कितने T-states में execute होगा।

आइये, कुछ अन्य उदाहरण लेते हैं। MOV r, M instruction को लीजिये। यह one-byte instruction है अत: इसकी op-code fetch cycle (OFC) होते ही पूरा instruction microprocessor में आ जायेगा। लेकिन इसका execution OFC में नहीं हो सकता क्योंकि इस instruction का कार्य होता है, memory से डाटा लाकर register में transfer करना। अत: इसको एक memory read cycle और करनी पड़ेगी अत: इसका execution होने में 7 T-states लगेंगे। तो instruction cycle में fetch cycle व execute cycle आता है। जैसे MOV r, M में

$$IC = OFC + MR = 4 + 3 = 7 \text{ T states}$$

इसे चित्र 8.11 द्वारा निम्नवत् दर्शा सकते हैं—



चित्र 8.11

तो इसमें OFC तो प्रत्येक instruction के लिये same होगी (4 T-states) लेकिन execution time प्रत्येक instruction का अलग अलग होगा।

प्रश्न उठता है कि Timing diagram क्या होते हैं? देखिये जब भी microprocessor की कोई cycle चलती है तो उस cycle को complete होने में निर्धारित T-states लगते हैं जैसे OFC के 4 T-states, MR के 3 T-states, MW के भी 3 T-states इत्यादि। अब इन प्रत्येक T-states में भी कब क्या होना है, कौन से T-state में address bus address लेकर जायेगी, कौन से state high में लायेगी या ले जायेगी, विभिन्न status signals कि विभिन्न T-states में क्यों स्थिति हो, control unit द्वारा जारी किये जाने वाले अन्य सिगनल्स (जैसे ALE, IO/$\overline{M}$, इत्यादि) की क्या स्थिति होगी) अर्थात प्रत्येक T-state में इन सिगनल्स की स्थिति को जब चित्र द्वारा दर्शाया जाता है, तो वह चित्र Timing diagram कहलाता है। लेकिन Timing diagram लगने से पहले आपको यह पता होना चाहिये कि किसी cycle में क्या क्या घटनायें होती हैं तभी आप Timing diagram बना सकते हैं। तो आइये, समझने की कोशिश करते हैं कि op-code fetch cycle के 4 T-states में क्या क्या होता है।

Op-code Fetch Cycle—जब भी आप टाइमिंग डायग्राम बनायें तो सबसे पहले यह देखें कि status signals की उस cycle में क्या स्थिति होगी। यहाँ एक बात तय है कि status signal उस पूरी cycle के दौरान change नहीं होंगे। तो table देखने से यह पता चलता है कि op-code Fetch cycle के दौरान S_0 व S_1 रहते हैं, अत: S_0 व S_1 में आपको high दिखाना है। अब देखें IO/$\overline{M}$ को। जब भी memory से कोई interaction होगा, तो IO/$\overline{M}$ low रहेगा। अत:, आपको पूरी OFC में IO/$\overline{M}$ को low दिखाना है। OFC के पहले T-state में address bus व address data बस पर op-code का address load किया जाता है (आपको ध्यान होगा कि यह address program counter से load किया जायेगा क्योंकि program counter में ही next instruction का address होता है)।

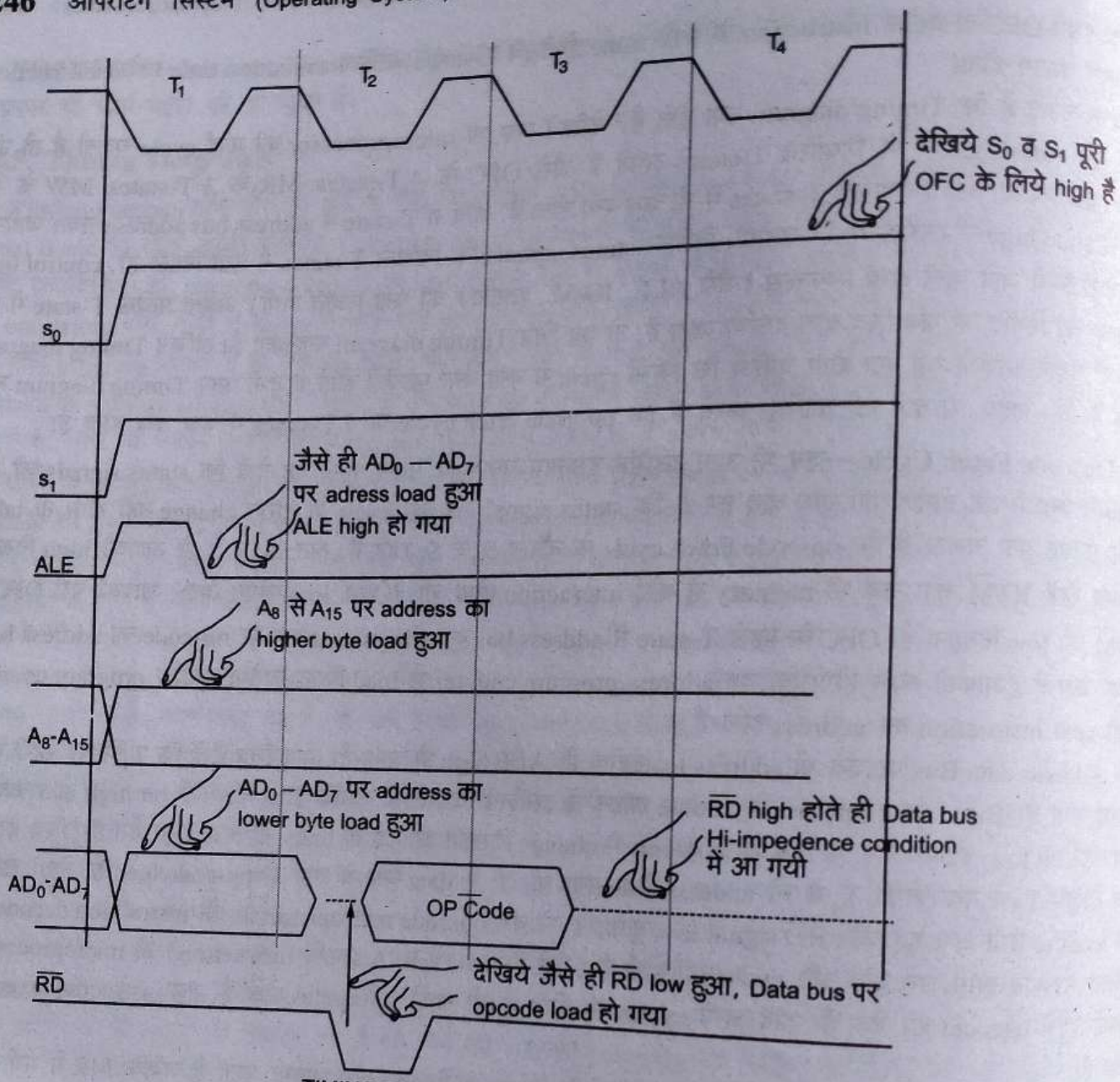
Address data Bus पर जब भी address load होता है, ALE high हो जाता है। आप चित्र देखें कि T_1 में यह सारी चीजें दिखाई गयी हैं। Buses के contents को double लाइन से इसलिये दिखाते हैं क्योंकि Bus की कोई bit high हो सकती है तथा कोई bit low. लेकिन जब भी bus की contents में change दिखानी होती है तो lines को cross कर देते हैं (चित्र देखें)। अब देखिए T_2 में क्या होगा। T_1 में जो address भेजा गया था, T_2 में data बस में वहाँ से op-code load हो गया। लेकिन यह loading उसी क्षण हुई जब $\overline{RD}$ signal low हुआ। T_3 में यह op-code microprocessor के instruction decoder में पहुँचता है। यदि इसमें अब कोई और cycle नहीं होनी है (जैसे कि MOV B, A इत्यादि instruction) तो microprocessor इसको T_4 में execute कर देता है। यदि अन्य cycles भी होती हैं, तो पहले वह cycles होती है, तब instruction execute होता है।

Memory Read Cycle—आपने देखा कि OFC में memory से op-code मंगाया जाता है जबकि MR में मैमोरी से अन्य 8-bit डाटा मंगाये जाते हैं। MR व OFC में पहला अंतर यह कि MR cycle में $S_0 = 0$, $S_1 = 1$ रहेगा। बाकी signals व में उसी प्रकार work करेंगे (बनाकर देखें)। MR cycle व MW cycles केवल 3-states में ही complete हो जाती हैं।

Memory Write Cycle—यह cycle memory read cycle से इस आशय में भिन्न है कि यहाँ bus data को memory में write करके आयेगी। अत:, $\overline{RD}$ की जगह $\overline{WR}$ सिगनल issue होगा, और तब Bus data memory में write करेगी।

I/O Read व Write Cycles—इन cycles में IO/$\overline{M}$ = 1 रहेगा। शेष सिगनल्स Memory Read व Memory Write की तरह कार्य करेंगे।

DMA (Direct Memory Acess)—DMA को समझने से पहले एक उदाहरण लेते हैं। यदि किसी व्यक्ति को एक शहर से दूसरे शहर जाना होता है तो सामान्यत: वह रोडवेज की बस या फिर ट्रेन से जाना होता है। लेकिन जब कोई विशेष स्थिति हो, बहुत सारे लोगों का एक साथ टूर पर जाना हो, बारात जानी हो या कोई industrial visit हो तो पूरी बस hire की जाती है। किसी bus operator को फोन किया जाता है कि कृपया हमें एक-दो दिन के लिये बस किराये पर दे दो। मान लीजिये बस वाला उत्तर देता है कि मेरी बस टूर से थोड़ी देर में लौटने वाली है, फिर आपके पास भेज देता हूँ। फिर बस वाला बस आपके पास भेजता है। आप अपना काम पूरा करते हैं व बस को लौटा देते हैं।



TIMING DIAGRAM OF OFC

ठीक यही स्थिति DMA transfer की है। यदि peripheral devices के मध्य बड़ी मात्रा में (bulk) data transfer होना होता है तो DMA controller (यह भी IC chip होती है जैसे 8237) HOLD singnal भेजकर microprocessor से बस की माँग करता है। स्वीकृति के लिये microprocessor Hold acknowledge ($\overline{\text{HLDA}}$) सिगनल लौटाता है। Current cycle को पूरा करने के बाद microprocessor कुछ समय के लिये अपनी bus का control DMA controller को सौंप देता है। DMA controller तीव्र गति से peripheral devices के मध्य डाटा transfer करवा देता है तथा फिर microprocessor को अपनी buses का कन्ट्रोल वापस मिल जाता है।

For the Instructions given below, specify the addressing mode, size of instruction, No. of T-states, machine cycle and its function; and complete the table

अभ्यास प्रश्न (Practice Questions)

Instruction	Addressing Mode	Size	No. of T-states	Machine Cycles	Function
MOV C, B	Register	3 bytes	4	OFC	moves the contents of B to C
MOV B, A					
MOV H, L					
MOV A, L					
MOV B, B					
MOV A, M					
MOV M, A					
MOV M, C					
MOV D, M					
MVI C, 35					
LDA 9000 H					
STA 531FH					
LDAX D					
ADD D					
ADIA 33H					
ANI A 35H					
XRA B					
SUB C					
SBB D					
CMP B					
CMP D					
CMA					
XCHG					
DAD D					
DAD H					
RIM					
SIM					

PUSH B					
POP D					
LXIH 3300					
HLT					
NOP					
PCHL					
LHLD 2315					
ADC B					
SBI 31H					
STA 1111H					
RET					
JMP 3101					
JNZ 2222					
CP 2323					
RPO					
EI					
RST 5					
RST 4					

Important Points to note :

- A microprocessor is a multipurpose, programmable logic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides result as output.
- The power supply of 8085 is + 5V (applied between pin no. 40 (V_{cc}) and pin no. 20 (V_{ss}) and clock frequency(applied between pin no. 1 and 2) is 3 MHz.
- Applications of microprocessor-based system:
  (i) For measurements, display and control of current, voltage, temperature, pressure, etc.
  (ii) For traffic control and industrial tool control.
  (iii) For speed control of machines.
- The accumulator is the register associated with the ALU operations and some input and output operations. It is an integral part of ALU. It holds one of data to be processed by ALU. It also temporarily stores the result of the operation performed by the ALU.
- 16-bit registers of 8085 are :
  Stack pointer (SP) and Program counter (PC).

REGISTER ORGANIZATION OF 8085:

W (8) Temp. Reg	Z (8) Temp. Reg.
B (8) Register	C (8) Register
D (8) Register	E (8) Register
H (8) Register	L (8) Register
Stack Pointer (16)	
Program Counter (16)	

- **Register pairs of 8085 :**

  B-C register pair

  D-E register pair

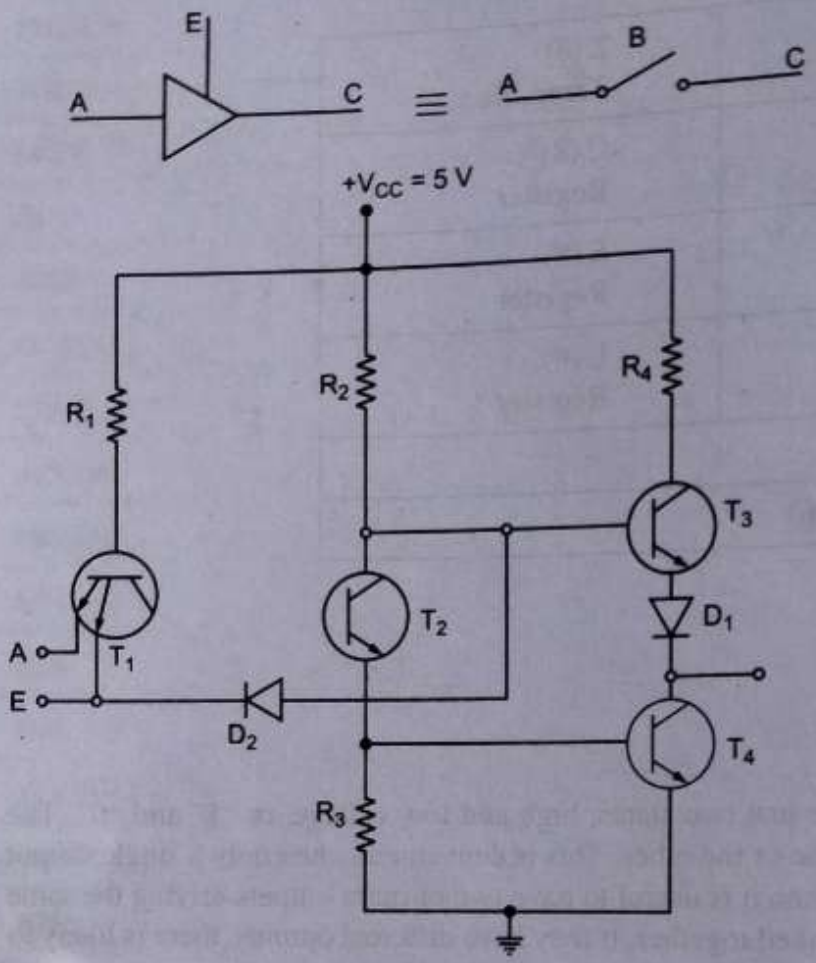
  H-L register pair

- Electronic logic circuits (Logic gates) use just two states, high and low voltage, or "1" and "0". The output of a gate will always be at one value or the other. This is convenient when only a single output is used to drive a signal. In some cases, when it is useful to have two or more outputs driving the same signal line. However, if two outputs are linked together, if they have different outputs, there is likely to be damage to the outputs and the level on the line will be un-predictable.

  A tri-state output (or three-state output) has the same high and low levels as standard logic outputs but it has a third state, namely high impedance. A high impedance state means that the output is not transferred to the line so effectively, the output is simply turned off (kept floating). Another logic gate can now drive the line and the level is entirely predictable. Numerous outputs can now drive a single line as long as only on is turned on at any time.

- In digital electronics 3-state logic allows an output port to assume a high impedance state (Hi—Z) in addition to the 0 and 1 logic levels, effectively removing the output from the circuit. This allows multiple circuits to share the same output line or lines (such as a bus which cannot listen to more than one device at a time can be used for multiple devices).
- Three-state outputs are implemented in many registers, bus drivers, and flip-flops in the 74 and 40 series as well as in other types, but also internally in many integrated circuits. Other typical uses are internal and external buses in microprocessors, memories, and peripherals.
- The whole concept of the third state (Hi-Z) is to effectively remove the device's influence (almost insulate or isolate it) from the rest of the circuit. If more than one device is electrically connected, putting an output into the Hi-Z state is often used to prevent short circuits, or one device driving high (logical 1) against another device driving low (logical 0).

Bus contention : Bus contention is when more than one device is trying to drive a data line at the same time. This is usually caused by timing errors in the design. If one device is trying to drive the bus line low and another is trying to drive it high, then it gives rise to bus contention (बस कन्टैन्शन). You can avoid this by having good bus arbitration. The master device should always control who has access to the bus at any given time. All slave devices need to be able to tri-state their outputs to the bus.



INPUT		OUTPUT
A	E	
0	1	0
1	1	1
X	0	high impedance

चित्र 8.12

- The stack is a group of memory locations in the Read/Write memory (RWM) that is used for the temporary storage of binary information during the execution of the program. The stack related instructions are PUSH & POP
- SID (Serial input data line): It is an input line through which the microprocessor accepts serial data.
- SOD (Serial output data line): It is an output line through which the microprocessor sends output serial data.
- Opcode (operation code):

The part of the instruction that specifies the operation to be performed is called the operation code or Opcode (8 bit in case of 8085).

FUNCTION OF IO/$\overline{M}$ SIGNAL IN THE 8085 :

It is a status signal. It is used to differentiate between memory locations and I/O operations.

- When this signal is low (IO/$\overline{M} = 0$) it denotes the memory related operations.
- When this signal is high (IO/$\overline{M} = 1$) it denotes an I/O operation.

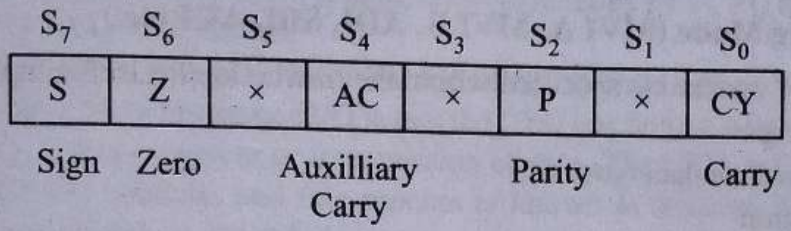
OPERAND:

The data on which the operation is to be performed is called as an Operand.

WAIT STATE:

This state is used by slow peripheral devices. The peripheral devices can transfer the data to or from the microprocessor by using READY input line. The Microprocessor remains in wait state as long as READY line is low. During the wait state, the contents of the address, address/data and control buses are held constant.

FLAG REGISTER :



The flags are used to reflect the data conditions in the accumulator. The 8085 flags are S-Sign flag, Z-Zero flag, AC-Auxiliary carry flag, P = parity flag, CY = Carry Flag.

Addressing Modes of 8085

- The method by which the address of source of data or the address of destination of result is given in the instruction is called Addressing **Mode.**
- The term addressing mode refers to the way in which the operand of the instruction is specified.Addressing Modes of 8085
- To perform any operation, we have to give the corresponding instructions to the microprocessor.
- In each instruction, programmer has to specify 3 things:
  - ❑ Operation to be performed.
  - ❑ Address of source of data.
  - ❑ Address of destination of result.

Types of Addressing Modes

- ❑ 'Intel 8085 uses the following addressing modes :
  - Direct Addressing Mode
  - Register Addressing Mode
  - Register Indirect Addressing Mode
  - Immediate Addressing Mode
  - Implicit (or Implied) Addressing Mode

Direct Addressing Mode (LDA, STA, LHLD, SHLD, JMP etc.)

- ❑ In this mode, the address of the operand is given in the instruction itself. exampleLDA 2200.
- ❑ LDA is the operation.
- ❑ 2200 H is the address of source.
- ❑ Accumulator is the destination.

Register Addressing Mode (MOV B, A, ADD B, SUB B, XRA B, ANA C etc.)

- ❑ In this mode, the operand is in general purpose register. Example MOV B, D
- ❑ MOV is the operation.
- ❑ D is the source of data.
- ❑ B is the destination

Register Indirect Addressing Mode (MOV M, A, ADD M, SUB M, XRA M, ANA M etc.) example MOV D M

- In this mode, the address of operand is specified by a register pair. (HL in this case)
- MOV is the operation.
- M is the memory location specified by H-L register pair.
- D is the destination.

Immediate Addressing Mode (MVI A, MVI B, ADI, SBI, ACI etc.)

- In this mode, the operand is specified within the instruction itself. Example MVI C 28H
- MVI is the operation.
- 28 H is the immediate data (source).
- C is the destination

Implicit Addressing Mode

- If address of source of data as well as address of destination of result is fixed, then there is no need to give any operand along with the instruction. (Examples CMA, RAL, RAR etc.)
- CMA is the operation.
- A is the source.
- A is the destination.

Addressing mode समझने के लिये एक छोटा सा example लेते हैं—मान लीजिये कि आपको किसी मित्र को एक Book देनी है, अब आप यदि यह Book सीधे उसके हाथ में पकड़ा दें तो यह हुई immediate addressing, यदि आप उसके हाथ में एक address slip पकड़ा दें कि इस address से जाकर book ले लो, तो यह हुई direct addressing, यदि आप उससे कहें कि जाओ अलमारी में एक address slip रखी है, उस address slip पर जो address लिखा है वहाँ से book ले लो तो यह हुई Indirect addressing. यदि आपके कहने मात्र से ही वह समझ जाये कि book library में से मिलेगी तो यह है Implied या Implicit addressing यदि book अलमारी से मिलेगी तो यह है register addressing ।

Labels. Labels when given to any particular instruction/data in a program, takes the address of that instruction or data as its value. For example JMP PRACHI, here PRACHI is the label and it will take the address on the instruction where it is written. In fact, when you store a branch instruction, the branch address depends on that where you are storing the program. So, you do not know the branch address at the time of writing the assembly language program. Hence you have to use labels. Labels must always be placed in the first column and must be followed by an instruction (no empty line). Labels must be followed by a : (colon), to differentiate it from other tokens.

छद्म आदेश Pseudo-operations or Assembler directives.

There are some instructions in the assembly language program which are not a part of processor instruction set. These instructions are instructions to the assembler, linker and loader. These are referred to as **pseudo-operations** or as **assembler directives. examples : ORG, ASSUME, DD, DW, EQU, DB, DS, DQ, PTR etc.**

Arithmetic and logical unit (ALU) performs all arithmetic operations and comparisons for equality. In the Von Neumann's architecture, the ALU and the CU are separate components, but in modern systems they are integrated into the processor. The ALU has 3 sections, the register, the ALU circuitry, and the pathways in between. The register is basically a storage cell that works like RAM and holds the results of the calculations. It is mush faster than RAM. The ALU circuitry is that actually performs the calculations, and it is designed from AND, OR, and NOT gates just as any chip.

The control unit is responsible for fetching the next program instruction to be run from memory, decode it to determine what needs to be done, and then issue the proper command to the ALU, memory and I/O

controllers to get the job done. These steps are done continuously until the last line of a program is done, which is usually a HLT instruction.

Input-output devices is the subsystem that allows the computer to interact with other devices and communicate to the outside world. It also is responsible for program storage, such as hard drive control.

प्रोग्राम्ड इनपुट-आउटपुट (Programmed I/O)

Programmed Input/Output (PIO) refers to data transfers initiated by a CPU under driver software control to access registers or memory on a device.

The CPU issues a command then waits for I/O operations to be complete. As the CPU is faster than the I/O module, the problem with programmed I/O is that the CPU has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The CPU, while waiting, must repeatedly check the status of the I/O module, and this process is known as Polling. As a result, the level of the performance of the entire system is degraded.

Programmed I/O basically works (कार्यप्रणाली) in these ways :

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

Interrupt (अर्थात् व्यवधान). The CPU issues commands to the I/O device then proceeds with its normal work until interrupted by I/O device on completion of its work.

For input, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports, memory mapping.

For output, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA (Direct Memory Access)-capable devices usually generate interrupts to tell the system they are done with the buffer.

Although Interrupt relieves the CPU of having to wait for the devices, but it is still inefficient in data transfer of large amount because the CPU has to transfer the data word by word between I/O module and memory.

Basic operations of Interrupt are :

- CPU issues read command.
- I/O module gets data from peripheral whilst CPU does other work.
- I/O module interrupts CPU.
- CPU requests data.
- I/O module transfers data.

मैमोरी तक सीधे पहुँच बनाना (Direct Memory Access) (DMA) :

Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without its involvement. DMA module controls exchange of data between main memory and the I/O device. Because of DMA device can transfer data directly to and from memory, rather than using the CPU as an intermediary, and can thus relieve congestion on the bus. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

Direct Memory Access needs a special hardware called DMA controller (8257) (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and

destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.

DMA increases system concurrency by allowing the CPU to perform tasks while the DMA system transfers data via the system and memory busses. Hardware design is complicated because the DMA controller must be integrated into the system, and the system must allow the DMA controller to be a bus master. Cycle stealing may also be necessary to allow the CPU and DMA controller to share use of the memory bus.

विचार प्रश्न

(i) How many operations are there in the instruction set of 8085 microprocessor?

(a) 74 (b) 76
(c) 78 (d) 80

(ii) Address lines in a 4096 X 8 EPROM chip :

(a) 12 (b) 15
(c) 18 (d) 20

(iii) Control signals used for DMA operation are :

(a) HOLD & $\overline{\text{HLDA}}$ (b) RIM AND SIM
(c) Read and Write (d) INTR AND $\overline{\text{INTA}}$

प्रश्नावली

1. Select the correct Alternative :

(a) IfA = 22 H and B = 38 H,the rusult of ADD B instruction will be :
(a) 5 A (b) 60 (c) 40 (d) 80.

(b) IfA = 33 H and B = 04 H then the result of SUB B instruction will be :
(a) 29 (b) 2F (c) 39 (d) 3F.

(c) Which of the following instruction compulsorily sets the zero flag :
(a) ORA A (b) ANA A (c) XRA A (d) CMA.

(d) Which of the following instruction is invalid :
(a) LDA 5500 (b) MOV B, M (c) MOV B, B (d) LDAXH.

(e) Which of the following is a 3-byte instructions :
(a) MVI (b) LDAX (c) LXI (d) DAD

(f) If A = 90, the DCR A will result in :
(a) 89 (b) 8F (c) 69 (d) 91.

(g) Which of the following instructions does not changes the contents of A register :
(a) CMA (b) RAR (c) CMP B (d) RIM.

(h) Which of the following is an 8-bit register :
(a) SP (b) PC (c) IR (d) None of these.

(i) Which of the following instruction is equivalent to a left shift :
(a) RAR (b) ADD A (c) ADD C (d) CMA.

2. Answer the following :

(i) IfA = 63 H, B = 54 H, C = 1 then what will be the contents of A and B after the execution of :

(a) ADD B (b) SUB B (c) CMA (d) CLA
(e) CMP B (f) RAL (g) RAR (h) RLC
(i) RAL (j) ANA B (k) ORA B (l) XRA B
(m) ANA A (n) ORA A (o) XRA A (p) ADC A

(q) ADC B (r) SBB A (s) SBB B (t) INRA
(u) INR B (v) DCR A (w) DCR B (x) ANI A 00H
(y) ANI A FFH (z) ORI A FFH.

(ii) If A = 54 H and B = 63 H, C = 1 what will be the contents of A and B after the excution of following instructions :

(a) ADD B (b) SUB B (c) CMA (d) CLA
(e) CMP B (f) RAL (g) RAR (h) RLC
(i) RAL (j) ANA B (k) ORA B (l) XRA B
(m) ANA A (n) ORA A (o) XRA A (p) ADC A
(q) ADC B (r) SBB A (s) SBB B (t) INR A
(u) INR B (v) DCR A (w) DCR B (x) ANI A 00H
(y) ANI A FFH (z) ORI A FFH.

(iii) Also find out the contents of flag register in each of the case give in part (i) and part (ii) above.

9 Chapter प्रयोगात्मक (MS विन्डोज़ का परिचय)
PRACTICALS (AN INTRODUCTION TO MS WINDOWS)

THINK ABOUT IT

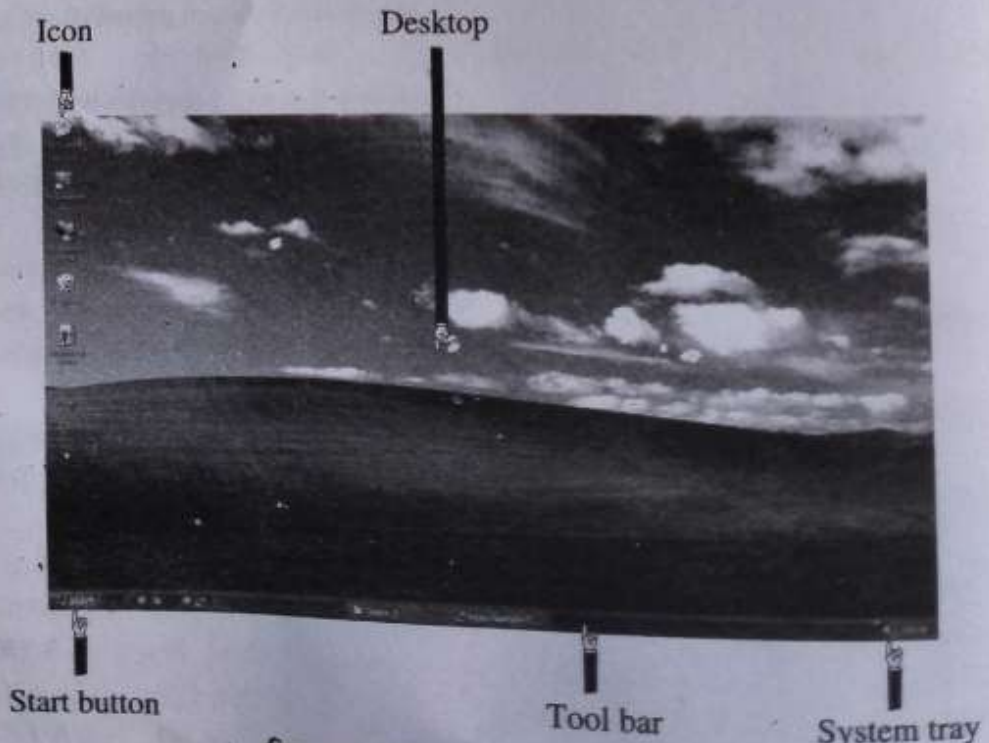
The only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. — **Steve Jobs**

Life is like photography. You need the negatives to develop. - *Unknown*

You need chaos in your soul to give birth to a dancing star. — **Friedrich Nietzsche**

Microsoft Windows (जिसे MS Windows भी कहा जाता है) एक GUI (Graphical User Interface) operating system है, जिसे microsoft द्वारा विकसित किया गया। इसमें आप icons (आइकन अर्थात् छोटे-छोटे ग्राफिकल चित्रों) पर क्लिक करके प्रोग्रामों को क्रियान्वित कर सकते हैं। इसमें Windows icons व menus की सहायता से commands दी जा सकती हैं। कम्प्यूटर को बूट करने से लेकर एप्लीकेशन साफ्टवेयर प्रयोग करने जैसे सभी कार्य विन्डोज़ द्वारा ही किये जाते हैं। इंटरनैट का प्रयोग भी विन्डोज़ माहौल (Windows environment) से सर्वाधिक है।

Microsoft Windows का पहला version (संस्करण Version 1.0) नवम्बर, 1985 में रिलीज़ किया गया। MS Windows 2.0 दिसम्बर 1987 में रिलीज़ हुआ। MS Windows 3.0 1990 में रिलीज़ हुआ। इसमें multitasking व virtual memory जैसी सुविधायें उपलब्ध थी। उसके पश्चात् Windows के कई अन्य version रिलीज़ हुये जैसे कि Windows 95



चित्र 9.1—Components of a Window

(अगस्त 1995 में) Windows 98 (जून 1998 में), Windows XP (2001 में) तथा Windows Vista (नवम्बर 2006 में)। Windows 7 Home Premium, Windows 7 Professional, Windows 7 Ultimate and Enterprise.

प्रयोग संख्या 1

उद्देश्य (Object)—

To study the various components of Windows (विन्डोज़ के विभिन्न घटकों का अध्ययन करना)।

वर्णन (Description)—

डेस्कटॉप (Desktop):

MS विन्डोज में सबसे पहले प्रकट होने वाली स्क्रीन को डेस्कटॉप कहा जाता है। जिस प्रकार वास्तविक जीवन में हम अपने कार्य की चीजे desktop पर रखते हैं, उसी प्रकार Windows के desktop पर icons, tool bars, folders, इत्यादि रखे जाते हैं। यह drag and drop feature तथा अन्य सुविधायें प्रदान करता है।

Desktop is the first screen that appears on MS Windows X. In real life, all the utility items are placed on desktop. Similarly a desktop environment typically consists of icons, toolbars, folders, Windows, desk accessories. It also provides drag and drop functionality and other features which make it useful.

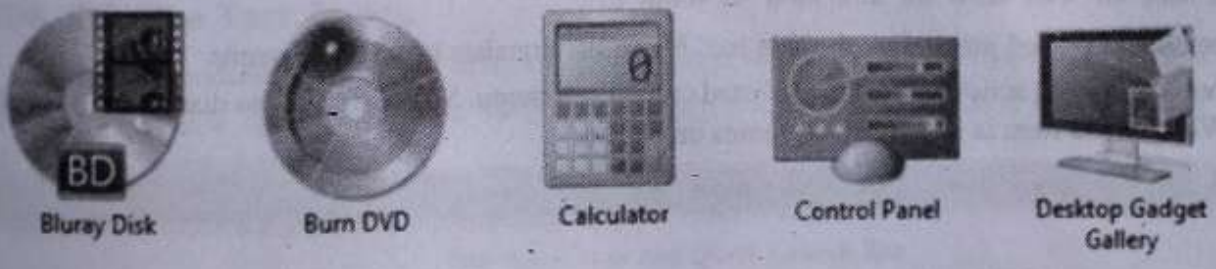
Icon:

आइकन अर्थात् एक छोटा सा ग्राफिकल symbol जो कम्प्यूटर पर उपलब्ध function के उद्देश को दर्शाता है।

A computer icon is a small pictorgram used as a tool for making computer interfaces easy. Icon may represent a file, folder, application or device on a computer operation system. It is a small graphical symbol which represents the purpose of available function on a computer.

Icons are found on desktops, toolbars and in the menus of computer application software such as Microsoft Word. Icons are very user friendly by being very different from each other. Each icon set have unifying features. Icons show these by-contrasting sizes, composition, pattern contrast, light on dark, dark on light, framed/shadowed, color contrasts, animation.

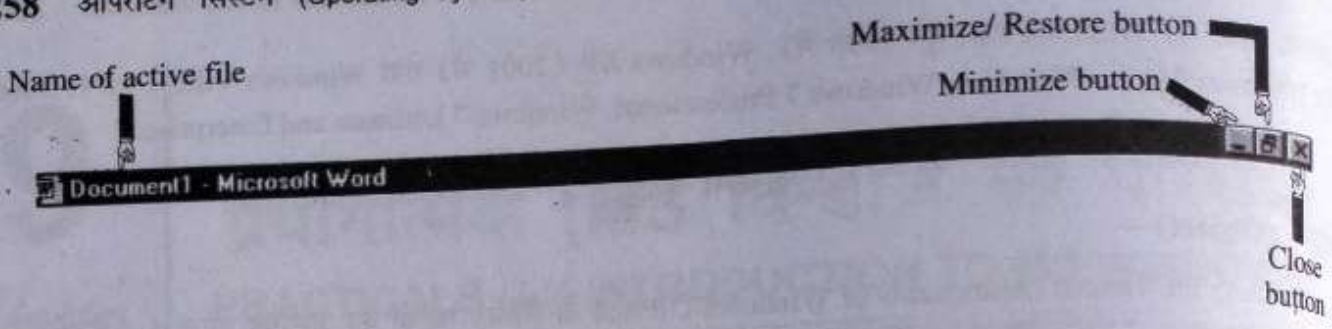
On this screen icons are used in many ways—to represent files, folders, disk drives, toolbar buttons, menu items and taskbar items.



चित्र 9.2—आइकन्स

टाइटल बार (Title Bar) :

विन्डोज़ की सबसे ऊपरी पट्टी को टाइटल बार कहा जाता है (चित्र 9.3)। इसमें program का नाम तथा खुली हुई विन्डो का नाम होता है। इसके दाहिने साइड में minimize (न्यूनतम करना), maximize (अधिकतम करना), close (बंद करना) व restore (वापस उसी size पर लौटना) के बटन होते हैं। टाइटल बार को विन्डो को move करने हेतु हैंडल के रूप में भी use किया जा सकता है। टाइटल bar पर pointer को ले जाये, mouse button को hold कर के रखें, तथा move करें, तो Window भी move करेगी। जब Window वांछित स्थान पर पहुँच जाये, तो buttorn को release कर दें।



चित्र 9.3—टाइटल बार

The title bar shows the name of the current program and file. When we start to create a new file, the title bar displays a generic default file name, such as untitled, document 1 etc. The Window with highlighted title bar is the active Window. The title bar can serve as a handle for moving a Window around the screen. Point at the title bar, hold down the mouse button and move the mouse around, the Window moves as you move the mouse. Release the mouse button and the Window stays where you finally stopped.

Minimize, Maximize/Restore and Close Buttons :

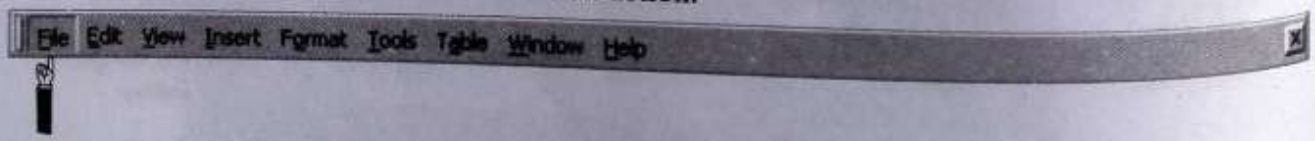
- **Minimize Button:** The minimize button is in the upper right corner of the Window. Minimising a Window does not destroy its content, but transforms the Windows into a button on the taskbar. To make the button on the taskbar turn back into an on screen Window, click on it. The buttons revert to a Window in the same size and location as it was before.
- **Maximize Button:** A click on the maximize button makes the Window occupying maximum space on screen as possible. The shortcut key for maximise is: Alt + Spacebar + X to maximize the Window.
- **Restore Button:** In the right upper corner of every maximized Window is the restore button. A click on this button restores (returns) the Window to the size it was maximized. The shortcut key is: press Alt + Spacebar, then + R.
- **Close Button:** The Close button is on the extreme right of title bar. It closes the specific Window.

मैनू बार (The Menu Bar) :

मैनू बार टाइटल बार के नीचे होती है तथा इसमें विभिन्न मेनूज़ होती है जिसमें विभिन्न कार्यों की सूची होती है। सूची में से वांछित कार्य का चयन करके वह कार्य किया जा सकता है।

Menu bar is placed just below the title bar. Menu bar consists of various menus.

Groups of related actions on tasks are listed under each menu. Selecting a menu displays the list of menu items. When menu item is selected it executes an action.

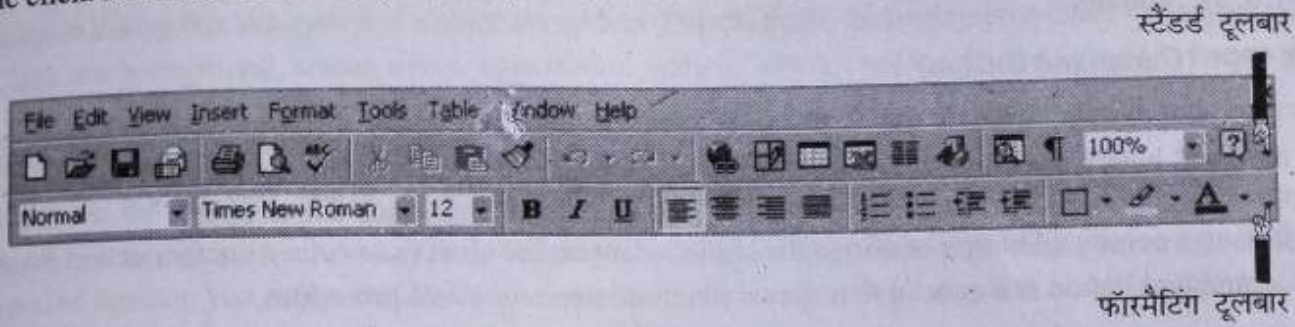


चित्र 9.4—Menu Bar

स्टैंडर्ड टूलबार (Standard Toolbar) :

स्टैंडर्ड टूलबार सामान्यत:, स्टैंडर्ड मैनूबार के नीचे होता है। इसमें विभिन्न buttons होते हैं। इन buttons की सहायता से कार्य तेजी से एक क्लिक द्वारा किये जा सकते हैं।

The standard tool bar normally displayed under menu bar. Standard toolbar consists of a number of related buttons plunked side by side. The buttons enable you to perform frequently used tasks speedily on single click. It consists of New, Open, Save, Print, Print Preview, Undo, Redo etc.



चित्र 9.5—Standard Toolbar

फारमैटिंग टूलबार (Formatting Toolbar) :

फारमैटिंग टूलबार स्टैंडर्ड टूलबार के नीचे होता है, इसमे commands के shortcuts हेतु buttons का use किया जाता है।

Formatting toolbar is placed under the standard toolbar. It consists of buttons used as shortcuts to commands that enlarge the appearance of any document and formatting text (font style, font size, bold, underline, italics etc.)

स्क्रॉल बार (The Scroll Bar) :

स्क्रॉल बार विंडो के एक छोर पर होता है। स्क्रॉल बाक्स की सहायता से page को ऊपर/नीचे तथा दायें/बायें move किया जा सकता है। स्क्रॉल बाक्स को देख कर यह अंदाजा लगाया जा सकता है कि वर्तमान में आप document के top, middle या bottorn के निकट स्थित है।

इस प्रकार Window के नीचे horizontal scroll box से page के दायें/बायें movement किया जा सकता है।

The scroll bar is along the edge of a Window. The vertical scroll bar travels up and down when to work is paged. By glancing at the scroll box, we can tell whether you are near the top of a document, middle or at the bottom of page. Scroll bar that run along the bottom of a Window is horizontal scroll bar; it moves your view from left to right then up and down. Shortcut key to view the top of your document, Ctrl + home, To see the bottom, Ctrl + End or press the PgUp or PgDn key to respectively move one page or down at a time.

टास्क बार (The Task Bars) :

टास्क बार Windows के नीचे पट्टी के रूप में (start button की right में) होता है तथा यह दर्शाता है कि इस समय कौन-कौन सी विन्डोज़ open हैं। अत: एक विन्डो से दूसरी विन्डो में स्विच करने हेतु task bar पर वांछित Window के नाम पर click करना होता है।

चित्र 9.6—Task and Quick Launch Bar

The task bar lies along the bottom of the screen and shows miniature version of the Windows that are currently open. Hence it keeps the trace of the action it is on the right of the start button. To switch from one Window to another, all what is needed is to click the desired Window's name on the task bar.

क्विक लॉंच बार (The Quick Launch Bar) :

क्विक लॉंच बार टास्क बार पर start button के right में होता है। इसमें icons पर click करके programs को start किया जा सकता है।

The quick launch bar lives on the task bar next to the start button. The icons represent programs you can start by clicking them.

डॉयलॉग विन्डो (The Dialog Window) :

The main types of buttons in the MS Windows are—command button, radio button, checkbox, drop-down list, tab, spinner.

कमान्ड बटन (Command Button) :

कमान्ड बटन डॉयलाग बाक्स में होते हैं तथा प्रदत्त सूचना के आधार पर कमान्ड एक्सीक्यूट करते हैं।

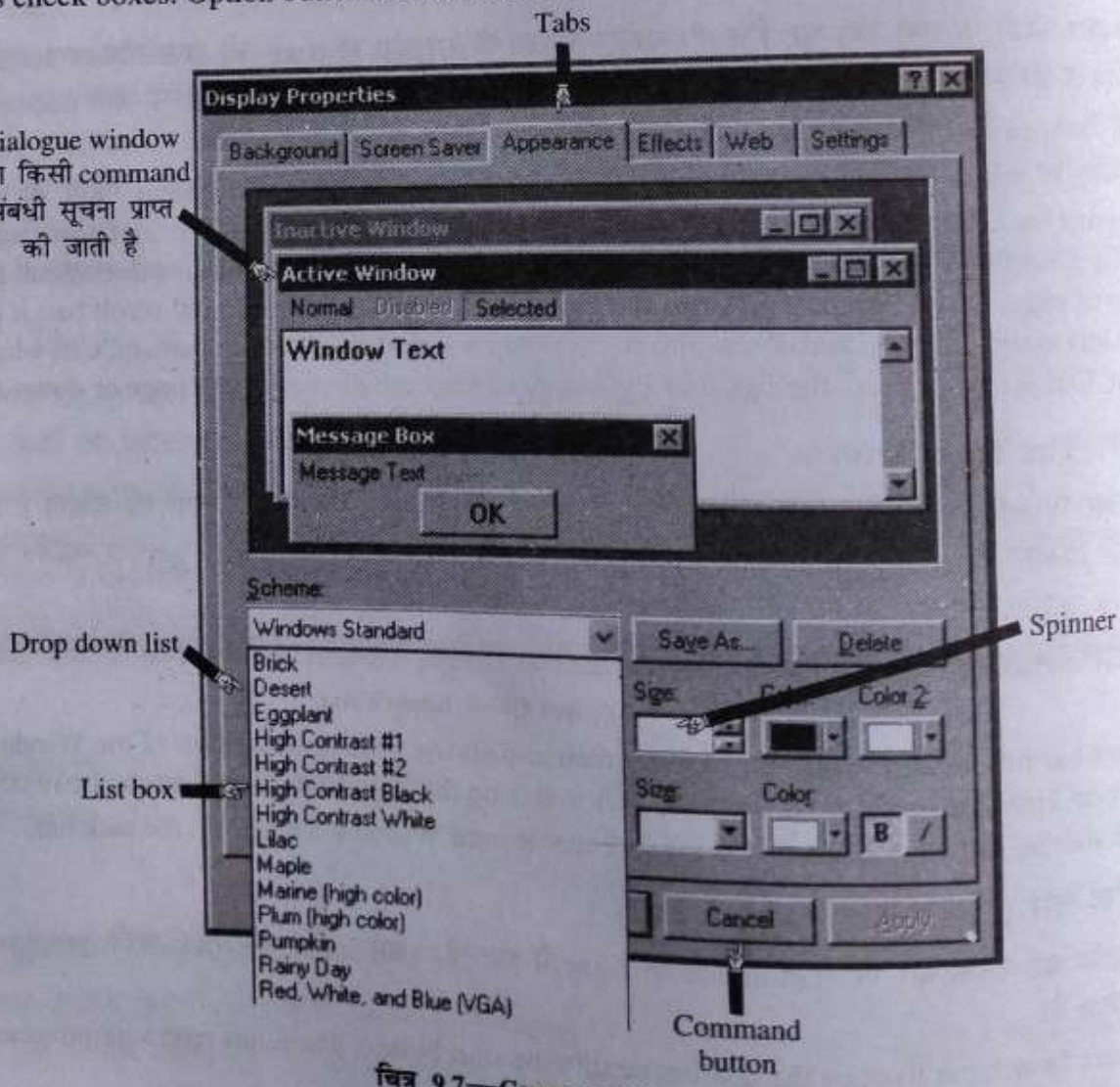
Command buttons are found in dialog boxes and execute a command with information provided. For example—when you click a file to open a dialog box appears asking which file to open.

Clicking a command button confirms the choices cancels the choices or causes another action. In other words, command button is a control that runs a function macro or event procedure.

रेडियो बटन (Radio Button) :

रेडियो बटन एक स्टैंडर्ड विंडो कन्ट्रोल है जिसकी सहायता से user कई mutually exclusive विकल्पों में से किसी एक का चयन कर सकता है।

Radio button is a standard Window control that allows the user to select from a fixed set of mutually exclusive options. To choose from several choices in an option box, it lets you select only one of them. It moves the dot back and forth between the options as your decision varies. Click the OK button, when you have reached a decision, the dotted option then takes effects. If you need to choose more than one option, then it offers check boxes. Option button is also refferred to as a radio button.



चित्र 9.7—Command Button

टेब (Tab) :

डॉयलॉग बाक्स में टैब एक समान properties का समूह होता है।

A tab in a dialog box is a group of similar properties. For example, in display properties dialog box, the various tabs are-background, screen saver, appearance, setting, which can be selected using tab button.

ड्रॉप डाउन लिस्ट (Drop-down List) :

Drop-down list वह सूची होती है, जिसमें से आप कई विकल्पों में से किसी एक का चयन कर सकते हैं।

Drop-down list is a list of items from which you can make selections. For example, to choose the appearance of desktop you select the appearance tab of display properties dialog box and then select color scheme, click drop down list button to view and select from the list.

स्पिनर (Spinner) :

स्पिनर एक ऐसा text box होता है, जिससे आप value को घटा या बढ़ा सकते हैं।

A spinner is a text box that allows you to increase or decrease the value. For example, in screen saver tab of display properties dialog box, you may choose the minutes in the wait box.

चैक बॉक्स (Check Box) :

चैक बॉक्स एक कन्ट्रोल है, जिसकी सहायता से user किसी option को activate कर सकता है।

Check box is a control that allows the user to activate an option. When you click in the box a ü shows that option is activated.

शॉर्टकट्स (Shortcuts) :

आप एक शॉर्टकट बनाकर उसे एक icon को assign कर सकते हैं। शॉर्टकट पर डबल क्लिक करके आप सीधे वह file लोड कर सकते हैं।

You can create a shortcut and assign it to an icon. When you double click the shortcut icon, MS Windows immediately takes you there and loads the file.

To view them click the start button, choose documents and you will see the list. To create a shortcut say for paint, select *Start > Programs > Accessories*. Right click on Paint, a shortcut menu appears, select *Create Shortcut*. A shortcut for that program will be created.

Windows automatically makes a shortcut to the last 15 documents you have opened.

प्रयोग संख्या 2—विन्डो को एक स्थान से दूसरे स्थान पर move करना (Moving a Window from One Place to Other)

Window क्या है? (What is a Window)—Window एक आयताकार visual area है जिसमें कम्प्यूटर में रन करने वाले विभिन्न process की output display होती है तथा इनपुट देखी जा सकती है। Windows को resize (आकार परिवर्तित करना), move (एक जगह से दूसरी जगह ले जाना), hide (छुपाना), restore (पुन: उसी आकार में लाना), close (बन्द करना) आदि किया जा सकता है।

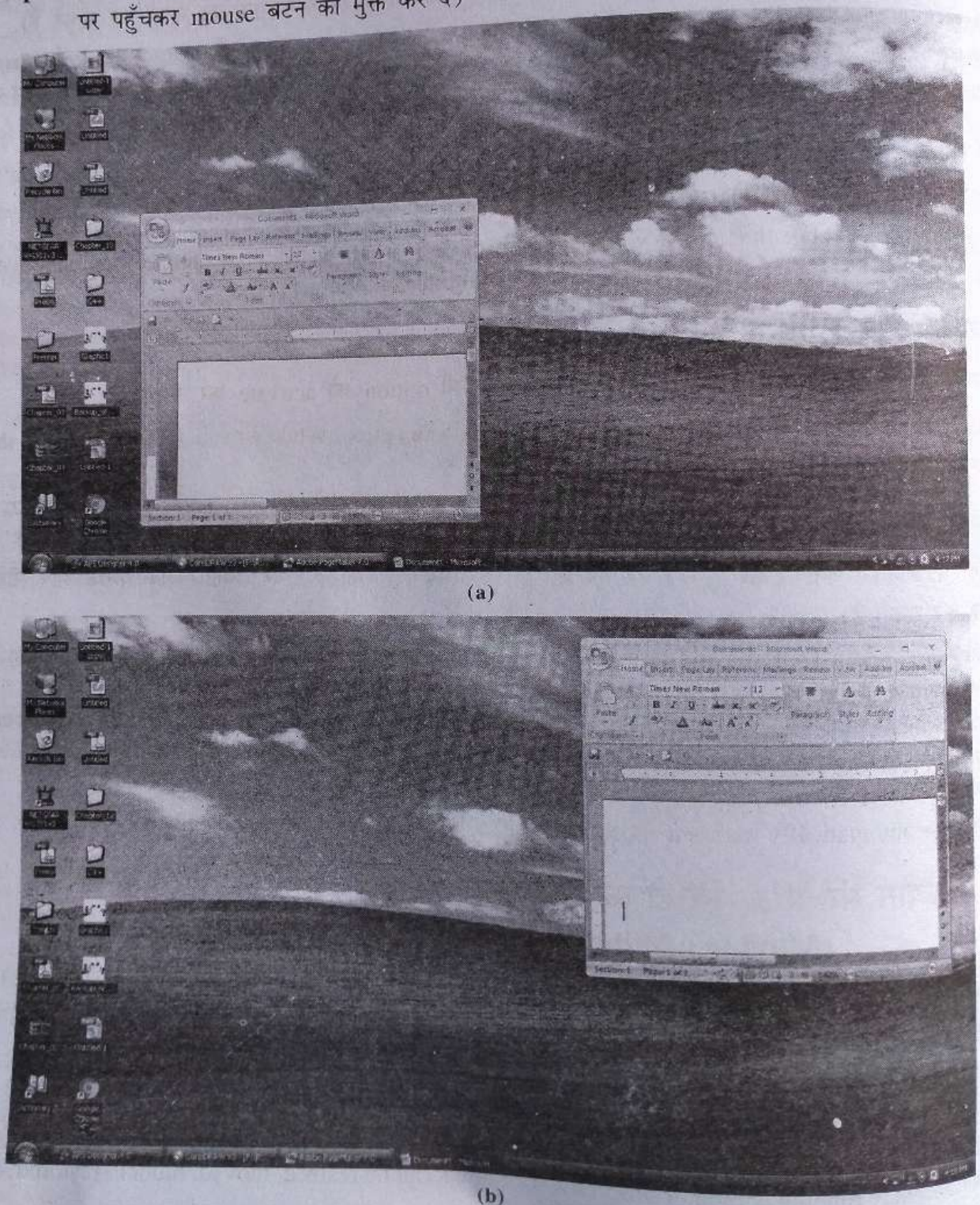
A Window is a rectangular visual area (a graphic display) displaying output and allowing input for one of a simultaneously running computer processes. Windows can be resized, moved, hidden, restored, closed as desired.

विधि (Procedure) :

A window can be moved from one part to another on screen.

Step 1: Place the mouse pointer on the title bar and drag the Window to the desired location. (Title bar पर pointer को रखें, एवं विंडो को वांछित स्थान पर ड्रैग करें (अर्थात् mouse का button दबा कर, move करें)

Step 2: Release the mouse button after the Window has been dragged to a new location. (वांछित स्थान पर पहुँचकर mouse बटन को मुक्त कर दें)



(a)

(b)

चित्र 9.8—Moving the Window

प्रयोग संख्या 3—Increasing or Decreasing the Size of Window (Window के size को कम या अधिक करना)

विधि (Procedure) :

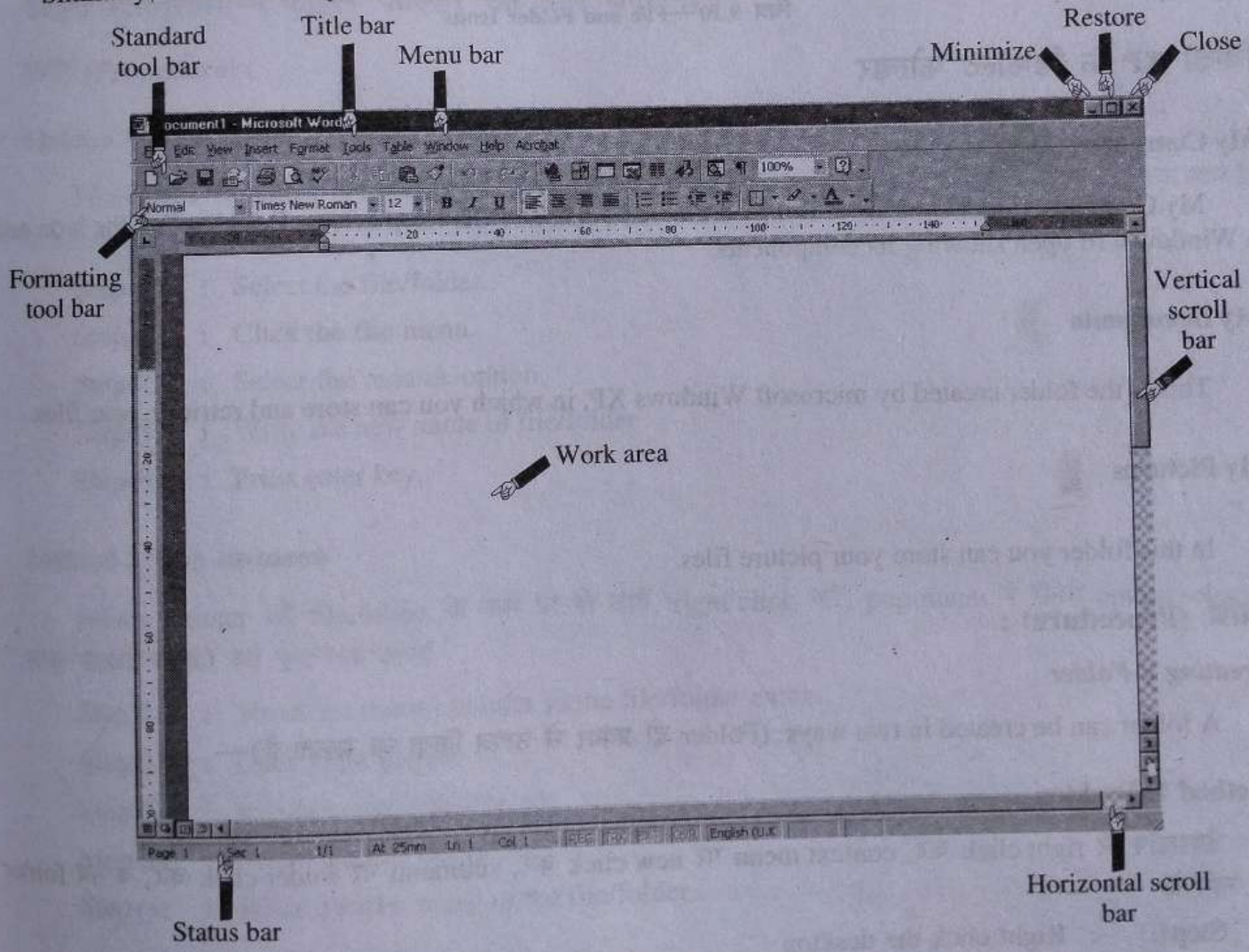
विन्डो के size को निम्नवत् increase या decrease किया जा सकता है—

Step 1 : Place the mouse pointer to the edge of Window. Mouse pointer changes to a double-headed arrow (Mouse pointer को Window के कोने पर रखें mouse pointer एक double head arrow बन जायेगा)।

Step 2 : Click and drag the mouse to increase or decrease the size of the Window (Click तथा drag करके Window का size घटायें या बढ़ायें)।

Step 3 : Release the mouse button when the desired size is obtained (वांछित साइज प्राप्त होने पर mouse के button को मूक्त कर दें)।

Similarly, follow the steps to increase or decrease the size of the Windows either towards left or right.



चित्र 9.9—Scroll Bars and Boxes

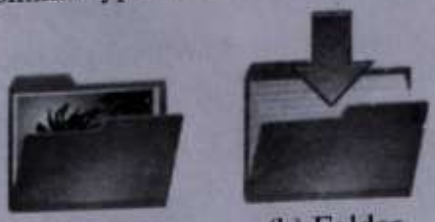
प्रयोग संख्या 4

उद्देश्य (Object) : Folder को उत्पन्न करना (To Create a Folder)

File तथा फोल्डर—

File is the collection of information. Files are stored in memory. A file is referred to by its file name. The file name is given to each file that helps to remind the contents and the file extension describes the type of file. A file name can have up to 255 characters including a period (.) and file extension up to 8 characters.

Folder is a collection of files of similar type. A folder can have subfolders in it.



(a) File (b) Folder

चित्र 9.10—File and Folder Icons

विन्डो XP के डिफाल्ट फोल्डर

My Computer 

My Computer folder consists of floppy drive, hard disk and CD Rom drives. Double click the icon and a Window will open showing its components.

My Documents 

This is the folder created by microsoft Windows XP, in which you can store and retrieve your files.

My Pictures 

In this folder you can store your picture files.

विधि (Procedure) :

Creating a Folder

A folder can be created in two ways: (Folder दो प्रकार से उत्पन्न किया जा सकता है)—

Method 1 : Desktop

डैक्सटॉप पर right click करें, context menu पर new click करें, submenu पर folder click करें, व नये folder को नाम दें

Step (i) : Right click the desktop

Step (ii) : Select new option from the context menu and a sub-menu appears.

Step (iii) : Click folder and name it.

Method 2 : Windows Explorer

Start button पर right click करें, explore पर click करें, start name Window खुलेगी, file name पर click करें, new option को select करें, folder option पर click करें

Step (i) : Place the mouse pointer in start button and right click on it.

Step (ii) : Click explore.

Step (iii) : Start menu Window will be opened.

Step (iv) : Click the file menu and select the new option.

Step (v) : Click the folder option.

प्रयोग संख्या 5

उद्देश्य (Object)—(फाइल फोल्डर को नाम बदलना)

विधि (Procedure) :

Method 1 : File Menu

File/folder को select करें, फाइल मैनू पर क्लिक करें, रीनेम option select करें, फाइल /फोल्डर का नया नाम लिखे, व enter key को press करें,

Step (i) : Select the file/folder.

Step (ii) : Click the file menu.

Step (iii) : Select the rename option.

Step (iv) : Write the new name of file/folder

Step (v) : Press enter key.

Mehtod 2 : Pop-up menu

Mouse pointer को file/folder के नाम पर ले जायें, right click करें, pop-menu में रीनेम option select करें, तथा फाइल/फोल्डर का नया नाम लिखें

Step (i) : Move the mouse pointer to the file/folder name.

Step (ii) : Click right mouse.

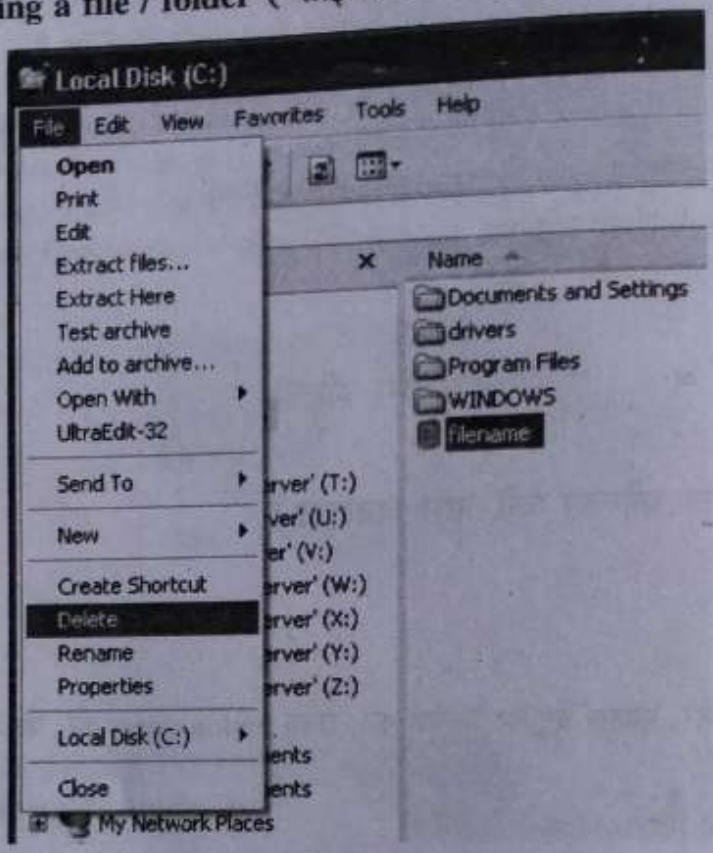
Step (iii) : Pop-menu gets displayed.

Step (iv) : Select rename option.

Step (v) : Write the new name of the file/folder.

प्रयोग संख्या 6

उद्देश्य (Object) : Deleting a file / folder (फाइल/फोल्डर को डिलीट करना)



चित्र 9.11

विधि (Procedure) :

फाइल को select करके delete button press करें, या file पर mouse pointer को ले जाकर right click करें, pop-up menu में delete पर click करें,

Step (i) : Select the file/folder.

Step (ii) : Press delete key.

Step (iii) : A Window will ask you: Are you sure that you want to delete the file, click on the "yes" button.

Step (iv) : Your file will be in recycle bin.

Step (v) : On right clicking on the file name in recycle bin, you will again be asked whether you want to permanently delete the file. By clicking yes, you can do the same.

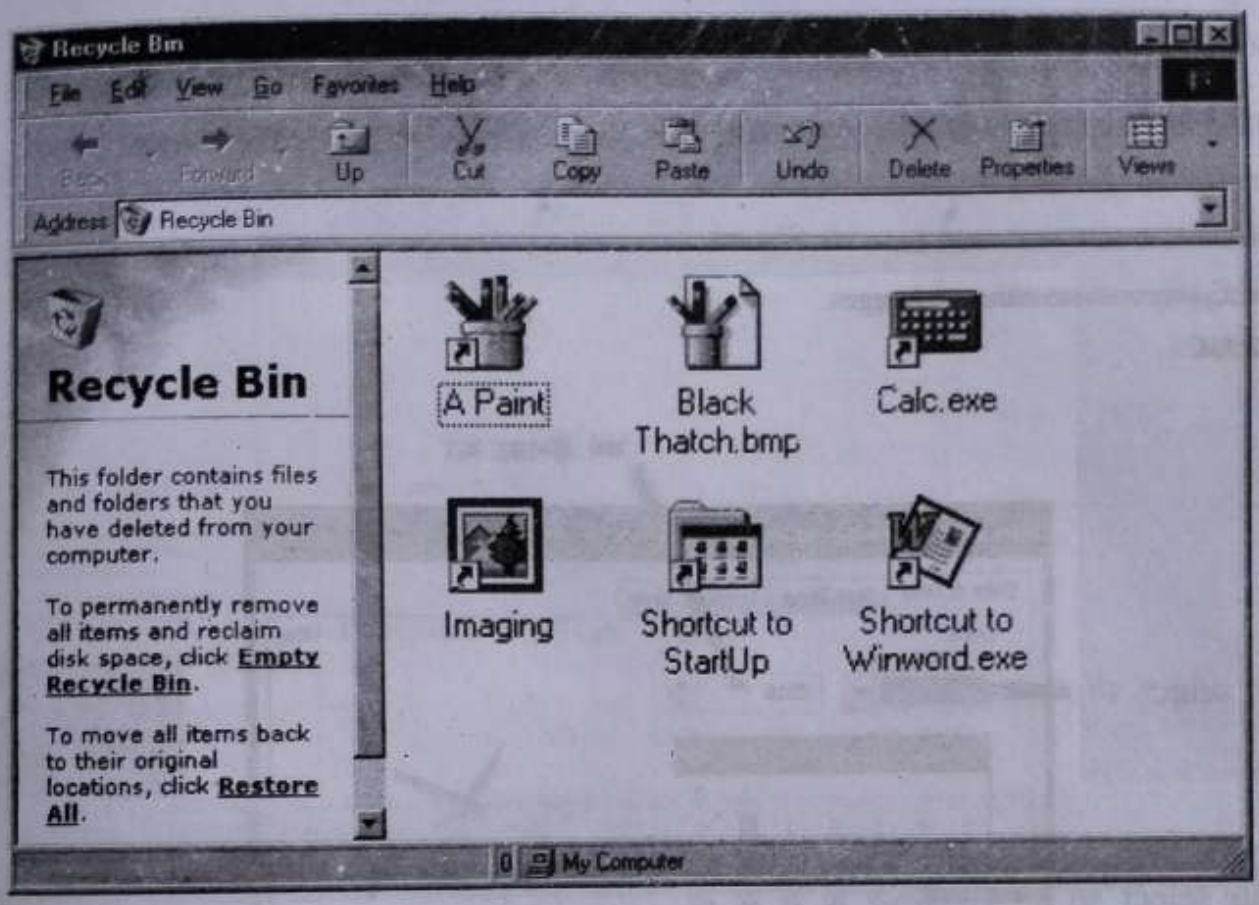
Method 2:

Step (i) : Move the mouse pointer on file/folder.

Step (ii) : Right click on tile and a pop-up menu will be displayed.

Step (iii) : Click the delete option.

Recycle Bin :



चित्र 9.12

When a file or folder is deleted, it gets stored in the Recycle Bin. You can easily restore the deleted files from Recycle Bin, by dragging the file or folder out of the Recycle Bin Window.

प्रयोग संख्या 7

उद्देश्य (Object) : Copying a File/Folder and Pasting it somewhere else (फाइल/फोल्डर को कॉपी करना) (अर्थात् नकल बनाना) तथा अन्यत्र कहीं पेस्ट करना)

File/folder को select करें, right click करें, copy option select करें, अब उस लोकेशन पर जायें, जहां pasting करनी है, right click व paste option का use करें, pasting करें।

Step (i) : Select the file/folder you want to copy.

Step (ii) : Righ click the file/folder icon and a pop-menu appears

Step (iii) : Choose the ***Copy*** option. Your file or folder is copied.

Step (iv) : Move the pointer where you want to paste double click to open the desired folder.

Step (v) : Right click and choose paste option.

नोट—Copy करने हेतु short cut होता है Control + C, Paste करने हेतु Control + V, Cut करने हेतु Control + X.

प्रयोग संख्या 8

उद्देश्य (Object): Setting Date and Time (System date व time set करना)

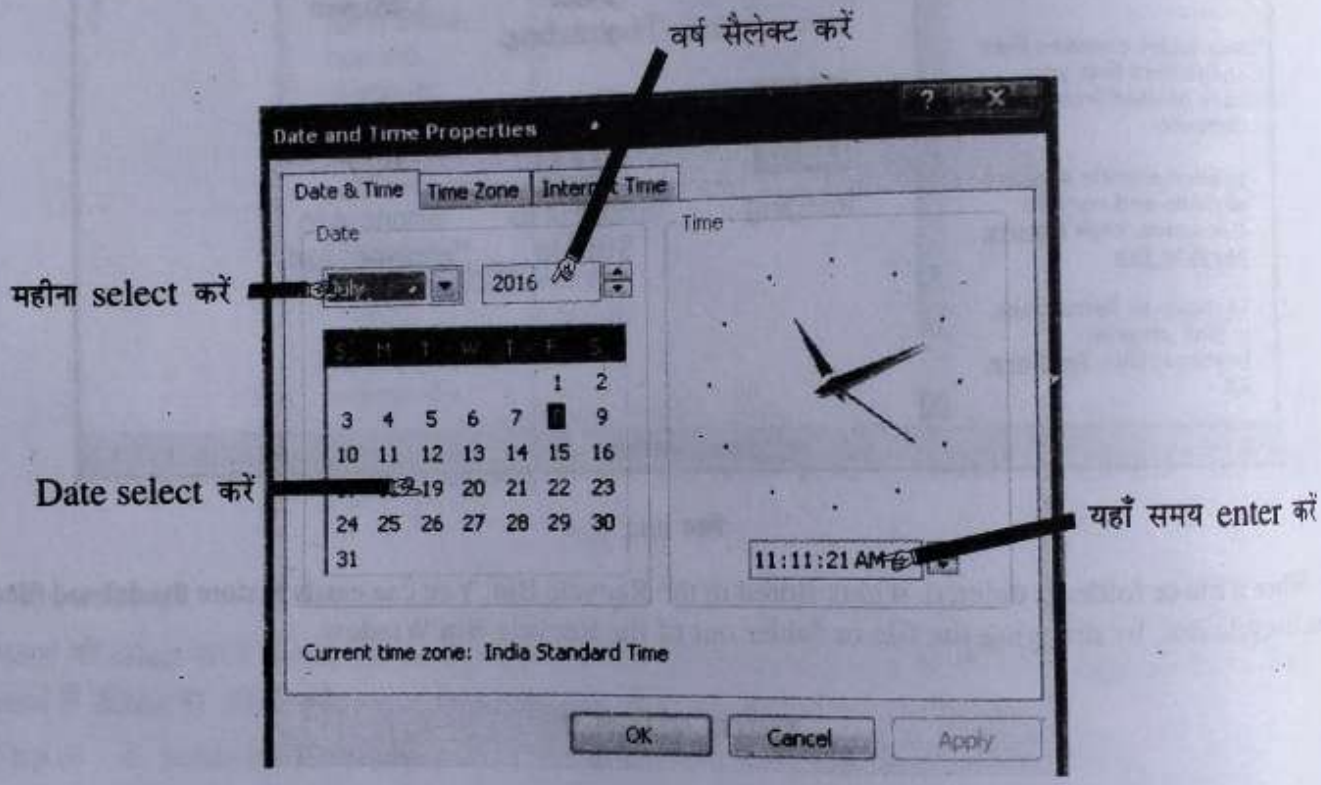
Click the time shown in the system tray (system tray में प्रदर्शित time पर click करें)

The folliwing dialog box is displayed.

अब date, month, year and time को चेंज करें।

Click *Apply* to save these changes.

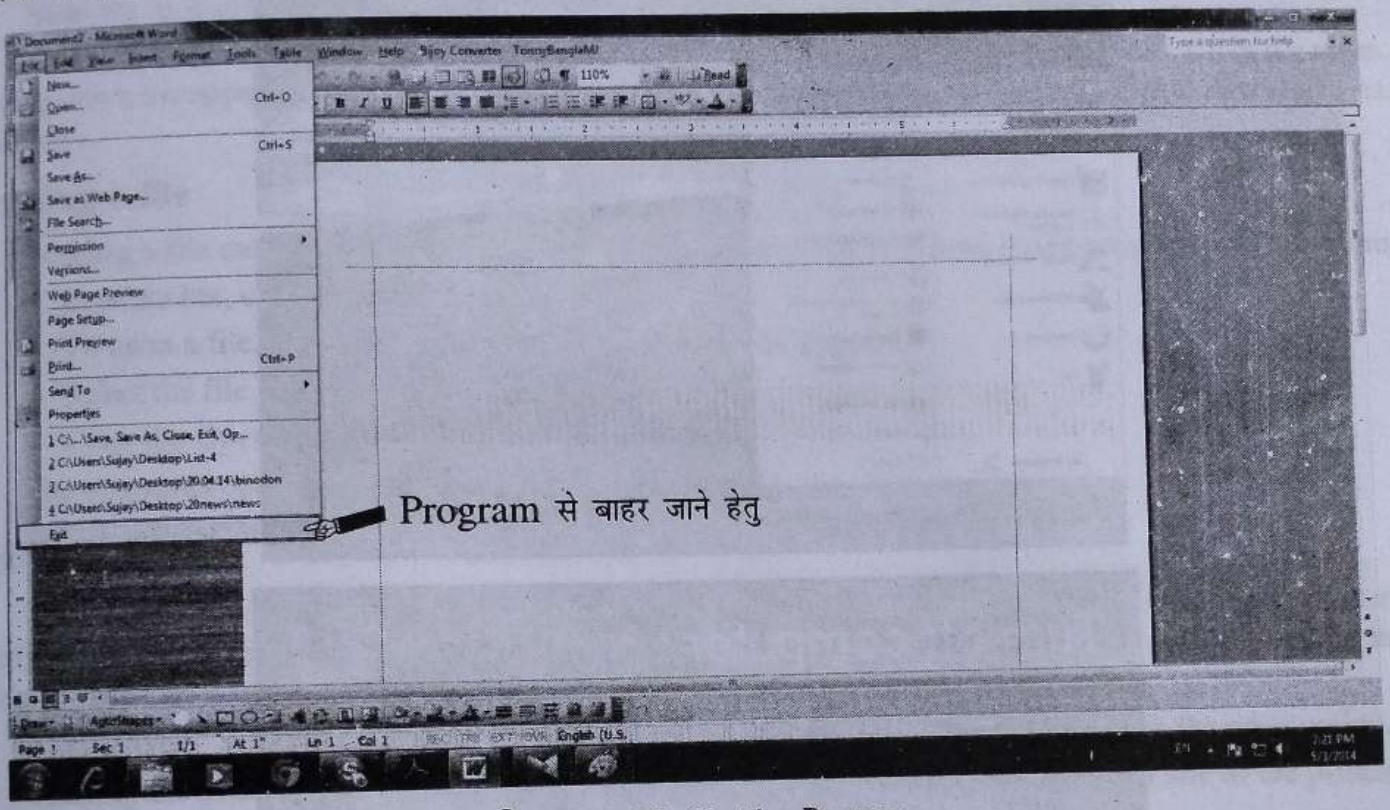
Click *OK*.



चित्र 9.13

प्रयोग संख्या 9

उद्देश्य (Object) : Exiting Programs and MS Windows and Shutting Down (प्रोग्राम से बाहर जाना तथा MS Windows से बाहर जाना व शट-डाउन करना)



चित्र 9.14—File Shutting Program

विधि (Procedure):

Close button को press करके, या Alt + F4 short cut का use करके प्रोग्राम से बाहर जा सकते हैं।

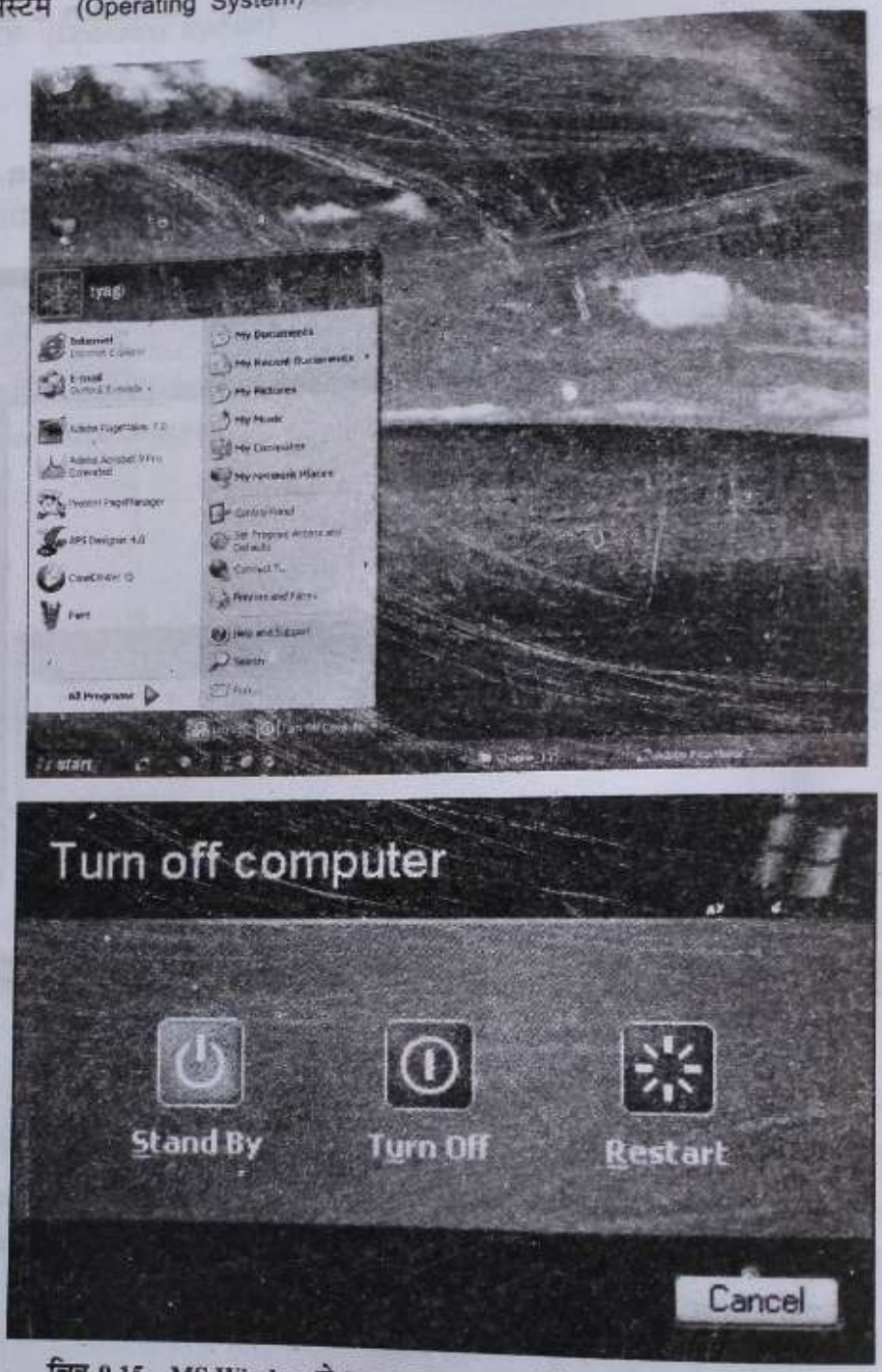
Exiting Windows is done by pressing Alt + F4 or click on the title × (cross box) in the upper right corner.

Program आप से पूछेगा कि क्या आप changes को save करना चाहते हैं। ''Yes'' press करने पर आपका work save हो जायेगा।

Exiting MS- Windows :

To quit MS Windows, click the start button and choose the shutdown or turn off computer option. A dialog box appears asking you to select one of the options.

Shutdown option logs you off the computers. The computers shutdown all currently running Window based programs asking whether you would like to save any unsaved work and then leaves a message on the screen. Turning off the computer is safe.



चित्र 9.15—MS Window से बाहर जाना तथा टर्मिनल को शट डाउन करना

Then you can switch off the UPS and then the main switch.

प्रयोग संख्या 10

उद्देश्य (Object) : To study the starting of MS Windows, Loading a file of MS word, Printing the work, Saving the work (MS Windows की starting का अध्ययन करना, MS word की फाइल लोड करना, प्रिंट करना तथा सेव करना)

Starting a Program

When MS-Windows starts, the system goes through its usual routine of checking to see that all of its parts are still present and working.

The user name dialog box appears, type the user name and type the password (if required) and click OK.

The first screen which appears is called desktop. Click the start button at the bottom left corner move the mouse pointer over the menus to see more button. The buttons have little pictures called icons. An icon offers hints to the program it represents.

Slide the mouse over Accessories to see another menu, called the sub-menu.

To start Microsoft Word click start > All programs > Microsoft office > Microsoft word. Click file menu. A drop down list appears. The drop down list contains file related options, such as new, Open, Save, Save As, Print etc.

Loading a file

Loading a file means opening a file. To open a file in any MS Windows based programs, the steps are:

- In menu bar, click the *File* menu.
- To open a file, choose *Open* option. *Open* dialog box appears.
- Select the file name.
- Click the *Open* button, in the *Open* dialog box.

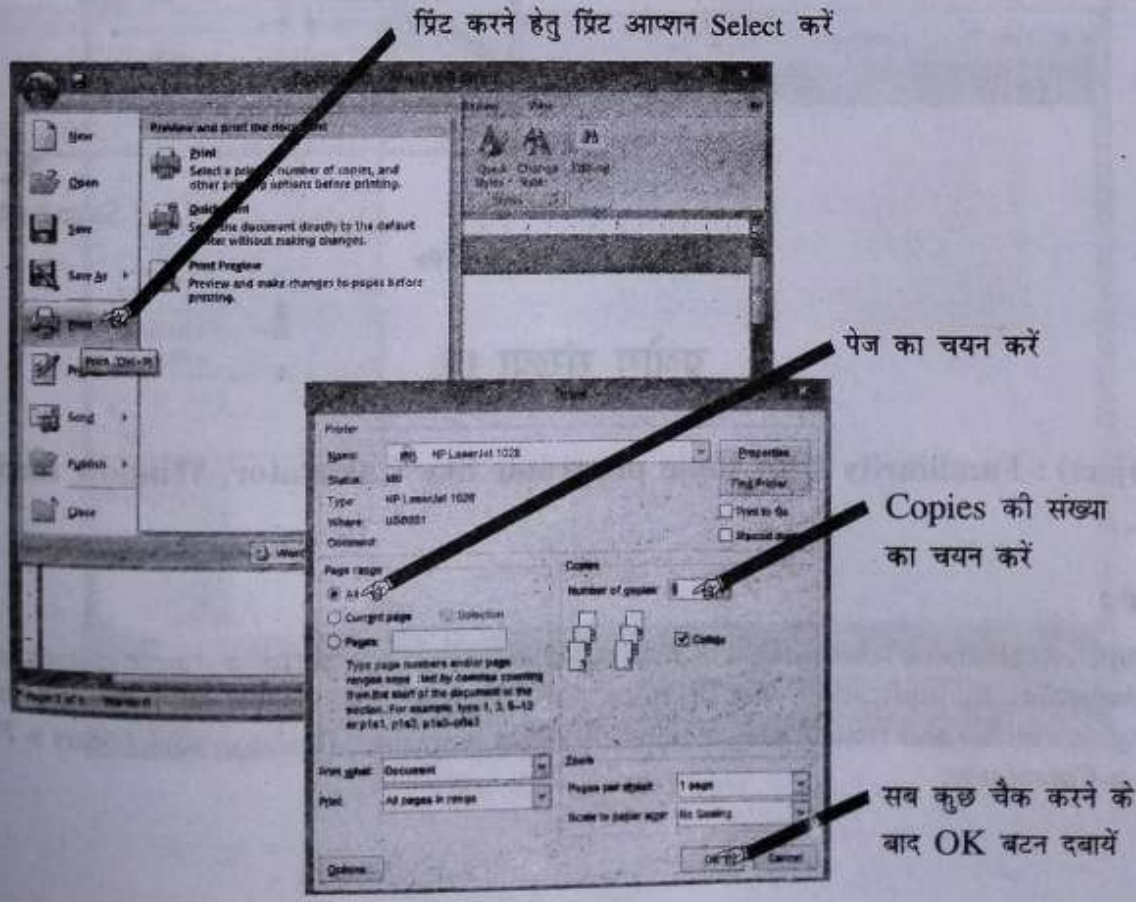
The programs open the file and displays it on screen.

Printing the file

To print you work in MS Windows based program you press Alt key and then press the letter *F*. It opens the file menu. Now press the letter *P* that tells the program to send its data to the printer (you can also use the short cut key Ctrl + P).

Alternatively, you can also use the mouse to click to open file menu and then click the print option.

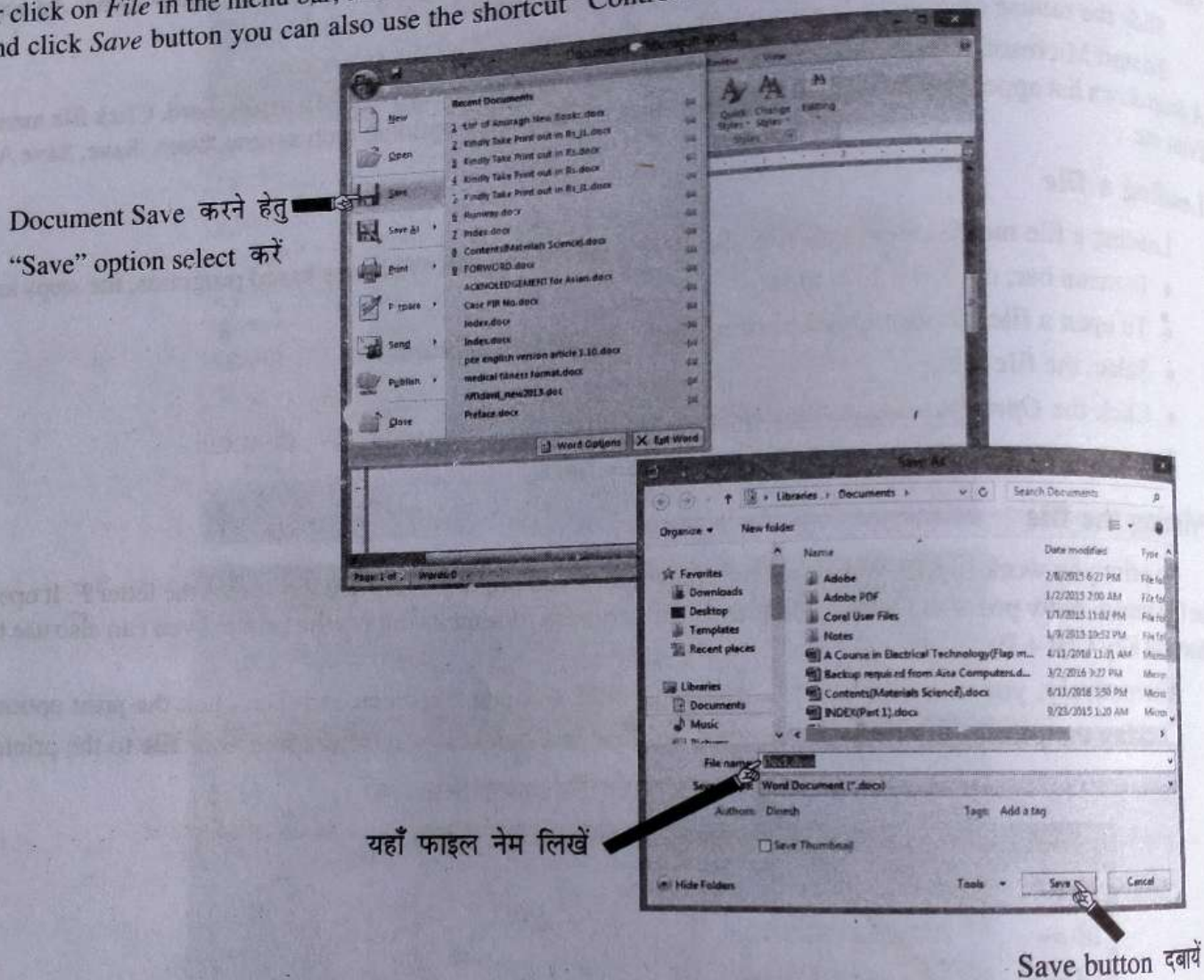
Clicking the printer icon on the standard toolbar is a quickest way of sending your file to the printer.



चित्र 9.16

Saving the File

Saving the file means placing a copy of it on to a hard disk. To save your work, you can use *Alt + F + S* or click on *File* in the menu bar, and select *Save As* option. Save as dialog box appears. Write the file name and click *Save* button you can also use the shortcut "Control + *S*".



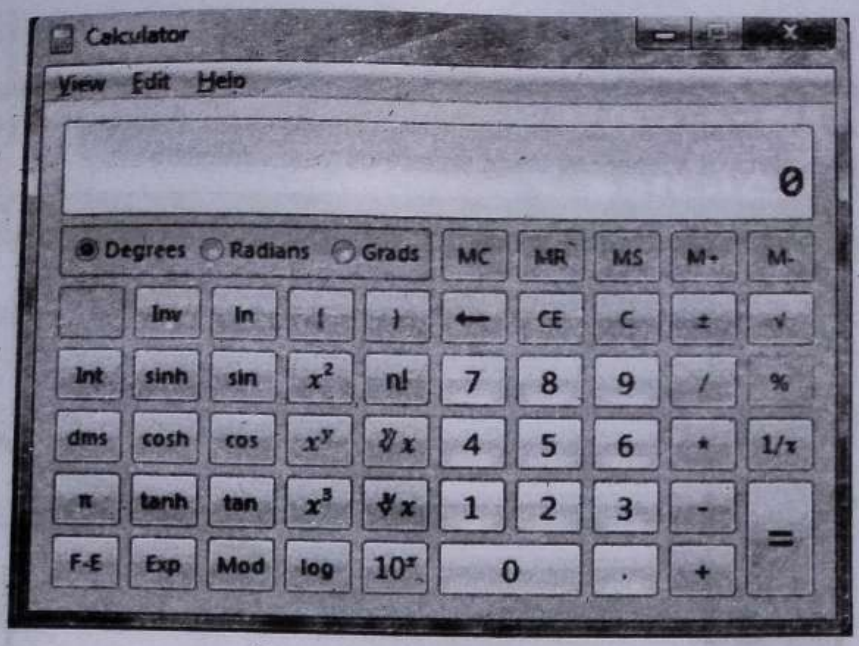
चित्र 9.17—Saving the File

प्रयोग संख्या 11

उद्देश्य (Object) : Familiarity with some programs like Calculator, Window media player and Paint.

Calculator :

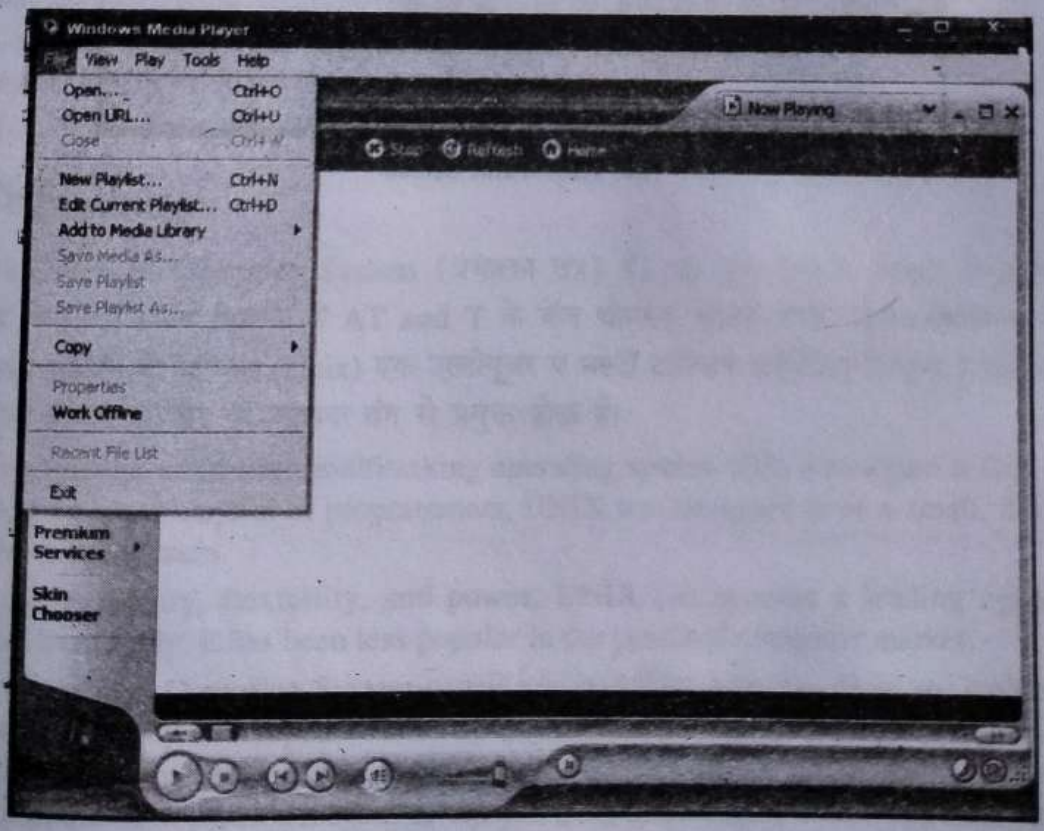
The computer calculator works like a normal calculator. You can perform simple calculations such as addition, subtraction, multiplication, and division. Calculator also offers the advanced capabilities of a programming, scientific, and statistical calculator. To start working calculator, select *Start* > *Programs* > *Accessories* > *Calculator*.



चित्र 9.18—Scientific Calculator

Windows Media Player :

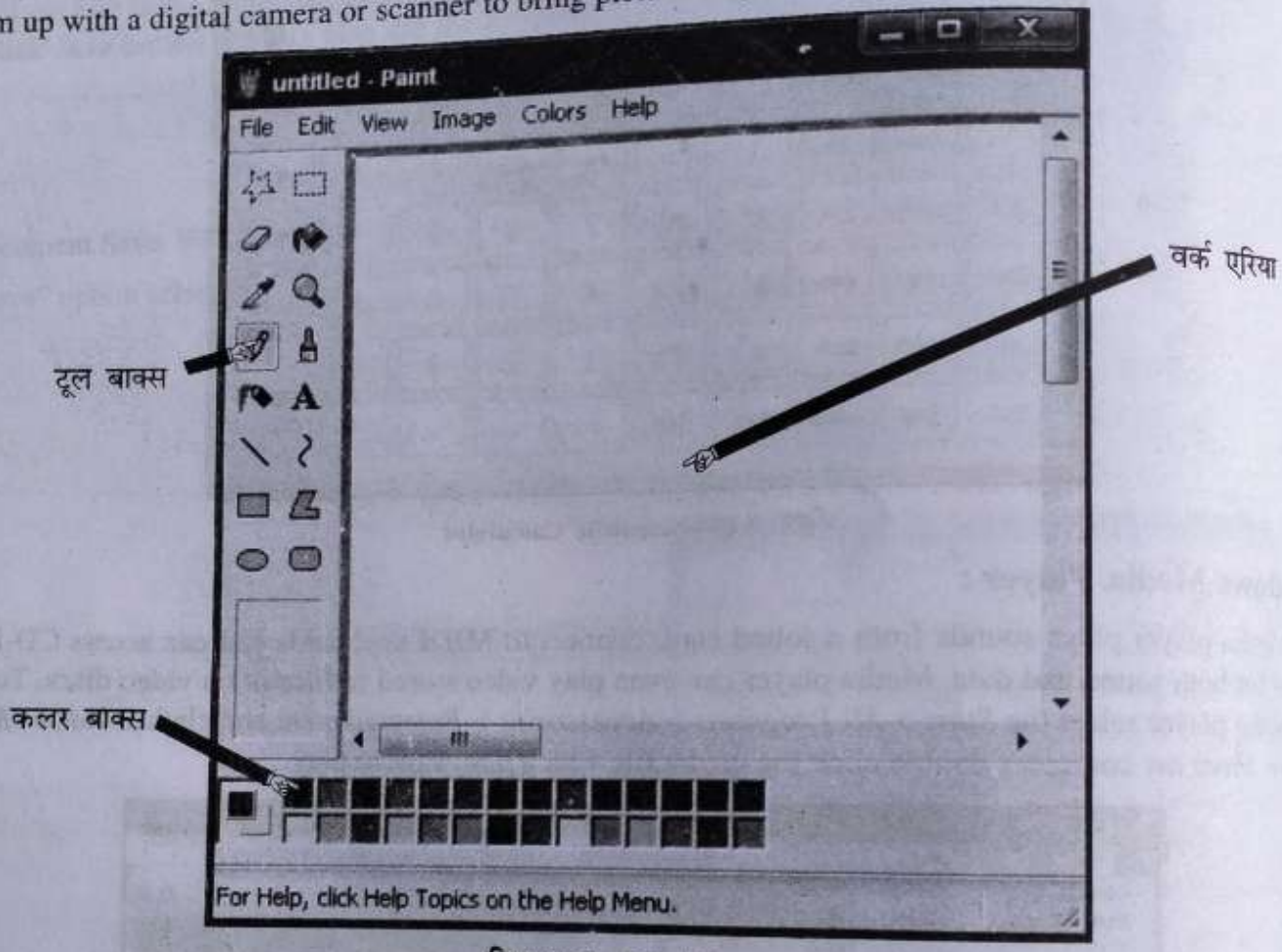
Media player plays sounds from a sound card, connect to MIDI keyboards and can access CD-ROM drives for both sound and data. Media player can even play video stored in files or on video discs. To start the media player select the *Start > All Programs > Accessories > Entertainment* and click *Windows Media Player*. From my computer double click the media file which you want to play.



चित्र 9.19—Media Player

Paint

Paint can copy drawings and pictures from paint and paste them into other Window programs. To start paint, select *Start > All programs > Accessories > Paint*. Paint has more features than paintbrush; paint can team up with a digital camera or scanner to bring picture on your screen.



चित्र 9.20—Paint Screen

10 Chapter केस स्टडी—यूनिक्स तथा लिनक्स (CASE STUDY—UNIX AND LINUX

THINK ABOUT IT

The only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. —**Steve Jobs**

Life is like photography. You need the negatives to develop. —**Unknown**

You need chaos in your soul to give birth to a dancing star. —**Friedrich Nietzsche**

§ 10.1. परिचय (Introduction)

केस स्टडी अर्थात् किसी विषय का गहराई से अध्ययन करना तथा उसकी details जानने का प्रयास करना। चूँकि अब तक आपने इस पुस्तक में operating systems के विभिन्न तथ्यों का अध्ययन कर लिया है, अत: अब इस अध्याय में हम UNIX व LINUX operating systems से संबंधित details का अध्ययन करेंगे।

A case study is a research strategy and an empirical inquiry that investigates a phenomenon within its real-life context. A case study is a research method involving an up-close, in-depth, and detailed examination of a subject of study (the case), as well as its related contextual conditions.

§ 10.2. यूनिक्स (UNIX)

यूनिक्स एक कम्प्यूटर Operating System (प्रचालन तंत्र) है। यह मूल रूप से 1969 में Bell Laboratories में विकसित किया गया था। इसके विकास में AT and T के केंन थोम्प्सन, डेनिस रिची, ब्रियन केर्निघन, दोग्ल्स मेक्लेरी और जो ओसाना आदि शामिल थे। यूनिक्स (Unix) एक एल्टीयूजर व मल्टी टास्किंग ऑपरेटिंग सिस्टम है यह वर्कस्टेशंस और सर्वर में मास्टर कंट्रोल प्रोग्राम के तौर पर व्यापक ढंग से प्रयुक्त होता है।

UNIX is a popular multi-user, multitasking operating system (OS) developed at Bell Labs in the early 1970s. Created by just a handful of programmers, UNIX was designed to be a small, flexible system used exclusively by programmers.

Due to its portability, flexibility, and power, UNIX has become a leading operating system for workstations. Historically, it has been less popular in the personal computer market.

Unix is a computer Operating System capable of handling activities from multiple users at the same time. Unix was originally developed in 1969 by a group of AT and T employees Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna at Bell Labs. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops. UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available.

- यूनिक्स (Unix) एक मल्टीयूजर व मल्टी टास्किंग ऑपरेटिंग सिस्टम है।

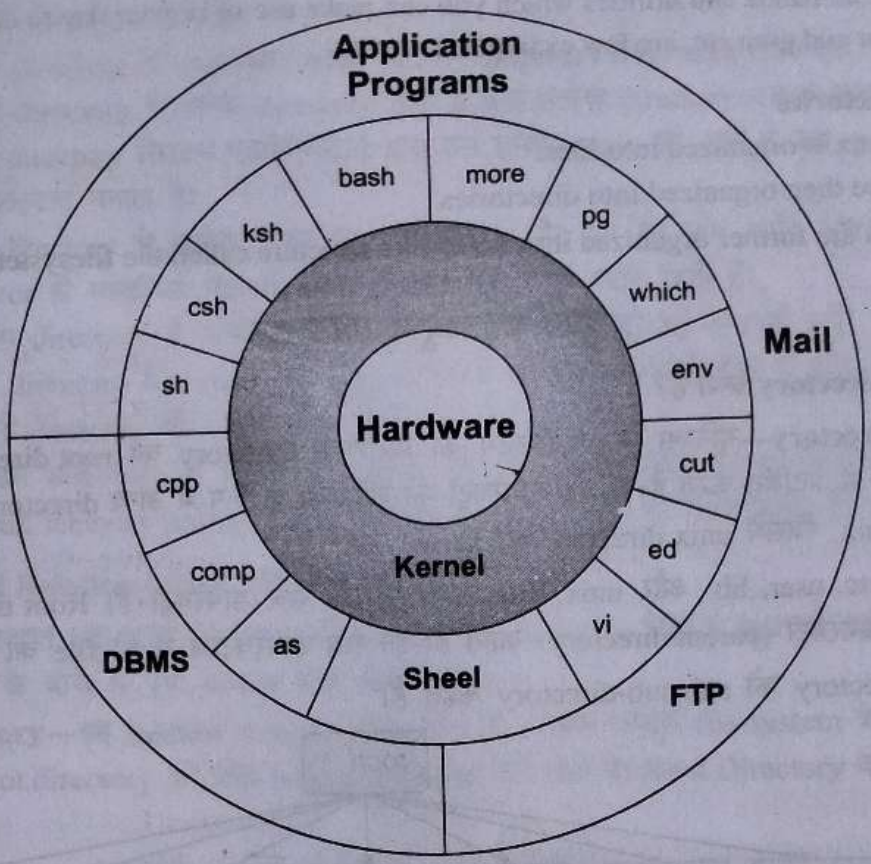
- वर्कस्टेशंस और सर्वर में मास्टर कंट्रोल प्रोग्राम के तौर पर व्यापक ढंग से प्रयुक्त होता है।
- C लैंग्वेज में लिखा जाता है।
- यूनिक्स सर्वर्स और इन्टरनेट के लिए प्रयुक्त ऑपरेटिंग सिस्टम है।
- यूनिक्स ऑपरेटिंग सिस्टम बहुत शक्तिशाली है।
- इसे अन्य ऑपरेटिंग सिस्टम के मुकाबले इंस्टाल और सेटअप करना कठिन है, लेकिन यह कम्प्यूटर के रिसोर्स और पॉवर पर उच्च नियंत्रण प्रदान करता है।
- यूनिक्स, दुर्घटनावश डिलीटीड और अनधिकृत यूजर्स की एक्सेस से इन्फोर्मेशंस को सुरक्षित रखने के लिए कई बिल्ट-इन सिक्योरिटी फीचर्स रखता है।
- यूनिक्स मेनफ्रेम कहलाने वाले सिंगल लार्ज कम्प्यूटर के लिए ऑपरेटिंग सिस्टम के तौर पर मूलरूप से विकसित हुआ था; क्योंकि मल्टीपल यूजर्स एक ही समय में मेनफ्रेम कम्प्यूटर एक्सेज कर सकते हैं।
- यूनिक्स कई प्रोग्राम और मल्टीटास्किंग के तौर पर एक साथ कई काम करने के लिए विकसित किया गया था। यूनिक्स की मल्टीटास्किंग क्षमता इसे नेटवर्क के लिए एफिसिएंट सिस्टम बनाती है।
- The Unix operating system is a set of programs that act as a link between the computer and the user.
- Computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the operating system or the kernel.
- Users communicate with the kernel through a program known as the shell.
- Shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.
- Originally developed in 1969 by a group of AT&T employees Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna at Bell Labs.
- Various Unix variants available in the market.
- Solaris Unix, AIX, HP Unix and BSD are a few examples.
- Linux is also a flavor of Unix which is freely available.
- Several people can use a Unix computer at the same time; hence Unix is called a multiuser system.
- User can also run multiple programs at the same time; hence Unix is a multitasking environment.
- Different versions of UNIX, although they share common similarities are Sun Solaris, GNU/Linux, and MacOS X.

Basic UNIX Commands

Examples of the basic UNIX commands:

- ls (Lists files)
- ls -l (Lists files in long format)
- cd *name* (Change directory)
- cd .. (Go to directory above current)
- cp *filename1 filename2* (Copies a file)
- chmod options *filename* (Change the read, write, and execute permissions on your files)
- mkdir *name* (Creates a directory)

Unix Architecture



चित्र 10.1—Unix architecture

Kernel

- Heart of the operating system.
- Interacts with the hardware and most of the tasks like memory management, task scheduling and file management.
- Kernel of UNIX is the hub of the operating system
- Allocates time and memory to programs and handles the filestore and communications in response to system calls.

Shell

- Shell is the utility that processes your requests.
- When you type in a command at your terminal, the shell interprets the command and calls the program that you want.
- Uses standard syntax for all commands.
- C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.
- Acts as an interface between the user and the kernel.
- When a user logs in, the login program checks the username and password, and then starts another program called the shell.
- Shell is a command line interpreter (CLI).
- Interprets the commands the user types in and arranges for them to be carried out.
- Commands are themselves programs: when they terminate, the shell gives the user another prompt (% on our systems).

Commands and Utilities

- Various commands and utilities which you can make use of in your day to day activities.
- cp, mv, cat and grep etc. are few examples.

Files and Directories

- Data of Unix is organized into files.
- All files are then organized into directories.
- Directories are further organized into a tree-like structure called the filesystem.

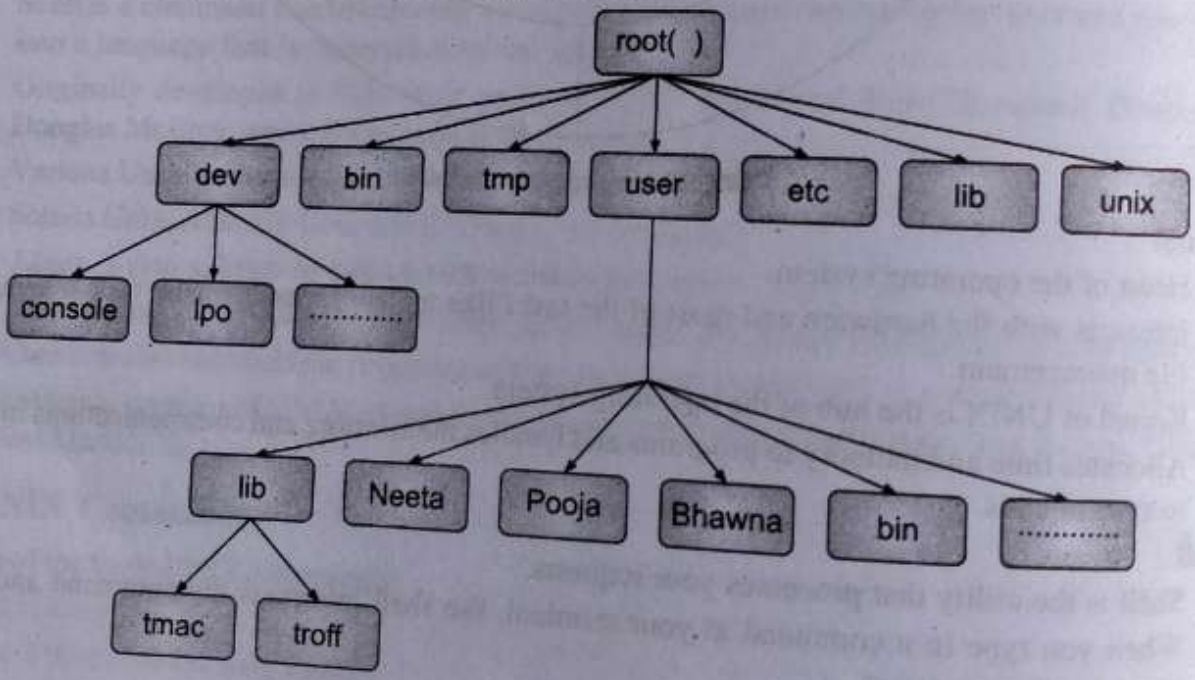
कुछ महत्वपूर्ण प्रश्नोत्तर

प्रश्न 1—Root directory क्या है?

उत्तर—Root Directory—यूनिक्स फाइल सिस्टम की प्रारम्भिक directory को root directory कहते हैं। Root डायरेक्ट्री को स्लैश (/) से प्रदर्शित करते हैं। Root डायरेक्ट्री की शाखाओं के रूप में अन्य directory हैं—dev, bin, tmp, etc. user, lib और unix, जिसमें unix directory को kernel कहते हैं।

dev, bin, tmp, etc, user, lib और unix, root डायरेक्ट्री की सब-डायरेक्ट्री हैं। Root directory को इन सभी directory की पैरेन्ट-डायरेक्ट्री (parent directory) कहते हैं। इन सब डायरेक्ट्रीज में भी file को store करते हैं।

इन फाइल या directory को sub-sub-directory कहते हैं।



चित्र 10.2

यूनिक्स में root directory के अन्तर्गत अलग-अलग directory create करने का मुख्य कारण यह है कि एक समूह से सम्बन्धित files को दूसरे समूह की files से अलग स्टोर करके रखा जा सके।

Root directory के अन्तर्गत सभी main directory जैसे—bin, etc, tmp, dev, usr, lib, min, home, kernel store होती हैं—

(i) **/bin**—इसमें linux के अधिकांश commands, executable of program file के रूप में store होते हैं। Directory में store किए गए सभी program files, binary form में होती हैं, इसलिए इसे /bin कहते हैं।

(ii) **/dev**—इस directory में प्रत्येक device, जैसे—printer, monitor, disk, modem, mouse आदि से सम्बन्धित अलग-अलग file होती हैं।

(iii) **/lib**—इस directory में system के installed विभिन्न compiler के routines जो programmers द्वारा use किए जाते हैं, को store करते हैं।

(iv) **/tmp**—इस directory में user द्वारा store की गई temporary files store होती है।

(v) **/user**—इस directory में अनेक directory होती हैं तथा प्रत्येक directory अलग-अलग user से सम्बन्धित होती है। यह directory सिस्टम एडमिनिस्ट्रेटर द्वारा उस समय create की जाती है जब system administrator यूजर्स का एकाउन्ट बनाता है।

(vi) **/etc**—इस directory में प्रत्येक यूजर का password और ग्रुप का नाम आदि सूचनाओं के साथ सिस्टम administrator से सम्बन्धित बाइनरी एक्ज़ीक्यूटेबल फाइल भी स्टोर रहती हैं।

(vii) **/home**—इस directory में सभी user से सम्बन्धित होम डायरेक्ट्री की सूचनाएँ स्टोर रहती हैं।

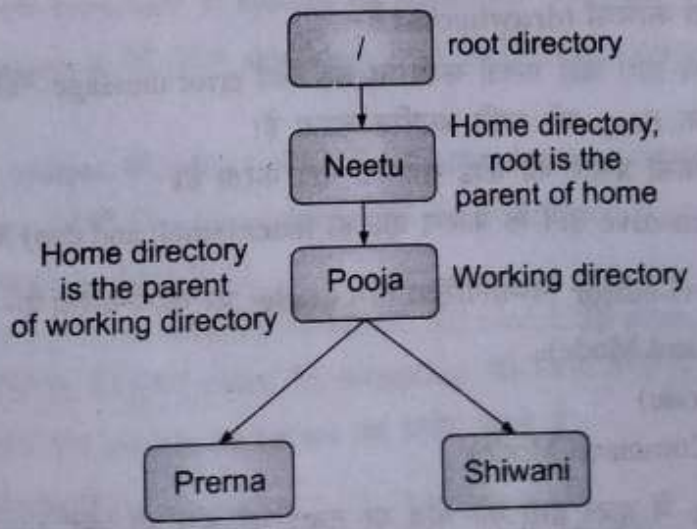
(viii) **/mnt**—इस directory में floppy और cdrom नामक दो सब-डायरेक्ट्री होती हैं और जब इन्हें mount किया जाता है तो ये contents को दर्शाती हैं।

(ix) **/kernel**—इस directory में kernel program का कोड लिखा जाता है। Kernel program ही process management, memory management तथा file management का कार्य करता है।

प्रश्न 2—Parent-child Relationship से आप क्या समझते हैं?

उत्तर—Unix में सभी फाइल एक-दूसरे से सम्बन्धित रहती हैं। यूनिक्स में File सिस्टम कुछ भी नहीं है, लेकिन इन फाइल के समूह (collection) के बीच में एक सम्बन्ध होता है—

- **Root directory**—यह highest level की directory है, जिसमें सम्पूर्ण file system के content list के रूप में होते हैं। Root directory की कोई parent directory नहीं होती है। Root Directory को slash (/) से प्रदर्शित करते हैं।
- **Home directory**—यह ऐसी directory होती है जिसका उपयोग user system में login के बाद करता है। यह वह directory होती है, जिसमें user का personal directory structure start होता है। प्रत्येक user अपनी login id से directory में login करता है।
- **Working directory**—इस directory का प्रयोग present time में एक session में user द्वारा work करने के लिए किया जाता है। जब user किसी shell में login करता है तो उसकी working directory, home directory होती है।
- **Parent directory**—इस directory का प्रयोग immediate working directory में आने के लिए करते हैं।



चित्र 10.3—Parent-Child Relationship

फाइल सिस्टम में directory के बीच relation ही parent-child relationship कहलाता है।

प्रश्न 3—यूनिक्स (Unix) में फाइल कितने प्रकार की होती है, व्याख्या कीजिए।

उत्तर—यूनिक्स में सभी डाटा (data) को फाइल में organize करते हैं और इन फाइल्स (files) को डायरेक्ट्री (directory) में सुरक्षित रखते हैं। ये डायरेक्ट्री (directory) एक tree-like structure (पेड़ जैसी संरचना) बनाती हैं जिसे फाइल सिस्टम (file-system) कहते हैं। यूनिक्स में तीन प्रकार की files होती हैं—

(i) साधारण फाइल (Ordinary File)

(ii) डायरेक्ट्री फाइल (Directory File)

(iii) डिवाइस फाइल (Device Fiie)

- Character device file (कैरेक्टर डिवाइस फाइल)
- Block device file (ब्लॉक डिवाइस फाइल)

साधारण फाइल (Ordinary File)—साधारण फाइल, वे फाइल होती हैं जो साधारण data, program, instruction और text को स्टोर करती हैं। साधारण तौर पर हम अपने सभी data को ordinary file में ही सुरक्षित रखते हैं।

डायरेक्ट्री फाइल (Directory File)—डायरेक्ट्री फाइल, वे फाइल होती हैं जिनमें data, program तो स्टोर नहीं होते हैं, लेकिन साधारण फाइल (ordinary file) की सूचनाएँ directory file में स्टोर होती हैं। अत: हम कह सकते हैं कि डायरेक्ट्री फाइल में सभी फाइल और उपडायरेक्ट्री (sub-directory) का लेखा-जोखा स्टोर करते हैं।

डिवाइस फाइल (Device File)—डिवाइस फाइल (Device FIle) वह File है जिसमें विभिन्न डिवाइसेज (devices) जैसे फ्लोपी ड्राइव, टेप ड्राइव, हार्ड डिस्क, प्रिन्टर इत्यादि को एक फाइल मानकर स्टोर करते हैं और ये सभी डिवाइस फाइल कहलाती हैं।

Device File को दो भागों में बांटा गया है—

(क) कैरेक्टर डिवाइस फाइल

(ख) ब्लॉक डिवाइस फाइल

प्रश्न 4—vi-editor पर टिप्पणी लिखिए।

उत्तर—vi-editor—vi विजुअल एडीटर (Visual Editor) का संक्षिप्त रूप है। यह एक स्क्रीन ओरियेन्टड (screen oriented) टेक्स्ट एडीटर (text editor) है।

vi editor का प्रयोग कर हम पूरे डॉक्यूमेन्ट (document) को एक साथ देख सकते हैं और edit भी कर सकते हैं।

vi editor की कुछ निम्नलिखित कमियाँ (drawbacks) हैं–

- Editing के समय यूज़र द्वारा कोई गलती करने पर यह कोई error message नहीं देता है। किसी error के घटित होने पर यह केवल एक Beep की ध्वनि प्रसारित करता है।
- vi editor यूज़र की किसी प्रकार की कोई सहायता नहीं करता है।
- vi editor के Case-sensitive होने के कारण एक ही letter (small and cap) रूप में क्रियात्मक अन्तर होता है।

vi के मोड्स (Modes of vi)—vi-editor निम्नलिखित तीन modes पर कार्य करता है—

(i) कमाण्ड मोड (Command Mode)

(ii) इन्सर्ट मोड (Insert Mode)

(iii) ex कमाण्ड मोड (ex Command Mode)

Command Mode—इस mode में यूज़र द्वारा की-बोर्ड पर दबाई गई कोई भी 'की' key editor की कमाण्ड के रूप में interpret की जाती है।

Insert Mode—इस mode में आप file में नए टैक्स्ट को Add, पहले टैक्स्ट को delete, या replace कर सकते हैं।

Command Mode से इन्सर्ट Mode में आने के लिए Esc key दबाते (press) हैं।

ex कमाण्ड मोड—इस मोड में vi के कमाण्ड, कमाण्ड लाइन (command line) दिए जाते हैं। vi स्क्रीन की सबसे नीचे वाली लाइन को कमाण्ड लाइन कहते हैं जहाँ vi-command लिखे जाते हैं। इसे ex कमाण्ड मोड इसलिए कहते हैं क्योंकि इस मोड के सभी कमाण्ड ex-editor के कमाण्ड से मिलते हैं।

प्रश्न 5—यूनिक्स ऑपरेटिंग सिस्टम (Unix Operating System) के विभिन्न प्रकार के shells का वर्णन करें।

उत्तर—Shell—Shell, unix operating system का एक प्रमुख भाग है। Shell, Unix के कर्नेल और यूज़र के मध्य इन्टरफेस स्थापित करता है।

यूनिक्स ऑपरेटिंग सिस्टम के सभी संस्करण (versions) में कम से कम तीन शैल रहते हैं—

(i) बॉर्न शैल (Bourne shell)
(ii) सी शैल (C shell)
(iii) कॉर्न शैल (Korn shell)

बॉर्न शैल (Bourne Shell): (Sh)—Bourne Shell unix operating system के shell का एक भाग है। इसे Steve Bourne ने विकसित (develop) किया था।

Bourne shell के पास कोई आकृर्षित (interactive) करने वाली properties नहीं होती है। Bourne shell की language काफी कठिन थी।

सी शैल (C Shell): (csh)—सी शैल को Bill Joy ने विकसित किया था। C shell के प्रोग्राम का नाम csh है। इसे % से प्रदर्शित करते हैं। C shell का syntax लगभग programming language C के syntax से काफी मिलता-जुलता है। C shell की परिवार (family) का एक प्रमुख सदस्य (member) tcsh है।

कॉर्न शैल (Korn Shell): (ksh)—कॉर्न शैल (Korn Shell) को डेविड कार्न (David Korn) ने विकसित किया था। यह शैल प्रोग्रामिंग भाग (Shell Programming Language) के साथ C और TC shells के भी गुणों को provide कराती है। Korn shell में अनेक गुण जैसे—एरे (Array), कमाण्ड एलियसिंग (Command Aliasing) इत्यादि उपलब्ध हैं।

Non-Interactive Mode—इस mode में fsck command यदि program run time में कोई error detect करता है तो वह user की बिना permission के उस error को fix तथा solve करके program को continue करता है।

Shutdown System—यूनिक्स ऑपरेटिंग सिस्टम में shutdown command का प्रयोग कम्प्यूटर को turn off या reboot करने के लिए करते हैं। केवल एक superuser ही system को shutdown कर सकता है।

किसी भी unix operating system को तुरन्त shutdown करने के लिए shutdown-h कमाण्ड का प्रयोग या init0 command का प्रयोग करते हैं।

किसी भी unix operating system को reboot करने के लिए shutdown-r command का प्रयोग करते हैं।

Mounting and Unmounting—किसी भी external device को internal device से connect करना ही mounting तथा उसे disconnect करना unmounting कहलाता है।

Ex : Mounting या Unmounting process का प्रयोग अधिकांशत: SD card, USB storage, DVD या other removable storage device में देखने को मिलता है। जिस point पर mounting की क्रिया होती है उसे mount point कहते हैं।

Mounting process के लिए हम mount command का प्रयोग करते हैं।

$mount <file name> <file name>

Unmounting process के लिए हम unmount command का प्रयोग करते हैं।

$unmount <file name> <file name>

प्रश्न 6—Unix operating system में administrator की आवश्यकता क्या है?

उत्तर—Unix operating system में administrator एक ऐसा person होता है जो पूरे सिस्टम को नियंत्रित अर्थात् administrate करने के लिए responsible होता है, जो कम्प्यूटर की running time के सारे administrative कार्य को refer करती है; जैसे—नया यूज़र बनाना, यूज़र का एकाउन्ट delete करना।

Responsibilities of Administrator

- Booting the system
- User Management
- Disk Management
- Backup Management
- Security
- File system checking
- Mounting & Unmounting
- Shutdown system

System Bootup

- As soon as you turn on the system, it starts booting up and finally it prompts you to log into the system, which is an activity to log into the system and use it for your day-to-day activities.

Login Unix

To log in

- You need to have your userid (user identification) and password.
- Contact your system administrator if you don't have these yet.
- Type your userid at the login prompt, then press ENTER.
- Note that your userid is case-sensitive (amisha and Amisha have a different meaning), so be careful that you type it exactly as your system administrator has instructed.
- Now, Type your password at the password prompt, then press ENTER. Your password is also case-sensitive.
- If you provide the correct userid and password, then you will be allowed to enter into the system.

Listing Directories and Files

- Data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem.
- You can use the **ls** command to list out all the files or directories available in a directory.

Logging Out

- Type the **logout** command at the command prompt, and the system will clean up everything and break the connection.
- Everything in UNIX is either a file or a process.
- A process is an executing program identified by a unique PID (process identifier).
- A file is a collection of data. They are created by users using text editors, running compilers etc.

Examples of files:

- A document
- The text of a program written in some high-level programming language
- Instructions comprehensible directly to the machine and incomprehensible to a casual user, for example, a collection of binary digits (an executable or binary file);
- A directory, containing information about its contents, which may be a mixture of other directories (subdirectories) and ordinary files.

Types of files?

- **Ordinary Files**—contains data, text, or program instructions. In this tutorial; you look at working with ordinary files.
- **Directories**—store both special and ordinary files. (equivalent to folders in windows).
- **Special Files**—provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. (similar to aliases or shortcuts and enable you to access a single file using different names).

Creating Files

Use the **vi** editor to create ordinary files on any Unix system i.e. give the following command—

```
$ vi filename
```

for example to create a file named seema, give the command—

```
$ vi seema
```

The above command will open a file with the given filename. Now, press the key **i** to come into the edit mode. Once you are in the edit mode, you can start writing your content in the file as in the following program—

```
Hello meenakshi, This is neha....I am creating a new file in UNIX.....
I'm happy to create my first file.............
```

Now, follow these steps—

- Press the key esc to come out of the edit mode.
- Press two keys Shift + ZZ together to come out of the file completely.

You will now have a file created with filename seema in the current directory.

```
$ vi seema
$
```

Editing Files

You can edit an existing file using the **vi** editor. To open an existing file—

```
$ vi seema
```

Once the file is opened, you can come in the edit mode by pressing the key **i** and then you can proceed by editing the file. If you want to move here and there inside a file, then first you need to come out of the edit mode by pressing the key **Esc**. After this, you can use the following keys to move inside a file—

- **l** key to move to the right side.
- **h** key to move to the left side.
- **k** key to move upside in the file.
- **j** key to move downside in the file.

Hence, you can position your cursor wherever you want to edit. Once you are positioned, then you can use the i key to come in the edit mode. Once you are done with the editing in your file, press **Esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

Display Content of a File

You can use the **cat** command to see the content of a file.

```
$ cat seema
Hello meenakshi, This is neha.... I am creating a new file in UNIX.....
I'm happy to create my first file.............
$
```

You can display the line numbers by using the **-b** option along with the **cat** command as follows—

```
$ cat -b filename
1 Hello meenakshi, This is neha....I am creating a new file in UNIX.....
2 I'm happy to create my first file.............
$
```

Counting Words in a File

You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file—

```
$ wc seema
2 20 119 seema
$
```

- **First Column**—Represents the total number of lines in the file.
- **Second Column**—Represents the total number of words in the file.
- **Third Column**—Represents the total number of bytes in the file. This is the actual size of the file.
- **Fourth Column**—Represents the file name.

You can give multiple files and get information about those files at a time. The syntax is—

```
$ wc filename1 filename2 filename3
```

Copying Files

To make a copy of a file use the **cp** command. The syntax of the command is—

```
$ cp source_file destination_file
```

Following is the example to create a copy of the existing file filename.

```
$ cp seema vishakha
$
```

You will now find one more file vishakha in your current directory. This file will exactly be the same as the original file seema.

Renaming Files

To change the name of a file, use the **mv** command. Following is the basic syntax—

```
$ mv old_file new_file
```

The following program will rename the existing file seema to muskan

```
$ mv seema muskan
$
```

The **mv** command will move the existing file completely into the new file. In this case, you will find only muskan in your current directory.

Deleting Files

To delete an existing file, use the **rm** command. Following is the syntax—

```
$ rm seema
```

Suggestion—A file may contain useful information and hence, one must be caeful while using **Delete** command. It is better to use the **-i** option along with **rm** command.

You can remove multiple files at a time with the command given below—

```
$ rm filename1 filename2 filename3
$
```

Command	Description
ls	list files and directories
ls -a	list all files and directories
mkdir	make a directory
cd *directory*	change to named directory
cd	change to home-directory
cd ~	change to home-directory
cd ..	change to parent directory
pwd	display the path of the current directory

Standard Unix Streams

Under normal circumstances, every Unix program has three streams (files) opened for it when it starts up—

- **stdin**—referred to as the *standard input* and the associated file descriptor is 0. This is also represented as STDIN. The Unix program will read the default input from STDIN.
- **stdout**—referred to as the *standard output* and the associated file descriptor is 1. This is also represented as STDOUT. The Unix program will write the default output at STDOUT
- **stderr**—referred to as the *standard error* and the associated file descriptor is 2. This is also represented as STDERR. The Unix program will write all the error messages at STDERR.

USER ADMINISTRATIONS

Types of accounts on a Unix system—

Root account

- Also called **superuser** and would have complete and unfettered control of the system.
- Superuser can run any commands without any restriction.
- Assumed as a system administrator.

System accounts

- Needed for the operation of system-specific components
- Examples are mail accounts and the sshd accounts.
- Needed for some specific function on your system, and any modifications to them could adversely affect the system.

User accounts

- Provide interactive access to the system for users and groups of users.
- General users are typically assigned to these accounts.
- Have limited access to critical system files and directories.
- Unix supports a concept of Group Account which logically groups a number of accounts.
- Every account would be a part of another group account.
- Unix group plays important role in handling file permissions and process management.

Managing Users and Groups

User administration files are—

- **/etc/passwd**—Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
- **/etc/shadow**—Holds the encrypted password of the corresponding account. Not all the systems support this file.
- **/etc/group**—Contains the group information for each account.
- **/etc/gshadow**—Contains secure group account information.

Check all the above files using the **cat** command.

Command	Meaning
`useradd`	Adds accounts to the system
`usermod`	Modifies account attributes
`userdel`	Deletes accounts from the system
`groupadd`	Adds groups to the system
`groupmod`	Modifies group attributes
`groupdel`	Removes groups from the system

You can use Manpage Help to check complete syntax for each command mentioned above.

Create a Group

- We need to create groups before creating any account otherwise, we can make use of the existing groups in our system.
- We have all the groups listed in **/etc/groups** file.
- Syntax to create a new group account—

```
groupadd [-g gid [-o]] [-r] [-f] groupname
```

The following table lists out the parameters—

Option	Meaning
`-g GID`	The numerical value of the group's ID
`-o`	This option permits to add group with non-unique GID
`-r`	This flag instructs groupadd to add a system account
`-f`	This option causes to just exit with success status, if the specified group already exists. With -g, if the specified GID already exists, other (unique) GID is chosen
`Groupname`	Actual group name to be created

If you do not specify any parameter, then the system makes use of the default values.

Modify a Group

To modify a group, use the **groupmod** syntax—

```
$ groupmod -n new_modified_group_name old_group_name
```

Delete a Group

Use **groupdel command** and the **group name.**

This removes only the group, not the files associated with that group. The files are still accessible by their owners.

Create an Account

Syntax to create a user's account—

```
useradd -d homedir -g groupname -m -s shell -u userid accountname
```

The following table lists out the parameters—

Option	Meaning
`-d homedir`	Specifies home directory for the account
`-g groupname`	Specifies a group account for this account
`-m`	Creates the home directory if it doesn't exist
`-s shell`	Specifies the default shell for this account
`-u userid`	You can specify a user id for this account
`Accountname`	Actual account name to be created

If you do not specify any parameter, then the system makes use of the default values. The **useradd** command modifies the **/etc/passwd**, **/etc/shadow**, and **/etc/group** files and creates a home directory.

After an account is created you can set its password using the **passwd** command as follows—

```
$ passwd sneha22
Changing password for user sneha22
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

When you type **passwd accountname**, it gives you an option to change the password, provided you a a superuser. Otherwise, you can change just your password using the same command but without specify your account name.

Modify an Account

The **usermod** command enables you to make changes to an existing account from the command line. It uses the same arguments as the **useradd** command, plus the -l argument, which allows you to change the account name.

For example, to change the account name **sneha22** to **anjali32** and to change home directory accordingly, you will need to issue the following command—

```
$ usermod -d /home/anjali32 -m -l sneha22 anjali32
```

Delete an Account

The **userdel** command can be used to delete an existing user. This can prove very dangerous command if not used with caution.

There is only one argument or option available for the command **.r**, for removing the account's home directory and mail file.

For example, to remove account yashasvi, issue the command—

```
$ userdel -r yashasvi
```

If you want to keep the home directory for backup purposes, omit the -r option. You can remove the home directory as needed at a later time.

§ 10.3. लिनक्स (Linux)

लिनक्स भी यूनिक्स से मिलता-जुलता (unix-like) operating system है। यह एक multiuser, multitasking system है, तथा इसका file system परम्परागत unix semantics का अनुसरण करता है तथा इसमें standard UNIX नेटवर्किंग मॉडल का पूर्णत: implement किया जाता है।

यूनिक्स में एक कमी थी—इसको समझना तथा चलाना मुश्किल है। एंड्रयू टेनेनबाम, ऐमस्टरडैम में कम्प्यूटर विज्ञान के प्रोफेसर हैं। उन्होंने इसकी सहायता के लिए मिनिक्स नाम का प्रोग्राम लिखा। इसमें भी कुछ कमियाँ थी। लिनूस टोरवाल्ड फिनलैण्ड के हेलसिन्की विश्वविद्यालय में computer science के छात्र थे। उन्होंने मिनिक्स की कमी को दूर करने के लिए एक प्रोग्राम लिखा जो कि बाद में 'लिनूस का यूनिक्स' या छोटे में लिनक्स कहलाया।

Features of Linux (Benefits (फायदे) of Linux)

- Multiuser Operating System
- किसी भी समय एक से अधिक User काम कर सकते हैं और अपने-अपने Program चला सकते हैं।
- Linux Files को Security भी प्रदान करता है।
- अपने से जुड़े सभी User को एक अलग File Set व Directory से जोड़ देता है।
- कोई User सिर्फ अपनी Directory में available Information को ही पढ़ सकता है, delete कर सकता है और modify कर सकता है।
- Other Users की Files safe रहती हैं।
- Linux एक Multi-Programming Operating System है। यह अलग-अलग User के कई प्रोग्रामों को एक साथ चला सकता है।
- Linux में Multi-Programming सुविधा Time sharing से उपलब्ध कराई गई है।
- Linux Operating System का मुख्य भाग Kernel होता है जो Application Crash Proof है।
- Application Crash हो जाने के बाद भी Kernel काम करता है।

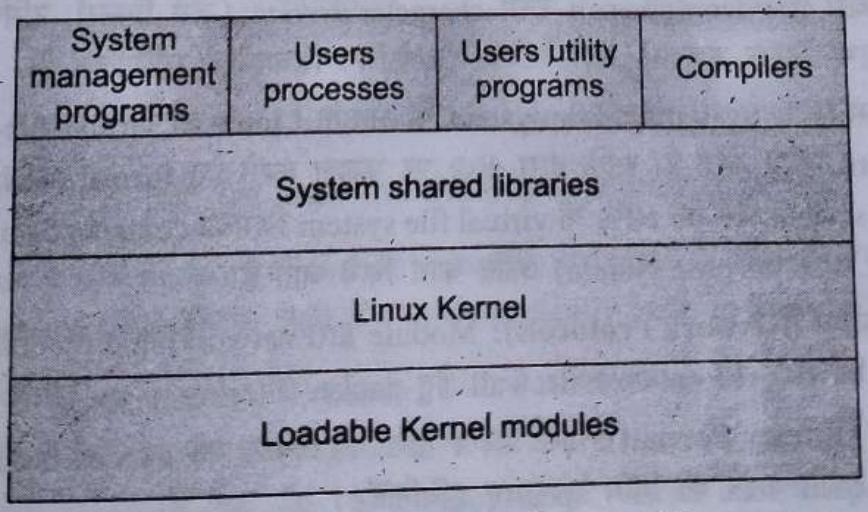
- इसका advantage यह होता है कि Linux Operating System को दोबारा चालू नहीं करना पड़ता है।
- Linux Operating System पूरी तरह Virus से सुरक्षित है क्योंकि सामान्य User इसके Kernel तक नहीं जा पाते।
- Linux Operating System भी Windows की तरह Graphical User Interface (GUI) पर काम करता है इसकी Windows को X-Windows कहा जाता है।
- Windows की तरह इसमें भी Desktop, Icon, Menu, Frame आदि समस्त सुविधाएँ होती है। वैसे इसमें जरुरत होने पर इसमें DOS की तरह Command Line Interface (CLI) में भी काम किया जा सकता है।
- Linux में Apache नाम का एक Web Server Program भी उपलब्ध है जिसमें Web pages तैयार किया जाता है। यह आजकल सबसे अधिक लोकप्रिय Web Server है।
- Linux में इसके अलावा बहुत से उपयोगी प्रोग्राम और Free software भी Available है जैसे—Text Editor, Web Browser, Science के काम में आने वाले Software आदि।

लिनक्स सिस्टम में मुख्यत: तीन घटक होते हैं—Kernel, system libraries तथा system utilities.

कर्नल (Kernel): Kernel द्वारा operating system के महत्वपूर्ण abstractions को maintain किया जाता है (Virtual memory तथा processes सहित)।

सिस्टम लाइब्रेरीज़ (System Libraries): सिस्टम लाइब्रेरीज़ में functions का मान समूह (standard set) परिभाषित होता है, जिससे applications Kernel से interact कर सकें।

सिस्टम यूटिलिटीज़ (System Utilities): सिस्टम यूटिलिटीज़ के अंतर्गत वह प्रोग्राम्स आते हैं जोकि individual realization management tasks को perform करते हैं। कुछ सिस्टम यूटिलिटीज़ केवल एक बार invoke की जाती हैं, तथा system को initialize तथा configure करने में सहायक होती हैं जबकि कुछ अन्य यूटिलिटीज़ जिनको की UNIX की शब्दावली में daemons (डैमन्स) कहा जाता है, permanently run करती हैं, तथा कई tasks (कार्य) जैसे कि incoming network connection को respond करना, terminals से logon (लॉग-ऑन) requests को स्वीकार करना, log files को update करना आदि को handle करती हैं।



चित्र 10.4—Linux System Components

चित्र 10.4 में लिनक्स सिस्टम के विभिन्न घटक प्रदर्शित हैं। Kernel code processor की privileged* mode में कार्य करता है तथा computer के सभी physical resources तक इसकी पहुँच (access) होती है। Linux Kernel लिनक्स ऑपरेटिंग सिस्टम का core होता है। यह processes को run करने हेतु सम्पूर्ण functionality प्रदान करता है तथा hardware resources को arbitration व protected access देने हेतु system services प्रदान करता है।

* विशेष अधिकार

सिस्टम लाइब्रेरीज़ कई कार्य करती है जैसे कि वह applications को Kernel service system requests करने की अनुमति देती है, आदि।

लिनक्स system कई प्रकार के User-mode programs प्रदान करता है (system utilities व user utilities, दोनों)। सिस्टम यूटिलिटीज़ के अंतर्गत वह प्रोग्राम्स होते हैं, जो कि system को initialize करने हेतु आवश्यक होते हैं (जैसे कि Kernel modules को load करने हेतु, network devices को configure करने हेतु, server programs जैसे कि user login request, incoming network connections, printer queues को run करने हेतु आदि।

कर्नल मॉड्यूल्स (Kernel Modules): Linux Kernel में Kernel code के arbitrary section को माँग के अनुसार (on demand) load तथा unload करने की क्षमता होती है। यह loadable kernel modules privileged kernel mode में run करते हैं, तथा मशीन (जिस पर यह run करते हैं) की सभी hardware capabilities को access करने का अधिकार रखते हैं। Linux के अंतर्गत module support के तीन घटक होते हैं—

मॉड्यूल प्रबंधन (Module Management): यह modules को memory में load होने तथा बाकी kernel से communicate (संचार) करने की अनुमति प्रदान करता है।

ड्राइवर पंजीकरण (Driver Registration): यह module को बाकी kernel को यह बताने की अनुमति प्रदान करता है कि एक नया driver उपलब्ध है। जब भी कोई module load होता है, तो जब तक बाकी kernel (rest of the kernel) को यह पता नहीं चलेगा कि यह कौन सी नयी functionality प्रदान करेगा, तब तक वह module memory के एक isolation region (पृथक क्षेत्र) की भांति व्यवहार करेगा)। Kernel सभी known drivers (ज्ञात ड्राइवर्स) की एक dynamic table maintain करता है तथा इन tables में ड्राइवर्स को add या remove करने हेतु routines प्रदान करता है। कर्नल यह सुनिश्चित करता है कि जब भी कोई मॉड्यूल लोड होता है वह modules की start-up routine को call करता है तथा उस module के unload होने से पहले module की clean-up routine को call करता है। यह routines module की functionality को रजिस्टर करने के लिये responsible होती है।

एक module कई प्रकार के ड्राइवर्स को रजिस्टर कर सकती है। रजिस्ट्रेशन तालिका (Registration table) में मुख्यत: निम्न items होते हैं—

- **डिवाइस ड्राइवर्स (Device drivers):** इनमें character drivers (जैसे प्रिन्टर्स, टर्मिनल्स, माइस आदि), ब्लॉक डिवाइसेज़ (सभी डिस्क ड्राइव्स) तथा नैटवर्क इंटरफेसिंग डिवाइसेज़ सम्मिलित हैं।
- **फाइल सिस्टम्स (File Systems):** File system के अंतर्गत Linux की virtual-file-system calling routines को implement किया जाता है। इनके द्वारा disk पर फाइल करने हेतु format को implement करना, या फिर network file system जैसे कि NFS या virtual file system जिसकी contents demand पर generate की जाती है (जैसे कि Linux का proc system) आदि कार्य किये जाते हैं।
- **नैटवर्क प्रोटोकॉल्स (Network Protocols):** Module द्वारा networking protocols को implement किया जा सकता है जैसे कि IPX या network firewall हेतु packet filtering rules आदि।
- **बाइनरी प्रारूप (Binary Format):** इस प्रारूप द्वारा नये प्रकार की executable files को recognise करने (पहचानने) व load करने की विधि specify (विनिर्दिष्ट) की जाती है।

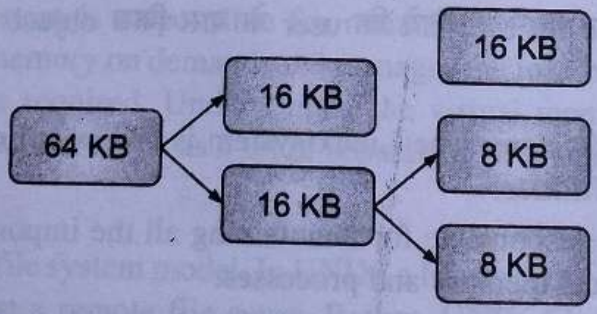
कॉनफ्लिक्ट रैसोल्यूशन (Conflict Resolution): Conflict resolution का शब्दिक अर्थ होता है किसी विवाद का हल निकालना। सामान्यत: व्यवसायिक UNIX implementations विक्रेता (vendor) के अपने हार्डवेयर पर run करने हेतु बेची जाती है। जबकि IBMPC हार्डवेयर कई प्रकार की configurations में आता है, तथा विभिन्न डिवाइसेज़ जैसे कि Network cards, SCSI controllers, video display adaptors हेतु कई प्रकार के ड्राइवर्स उपलब्ध होते हैं।

अत: hardware resources तक पहुँच को आसान बनाने हेतु Linux एक central conflict resolution mechanism प्रदान करता है जिसके मुख्य उद्देश्य होते हैं—(i) Modules को hardware resource को access करते वक्त clash होने से बचाना (ii) Autoprobes से बचाव अर्थात् वह device-driver probes device configuration को existing device drivers से interface करने को auto detect करती हैं, (iii) यदि multiple drivers एक ही हार्डवेयर को access करने का प्रयास करते हैं, तो उनकी conflicts को resolve करना।

प्रौसेस मैनेजमैंट (Process Management): प्रौसेस मैनेजमैंट के अंतर्गत वह सभी कार्य आते हैं जिनके द्वारा सभी user-requested activity को operating system में service किया जाता है। Unix का process management दो operations में विभाजित है—(i) Fork system call द्वारा new process create किया जाता है, तथा (ii) Call के पश्चात् नया प्रोग्राम run किया जाता है।

मैमोरी मैनेजमैंट (Memory Management): Linux के मैमोरी मैनेजमैंट के दो घटक होते हैं—

(i) पहला घटक physical मैमोरी का allocation (आवंटन) व freeing (मुक्त) करता है अत: pages, group of pages, तथा memory के छोटे-छोटे blocks का management करता है। Linux का primary physical memory manager page allocator होता है। यह allocator उपलब्ध physical pages को track करने हेतु buddy-heap algorithm का use करता है। Buddy heap allocator allocatable memory के दो adjacent (पड़ोसी) units को pair (जोड़ा बनाना) करती है। अत:, प्रत्येक allocatable memory unit को एक पड़ोसी partner (अर्थात् buddy) उपलब्ध रहता है तथा यदि दोनों memory region एक साथ free हो जाती हैं तो वह मिलकर एक बड़े memory क्षेत्र के रूप में combine हो जाती हैं। इसी प्रकार इस बड़े region को भी अपना buddy उपलब्ध रहता है तथा यह दोनों बड़े buddy combine करके और बड़े memory region का आकार ले सकते हैं। इसके विपरीत यदि small memory request किसी small free region द्वारा satisfy नहीं की जा सकती तो एक बड़ी free region को दो partners में divide करके request को satisfy किया जाता है।



चित्र 10.5—The Buddy Heap Splitting of Memory

Linux Kernel में सभी memory allocation statically (drivers द्वारा जो system booting के समय memory का एक contiguous area reserve करके रखते हैं) या dynamically page allocator द्वारा की जाती है।

(ii) दूसरा घटक virtual memory को handle करता है अर्थात् वह memory जोकि running processes के address space से mapped होती है, उसका management करता है।

Linux का virtual memory system प्रत्येक process को visible address space को maintain करता है। यह demand होने पर virtual memory के pages create (उत्पन्न) करता है, तथा इन pages को आवश्यकतानुसार disk से load करने या disk से बाहर swap करने को manage करता है।

फाइल सिस्टम्स (File Systems): Linux का file system unix के standard file system के समान होता है। UNIX में file के disk पर एक object के रूप में या remote file server से fetched हुआ होना अनिवार्य नहीं होता। UNIX files के अंतर्गत केवल data की input या output stream को handle करने की capability रखना ही पर्याप्त होता है। LINUX

VFS (Virtual File System) object oriented सिद्धान्त पर आधारित होता है तथा इसके दो घटक होते हैं—(i) set of definitions that define what a file object is allowed to look like (ii) a layer of software to manipulate these objects VFS द्वारा define तीन मुख्य object types हैं—(i) mode object, (ii) file object, (iii) file-system object. इनमें से पहले दो individual files को represent करते हैं तथा तीसरा सम्पूर्ण file system को प्रदर्शित करता है।

इनपुट तथा आउटपुट (Input and Output): Unix के समान, Linux में भी user किसी devices के लिये access channel open कर सकता है। System administrator file system में special files create कर सकता है। Linux सभी devices को तीन classes में विभक्त कर देता है—block devices, character devices, तथा network devices. Block devices के अंतर्गत वह devices आती हैं जो कि completely independent fixed size blocks को random access allow करती हैं (अर्थात् hard disk, floppy disks तथा CD ROMs)। Character devices के अंतर्गत अन्य devices जिनको कि regular files की सभी functionality को support करना आवश्यक नहीं होता, आती हैं (e.g. loudspeaker device)। Network devices में user सीधे data transfer नहीं कर सकते, बल्कि indirectly (अप्रत्यक्ष रूप से) commnicate करते हैं (Kernel Networking subsystem में connection open करके)।

अंत:प्रौसेस संचार, नैटवर्क स्ट्रक्चर तथा सुरक्षा (Interprocess communication, Network structure and Security): UNIX में processes के आपस में communicate करने हेतु synchronization तथा सिगनल्स उपलब्ध होते हैं जिससे process अन्य process में होने वाली events (घटनाओं) की जानकारी प्राप्त करते हैं व आपस में डेटा (data) का आदान-प्रदान करते हैं।

Linux Kernel की Networking Software की तीन परतों (layers) द्वारा implement की जाती है—Socket Interface, Protocol drivers तथा Network device drivers. Linux networking सिस्टम की सबसे महत्वपूर्ण प्रोटोकॉल IP (Internet Protocol) है। Linux की security mechanism को दो groups में classify किया जाता है—(i) Authenciation जिसके द्वारा यह सुनिश्चित किया जाता है कि कोई भी अनधिकृत व्यक्ति system को access न कर सके तथा Access control, यह सुनिश्चित करने के लिये कि user के पास किस object को access करने का अधिकार है तथा किसको नहीं।

Components of a Linux System: The Linux system is composed of three main bodies i.e., Kernel, System Libraries and System Utilities.

- **Kernel:** The kernel is responsible for maintaining all the important abstractions of the operating system, including virtual memory and processes.

  All kernel code executes in the processor's privileged mode with full access to all physical resources of the computer. Linux refers to the privileged mode as kernel mode equivalent to monitor mode. Under Linux no user mode code is built into the kernel.
- **System Libraries:** The system libraries define a standard set of functions through which applications can interact with the kernel, and that implement much of the operating system functionality that does not need the full privileges of kernel code.
- **System Utilities:** The system utilities are programs that perform individual, realization management tasks. Some system utilities may be invoked just once to initialize and configure some aspect of the system; others known as daemons run permanently, handling such tasks as responding to incoming network connections, accepting logon requests from terminals, or updating log files.

The Linux kernel has the ability to load and unload arbitrary sections of kernel code on demand. These loadable kernel modules run in privileged kernel mode, and as a consequence have full access to all hardware capabilities of the machine on which they run. Theoretically, there is no restriction on what a kernel module is allowed to do.

Linux's source code is free so anyone can modify or reload the new functionality. Recompiling, the reloading and relinking the entire kernel is a cumbersome job. If we are using kernel modules, we do not have to make a new kernel to test a new driver.

The module support under Linux has three components:

- **Module Management:** It allows modules to be loaded into memory and to communicate to the rest of kernel.
- **Driver Registration:** It allows modules to tell the rest of kernel that a new driver has become available.
- **Conflict-resolution Mechanism:** It allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

Memory management under Linux has two components:

- First components deals with allocating and freeing physical memory: pages, group of pages, and small blocks of memory.
- Second component handles virtual memory which is memory mapped into the address space of running processes.

The primary physical memory manager in the Linux kernel is the **page allocator**. This allocator is responsible for allocating and freeing all physical pages, and it is capable of allocating ranges of physically contiguous pages on request. The allocator uses a buddy heap algorithm to keep track of available physical pages. A buddy heap allocator pairs adjacent units of allocatable memory together. Each allocatable memory region has an adjacent partner called buddy and when two allocated partner regions are both freed up, they are combined to form a larger region. That larger region has a partner, with which it can combine to form a still larger free region. Alternatively, if a small memory request cannot be satisfied by existing small regions, a larger free region is splitted by two partners so to satisfy the request.

The Linux virtual memory system is responsible for maintaining the address space visible to each process. It creates pages of virtual memory on demand, and manages the loading of those pages from disk or their swapping back out to disk as required. Under Linux, the virtual memory manager maintains two separate views of an address space of a process: as a set of separate region and as a set of pages.

File Systems

Linux retains UNIX's standard file system model. In UNIX, a file does not have to be an object stored on disk *or* fetched over a network from a remote file sever. Rather, UNIX files can be anything capable of handling the input or output stream of data. Device driver can appear as files, and interprocess communication channels or network connections also look like files to the user.

Input and Output

For a user, I/O system in Linux looks like in UNIX. A user can open an access channel to a device in the same way as he/she open any other file devices appear as an object within a file system. The system administrator can create special files within a file system that contain references to a specific driver and a user opening such a file will be able to read from and write to the referenced device.

Linux splits all devices into three classes: block devices, character devices, and network devices. **Block devices** include all devices that allow random access to completely independent, fixed sized blocks of data, including hard disks, floppy disks and CD-ROMs. Block devices are used to store file systems, but direct access to a block device is also allowed so that programs can create and repair the file system that the device contains.

Character devices include most other devices, with the main expectation of network devices. Character devices do not need to support all the functionality of regular files. Network devices have to be indirectly transferred the data via kernel's networking subsystems.

§ 10.4. Comparison of Linux and Unix

Linux is an open source, free to use operating system widely used for computer hardware and software, game development, tablet PCs, mainframes etc. Unix is an operating system commonly used in internet servers, workstations and PCs by Solaris, Intel, HP etc.

Linux versus Unix Comparison Chart

	Linux	Unix
Availablity	Can be freely distributed, downloaded freely, distributed through magazines, Books etc.	Different flavors of Unix have different cost structures according to vendors.
Development	Developed by Open Source development i.e. through sharing and collaboration of code and features through forums etc and it is distributed by various vendors.	Divided into various other flavors, mostly developed by AT&T as well as various commercial vendors and non-profit organizations.
Users	Everyone. From home users to developers and computer enthusiasts alike.	Unix operating systems were developed mainly for mainframes, servers and workstations except OSX, Which is designed for everyone. The Unix environment and the client-server program model were essential elements in the development of the Internet.
Usage	Can be installed on a wide variety of computer hardware, ranging from mobile phones, tablet computers and video game consoles, to mainframes and supercomputers.	Used in internet servers, workstations & PCs. Backbone of the majority of finance infastructure and many 24 × 365 high availability solutions.
Text mode interface	BASH (Bourne Again Shell) is the Linux default shell. It can support multiple command interpreters.	Originally the Bourne Shell. Now it's compatible with many others including BASH, Korn & C.
Applications	Open Source software development and Free Operating System (OS).	Popular in universities, companies, big enterprises etc.
GUI	Provides two GUIs, KDE and Gnome. alternatives such as LXDE, Xfce, Unity, Mate, twm, etc also.	Initially Unix was a command based OS, but later a GUI was created called Common Desktop Environment.
Threat detection	Threat detection and solution is very fast.	Slow.
Architectures	Originally developed for Intel's x86 hardware.	Available on PA-RISC and Itanium machines. Solaris also available for x86/x64 based systems.

Summary

- UNIX is old and Linux is based on it.
- Linux is a good desktop OS while UNIX lacks the user friendliness needed for general computing.
- UNIX is intended for mainframes and high end computers and cannot run on mos PCs while Linux can go from mainframes down to low end personal computers.
- UNIX is proprietary while Linux is shared using the Licensing of the GNU.

कुछ महत्वपूर्ण प्रश्नोत्तर

प्रश्न 1—लिनक्स क्या है? लिनक्स के Features बताइए।

उत्तर—लिनक्स (Linux)—Linux एक multiuser और multitasking operating system है। Linux 32 bit operating system है, जो Intel, Sparc, Alpha और Power-PC सहित विभिन्न platform पर run होता है।

Linux operating system को फिनलैंड के हेलसिंकी University के student Linus Torvalds (लिनस टॉरवाल्ड्स) ने सन् 1991 में develop किया था।

Linus Torvalds ने Unix operating system की विशेषताओं से प्रभावित होकर Linux को deveop किया था। Linux operating system को Unix operating system का clone भी कहा जाता है।

Linus Torvalds ने केवल Linux का Kernel विकसित किया और उसके source code को internet पर उपलब्ध करा दिया। Internet पर sourcc code को बहुत से programmers ने पढ़ा और उसमें परिवर्तन किया, जिसे Torvalds ने स्वीकार और Linux में implement किया।

Linux में change करने के लिए Torvalds ने internet पर source code को open रखा है।

लिनक्स की विशेषताएँ (Features of Linux)—

(i) System Features

- **Multiuser**—Linux Multiuser होने के कारण इसको किसी main computer पर install करके उसके resources को एक से अधिक user terminal के माध्यम से access कर सकते हैं।

  Multiuser में main computer को host machine या server या console कहते हैं। Host Machine के controller card में जितने port होते हैं उतने Host machine से terminal को जोड़ा जाता है।

- **Multitasking**—Linux एक साथ एक से अधिक task/jobs को सम्पादित करता है। Linux user द्वारा दिए गए jobs को foreground और background में एक्जीक्यूट करती है। User के task को execute करने के लिए CPU के Scheduler (time-slice में device करते हैं और job को priority) के अनुसार time-slice देते हैं।

- **Security**—Linux Multiuser Operating System है, अत: यह data और program की sharing को suport करता है, (जिसके लिए security बहुत जरूरी है)। अत: linux में security के लिए प्रत्येक user का एक Login और Password होता है, जिसकी जानकारी के बिना कोई अन्य user system में login नहीं कर सकता है।

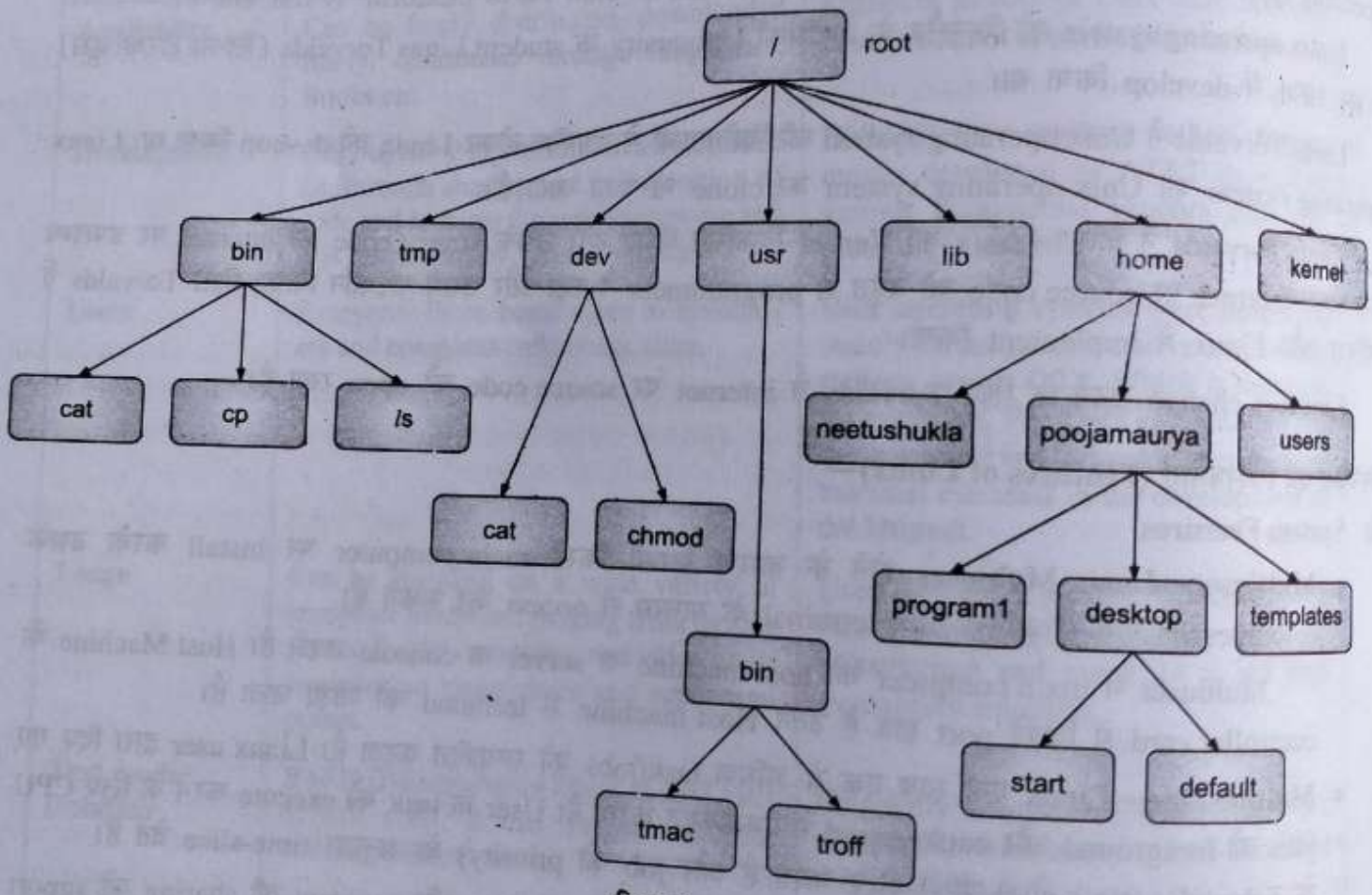
- **Virtual Memory**—Linux Memory Management के लिए Virtual Memory का use करती है। Virtual Memory, Memory Management की एक technique है जिससे program को pages में devide करके आवश्यकतानुसार disk से main memory में किया जाता है तथा main memory से disk पर भेजा जाता है। Page को disk से main memory में load करने की प्रक्रिया को swap in तथा page को main memory से disk में भेजने की प्रक्रिया को swap out कहते हैं।

(ii) Software Features

- **Automated Backup**—Linux में Backup utility के लिए tar, cpio, dd का use data और programs का backup लेने के लिए करते हैं परन्तु Red Hat Linux में BRU (Backup and Restore System Unit) utility schedule time से data के back की सुविधा देती है।
- **Office Suites**—Linux के साथ अनेक office suite जैसे—Applixware, Star office का use Microsoft Word और Microsoft Excel की files को पढ़ने और बनाने के लिए किया जाता है।
- **Web Server**—Web server एक software होता है जो web pages को serve/execute करता है। Linux के साथ Apache web server का use करते हैं।

प्रश्न 2—Linux के file structure की व्याख्या कीजिये।

उत्तर—File structure का use operating system द्वारा किसी storage device पर files और directories को manage और store करने के लिए करते हैं। Linux का file structure hierarchical होता है।



चित्र 10.6

- Linux में directory को फाइल की तरह treat किया जाता है। Linux की सभी files main directory (root directory represented by slash) में store होती है।
- Linux retains UNIX's standard file system model.
- File does not have to be an object stored on the disk or fetched over a network from a remote file server.
- Files can be anything capable of handling the input or output stream of data.
- Linux VFS (Virtual File System) is designed around object oriented principles.

- Three main object types defined by VFS are the mode object, file object (represent individual files) and file system object (represent entire file system).

प्रश्न 3—What is a 'Daemon' in UNIX?

उत्तर—Daemons are processes that are not connected to a terminal they may run in the background and they do useful work. Several daemons are normally bound on unix systems: User daemons that clean up your files and system daemons that handle scheduling and administration. For example; unix communication via uucp involve several daemons. UUCCU handles scheduling and administration of uucp jobs, checking to see if there is a job to run, selecting a device, establishing the connection, executing the job and updating the log file; uuxgt controls remote execution of commands. Some daemons handle printing and the operation of the printer spool, file backup cleanup of temporary directions and billing operations. Each of these daemons is controlled by cron which is itself run by unit.

प्रश्न 4—What is Driver Registration?

उत्तर—Once a module is loaded, it remains no more than an isolation region of memory unless it lets the rest of the kernel know what new functionality it provides. The kernel maintains dynamic tables of all known drivers, and provides a set of routines to allow drivers to be added or removed from these tables at any time.

The registration table includes—

- **Device drivers:** These drivers include character devices (such as printers, terminals or mice), block devices (including all disk drives) and network interface devices.
- **File systems:** The file system may be anything that implements Linux's virtual-file-system calling routines. It might implement a formal for storing files on disk, but it might equally will be a network file system, such as NFS or a virtual file system whose contents are generated on demand, such as *proc* system of Linux.
- **Network protocols:** A module may implement an entire networking protocol, such as IPX, or simply a new set of packet-filtering rules for a network firewall.
- **Binary format:** This format specifies a way of recognizing and loading a new type of executable file.

Q.5. What is Thrashing?

Ans. Thrashing occurs when the virtual memory of a computer is rapidly exchanging data for data on hard disk, to the exclusion of most application-level processing. As the main memory gets filled, additional pages need to be swapped in and out of virtual memory. If a process does not have access to a sufficient number of memory pages, a futile, repetitive swapping condition known as thrashing can arise resulting in high page fault rate. This leads to high, runaway CPU utilization that can make the system to a halt. Thrashing may occur in the paging system (if there is not sufficient physical memory or the disk access time is overly long), or in the I/O communications subsystem (for example, in conflicts over internal bus access), etc. In virtual memory systems, thrashing may be caused by programs or workloads that present insufficient locality of reference: if the working set of a program or a workload cannot be effectively held within physical memory, then constant data swapping, i.e., thrashing, may occur.

Hence, Thrashing can be described as a computer activity that makes almost no progress because memory or other resources have become exhausted or too limited to perform needed operations. When this happens, a pattern develops in which a request is made of the operating system by a process or program, the

operating system tries to find resources by taking them from some other process, which makes new requests that can't be satisfied. In a virtual storage system (an operating system that manages its logical storage or memory in units called pages), thrashing is a condition in which excessive paging operations are taking place. A system that is thrashing may result in either a very slow system or even get halted.

प्रश्नावली

1. (a) What is Unix? Explain the main features of Unix.
   (b) What is Linux? Explain the main features of Linux.
   (c) Explain the basic concepts of Unix architecture.
   (d) What are the main components of Linux system?
   (e) What are Kernel modules and their function?
   (f) What are daemons?
   (g) Explain the process management, memory management and file system in Unix and Linux.
2. लिनक्स क्या है? इसकी मुख्य विशेषतायें बताइये।
3. Root directory क्या है?
4. Linux के file structure की व्याख्या कीजिये।
5. Unix operating system के portability feature की व्याख्या कीजिये।
6. Unix में file तथा directory बनाने हेतु किस command का use करते हैं?
7. Parent child relation से आप क्या सकझते हैं?
8. Unix में file कितने प्रकार की होती हैं?
9. vi editor पर टिप्पणी लिखिये।
10. `ls` command का क्या उपयोग है?
11. Unix operating system में administration की क्या भूमिका है?
12. निम्नलिखित Commands को समझाइये—
    (i) mkdir (ii) cd (iii) rundir (iv) date (v) who (vi) whoami (vii) cp (viii) mv.

❑❑❑

11 Chapter यूनिक्स तथा लिनक्स—प्रैक्टीकल्स
(UNIX AND LINUX—PRACTICALS)

THINK ABOUT IT

- *You have to grow from the inside out. None can teach you, none can make you spiritual. There is no other teacher but your own soul.*
  — **Swami Vivekananda**

EXPERIMENT 1

Object: To log in into your Unix account

Description: When you first connect to a Unix system, you see a prompt

```
login:
```

To log in

- Contact your system administrator to get your user id and password
- Type your userid at the login prompt, then press ENTER. Your userid is case-sensitive and so is your password.
- Type your password at the password prompt, then press ENTER.
- If you provide the correct userid and password, only then you will be allowed to enter into the system.

EXPERIMENT 2

Object: Listing files and directories in Unix

Description: When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name, for example, **anjali22**, and it is where your personal files and subdirectories are saved.

To find out what is in your home directory, type

```
% ls
```

The `ls` command (lowercase L and lowercase S) lists the contents of your current working directory.

There may be no files visible in your home directory, in which case, the UNIX prompt will be returned. Alternatively, there may already be some files inserted by the System Administrator when your account was created.

`ls` does not, in fact, cause all the files in your home directory to be listed, but only those ones whose name does not begin with a dot (.) Files beginning with a dot (.) are known as hidden files and usually

contain important program configuration information. To list all files in your home directory including those whose names begin with a dot, type

```
% ls -a
```

Making Directories

mkdir (make directory)

To make a subdirectory called muskan33 in your current working directory type

```
% mkdir muskan33
```

To see the directory you have just created, type

```
% ls
```

Changing to a different directory

cd (change directory)

The command **cd directory** means change the current working directory to 'directory'. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree. To change to the directory you have just made, type

```
% cd muskan33
```

The current directory (.)

In UNIX, (.) means the current directory (leave a space between cd and the dot), so typing

```
% cd .
```

means stay where you are.

The parent directory (..)

(..) means the parent of the current directory, so typing

```
% cd ..
```

will take you one directory up the hierarchy

Note that by typing **cd** with no argument always returns you to your home directory. This is very useful if you are lost in the file system.

EXPERIMENT 3

Object: Using the pwd command

Description:

pwd (print working directory)

Pathnames enable you to work out where you are in relation to the whole file-system. For example, to find out the absolute pathname of your home-directory, type **cd** to get back to your home-directory and then type

```
% pwd
```

The full pathname will look something like this—

```
/home/shilpi/neha/vishakha
```

which means that **vishakha** (your home directory) is in the sub-directory **neha** (the group directory), which in turn is located in the **shilpi** sub-directory, which is in the **home** sub-directory, which is in the top-level root directory called " / " .

EXPERIMENT 4

Object: Copying and moving files in Unix

Description:

Copying Files

cp (copy)

cp *file1 file2* is the command which makes a copy of **file1** in the current working directory and calls it **file2**

Moving files

mv (move)

mv *file1 file2* moves (or renames) **file1** to **file2**

To move a file from one place to another, use the **mv** command. This has the effect of moving rather than copying the file, so you end up with only one file rather than two.

It can also be used to rename a file, by moving the file to the same directory, but giving it a different name.

Removing files and directories

rm (remove), rmdir (remove directory)

To delete (remove) a file, use the **rm** command.

You can use the **rmdir** command to remove a directory (make sure it is empty first).

EXPERIMENT 5

Object: Displaying the contents of a file on the screen

Description:

clear (clear screen)

To clear the terminal window of the previous commands, At the prompt, type

```
% clear
```

This will clear all text and leave you with the % prompt at the top of the window.

cat (concatenate)

The command cat can be used to display the contents of a file on the screen. Type:

```
% cat maanya.txt
```

As you can see, the file is longer than the size of the window, so it scrolls past making it unreadable.

less

The command less writes the contents of a file onto the screen a page at : time. Type

```
% less maanya.txt
```

Press the [**space-bar**] if you want to see another page, and type [**q**] if you want to quit reading. As you can see, **less** is used in preference to **cat** for long files.

head

The **head** command writes the first ten lines of a file to the screen.

First clear the screen then type

```
% head maanya.txt
```

Then type

```
% head -5 maanya.txt
```

What difference did the -5 do to the head command?

tail

The **tail** command writes the last ten lines of a file to the screen.

Clear the screen and type

```
% tail maanya.txt
```

EXPERIMENT 6

Object: To use the vi editor and its modes.

Description: There are many ways to edit files in Unix. Editing files using the screen-oriented text editor **vi** is one of the best ways. This editor enables you to edit lines in context with other lines in the file.

An improved version of the vi editor which is called the **VIM**. VIM stands for **Vi IM**proved.

vi is generally considered a standard in Unix editors because—

- It's usually available on all the flavors of Unix system.
- Its implementations are very similar across the board.
- It requires very few resources.
- It is more user-friendly than other editors such as the **ed** or the **ex**.

You can use the **vi** editor to edit an existing file or to create a new file from scratch. You can also use this editor to just read a text fill

Basic commands to use the vi editor—

Commands and their meaning

vi filename Creates a new file if it already does not exist, otherwise opens an existing file.
vi -R filename Opens an existing file in the read-only mode.
view filename Opens an existing file in the read-only mode.
To create a new file shweta if it already does not exist in the current working directory—

```
$vi shweta
```

Note that a **tilde** (~) on each line following the cursor represents an unused line. If a line does not begin with a tilde and appears to be blank, there is a space, tab, newline, or some other non-viewable character present.

Operation Modes

- **Command mode** enables you to perform administrative tasks such as saving the files, executing the commands, moving the cursor, cutting and pasting the lines or words, as well as finding and replacing. In this mode, whatever you type is interpreted as a command.
- **Insert mode** enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and placed in the file.

vi always starts in the **command mode**. To enter text, you must be in the insert mode for which simply type **i**. To come out of the insert mode, press the **Esc** key, which will take you back to the command mode. If you are not sure which mode you are in, press the Esc key twice; this will take you to the command mode. You open a file using the vi editor. Start by typing some characters and then come to the command mode to understand the difference.

The command to quit out of vi is **:q**. Once in the command mode, type colon, and 'q', followed by return. If your file has been modified in any way, the editor will warn you of this, and not let you quit. To ignore this message, the command to quit out of vi without saving is **:q!**. This lets you exit vi without saving any of the changes.

The command to save the contents of the editor is **:w**. You can combine the above command with the quit command, or use **:wq** and return.

To save your changes and exit **vi** is with the ZZ command. When you are in the command mode, type **ZZ**. The **ZZ** command works the same way as the **:wq** command.

Moving within a File

To move around within a file without affecting your text, you must be in the command mode (press Esc twice).

Commands and their meaning
k Moves the cursor up one line
j Moves the cursor down one line
h Moves the cursor to the left one character position
l Moves the cursor to the right one character position

- vi is case-sensitive. You need to pay attention to capitalization when using the commands.

Control Commands

Commands and their meaning	
CTRL+d	Moves forward 1/2 screen
CTRL+f	Moves forward one full screen
CTRL+u	Moves backward 1/2 screen
CTRL+b	Moves backward one full screen
CTRL+e	Moves the screen up one line
CTRL+y	Moves the screen down one line
CTRL+u	Moves the screen up 1/2 page
CTRL+d	Moves the screen down 1/2 page
CTRL+b	Moves the screen up one page
CTRL+f	Moves the screen down one page
CTRL+I	Redraws the screen

Editing Files

To edit the file, you need to be in the insert mode.

Commands and their meaning	
i	Inserts text before the current cursor location
I	Inserts text at the beginning of the current line
a	Inserts text after the current cursor location
A	Inserts text at the end of the current line
o	Creates a new line for text entry below the cursor location
O	Creates a new line for text entry above the cursor location

Deleting Characters

Commands and their meaning	
x	Deletes the character under the cursor location
X	Deletes the character before the cursor location
dw	Deletes from the current cursor location to the next word
d^	Deletes from the current cursor position to the beginning of the line
d$	Deletes from the current cursor position to the end of the line
D	Deletes from the cursor position to the end of the current line
dd	Deletes the line the cursor is on

Change Commands

To change characters, words, or lines in vi without deleting them.

Commands and their meaning
cc Removes the contents of the line, leaving you in insert mode.
cw Changes the word the cursor is on from the cursor to the lowercase **w** end of the word.
r Replaces the character under the cursor. vi returns to the command mode after the replacement is entered.
R Overwrites multiple characters beginning with the character currently under the cursor. You must use Esc to stop the overwriting.
s Replaces the current character with the character you type. Afterward, you are left in the insert mode.
S Deletes the line the cursor is on and replaces it with the new text. After the new text is entered, vi remains in the insert mode.

Copy and Paste Commands

Commands and their meaning
yy Copies the current line.
yw Copies the current word from the character the lowercase **w** cursor is on, until the end of the word.
p Puts the copied text after the cursor.
P Puts the yanked (cut) text before the cursor.

Word and Character Searching

The vi editor has two kinds of searches: **string** and **character**. For a string search, the **/** and **?** commands are used. When you start these commands, the command just typed will be shown on the last line of the screen, where you type the particular string to look for.

These two commands differ only in the direction where the search takes place—

- The **/** command searches forwards (downwards) in the file.
- The **?** command searches backwards (upwards) in the file.

The **n** and **N** commands repeat the previous search command in the same or the opposite direction, respectively. Some characters have special meanings. These characters must be preceded by a backslash (\) to be included as part of the search expression.

Running Commands

The vi has the capability to run commands from within the editor. To run a command, you only need to go to the command mode and type **:!** command.

EXPERIMENT 7

Object: To study the shut down commands.

Description:

Commands and their meaning	
halt	Brings the system down immediately
init 0	Powers off the system using predefined scripts to synchronize and clean up the system prior to shutting down
init 6	Reboots the system by shutting it down completely and then restarting it
poweroff	Shuts down the system by powering off
reboot	Reboots the system
shutdown	Shuts down the system

You typically need to be the super user or root (the most privileged account on a Unix system) to shut down the system.

VIVA QUESTIONS

Q.1. What is a Kernel?

Ans.
- Master program with Unix operating system.
- Controls the resources of the computer.
- Resources allocation to different users and tasks handle by this section.
- Do not have direct communication with the user.
- Starts separate interactive program call shell to each user when login to the system.

Q.2. Enlist the main features of UNIX.

Ans.
- Machine independent
- Portability
- Pipes and filters
- Background processors
- Utilities
- Development tools
- Multi-user operations
- Unix Shells
- Hierarchical file system

Q.3. What is called Shell?

Ans.
- The interface between user and system called a shell.
- Shell accepts commands and set them to execute for user operations.

Q.4. What are the responsibilities of a shell?

Ans.
- Program Execution

- Input/output redirection
- Filename and variable substitution
- Environment control
- Integrated programming language
- Pipeline hookup

Q.5. What are the different file types available with Unix?

Ans.
- Regular files
- Directory files
- Character special files
- Symbolic links
- Socket
- FIFO
- Block special files

Q.6. Explain file system in UNIX.

Ans.
- Functional unit or a logical collection of files, where the disk is set aside to store files and inode entries.
- Consists of the files that are organized into a multi-level hierarchy called a directory tree.
- File system is a collection of files and directories.
- Very top of the file system is defined as the single directory called root which contains other files and directories and is represented by slash (/).
- These are self-independent and have no dependencies on other file systems.
- Every file and directory are uniquely identified by: Name, Directory in which it resides and a unique identifier.
- All files are organized into a multi-level directory known as 'Directory tree'.

Q.7. Mention the types of path names that can be used in UNIX.

Ans. In a file system ,there exists the hierarchy of directories, their Path is defined as the unique location to a file/ directory to access it

Absolute Pathname: defines a complete path specifying the location of a file/ directory from the very start of the actual file system i.e. from the root directory (/). Absolute pathname addresses system configuration files that do not change location. Defines a complete path specifying the location of a file/ directory from the very start of the actual file system i.e. from the root directory (/).

Relative Pathname: path from the current working directory where the user is i.e. the present working directory (pwd). Signifies current directory, parent directory as well as also refers to file that are either impossible or inconvenient to access. Signifies current directory, parent directory as well as also refers to file that are either impossible or inconvenient to access.

Q.8. Explain mount and unmount command.

Ans. *Mount command* mounts a storage device or file system onto an existing directory and thus making it accessible to users. *Unmount command* unmounts the mounted file system by safely detaching it. It also the task of this command to inform the system to complete any pending read and write operations.

Q.9. What is chmod command? Give its syntax.

Ans. Used to change file or directory access permission and is the most frequently used command in Unix. chmod command changes the permission of each given file.

Syntax:

chmod [options] mode filename.

- -R: recursively change the permission of file or directory.
- -v: verbose, i.e. output a diagnostic for every file processed.
- -c: report only when the change is made.

Q.10. What is called piping?

Ans.
- Used to combine two or more commands together.
- Output of the first command work as the input of the second command, and so on.
- Pipe character (|) is represent piping.
- Allows the user to set the output of a specific operation as input to another.
- Piping of output from one file to another is performed by using the | symbol.

Q.11. What is filter?

Ans.
- A program, which takes input from the standard input, and displays results to the standard output by performing some actions on it.
- Standard Input could be text typed on the keyboard, input from other file or output of other file serving as input. Standard output is y default the display screen.
- Example of Unix filter is grep command.
- Program look for a certain pattern in a file or list of files and only those lines are displayed on the output screen which contains the given pattern.

Q.12. Enlist the main features of Bourne shell.

Ans.
- Standard shell of Unix.
- Gives the ability of input/output redirection.
- Allows the usage of metacharacters for file name abbreviations.
- Environment can be customized using the shell variables.
- Provides the user with a built in command set for the creation of shell programs.
- Allows the user to control a job.

Q.13. Enlist the main features of Korn Shell.

Ans.
- Most advanced as well as an extension to the Bourne Shell which is backward- compatible.
- Allows the user to do command line editing.
- Maintains a command history, which enables the user to check the last commands executed.
- Provision for integer arithmetic.
- Provides the support for arrays and arithmetic expressions.
- Give the user the option to use aliases which is used to abbreviate the command name.

Q.14. What is super block?

Ans.
- The file system of a system is described by a system block.
- Right at the beginning the super block is built when the file system is being created.

- purpose is to contain the basic parameters of a system, e.g., the number of blocks and a count of the total number of files.

Two superblocks are:

- Default super block is always present at a fix offset from the beginning of the system's disk partition.
- Redundant super block is not usually referenced unless the default superblock is effected by some error or system crash.

Q.15. Enlist some unix commands.

Ans.
- **ls**
- **cd**
- **mkdir**
- **rmdir**
- **mv**
- **more**
- **lpr**
- **man**
- **cp**
- **rm**

Questions

1. What is the role kernel?
2. Which program is responsible to bring the login prompt?
3. What is the name of the command to change the password?
4. What is the second column designates in the long listing of `ls` commands's output
5. What are the various ways that you can disconnect/logout with the UNIX server from your console terminal?
6. Name the command which can be used to shut down the system?
7. Is Unix/Linux case sensitive from the console terminal end.
8. Name a command which can be used to restart the machine?
9. What is a process?
10. How to open a file in read only mode using vi editor?
11. Which alphabet keys can move the cursor equivalent to arrow keys?
12. Which command can be used to alter the file access permissions?
13. How can we retrieve the value of the shell variable?
14. Which command can be used to delete a shell variable?
15. What is hidden file?
16. Who designed the UNIX operating system?
17. Which programming language is used in the design of UNIX OS.
18. At which company UNIX is designed first.

19. What does the second column signifies in the output generated by 'who' command?
20. Which command can give the manual of a command?
21. Which command can be used to know about the directory path of a command?
22. What is the role of book block?
23. Which command can be used to create alternate name for an existing command?
24. How can you suppress trailing new line for echo command?
25. Distinguish between **`cmp`** and **`diff`** commands?
26. How can be redirect the output of one command to the another command?
27. What is the effect of '**`cd...`**' if your current working directory is root (*/*)?
28. What is a mount point?
29. What is buddy heap algorithm?
30. What is vi editor?
31. What is a password?
32. What is a root directory?

❑❑❑

12 Chapter

संख्यात्मक प्रश्न बैंक
(NUMERICAL QUESTION BANK)

THINK ABOUT IT

- *There are two types of seeds in the mind: those that create anger, fear, frustration, jealousy, hatred and those that create love, compassion, equanimity and joy. Spirituality is germination and sprouting of the second group and transforming the first group.* — **Amit Ray**

1. State why each of the following is incorrect:
   (i) The shorter the job, the better the service it should receive.
   (ii) Because SJF given performance to short jobs, it is useful in time sharing system.
   (iii) SJF is fair.

2. Assume that you have the following jobs to execute with one processor.

Job	*Burst Time*	*Arrival Time*
0	75	0
1	50	10
2	25	10
3	20	80
4	45	85

Suppose a system uses round-robin scheduling with a quantum of 15.
   (i) Draw a Gantt-chart.
   (ii) Find the average wait and turnaround time.

3. A computer has 6 tapes drive, with '*n*' processes competing for them. Each process may need two drives. For which maximum value of '*n*' in the system deadlock free?

4. Sets of processes are going to execute on a single CPU. They are

Process	*Arrival Time*	*Expected CPU Time*
1	0	14
2	3	12
3	5	7
4	7	4
5	19	7

Determine the sequence of execution for Round Robin (quantum size = 4) CPU scheduling algorithm.

5. Five jobs are waiting to be run. Their expected run time are 9, 6, 3, 5 and X. In what order should they be run to minimize average response time? (Your answer will depend on X).
6. Consider a swapping system in which memory consists of the following hole sizes in memory order: 10 K, 4 K, 20 K, 18 K, 7 K, 9 K, 12 K, and 15 K. Which hole is taken for successive segment requests of 12 K, 10 K, 9 K for worst fit?
7. Calculate turn around time and average waiting time for following set of processes, if these processes are scheduled using
   (i) SJF
   (ii) Priority (both preemptive)

Process id	*Arrival Time*	*Execution Time*	*Priority*
P_1	7	1	0
P_2	3	2	4
P_3	9	3	7

8. Suppose that a disk has 5000 cylinders. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending request in FIFO order is

   80, 1470, 913, 1774, 1509, 1022, 1750, 130.

   What is the total distance that the disk arm moves for the following algorithms:
   (i) FCFS
   (ii) SSTF
   (iii) LOOK
   (iv) C-SCAN
9. Suppose that the following processes arrive for execution at the times indicated:

Process	*Arrival Time*	*Burst Time*
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

   (i) What is the average turnaround time for these processes with the FCFS scheduling algorithm?
   (ii) What is the average waiting and turnaround time for these processes with preemptive SJF algorithm?
10. Consider the set of the processes given in Table and the following scheduling algorithms:
    (i) Shortest Job First (SJF)
    (ii) Shortest Remaining Job First (SRJF)

Process id	*Arrival Time*	*Expected CPU Time*
A	0	7
B	1	5
C	2	3
D	6	2
E	12	3

11. If there is tie within the processes, the tie is broken in the favour of the oldest process.
    (i) Draw the Gantt chart and find the average waiting time and turnaround time for the two algorithms. Comment on your result which one is better and why?
    (ii) If the scheduler takes 0.2 unit of CPU time in context switching for the completed job and 0.1 unit of additional CPU time for incomplete jobs for saving their context, calculate the percentage of CPU time wasted in each case.
12. The head of a moving disk with 100 tracks numbered 0 to 99 is currently serving a request at track 55. If the queue of requests kept in FIFO order is 10, 70, 75, 23, 64, which if the two disk scheduling algorithms FCFS and SSTF will require less head movement? Find the total head movement for each of the algorithms.
13. How long does it take to load a 64 K program from disk whose average seek time is 30 m sec, whose rotation time is 20 m sec and whose tracks hold 32 K:
    (i) for a 2 K page size,
    (ii) for a 4 page size?
14. Explain the difference between external and internal fragmentations. If memory partitions of 100 K, 500 K, 200 K, 300 K, and 600 K (in order) are given, how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 K, 417 K, 112 K, and 426 K (in order)? Which algorithm makes the most efficient use of memory?
15. How many page-faults would occur for the following reference string, for four page frames using LRU algorithm:

    1, 2, 3, 4, 5, 5, 3, 4, ,1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2
16. Suppose that a disk drive has 5000 cylinder. The drive is currently serving a request at cylinder 143 and the previous request was at cylinder 125. The queue of pending request, in FIFO order, is

    86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130.

    What is the total distance (in cylinders) that the disk arm moves to satisfy all the pending request for:
    (i) SSTF
    (ii) SCAN and C-SCAN?
17. Suppose that the following processes arrive for execution at the time indicated.

Process	*Arrival Time*	*Burst Time*
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

    What is the average waiting and twin-around time for these processes with:
    (i) FCFS scheduling algorithm
    (ii) Preemptive SJF algorithm
    (iii) Non-preemptive SJF algorithm.

18. Describe the Banker's algorithm for safe allocation. Consider a system with three processes and three resource types and at time T_0 the following snapshot of the system has been taken:

	Allocated			Maximum			Available		
Process	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	4			

(i) Is the current allocation state safe?

(ii) Would the following requests be granted in the current state?

(a) Process P_2 requests (1, 0)

(b) Process P_1 requests (1, 0)

19. If FIFO page replacement is used with four page-frames and eight pages, how many page-faults will occur with the reference string 0 1 7 2 3 2 7 1 0 3 if the four frames are initially empty?

20. Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

(a) 0, 430

(b) 1, 12

(c) 2, 500

(d) 3, 400

(e) 4, 110

21. Page size are kept in powers of 2. Why?

22. Explain the priority scheduling algorithm and its major drawbacks with their solution. Draw the Gantt chart and find average waiting time and response time of the process set given in the following table:

Process id	Arrival Time	Execution Time	Priority
A	0	10	3
B	0	2	1
C	1	3	3
D	2	1	5
E	2	4	2

23. Consider a logical address space of eight pages of 2^{10} words, each mapped on to a physical memory of 64 frames:
    (i) How many bits are there in the logical address?
    (ii) How many bits are there in the physical address?

24. Consider the following segment table:

Segment	*Base*	*Length*
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?
(i) 0, 430
(ii) 1, 10
(iii) 1, 11
(iv) 2, 500
(v) 3, 400
(vi) 4, 112
(vii) 4, 89
(viii) 3, 505
(xi) 3, 600
(x) 1, 100

25. It is proposed to use a multi-resource Banker's algorithm in an operating system containing 3 resource classes. The number of resource units available for allocation is 7, 7 and 10 respectively. The current resource allocation state is shown as:

	Allocated resources			*Maximum requirement*			*Available resources*		
Process	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	4			

(i) Is the current allocation state safe?
(ii) Would the following requests be granted in the current state?
- Process P_1 requests (1, 1, 0)
- Process P_2 requests (0, 1, 0)
- Process P_3 requests (0, 1, 0)

26. Assume, you have the following jobs to execute with one processor:

Job	*Burst Time*	*Arrival Time*	*Priority*
1	10	0	3
2	3	1	1
3	2	2	3
4	4	3	4
5	5	4	2

where priority is defined as

1 > 2 > 3 > 4 > ... ...

(i) Give a Gantt chart illustrating the execution of these jobs using pre-emptive priority based algorithm.

(ii) Calculate average turnaround and waiting time.

27. It is proposed to use a multi-resource Banker's algorithm in an OS containing 3 resource classes. The number of resource units available for allocation is 7, 7 and 10 respectively. The current resource allocation state is shown as given below:

	Allocated resources			*Maximum requirement*			*Available resources*		
Process	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	4			

(i) Is the current allocation state safe?

(ii) Would the following requests be granted in the current state?

- Process P_1 requests (1, 1, 0)
- Process P_2 requests (0, 1, 0)
- Process P_3 requests (0, 1, 0)

28. Consider the following shape shot of processes and compute average turn around time and waiting time of processes for FCES, SJF algorithms.

Process	*Arrival Time (ms)*	*Next Burst Time (ms)*
P_1	0.0	6
P_2	0.5	4
P_3	1.0	2

29. If a FIFO page replacement is used with four page frames and eight pages, how many page faults will occur with reference string 0, 1, 7, 2, 3, 2, 7, 1, 0, 3 if the four frames are initially empty?

30. It is proposed to use a multi-resource Banker's algorithm in an operating system containing 3 resource classes. The number of resource units available for allocation is 7, 7 and 10 respectively. The current resource allocation state is shown as given below:

	Allocated resources			*Maximum requirement*			*Available resources*		
Process	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	4			

(i) Is the current allocation state safe?

(ii) Would the following requests be granted in the current state?

- Process P_1 requests (1, 1, 0)
- Process P_2 requests (0, 1, 0)
- Process P_3 requests (0, 1, 0)

31. Assume, you have the following jobs to execute with one processor:

Job	*Burst Time*	*Arrival Time*	*Priority*
1	10	0	3
2	3	1	1
3	2	2	3
4	4	3	4
5	5	4	2

where priority is defined as

1 > 2 > 3 > 4 > ... ...

(i) Give a Gantt chart illustrating the execution of these jobs using pre-emptive priority based algorithm.

(ii) What is the average turn-around time?

(ii) What is average waiting time?

32. Five batch jobs *A* through *E*, arrive at a computer centre at the same time. They have estimated running times of 10, 6, 2, 4 and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1 and 4 respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turn around time. Ignore process switching overhead:

(i) Round Robin

(ii) First come, first-served (run in order 10, 6, 2, 4, 8)

(iii) Shortest job first.

For (i) assume that the system is multiprogrammed and that each job gets its fair share of CPU. For (ii) and (iii) assume that only one job at a time runs, until it finishes.

33. Suppose that a total of 64 MB memory is available in a system. This memory space is partitioned into 8 fixed size slot of 8 MB each. Assume 8 processes are currently requesting memory usage with sizes indicated as below:

[2 MB, 4 MB, 3 MB, 7 MB, 9 MB, 1 MB, 8 MB]

Calculate the size of memory wasted due to external and internal fragmentation. Derive the memory utilization ratio by dividing the total allocated memory by total requested memory.

34. Consider the following page reference string:

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3

How many page faults would occur for the following replacement algorithm, assuming three frames (remember that all frames are initially empty)

(i) FIFO replacement
(ii) LRU replacement
(iii) Optimal replacement

35. What do you understand by Belady's anomaly. Which popular page replacement algorithm suffers from the Belady's anomaly? Also give the name of the class of algorithms, which can never suffer from Belady's anomaly and why?

A system using demand-paged memory, takes 250 ns to satisfy a memory request if the page is in memory. If the page is not in memory, the request takes on an average 5 ms if a free frame is available or the page to the swapped out has not been modified or 12 ms if the page to be swapped out has been modified. What is the effective access time if the page fault rate is 2%, and 40% of the time the page to be replaced has been modified? Assume the system is running only a single process and the CPU is idle during page swaps.

36. The head of a moving head disk with 200 tracks numbered 0 to 199 is currently serving a request at track 143, consider an ordered disk queue with requests involving track numbers:

86, 147, 91, 177, 94, 150, 102, 175, 130.

What is the total head movement to satisfy this request for the following disk scheduling algorithms?

(i) FCFS (First come first serve)
(ii) SSTF (Shortest seek time first)

37. Consider a variation of round robin, we will call progressive round robin. In progressive round robin each process has its own quantum. This starts out as 50 ms, and increases by 50 ms each time it goes through the round robin queue. So long jobs keep getting longer and longer time slices. Give the advantages of this variant over ordinary round robin. You are also required to give disadvantages of this variant over ordinary round robin.

38. Assume you have the following job to execute with one processor:

Jobs	*Burst Time*	*Arrival Time*
1	9	0
2	5	2
3	6	3
4	4	5
5	8	6

(i) Given Gantt Chart illustrating the execution of these jobs using Shortest Remaining Time First (SRTF) algorithm.
(ii) Calculate average turn around and waiting time.

13 Chapter

तैरते को नाव का सहारा
(TAIRTE KOH NAAV KA SAHARA)

THINK ABOUT IT

"Believe in yourself! Have faith in your abilities! Without a humble but reasonable confidence in your own powers you cannot be successful or happy."

—Norman Vincent Peale

प्रश्न 1—ऑपरेटिंग सिस्टम क्या होता है? यह कितने प्रकार का होता है।

उत्तर—ऑपरेटिंग सिस्टम एक ऐसा प्रोग्राम होता है जो user, तथा computer hardware के बीच एक मध्यस्थ (mediator) की भूमिका निभाता है। यह प्रोग्राम एक ऐसा वातावरण प्रदान करता है जिसमें कि user अपने प्रोग्राम्स को execute कर सके। एक ऑपरेटिंग सिस्टम कम्प्यूटर को user के लिए आसान (user friendly) बनाता है तथा कम्प्यूटर हार्डवेयर के बेहतर प्रयोग हेतु वातावरण तैयार करता है।

"ऑपरेटिंग सिस्टम वह प्रोग्राम या सॉफ्टवेयर होता है जो यूज़र तथा हार्डवेयर के मध्य इंटरफेस या link का कार्य करता है। अतः, ऑपरेटिंग सिस्टम एक प्रोग्राम का होता है, जो कम्प्यूटर को कन्ट्रोल करता है।"

"An Operating System is a program that acts as an intermediatly (मध्यस्थ) between the user of the computer and the hardware of the computer. It provides an environment in which a user can execute the programs in efficiently and conviniently."

विभिन्न प्रकार के ऑपरेटिंग सिस्टम (Different Types of Operating System)

आधुनिक कम्प्यूटर सिस्टम तीन ग्रुप्स में विभक्त किये जा सकते हैं—

बैच सिस्टम, टाइम शेयर्ड सिस्टम तथा रियल टाइम ऑपरेटिंग सिस्टम इनकों classify करने का आधार यह है कि कम्प्यूटर के user तथा उसके प्रोग्राम का किस प्रकार से प्रौसेसिंग के समय interaction होता है।

बैच प्रोसैसिंग ऑपरेटिंग सिस्टम में user एक central place पर अपने jobs को submit करता है जहाँ jobs को एक batch के रूप में collect किया जाता है। इनको कम्प्यूटर पर एक पंक्ति (queue) के रूप में रखा जाता है तथा एक एक करने proceps किया जाता है। अतः, processing के समय user का job से कोई interaction नहीं होता।

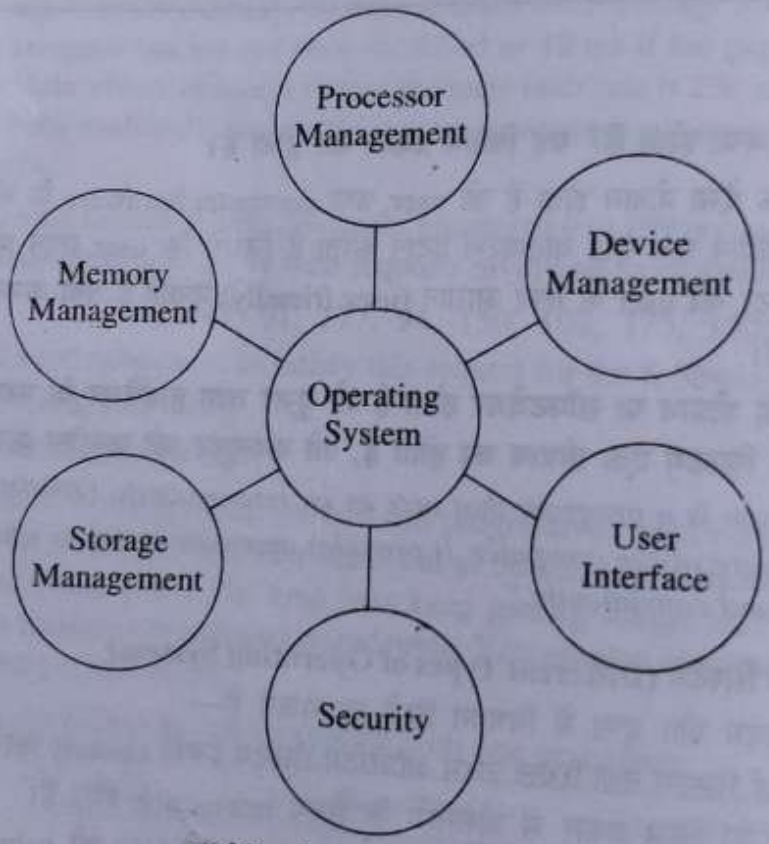
टाइम शेयरिंग सिस्टम में ऑपरेटिंग सिस्टम कम्प्यूटर एक समय में कई users को services प्रदान करता है। इस प्रकार कई users एक साथ central processor, memory तथा कम्प्यूटर के दूसरे resources को शेयर करते हैं। इस प्रकार के सिस्टम में user को प्रोग्राम से execution के समय full interaction रहता है।

रियल टाइम सिस्टम्स द्वारा उन applications को serve किया जाता है जहाँ कम्प्यूटर से immediate response की आवश्यकता होती है, अर्थात response time अत्यंत कम रखना होता है। Airline reservation systems, machine tool control, Nuclear Power Statio.ı की monitoring, getting a stock market quotation इत्यादि में real time system use किये जाते हैं।

प्रश्न 2—ऑपरेटिंग सिस्टम्स द्वारा प्रदान की जाने वाली सेवाओं का उल्लेख कीजिये।

उत्तर—ऑपरेटिंग system के मुख्य कार्य निम्नवत् है—

- मैमोरी प्रबन्धन (Memory Management)
- प्रौसेसर प्रबन्धन (Processor Management)
- युक्ति प्रबन्धन (Device Management)
- सचिका प्रबन्ध (File Management)
- सुरक्षा (Security)
- सिस्टम के कार्य करने की क्षमता पर नियन्त्रण (Control Over System Performance)
- जॉब एकाउन्टिग (Job Accounting)
- त्रुटियों को डिटेक्ट करने में Help (Error Detecting Aids)
- साफ्टवेयर तथा यूज़र के मध्य समन्वय स्थापित करना (Coordination between other Software and Users)



ऑपरेटिंग सिस्टम के विभिन्न कार्य

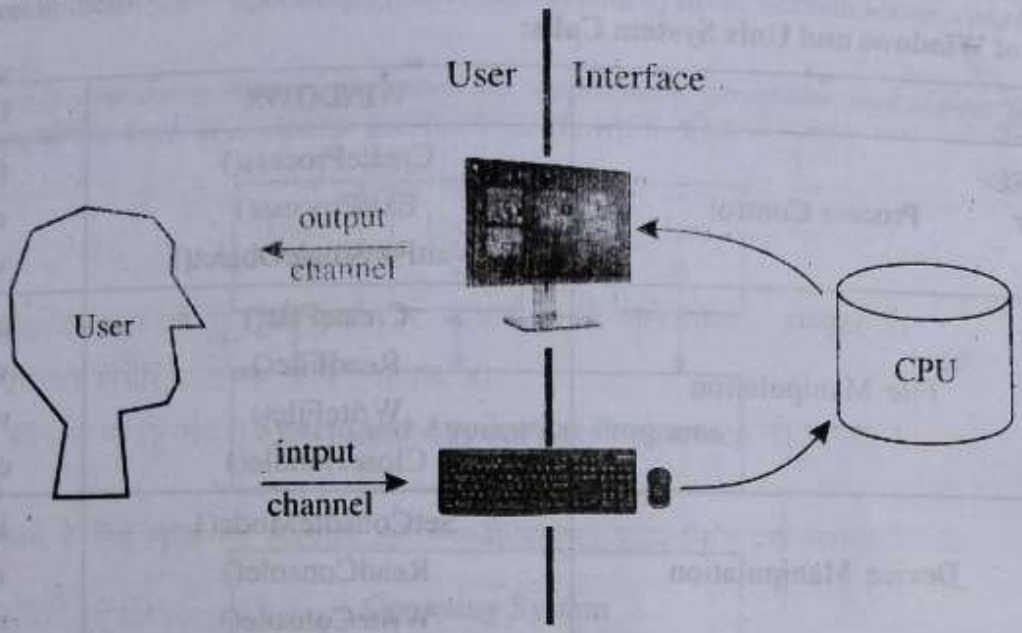
प्रश्न 3—User Operating System Interface से आप क्या समझते हैं? यह कितने प्रकार का होता है?

उत्तर—Character User Interface में Character को keyboard द्वारा टाइप किया जाता है। इसमें computer को निर्देश Command द्वारा दिये जाते हैं। Graphical user interface में user graphics के द्वारा application के साथ interact करता है जैसे कि Icons, button, window आदि।

A user interface (UI) refers to the part of an operating system, program, or device that allows a user to enter and receive information. A text-based user interface or a character user interface displays text, and its commands are usually typed on a command line using a keyboard. With a graphical user interface, the functions are carried out by clicking or moving buttons, icons and menus by means of a pointing device.

Most GUIs have the following basic components:

- *A start menu with program groups*
- *A taskbar showing running programs*
- *A desktop*
- *Various icons and shortcuts.*



प्रश्न 4—सिस्टम कॉल्स (System Calls) क्या होती हैं? विभिन्न प्रकार की सिस्टम कॉल्स का उल्लेख कीजिये। * ट * ट

उत्तर—सिस्टम कॉल (System Call)—System Call वह विधि है जिसके द्वारा कोई Computer Program ऑपरेटिंग सिस्टम के kernel से किसी सेवा (service) का अनुरोध करता है। अत:, System Call द्वारा Programs ऑपरेटिंग सिस्टम से interact करते हैं। जब भी Computer Program ऑपरेटिंग सिस्टम के kernel को कोई request भेजता है तो वह सिस्टम कॉल द्वारा ऐसा करता है। System Call ऑपरेटिंग सिस्टम की services को User Programs को Application Program Interface (API) द्वारा मुहैया (उपलब्ध, provide) कराती है। API या Application Program interface process तथा operating system के मध्य interface (मध्यस्थता) प्रदान करता है ताकि user level processes ऑपरेटिंग सिस्टम की services के लिए request भेज सके। System calls kernel system के प्रवेश द्वारा कार्य करती है। अत:, जिस भी प्रोग्राम को resources की आवश्यकता होती है, वह system calls का use करता है।

A system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system's kernel. System call provides the services of the operating system to the user programs via Application Program Interface (API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

Services Provided by System Calls:

- *Process creation and management*
- *Main memory management*
- *File Access, Directory and File system management*
- *Device handling(I/O)*
- *Protection*
- *Networking, etc.*

Types of System Calls:

- *Process control: end, abort, create, terminate, allocate and free memory.*
- *File management: create, open, close, delete, read file etc.*
- *Device management.*
- *Information maintenance.*
- *Communication.*

Examples of Windows and Unix System Calls:

	WINDOWS	UNIX
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

प्रश्न 5—Application Programs क्या होते हैं?

उत्तर—Application Programs—यह प्रोग्राम्स User द्वारा कोई specific function करने के लिये use किये जाते हैं। इनके उदाहरण हैं—word processors, web browsers, gaming softwares, graphic software, media players, etc. चूँकि यह सब program और user को कोई न कोई application provide करते हैं, इसलिए यह application program कहलाते हैं। उदाहरण के लिये— web browser internet से सूचना search करने के लिए use होता है, gaming software games खेलने के लिए use होता है, Word processor document उत्पन्न करने के लिए use होता है, आदि।

An operating system is a construct that allows the user application programs to interact with the system hardware. Since the operating system is such a complex structure, it should be created with utmost care so it can be used and modified easily. An easy way to do this is to create the operating system in parts. Each of these parts should be well defined with clear inputs, outputs and functions.

प्रश्न 6—System Program क्या होते हैं?

उत्तर—यह प्रोग्राम operating system software को program करने के लिए use होते हैं। जबकि application program वह software प्रदान करते हैं जोकि user द्वारा directly use किया जाता है जबकि system program वह software प्रदान करते हैं जोकि दूसरे systems जैसे कि SaaS applications, computational science applications आदि use किये जाते हैं। System program के उदाहरण हैं—operating system, networking system, website server, data backup server आदि।

There are mainly two categories of programs i.e. application programs and system programs (see figure). Their place in the logical computer hierarchy is shown in figure:

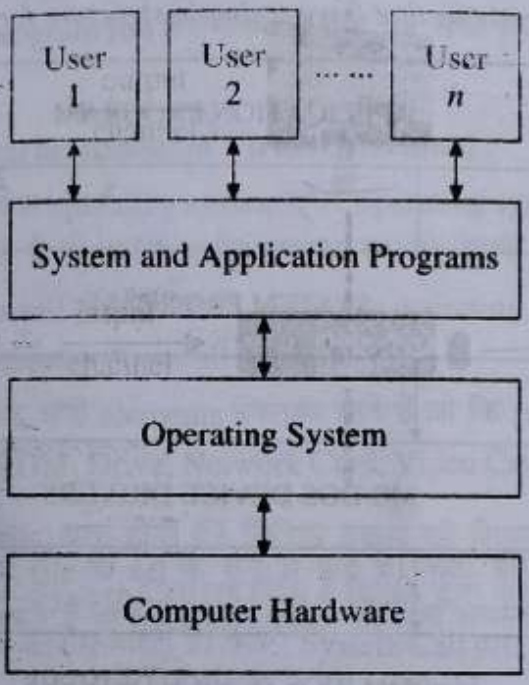


Figure: *Application programs and system programs and their place in the logical computer hierarchy*

Application Programs

These programs perform a particular function directly for the users. Some of the common application programs include email, web browsers, gaming software, word processors, graphics software, media player etc.

All of these programs provide an application to the end users, so they are known as application programs. For example: a web browser is used to find information, a gaming software is used to play games, a word processor is used to create documents etc.The requests for service and application communication systems used in an application by a programmer is known as an application program interface (API).

System Programs

The system programs are used to program the operating system software. While application programs provide software that is used directly by the user, system programs provide software that are used by other systems such as SaaS applications, computational science applications etc. Some examples of system programs are operating system, networking system, web site server, data backup server etc.

प्रश्न 7—Operating System Structure पर टिप्पणी लिखिये।

उत्तर—हम जानते हैं कि operating system एक software है जिसके माध्यम से application program system hardware से interact कर सकते हैं। चूँकि operating system एक complicated संरचना होती है अतः इसको अत्यन्त सावधानी से

बनाना चाहिए ताकि इसको आसानी से use तथा modify किया जा सके। इसके लिए इसको parts में बनाया जाता है तथा प्रत्येक part के inputs, outputs तथा functions को clearly defined किया जाता है। चित्र में MS-DOS का structure प्रदर्शित है।

कुछ operating system का structure layered होता है जिससे modularity प्राप्त की जा सकती है (चित्र देखें)। इस प्रकार के structure में bottom layer hardware होता है जबकि top most layer user interface होता है जैसा कि चित्र से स्पष्ट है प्रत्येक upper layer bottom layer के ऊपर बनाया जाता है अत: सभी layers अपने से ऊपर वाले layers से कुछ structure, operation आदि hide कर सकते हैं। Layered structure में प्रत्येक layer सावधानीपूर्वक define होना आवश्यक है।

There are many operating systems that have a simple structure for example MS-DOS (see Figure).

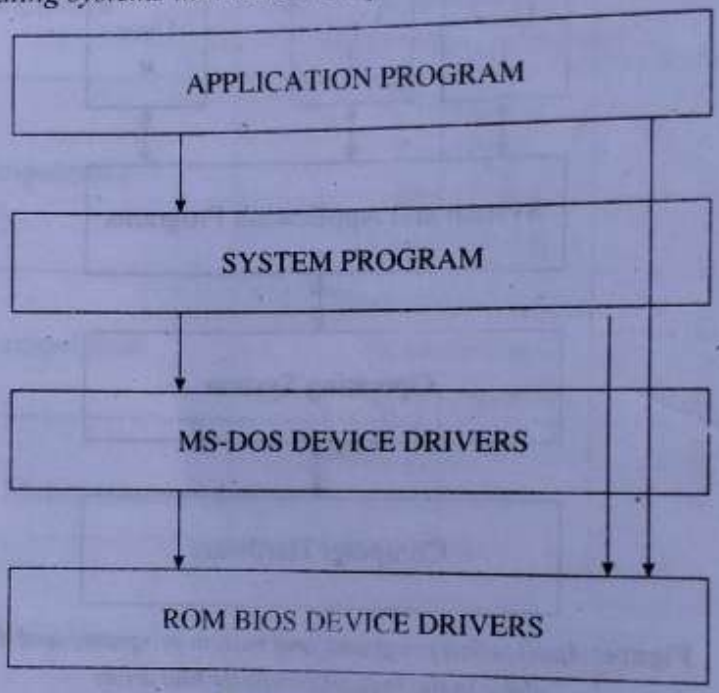


Figure: *MS-DOS Structure*

Some operating systems have a modular structure, unlike MS-DOS leading to greater control over the computer system and its various applications. The modular structure would allows the programmers to hide information as required and implement internal routines as they see fit without changing the outer specifications.

One way to achieve modularity in the operating system is the layered approach. In this, the bottom layer is the hardware and the topmost layer is the user interface(see Figure).

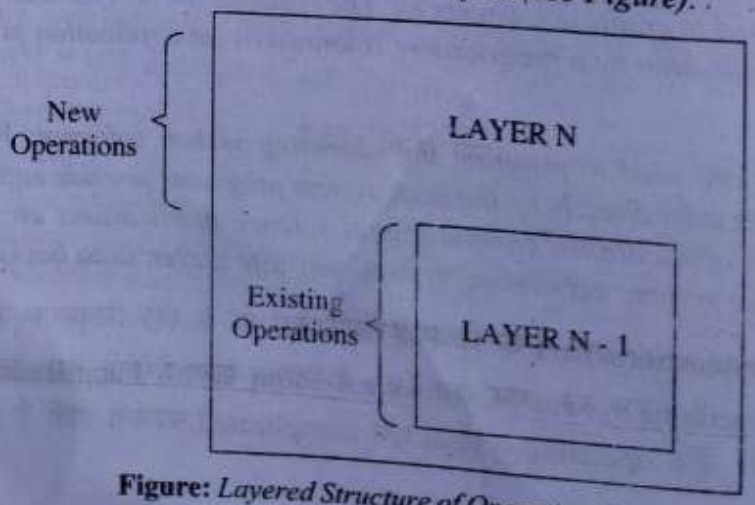


Figure: *Layered Structure of Operating System*

As seen from the image, each upper layer is built on the bottom layer. All the layers hide some structures, operations etc from their upper layers.

One problem with the layered structure is that each layer needs to be carefully defined. This is necessary because the upper layers can only use the functionalities of the layers below them.

प्रश्न 8—Virtual Machine क्या होती है? इसके गुण, कार्यप्रणाली व लाभों का वर्णन कीजिये।

उत्तर—Virtual machine एक software होता है जोकि हम अपने computer पर install करते हैं। इस software से हम अपने operating system पर दूसरे operating system को install तथा use कर सकते हैं। इस प्रकार हम कह सकते हैं कि virtual machine एक ऐसा software है जिसकी सहायता से हम एक ही hardware पर कई सारे operating systems को एक साथ run कर सकते हैं। अत: वह operating system जो virtual machine run करती है, virtualized operating system कहलाते हैं। Virtualized OS program run कर सकते हैं तथा वह सभी कार्य कर सकते हैं जोकि real operating system करते हैं।

Virtual Machine की विशेषतायें (Characteristics of Virtual Machine) :

(i) Virtualize किया जाने वाला प्रत्येक operating system दूसरे operating system से स्वतन्त्र होता है। अत: यदि कोई मशीन कार्य करना बन्द कर देती है तो दूसरी मशीन स्वतन्त्र रूप से अपना कार्य करती है।

(ii) एक बार virtual machine में install होने के पश्चात virtualize operating system ठीक उसी प्रकार से कार्य करता है जैसे कि हमने उसे अपने computer में install कर लिया हो।

(iii) Virtual machine के पास भी वह सभी elements उपलब्ध रहते हैं जो कि real computer को उपलब्ध रहते हैं जैसे कि—Hard disk, RAM, CD ROM, Drive, Network Card, Video Card आदि किन्तु यह elements physical होने के बजाय virtual होते हैं।

(iv) Virtual machine के सभी तत्व file के set के रूप में होते हैं। अत: हम virtual operating system को दूसरे virtual system में copy कर सकते हैं या back up copies को बिना समस्या के तेजी से तथा आसानी से बना सकते हैं।

Virtual Machine कैसे कार्य करती है (How does a Virtual Machine Work) :

Virtual machine एक software होता है जोकि virtualization की परतों के माध्यम से hardware से संपर्क करता है तथा computer के components उपलब्ध कराता है। अत: virtual machine की सहायता से hard disk, RAM Memory, Network card, Processor आदि को emulate (अनुकरणित) किया जा सकता है।

Virtual machine द्वारा प्रदत्त Utilities (Utilities Provided by Virtual Machine) :

- Operating system को test करना।
- वह software use करना जोकि हमारे operating system में उपलब्ध नहीं है।
- उन software को use करना जोकि ऐसे operating system पर run किये जा सकते हैं जोकि absolute (अप्रचलित) हो चुके हैं।
- ऐसे experiments करना जोकि हम अपने operating system में करना risky समझते हैं, जैसे कि कोई software update करना, किसी suspicious website को surf करना, suspicious email खोलना आदि।
- एक ही computer की सहायता से computer network उत्पन्न तथा simulate करना तथा इस network को training purpose या network administration के ज्ञान के लिए use करना।

वर्चुअल मशीन के लाभ (Advantages of Virtual Machine) :

- यदि सर्वर या virtualised operating system unconfigured है (अर्थात् configured नहीं है), तो real machine की तुलना में इसको restore करना काफी सरल होता है।

- Business वातावरण में operating system तथा servers के virtualisition से धन व स्थान की बचत होती है।
- इसमें कम physical equipments की आवश्यकता होती है, अत: रखरखाव एवं ऊर्जा व्यय के खर्च कम हो जाते हैं।
- Production environment में operating system तथा servers का विस्तार आसानी से किया जा सकता है।
- विभिन्न प्रकार की टैस्टिंग हेतु वातावरण तैयार किया जा सकता है जोकि बाकी सिस्टम से isolated रहता है।
- किसी virtualised system को हटा देने पर भी शेष अपना करते रहते हैं।

A virtual machine is software that we install on our computer. This software allows us to install and use other operating systems simultaneously on our operating system. A virtual machine is a software that will allow us to run several operating systems simultaneously on the same hardware. The operating systems that the virtual machine runs are called virtualized operating systems. These virtualized operating systems can run programs and perform all the tasks that we could perform in a real operating system.

Characteristics does a virtual machine :

- *The vast majority of virtual machines, such as Virtual box allow installing virtually any operating system such as Linux, Android, Mac OS X, Windows, Chrome OS, etc.*
- *Each of the operating systems that we virtualize is completely independent of the other operating systems. In this way, in the case that one of the virtual machines stops working, the rest will continue working without any type of problem.*
- *Once an operating system is installed in the virtual machine, we have to use the virtualized operating system in the same way that we would use it if we had installed it in our computer.*
- *A virtual machine has all the elements available to a real computer. It has a hard disk, RAM, CD-ROM drive, network card, video card, etc., but unlike a real computer, these elements, instead of being physical, are virtual.*
- *All the elements of a virtual machine are encapsulated in a set of files. This allows us to copy a virtual operating system from one computer to another or we can make backup copies without any problem and very easily and very quickly.*

Utilities are provided by virtual machines:

- *To test operating systems.*
- *To use software that is not available in our operating system.*
- *to use software that can only be run on operating systems that are obsolete.*
- *We can experiment in the operating system that runs inside the virtual machine doing things that we would not dare to do with our operating system, such as applying a software update, surfing safely on a web page that we consider suspicious, etc.*
- *We can create/simulate a computer network with just one computer. We can use this network of virtualized computers for training purposes and in this way acquire knowledge about network administration.*
- *If you are a software developer you can test if the program you are developing works correctly in several operating systems.*
- *To test alpha, Beta and Release candidate versions of certain programs and operating systems.*
- *To mount a web server, a VPN server, a mail server or any other type of server.*

Advantages of virtual machines :

- *If a server or a virtualized operating system is unconfigured, it is extremely easy to restore if compared to a real machine.*
- *If we talk about the business environment, the virtualization of operating systems and servers supposes an economic saving and considerable space.*

- *use of virtual machines implies having less physical equipment. Therefore, the fact of virtualizing servers or operating systems can mean significant savings in maintenance and energy consumption.*
- *Through virtualization and dynamic balancing we can increase the service rates of a server*
- *If we are using a virtual machine in a production environment, we can expand the resources of an operating system or server*
- *to create an environment for testing of all kinds. In this way we will easily obtain a test environment completely isolated from the rest of the systems.*
- *Virtual machines and virtualization allow to use a single service per virtualized server easily and simply. In this way, even if one of the virtualized servers is dropped, the other will continue to work.*

प्रश्न 9—प्रोसेस मैनेजमेंट से आप क्या समझते हैं? Process, Process State व Process Control Block क्या होता है?

उत्तर—जो Program वर्तमान में execute हो रहा हो उसे process कहा जाता है। चूँकि process का execution एक क्रमिक ढंग में चलना चाहिए अत: process वह entity होती है जोकि किसी system में implement होने वाले basic unit of work को प्रदर्शित करती है। साधारण शब्दों में कहा जाये तो हम अपना program text file में लिखते हैं तथा जब हम उस program को execute करते हैं तब वह process बन जाता है तथा वह सारे कार्य करता है जोकि program में वर्णित होते है। अत: जब program memory में load होता है और process बन जाता है तब उसे four sections में divide किया जा सकता है— stack, heap, text, data.

Process states—जब Program execute करता है तो वह कई state से pass करता है। यह states अलग-अलग operating system में अलग-अलग हो सकते हैं किन्तु सामान्यत: कोई process निम्न में से किसी एक state में हो सकता है—

- Start
- Ready
- Running
- Waiting
- Terminated
- Exit

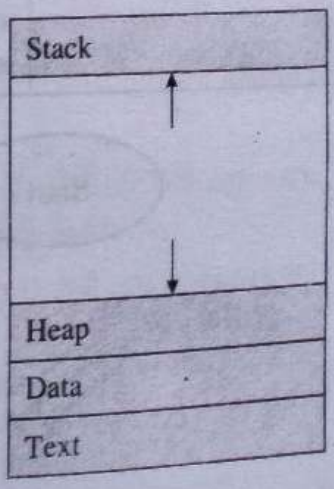
प्रोसेस कंट्रोल ब्लॉक (Process Control Block)—Process control block (या PCB) एक data structure होता है जोकि operating system द्वारा maintain किया जाता है। PCB को एक integer process ID (PID) द्वारा पहचाना जाता है तथा PCB किसी भी process की सभी जानकारी रखता है जोकि उस process को track करने के लिए आवश्यक होती है।

Operating system द्वारा किसी process को manage करने के लिए किये गये सभी कार्य process management कहलाते हैं।

Process

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

A process is defined as an entity which represents the basic unit of work to be implemented in the system. We write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program. When a program is loaded into the memory and it becomes a process, it can be divided into four sections—stack, heap, text and data.



Component	Description
Stack	The process Stack contains the temporary data such as method/function parameters, return address and local variables.
Heap	This is dynamically allocated memory to a process during its run time.
Text	This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.
Data	This section contains the global and static variables.

Program

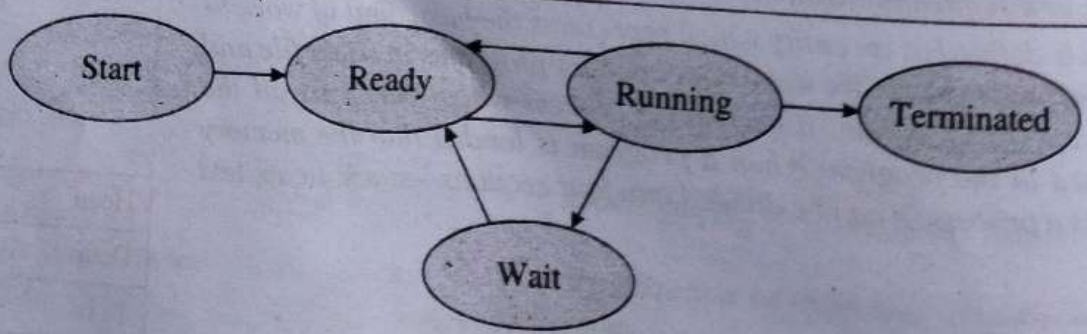
A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language. A computer program is a collection of instructions that performs a specific task when executed by a computer. A process is a dynamic instance of a computer program.A part of a computer program that performs a well-defined task is known as an algorithm. A collection of computer programs, libraries and related data are referred to as a software.

Process Life Cycle

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

A process can have one of the following five states at a time.

State	Description
Start	This is the initial state when a process is first started/created.
Ready	The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.
Running	Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.
Waiting	Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.
Terminated or Exit	Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.



Process Control Block (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process:

Information	Description
Process State	The current state of the process i.e., whether it is ready, running, waiting, or whatever.
Process Privileges	This is required to allow/disallow access to system resources.
Process ID	Unique identification for each of the process in the operating system.
Pointer	A pointer to parent process.
Program Counter	Program Counter is a pointer to the address of the next instruction to be executed for this process.
CPU registers	Various CPU registers where process need to be stored for execution for running state.
CPU Scheduling Information	Process priority and other scheduling information which is required to schedule the process.
Memory Management Information	This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
Accounting Information	This includes the amount of CPU used for process execution, time limits, execution ID etc.
IO Status Information	This includes a list of I/O devices allocated to the process.

The architecture of a PCB is completely dependent on Operating System and may contain different information in different operating systems.

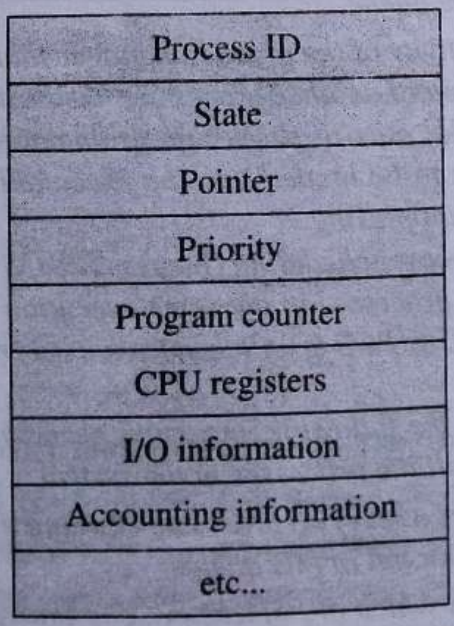


Figure: *Simplified diagram of a PCB*

The PCB is maintained for a process throughout its lifetime, and is deleted once the process terminates.

***Process scheduling** is an essential part of a multiprogramming operating systems. It contains various operation such as:*

- ***Job Queue**: It contains the process entered into the system for execution. This queue consists of all processes in the system; those processes are entered into the system as a new process*
- ***Ready Queue**: Ready Queue contains the process that is ready for execution. This queue consists of the processes that are residing in the main memory and are ready and waiting to execute by CPU. This queue is generally stored as a linked list. A ready-queue header contains pointers to the first and final PCB in the list. Each PCB includes a pointer field that points to the next PCB in the ready queue.*

प्रश्न 10—Scheduler क्या है? Job scheduler और process scheduler क्या होता है?

उत्तर—शैडयूलर्स विशेष प्रकार के सिस्टम साफ्टवेयर होते हैं जो विभिन्न तरीकों से process scheduling को handle करते हैं। इनका मुख्य कार्य सिस्टम को submit किये गये विभिन्न jobs को चयन करने तथा run करने का निर्णय लेना होता है।

Schedulers are special system softwares which handles process scheduling in different ways. Their main work is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types:

- *Long Term Scheduler (लॉंग टर्म शैडयूलर्स)*
- *Short Term Scheduler (शॉर्ट टर्म शैडयूलर्स)*
- *Medium Term Scheduler (मीडियम टर्म शैडयूलर्स)*

Job scheduler job queue से process select करता है और उसे ready queue में allocate करता है जबकि CPU scheduler ready queue में से किसी एक process को select करता है और उस process को processor allocate करता है।

Job Scheduler and CPU Scheduler

The job scheduler selects a process from job queue and allocates it into the ready queue whereas the CPU scheduler, select one of the processes from the ready queue and allocate the processor to that process.

- ***Device Queue**: This queue consists of the processes that are waiting for a particular I/O device. Each device has its own device queue.*

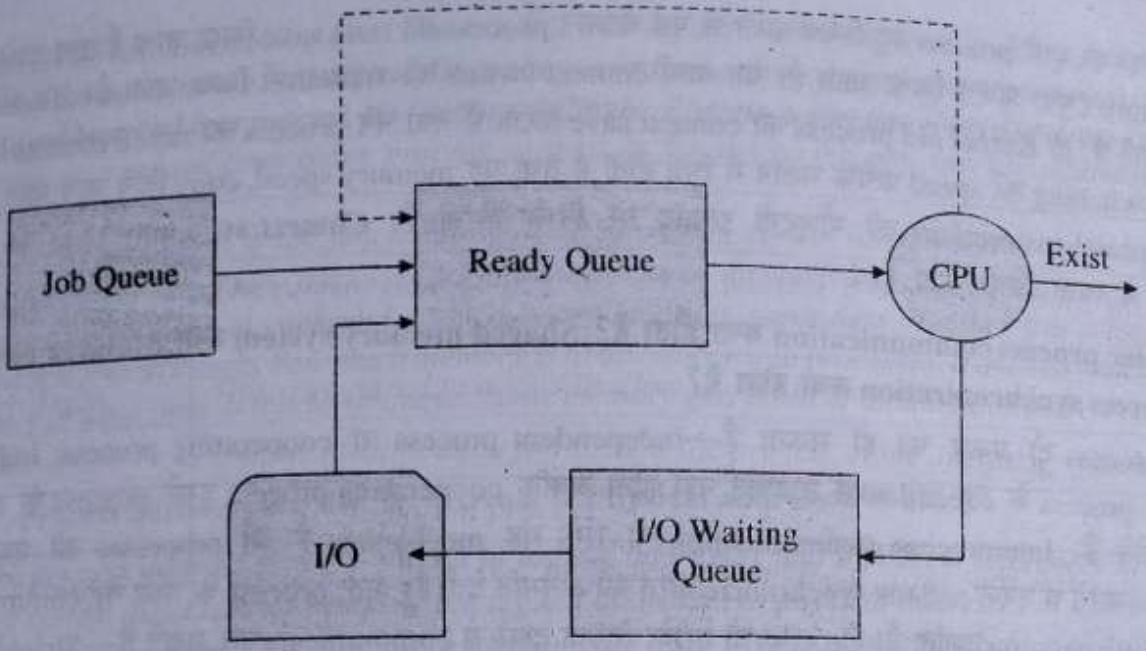
The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues—

- ***Job queue**: This queue keeps all the processes in the system.*
- ***Ready queue**: This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.*
- ***Device queues**: The processes which are blocked due to unavailability of an I/O device constitute this queue.*



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states:

State	Description
Running	*When a new process is created, it enters into the system as in the running state.*
Not Running	*Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.*

Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types —

- *Long-Term Scheduler*
- *Short-Term Scheduler*
- *Medium-Term Scheduler*

प्रश्न 11—Context switch क्या होता है?

उत्तर—जब भी CPU को interrupt (व्यवधानित) किया जाता है, तो वह अपना वर्तमान कार्य छोड़कर Interrupt service subroutine को serve करता है। जब भी कोई Interrupt होती है तो System उस समय CPU पर run कर रहे process का current context save करता है तथा उसके बाद ही Interrupt को serve करता है ताकि processing के पश्चात् वापस लौटने पर process को पुन: शुरू करते समय उसे वह context वापस प्राप्त हो जायें। Context को process के PCB (Process Control Block) से व्यक्त किया जाता है, इसमें CPU registers की स्थिति, process state, memory management information इत्यादि होती है।

अत: CPU को दूसरे process हेतु स्विच करने से पूर्व वर्तमान process का state save किया जाता है तथा दूसरे process का state restore (पुन: प्राप्त) किया जाता है। यह कार्य context switch की सहायता से किया जाता है। जब भी context switching होती है, तो Kernel old process का context save करता है तथा नये process का saved context load करता है। Context switching की speed प्रत्येक मशीन में भिन्न होती है तथा यह memory speed, copy किये जाने वाले registers की संख्या, special instructions की मौजूदगी इत्यादि पर निर्भर करती है। Context switching time को सामान्यत: मिलीसैकेन्ड्स में व्यक्त किया जाता है।

प्रश्न 12—Inter process communication क्या होता है? Shared memory system तथा Message passing क्या होता है? Process synchronization क्या होता है?

उत्तर—एक process दो प्रकार का हो सकता है—independent process या cooperating process. Independent process दूसरे process के execution से प्रभावित नहीं होता जबकि cooperating process दूसरे process के execution से प्रभावित होते हैं। Interprocess communication या IPS वह mechanism है जो processes को एक दूसरे से communicate करने व उनके actions synchronize करने की अनुमति देती है। अत: process के मध्य यह communication उनके मध्य का cooperation दर्शाता है। Ûrocess दो तरीके से एक दूसरे से communicate कर सकते हैं—Shared memory method तथा Message passing method।

अंत:प्रौसेस संचार, नैटवर्क स्ट्रक्चर तथा सुरक्षा (Interprocess communication, Network structure and Security): UNIX में processes के आपस में communicate करने हेतु synchronization तथा सिगनल्स उपलब्ध होते हैं जिससे process अन्य process में होने वाली events (घटनाओं) की जानकारी प्राप्त करते हैं व आपस में डेटा (data) का आदान-प्रदान करते हैं।

Linux Kernel की Networking Software की तीन परतों (layers) द्वारा implement की जाती है—Socket Interface, Protocol drivers तथा Network device drivers. Linux networking सिस्टम की सबसे महत्वपूर्ण प्रोटोकॉल IP (Internet Protocol) है। Linux की security mechanism को दो groups में classify किया जाता है—(i) Authenciation जिसके द्वारा यह सुनिश्चित किया जाता है कि कोई भी अनधिकृत व्यक्ति system को access न कर सके तथा Access control, यह सुनिश्चित करने के लिये कि user के पास किस object को access करने का अधिकार है तथा किसको नहीं।

Inter Process Communication (IPC)

A process can be of two type:

- *Independent process.*
- *Co-operating process.*

An independent process is not affected by the execution of other processes while a co-operating process can be affected by other executing processes. Inter process communication (IPC) is a mechanism which allows processes to communicate each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other using these two ways:

- *Shared Memory*
- *Message passing*

The figure below shows a basic structure of communication between processes via shared memory method and via message passing.

An operating system can implement both method of communication. Communication between processes using shared memory requires processes to share some variable and it completely depends on how programmer will implement it. Suppose process1 and process 2 are executing simultaneously and they share some resources or use some information from other process, process1 generate information about certain

computations or resources being used and keeps it as a record in shared memory. When process 2 need to use the shared information, it will check in the record stored in shared memory and take note of the information generated by process1 and act accordingly. Processes can use shared memory for extracting information as a record from other process as well as for delivering any specific information to other process.

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

प्रश्न 13—Process adress space क्या है? Virtual व Physical address क्या होते हैं?

उत्तर—*The process address space is the set of logical addresses that a process references in its code. The operating system takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program. There are three types of addresses used in a program before and after memory is allocated—*

Memory Addresses	*Description*
Symbolic addresses	*The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.*
Relative addresses	*At the time of compilation, a compiler converts symbolic addresses into relative addresses.*
Physical addresses	*The loader generates these addresses at the time when a program is loaded into main memory.*

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

The set of all logical addresses generated by a program is referred to as a logical address space. The set of all physical addresses corresponding to these logical addresses is referred to as a physical address space.

The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device.

प्रश्न 14—Static व Dynamic linking क्या होती है?

उत्तर—Dynamic loading में routine तब तक load नहीं होती है, जब तक कि उसे call नहीं किया जाता। सभी routines को डिस्क में relocatable load format में रखा जाता है। Main program को memory में load करके execute किया जाता है। जब कोई routine किसी अन्य routine को call करती है, तो calling routine पहले यह check करती है कि यह दूसरी routine load हुई है या नहीं। यदि नहीं, तो relocatable linking register की सहायता से वांछित subroutine को memory में load किया जाता है तथा इस change को दर्शाने हेतु program की address tables (पता तालिकाओं) को update किया जाता है। अब कन्ट्रोल इस नयी loaded routine को pass कर दिया जाता है।

डायनैमिक लोडिंग का मुख्य लाभ यह है कि unused routine (अर्थात अप्रयुक्त रूटीन) को load करने की आवश्यकता नहीं होती। विशेष तौर पर उन स्थितियों में जहाँ infrequently होने वाली घटनाओं (अर्थात कभी-कभार होने वाली घटनाओं, या ऐसी घटनाओं जिनके होने के chance बहुत कम होते हैं) हेतु बहुत बड़े codes आवश्यक होते हैं (जैसे कि error routine

में dynamic loading काफी उपयोगी सिद्ध होती है) क्योंकि इन स्थितियों में total program size बहुत बड़ा होने के बावजूद use किया जाने वाला अंश (portion that is used and loaded) काफी कम होता है।

यहाँ एक बात उल्लेखनीय है कि dynamic loading हेतु operating system से किसी special support की आवश्यकता नहीं होती। यह user की जिम्मेदारी है कि वह program को इस प्रकार से design करे जिससे dynamic loading का लाभ उठाया जा सके। हालांकि, operating systems library routines की सहायता प्रदान करके user को dynamic loading implement करने हेतु सहायता प्रदान कर सकते हैं।

Static vs Dynamic Loading

If you have to load your program statically, then at the time of compilation, the complete programs will be compiled and linked without leaving any external program or module dependency. The linker combines the object program with other necessary object modules into an absolute program, which also includes logical addresses.

If you are writing a dynamically loaded program, then your compiler will compile the program and for all the modules which you want to include dynamically, only references will be provided and rest of the work will be done at the time of execution.

At the time of loading, with static loading, the absolute program and data is loaded into memory in order for execution to start.

In dynamic loading, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program.

प्रश्न 15—Static व Dynamic Linking क्या है?

उत्तर—Static vs Dynamic Loading—When static linking is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency.

When dynamic linking is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking. Dynamic Link Libraries (DLL) in Windows and Shared Objects in Unix are good examples of dynamic libraries.

प्रश्न 16—Scheduling Criteria से आप कया समझते है?

उत्तर—जब भी CPU idle (खाली, कोई काम न होना) हो जाता है तो operating system ready queue (पंक्ति) में खड़े processes (जो कि execution के लिए CPU प्राप्त होने की प्रतीक्षा में हैं) में से किसी process को select करके उसको CPU आवंटित करता है। यह selection process (चयन प्रक्रिया) CPU scheduler (या short term scheduler) द्वारा सम्पन्न की जाती है। शैडयूलर का कार्य होता है— Memory में से execute होने के लिये तैयार processes में से एक को select करना तथा उसे CPU allocate (आवंटित) करना।

विभिन्न शैडयूलिंग algorithms के भिन्न अभिलक्षण होते हैं तथा किसी शैडयूलिंग algorithm का चुनाव कुछ processes के पक्ष में हो सकता है (दूसरे processes की तुलना में)। अत:, भिन्न स्थितियों में भिन्न algorithm को choose किया जाता है।

- **CPU Utilization : Utilization का अर्थ है, किसी वस्तु का बेहतर तरीके से उपयोग। CPU का utilization तभी सम्भव है जब CPU को अधिक समय के लिये busy (व्यस्त) रखा जाये। CPU utilization की थ्यौरेटिकल रेंज 0 से 100% हो सकती है। सामान्यत: (real systems में) इसकी रेंज lightly loaded systems (हल्के भार वाले system) के लिये 40% से लेकर heavily loaded systems (अधिक भार वाले सिस्टम्स) के लिए 90% तक हो सकती है।**

Throughput : एक निश्चित समय में कम्प्यूटर द्वारा किया जाने वाला कार्य throughput कहलाता है। जब CPU processes को execute करता है, तो यह कार्य सम्पन्न होते हैं। अत:, per unit time में सम्पन्न कार्यों की संख्या को throughput कहा जाता है।

Turnaround Time—

किसी process को complete होने में कितना समय लगता है, इसको turnaround time द्वारा व्यक्त किया जाता है। अर्थात् process के submission (प्रस्तुत करने) के बाद उसके completion (पूर्ण होने) में लगने वाला समय turnaround time कहलाता है।

Waiting Time—

CPU scheduling algorithm न केवल process के execution तथा I/O में लगने वाले समय को प्रभावित करती है बल्कि वह process द्वारा ready queue में इंतज़ार में लगने वाले समय को भी प्रभावित करती हैं। Ready queue में इंतज़ार करने में लगने वाली अवधियों (समयों) का योग waiting time कहलाता है।

Response Time—

कई बार process से यह अपेक्षा होती है कि उसके कुछ results जल्दी प्राप्त हो जाये तथा शेष results बाद में compute होते रहें तथा जल्दी प्राप्त होने वाले results user को उपलब्ध कराये जा सकें। अतः, ऐसी स्थितियों में turnaround time के बजाय response time criteria महत्वपूर्ण हो जाता है जो कि request की submission के पश्चात् पहली response (प्रतिक्रिया या output) प्राप्त होने के मध्य के समय को व्यक्त करता है।

"Request के submission के पश्चात् पहली response प्राप्त होने में लगने वाला समय response time कहलाता है। अतः, response time request submission के पश्चात् लगने वाला वह समय है जिसके बाद पहली response प्राप्त होती है न कि final output।"

प्रश्न 17—Preemptive तथा Non-preemptive scheduling क्या होती है?

उत्तर—

Non-preemptive Scheduling	Preemptive Scheduling
• Non-preemptive scheduling में process को CPU देने के पश्चात् उसे बीच में interrupt करके CPU वापस नहीं लिया जा सकता (Running task is executed till completion)।	• Preemptive scheduling में process से कार्य के बीच में ही CPU वापस लिया जा सकता है (Running task is interrupted for sometime and can be resumed later when priority task has finished its execution)।
• छोटे jobs को बड़े jobs के पूरा होने का अधिक समय तक इंतज़ार करना पड़ सकता है।	• इसमें छोटा job आने पर उसको CPU उपलब्ध कराया जा सकता है।
• Response time अधिक predictable होता है।	• Response time अपेक्षाकृत कम predictive होता है।
• Scheduler दो स्थितियों में CPU वापस लेता है—	• यदि process running से ready state या waiting से ready state में switch करता है तो उसे CPU preempt कराया जा सकता है।
• Process running state से waiting state में जाये।	
• Process terminate हो जाये।	

प्रश्न 18—FCFS Scheduling क्या होती है?

उत्तर—फर्स्ट-कम, फर्स्ट सर्व्ड शैडयूलिंग (First Come-First Served Scheduling or FCFS scheduling) : First come first served का अर्थ होता है—पहले आओ, पहले पाओ। यह सबसे सरल शैडयूलिंग एल्गोरिद्म है। **FCFS algorithm के अनुसार सबसे पहले request करने वाले process को सबसे पहले CPU allocate किया जाता है, उसके अगली request को उसके बाद तथा यही क्रम चलता रहा है।**

महत्वपूर्ण बिन्दु

- Jobs are executed on first come, first serve basis. (Jobs "पहले-आओ पहले पाओ" के आधार पर execute किये जाते हैं)

- Easy to understand and implement (समझने व implement करने में आसान)
- Poor in performance as average wait time is high (अधिक average wait time)

प्रश्न 19—SJF Scheduling क्या होती है?

उत्तर—Shortest-job-first अर्थात् सबसे छोटा-सबसे पहले। **SJF algorithm में किसी process को CPU allot करने से पहले यह देखा जाता है कि उस process की अगली CPU burst कितनी अवधि की है तथा जिस process की CPU burst सबसे छोटी होती है उसको CPU allocate किया जाता है अर्थात् CPU burst समय की अवधि के बढ़ते हुये क्रम के अनुसार बारी-बारी से processes को CPU का allocation किया जाता है।**

Shortest Jobs-First (SJF) Scheduling :

- Best approach to minimize waiting time (Waiting time को न्यूनतम करने हेतु best approach)
- Actual time taken by the process should known to processor in advance (Process द्वारा लिया जाने वाला समय प्रौसेसर को पहले से पता होना चाहिये)
- Impossible to implement.

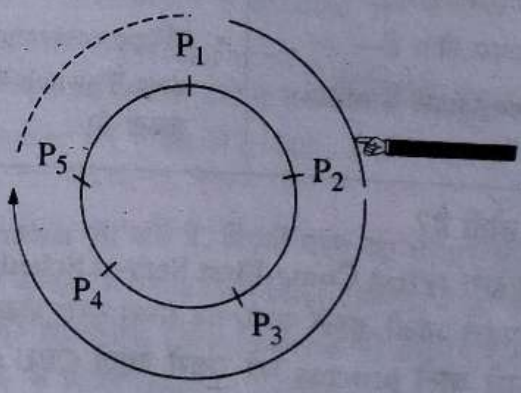
प्रश्न 20—Priority Scheduling क्या होती है? RR Scheduling क्या होता है?

उत्तर—Priority का अर्थ होता है प्राथमिकता, अर्थात् **यदि शैडयूलिंग प्राथमिकता के आधार पर की जायें तो इसे priority scheduling कहा जाता है। इस प्रकार की scheduling में प्रत्येक process के साथ एक priority attach कर दी जाती है तथा उच्चतम प्राथमिकता (highest priority) वाले process को CPU allocate किया जाता है। यदि दो processes की priority same है तो FCFS (first come first served) क्रम में allocation किया जाता है।**

- Priority is assigned for each process (प्रत्येक process से प्राथमिकता सम्बद्ध की जाती है)
- Process with highest priority is executed first and so on (सर्वोच्च प्राथमिकता वाला process सबसे पहले execute किया जाता है तथा यह क्रम चलता रहता है)
- Processes with same priority are executed in FCFS manner (समान प्राथमिकता वाले processes का निर्णय First Come-First Served (FCFS) के आधार पर लिया जाता है)
- Priority can be decided based on memory requirements, time requirements or any other resource requirement (internally) or funds, importance, sponsor (externally).

राउन्ड रॉबिन शैडयूलिंग (Round Robin Scheduling Algorithm)—

राउन्ड रॉबिन शैडयूलिंग एल्गोरिद्म विशेषतया time sharing systems हेतु design की गई है। यह FCFS के समान है किन्तु इसमें processes के मध्य preemption की व्यवस्था की जाती है। समय की एक छोटी इकाई, जिसको time quantum (या time slice) कहा जाता है, परिभाषित की गई है जिसका मान लगभग 10 से 100 मिलीसैकेन्ड रखा जाता है। Ready queue के एक circular queue के रूप में treat किया जाता है। CPU scheduler queue के सभी सदस्यों को बारी-बारी से अधिकतम 1 quantum का समय allocate करता है। यदि process इतने समय में complete नहीं



चित्र—Round Robin Scheduling

सबको अधिकतम 1 quantum समय दिया जाता है, यदि उतने समय मे कार्य पूर्ण नहीं हो पाता तो उस process को preempt करके CPU queue के अगले सदस्य को allocate कर दिया जाता है।

हो पाता तो उस process को preempt करके queue में next member को CPU allocate कर दिया जाता है। यह process तब तक चलता रहता है तब तक process complete नहीं हो जाता।

- A fixed time is allotted to each process, called quantum, for execution (प्रत्येक process को processing हेतु एक निश्चित time quantum allot किया जाता है)
- Once a process is executed for given time period that process is preempted and other process executes for given time period (Time पूरा होने पर उससे CPU वापस लेकर next process को दिया जाता है तथा यह प्रक्रिया एक circular queue के रूप में चलती रहती है)।
- Context switching is used to save states of preempted processes (Preempted process के state save करने हेतु context switch का use किया जाता है)

प्रश्न 21—Deadlock क्या होते हैं? इनकी शर्तें, detection, recovery के बारे मे बताइये।

उत्तर—"A set of processes is in a deadlock state, if every process in the set is waiting for an event that can only be caused by some other process in the same set."

Deadlock is when two or more tasks never make progress(blocked process) because each is waiting for some resource held by another process. यानि एक ऐसी स्थिति जहां दो (या अधिक) process कभी आगे नहीं बढ़ पाते क्योंकि प्रत्येक किसी दूसरे द्वारा hold किये गये resource के free होने का इंतज़ार कर रहा होता है।

Deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

Processes का कोई set deadlock अवस्था में तब कहा जाता है जब set का प्रत्येक process एक ऐसे event के होने के लिये इंतज़ार में खड़ा है जो कि उस set के किसी दूसरे process द्वारा ही किया जाना संभव है (यहाँ event का तात्पर्य resource को अधिग्रहीत करने (aquisition) तथा मुक्त करने (release) से है)। Resource physical हो सकते हैं (उदाहरणत:, printers, tape drives, memory space तथा CPU cycles) या logical हो सकते हैं (उदाहरणत: files, semaphores तथा monitors)।

डैडलॉक हेतु आवश्यक शर्तें (Necessary Conditions for Deadlock) :

यदि किसी सिस्टम में नीचे दी गई चार शर्तें एक साथ पूरी हो जाती हैं तो डेडलॉक की स्थिति उत्पन्न हो सकती है (Coffman's condition for deadlock)—

(i) Mutual Exclusion—कम से कम एक resource non-shareable mode में hold होना चाहिये अर्थात् इस resource का एक समय में केवल एक process द्वारा ही use किया जाना सम्भव हो। अब यदि दूसरा process इस resource की request करता है तो उसको इस resource के release (मुक्त) होने तक प्रतीक्षा करनी होगी।

(ii) Hold and Wait—Process द्वारा कम से कम किसी एक resource को hold करके रखा गया हो तथा कुछ अतिरिक्त resources के अधिग्रहण (acquisition) हेतु प्रतीक्षारत हो जो कि इस समय दूसरे processes द्वारा hold किये गये हो।

(iii) No Preemption—Resources को preempt करना सम्भव न हो अर्थात् process द्वारा hold किये गये resource को तभी स्वत: मुक्त करना संभव हो जब process द्वारा task (कार्य) पूरा कर लिया जाये अर्थात् कार्य पूर्ण होने से पहले process resource को स्वत: मुक्त करने की स्थिति में नहीं हो।

(iv) Circular Wait—Circular wait की स्थिति हो अर्थात् waiting processes का एक set (समूह) (P_0, P_1, P_2 ..., P_{n-1}, P_n) हो जिसमें P_0, P_1 द्वारा hold किये गये रिसोर्स को मुक्त किये जाने का इंतज़ार करता हो, P_1, P_2 द्वारा hold किये गये resource को मुक्त किये जाने का इंतज़ार करता है, P_{n-1}; P_n द्वारा hold किये गये resource के मुक्त किये जाने का इंतज़ार करता है तथा P_n, P_0 द्वारा hold किये गये resources के मुक्त किये जाने का इंतज़ार करता हो। साधारण शब्दों में समझा जाये तो हम कह सकते हैं कि तीन process A, B, व C हैं जिसमें A को B के द्वारा hold resource के free होने का इंतज़ार है तभी उसका कार्य पूरा होगा, B को C के द्वारा hold resource के free होने का इंतज़ार है तभी उसका कार्य

पूरा होगा तथा *C* को *A* द्वारा hold resource के free होने का इंतज़ार है तभी उसका कार्य पूरा होगा। ऐसी स्थिति को circular wait स्थिति कहा जाता है।

Mutual Exclusion—Non-shareable resources (अर्थात् वह resources जो share न किये जा सकें) हेतु mutual exclusion की शर्त लागू होती है। उदाहरणत: कोई प्रिंटर एक साथ कई processes द्वारा share नहीं किया जा सकता। Shareable resources (अर्थात् वह resources जिनको share किया जा सकता है) पर mutual exclusion की शर्त लागू नहीं होती, इसलिये यह deadlock में involve (शामिल) नहीं हो सकते। Shareable resources का उदाहरण है—read only files, तथा यदि एक से अधिक process किसी read only file पर पहुँच बनाने का प्रयास करते हैं, तो उनको एक साथ उस file को access (पहुँच बनाना) प्रदान की जा सकती है। किसी shareable resource के लिये process को कभी प्रतीक्षा करने की आवश्यकता नहीं पड़ती। किन्तु mutual exclusion condition को समाप्त नहीं किया जा सकता क्योंकि कई resources unshareable होते हैं।

Hold and Wait—Hold and wait का तात्पर्य है कि process ने एक resource hold कर रखा है तथा दूसरे को प्राप्त करने हेतु उसने request send की हुई है तथा request पूरी होने का इंतज़ार कर रहा है। अत: यह सुनिश्चित करने के लिये कि सिस्टम में hold and wait स्थिति उत्पन्न न हो, इस बात को सुनिश्चित करना चाहिये कि जब कोई process किसी resource हेतु request करे, उस समय उसके द्वारा कोई दूसरा resource hold न किया गया हो। एक protocol यह use की जा सकती है कि process द्वारा प्रारम्भ होते समय ही सभी resource request कर लिये जायें तथा process को वह resources allocate कर दिये जायें। अत:, इसको implement करने हेतु resource request करने हेतु system calls अन्य system calls से पहले प्राप्त हो जानी चाहिये। एक वैकल्पिक protocol यह use की जा सकती है कि कोई process तभी किसी resource हेतु request भेज सके जबकि वह कोई अन्य resource hold न कर रहा हो। Process कुछ resources की request करे और उनको use करे तथा अब यदि उसे कोई नयी resource requert भेजनी हो तो उससे पहले वह allocated सभी resources को release कर दें तथा उनको release करने के पश्चात् ही उसे नयी request भेजने का अधिकार दिया जाये।

No Preemption—

Non preemption का अर्थ है कि यदि process को CPU allocate कर दिया जाता है तो process उस CPU को तब तक अपने पास रखता है जब तक कि process terminate न हो जाये या फिर waiting state में न चला जाये। Deadlock की तीसरी आवश्यक शर्त यही है कि यदि allot किये गये resources का preemption सम्भव न हो, तो deadlock उत्पन्न हो सकता है। अत: deadlock से बचने के लिये यह protocol निर्धारित की जा सकती है कि यदि process ने किसी resource को hold करके रखा हो तथा वह किसी दूसरे resource की request करे जिसे कि तत्काल (immediately) allocate करना संभव न हो (अर्थात् process को wait करना पड़े) तो उस process द्वारा hold किये गये मौजूदा resources को भी preempt कर दिया जाये (अर्थात् वापस ले लिया जाये) तथा यह preempted resources भी process की request list (list of resources for which the process is waiting अर्थात् resources की list जिसके लिये process wait कर रहा हो) में add कर दिये जायें। अत: अब process restart (फिर से start) तभी हो सकेगा जब अपने पुराने resources (जो उसके पास पहले से थे तथा जिनको preempt किया गया) तथा नये resources (जिसकी उसने request भेजी थी) एक साथ प्राप्त करेगा।

डैडलॉक डिटैक्शन (Deadlock Detection) :

यदि सिस्टम deadlock prevention या deadlock detection avoidance algorithm use नहीं करता, तो deadlock स्थिति उत्पन्न हो सकती है। ऐसी स्थिति में सिस्टम के पास एक ऐसी algorithm होनी चाहिये जो system के स्टेट का निरीक्षण (examination) करके यह चैक करे कि कहीं डैडलॉक तो उत्पन्न नहीं हो गया, और यदि हाँ, तो एक ऐसी algorithm जो डैडलॉक से रिकवरी प्रदान कर सके।

यदि resource type के कई instances हो (deadlock delectation algorithm for multiple instances of a resource type): यदि प्रत्येक resource type के कई instances उपलब्ध हैं, तो wait-for-graph स्कीम लागू नहीं हो

पाता। ऐसी स्थिति में प्रयुक्त deadlock detection algorithm कई time varying data structures का use करती है (Banker's algorithm की भांति)

डैडलॉक रिकवरी (Deadlock Recovery) :

यदि detection algorithm से यह पता चलता है कि सिस्टम deadlock अवस्था में है तो उससे बाहर आने हेतु कई विकल्प होते हैं जैसे कि operator को सूचित किया जाये तथा वह manually इससे deal करे, या फिर circular wait break की जाये, या कुछ resources को preempt कर दिया जाये।

- Process termination अर्थात् प्रौसेस को abort (लक्ष्य पूरा होने से पहले समाप्त करना) कर देना, अत: सभी deadlocked processes को abort करके या फिर एक-एक करके processes को abort करके, जब तक कि cycle eliminate नहीं हो जाती, Deadlock से recovery प्राप्त की जा सकती है।
- Preemption को use करके deadlock से recovery प्राप्त करने हेतु victim का selection roll back (preempted process को फिर से कब शुरू करना है) starvation न हो, इत्यादि का ध्यान रखा जाना चाहिये।

प्रश्न 22—Memory management क्या होता है? Logical and Physical space क्या होती है? Swepping क्या होती है? Internal Fragmentation क्या होती है?

उत्तर—मैमोरी प्रबन्धन—इसका तात्पर्य प्राइमरी मेमोरी (प्राथमिक मेमोरी) तथा मुख्य मेमोरी (main memory) के प्रबन्धन से है। मेन मेमोरी बाइट्स या वर्ड्स का एक बड़ा व्यूह होता है (large array of words or bytes), जिसमें से प्रत्येक बाइट का या वर्ड का भिन्न address होता है।

Main मेमोरी से तीव्र स्टोरेज (fast storage) प्राप्त हो जाती है जिस पर कि CPU द्वारा सीधे पहुँच (access) बनाई जा सकती है।

मेमोरी प्रबन्धन हेतु ऑपरेटिंग सिस्टम निम्न कार्य करता है—

- प्राइमरी मेमोरी को ट्रैक करता है अर्थात् यह ध्यान रखता है कि उसके कौन से भाग का कौन प्रयोग कर रहा है एवं कौन-कौन से भाग प्रयोग में नहीं है।
- मल्टी प्रोग्रामिंग में ऑपरेटिंग सिस्टम यह निर्धारित करता है कि कौन सा प्रौसेस मेमोरी प्राप्त करेगा एवं कब और कितनी मेमोरी प्राप्त करेगा।
- जब कोई प्रौसेस मेमोरी request करता है तो ऑपरेटिंग सिस्टम मेमोरी का आंवटन करता है।
- जब प्रौसेस को मेमोरी की आवश्यकता नहीं होती या प्रौसेस समाप्त हो जाता है तो ऑपरेटिंग सिस्टम मेमोरी को deallocate (आवंटन को निरस्त करना या वापिस प्राप्त करना) करता है।

Memory management is the functionality of an operating system which handles or manages primary memory. It keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed and updates the status accordingly.

Program द्वारा generated logical address का set logical address space कहलाता है तथा इन logical address के corresponding सभी physical address का set physical address space कहलाता है।

स्वैपिंग—Swapping वह तकनीक है जिसमें किसी process को temporarily (थोड़े समय के लिये) memory से backing store को swap किया जाता है तथा बाद में execution हेतु पुन: memory में लाया जाता है।

बैकिंग स्टोर सामान्यत: एक हार्ड डिस्क ड्राइव या कोई अन्य सैकेन्ड्री स्टोरेज़ होती है। इसका access fast होना चाहिये तथा इसमें पर्याप्त space (स्थान) होना चाहिये ताकि सभी users हेतु सभी memory images की copies को स्थान उपलब्ध कराया जा सके इसमें इन सभी memory images हेतु direct access उपलब्ध कराने की क्षमता होनी चाहिये।

"First fit व best fit memory allocation strategies के कारण memory में छोटे-छोटे अप्रयुक्त टुकड़ो का उत्पन्न हो जाना external fragmentation कहलाता है।"

Internal Fragmentation vs External Fragmentation—

1. **External Fragmentation**—*Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so cannot be used.*
2. **Internal Fragmentation**—*Memory block assigned to process is bigger. Some portion of memory is left unused as cannot be used by another process.*

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

Programe द्वारा generated logical address का set logical address space कहलाता है तथा इन logical address के corresponding सभी physical address का set physical address space कहलाता है।

प्रश्न 23—Paging क्या होती है? Page allocation क्या होता है?

उत्तर—*Paging is a memory-management scheme that allows the physical address space of a process to be noncontiguous. It avoids the problem of fitting memory chunks of varying sizes onto the backing store.*

पेजिंग वह तकनीक है जिसमें physical memory को same sizes के blocks में बाँट दिया जाता है तथा यह blocks pages कहलाते हैं। पेज का size 2 की power (घात) में होता है, (512 bytes से 8192 bytes तक) जब कोई process execute होना होता है तो उससे संबंधित pages को उपलब्ध memory frames में load किया जाता है। Paging तकनीक से external fragmentation की समस्या उत्पन्न नहीं होती।

Process का logical space non-contiguous हो सकता है तथा free मैमोरी फ्रेम की उपलब्धता होने पर process को physical memory allocate की जा सकती है। Operating system सभी free frames को track (जानकारी रखना) करता रहता है। किसी *n*-page size के प्रोग्राम को run करने हेतु operating system को *n* free frames की आवश्यकता होती है।

CPU द्वारा generate किया गया address दो पार्ट्स में विभाजित होती है—(i) Page number (P) जिसको कि page table में index के रूप में use किया जाता है (जिसमें physical memory के प्रत्येक पेज का base address होता है) तथा (ii) Page offset (d) जिसको कि base address से combine करके physical memory address define किया जाता है।

प्रश्न 24—Segmentation क्या होती है?

उत्तर—सैगमैंटेशन—सैगमैंटेशन वह तकनीक है जिसकी सहायता से मैमोरी को logical भागों में विभाजित किया जाता है तथा प्रत्येक भाग संबंधित सूचना के समूह को व्यक्त करता है अर्थात् segmentation is a technique to break memory into logical pieces or parts where each piece represents a group of related information. उदाहरणत: प्रत्येक process की code segment या data segment, operating system की code segment इत्यादि। Segmentation को paging का use करने या paging का use किये बिना, अर्थात् दोनों ही तरीके से implement किया जा सकता है।

Segments के साइज़ अलग-अलग हो सकते हैं तथा इस प्रकार इसमें internal fragmentation नहीं होता। External fragmentation की संभावना रहती है किन्तु यह बहुत कम होता है।

CPU द्वारा generate किया गया address दो भागों में विभाजित होता है—(i) Segment number(s), जिसको कि segment table में index के रूप में use किया जाता है (जिसमें physical memory की प्रत्येक segment का base address होता है तथा segment की limit होती है) तथा (ii) Segment offset जिसको कि लिमिट के विरुद्ध चैक किया जाता है तथा limit से कम पाये जाने पर बेस address से combine करके physical memory address define किया जाता है।

प्रश्न 25—Compaction क्या होता है?

उत्तर—Compaction वह Process है जिसके द्वारा memory की free space को एकत्रित करके एक बड़ी memory space तैयार की जा सकती है जिससे processes को store किया जा सके अर्थात् सभी खाली space को एक साथ करके एक बड़ी memory space तैयार करना compaction कहलाता है।

प्रश्न 26—Dedicated device, Shared device, IO device, Storage device, Buffering व Spooling के विषय में बताइये।

उत्तर—Dedicated device वह होता है जो कि किसी विशेष procedure algorithms को करने के लिए बना होता है।

Shared device वह होते हैं जो कि कई process के मध्य share किये जा सकते हैं तथा एक समय में एक से अधिक process को assign किये जा सकते हैं जैसे कि direct access storage devices (Magnetic disc, optical disc etc)।

IO device वह होते हैं जिनकी सहायता से computer outside world से interact करता है अर्थात् उनसे input लेता है तथा उनके द्वारा output प्रदान करता है जैसे—Keyboard, Mouse, Touch pad, Disc drives, Video controllers, Audio controllers, Networking parts, LED's, BitMapped screens, ADC's, ON-off switches, Printers, Scanners, Audio I/O's, USB device etc.

स्टोरेज डिवाइस—Storage devices कोई भी कम्प्यूटिंग हार्डवेयर होता है जो डेटा फाइलों और ऑब्जेक्ट्स को स्टोर करने, पोर्ट करने और निकालने के लिए उपयोग किया जाता है। यह अस्थायी और स्थायी रूप से जानकारी को Hold और संग्रहीत कर सकता है, और कम्प्यूटर, सर्वर या किसी भी समान कम्प्यूटिंग डिवाइस में आंतरिक या बाहरी हो सकता है।

Buffering तथा spooling वह विधियाँ हैं जिनके द्वारा Input output sub-systems main memory तथा disc की memory space को use करके computer की performance तथा efficiency में सुधार ला सकते हैं।

Buffering—Computer की main memory में एक विशेष area होता है जो Buffer कहलाता है, जिसमें उस data को temporarily store व hold किया जाता है जोकि दो devices के बीच में या किसी device या application के बीच में transmit हो रहा हो। अत: data को buffer में temporarily store करने की प्रक्रिया Buffering कहलाती है। Buffering द्वारा sender और receiver के बीच में data transfer की speed को match करने में सहायता मिलती है। यदि sender की transmission speed receiver से कम है तो main memory में एक buffer बना दिया जाता है जो कि sender से receive हुयी bytes को (तथा इसके विपरीत) एकत्रित करता रहता है।

Spooling—Spooling का full form simultaneous peripheral operations online है, Spool का कार्य buffer के समान ही होता है तथा यह भी device के लिए job को तब तक hold करके रखता है जब तक कि device job को accept करने के लिए तैयार नहीं हो जाता है।

***Dedicated Devices**: A dedicated device is something that is built or programmed to do only a specific procedure, or algorithm. These are devices that are assigned to one process at a time, and the process only releases the device once it is completed. So a printer is a dedicated device, but using the spooling means it can be shared.*

***Shared Devices**: Shared devices are the devices that can be shared among several processes. They can be assigned to more than one process at a time. These include direct access storage devices such as magnetic disks and optical discs.These include direct access storage devices such as magnetic disks and optical discs.*

***I/O Devices**: An input/output device is any hardware used by a human operator or other systems to communicate with a computer. As the name suggests, input/output devices are capable of sending data (output) to a computer and receiving data from a computer (input).*

***Storage Devices**: A storage device is any computing hardware that is used for storing, porting and extracting data files and objects. It can hold and store information both temporarily and permanently, and can be internal or external to a computer, server or any similar computing device.*

***Buffering and Spooling**: The buffer is an area in the main memory that is used to store or hold the data temporarily that is being transmitted either between two devices or between a device or an application. Buffer temporarily stores data that is being transmitted from one place to another.*

Spooling and buffering are the two ways by which I/O subsystems improve the performance and efficiency of the computer by using a storage space in main memory or on the disk. The basic difference between

Spooling and Buffering is that Spooling overlaps the I/O of one job with the execution of another job while the buffering overlaps I/O of one job with the execution of the same job.

	SPOOLING	BUFFERING
Purpose	Spooling overlap the I/O of one job with the computation of another job.	Buffer overlaps the I/O of one job with the computation of the same job.
Full form	Simultaneous peripheral operation online.	No full form.
Efficiency	Spooling is more efficient than buffering.	Buffering is less efficeint than spooling.
Size	Spooling considers disk as a huge spool or buffer.	Buffer is a limited area in main memory.

प्रश्न 27—File system कितने प्रकार का होता है? Simple file system, Basic file system, Logical file system व Physical file system क्या होता है?

उत्तर—A file is a named collection of related information on a secondary storage अर्थात् सम्बंधित सूचनाओं के समूह को एक नाम देकर उसे store कर देना ही फाइल कहलाता है।

File type means to the ability of the operating system to distinguish different types of file such as text files, source files, binary files etc. Operating system like MS-DOS and UNIX have the following types of files:

Ordinary Files (साधारण फाइल्स)

- These are the files that contain user information (यूज़र इनफॉरमेशन).
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the file.

Directory Files (डायरैक्ट्री फाइल्स)

- These files contain list of file names and other information related to these files.(फाइल्स के नामों की सूची)

Special Files (विशेष फाइल्स or Device फाइल्स)

- These represent physical device like disks, terminals, printers, networks, tape drive etc. They may be of following types:
- **Character Special Files**—In these files, data is handled character by character as in case of terminals or printers.
- **Block Special Files**—In these files, data is handled in blocks as in the case of disks and tapes.

CPU द्वारा जनरेट किये गये एड्रैस को लॉजिकल एड्रैस (तार्किक पता) कहा जाता है जबकि memory द्वारा देखे जाने वाले address (अर्थात् वह adderss जिनको memory के memory address register में लोड किया गया है), physical address (भौतिक पता) कहा जाता है।

कम्पाइल टाइम तथा लोड टाइम एड्रैस बाइंडिंग विधियाँ समान लॉजिकल व फिज़िकल एड्रैस उत्पन्न करती हैं जबकि executing time binding विधि में भिन्न-भिन्न लॉजिकल व फिजिकल एड्रैस उत्पन्न होते हैं। ऐसी स्थिति में logical address को virtual address कहा जा सकता है। किसी प्रोग्राम द्वारा उत्पन्न किये गये समस्त logical addresses का समूह logical address space कहलाता है जबकि किसी प्रोग्राम द्वारा उत्पन्न किये गये समस्त physical addresses का समूह physical address space कहलाता है। अतः, execution time binding विधि में logical address space व physical address

space भिन्न होते हैं जबकि compile time binding scheme व load time binding scheme में यह identical (एक समान) होते हैं।

Physical file contains the actual data that is stored in series system. The Physical file contains only one record format. Records in database files either a field level description or record level description.

Logical file does not contain data. They contain a description of records that are found in one or more physical files. The Logical file also contain more than one record format.

Physical file

- *Occupies the portion of memory. It contains the original data.*
- *A physical file contains one record format.*
- *Can exist even without Logical file.*
- *If there is a logical file for a PF, the PF can't be deleted until and unless we delete the Logical file.*
- *CRTPF command is used to create such object.*

Logical file

- *Does not contain any data. It loads itself at run time as per the defined access path.*
- *Does not occupy any memory space.*
- *Logical file can contain up to 32 record formats.*
- *Can't exist without PF.*
- *If there is a logical file for a physical file, the Logical file can be deleted without deleting the PF.*
- *CRTLF command is used to create such type object.*

The logical file system is the level of the file system at which users can request file operations by system call. This level of the file system provides the kernel with a consistent view of what might be multiple physical file systems and multiple file system implementations. As far as the logical file system is concerned, file system types, whether local, remote, or strictly logical, and regardless of implementation, are indistinguishable.

A consistent view of file system implementations is made possible by the virtual file system abstraction. This abstraction specifies the set of file system operations that an implementation must include in order to carry out logical file system requests. Physical file systems can differ in how they implement these predefined operations, but they must present a uniform interface to the logical file system. Each set of predefined operations implemented constitutes a virtual file system. As such, a single physical file system can appear to the logical file system as one or more separate virtual file systems.

Virtual file system operations are available at the logical file system level through the virtual file system switch. This array contains one entry for each virtual file system, with each entry holding entry point addresses for separate operations. Each file system type has a set of entries in the virtual file system switch.

The logical file system and the virtual file system switch support other operating system file-system access semantics. This does not mean that only other operating system file systems can be supported. It does mean, however, that a file system implementation must be designed to fit into the logical file system model. Operations or information requested from a file system implementation need be performed only to the extent possible.

Logical file system can clso refer to the tree of known path names in force while the system is running. A virtual file system that is mounted onto the logical file system tree itself becomes part of that tree. In fact, a single virtual file system can be mounted onto the logical file system tree at multiple points, so that nodes in the virtual subtree have multiple names. Multiple mount points allow maximum flexibility when constructing the logical file system view.

प्रश्न 28—Linux तथा Unix क्या है? इनकी संरचना बताइये। किन्हीं 10 linux commands एवं filter के बारे में बताइये।

उत्तर—यूनिक्स एक कम्प्यूटर Operating System (प्रचालन तंत्र) है। यह मूल रूप से 1969 में Bell Laboratories में विकसित किया गया था। इसके विकास में AT and T के केंन थोम्प्सन, डेनिस रिची, ब्रियन केर्निघन, दोग्ल्स मेक्लेरी और जो ओसाना आदि शामिल थे। यूनिक्स (Unix) एक एल्टीयूजर व मल्टी टास्किंग ऑपरेटिंग सिस्टम है यह वर्कस्टेशंस और सर्वर में मास्टर कंट्रोल प्रोग्राम के तौर पर व्यापक ढंग से प्रयुक्त होता है।

- यूनिक्स (Unix) एक मल्टीयूजर व मल्टी टास्किंग ऑपरेटिंग सिस्टम है।
- वर्कस्टेशंस और सर्वर में मास्टर कंट्रोल प्रोग्राम के तौर पर व्यापक ढंग से प्रयुक्त होता है।
- C लैंग्वेज में लिखा जाता है।
- यूनिक्स सर्वर्स और इन्टरनेट के लिए प्रयुक्त ऑपरेटिंग सिस्टम है।
- यूनिक्स ऑपरेटिंग सिस्टम बहुत शक्तिशाली है।
- इसे अन्य ऑपरेटिंग सिस्टम के मुकाबले इंस्टाल और सेटअप करना कठिन है, लेकिन यह कम्प्यूटर के रिसोर्स और पॉवर पर उच्च नियंत्रण प्रदान करता है।
- यूनिक्स, दुर्घटनावश डिलीटीड और अनधिकृत यूजर्स की एक्सेस से इन्फोर्मेशंस को सुरक्षित रखने के लिए कई बिल्ट-इन सिक्योरिटी फीचर्स रखता है।
- यूनिक्स मेनफ्रेम कहलाने वाले सिंगल लार्ज कम्प्यूटर के लिए ऑपरेटिंग सिस्टम के तौर पर मूलरूप से विकसित हुआ था; क्योंकि मल्टीपल यूजर्स एक ही समय में मेनफ्रेम कम्प्यूटर एक्सेज कर सकते हैं।
- यूनिक्स कई प्रोग्राम और मल्टीटास्किंग के तौर पर एक साथ कई काम करने के लिए विकसित किया गया था। यूनिक्स की मल्टीटास्किंग क्षमता इसे नेटवर्क के लिए एफिसिएंट सिस्टम बनाती है।

लिनक्स भी यूनिक्स से मिलता-जुलता (unix-like) operating system है। यह एक multiuser, multitasking system है, तथा इसका file system परम्परागत unix semantics का अनुसरण करता है तथा इसमें standard UNIX नेटवर्किंग मॉडल का पूर्णत: implement किया जाता है।

यूनिक्स में एक कमी थी—इसको समझना तथा चलाना मुश्किल है। एंड्रयू टेनेनबाम, ऐमस्टरडैम में कम्प्यूटर विज्ञान के प्रोफेसर हैं। उन्होंने इसकी सहायता के लिए मिनिक्स नाम का प्रोग्राम लिखा। इसमें भी कुछ कमियाँ थी। लिनूस टोरवाल्ड फिनलैण्ड के हेलसिन्की विश्वविद्यालय में computer science के छात्र थे। उन्होंने मिनिक्स की कमी को दूर करने के लिए एक प्रोग्राम लिखा जो कि बाद में 'लिनूस का यूनिक्स' या छोटे में लिनक्स कहलाया।

Features of Linux (Benefits (फायदे) of Linux)

- Multiuser Operating System
- किसी भी समय एक से अधिक User काम कर सकते हैं और अपने-अपने Program चला सकते हैं।
- Linux Files को Security भी प्रदान करता है।
- अपने से जुड़े सभी User को एक अलग File Set व Directory से जोड़ देता है।
- कोई User सिर्फ अपनी Directory में available Information को ही पढ़ सकता है, delete कर सकता है और modify कर सकता है।
- Other Users की Files safe रहती हैं।
- Linux एक Multi-Programming Operating System है। यह अलग-अलग User के कई प्रोग्रामों को एक साथ चला सकता है।
- Linux में Multi-Programming सुविधा Time sharing से उपलब्ध कराई गई है।

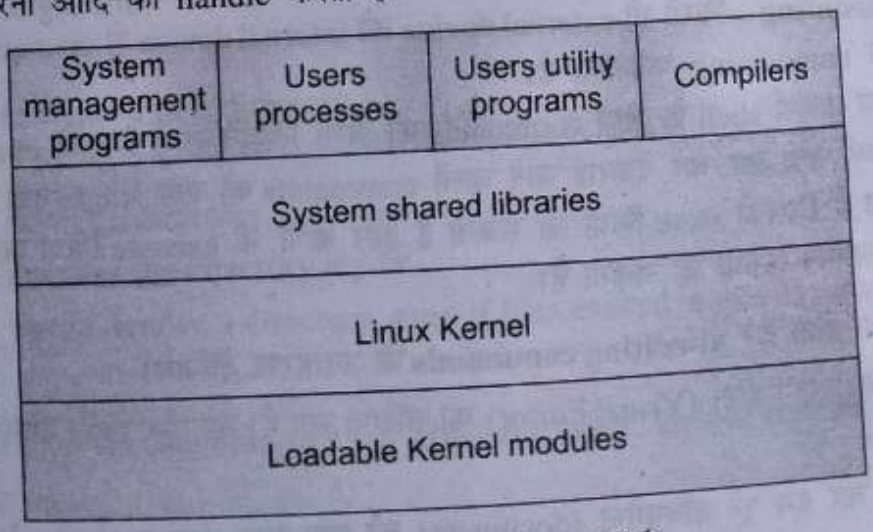
- Linux Operating System का मुख्य भाग Kernel होता है जो Application Crash Proof है।
- Application Crash हो जाने के बाद भी Kernel काम करता है।
- इसका advantage यह होता है कि Linux Operating System को दोबारा चालू नहीं करना पड़ता है।
- Linux Operating System पूरी तरह Virus से सुरक्षित है क्योंकि सामान्य User इसके Kernel तक नहीं जा पाते।
- Linux Operating System भी Windows की तरह Graphical User Interface (GUI) पर काम करता है इसकी Windows को X-Windows कहा जाता है।
- Windows की तरह इसमें भी Desktop, Icon, Menu, Frame आदि समस्त सुविधाएँ होती है। वैसे इसमें जरुरत होने पर इसमें DOS की तरह Command Line Interface (CLI) में भी काम किया जा सकता है।
- Linux में Apache नाम का एक Web Server Program भी उपलब्ध है जिसमें Web pages तैयार किया जाता है। यह आजकल सबसे अधिक लोकप्रिय Web Server है।
- Linux में इसके अलावा बहुत से उपयोगी प्रोग्राम और Free software भी Available है जैसे—Text Editor, Web Browser, Science के काम में आने वाले Software आदि।

लिनक्स सिस्टम में मुख्यत: तीन घटक होते हैं—Kernel, system libraries तथा system utilities.

कर्नल (Kernel): Kernel द्वारा operating system के महत्वपूर्ण abstractions को maintain किया जाता है (Virtual memory तथा processes सहित)।

सिस्टम लाइब्रेरीज़ (System Libraries): सिस्टम लाइब्रेरीज़ में functions का मान समूह (standard set) परिभाषित होता है, जिससे applications Kernel से interact कर सकें।

सिस्टम यूटिलिटीज़ (System Utilities): सिस्टम यूटिलिटीज़ के अंतर्गत वह प्रोग्राम्स आते हैं जोकि individual realization management tasks को perform करते हैं। कुछ सिस्टम यूटिलिटीज़ केवल एक बार invoke की जाती हैं, तथा system को initialize तथा configure करने में सहायक होती हैं जबकि कुछ अन्य यूटिलिटीज़ जिनको की UNIX की शब्दावली में daemons (डैमन्स) कहा जाता है, permanently run करती हैं, तथा कई tasks (कार्य) जैसे कि incoming network connection को respond करना, terminals से logon (लॉग-ऑन) requests को स्वीकार करना, log files को update करना आदि को handle करती हैं।



चित्र—Linux System Components

उपरोक्त चित्र में लिनक्स सिस्टम के विभिन्न घटक प्रदर्शित हैं। Kernel code processor की privileged* mode में कार्य करता है तथा computer के सभी physical resources तक इसकी पहुँच (access) होती है। Linux Kernel लिनक्स ऑपरेटिंग सिस्टम का core होता है। यह processes को run करने हेतु सम्पूर्ण functionality प्रदान करता है तथा hardware resources को arbitration व protected access देने हेतु system services प्रदान करता है।

प्रश्न 29—Shells क्या है? Shells scripts क्या है?

उत्तर—Shell—Shell, unix operating system का एक प्रमुख भाग है। Shell, Unix के कर्नेल और यूज़र के मध्य इन्टरफेस स्थापित करता है।

यूनिक्स ऑपरेटिंग सिस्टम के सभी संस्करण (versions) में कम से कम तीन शैल रहते हैं—

(i) बॉर्न शैल (Bourne shell)

(ii) सी शैल (C shell)

(iii) कॉर्न शैल (Korn shell)

बॉर्न शैल (Bourne Shell): (Sh)—Bourne Shell unix operating system के shell का एक भाग है। इसे Steve Bourne ने विकसित (develop) किया था।

Bourne shell के पास कोई आकृर्षित (interactive) करने वाली properties नहीं होती है। Bourne shell की language काफी कठिन थी।

सी शैल (C Shell): (csh)—सी शैल को Bill Joy ने विकसित किया था। C shell के प्रोग्राम का नाम csh है। इसे % से प्रदर्शित करते हैं। C shell का syntax लगभग programming language C के syntax से काफी मिलता-जुलता है। C shell की परिवार (family) का एक प्रमुख सदस्य (member) tcsh है।

कॉर्न शैल (Korn Shell): (ksh)—कॉर्न शैल (Korn Shell) को डेविड कार्न (David Korn) ने विकसित किया था। यह शैल प्रोग्रामिंग भाग (Shell Programming Language) के साथ C और TC shells के भी गुणों को provide कराती है। Korn shell में अनेक गुण जैसे—एरे (Array), कमाण्ड एलियसिंग (Command Aliasing) इत्यादि उपलब्ध हैं।

Non-Interactive Mode—इस mode में fsck command यदि program run time में कोई error detect करता है तो वह user की बिना permission के उस error को fix तथा solve करके program को continue करता है।

Shutdown System—यूनिक्स ऑपरेटिंग सिस्टम में shutdown command का प्रयोग कम्प्यूटर को turn off या reboot करने के लिए करते हैं। केवल एक superuser ही system को shutdown कर सकता है।

किसी भी unix operating system को तुरन्त shutdown करने के लिए shutdown-h कमाण्ड का प्रयोग या init0 command का प्रयोग करते हैं।

किसी भी unix operating system को reboot करने के लिए shutdown-r command का प्रयोग करते हैं।

Mounting and Unmounting—किसी भी external device को internal device से connect करना ही mounting तथा उसे disconnect करना unmounting कहलाता है।

Shell scripting का तात्पर्य shell के लिए commands की श्रेणी लिखने से है जिनको execute किया जा सके। अत: scripting द्वारा बहुत लम्बी तथा बार-बार दोहराई जाने वाली commands को एक single तथा simple script के रूप में combine कर दिया जाता है जिसको store किया जा सकता है और कभी भी execute किया जा सकता है और इस प्रकार End user के कार्य को आसान बनाया जा सकता है।

प्रश्न 30—vi-editor क्या होता है? vi-editing commands के उदाहरण दीजिये।

उत्तर—vi-editor—vi विजुअल एडीटर (Visual Editor) का संक्षिप्त रूप है। यह एक टेक्स्ट एडीटर (text editor) है।

vi editor का प्रयोग कर हम पूरे डॉक्यूमेन्ट (document) को एक साथ देख सकते हैं और edit भी कर सकते हैं।

vi के मोड्स (Modes of vi)—vi-editor निम्नलिखित तीन modes पर कार्य करता है—

(i) कमाण्ड मोड (Command Mode)

(ii) इन्सर्ट मोड (Insert Mode)

(iii) ex कमाण्ड मोड (ex Command Mode)

Command Mode—इस mode में यूज़र द्वारा की-बोर्ड पर दबाई गई कोई भी 'की' key editor की कमाण्ड के रूप में interpret की जाती है।

Insert Mode—इस mode में आप file में नए टैक्स्ट को Add, पहले टैक्स्ट को delete, या replace कर सकते हैं। Command Mode से इन्सर्ट Mode में आने के लिए Esc key दबाते (press) हैं।

ex कमाण्ड मोड—इस मोड में vi के कमाण्ड, कमाण्ड लाइन (command line) दिए जाते हैं। vi स्क्रीन की सबसे नीचे वाली लाइन को कमाण्ड लाइन कहते हैं जहाँ vi-command लिखे जाते हैं। इसे ex कमाण्ड मोड इसलिए कहते हैं क्योंकि इस मोड के सभी कमाण्ड ex-editor के कमाण्ड से मिलते हैं।

प्रश्न 31—Explain the following Linux commands:

mkdir, cd, rmdir, pwd, ls, who, whoami, date, cat, chmod, cp, mv, rm, pg, more, pr, tail, head, cut, paste, nl, grep, wc, sort, kill, write, talk, mseg, wall, merge, mail, news

mv = renames a file or moves it from one directory to another directory

Syntax

mv [-f] [-i] oldname newname (to rename)

mv [-f] [-i] filename newdirectory

-f mv will move the file(s) without prompting even if it is writing over an existing target. Note that this is the default if the standard input is not a terminal.

-i Prompts before overwriting another file.

cp = copies files from one location to another

Examples

cp file1.txt newdir

rm : Deletes a file without confirmation (by default).

rm myfile.txt : Remove the file myfile.txt without prompting the user.

rm -r directory : Remove a directory, even if files existed in that directory. It will prompt for every single file

rm -rf directory

mkdir : to create a new directory

mkdir filename

rmdir : deletes a directory

rmdir mydir : removes the directory mydir

rm -r directory : would remove a directory, even if files existed in that directory.

head Command

The head command reads the first ten lines of a any given file name. The basic syntax of head command is:

head [options] [file(s)]

tail Command

The tail command allows you to display last ten lines of any text file. Similar to the head command above, tail command also support options 'n' number of lines and 'n' number of characters.

The basic syntax of tail command is:

tail [options] [filenames]

Cat Command

The 'cat' command is most widely used, universal tool. It copies standard input to standard output. The command supports scrolling, if text file doesn't fit the current screen.

The basic syntax of cat command is:

cat [options] [filenames] [-] [filenames]

The Linux "who" command lets you display the users that are currently logged into your Unix computer system. The who command is used to get information about currently logged in user on to system.

Syntax: $who [options] [filename]

pwd command

The pwd command is a command line utility for printing the current working directory. It will print the full system path of the current working directory to standard output. By default the pwd command ignores symlinks, although the full physical path of a current directory can be shown with an option. The pwd command is normally a shell builtin meaning it is part of the code that runs the shell rather than an external executable.

chmod: To change access permissions, change mode.

```
chmod [Options]... Mode [,Mode]... file...
chmod [Options]... Numeric_Mode file...
chmod [Options]... --reference=RFile file...
```

chmod changes the permissions of each given file according to mode, where mode describes the permissions to modify. Mode can be specified with octal numbers or with letters.

OPTIONS

Tag	Description
-f, --silent, --quiet	suppress most error messages
-v, --verbose	output a diagnostic for every file processed
-c, --changes	like verbose but report only when a change is made
-c, --reference=RFile	use RFile's mode instead of MODE values
-R, --recursive	change files and directories recursively
--help	display help and exit
--version	output version information and exit

cut: To divide a file into several parts (columns)

```
cut [OPTION]... [FILE]...
```

cur writes to standard output selected parts of each line of each input file, or standard input if no files are given or for a file name of '-'.

kill: terminate a process

kill [-s signal | -p] [-a] [--] pid ...

kill -l [signal]

The command kill sends the specified signal to the specified process or process group. If no signal is specified, the TERM signal is sent. The TERM signal will kill processes which do not catch this signal.

ls - list directory contents.

• `ls [OPTION]... [FILE]...`

ls List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified. Mandatory arguments to long options are mandatory for short options too.

mail - send and receive mail

mail [-iInv] [-s *subject*] [**-c** *cc-addr*] [**-b** *bcc-addr*] *to-addr*... [-- *sendmail-options*...] **mail [-iInNv] -f** [*name*] **mail [-iInNv] [-u** *user*]

Mail is an intelligent mail processing system, which has a command syntax reminiscent of ed(1) with lines replaced by messages.

Tag	Description
-v	Verbose mode. The details of delivery are displayed on the user's terminal.
-i	Ignore tty interrupt signals. This is particularly useful when using mail on noisy phone lines.
-I	Forces mail to run in interactive mode even when input isn't a terminal. In particular, the '~' special character when sending mail is only active in interactive mode.
-n	Inhibits reading /etc/mail.rc upon startup.
-N	Inhibits the initial display of message headers when reading mail or editing a mail folder.
-s	Specify subject on command line (only the first argument after the -s flag is used as a subject; be careful to quote subjects containing spaces.)
-c	Send carbon copies to list of users.
-b	Send blind carbon copies to list. List should be a comma-separated list of names.
-f	Read in the contents of your mbox (or the specified file) for processing; when you quit, mail writes undeleted messages back to this file.
-u	Is equivalent to: `mail -f /var/spool/mail/user`

paste: merge lines of files

paste [*OPTION*]... [*FILE*]...

Write lines consisting of the sequentially corresponding lines from each FILE, separated by TABs, to standard output. With no FILE, or when FILE is -, read standard input.

pr: convert text files for printing

pr [OPTION]... [FILE]...

Paginate or columnate FILE(s) for printing.

sort: sort lines of text files

sort [*OPTION*]... [*FILE*]...

Write sorted concatenation of all FILE(s) to standard output.

talk: talk to another user

talk person [ttyname]

Talk is a visual communication program which copies lines from your terminal to that of another user.

wall -- send a message to everybody's terminal.

wall [-n] [message]

Wall sends a message to everybody logged in with their **mesg**(1) permission set to **yes**. The message can be given as an argument to *wall*, or it can be sent to *wall*'s standard input. When using the standard input from a terminal, the message should be terminated with the **EOF** key (usually Control-D).

The length of the message is limited to 22 lines.

write: send a message to another user

write *user [ttyname]*

Write allows you to communicate with other users, by copying lines from your terminal to theirs.

wc: print the number of newlines, words, and bytes in files.

wc *[OPTION]... [FILE]...*

Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified.

whoami: print effective userid

SYNOPSIS

whoami *[OPTION]...*

DESCRIPTION

Print the user name associated with the current effective user ID.

grep: print lines matching a pattern

```
grep [OPTIONS] PATTERN [FILE...]
grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]
```

grep searches the named input FILEs (or standard input if no files are named, or if a single hyphen-minus (-) is given as file name) for lines containing a match to the given PATTERN. By default, **grep** prints the matching lines.

more: file perusal filter for crt viewing

more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] []

More is a filter for paging through text one screenful at a time

nl is a linux command to number lines of the files, it copies its files to standard output, prepending line numbers.It's more flexible than cat with its -n and -b options, providing an almost bizarre amount of control over the numbering. nl can be used in two ways: on ordinary test files, and on specially marked up text files with predefined headrs and footers.

```
nl [OPTION]... [FILE]...
```

The date command displays or sets the system date. It is most commonly used to print the date and time in different formats and calculate future and past dates.

The syntax for the date command is as follows:

```
date [OPTION]... [+FORMAT]
```

Copy

To display the current system time and date using the default formatting, invoke the command without any options:

```
date
```

The output will include the day of the week, month, day of the month, time, time zone, and year:

प्रश्न 32—Linux command के नाम, description व उदाहरण दीजिये।

उत्तर—

Linux command	Description	Linux command example
cd	Change directory with a specified path	*cd/path/directory 1*
clear	Clear the screen	clear
cp	Copy file (s)	cp/*path1/file1/path2/file1*
diff	Compare the contents of files	diff *file 1 file1*
exit	Log out of Linux	exit
grep	Find a string of text in a file	grep ''word or phrase'' *file 1*
head	Display beginning of a file	head *file 1*
less	View a file	less *file 1*
ls	List contents of a directory	ls/*path/directory 1*
mv	Move file(s) or rename file(s)	mv/*path1/file1/path2/file1*
mkdir	Create a directory	mkdir *directory*
rm	Delete file(s)	rm *file1*
rmdir	Remove a directory	rmdir *directory*
tail	Display end of a file	tail *file 1*
tar	Store, list or extract files in an archive	tar *file 1*
vi	Edit file(s) with simple text editor	vi *file 1*

❑❑❑

13 Chapter

बहुविकल्पीय प्रश्न बैंक
(OBJECTIVE TYPE QUESTIONS)

THINK ABOUT IT

The only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. — **Steve Jobs**

Life is like photography. You need the negatives to develop. - Unknown

You need chaos in your soul to give birth to a dancing star. — **Friedrich Nietzsche**

1. कम्प्यूटर का दिमाग होता है—
   (a) CPU (b) Keyboard
   (c) Monitor (d) DVD
2. निम्नलिखित में से processing कौन करता है—
   (a) CPU (b) Keyboard
   (c) Monitor (d) DVD
3. कम्प्यूटर द्वारा किये गये कार्य का सिद्धान्त कहा जाता है—
   (a) PIO चक्र (b) IPO चक्र
   (c) OIP चक्र (d) OPI चक्र
4. निम्नलिखित में से कौन सी कम्प्यूटर का strength नहीं है—
   (a) High speed
   (b) Reliability (विश्वसनीयता)
   (c) Large storage capacity
   (d) Strong IQ
5. कम्प्यूटर की प्रथम जनरेशन किस पर आधारित थी—
   (a) Vacuum tables
   (b) Transistors
   (c) ICs
   (d) Artificial intelligence
6. निम्नलिखित में से कौन सा Portable computer है—
   (a) Super computer
   (b) Mainframe computer
   (c) Laptop
   (d) Mini computer
7. कम्प्यूटर किसकी प्रौसेसिंग करता है—
   (a) Hardware (b) Software
   (c) Data (d) Information
8. निम्नलिखित में से Input युक्ति नहीं है—
   (a) Keyboard (b) Mouse
   (c) Touch screen (d) Plotter
9. निम्नलिखित में से Output युक्ति नहीं है—
   (a) Printer (b) Speaker
   (c) Plotter (d) OMR
10. Optical media का उदाहरण है—
    (a) Blu ray (b) Pen drive
    (c) ROM (d) Hard disk
11. ASCII का full form है—
    (a) American Standard Code for International Interchange
    (b) Australian Standard Code for International Interchange
    (c) American Standard Code for Information Interchange
    (d) Australian Standard Code for Information Interchange
12. CAD का full form है—
    (a) Computer Aided Drawing
    (b) Computer Aided Design
    (c) Computer Advanced Drawing
    (d) Computer Advanced Design
13. OMR का full form है—

(a) Office Mark Reader
(b) Office Manual Reader
(c) Optical Mark Reader
(d) Optical Manual Reader

14. OCR का full form है—
(a) Office Character Reader
(b) Office Cursor Reader
(c) Optical Character Reader
(d) Optical Cursor Reader

15. कम्प्यूटर का जनक कहा जाता है—
(a) John Henry
(b) David Boon
(c) Malcolm Marshall
(d) Charles Babbage

16. 1 byte तुल्य होती है—
(a) 2 bits (b) 4 bits
(c) 6 bits (d) 8 bits

17. 1 nibble तुल्य होती है—
(a) 2 bits (b) 4 bits
(c) 6 bits (d) 8 bits

18. Compilers translate करते हैं—
(a) Low level language program को High level language program में, line by line
(b) Low level language program को High level language program में, पूरे प्रोग्राम को एक साथ
(c) High level language program को Low level language program में, line by line
(d) High level language program को Low level language program में, पूरे प्रोग्राम को एक साथ

19. Interpreters translate करते हैं—
(a) Low level language program को High level language program में, line by line
(c) Low level language program को High level language program में, पूरे प्रोग्राम को एक साथ
(b) High level language program को Low level language program में, line by line
(d) High level language program को Low level language program में, पूरे प्रोग्राम को एक साथ

20. Assemblers convert करते हैं—
(a) Assembly language program को Machine language program में
(b) Machine language program को Assembly language program में
(c) High level language program को Low level language program में, line by line
(d) High level language program को Low level language program में, पूरे प्रोग्राम को एक साथ

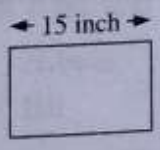
21. Windows 95 क्या है?
(a) Hardware (b) CPU
(c) Memory (d) Operating system

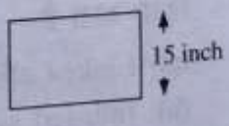
22. निम्नलिखित में से कौन सा एक smartphone operating system है—
(a) Unix (b) Linux
(c) Blackberry (d) MS Windows

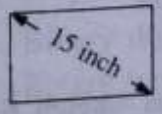
23. यदि CPU को आवंटित करने के बाद कार्य पूरा होने तक CPU वापस ना लिया जाए तो ऐसी scheduling कहलाती है—
(a) Preemptive scheduling
(b) Nonpreemptive scheduling
(c) Long-term scheduling
(d) Short term scheduling

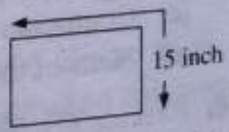
24. Belady's Anomaly किस page replacement algorithm से सम्बन्धित है—
(a) FIFO (b) LRU
(c) MFU (d) LFU

25. यदि किसी मॉनीटर की स्क्रीन का size 15 inch specify किया गया है तो 15 inch कौन की लम्बाई को इंगित करना है—

(a) 

(b) 

(c) 

(d) 

26. यदि किसी computer system में एक से अधिक CPU's से कार्य लिया जाता है, तो यह कहलाता है—

(a) Multitasking (b) Multiprocessing
(c) Batch processing (d) Time sharing

27. किसी file name के आगे प्रयुक्त .doc, .bat, .exe इत्यादि कहलाते हैं—
(a) Support names (b) Extensions
(c) Batch names (d) Speeial names

28. निम्नलिखित में से कौन सा कार्य file operations के अंतर्गत नहीं आता—
(a) Create (b) Open
(c) Close (d) Locate

29. यदि file क्रम में process की जाये तो यह कहलाता है—
(a) Sequential access (b) Direct access
(c) Indexed access (d) Locked access

30. यदि file को लगातार (सतत्) memory locations आवंटित की जाये, तो यह कहलाता है—
(a) Contiguous allocation
(b) Indexed allocation
(c) Linked allocation
(d) Locked allocation

31. Memory allocation के साथ small unused holes का उत्पन्न होना कहलाता है—
(a) Locking (b) Compaction
(c) Swapping (d) Fragmentation

32. निम्नलिखित में से किस allocation विधि द्वारा memory allocate करते समय file के size का पूर्वानुमान होना आवश्यक है—
(a) Linked allocation
(b) Indexed allocation
(c) Contiguous allocation
(d) None of these

33. Index block का use किस allocation विधि में किया जाता है—
(a) Linked allocation
(b) Indexed allocation
(c) Contiguous allocation
(d) None of these

34. Execute होने वाला program कहलाता है—
(a) Resource (b) Data
(c) Process (d) Stack

35. New, ready, running, waiting, halted क्या है—
(a) Process states
(b) CPU states
(c) Scheduling algorithms
(d) Allocation methods

36. एक साथ होने वाले process कहलाते हैं—
(a) Deadlocked processes
(b) Logical processes
(c) Concurrent processes
(d) Idle processes

37. CPU scheduling का मुख्य उद्देश्य है—
(a) Fairness and policy enforcement
(b) Efficiency and throughput
(c) Response time and turn around time
(d) All of the above

38. Time slice या quantum किस CPU scheduling algorithm से संबंधित है—
(a) FIFO scheduling
(b) Round Robin scheduling
(c) SJF scheduling
(d) Priority scheduling

39. निम्न में से कौन सी disc scheduling में starvation की समस्या उत्पन्न हो सकती है—
(a) FCFS (b) SSTF
(c) SCAN (d) LOOK

40. किस disc scheduling algorithm को elevator algorithm भी कहा जाता है—
(a) FCFS (b) SSTF
(c) SCAN (d) C-SCAN

41. वह तकनीक जिसके द्वारा process को run न होने की स्थिति में disk में भेज दिया जाता है, कहलाती है—
(a) Swapping
(b) Mapping
(c) Internal fragmentation
(d) External fragmentation

42. उस block का चयन जिसका size requested size के निकटतम है, कहलाता है—
(a) First fit (b) Next fit
(c) Best fit (d) Worst fit

43. सभी holes को combine करके एक कर देना, ताकि fragmentation न रहे, कहलाता है—
(a) Mapping (b) Addressing
(c) Swapping (d) Compaction

44. यदि MMV एक ऐसे पेज की तलाश करता है, जो memory में उपलब्ध नहीं है, तो इसे कहा जाता है—
(a) Page table (b) Page frame
(c) Page fault (d) Page directory

45. Memory को variable size chunks में विभाजित करना, कहलाता है—
(a) Segmentation (b) Paging
(c) Demand paging (d) Swapping

46. केवल उन्हीं pages को memory में प्रवेश देना, जिनकी current running process को आवश्यकता है, कहलाता है—
(a) Segmentation (b) Paging
(c) Demand paging (d) Swapping

47. MS Word file से selected text को copy करने हेतु shortcut है—
(a) Ctrl + X (b) Ctrl + V
(c) Ctrl + C (d) Ctrl + B

48. MS Word file पर selected text को paste करने हेतु short cut है—
(a) Ctrl + X (b) Ctrl + V
(c) Ctrl + C (d) Ctrl + B

49. सभी open Windows को एक साथ minimize करने हेतु use किया जाने वाला shortcut
(a) Windows + M (b) Ctrl + M
(c) Windows + T (d) Control + X

50. Window screen के नीचे वाली row जहाँ currently opened applications listed होती है, कहलाती है—
(a) Tool bar (b) Format bar
(c) Desktop (d) Task bar

51. किसी file को delete करने पर वह कहाँ पहुँचा दी जाती है—
(a) Desktop (b) Recycle bin
(c) Control panel (d) My documents

52. यदि कोई process एक ऐसे event के पूर्ण होने प्रतीक्षा कर रहा है, जो कभी पूर्ण होने वाला नहीं है, तो यह स्थिति कहलाती है—
(a) Page fault (b) Segmentation
(c) Swapping (d) None of these

53. Deadlock हेतु four conditions कहलाती है—
(a) Barkhausen's conditions
(b) Computer conditions
(c) Coffman's conditions
(d) Newton's conditions

54. Deadlock prevention से संबंधित algorithm है—
(a) Marshall's algorithm
(b) Hook's algorithm
(c) Holding's algorithm
(d) Banker's algorithm

55. यदि real time system में कार्य समय से पूरा न करने पर दी जाने वाली सजा कार्य समय से पूरा करने पर दिये जाने वाले पुरस्कार से अधिक हो तो ऐसा real time system कहलाता है—
(a) Hard real time system
(b) Soft real time system
(c) Correct real time system
(d) Incorrect real time system

56. Integrated circuits या ICs को सामान्यत: किस नाम से जाना जाता है—
(a) डायोड (b) ट्रांजिस्टर
(c) चिप (d) प्रतिरोध

57. 1 बाइट का तात्पर्य है—
(a) 8 bits (b) 4 bits
(c) 2 bits (d) 16 bits

58. 1 Nibble का तात्पर्य है—
(a) 8 bits (b) 4 bits
(c) 2 bits (d) 16 bits

59. 1 या 0 को कहा जाता है—
(a) Nibble (b) Byte
(c) Bit (d) Symbol

60. Bit का पूरा नाम है—
(a) Binary digit (b) Binary team
(c) Binary transmission (d) Best Indian team

61. नियमों का समूह कहलाता है—
(a) Modem (b) Multiplexer
(c) Protocol (d) Modulator

62. rajeev222@gmail.com दर्शाता है—
(a) Website address (b) E-mail address
(c) Computer address (d) Virtual address

63. www.facebook.com दर्शाता है—
(a) Website address (b) E-mail address
(c) Computer address (d) Virtual address

64. Internet Explorer क्या है—
(a) Browser (b) Virus
(c) Process (d) Repeater

65. www का तात्पर्य है—
(a) Word Wide Web (b) Wide Wide Web
(c) World Wide Web (d) World Wide West

66. कम्प्यूटर कौन सी भाषा समझता है—
(a) English (b) Japanese
(c) Hindi (d) Binary

67. C क्या है—
(a) low level language
(b) High level language
(c) Assembly language
(d) Binary language

68. कम्प्यूटर द्वारा कार्य करवाने हेतु दिये गये निर्देशों का समूह कहलाता है—
(a) Program (b) Instruction
(c) Icon (d) Peripheral

69. कौन सी function key "save as" को प्रदर्शित करती है—
(a) F9 (b) F10
(c) F11 (d) F12

70. कौन सा shortcut "save" को represent करता है—
(a) Shift + F12 (b) Ctrl + F12
(c) Shift + F11 (d) Ctrl + F11

71. .rar, .win तथा .zip प्रदर्शित करते हैं—
(a) Web file extensions
(b) Backup file extensions
(c) Compressed file extensions
(d) Image file extensions

72. .jpg, .jpeg, .bmp, .tif, .png, .gif प्रदर्शित करते हैं—
(a) Web file extensions
(b) Back up file extensions
(c) Compressed file extensions
(d) Image file extensions

73. कम्प्यूटर में प्रयुक्त ICs सामान्यत: बनी होती है—
(a) कार्बन (b) सिलिकॉन
(c) Silver (d) Gold

74. Software code में त्रुटियों को जाँचने के प्रक्रिया कहलाती है—
(a) Compiling (b) Debugging
(c) Coding (d) Decoding

75. ASCII code ने English alphabets numbers व symbols को binary code allot किया। एक नया स्टैंडर्ड जो कि विश्व की लगभग सभी भाषाओं को कोड allot करता है, कहलाता है—
(a) Alpha code (b) Beta code
(c) Uni code (d) Language code

उत्तरमाला (Answers)

1. (a)	2. (a)	3. (b)	4. (d)	5. (a)	6. (c)	7. (c)	8. (d)	9. (d)
10. (a)	11. (c)	12. (b)	13. (c)	14. (c)	15. (d)	16. (d)	17. (b)	18. (d)
19. (c)	20. (a)	21. (d)	22. (c)	23. (b)	24. (a)	25. (c)	26. (b)	27. (b)
28. (d)	29. (a)	30. (a)	31. (d)	32. (c)	33. (b)	34. (c)	35. (a)	36. (c)
37. (d)	38. (b)	39. (b)	40. (c)	41. (a)	42. (c)	43. (d)	44. (c)	45. (a)
46. (c)	47. (c)	48. (b)	49. (a)	50. (d)	51. (b)	52. (d)	53. (c)	54. (d)
55. (a)	56. (c)	57. (a)	58. (b)	59. (c)	60. (a)	61. (c)	62. (b)	63. (a)
64. (a)	65. (c)	66. (d)	67. (b)	68. (a)	69. (d)	70. (a)	71. (c)	72. (d)
73. (b)	74. (b)	75. (c)						

बोर्ड परीक्षा में पूछे जाने वाले महत्वपूर्ण प्रश्न

1. (a) Operating system से आप क्या समझते हैं?
   (b) Operating systems के विभिन्न कार्यों का वर्णन कीजिये।
   (c) ऑपरेटिंग सिस्टम के विकास पर टिप्पणी लिखिये।
   (d) ऑपरेटिंग सिस्टम के विभिन्न प्रकारों का वर्णन कीजिये।
   (e) निम्न को समझाइये—(i) Batch operating system (ii) Interactive operating system (iii) Time sharing operating system (iv) Real time system (v) Network operating system (vi) Distributed operating systems
2. (a) File क्या होती है? File के विभिन्न attributes व types बताइये।
   (b) File access methods क्या होते हैं? विभिन्न file access methods का वर्णन कीजिये।
   (c) File directory system का वर्णन कीजिये। Single level, Two level, Tree structured, Acyclic व General graph directory का वर्णन कीजिये।
   (d) File allocation methods क्या होते है? विभिन्न File allocation methods का वर्णन कीजिये।
3. (a) CPU scheduling से आप क्या समझते हैं?
   (b) CPU scheduling की मुख्य एल्गोरिद्म्स पर चर्चा कीजिये
   (c) डिस्क scheduling की मुख्य algorithms पर चर्चा कीजिये।
4. (a) Memory management से आप क्या समझते हैं? इसके क्या उद्देश्य हैं।
   (b) निम्न को समझाइये
      (i) Swapping (ii) Paging (iii) Segmentation
      (iv) Demand paging (v) Multiple partitions
   (c) Page replacement क्या है। विभिन्न page replacement policies को समझाइये।
5. (a) डैडलॉक से आप क्या समझते हैं?
   (b) डैडलॉक हेतु चार शर्तों का उल्लेख कीजिये।
   (c) डैडलॉक को Handle करने की विधियाँ बताइये।
   (d) निम्न को समझाइये—
      (i) Deadlock Prevention (ii) Deadlock Avoidance
      (iii) Banker's Algorithm (iv) Deadlock Detection and Recovery
6. (a) What is Unix? Explain the main features of Unix.
   (b) What is Linux? Explain the main features of Linux.
   (c) Explain the basic concepts of Unix architecture.
   (d) What are the main components of Linux system?
   (e) What are Kernel modules and their function?
   (f) What are daemons?
   (g) Explain the process management, memory management and file system in Unix and Linux.

MODEL PAPER

Operating System

[Time : 2:30 Hours] **[Maximum Marks : 50]**

1. Answer any **two** of the following:
   (a) What do you mean by Operating system? Explain the characteristics of operating system.
   (b) What are GUI and CUI? Explain Single user, Multi user operating system, Time Sharing and Real Time System.
   (c) Explain Process Management? What are Process Synchronization, Inter process communication, CPU scheduling and dead lock.
2. Answer any **three** of the following:
   (a) What is Memory Management? What are Main memory, Contiguous memory allocation, Segmentation?
   (b) Explain about Paging, Virtual memory, Demand paging, Page replacement, Allocation.
   (c) Briefly describe Disk scheduling and Management.
   (d) Calculate turn around time and average waiting time for following set of processes, if these processes are scheduled using
      (i) SJF
      (ii) Priority (both preemptive)

Process id	*Arrival Time*	*Execution Time*	*Priority*
P_1	7	1	0
P_2	3	2	4
P_3	9	3	7

3. Answer any **three** of the following:
   (a) Explain the File concepts, File system and structure, Directory structure.
   (b) Explain Input Output Management and Mass storage structure.
   (c) Describe File Management.
   (d) The head of a moving head disk with 200 tracks numbered 0 to 199 is currently serving a request at track 143, consider an ordered disk queue with requests involving track numbers:

   86, 147, 91, 177, 94, 150, 102, 175, 130.

   What is the total head movement to satisfy this request for the following disk scheduling algorithms?
      (i) FCFS (First come first serve)
      (ii) SSTF (Shortest seek time first)

4. Answer any **two** of the following:

(a) Explain the meaning of Process schedule.

(b) Differentiate between FCFS and SJF algorithms.

(c) Describe Linux and Unix basic concepts.

5. Answer any **four** of the following:

(a) Describe about system administration in Unix.

(b) Describe about Evolution of Operating system .

(c) Describe the conditions of Deadlock.

(d) Write some commands of Unix and Linux. What is vi editor?

(e) Write short note on Banker's algorithm.

(EVEN SEMESTER) JUNE-2019 EXAMINATION

ऑपरेटिंग सिस्टम

(Operating System)

[Time : 2:30 Hours] [Maximum Marks : 50]

नोट: सभी प्रश्नों के उत्तर दीजिए।

1. निम्न में से किन्हीं **दो** का उत्तर दीजिए। [2 × 5 = 10]
   (अ) प्रचालन तन्त्र से आप क्या समझते हैं? प्रचालन तन्त्र के प्रमुख कार्य कौन से हैं?
   (ब) Time sharing तन्त्र क्या है? इसके लाभ बताइए।
   (स) Real Time तन्त्र की व्याख्या कीजिये।

2. निम्न में से किन्हीं **दो** का उत्तर दीजिए : [2 × 5 = 10]
   (अ) फाइल क्या है? फाइल के access विधियों की व्याख्या कीजिये।
   (ब) Linux में उपलब्ध विभिन्न प्रणाली प्रबंधन समादेश कौन-कौन से हैं? व्याख्या कीजिये।
   (स) UNIX की directory संरचना की व्याख्या कीजिये।

3. निम्न में से किन्हीं दो का उत्तर दीजिए: [2 × 5 = 10]
   (अ) Preemptive एवं Non-preemptive CPU scheduling में विभेद कीजिए।
   (ब) SJF CPU scheduling की उदाहरण सहित व्याख्या कीजिए।
   (स) स्मृति प्रबंधन में swapping का क्या तात्पर्य है?

4. निम्न में से किन्हीं **दो** का उत्तर दीजिए– [2 × 5 = 10]
   (अ) स्मृति आवंटन के लिए First fit, Best fit एवं Worst fit रणनीतियों की उपयुक्त उदाहरण सहित वर्णन कीजिए।
   (ब) Demand paging की अवधारणा की व्याख्या कीजिए।
   (स) Deadlock का पता लगाने की रणनीतियों की व्याख्या कीजिए।

5. निम्न में से किन्हीं **दो** पर संक्षिप्त टिप्पणी लिखिए: [2 × 5 = 10]
   (अ) Scan Disk Scheduling
   (ब) Segmentation
   (स) Deadlock Prevention

एक बार एक व्यक्ति चित्रों की दुकान पर गया। उसने वहाँ पर अजीब से चित्र देखे। पहले चित्र में चेहरा पूरी तरह बालों से ढका हुआ था और पैरों में पंख थे। एक दूसरे चित्र में सिर पीछे से गंजा था। व्यक्ति ने पूछा - यह चित्र किसका है? दुकानदार ने कहा - अवसर का। व्यक्ति ने पूछा - इसका चेहरा बालों से ढका क्यों है? दुकानदार ने कहा - क्योंकि अक्सर जब अवसर आता है तो मनुष्य उसे पहचान नहीं पाता । व्यक्ति ने पूछा - और इसके पैरों में पंख क्यों हैं? दुकानदार ने कहा - वह इसलिये कि यह तुरंत वापस भाग जाता है, यदि इसका उपयोग न हो तो यह तुरंत उड़ जाता है। व्यक्ति ने पूछा - और यह दूसरे चित्र में पीछे से गंजा सिर किसका है? दुकानदार ने कहा - यह भी अवसर का है। यदि अवसर को सामने से ही बालों से पकड़ लेंगे तो वह आपका है। अगर आपने उसे थोड़ी देरी से पकड़ने की कोशिश की तो पीछे का गंजा सिर हाथ आयेगा और अवसर फिसलकर भाग जायेगा।

वह व्यक्ति इन चित्रों का रहस्य जानकर हैरान था पर अब वह बात समझ चुका था। ईश्वर ने हमें ढेरों अवसरों के बीच जन्म दिया है । अवसर हमेशा हमारे सामने से आते जाते रहते हैं पर हम उन्हें पहचानने में देरी कर देते हैं। और कई बार हम सिर्फ इसलिये नहीं पहचानते क्योंकि हम बड़े अवसर की ताक में रहते हैं। पर अवसर बड़ा या छोटा नहीं होता।

वो पथ क्या पथिक कुशलता क्या, जिस पथ में बिखरे शूल न हों,
नाविक की धैर्य परीक्षा क्या, जब धाराएँ प्रतिकूल न हों।
यह अरण्य झुरमुट जो काटे अपनी राह बना ले,
कृत दास यह नहीं किसी का जो चाहे अपना ले।
जीवन उनका नहीं युधिष्ठिर जो इससे डरते हैं,
यह उनका जो चरण रोप निर्भय होकर चलते हैं।

आपकी सफलता की शुभकामनाओं सहित,

राहुल वाधवा