Saiphy.com

12th Standard

Computer Science 1 and 2

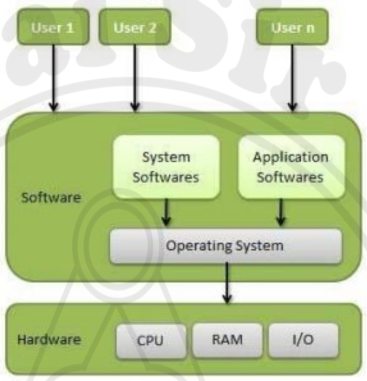
Revision Notes - By Sai Sir

CS XII STD

TOPIC: OPERATING SYSTEM BASICS

Q1. **Explain the idea of an Operating System**
OR
What is an Operating System? Write its functions

An operating system is a ***program that acts as an interface*** between the **user and the computer hardware** and controls the execution of all kinds of programs.
An operating system is a collection of integrated, interrelated and interdependent groups of programs , which are designed to perform various internal operations of Computer efficiently and make it easier for people to use the computer .



Tasks/Functions of Operating System :

1) Recognizing input from input devices
2) Sending Output to Output devices
3) keeping track of files and directories on secondary storage devices
4) Controlling Peripheral Devices
5) Resource Allocation
6) Error Detection
7) Accounting
8) Security and protection

Q2. **Explain the services provided by the Operating System.**
OR
What are the three main areas in which the operating system divides its services. Give examples.

I. Information Management:
It provides facilities to store, retrieve, modify, remove the information on various devices.
Services provided are-
a. Create files or directories
b. Open files or explore directories
c. Delete, copy or close files.
d. Change working directory.

II. Process Management:
Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Process is a program that is under execution, which is an important part of modern-day operating systems.

III. Memory Management:
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.

Q3.	**What is system call? List any two system calls for Memory management, Process management and information management.**
	The interface between a process and an operating system is provided by system calls. In general, system calls are available as assembly language instructions. System calls are usually made when a process in user mode requires access to a resource. Here are the types of system calls − **Process Management** These system calls deal with processes such as process creation, process termination, process blocking, process delay. **Information Management** These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file, closing a file, creating a link. **Memory Management** These system calls are responsible for allocating a chunk of memory to a process, deallocating a chunk of memory from the process. **Communication** These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.
Q4.	**Explain in brief the following programs of Windows OS.** **a) Control Panel b) File Manager c) Program Manager**
	Control Panel: • The Microsoft Windows Control Panel is a management tool for the Windows operating system (OS). • Among other things, end users can use the Control Panel to troubleshoot problems, backup the computer, view the computer's network status, choose sharing options, view devices and printers, add devices, add or remove programs, connect to a projector, manage the user account, adjust screen resolution, change the desktop theme, change the display background or language and specify input methods. **File Manager:** • A **file manager** is a software program that helps a user manage all the files on their computer. • For example, all file managers allow the user to view, edit, copy, and delete the files on their computer storage devices. **Program Manager:** • Program Manager refers to the basic windows of Microsoft Windows 3.x that allow users to select and run each program on their operating system. • It was the main screen of Windows 3.x. • All the programs were loaded at the time of startup, and the programs appearing could be customized by the user. • This feature was also made available in Windows 95, 98, NT, 2000 and XP.
Q5.	**Write a note on Information Management** **OR** **What is Information Management? List the system calls in it.**
	Information Management: It provides facilities to store, retrieve, modify, remove the information on various devices. **Some of the system calls in IM are:** ❑ Create a file.

- ❑ Create a directory.
- ❑ Open a file (R/W/RW).
- ❑ Explore a directory.
- ❑ Close a file.
- ❑ Read data from file to buffer.
- ❑ Write data from buffer to file.
- ❑ Move the files pointer.
- ❑ Read and return file's status.
- ❑ Create a link.
- ❑ Change working directory.

Q6. **Write a note on Process Management.**

OR

What is Process Management? List the system calls in it.

Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Process is a program that is under execution, which is an important part of modern-day operating systems.

Some of the system calls in PM are:

- Create a child process identical to parent.
- Terminate a process
- Wait for a child process to terminate.
- Change the priority of a process.
- Block the process.
- Ready the process.
- Dispatch the process.
- Suspend the process.
- Resume the process.
- Delay a process.
- Fork a process.

Q7. **Write a note on Memory Management.**

OR

What is Memory Management? List the system calls in it.

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.

Some of the system calls in MM are:

- Allocating a chunk of memory to a process,
- Deallocating a chunk of memory from the process.

Q8. **State the features of the following Operating systems.**
I. Windows NT II. Linux III. Windows 98

I. Windows NT

Following are some of the important features of Windows NT Operating System.

- **Interface**
  Contains the Windows 95 interface and features like the Start button, Taskbar, Explorer, Network Neighborhood, and Briefcase

- **Networking**
  NetWare client and login script support

Support for 15 network protocols

- **Graphics and Multimedia**
  Significant performance gains for graphic intensive applications

- **Utilities**
  File compression with NTFS
  User Manager for configuration and security

- **Hardware Support**
  Multiple hardware configuration; you can specify a hardware profile at start time, including services, devices, and video resolutions

II. Linux
Following are some of the important features of Linux Operating System.

- **Portable** − Portability means software can works on different types of hardware in same way.
- **Open Source** − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** − Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** − Linux provides a standard file structure in which system files/ user files are arranged.
- **Security** − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

III. Windows 98
Following are some of the important features of Windows98 System.

- Windows98 has integrated the Internet standard comprehensively.
- It simplifies desktop with Internet technology and allows the users to find and browse the information on the computer or on the internet more simply and faster.
- It has a faster speed and greater stability.
- With the brand new self-maintenance and updating function, users can have more spare time to concentrate on work or games instead of system management.
- It is very Convenient to install.
- The hardware being supported runs faster and more effectively under Windows98 environment.
- When using the "plug and play" device on the computer Windows98 will set it automatically.
- The desktop of Windows98 makes you concentrate more on your own task by providing a simpler User Interface.

Q9. **What are the components of Linux OS? Explain any three features.**
OR
List any 6 features of Linux OS.

- Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System:

- **Kernel** − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware.
- **System Library** − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system.
- **System Utility** − System Utility programs are responsible to do specialized, individual level tasks.

Features of Linux OS:

- **Portable** − Portability means software can works on different types of hardware in same way.
- **Open Source** − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** − Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** − Linux provides a standard file structure in which system files/ user files are arranged.
- **Security** − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

TOPIC: INFORMATION MANAGEMENT(FILE SYSTEM)

Q10. What is a file system? Explain the various types of file systems.

- A file is a named collection of related information that is recorded on secondary storage.
- In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.
- Every file generally has these properties : name , extension , size , type , etc.
- There are two types of file systems:
  1. Tape based
  2. Disk Based

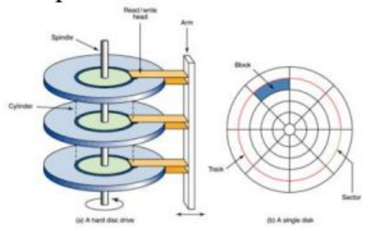
1. Tape based :

- Simple but inefficient
- Files stored on physical tape reels
- One or more files stored on to one tape
- Used for data transfer, backup or archival storage

Disadvantages:
- Finding a file on tape is time consuming
- If a file is to be modified, entire tape needs to be copied.

2. Disk based :
- Each disk is divided into tracks and each track is divided into sectors.
- A sector is the smallest unit of information which can be read from and written to the disk.

- Sector varies from 32 bytes to 4096 bytes .
- A track contains 4 to 32 sectors per track and from 75 to 500 tracks per disk .



Q11. **Explain the following terms in case of magnetic disk:**
a. Tracks and sectors **b. Seek Time**
c. Transmission time **d. Rotational delay/ Latency time**

a. Tracks and sectors:
Magnetic disk is made up of concentric circles called **tracks**. The number of tracks varies depending on the disk type. Each track is further divided into **sectors**.
A sector is a smallest unit of information which can be read from or written to the disk.
Sector varies from 32 bytes to 4096 bytes .
A track contains 4 to 32 sectors per track and from 75 to 500 tracks per disk .

b. Seek Time:
A disk is divided into many circular tracks. Seek Time is defined as the time required by the read/write head to move from one track to another.

c. Transmission time:
It is the time required for activate Read/Write head for appropriate surface and read data is called Transmission time.

d. Rotational delay –
It is the time required by the read/write head to move from the current sector to the requested sector.

Q12. **State the different operations that can be performed on files.**
OR
Explain file system related to Information Management with file operations only.

A file is a named collection of related information that is recorded on secondary storage.
There are various operations which can be implemented on a file.

1. Create: Creation of the file is the most important operation on the file. Different types of files are created by different methods.

2. Write: Writing the file is different from creating the file. The OS maintains a write pointer for every file which points to the position in the file from which, the data needs to be written.

3. Read: Every file is opened in three different modes : Read, Write and append. A Read pointer is maintained by the OS, pointing to the position up to which, the data has been read.

4. Re-position: Re-positioning is simply moving the file pointers forward or backward depending upon the user's requirement. It is also called as seeking.

5. Delete: Deleting the file will not only delete all the data stored inside the file, It also deletes all the attributes of the file. The space which is allocated to the file will now become available and can be allocated to the other files.

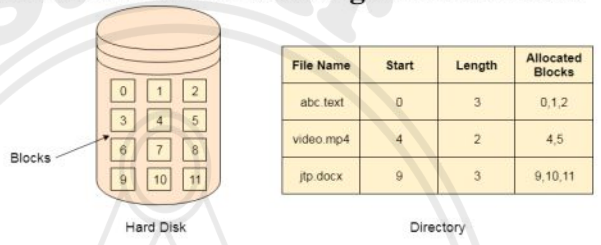
Q13. **Explain the various File Access mechanisms.**

File Access Mechanisms
File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

1. Contiguous Allocation
2. Noncontiguous Allocation

1. Contiguous Allocation:

- If the blocks are allocated to the file in such a way that all the logical blocks of the file get the contiguous physical block in the hard disk then such allocation scheme is known as **contiguous allocation.**



File Name	Start	Length	Allocated Blocks
abc.text	0	3	0,1,2
video.mp4	4	2	4,5
jtp.docx	9	3	9,10,11

Contiguous Allocation

- Whenever a new file is created depending upon the size of file operating system can allocate continuous area. If there are multiple free blocks then which should be chosen? For that the following methods are used.
- **First fit:** In this method, first job claims the first available memory with space more than or equal to it's size. The OS doesn't search for appropriate partition but just allocate the job to nearest memory partition available with sufficient size.
- **Best fit:** This method keeps the free/busy list in order by size – smallest to largest. In this method, the OS first searches the whole of memory according to the size of the given job and allocates it to closest-fitting free partition in the memory. Jobs are in the order from smallest job to largest job.
- **Worst fit:** In this allocation technique the process traverse the whole memory and always search for largest hole/partition, and then the process is placed in that hole/partition

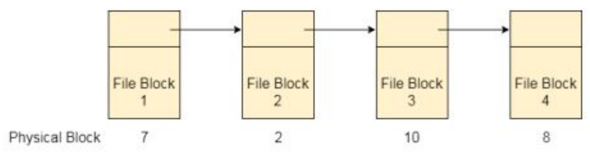
2. Noncontiguous Allocation:

In this method, size of file does not have to be predicted at the beginning. File can grow with time as per needs. It reduces wastage of space and OS automatically allocates additional blocks as per requirement without aborting program. There are two methods for implementing this allocation.
a. Chained allocation
b. Indexed allocation

a. Chained/Linked allocation:

- Linked allocation solves all problems of contiguous allocation.
- In linked list allocation, each file is considered as the linked list of disk blocks.
- However, the disks blocks allocated to a particular file need not to be contiguous on the disk.

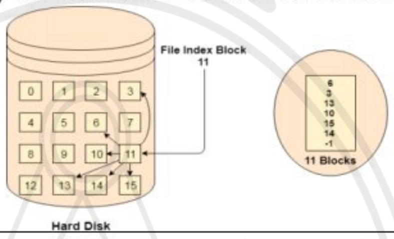
- Each disk block allocated to a file contains a pointer which points to the next disk block allocated to the same file.



Linked List Allocation

b. Indexed allocation

- Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed block.
- Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file.
- Directory entry will only contain the index block address.



Q14. What is VDU? Difference between Dumb Terminal and Intelligent Terminal

A computer terminal means combination of input and output.

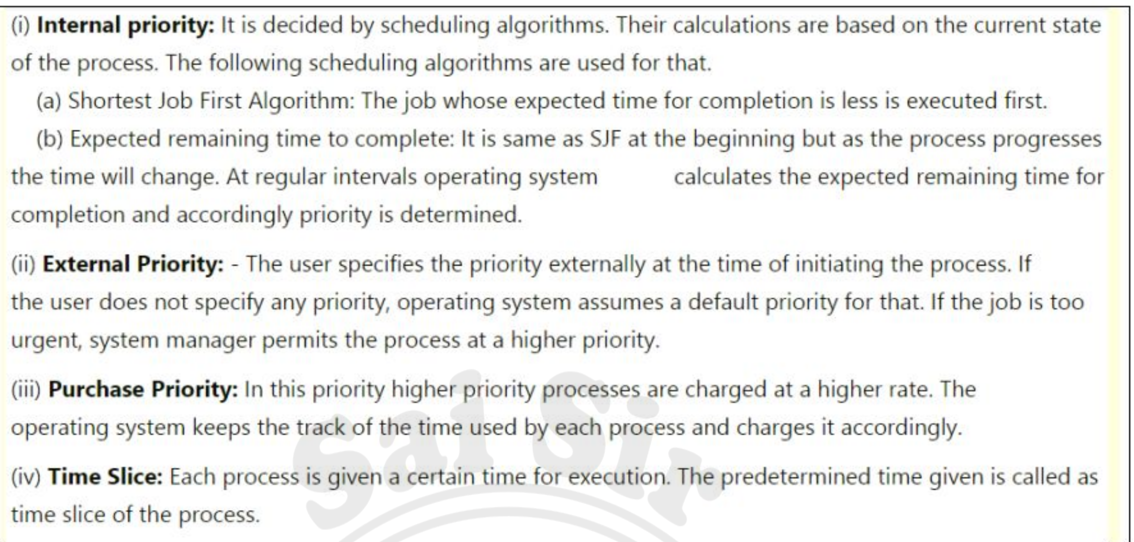
In general term keyboard + VDU are known as computer terminal. There can be so many types of terminal Dumb Terminal, Intelligent Terminal, Smart Terminal and Remote Job Terminal etc.

Dumb Terminal	**Intelligent Terminal**
Dumb Terminal is Input and Output Device only uses for Input and output operation only.	Intelligent Terminals are not only input and output devices but they are used for processing purposes also.
These terminals are used for data entry only.	These terminals are used for processing the data also.
These terminal does not have their own storage capacity etc.	These terminals have their own storage capacity.
These terminals uses the processing power of central computer only i.e. does not have their own processing power or CPU.	These terminals have their own processing power i.e. these terminals have local processing power or CPU.
These are low cost terminals.	These Terminals are costly.
These terminals are used with minicomputers and mainframe as input and output devices.	These terminals are used as standalone computers as well as well with minicomputer and mainframe computers also

Q15.	**Explain the use of video RAM. Explain data bytes and attribute bytes**
	• Stands for "Video Random Access Memory" and is pronounced "V-RAM." • System RAM is great for loading and running programs, but when you need graphics power, VRAM is used. • This is the memory used to store image data that the computer displays; it acts as a buffer between the CPU and the video card. • When a picture is to be displayed on the screen, the image is first read by the processor and then written to the VRAM. • The data is then converted by a RAM digital-to-analog converter (RAMDAC) into analog signals that are sent to the display. • VRAM consists of 2000 data bytes provided by 2000 corresponding attribute bytes. **Data bytes:** All 2000 characters are stored in VRAM. To display any specific character on screen at a specific position all ASCII or EBCDIC code for that character is to move in video RAM. **Attribute bytes:** There is one attribute byte for each data byte. This byte tells the video controller how the character is to be displayed.
TOPIC:PROCESS MANAGEMENT	
Q16.	**What is a process?**
	• A process is basically a program in execution. The execution of a process must progress in a sequential fashion. • To put it in simple terms, 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.
Q17.	**Explain the various states of a process.** **OR** **Discuss various process states with examples** **OR** **Explain Running, Ready and blocked process states with examples.**
	• A process is basically a program in execution. The execution of a process must progress in a sequential fashion. • To put it in simple terms, 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. **1. New** A program which is going to be picked up by the OS into the main memory is called a new process**.** **2. Ready** Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned. The OS picks the new processes from the secondary

	memory and put all of them in the main memory. The processes which are ready for the execution and reside in the main memory are called ready state processes.. **3. Running** One of the processes from the ready state will be chosen by the OS for execution**.** **4. Block or wait** From the Running state, a process can go to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process. When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes. **5. Completion or termination** When a process finishes its execution, it comes in the termination state.
Q18.	**Explain Context switching at process level in multiprogramming system.**
	• A context switching is a process that involves switching of the CPU from one process or task to another. In this phenomenon, the execution of the process that is present in the running state is suspended by the kernel and another process that is present in the ready state is executed by the CPU. • It is one of the essential features of the multitasking operating system.
Q19.	**Write a note on process scheduling** **OR** **What objectives are considered while scheduling process? Explain.**
	• 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. • **While scheduling various processes, the operating system have to choose a set of objective to be achieved.** • FAIRNESS: Fairness refer to being fair to every user in terms of CPU time. • CPU UTILIZATION: CPU utilization is the fraction of the time that the CPU is busy. • TURNAROUND TIME: Elapsed time between time a program or job is submitted and time when it is completed. • RESPONSE TIME: Time to respond with an answer or result to a question or an event.

Q20.	**Explain the term Multitasking with a suitable example.**
	• A process can consist of multiple task, which can run simultaneously. The multiple task should be able to run concurrently within a process. • Multitasking allows programmer flexibility and improves CPU utilization. E.g. Task 0: Read a record process a record Task 0 end Task 1: write a Record Task 1 end
Q21.	**Explain the following:** **a) Preemptive Scheduling b) Non - Preemptive Scheduling**
	Preemptive Scheduling: • Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. • The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. • That process stays in ready queue till it gets next chance to execute. **Non-Preemptive Scheduling:** • Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. • In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. • In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution. • Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.
Q22.	**Explain the following terms related to PROCESS management.** **a) Internal priority** **b) External priority** **c) Purchased priority** **d) Time slice.**



(i) **Internal priority:** It is decided by scheduling algorithms. Their calculations are based on the current state of the process. The following scheduling algorithms are used for that.

(a) Shortest Job First Algorithm: The job whose expected time for completion is less is executed first.

(b) Expected remaining time to complete: It is same as SJF at the beginning but as the process progresses the time will change. At regular intervals operating system calculates the expected remaining time for completion and accordingly priority is determined.

(ii) **External Priority:** - The user specifies the priority externally at the time of initiating the process. If the user does not specify any priority, operating system assumes a default priority for that. If the job is too urgent, system manager permits the process at a higher priority.

(iii) **Purchase Priority:** In this priority higher priority processes are charged at a higher rate. The operating system keeps the track of the time used by each process and charges it accordingly.

(iv) **Time Slice:** Each process is given a certain time for execution. The predetermined time given is called as time slice of the process.

Q23. **Explain the following terms in case of process scheduling**
a) Turnaround time b)Throughput
c) Waiting time d)Response time

Turnaround time:
Turnaround time is the total amount of time spent by the process from coming in the ready state for the first time to its completion.
Throughput:
Throughput is a way to find the efficiency of a CPU. It can be defined as the number of processes executed by the CPU in a given amount of time
Waiting time:
Waiting time is the total time spent by the process in the ready state waiting for CPU.
Response time:
Response time is the time spent when the process is in the ready state and gets the CPU for the first time.

Q24. **Explain the concept of Time-sharing operating system.**

Time-sharing operating systems:

- Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time.
- Time-sharing or multitasking is a logical extension of multiprogramming.
- Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

Advantages of Timesharing operating systems are as follows −

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

TOPIC: MEMORY MANAGEMENT

Q25. What functions are performed by memory management of OS? State any four memory management systems.

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.

Some of the system calls in MM are:

- Allocating a chunk of memory to a process,
- Deallocating a chunk of memory from the process.

The following are the memory management systems:
A) Contiguous, Real memory management system:
a) Single contiguous b) Fixed partitioned
c) Variable partitioned

B) Non - Contiguous, Real memory management system:
a) Paging b)segmentation
c) Combined

C) Non – contiguous, Virtual Memory management system:
a) Virtual memory

Q26. What is Paging? Explain in detail

OR

Explain page memory management system with a suitable page map table(PMT).

- In Operating Systems, Paging is a storage mechanism used ***to retrieve processes from the secondary storage into the main memory*** in the form of pages.
- The main idea behind the paging is to divide each process in the form of pages.
- The main memory will also be divided in the form of frames.
- One page of the process is to be stored in one of the frames of the memory.
- The pages can be stored at the different locations of the memory but the priority is always to find the contiguous frames or holes
- *Here the size of page is same as size of page frame, so that a page can exactly fit into a page frame and it can be assigned to any page frame which is free.*
- Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage.
- Different operating system defines different frame sizes. The sizes of each frame must be equal.

0
1
2
3

Logical Memory Pages

0	**1**
1	**3**
2	**5**
3	**6**

Page table

Physical frames memory

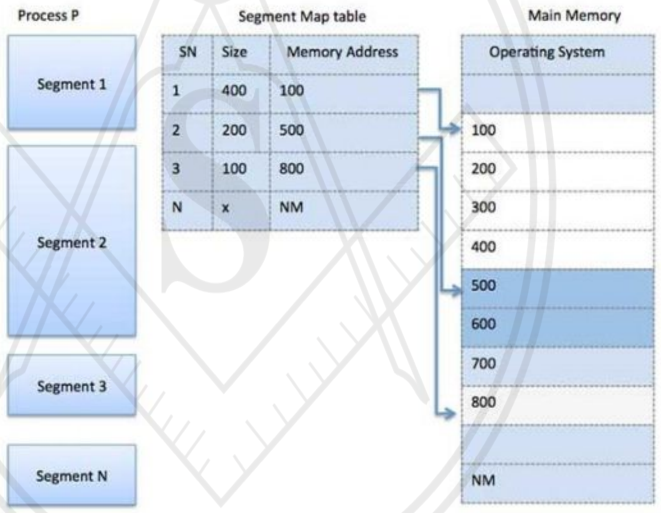
0	
1	Page 0
2	
3	Page 1
4	
5	Page 2

6	Page 6
7	
8	

Q27. **What is segmentation.**

Segmentation

- Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions.
- Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.
- The operating system maintains a segment map table for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory.



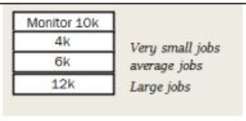
Q28. **What is Partitioning? Explain fixed partitioning and variable partitioning.**

- Partitioning means dividing main memory into various sections. These sections are called partitions.

Fixed Partitioning:

- In this technique, the main memory is divided into partitions of equal or different sizes.
- The operating system always resides in the first partition while the other partitions can be used to store user processes. The memory is assigned to the processes in contiguous way.
- These partitions could be of different sizes, but once decided could not be changed. Hence these partitions are called as FIXED PARTITIONS.

E.g. A memory of 32k words might be divided into regions



- Here jobs enter a system, they are put in job queue. The job scheduler takes into account the memory requirement of each job and allocates the memory.
- When any of the job terminates it releases the memory later which job scheduler may fill with another job from the job queue.

Variable Partitioning.

- The fixed partition suffers from the problem of internal fragmentation. It also limits the degree of multiprogramming and CPU utilization.
- Variable partitions overcome this problems. The number of partitions and their sizes are variable. They are not defined at the time of system generation
- Any partition can be free or allocated to some process. In variable partition the starting address of any partition is not fixed.

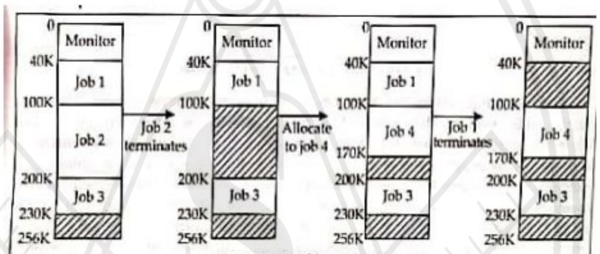


Fig. 1.15 : Variable partitions.

Q29. **What is virtual memory? Explain the following terms with respect to virtual memory.**
a) Page fault b) Dirty bit
OR
Explain the concept of virtual memory. Explain any two terms used in virtual memory

Virtual Memory

- Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory.
- This is done by treating a part of secondary memory as the main memory.
- In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.
- Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

Advantages of Virtual Memory:

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.

3. There is no need to buy more memory RAMs.

Disadvantages of Virtual Memory:

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

Some terms related to virtual memory system are as follows:

1. Locality of reference:

This gives some basis to forecast whether a page is likely to be referenced in the near future based on its passed behavior and whether you could throw that page out to make room for a new one. Clustering of page references in a certain time zone is called principle of Locality of reference.

2. Page Fault:

When a process is executing with only a few pages in memory, and when an instruction is encountered which refers to any instruction or data in some other page, which is outside the memory, a page fault occurs.

3. Page replacement policy:

As no. of processes and no. of pages in main memory for each process increase; at some point of time, all page frames become occupied. At this time, if a new page is to be brought in, the OS has to overwrite some existing page in memory. The page to be replaced is selected by page replacement policy.

TOPIC NAME:GUI

Q30. **What is GUI. Explain the components of GUI.**

OR

Explain the following with respect to GUI.

a) scroll bar b) title bar c)Minimize

OR

Explain in short the function of scroll bar and menu bar components of GUI.

OR

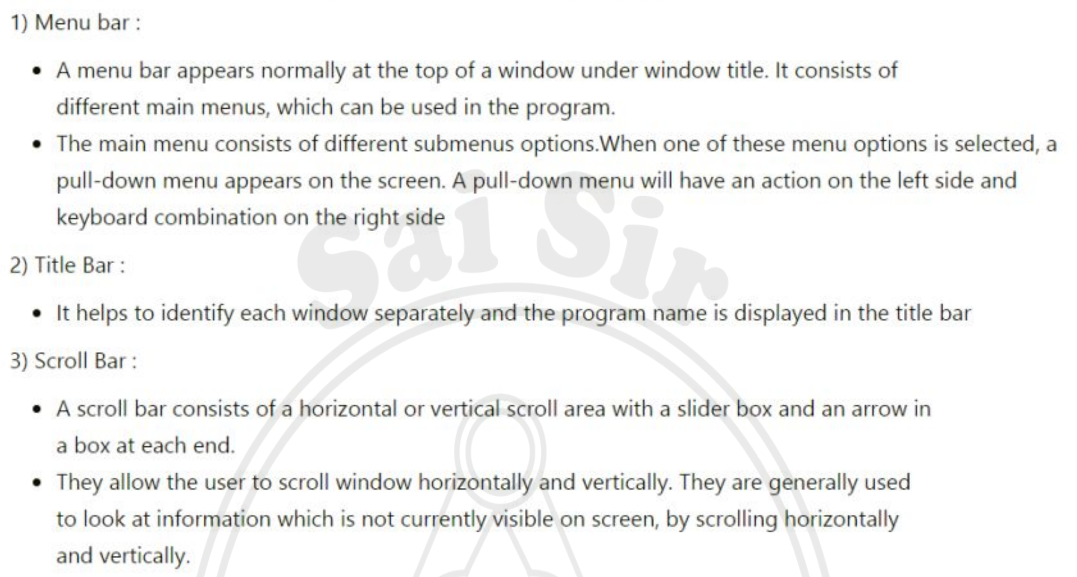
What is GUI? Write any two features of GUI.

The aim of a GUI is to provide an interface that is:

- Easy to learn.
- Consistent across applications (needs learning only once).
- Easy to use.

The GUI encourages users to concentrate on their tasks, rather than on the functions provided by the application. A GUI is intuitive, and encourages users to experiment and explore. It also forgives mistakes. This means that users can learn in their own way and at their own pace.

Following are the different componenets of a GUI:



1) Menu bar :

- A menu bar appears normally at the top of a window under window title. It consists of different main menus, which can be used in the program.
- The main menu consists of different submenus options.When one of these menu options is selected, a pull-down menu appears on the screen. A pull-down menu will have an action on the left side and keyboard combination on the right side

2) Title Bar :

- It helps to identify each window separately and the program name is displayed in the title bar

3) Scroll Bar :

- A scroll bar consists of a horizontal or vertical scroll area with a slider box and an arrow in a box at each end.
- They allow the user to scroll window horizontally and vertically. They are generally used to look at information which is not currently visible on screen, by scrolling horizontally and vertically.

4) Minimize:
The Minimize button shrinks the window and places it on the taskbar while leaving the program running.
5)Maximize
The Maximize button, which looks like a small window, is used to enlarge a window to cover the entire desktop. After a window is maximized, the Maximize button changes to the Restore button.

Q31. State any four advantages of GUI.

The GUI encourages users to concentrate on their tasks, rather than on the functions provided by the application. A GUI is intuitive, and encourages users to experiment and explore. It also forgives mistakes. This means that users can learn in their own way and at their own pace.

Advantages:

- ❑ ***Easy to use consistent GUI for virtually all programs.***
- ❑ ***User can communicate and exchange data between programs without transferring or copying files.***
- ❑ ***With GUI, commands are replaced by graphics. Hence it is not necessary to remember commands and its meaning***
- ❑ ***User can run several program simultaneously.***

TOPIC:SECURITY ASPECTS OF OS

Q32. Explain Virus detection, removal and prevention.
OR
Discuss Virus detection, removal and prevention philosophies.

(1) Virus detection

(i) Normally virus detection program checks integrity of binary files. It maintains a checksum on each file. At regular frequency detection program calculates checksum and matches with original one. If there is mismatch then that program may be infected.

(ii) Some programs reside in the memory and continuously monitor memory and I/O operations against virus.

(2) Virus removal

There are some viruses whose bit pattern in the code can be predicted. The virus removal program scans the disk for the patterns of known viruses and on detection it removes them.

(3) Virus prevention

For prevention of virus the user can take the following precautions

(i) Always buy legal copies of software.

(ii) Take frequent backups of data

(iii) Run monitor programs frequently to detect virus.

Q33. What is a computer virus? What are the different methods by which virus can infect other programs.

Computer Virus:

- It is program segment written with clear intention of infecting other programs. It is not a standalone program.
- It cannot operate independently and spreads by inserting its virus code to multiply itself by altering the programs and applications. Thus causing direct harm to system by corrupting code and data.

Infection Methods:

(a) Append: In this method viral code appends itself to the unaffected program.

(b) Replace: In this method viral code replaces original executable program completely or partially.

(c) Insert: In this method the viral code is inserted in the body of an executable code to carry out some funny actions.

(d) Delete: In this case viral code deletes some codes from executable program.

(e) Redirect: In this case the normal control flow of a program is changed to execute some other code.

Q34. Difference between computer worm and virus

No.	Criteria	Computer Worm	Computer Virus
1	Definition	A computer worm is a complete program	A computer virus is not a complete program , but a part of a program
2	Action	A computer worm act independently	A computer virus cannot act independently.
3	Corruption	It does not cause direct harm to computer system. It just goes on spreading on to network	It causes direct harm to the computer system, corrupts code as well as data.

Q35. Define security with respect to an OS. Explain the different elements of security.

- In multiuser enviornment, the concepts of security and protection of important as user programs have to be prevented by interfacing with each other and each user's memory must be protected.

- Security is defined as - “Secure systems will control, through use of specific security features, access to information that only authorized individuals or processes operating on their behalf will have access to read, write, create or delete.“
- **Confidentiality**: Ensuring that the information is not accessed in an unauthorized manner i.e. by controlling read operations.
- **Integrity**: Ensuring that the information is not amended or deleted in an unauthorized manner. i.e. by controlling write operations
- **Availability**: Ensuring that they information is available to authorized users at there right time. i.e. By controlling read and delete operations and ensuring fault recovery.

Q36. What are attacks on security? Explain in short the ways in which a system can be attacked.

- ❑ Authentication:
- ❑ Browsing
- ❑ Tap doors
- ❑ Invalid parameters
- ❑ Line trapping
- ❑ Electric data capture
- ❑ Lost line
- ❑ Improper access control
- ❑ Waste recovery
- ❑ Rouge software

Q37. What are computer worms? Explain its mode of operation

- ❑ A computer worm is a standalone computer program that replicates itself in order to spread to other computers over a network. While doing this it consumes network resources to a large extent and can bring network to halt.
- ❑ Worms are designed only to spread, and do not harm other program thereby consuming large resources such as transmission capacity, disk storage and denying services to others.
- ❑ Worm is harmful to organizations' operation over a network and can be protected by strong security and various check points in communication system.

Q38. What is a computer virus? State various types of viruses and the basis on which they are classified.

Computer Virus:

- ❑ It is program segment written with clear intention of infecting other programs. It is not a standalone program.
- ❑ It cannot operate independently and spreads by inserting its virus code to multiply itself by altering the programs and applications. Thus causing direct harm to system by corrupting code and data.

Different types of viruses:

- ❑ Boot sector virus: It gets into the system memory of machine is booted with an infected floppy or hard disk.
- ❑ Memory resident infectors: It loads upon execution of an infected file and then if a non infected file is executed, the virus infects it.

	❑ Filter specific infectors: Infection occurs when an infected file is executed. Then virus loads it's vital code into memory and also reboots the system. ❑ Command processor infectors: It infects com files. Size of virus is 205 bytes and so infected files size increases by 205 bytes. Infection occurs when infected file is executed. ❑ **General purpose infectors:** It infects exe files
	**********

2. DATA STRUCTURES

- Data are simply value or set of values.
- A data item refers to a single unit of values.
- Data items that are divided into sub items are called "group items"; those that are not divided are called "elementary items". E.g.: name can be divided into sub items – first name, middle name & last name but the Pincode number can not be divided further so it is treated as a single item.
- An entity is something that has certain attributes or properties, which may be assigned values.
  Example:

Attributes:	Name	Age	Sex	Pincode
Values:	Ram	18	M	1423

- Entities with similar attributes (e.g. all the employees in an organization) form an entity set.
- The term "information" is sometimes used for data with given attributes, or, in other words, meaningful or process data.
- The way that data is organized into the hierarchy of fields, records & files reflects the relation between attributes, entities and entity sets.
- That is, a field is elementary unit of information representing an attribute of an entity, a record is the collection of field values of a given entity & a file is the collection of records of the entities in a given entity set.

Q. What is Data Structure? What are its different types?

Ans. Data may be organized in many different ways; the logical or mathematical model of a particular organization of data is called a data structure.

There are many types of data structure. They are as follows:

1) Arrays:

   The simplest type of structure is a linear (or one dimensional) array. By a linear array, we simply mean a list of finite number n of similar data element under same name. If we choose the name 'A' for the array, then the elements of 'A' are denoted by:

   subscript notation -.............................A_1, A_2,……… ,A_n

   or by the parenthesis notation......A[1], A[2], ... ,A[n]

   or by the bracket notation...............A(1), A(2), ... ,A(n)

   The number k in A[k] is called a subscript and A[k] is called a subscripted variable.

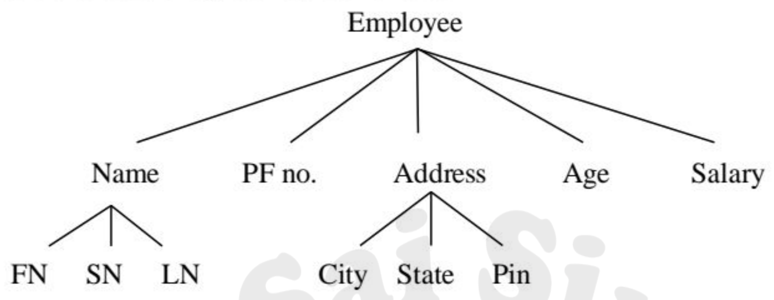
2) Linked List:

   A linked list, or one-way list, is a linear collection of data elements, called nodes where the linear order is given by means of pointers. That is, each node is divided into two parts: the first part contains the information of the element, and the second part, called the link field or next pointer field, contains the address of the next node in the list.

3) Record Structure:

   A file may be maintained by means of one or more arrays or records, where one indicates both the group items and the elementary items. It can also be described best by means of a tree structure. For example, an employee personnel record may contain the following data items: P.P.F. no, name, address, age and salary.

The record structure is presented as follows:



Hierarchical representation Fig. (a)

Or

```
01. Employee
   02. Name
      03.FN
      03. SN
      03. LN
   02. PF No
   02. Address
      03. City
      03. State
      03. Pin
   02. Age
   02. Salary
```

Level no. representation (Fig a)

A linear array is a list of finite number "n" of homogenous data elements (data elements of same data) such that:

a) The elements of array are referred respectively by an index set consisting of "n" consecutive number.

b) The elements of array are stored respectively in successive memory locations.

Q. Explain One Dimensional Array. How it is represented in memory?

Ans. The number 'n' of elements is called the length or size of the array. The length or the number of data elements of the arrays can be obtained from the index set by the formula.

$$\boxed{\text{Length} = UB - LB + 1}$$

where, UB is the largest index called upper bound.
LB is the smallest index called lower bound.

- The following is an example of one dimensional array:-

Let NAME be a 6-element array of integers such that NAME [1]=311, NAME[2] = 312, NAME[3]= 313, NAME[4] = 75, NAME[5] = 112, NAME[6] = 155

Simply writing as, NAME: 311,311,313,75,112,155

The array NAME can be viewed as

NAME

311	0
312	1
313	2
75	3
112	4
155	5

NAME

0	1	2	3	4	5
311	312	313	75	112	155

Representation of linear arrays in memory:

Let "LA" be a linear array in the memory of computer. We know that the memory of the computer is simply a sequence of addressed locations as pictured below

1000	1001	1002	1003	1004	1005

Loc (LA[k]) = address of the element LA[k] of the array LA. As previously noted the elements of LA are stored in successive memory cells. Accordingly, the computer does not need to keep the track of the address of every element of LA, but needs of keep tracks only of the address of the first elements of LA, denoted by.

Base (LA) i.e., starting address of array

Using this address Base (LA), the computer calculates the address of any elements of LA by the following formula:

LOC(LA[k]) = Base(LA) + W(k-lower bound)

where W is the number of words per memory cells for the array LA.

- **TRAVERSING LINEAR ARRAYS:**

The following algorithm traverses a linear array LA. The simplicity of the algorithm comes from the fact that LA is a linear structure. Other linear structures, such as linked lists, can also be easily traversed. On the other hand, the traversal of nonlinear structure, such as trees and graphs, is considerably more complicated.

Algo : (Traversing a linear Array)

Suppose LA is a linear array with lower bound LB and upper bound UB. This algorithm traverses LA applying an operation PROCESS to each elements of LA.

```
1. Set K : = LB                          // Initialize counter
2. Repeat Steps 3 and 4 while K<=UB
3. Apply PROCESS to LA[k].               // Visit element
4. Set K : = K+ 1.                       // Increase counter
     [End of Step 2 loop]
5. Exit.
```

We also state an alternative form of the algorithm, which uses a repeat-for loop instead of the repeat-while loop.

Algo using 'for' loop:- (Traversing a Linear Array) this algorithm traverses a linear array LA with lower bound LB and upper bound UB.

```
1. Repeat for K = LB to UB:
    Apply PROCESS to LA[k]
    [End of loop]
2. Exit.
```

- **INSERTING AND DELETING element :**

Let LA be a collection of data elements in the memory of the computer. "Inserting" refers to the operation of adding one more element to the collection LA, and "deleting" refers to the operation of removing one of the elements from LA. This section discusses inserting and deleting when LA is a linear array.

Inserting an elements at the "end" of the linear array can be easily done provide the memory space allocated for the array is large enough to accommodate the additional element. On the other hand, suppose we need to insert an element in the middle of the array. Then, on the average, half of the elements must be moved downward to new locations to accommodate the new element and keep the order of the other elements.

Similarly, deleting an element at the "end" of an array presents no difficulties, but deleting an element somewhere in the middle of the array would require that each subsequent element be moved one location upward in order to "fill up" the array.

Algorithm: (**Inserting** into a Linear Array)

INSERT(LA, N,K,ITEM)

Here LA is a linear array with N elements and K is a positive integer such that K <= N.
This algorithm inserts an elements ITEM into the Kth position in LA.

```
1. Set J = N                                //Initialize countes
2. Repeat Steps 3 and 4 while J >= K.
3. Set                                      // Move J^th element downward
   LA[J+1]: = LA[J]
4. Set J : =J-1.                            // Decrease counter
   [End of step 2 loop.]
5. Set LA [k] : = ITEM.                     // Insert element
6. Set N: = N+1                             // Reset N
7. Exit.
```

The following algorithm deletes the Kth element from a linear array LA and assigns it to a variable ITEM.

<u>Algorithm</u>: (**Deleting** from Linear Array)

```
DELETE (LA,N,K,ITEM)
Here LA is a linear array with N elements and K is a positive integer such that K<=N.
This algorithm deletes the K^th elements from LA.
1. Set ITEM: = LA[K].
2. Repeat for J = K to N-1.
   Move J+1^st element upward
   Set LA [J] := LA[J+1]
   [End of loop.]
3. Set N :=N-1                              Reset the number N of elements in LA
4. Exit.
```

Q. Explain bubble sort algorithm with the help of example.

Ans. *Sorting* means the operation of rearranging the elements of A (Let A be a list of n numbers) so they are in increasing order. That is,

$A[0]<A[1]<A[2]< \ldots\ldots\ldots\ldots <A[N]$.

Example: 5,4,7,9,2,3; the sorted elements are 2,3,4,5,7,9

<u>Bubble sort:</u>

Suppose the list of numbers A[1], A[2],.............A[N] is in memory. The bubble sort algorithmic works as follows:

<u>Step1:</u> Compare A[1] and A[2] and arrange them in the desired order, so that A[1]<A[2]. Then compare A[2] and A[3] & arrange them so that A[2]<A[3]. Then compare A[3] and A[4] and arrange them so that A[3]<A[4]. Continue until we compare A[N-1] with A[N] and arrange them A[N-1]<A[N].

Note: During step 1, the largest element is "<u>bubbled up</u>" to the n^th position or "<u>sinks</u>" to the n^th position.

<u>Step2:</u> Repeat step 1 with one less comparison, that is, now we stop after we compare and possible rearrange A[N-2] and A[N-1].

<u>Step3:</u> Repeat step 1 with two fewer comparisons, that is, we stop after we compare and possible rearrange A[N-3] and A[N-2].

....

....

<u>Step N-1:</u> Compare A[1] with A[2] and arrange them so that A[1]<A[2].

The process of sequentially traversing through <u>all or part of a list</u> is frequently called a "pass" so each of the above step is called a <u>pass</u>.

Example:

Suppose the following numbers are stored in an array A

29,49,25,99,77,19,03,61

Now, we will apply bubble sort to the array

<u>Pass1</u>:

(a) Compare A_1 and A_2. Since 29<49, the list is not altered.
Compare A_2 and A_3. Since 49>25, interchange the nos.

29,25,49,99,77,19,03,61

(b) Compare A_3 and A_4. Since 49<99, the list is not altered.

(c) Compare A_4 and A_5. Since 99>77, interchange the nos.

29,25,49,77,99,19,03,61

(d) Compare A_5 and A_6. Since 99>19, interchange the nos.

29,25,49,77,19,99,03,61

(e) Compare A_6 and A_7. Since 99>03, interchange th nos.

29,25,49,77,19,03,99,61

(f) Compare A_7 and A_8. Since 99>61, interchange the nos.

29,25,49,77,19,03,61,99

At the end of pass 1, the largest number 99, has moved to the last position. However the rest of the numbers are not sorted, even though some of them has changed their positions.

<u>Pass2:</u> 29,25,49,77,19,03,61,99

25,29,49,77,19,03,61,99

25,29,49,77,19,03,61,99

25,29,49,19,77, 03,61,99

25,29,49,19,03, 77,61,99

25,29,49,19,03, 61,77,99

At the end of pass 2, the second largest number, 77, has moved to its way down next to the last position.

<u>Pass3:</u> 25,29,49,19,03, 61,77,99

25,29,19,03,49, 61,77,99

<u>Pass4:</u> 25,29,19,03,49, 61,77,99

25,19,03,29,49, 61,77,99

<u>Pass5:</u> 25,19,03,29,49, 61,77,99

19,03,25,29,49, 61,77,99

<u>Pass6:</u> 03,19,25,29,49, 61,77,99

<u>Pass7:</u> Finally, there is no comparison now, as the elements are sorted in the array A.

Algorithm: Bubble sort

```
    Sort (DATA , N)
    Here DATA is an array with N elements. This algorithm sorts the elements in DATA.
1. Repeat step 2 and 3 for K = 1 to N-1.
2. Set PTR:=1                  //Initialize pass pointer PTR
3. Repeat while PTR ≤ N-K.     //Execute pass
   a) If DATA [PTR]> DATA [PTR+1],then
         Interchange DATA [PTR] and DATA [PTR+1]
      [End of If structure]
   b) Set PTR:= PTR+1.
      [End of inner loop.]
      [End of step 1 outer loop]
4. Exit.
```

There is an inner loop, which is controlled by the variable PTR and the loop is contained in an outer loop, which is controlled by an index K.

Q. Explain Binary Search algorithm with the help of example.

Ans. Algorithm: Binary Search

Binary (DATA, LB, UB, ITEM, LOC)

Here DATA is a sorted array with lower bound LB and upper bound UB, and ITEM is a given item of information. The variables BEG, END and MID, respectively, denotes the beginning, end and middle locations of a segment of element of DATA. This algorithm finds the location LOC of ITEM in DATA or sets LOC = NULL.

```
1. [Initialize variables.]
   Set BEG=LB, END=UB and MID=INT(BEG+END/2)
2. Repeat step 3 & 4 [while BEG ≤ END and DATA [MID] ≠ ITEM]
3. If ITEM < DATA [MID],
   then: Set END=MID-1.
   Else:
   Set BEG=MID+1.
     [End of If structure]
4. Set MID=INT((BEG+END)/2)
   [End of step 2 loop]
5. If DATA [MID]= =ITEM, then:
   Set LOC=MID
   Else:
   Set LOC=NULL
   [End of If structure]
6. Exit.
```

Example:

L et DATA be the following sorted 11 element array:
DATA: 07 ,19,22,43,55,59,65,70,77,88,93 apply the binary search to DATA for different values of ITEM.

Suppose we are searching ITEM=65
So, it will be like
Initially, BEG=1 and END=11. Hence
MID = INT(1+11)/2=6 & so DATA[MID]=59
Since 65<59, BEG has its value changed by
BEG=MID+1 =7. Hence,
MID=INT[(7+11)/2] =9 &
Now DATA[MID]= 77

Since 65>77, END has its value changed by
END=MID-1 = Hence MID=INT [(7+8)/2] =7 &
now DATA[MID] =65
We have found ITEM in location LOC=MID=7.

Q) Write an algorithm for linear search technique with suitable example.

Ans. Algorithm : Linear Search

LINEAR (DATA, N, ITEM, LOC)

Here DATA is a linear array with N elements and ITEM is given element. This algorithm finds the location LOC of ITEM in DATA or sets LOC = 0, if search is unsuccessful.

```
Step 1 :    [Insert ITEM at the end of DATA]
            Set DATA[N+1] : = ITEM
Step 2 :    [Initialize counter]
            Set LOC : = 1
Step 3 :    [Search for item]
            Repeat While DATA[LOC] ≠ ITEM :
            Set LOC = LOC + 1
            [End of loop]
Step 4:     If LOC = N + 1, then :
            Set LOC : = 0
Step 5 :    Exit
```

For example : Given DATA array with following 5 elements

09 72 45 69 11

Suppose ITEM = 69

Step 1 : Set DATA [6] = 69, ∴ List becomes

09 72 45 69 11 69

Step 2 : LOC = 1

Step 3 : Since DATA [1] = 09 ≠ 69 ∴ LOC = 2

Since DATA [2] = 72 ≠ 69 ∴ LOC = 3

Since DATA [2] = 45 ≠ 69 ∴ LOC = 4

Here DATA [3] = 69 = 69 = ITEM

Step 4 : Hence ITEM = 69 found at position, LOC = 4.

Q) Write difference between Linear search and Binary search.

Ans.

Linear Search	**Binary Search**
1. Linear search performs on unsorted list of elements as well as sorted list.	1.For binary search, the elements in array are stored in alphabetically or numerically in sorted manner.
2. Compare the desired element with all elements in an array until the match is found.	2. Compare the value of midpoint with desired value. If the value is greater than midpoint value the first half is checked, otherwise second half is checked until search is successful or interval is empty.
3. Insertion of an element in an array can be performed very efficiently when array is not ordered.	3. An insertion of a new element requires that many elements be physically moved to preserved order.
4. For large size of array, time required for this search is very large.	4. For large size of array, comparatively time required is less.
5. `Time complexity is as follows : worst case : N comparison Best case : 1 comparison	5. time complexity as follows : worst case : $\log^2 N$ comparison Best case : 1 comparison

Arrays, Records and Pointers

INTRODUCTION:

Data structures are classified as either linear data structure or non-linear data structure.

- **Linear data structure-**

A data structure is said to be linear if its elements form a sequence, or in other words, a linear list. There are two basic ways of representing such linear structure in memory.

1) The first way is to have the linear relationship between the elements of sequential memory locations. These linear structures are called as arrays.
2) The second way is to have the linear relationship between the elements represented by means of pointers or links. These linear structure are called linked lists.

- **Non-linear data structure-**

This structure is mainly used to represent data containing a hierarchical relationship between elements, that is, records, family trees and table of contents such as trees and graphs.

Linear arrays:

Definition : An array can be defined as the collection of the sequential memory locations, which can be referred to a single name along with a number, known as index, to access a particular field or data. When the elements need to be referred by more than one index, it is known as Multidimensional arrays.

There are data structures other than arrays, linked lists and trees. Some of these structures are as follows:

a) **Stack:** A stack, also called a last-in-first-out (LIFO) system, is a linear list in which insertion and deletion can take place only at one end, called the top. This structure is similar in its operation to a stack of dishes or glasses placed at one above another and you can take out the dishes or glass only from one end, that is , top.

b) **Queue**: A queue, also called a first-in-first –out (FIFO) system, is a linear list in which deletion can take place only at one end of the list "front", and insertions can take place only at the other end of the list "rear". For example, Queue in the bus stop, the first person in the queue gets in the bus first and to join the queue the person has to come in the last.

c) **Graph:** Data sometimes contain a relationship between pairs of elements, which is not hierarchical in nature. For Example, suppose an airline flies only between the cities connected by lines. The data structure that reflects this type of relationship is called a graph.

Q. What are data structure operations ?

Ans. The data appearing in our data structures are processed by means of certain operations. The following are the data structure operations, which are used:

1. Traversing: Accessing each record exactly only once so that certain items in the record may be processed. (This accessing & processing is sometimes called "visiting" the record)

2. Searching: Finding the location of the record with the given key value, or finding the locations of all the records.
3. Inserting: Adding a new record to the structure.
4. Deleting: Removing a record from the structure.
5. Sorting: Arranging the records in some logical order (e.g. alphabetically or numerically).
6. Merging: Combining the records in two different sorted files into single sorted file operations.

Q. What are the different control structures?

Ans. The control structures are as follows:

1. Sequence logic, or sequential flow
2. Selection logic, or conditional flow
3. Iteration logic, or repetitive flow

Explanation:

1. Sequential flow:

It means that the flow of the structure flows in sequence. The sequence can be presented by means of numbered steps or by which the modules are written. Example:

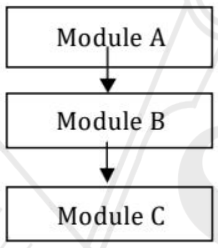


Fig (a) Sequential flow

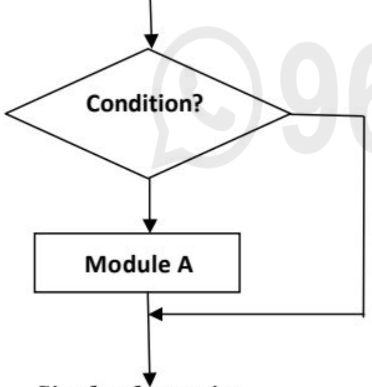
2. Selection logic: (Conditional Flow):

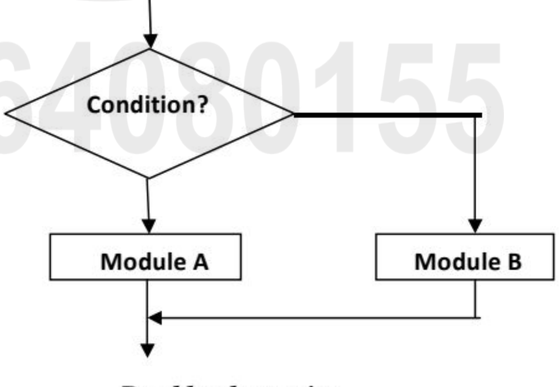
Selection logic employs a number of conditions, which lead to a selection of one out of several alternative modules. The structures which implement this logic are called conditional structures or IF structures. These conditional structures fall into three types. They are

(a) Single alternative: This structure has the form:

```
If condition, then:
[Module A]
[End of If structure]
```

The logic of this structure is pictured as:





Single alternative ***Double alternative***

(b) Double alternative**:** This structure has the form

```
IF condition, then:
[Module A]
Else:
[Module B]
[End of If structure]
```

The logic of this structure is pictured in Fig (b), above.

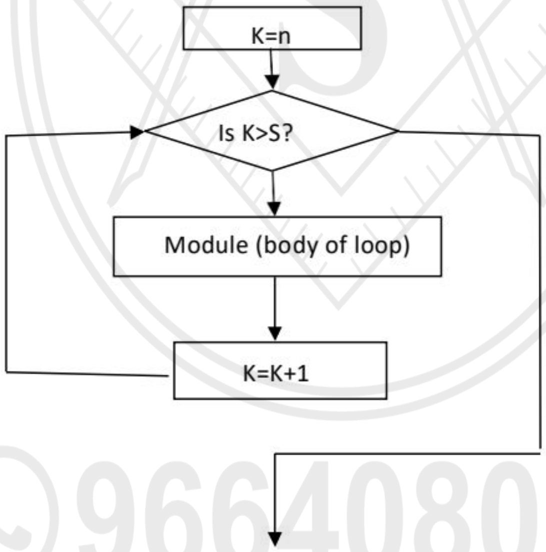
(c) Multiple alternative: This structure has the form

```
If condition (1), then:
[Module A]
Else if condition (2), then :
[Module B]
:
Else:
 [Module C]
[End of If structure]
```

This logic of this structure allows only one of the modules to be executed.

3. Iteration Logic (Repetitive Flow):

This type of logic refers with a repeat statement involving loops. Each type begins with a Repeat statement and is followed by a module, called the *body of the loop*.



Q. What is the Complexity of Algorithms?

Ans.

- An algorithm is a well-defined list of steps for solving a particular task or problem.
- One major purpose of complexity is to develop efficient algorithm for the processing of or data.
- The time and space it uses are two major measures of the efficiency of an algorithm.
  ∴ The complexity of an algorithm is the function which gives the running time / space in terms of the input size.
- The following two complexities are the important factor for complexity of an algorithm.

- Space complexity
- Time complexity

Definition:

(i) **Space complexity**: The space complexity of an algorithm is the amount of memory it needs to run or to complete an algorithm.

(ii) **Time complexity**: The time complexity of an algorithm is the amount of computer time it needs to run or to complete an algorithm.

Q. What is a record? How it differs from a linear array?

Ans. A record is a collection of field or attributes i.e. relative data items.

Collection of data is frequently organized into hierarchy of field i.e. records. A file is nothing but collection of records.

Difference between records and linear arrays :

(i) A record is a collection of fields, while an array is list of homogeneous data elements.

(ii) A record may contain non-homogeneous data i.e. data elements may be of different data types. An array always contains homogeneous data.

(iii) In a record, natural ordering of elements is not possible. Array elements can be naturally ordered.

(iv) Elements of record are referenced by level number, while those of array can be referenced by an index set consisting of n consecutive numbers.

Linked List

ARRAY

- Data processing involves storing and processing data organized into lists.
- One-way store such data is by means of arrays.
- The linear relationship between the data elements of an array is reflected by the physical relationship of the data in memory not by any information contained in the data elements themselves.
- This makes easy to compute the address of an element in an array.
- Disadvantages: It is relatively expensive to insert and delete elements in an array.
- Also, since an array usually occupies a block of memory, one cannot simply double or triple the size of the array.

Link or Pointer

- Another way of storing a list in memory is to have each element in the list contain a field, called list or pointer, which contains the address of the next element in the list.
- It does not occupy the adjacent space in memory, this will make it easier to insert and delete elements in the list.

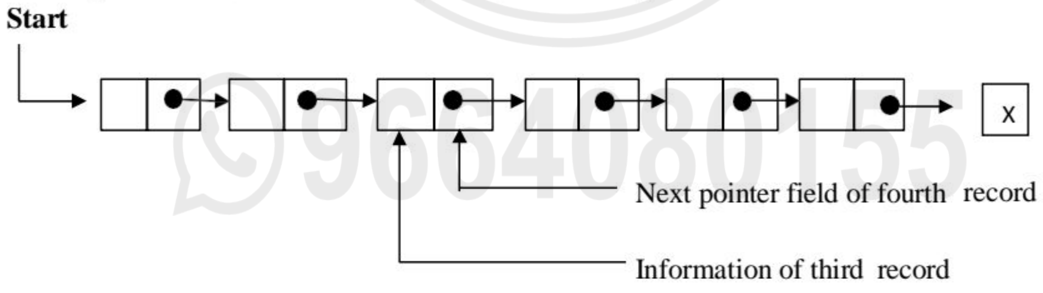
Q. What is Linked list? Explain.

Ans.

A linked list, or *one-way list*, is a linear collection of data elements, called nodes where the linear order is given by means of pointers. That is, each node is divided into two parts: the first part contains the information of the element, and the second part, called the link field or next pointer field, contains the address of the next node in the list.

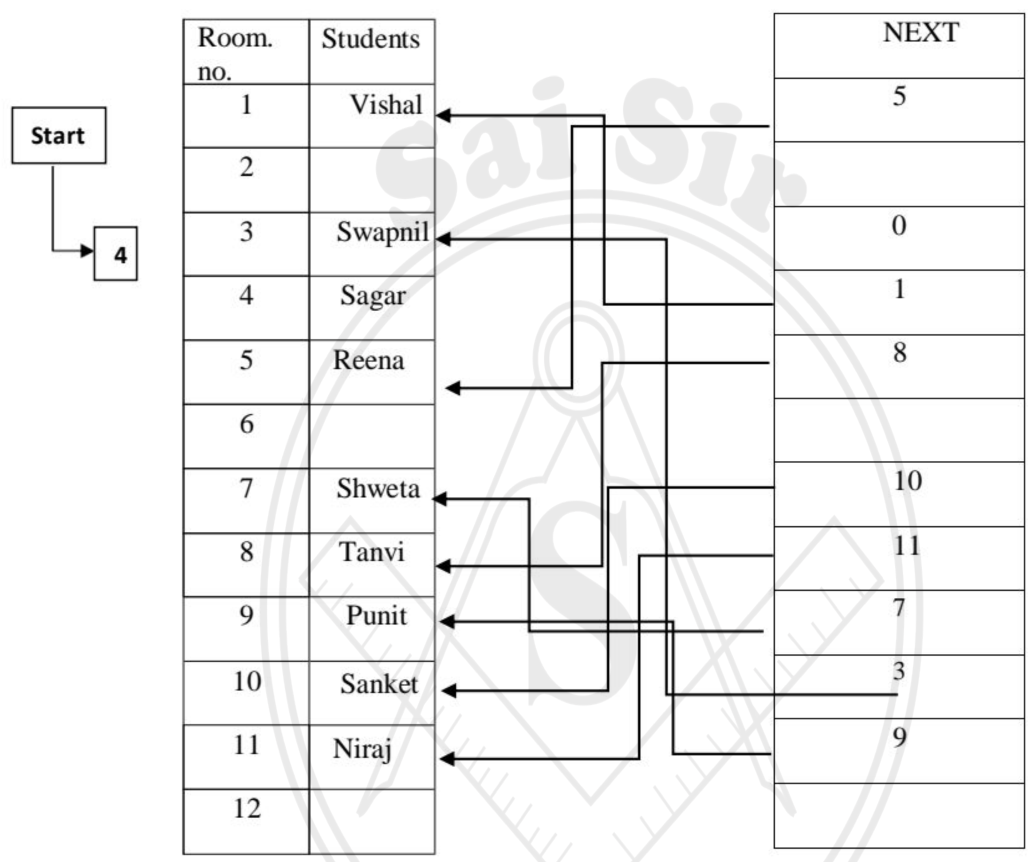
Basic example of linked list:

The diagram below of a linked list contains 6 nodes. Each node contains two parts. The left part represents the information of the present node, which may contain an entire record of data items (e.g. NAME, ADDRESS .....). The right part represents the next pointer field of the node, and there is an arrow drawn from it to the next node in the list. The pointer of the last node contains a special value, called the *null* pointer, which is any invalid address.


Start
x
Next pointer field of fourth record
Information of third record

Example:

A hostel ward contains 12 Rooms, of which 8 are occupied. Suppose we, arrange as listing of the Students who have allotted with the rooms. This list can be given in the pointer field, Called NEXT. We use the variable START to point to the first patient. Hence START contains 4. Here 4 contain the information Sagar & points to room 1. Since 1 contain the student Vishal & Points to Room 5. Since 5 contain the student Reena & points to bed 8 and so on.



Room. no.	Students	NEXT
1	Vishal	5
2		
3	Swapnil	0
4	Sagar	1
5	Reena	8
6		
7	Shweta	10
8	Tanvi	11
9	Punit	7
10	Sanket	3
11	Niraj	9
12		

Q. How linked list is represented in memory?

Ans.

Let LIST be a linked list. LIST requires two linear arrays-called as INFO and LINK (INFO contain the information of the present node and LINK contain the next pointer field of a node). LIST also requires a variable name- such as NAME or START, which contain the location of the beginning of the node and a NULL to indicate the end of the node.

The following example of linked list indicaes that the nodes of a list need not occupy adjacent elements in the arrays INFO and LINK. However, each node has its own pointer variable. From the given example, we can obtain the actual list of characters, or in other words the string.

Refer the above example

START =4, so INFO[4] = Sagar is the first data.
LINK [4]=1, so INFO [1] = Vishal is the second data.
LINK [1] = 5, so INFO [5] = Reena is the third data.
LINK [5] = 8, so INFO [8] = Tanvi is the fourth data.

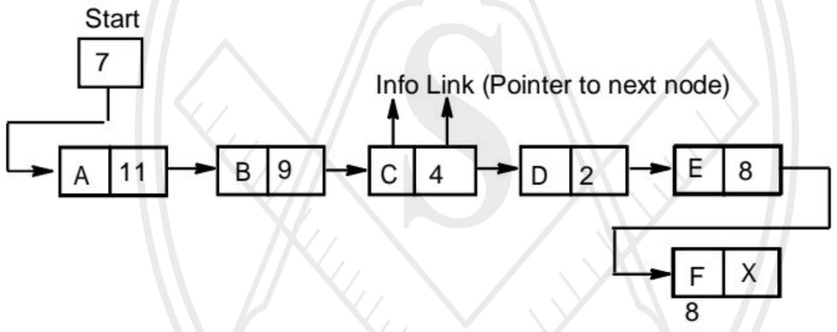
LINK [8] = 11, so INFO [11] = Niraj is the fifth data.
LINK [11] = 9, so INFO [9] = Punit is the sixth data.
LINK[9]= 7, so INFO [7] = Shweta is the seventh data.
LINK [7] = 10, so INFO [10] = Sanket is the Eighth data.
LINK [10] = 3, so INFO [3] = Swapnil is the Ninth data.
LINK [3] = 0, the NULL value, so the list has ended.
In other words, NO EXIT is the data.

Q. What are linked lists? Show a liked list with suitable example having six nodes with a properly labelled diagram.

OR

With suitable example, show labelled diagram for link between two nodes having the information part and next pointer field.

Ans. i) A linked list is a linear collection of data elements, called nodes, where the linear order is maintained with the help of pointers.
ii) Linked list is also called as one-way list.
iii) Each node in the linked list is divided into two parts. First part is called as INFO part, called as LINK part, which is next pointers field i.e., it contains the address of next node in the list.
iv) e.g.



(a) The above figure shows a linked list having six nodes.
(b) The left part of the node is the Info part, which contains information of the element, while the right part is Link part, which is next pointers field i.e., it points to next node.
(c) An arrow is drawn from Link part of one node to the next node to indicate link.
(d) The address of the list is stored in Start or Name of the list.
(e) The Link part of last node is null pointer i.e., it contains nothing.
(f) To trace the linked list, we just require the address of Start or Name.

Binary Tree

- We know the linear types of data structures: strings, arrays, lists, stacks and queues.
- Data structures also define nonlinear data structure called a tree.
- The tree structure is mainly used to represent data containing a hierarchical relationship between elements: example: family trees, records etc.
- Binary tree is a tree, which can be easily maintained in the computer.

Q. What is Binary tree?

Ans. A binary tree T is defined as a finite set of elements, called nodes, such that:

(a) T is empty (called the null tree or empty tree),or

(b) T contains a distinguished node R, called the root of T and the remaining nodes of T form an ordered pair of binary trees T_1 and T_2.

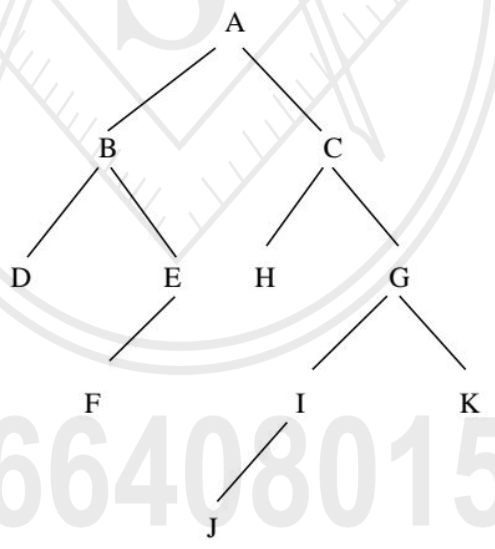
(c) If T is non-empty, then its root is called the left successor of R; similarly if T is non-empty, then its root is called the right successor of R.

A binary tree T is frequently presented by means of a diagram. Specifically, the diagram below represents a binary tree as follows:

(i) T consists of 11 nodes

(ii)The root of T is the node A.

(iii)A left-downward slanted line from a node N indicates a left successor of N, and a right-downward slanted line from N indicates right successor of N. Observe the following tree:-



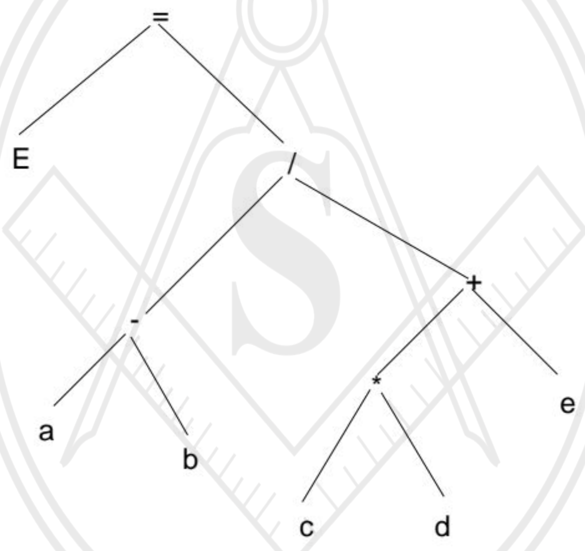
Any node N of a binary tree T has either 0,1,or 2 successors. The nodes A, B,C,G has two successors while the nodes E, I have only one successor and the nodes D,F,J, K have no successors. The nodes with no successors are called terminal nodes.

The above definition of binary tree T is recursive since T is defined in terms of binary tree T and T_1. This means, that every node N of T contains a left and a right subtree. Moreover, if N is a terminal node, then both its left and right subtrees are empty.

Binary trees T and T_1 are said to be similar if they have same structure or shape. The trees are said to be copies if they are similar and if they have same contents at corresponding nodes.

Example of Binary Tree:

Consider any algebraic expression E involving only binary operations, such as
E=(a-b)/((c*d)+e)
The binary tree will be constructed as:

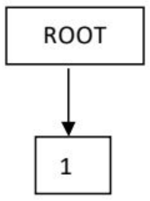


Q. **How binary tree is represented in memory?**

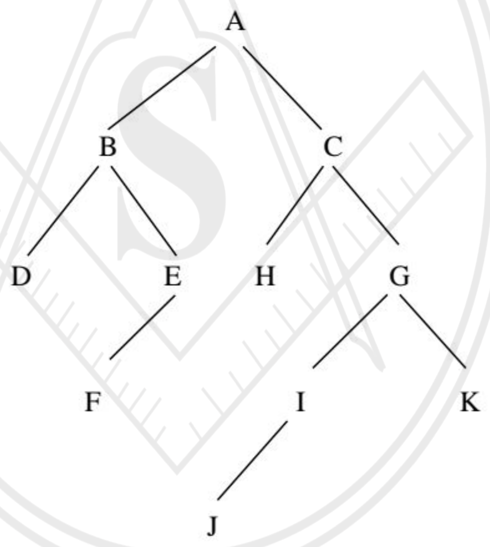
Ans. A tree T can be maintained in memory be means of a linked representation which uses three parallel arrays INFO , LEFT and RIGHT and a pointer variable ROOT.
Each node N of T will correspond to a location K such that:

1) INFO[K] contains the data at the node N.
2) LEFT[K] contains the location of the left child of node N.
3) RIGHT[K] contains the location of the right child of node N.

Example:-



Sr. No.	NAME	LEFT SUCCESSOR	RIGHT SUCCESSOR
1	A	2	3
2	B	4	5
3	C	8	7
4	D	0	0
5	E	6	0
6	F	0	0
7	G	9	11
8	H	0	0
9	I	10	0
10	J	0	0
11	K	0	0
12			



a) The value ROOT = 1 indicates that A is the root of the tree.
LEFT[1]= 2 indicates that B is the left child of A, and RIGHT[1]=3 indicates that C is the right child of A.
(Repeat the step for each node in the diagram)

Q. How to traverse binary tree?

Ans. There are three standard ways of traversing a binary tree T with root R. These three logarithms, called preorder, inorder and postorder, are as follows:

Preorder: (NLR)

1. Process the root R.

2. Traverse the left subtree of R in preorder.
3. Traverse the right subtree of R in preorder.

Inorder: (LNR)

1. Traverse the left subtree of R in inorder.
2. Process the root R.
3. Traverse the right suntree of T in inorder.

Postorder: (LRN)

1. Traverse the left subtree of R in postorder.
2. Traverse the right subtree of R in postorder.
3. Process the root R.

The three algorithms are sometimes called, respectively, the node-left-right (NLR) traversal, the left-node-right (LNR) traversal and the left-right-node(LRN) traversal.

Q. Explain LIFO and FIFO.

Or

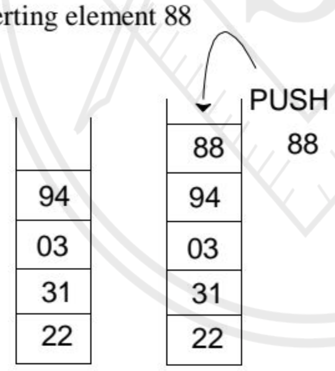
Explain Stack & Queue.

Ans. A stack is a linear data structure in which the data elements can be inserted and deleted at one end only i.e. the top. LIFO stands for Last-in-First-out. The element which is inserted last in stack will be removed (deleted) first. The everyday example of Stack is stack of dishes, stack of plastic cups.

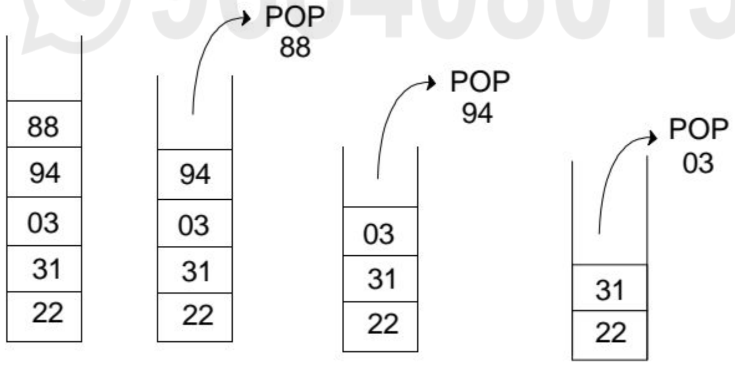
When you want to insert any element in stack, it is known as "PUSH" and when you want to delete any element from stack, it is known as "POP".

Example: Suppose 22,31,03,94 ; is a set of data elements

Inserting element 88



Suppose if we want to delete 03 from the stack, so we have to first delete the top element i.e. 88, then the second, then third, until we pop the required element.



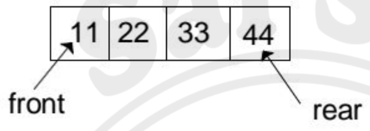
Now, the stack contains the elements 22, 31.

FIFO

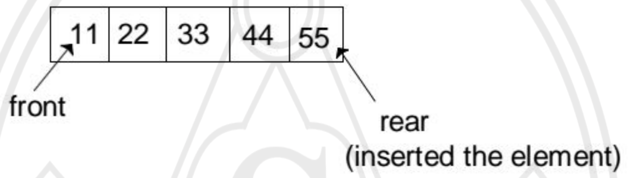
A queue is a linear data structure in which data elements can be inserted from one end and deleted from the other end. The element which is inserted first is deleted first and the element which has entered in the end will be deleted last, so the term FIFO. The everyday example of queue is line in a bus stop. The person standing first in the queue enters first in the bus and the person who has entered last in the queue , gets in last in the bus.

We insert the element from the "rear" and delete the element from the "front".

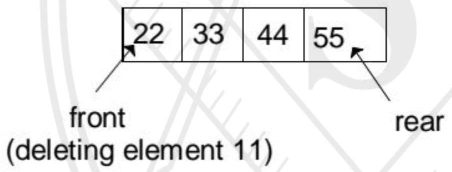
Example: Suppose, 11,22,33,44 is a set of data elements



Inserting element 55.



Now, if we want to delete the element 11.



Q. What are pointer arrays?

Ans. i) An array is called pointer array, if each element of that array is a pointer.

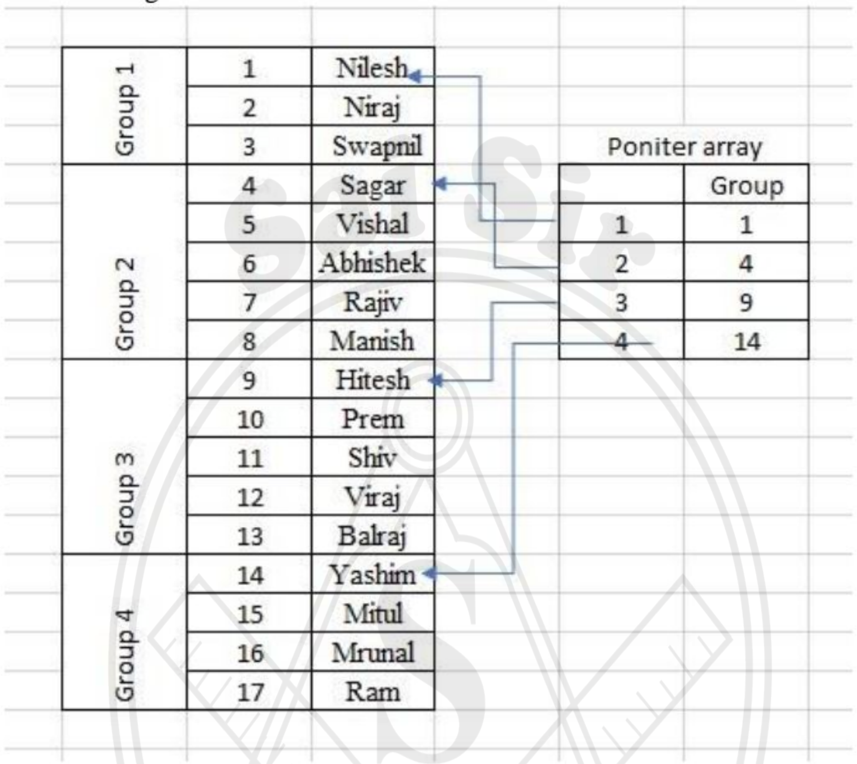
ii) The variable is called as pointer variable, it points to another variable i.e. it contains the memory address of other variable.

iii) consider a class, with divides its students list into four groups, depending on certain conditions. Following figure shows the list of 4 groups. There are 17 students and groups contains 3, 5, 5 and 4 students respectively as

Group 1	**Group 2**	**Group 3**	**Group 4**
Nilesh	Sagar	Hitesh	Yashim
Niraj	Vishal	Prem	Mitul
Swapnil	Abhishek	Shiv	Mrunal
–	Rajiv	Viraj	Ram
–	Manish	Balraj	–

iv) If these groups are to be represented in memory, the most efficient way is to use 2 arrays. The first is Students array, which contains list of students in all four groups sequentially, while the second array is Group array, which is a pointer array, which contains the starting address of each group in the Students array, respectively.

v) It is shown is figure :



Group 1	1	Nilesh
	2	Niraj
	3	Swapnil
Group 2	4	Sagar
	5	Vishal
	6	Abhishek
	7	Rajiv
	8	Manish
Group 3	9	Hitesh
	10	Prem
	11	Shiv
	12	Viraj
	13	Balraj
Group 4	14	Yashim
	15	Mitul
	16	Mrunal
	17	Ram

Poniter array

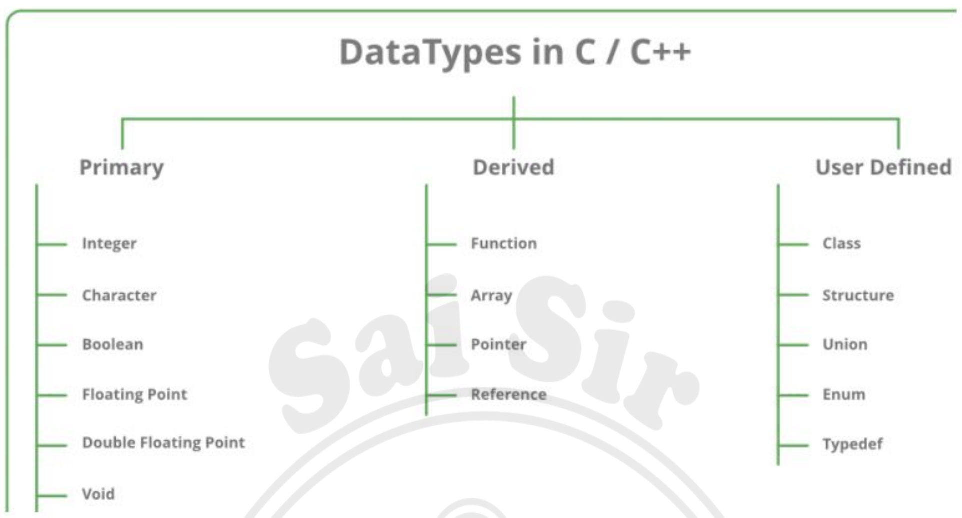
	Group
1	1
2	4
3	9
4	14

vi) Each element of Group array is a pointer, which holds the starting addresses of different groups. Hence, it is called as pointer array.

CHAPTER NAME :- C++

Q.	**Explain the different programming approaches.**
	Following are the different programming approaches: **(i) Procedural Programming:** Procedural programming paradigm lays more emphasis on procedure or the algorithm. A single program, which is divided into small pieces, is called as procedure. **(ii) Modular Programming:** Main program coordinates calls to procedures in separate modules and hand over appropriate data as parameters. Each module can have its own data. This allows each module to manage an internal state, which is modified by call to procedures of this module. **(iii) Object Oriented Programming:** Object oriented programming offers all the features of object based programming. The object oriented approach views a problem in terms of objects involved rather than procedure for doing it.
Q.	**State the various benefits of OOP.**
	Benefits of Object oriented Programming 1. Simplicity: Software objects model real world objects, so the complexity is reduced and the program structure is very clear. 2. Modularity: Each object forms a separate entity whose internal workings are decoupled from other parts of the system. 3. Modifiability: It is easy to make minor changes in the data representation or the procedures in an OO program. 4. Extensibility: adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones. 5. Maintainability: objects can be maintained separately, making locating and fixing problems easier. 6. Re-usability: objects can be reused in different programs.
Q.	**What are different data types in CPP?** **OR** **Explain different data types in CPP with size of data in terms of bytes for each**

CHAPTER NAME :- C++



Basic/Primitive data types:

Integer: Used for integer data types is int. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.

Character: Used for storing characters. Keyword used for character data type is char. Characters typically requires 1 byte of memory space and ranges from -128 to 127 or 0 to 255.

Boolean: Used for storing boolean or logical values. A boolean variable can store either true or false. Keyword used for boolean data type is bool.

Floating Point: Used for storing single precision floating point values or decimal values. Keyword used for floating point data type is float. Float variables typically requires 4 byte of memory space.

Double Floating Point: Used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is double. Double requires 8 byte of memory space.

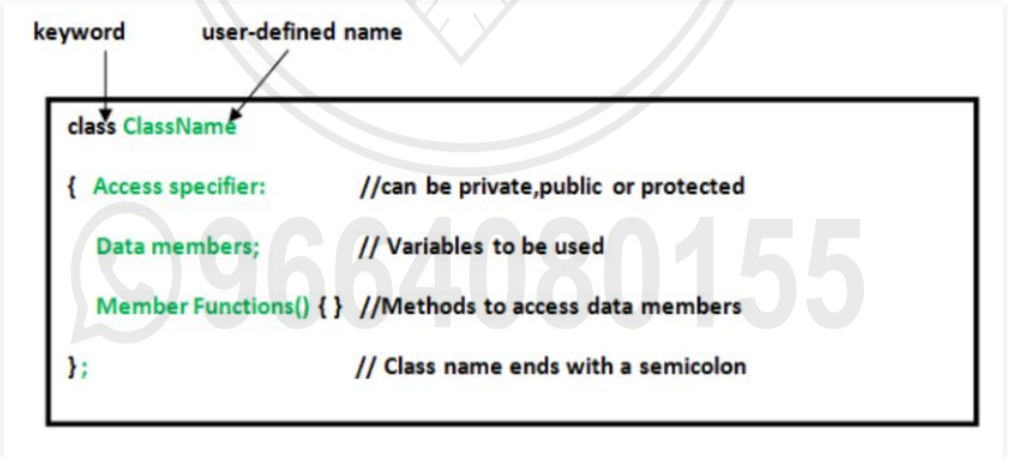
void: Void means without any value. It represents a valueless entity. Void data type is used for those function which does not returns a value.

Q. **Explain the various Object Oriented concepts.**
OR
Explain the following concepts:
1. **Abstraction 2. Encapsulation 3. Inheritance 4. Polymorphism**

Following are the general OOP concepts:
1. **Data Abstraction:** Abstraction refers to the act of representing essential features without including the background details or explanations. Abstraction is implemented through public members of a class, as the outside world is given only the essential and necessary information

CHAPTER NAME :- C++

	through public members, rest of the things remain hidden. 2. **Data Encapsulation**: The wrapping up of data and operations/functions (that operate o the data) into a single unit (called class) is known as Encapsulation. Encapsulation is implemented with the help of a class as a class binds together data and its associated function under one unit. 3. **Inheritance**: Inheritance is the capability of one class of things to inherit capabilities or properties from another class. Inheritance is implemented in C++ by specifying the name of the (base) class from which the class being defined (the derived class) has to inherit from. 4. **Polymorphism:** Polymorphism is the ability for a message or data to be processed in more than one form. C++ implements polymorphism through virtual functions, overloaded functions and overloaded operators.
Q.	**Explain the Components of C++ Program**
	• **Object** − Objects have states and behaviors. Example: A dog has properties- **color, name, breed** as well as behaviors - barking, eating. **An object is an instance of a class.** • **Class** − A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support. • **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed. • **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.
Q.	**Explain with example, how to define a class in CPP.** **OR** **What is a class? Give general form of class declaration.**

	• **A class is defined in C++ using keyword class followed by the name of class.**
	• **The body of class is defined inside the curly brackets and terminated by a semicolon at the end.**
	
	class ClassName
	{ Access specifier: //can be private,public or protected
	Data members; // Variables to be used
	Member Functions() { } //Methods to access data members
	}; // Class name ends with a semicolon
	Example:
	/* C++ program to create a simple class and object.*/
	#include <iostream.h>
	class Hello

CHAPTER NAME :- C++

```
{
   public:
      void sayHello()
      {
            cout << "Hello World" << endl;
      }
};
int main()
{
   Hello h;
   h.sayHello();
   return 0;
}
```

Q. **Explain scope resolution operator in CPP.**

- The scope resolution operator (::) is used for several reasons.
- For example: If the global variable name is same as local variable name, the scope resolution operator will be used to call the global variable.
- It is also used to define a function outside the class and used to access the static variables of class.

Program:

```
#include <iostream.h>
char a = 'm';
static int b = 50;

int main() {
  char a = 's';

  cout << "The static variable : "<< ::b;
  cout << "\nThe local variable : " << a;
  cout << "\nThe global variable : " << ::a;

  return 0;
}
```

Output:

```
The static variable : 50
The local variable : s
The global variable : m
```

Q. **Describe how member functions of a class can be defined outside the class definition and inside the class definition.**

Methods are functions that belongs to the class.There are two ways to define functions that belongs to a class:

CHAPTER NAME :- C++

	1. Inside class definition
	2. Outside class definition
	
	In the following example, we define a function inside the class, and we name it "myMethod".
	`class MyClass {        // The class`
	`  public:              // Access specifier`
	`    void myMethod() {  // Method/function defined inside the class`
	`      cout << "Hello World!";`
	`    }`
	`};`
	`int main() {`
	`  MyClass myObj;     // Create an object of MyClass`
	`  myObj.myMethod();  // Call the method`
	`  return 0;`
	`}`
	
	To define a function outside the class definition, you have to declare it inside the class and then define it outside of the class. This is done by specifiying the name of the class, followed the scope resolution :: operator, followed by the name of the function.
	
	`class MyClass {        // The class`
	`  public:              // Access specifier`
	`    void myMethod();   // Method/function declaration`
	`};`
	`// Method/function definition outside the class`
	`void MyClass::myMethod() {`
	`  cout << "Hello World!";`
	`}`
	`int main() {`
	`  MyClass myObj;     // Create an object of MyClass`
	`  myObj.myMethod();  // Call the method`
	`  return 0;`
	`}`

Q.	**What are pointers in CPP? Explain the use of pointer variables for function definition using call by value and call by reference.**
	• A pointer is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it. The general form of a pointer variable declaration is − – type *var-name; • Here, type is the pointer's base type; it must be a valid C++ type and var-name is the name of the pointer variable. **Call by Value:**

CHAPTER NAME :- C++

In call by value, value being passed to the function is locally stored by the function parameter in stack memory location. If you change the value of function parameter, it is changed for the current function only.

Example:

```
#include <iostream.h>
void change(int data);
int main()
{
int data = 3;
change(data);
cout << "Value of the data is: " << data<< endl;
return 0;
}
void change(int data)
{
data = 5;
}
```

Output:

Value of the data is: 3

Call by Reference:
In call by reference, original value is modified because we pass reference (address).

Here, address of the value is passed in the function, so actual and formal arguments share the same address space. Hence, value changed inside the function, is reflected inside as well as outside the function.

```
#include<iostream.h>

void swap(int *x, int *y)
{
 int swap;
 swap=*x;
 *x=*y;
 *y=swap;
}
int main()
{
 int x=500, y=100;
 swap(&x, &y);  // passing value to function
 cout<<"Value of x is: "<<x<<endl;
 cout<<"Value of y is: "<<y<<endl;
 return 0;
}
```

CHAPTER NAME :- C++

	Output: Value of x is: 100 Value of y is: 500
Q.	**Explain Access Modifiers or Access Specifier in CPP.** **OR** **Explain Scope rules for different members of the class.**
	Access Modifiers or Access Specifiers in a class are used to assign the accessibility to the class members. That is, it sets some restrictions on the class members not to get directly accessed by the outside functions. **There are 3 types of access modifiers available in C++:** **1. Public** **2. Private** **3. Protected** 1. Public: All the class members declared under the public specifier will be available to everyone. The data members and member functions declared as public can be accessed by other classes and functions too. 2. Private**:** The class members declared as private can be accessed only by the member functions inside the class. They are not allowed to be accessed directly by any object or function outside the class. Only the member functions or the friend functions are allowed to access the private data members of a class. 3. Protected: Protected access modifier is similar to private access modifier in the sense that it can't be accessed outside of it's class unless with the help of friend class, the difference is that the class members declared as Protected can be accessed by any subclass(derived class) of that class as well.
Q.	**Explain Constructor and destructor with example.** **OR** **Explain Constructor in C++? Write a suitable example how a constructor is declared and defined.** **OR** **What is a constructor & destructor. Give one example for each.** **OR** **Explain different types of constructors in CPP.**
	Depending on the question asked, write the answer
	A constructor is a special member function whose task is to initialize the objects of its class. It is special because its name is same as the class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it construct the value data members of the class.

CHAPTER NAME :- C++

The constructor functions have some special characteristics.

- They should be declared in the public section.
- They are invoked automatically when the objects are created.
- They do not have return types, not even void and therefore, they cannot return values.
- They cannot be inherited, though a derived class can call the base class constructor.

Different types of constructors: Default, Parameterized and Copy constructor.

I. ***Default Constructors:* Default constructor is the constructor which doesn't take any argument. It has no parameters.**

```
// Cpp program to illustrate the   concept of Constructors
#include <iostream>
using namespace std;

class construct {
public:
   int a, b;
 construct()
   {
      a = 10;
      b = 20;
   }
};
int main()
{
   // Default constructor called automatically
   // when the object is created
   construct c;
   cout << "a: " << c.a << endl
       << "b: " << c.b;
   return 1;
}
```

II. ***Parameterized Constructors: It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created.***

```
#include <iostream>
class Point {
private:
   int x, y;

public:
```

CHAPTER NAME :- C++

```
    // Parameterized Constructor
    Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }
    int getX()
    {
        return x;
    }
    int getY()
    {
        return y;
    }
};
```

III. Copy Constructor: A copy constructor is a member function which initializes an object using another object of the same class

```
#include <iostream.h>
using namespace std;

class point {
private:
  double x, y;

public:
  // Non-default Constructor & default Constructor
  point (double px, double py) {
    x = px, y = py;
  }
};

int main(void) {
    point b = point(5, 6);
}
```

Destructor

A destructor destroys an object after it is no longer in use. The destructor, like constructor, is a

member function with the same name as the class name. But it will be preceded by the character Tilde (~).
A destructor takes no arguments and has no return value. Each class has exactly one destructor.
If class definition does not explicitly include destructor, then the system automatically creates one by default.
It will be invoked implicitly by the compiler upon exit from the program to clean up storage that is no longer accessible.

```
// A Program showing working of constructor and destructor
#include<iostream.h>
#include<conio.h>
class Myclass{
public:
int x;
Myclass(){ //Constructor
x=10; }
~Myclass(){ //Destructor
cout<<"Destructing...." ;
}
int main(){
Myclass ob1, ob2;
cout<<ob1.x<<" "<<ob2.x;
return 0; }
```

Q **State the characteristics of friend function.**
OR
What is a friend function?

Friend function

- In general, only other members of a class have access to the private members of the class.
- However, it is possible to allow a nonmember function access to the private members of a class by declaring it as a friend of the class.
- To make a function a friend of a class, you include its prototype in the class declaration and precede it with the friend keyword.
- The function is declared with friend keyword.

Characteristics:

- The function is not in the scope of the class to which it has been declared as a friend.
- It cannot be called using the object as it is not in the scope of that class.
- It can be invoked like a normal function without using the object.
- It cannot access the member names directly and has to use an object name and dot membership operator with the member name.
- It can be declared either in the private or the public part.

Example:

```
//Friend function
```

CHAPTER NAME :- C++

```
#include<iostream.h>
#include<conio.h>
class B;
class A
{
 private: int data;
 public: A()
{
data=100;
}
friend int friendfunction(A,B);
};
class B
{
private:
int data;
public:
 B()
{       data=50;
}
friend int friendfunction(A aobj,B bobj);
};
 int friendfunction(A aobj,B bobj)
{
        return(aobj.data+bobj.data);
}
void main()
{ clrscr();
A obj1;
B obj2;
cout<<friendfunction(obj1,obj2);
}
```

In the above example, a friend function is defined in both the classes A and B and it is accessing the data(private member) of class A as well as class B.

Q. **Explain the concept of Friend class in CPP.**

//Friend class:

A friend class can access private and protected members of other class in which it is declared as friend. It is sometimes useful to allow a particular class to access private members of other class.

```
#include<iostream.h>
class A
```

CHAPTER NAME :- C++

```
{
 private:
       int data;
 public:
  A()
{
data=100;
}

friend class B;
};
class B
{
private: int b;
public:void friendclass(A aobj)
{
cout<<"value of data is:"<<aobj.data;
}

};
void main()
{
B obj;
A obj1;

obj.friendclass(obj1);
}
```

Q.	**Explain the concept of function overloading with example.**
	• Function overloading is a feature in C++ where two or more functions can have the same name but different parameters. • Function overloading can be considered as an example of polymorphism feature in C++. • Following is a simple C++ example to demonstrate function overloading.

CHAPTER NAME :- C++

```
#include <iostream>

void print(int i)

{

  cout << " Here is int " << i << endl;

}

void print(double  f) {

  cout << " Here is float " << f << endl;

}

void print(char const *c) {

  cout << " Here is char* " << c << endl;

}

int main() {

  print(10);

  print(10.10);

  print("ten");

  return 0;

}
```

Output:

```
Here is int 10
```

CHAPTER NAME :- C++

	Here is float 10.1 Here is char* ten

CHAPTER NAME :- C++

Q. What is operator overloading? Explain with suitable example and state Advantages of operator overloading.

Ans:

1. The mechanism of giving some special meaning to an operator is called an operator overloading.
2. In C++ the user defined data types behave in much the same way as a built-in data types.
3. Operator overloading provides a flexible option for the creating of new definition for most of the C++ operators.
4. When an operator is overloaded, its original meaning is not lost. For instance, the operator has been overloaded to add two vectors, can still be used to add two integers.
5. A special function called 'operator function' is used to specify the relation of the operator to the class.

E.g.

```
//increment counter variable with ++ operator
        #include <iostream.h>
        class counter
        {
        private:
        int count;
        Public:
        counter ()
        {  count = 0; }
        int get_count ()
        { return count; }
        void operator ++()
        { count++l; }

        };
void main()
{ counter C1,C2;
cout <<"C1 = "<< C1.get_count();
cout <<"C2 = "<< C2.get_count();
C1++;
C2++;
++C2;
cout <<"C1 = "<< C1.get_count();
cout <<"C2 = "<< C2.get_count();
}
```

CHAPTER NAME :- C++

In the above program two objects of class counter: C1 & C2 are created, they are initialize to 0. Then using overloaded ++ operator, increment C1 once & C2 twice and display the resultant values.

Advantages of operator overloading.

1. Operator overloading extends capability of operators to operate on user defined data.
2. It can also be applied to data conversion.
3. Using operator overloading technique, user defined data types behave in much the same way as a built in types.

Q. State steps involved in operator overloading.

Ans:

a. First create a class that defines the data type that is to be used in overloading operation.
b. Declare the operator function op() in the public part of the class. It may be either a member function or a friend function.
c. Define the operator function to implement the required operations.

Q. What is operator function? Explain difference between operator function as member function and friend function.

Ans:

1. To define additional task to an operator, it specify what it means in relation to the class to which the operator is applied. This is done with the help of special function, called operator function, which describe the task.
2. Function defines additional task to an operator or which gives a special meaning to an operator is called the operator function.
3. The general form of operator function is:
```
return-type  class_name : :  operator op(argument list)
{
        Function body //task defined
}
```
4. When return type is the type of value returned by the specified operation and op is the operator being overloaded.
   The op is preceded by the keyword operator. Operator op is the function name.

CHAPTER NAME :- C++

5. Operator functions must be either member functions or friend functions
6. The basic difference between operator function as a friend function and as a member function is that a friend function will have only one argument for unary operators and two for binary operators while a member function has no arguments for unary operators and one for binary operators. This is because the operator used to invoke the member function its passed implicitly and therefore is available for the member function. This is not the case with the friend function. Arguments may be passed either by value or reference.

Q. State any eight rules for overloading operators.

Ans: There are certain restrictions and limitations for overloading operators. Some of them are listed below:

1. Only existing operators can be overloaded. New operators cannot be created
2. The overloaded operator must have atleast one operand that is of user-defined type.
3. The basic meaning of an operator cannot change.ie we cannot redefine the plus (+) operator to subtract one value from another.
4. The overloaded operators follow the syntax rules of original operators.
5. Following are some operators that cannot be overloaded.

Size of	Size of operator
.	Membership operator
.*	Pointer to member operato
: :	Scope resolution operator
?:	Conditional operator

6. Following certain operators cannot be overloaded using functions but member functions can be used to overload them.

=	assignment operator
()	Function operator
[]	subscripting operator

➔ Class member access operator

CHAPTER NAME :- C++

7. Unary operators, overloaded by means of a member function taken no explicit arguments and return no explicit values.
8. Unary operators, overloaded by means of a friend function take one reference arguments.
9. Binary operators overloaded through a member function take one explicit arguments
10. Binary operators overloaded through a friend function takes two explicit arguments.
11. When using binary operators overloaded through a member function, the left hand operand must be an object of the relevant class
12. Binary arithmetic operators such as + , ., * & / must explicitly retum a value. They must not attempt to change their own arguments.

Q. Write Short notes on type conversion.

Ans: when constant and variables of different types are mixed in an expression, compiler applies rules of automatic type conversions. The assignment operation also causes automatic type conversion. The type of data on right of an assignment operator is automatically covered of the type of variable on left.

Eg.
```
int  m;
Float x = 3.14;
m = x;
```

Convert x in to an integer before it is assigned to m. the fractional part is truncated. Type conversion is automatic for built in data types.

Consider, v3 = v2+ v1;

Where v1, v2 & v3 are class type objects. When objects are of the same class type, the operation is carried out. But if one of the operand is an object and the other is in built in type variable, compiler gives error.

Compiler does not support automatic conversion for user defined data types. We need to design conversion routines by ourselves, if such operation is required.

Three types of situation may arise:

1. Conversion from built in type to class type.
2. Conversion from class type to built in type.
3. Conversion from one class type to another class type.

CHAPTER NAME :- C++

Q. What is Inheritance?

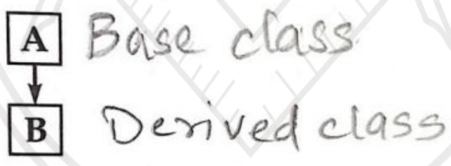
Ans:

1. The mechanism of deriving a new class from an old one is called as Inheritance.
2. The old class is referred as base class and new class is referred as derived class.
3. C++ strongly supports the concept of reusability. Once a class has been written and tested. It can be adapted by other programmers to suit their requirements.
4. This is basically done by creating new classes, reusing the properties of the existing ones.
5. Functions and variables of a class that has been tested can be used by object of another class. This is known as inheritance.
6. The reuse of a class that has already been tested, debugged and used many times can save the efforts of developing and testing the same again.

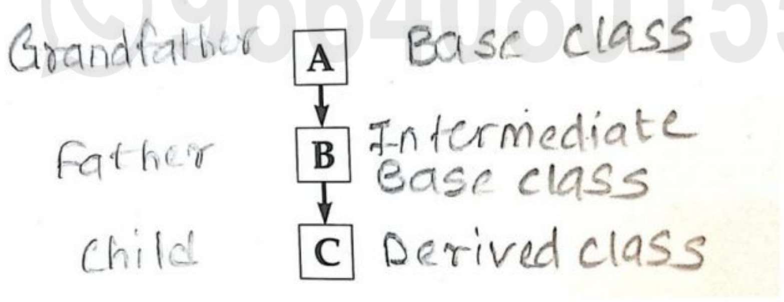
Q. Explain different types of inheritances with suitable diagram.

Ans: There are five different types of inheritances in C++ :

1. Single inheritance: A derived class with only one base class is called as single inheritance.

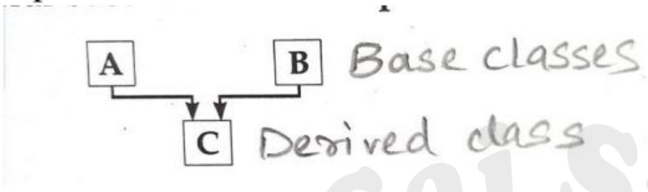


2. Multilevel inheritance: The mechanism of deriving one class from another derived class is multilevel inheritance.

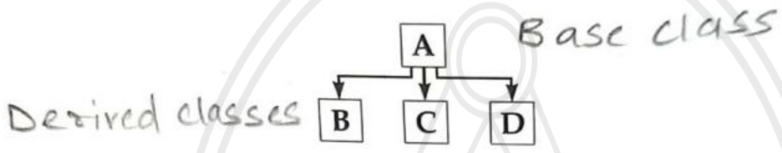


CHAPTER NAME :- C++

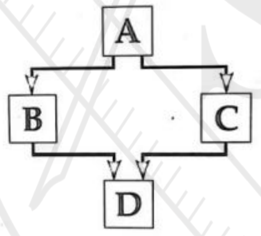
3. Multiple inheritance: When a class is derived from several base classes. It is called as multiple inheritance.



4. Hierarchical inheritance: The traits of one class may be inherited by more than one classes. This process id known as hierarchical inheritance.



5. Hybrid inheritance: the inheritance which involves more than one inheritance is called as hybrid inheritance.



Above figure involves hierarchical multiple and multilevel inheritances and the resultant inheritance is called hybrid inheritance.

Q. Explain multiple inheritance and multilevel inheritance.

Ans: 1. Multilevel inheritance:

The mechanism of deriving one class from another derived class is multilevel inheritance.

Following figure shows multilevel inheritance.

CHAPTER NAME :- C++

The class A serves as a base class for derived class B which in tum serves as a base class for the derived class C The class B is known as intermediate base class since it provides link for the inheritance between A and C The chain ABC is known as inheritance path

A derived class withy multilevel inheritance is declared as follows:

```
class A       //base class
{
------
};
class B: public A   //B derived A
{
---
};
class C: public B   // c derived from B
{
---
};
```

2. Multiple Inheritance:

A class can inherit the attributes of two or more classes. This is known as multiple inheritance.

Multiple inheritance allows us to combine the features of several existing classes as a starting point for defining new classes.
The syntax of derived class with multiple base classes is as follows:

```
class D: visibility B1, visibility B2,…….
{
(body of class D)
};
```

Where visibility may be either public or private. The base classes are separated by commas.

Q. What is polymorphism? Explain runtime and compile time polymorphism.

Ans:

1) Polymorphism refers to identically named methods (member functions) that has different behavour depending on the type of object they refer."

2) Polymorphism simply means "one name, multiple forms

CHAPTER NAME :- C++

3) The types of polymorphisms and their examples are shown in following figure.

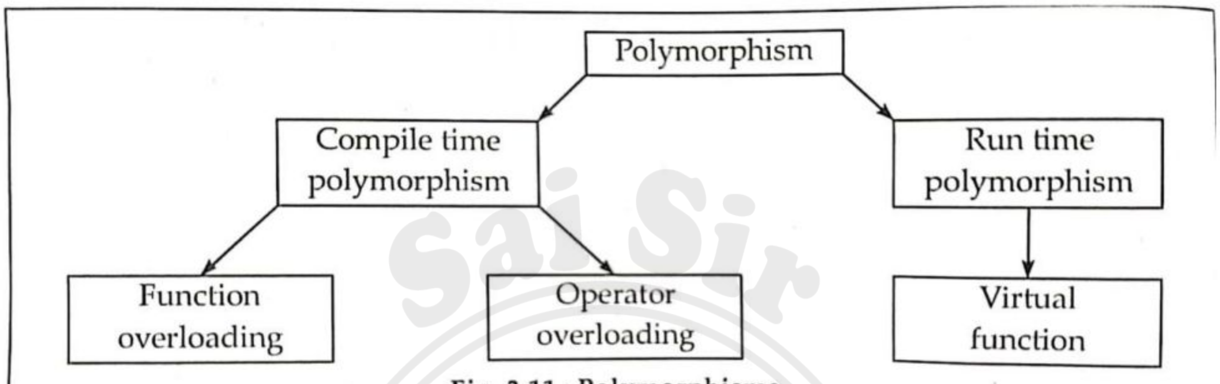


Fig. 3.11 : Polymorphisms.

➢ <u>Compile time polymorphism:</u>

1. Function overloading and operator overloading are the examples of compile time polymorphism.
2. In this case the overloaded member functions are selected for invoking by matching arguments both types and number.
3. This information is known to the compile at the compile time and therefore the compiler is able to select the appropriate function for a particular call at the compile time itself. This is known as compile time polymorphism.
4. Compile time polymorphism is also called as early binding or static binding or static linking. Early binding simply means that an object is bound to its function at compile time.

➢ <u>Run time polymorphism:</u>

1. In some situations it is nice to select appropriate member function to be invoked while the program is running this is known as runtime polymorphism.
2. Consider a situation where the function name and prototype is the same in both the base and derived classes as shown in following class definitions

```
class A
{
        int x;                  // private by default
        public:
```

CHAPTER NAME :- C++

```
        void show (void)            //show() in base class
        {……}
};
class B: public A
{
        int y;
         public:
        void show (void)            //show in derived class
        {…..}
};
```

Here, show() function is used to print values of object of both the classes A and B The prototype of show is the same in both the places, the function is not overloaded and therefore static binding does not apply

3. In such situations, the appropriate member function can be selected at runtime and it is known as runtime polymorphism
4. To achieve runtime polymorphism, C++ supports mechanism of virtual functions
5. At runtime, it is known what class objects are under consideration, the appropriate version of function is called
6. Since the function is linked with a particular class much after its compilation, this process is termed as late binding. It is also called as dynamic binding because the selection of the appropriate function is done dynamically at runtime

Q. Explain concept of virtual functions:

Ans:

1. When user use the same function name in both the base and derived classes, the function in base class is declared as virtual using the keyword virtual preceding its normal declaration.

CHAPTER NAME :- C++

2. When a function is made virtual C++ determines which function to use at run time, based on the type of object pointed to by the base pointer.
3. Thus, by making the base pointer to point two different objects, it can execute different versions of the virtual function
4. Virtual functions can be accessed through the use of a painter declared as a pointer to the base class
5. Also, the prototypes of the base class version of a virtual function and all the derived class version must be identical
6. If two functions with the same name have different prototype, C++ consider then an overloaded functions, and the virtual function mechanism is ignored.

Q. State any eight basic rules for virtual functions that satisfy the compiler requirements.

Ans:

When virtual functions are created for implementing late binding we should observe following basic rules that satisfy the compiler requirements

1. The virtual functions must be members of some class
2. They cannot be static members
3. They are accessed by using object pointers
4. A virtual function can be a friend of another class
5. A virtual function in a base class must be defined, even though it is not used
6. The prototypes of the base class version of virtual function and all derived class version must be identical. If two functions have different prototypes, then C++ considers them as overloaded functions and not as virtual functions.
7. We cannot have virtual constructors, but we can have virtual destructors
8. A base pointer can point to any type of derived object, the reverse is not true i.e. we cannot use a pointer to derived class to access an object of the base type
9. When base pointer points to derived class, the incrementation and decrementation is only relative to its base type.
10. Virtual functions are defined in base class, they need not be redefined in derived class

CHAPTER NAME :- C++

Q. What are input & output streams?

Ans: The I/O system of C++ handles file operations which are very much similar to console input and output operations

It uses file streams as an interface between the programs and the files

The streams that supply data to the program is known as input stream while the stream that receives data from the program is known as output stream.

In other words, input stream extracts (or reads) data from the file and the output stream Inserts (or writes) data to the file.

The input operation involves the creation of an input stream & linking it with the program and the input file. Similarly the output operation involves establishing an output stream and linking it to the program and the output file.

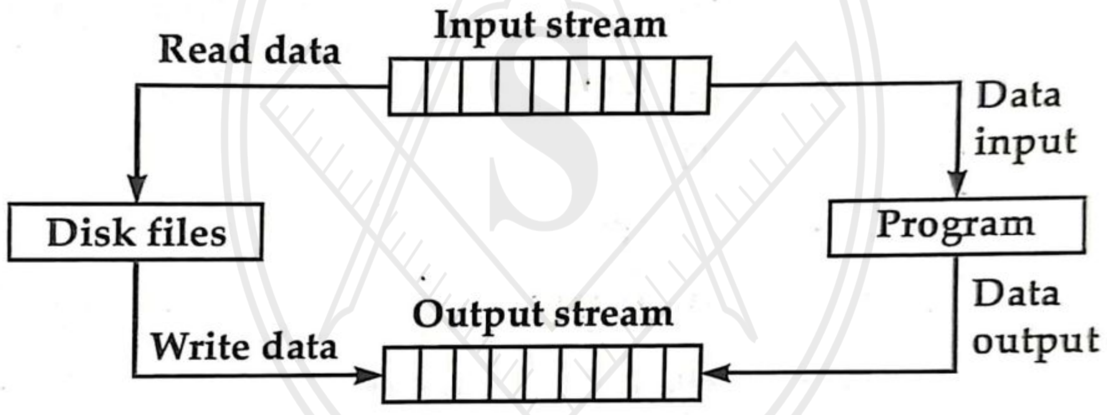


Fig. 3.12 : File input and output streams

Q. State the details of the following file stream classes:

i. ifstream ii. ofstream

Ans:

(a) ifstream:

1. This class is used for file handling methods.
2. This class is designed to manage the disk files and the user must include this fine in any program that uses files.
3. It provides input operations.

CHAPTER NAME :- C++

4. It contain open () function with default input mode.
5. It inherits get (), getline(), read(), seekg() & tellg() functions from istream.

(b) ofstream:

1. This class is alson used for file handling methods.
2. It provides output operations related to files.
3. It contain open () function with default output mode.
4. It inherits put (), seekp(), tellp() & write() function from ostream.

Q. What is the functions of each of the following file stream classes?

Ans: 1. ifstream: provides input operations. This file stream class is used to read a stream of objects from a file.

2. ofstream: provides output operations. Ofstream class is used to write a stream of objects in a file.

3. filebuf: its purpose is to set the file buffers to read and write. It contains close() and open() as members.

Q. What are different file modes?

Ans: The following table list the file mode parameters and their meanings.

Parameters	Meaning
ios::app	Append to end of life
ios::ate	Go to end of file on operating
ios::binary	Binary file
ios::in	Open a file for reading only
ios::nocreate	Open fails if the file does not exist
ios::noreplace	Open fails if the file already exist
ios::out	Open file for writing only
ios::trunc	Delete contents of file, if it exists

CHAPTER NAME :- C++

Q. What are classes in C++ for file stream operation? How do you open and close file in C++? Explain any four file modes.

Ans: Classes for file stream operation:

1. The I/O system of c++ contains a set of classes that define the file handling methods. They include:
    i. Ifstream
    ii. Ofstream
    iii. Fstream
2. These classes are derived from fstreambase and the corresponding iostream.h class.
3. They are defined to manage the disk files and declared in fstream.h

Opening and closing a file:

1. The general format for opening a file as :

    File-stream-class stream-object;
    Stream-object.open("file name");
2. Here ifstream class is used to read a stream of objects from a file and ofstream class is used to write a stream of objects in a file.
3. For E.g:

    Open a file to read stream of object from "data"

    Ifstream infile;
    Infile.open("data");

    Open a file to write stream of object from "data"

    Ofstream outfile;
    Outfile.open("data");

4. Closing a file: function close () is used to close a file, which is opened for read, write or read & write operations.

    For example: infile.close();

File modes:

With class fstream, file mode can be specified. The form of open() function as

Stream-object.open ("file name", mode)

CHAPTER NAME :- C++

File mode parameters are as follows:

1. ios::app – append to end of file.
2. ios::ate – go to the end of file on opening.
3. ios::binary -- binary file.
4. ios::in --- open a file for reading only.

Q. Write a c++ program to replace every space is an inputed string(less than 80 characters) with hyphen(i.e. -)

Ans:

```
#include<iostream.h>
#include<string.h>
int main()
{
int len;
char str[80];
cout<<"enter a string";
cin.get (str,80);
len = strlen(str);
for (int i=0; i<len; i++)
{
if(str[i] == ' ')
str[i]='-';
}
cout<<"the final string is";
cout<<str;
}
```

Q. Write a C++ program with ComputeCircle() function that returns the area a and circumference c of a cirle with given radius r.

Void ComputeCircle(float & a, float & c, float r)

Ans:

```
#include <iostream. h>
#include <conio.h>
void ComputeCircle (float & a, float & c, float r);
```

CHAPTER NAME :- C++

```
void main()
{
clrscr() ;
float r, a,c;
cout<<"Enter the value for radius r";
 cin>>r;
ComputeCircle (a,c,r);
cout << "The area with radius" <<r<<"is"<<a<" and its circumference
is"<<c<< endl;
getch ();
}

void ComputeCircle (float & a, float & c, float r)
{
a = 3.14*r*r;
c=3.14*2*r;
}
```

Q. Write a program to find a factorial of a natural number.

Ans:

```
#include<iostream.h>
#include<conio.h>
void main()
{
int fact, number;
clrscr();
fact=1;
cout<<"enter the number"<<endl;
cin>>number;
for(i=1; i<=number; i++)
{
fact=fact*i;
}
cout<<"the factorial of number is:"<<fact;
}
```

CHAPTER NAME :- C++

Q. Implement a class average. include a constructor in it which will accept value of three variables from user. include two more functions in it. one functions calculates average and other print it.

Ans:

```
#include<iostream.h>
#include<conio.h>
class average{
float a,b,c,avg;
public: average();
void calculate();
void print();
};

average:: average()
{
cout<<"enter numbers";
cin>>a>>b>>c;
}
void average::calculate()
{
avg={a+b+c)/3;
}
void average::print()
{
cout<<"the average of 3 nos is:"<<avg;
}
void main()
{
average obj;
obj calculate();
obj print();
}
```

Q. WAP to generate 20 terms of Fibonacci series.

Ans:

```
#include <iostream.h>
void main()
{
int f0, f1, f, n;
```

CHAPTER NAME :- C++

```
F0 = 0;
f1 = 1;
cout <<"Fibbonacci Series :";
cout << f0;
cout << f1;
for (n = 2; n < 20; n++)
{
f = f0 + f1;
cout << f;
f0= f1;
fl = f;
}
}
```

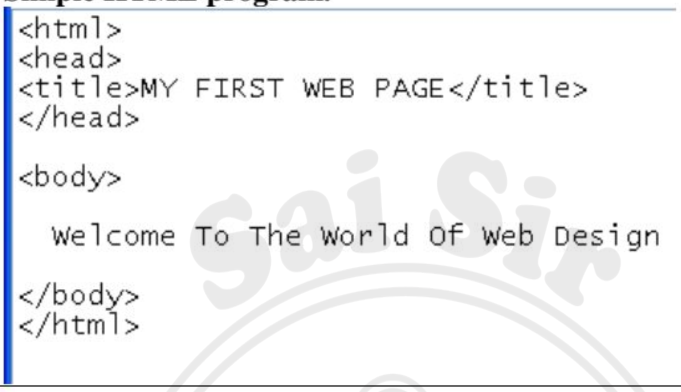
Q. WAp in C++ to read a set of 10 numbers from keyboard and find out largest number in the given array.
Ans:

```
#include<iostream.h>
int main ()
{
int num [10], max;
cout<<"“Enter the 10 numbers”;
for (int i = 0; i < 10; i++)
cin>>num [i];
max = num [0];
for (int j = 1; j<10;j++)
{
if(max<num [j])
max = num [j];
}
cout<<"The largest number in the array is " <<max;
}
```

Q1.	**What is HTML? State the advantages and disadvantages of using HTML.**
	• HTML describes the content and format of web pages using tags. • Ex. Title Tag: <title>A title </title> • It's the job of the web browser to interpret tags and display the content accordingly. • An HTML file contains both formatting tags and content • Document content is what we see on the webpage. • Tags are the HTML codes that control the appearance of the document content. HTML syntax: **two-sided tag:** • <tag attributes>document content</tag> ***Examples: <p> CGS 2100 </p>*** ***<body bgcolor = "yellow"> UCF </body>*** **Advantages:** 1. For creating HTML document, what you need is text editor. No special software is required. E.g. Notepad in windows. 2. HTML can be created on any hardware platform using any text editor. You can work on your website even when you are away from your usual computer. 3. If something is not working, then finding error is easy in HTML. 4. HTML will not cost you anything for its use. There are no expensive licenses to buy or no upgrades to purchase. 5. You can work independently and you need not to rely on vendor or program. You need not to worry about your editing programs. 6. Learning HTML is simple than any programming language. **Disadvantages:** • HTML is not a programming language in true sense. • Any **simple calculation** cannot be done in HTML. • It cannot be used to **display even date**. • The **interactive web pages** cannot be built by HTML. • The web pages developed in HTML cannot behave like an application. • The web pages developed in HTML do not have their own interface.
Q2.	**Explain the structure of HTML webpage**
	Structure of HTML document. **<HTML> … </HTML>** Beginning and end of every HTML document **<HEAD> … </HEAD>** Contains information about the document including the title that is to appear in the title bar **<TITLE> … </TITLE>** Causes the page title to be displayed in the title bar **<BODY> … </BODY>**

All content for the page is placed between these tags.

Simple HTML program.



```
<html>
<head>
<title>MY FIRST WEB PAGE</title>
</head>

<body>

  Welcome To The World Of Web Design

</body>
</html>
```

Q3. **Explain the following tags:**
<HTML>
<HEAD>
<TITLE>
<BODY>

<HTML> tag.

<HTML> tag declares that the text that follows defines an HTML web page that can be viewed in a web browser. </HTML> ends the page. The document is started with <HTML> tag

<HEAD> tag:

Defines the header area of page, which is not displayed within the page itself in the browser. In the <HEAD> section we have <TITLE> tag. The end tag is </HEAD> ends the head section.

<TITLE> tag:

Text between this tag & end tag is the title of the web page & is displayed in the title bar of browser. The title should be descriptive as it is frequently used by web indexing & searching programs to name your web page.

<BODY> tag:

Actual contents of the web page that will be displayed on browser will appear in body section of document. There are several optional attributes for this tag. For E.g. we can set back ground images, change font of text by using attributes.

Q4. **State the use of comments in HTML.**

- Comments can be inserted in the HTML code to make it more readable and understandable.

	• Comments are ignored by the browser and are not displayed. • Comments are written like this: <! -- This is a comment -->
Q5.	**Explain the heading tags(<h1> to <h6> in HTML**
	• **For proper layout of web document headings are useful. A web page can have maximum 6 levels of headings.** • **The <h1> to <h6> tags are used to define HTML headings.** • <h1> defines the largest heading (Highest Level) and <h6> (Lowest Level) defines the smallest heading. • **Syntax** <h1> This is highest heading</h1> • **Attributes:** Align: The headings can be aligned right or in center by setting align attribute for heading tag. • **Syntax:** <h1 Align=”value”>Text that you want to place</h1> • **The possible values are:** 1. Center 2. Right 3. Left **Example:** <html> <head> <title>Head Tag</title> </head> <body> <h1 align=”left”>This is heading 1</h1> <h2>This is heading 2</h2> </body> </html>
Q6.	**Explain container and standalone tags in HTML.**
	1) Container tag 2) Empty tag or stand alone tag Container Tag: Container tags are those who have opening tag and their respective closing tag. Eg: <body>, <html>, <head>, <title> , <p>, etc. Empty or stand alone tag: Stand alone tags are those who have opening tag but no closing tag. Eg:
, <HR>
Q7.	**Explain the following tags with examples:**

1. <P>　2. <SUP>　3. <pre>　4. <MARQUEE>
5. <OL>　6. <TR>　7. <B>　8. <LI>
9. <SUP>　10.<SCRIPT>　11.<table>　12. <div>
13. <a>

1. <P> … </P>

- The <p> element defines a paragraph.
- It has a start tag <p> and an end tag </p>:
- Blank line inserted before the start of the paragraph

Example:
<P>Learning HTML is a lot of fun!</P>

Attributes:
- left, center, and right justify a paragraph

<P ALIGN=LEFT>imagine a BIG paragraph in here</P>
<P ALIGN=CENTER> imagine a BIG paragraph in here </P>
<P ALIGN=RIGHT> imagine a BIG paragraph in here </P>

Note: </P> is used here to end the paragraph and turn off the alignment

2. <sup> tag:

The <sup> tag defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font.

Example:

<p>The following word uses a ^{superscript} typeface.</p>

3. <pre> tag:

The <pre> tag defines preformatted text.

Text in a <pre> element is displayed in a fixed-width font, and the text preserves both spaces and line breaks. The text will be displayed exactly as written in the HTML source code.

Example:

```
<pre>
       function testFunction( strText ){
         alert (strText)
       }
```

	</pre>
	4. <MARQUEE> tag The text enclosed in <marquee> - </marquee> would be rolled over horizontal line on the web page. This is used to roll the current news or position of different companies in stock market. **Example:** <marquee> STOCK MARKET COLLAPSES </MARQUEE> The display will be continuous rolling of words "stock market collapses"
	5. <ol> tag:
	Answer for this can be referred from the next question
	6. <tr> tag
	The <tr> tag defines a row in an HTML table. A <tr> element contains one or more <th> or <td> elements **Example:** <tr> <th>Month</th> <th>Savings</th> </tr>
	7. <b> tag
	The <b> tag specifies bold text without any extra importance. **Example:** <b> This text appears in bold</b>
	8. <li> tag
	Answer for this can be referred from the next question
	9. <sub> tag: The <sub> tag defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H2O. **Example:** <p>The following word uses a <sub>superscript</sub> typeface.</p>
	10. <script> tag: The <script> tag is used to embed a client-side script (JavaScript).

The <script> element either contains scripting statements, or it points to an external script file through the src attribute.

Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

11. <table>

- The <table> tag defines an HTML table.
- Each table row is defined with a <tr> tag.
- Each table header is defined with a <th> tag. Each table data/cell is defined with a <td> tag.
- By default, the text in <th> elements are bold and centered.

Example: Creating a table with 2 rows and 3 columns.

```
<table border="1">
 <tr>
  <th>Firstname</th>
  <th>Lastname</th>
  <th>Age</th>
 </tr>
 <tr>
  <td>Jill</td>
  <td>Smith</td>
  <td>50</td>
 </tr>
</table>
```

12. <DIV> tag:

This tag is used for applying alignment and style to only a section of a document.
Syntax:

```
<div align="value">Text that you want to display.
</div>
```

The possible values are:

1. Center
2. Left
3. Right

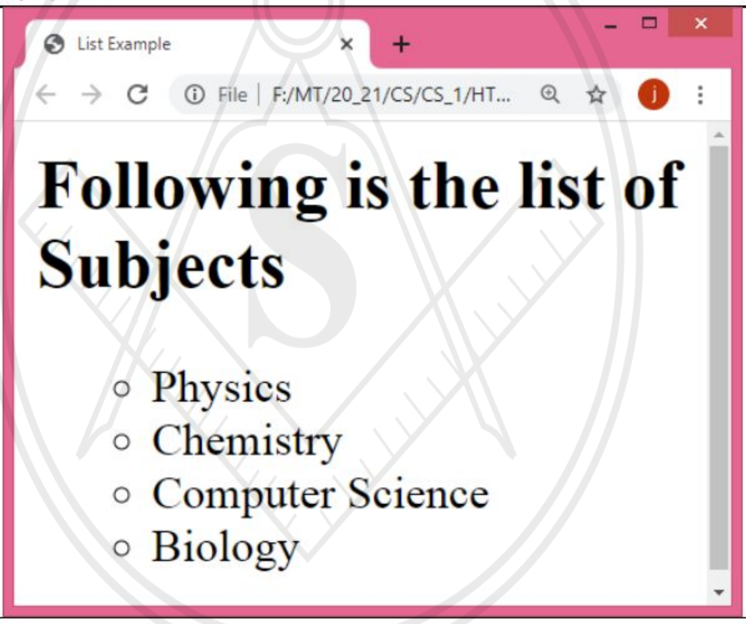
13. <a> tag:

The <a> tag defines a hyperlink, which is used to link from one page to another.

	The most important attribute of the <a> element is the href attribute, which indicates the link's destination. By default, links will appear as follows in all browsers: An unvisited link is underlined and blue A visited link is underlined and purple An active link is underlined and red **Examples:** <A HREF="name">hypertext</A> <A HREF="http://www.kodak.com">Kodak</A>
	14. <img> tag
	<img> Defines an image Attributes:SRC, ALT, HEIGHT, WIDTH, ALIGN, HSPACE ,VSPACE <p> An image: <img src="constr4.gif" width="144" height="50"> </p> <p> A moving image: <img src="hackanm.gif" width="48" height="48"> </p>
Q8.	**What are the types of list in HTML?**
	HTML lists allow web developers to group a set of related items in lists. • ***An unordered List*** • An ordered List • Definition List • Nested List **Unordered Lists**: An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. Unordered (bulleted) lists <UL> can use a disc, circle, or square **Example:** <h4>An Unordered List:</h4> <ul> <li>Coffee</li> <li>Tea</li> <li>Milk</li>

	</ul> **Output:** An Unordered List: • Coffee • Tea • Milk **Ordered (numbered) lists**: An ordered list starts with the <ol> tag. Each list item starts with the <li> tag. <OL> can use numbers (1), capital letters (A), lowercase letters (a), or Roman numerals (i) <OL TYPE=1 START=5> <LI>first line</LI> <LI>second line </LI> </OL> **All lists use <LI> to specify a new list item** **Nested List:** Example: <h4>A nested List:</h4> <ul> <li>Coffee</li> <li>Tea <ul> <li>Black tea</li> <li>Green tea</li> </ul> </li> <li>Milk</li> </ul> **Definition List:** Example: <h4>A Definition List:</h4> <dl> <dt>Coffee</dt> <dd>Black hot drink</dd> <dt>Milk</dt> <dd>White cold drink</dd> </dl>
	PROGRAMS
Q1.	**HTML program for creating an unordered list.**

```
<html>
        <head>
                <title>List Example </title>
        </head>
        <body>
                <H2> Following is the list of Subjects</H2>
                <ul type="circle">
                        <li>Physics</li>
                        <li> Chemistry</li>
                        <li>Computer Science</li>
                        <li>Biology</li>

                </ul>
        </body>
</html>
```



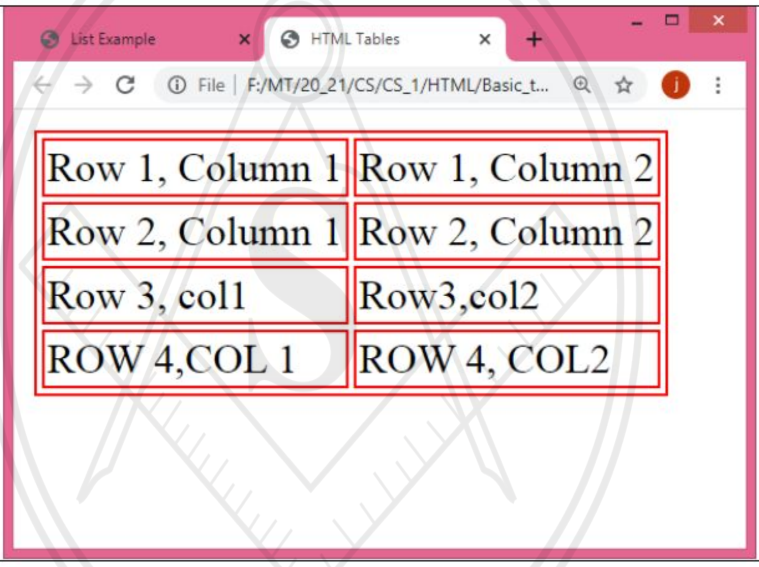
Following is the list of Subjects

- Physics
- Chemistry
- Computer Science
- Biology

Q2. **HTML Program for creating a simple table.**

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Tables</title>
  </head>
  <body>
    <table border="1" bordercolor="red">
      <tr>
        <td>Row 1, Column 1</td>
        <td>Row 1, Column 2</td>
      </tr>
       <tr>
```

```
            <td>Row 2, Column 1</td>
            <td>Row 2, Column 2</td>
         </tr>

            <tr>
                        <td>Row 3, col1</td>
                        <td>Row3,col2</td>

            </tr>
<TR>
             <TD>ROW 4,COL 1  </TD>
<TD> ROW 4, COL2 </TD>
     </TR></table>
        </body>
</html>
```

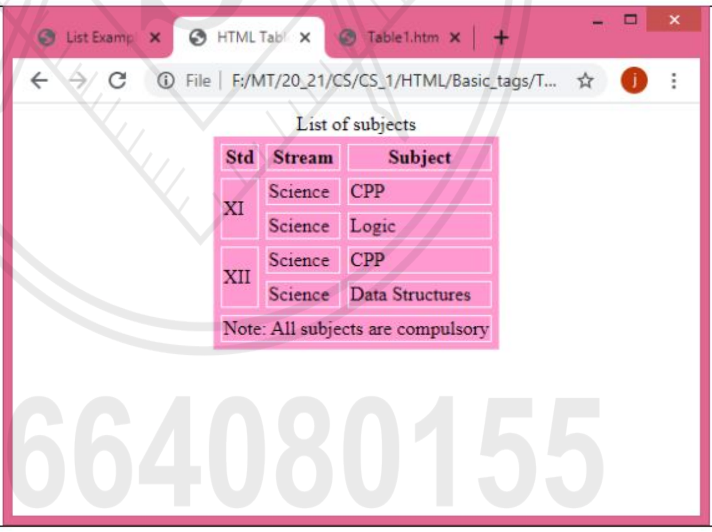


Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2
Row 3, col1	Row3,col2
ROW 4,COL 1	ROW 4, COL2

Q3. HTML Program to illustrate use of rowspan and colspan

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML Table Background</title>
  </head>

  <body text="#000000">
    <table align="center" border = "1" bordercolor = "white" bgcolor= "#ff99D0"
cellspacing="6">
<caption>List of subjects</caption>
<tr>
          <th>Std</th>
          <th>Stream</th>
```

```
        <th>Subject</th>

      </tr>
      <tr>
        <td rowspan = "2">XI</td>
        <td>Science</td><td>CPP</td>
      </tr>
      <tr>
        <td>Science</td>
        <td>Logic</td>
      </tr>
      <tr>
        <td rowspan = "2">XII</td>
        <td>Science</td><td>CPP</td>
      </tr>
      <tr>
        <td>Science</td>
        <td>Data Structures</td>
      </tr>
      <tr>
         <td colspan="3">Note: All subjects are compulsory</td>
      </tr>
    </table>
  </body>

</html>
```



List Examp | HTML Tabl | Table1.htm

File | F:/MT/20_21/CS/CS_1/HTML/Basic_tags/T...

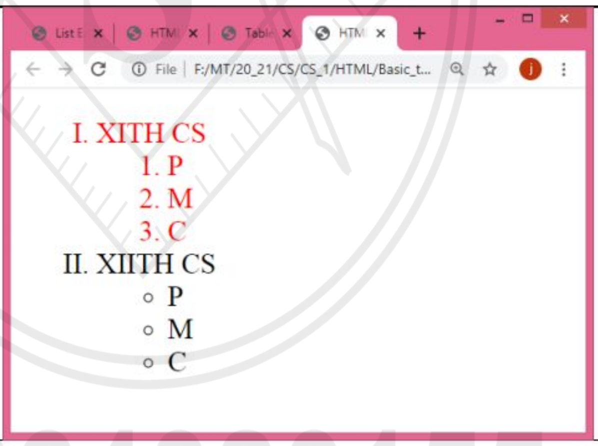
List of subjects

Std	Stream	Subject
XI	Science	CPP
	Science	Logic
XII	Science	CPP
	Science	Data Structures
Note: All subjects are compulsory		

Q4. HTML program to illustrate use of nested lists.

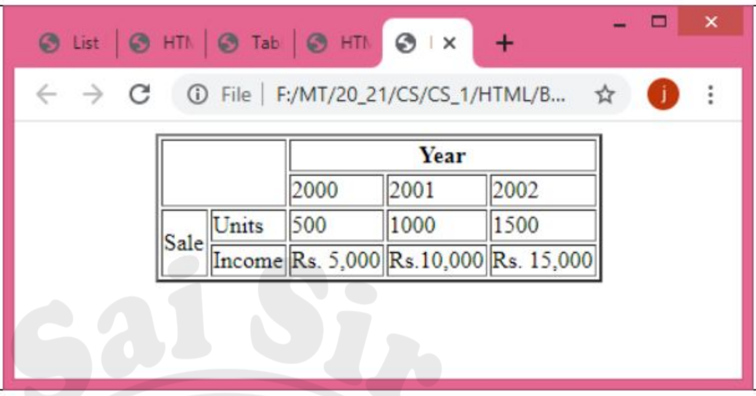
```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML NESTEd List</title>
  </head>
  <body>
```

```
<font color="red">
<OL TYPE="I">
<LI>XITH CS</LI>
        <OL>

        <LI>P</LI>
        <LI>M</LI>
        <LI>C</LI>
        </OL>
</FONT>
<FONT COLOR="BLACK">
<LI>XIITH CS</LI>
        <UL>
        <LI>P</LI>
        <LI>M</LI>
        <LI>C</LI>
        </UL>
</OL>
</BODY>
</HTML>


</body>

</html>
```



```
I. XITH CS
      1. P
      2. M
      3. C
II. XIITH CS
      ◦ P
      ◦ M
      ◦ C
```

Q5. **HTML program to generate the given output(table).**



		Year		
		2000	2001	2002
Sale	Units	500	1000	1500
	Income	Rs. 5,000	Rs.10,000	Rs. 15,000

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Table Background</title>
  </head>

  <body text="#000000">
    <table align="center" border = "2" >
<tr>
       <th colspan="2" rowspan="2"></th>
         <th colspan="3">Year</th>
 </tr>
  <tr>
         <td>2000</td>
         <td>2001</td>
        <td>2002</td>
</tr>
<tr>
<td rowspan="2">Sale</td>
<td>Units</td>
<td>500</td>
<td>1000</td>
<td>1500</td>
</tr>
<tr>
<td>Income</td>
<td>Rs. 5,000</td>
<td>Rs.10,000</td>
<td>Rs. 15,000</td>
</tr>
    </table>
  </body>
</html>
```

Output: As given in the output.

CHAPTER 1: INTRODUCTION TO MICROPROCESSORS AND ORGANIZATION OF 8085

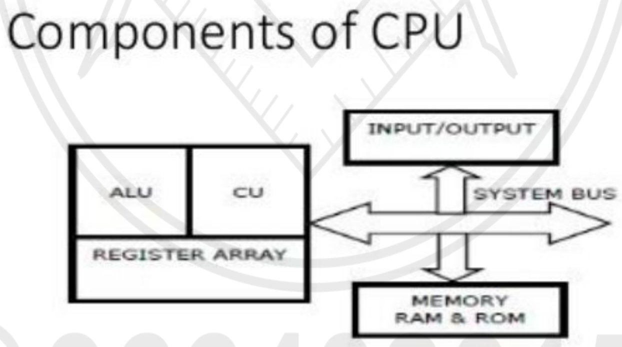
Qs 1. What is a microprocessor. Write down the primary functions of microprocessor.(3M)

Ans: (1) Microprocessor is a semiconductor, multipurpose, programmable logic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to the instructions and provides result as output.

(2) The electronic logic circuits in microprocessor are capable of performing various computing functions and making decisions to change the sequence of program execution.

(3) Microprocessor can also be viewed as an integrated circuit, that contains processing capabilities of large computers.

(4) A microprocessor can be roughly divided into three parts: Microprocessor



1. Arithmetic and logic unit (ALU)

2. Registers

3. Control Unit

A.L.U is arithmetic and logic unit, where arithmetical and logical operations are carried out. Registers are primarily used to store data temporarily during execution of program. Control unit provides timing and control signals to the whole system. It also controls flow of data.

The functions of microprocessor are given below:

(a) To fetch, decode and execute instructions.

(b) To transfer data from one block to another block or from one block to I/O lines.

(c) To give proper response to different externally produced interrupts according to their priority.

(d) To provide control and timing signals to the whole system according to the instructions.

Qs 2. Write a short note on Evolution of microprocessor giving one example of each generation.(3M)

Ans: The Evolution of microprocessor is as follows:

First Generation:
First microprocessor available in market was INTEL 4004 (4 bit) designed to be used in calculators in 1971.
In 1972, Intel introduced first general purpose 8 – bit microprocessor Intel 8008 with 45 instructions. Ex: Intel 4040, Toshiba T-3472

Second Generation:
In 1976, 8085, 8 bit microprocessor was introduced. Development of microprocessor has been in the direction towards a complete microcomputer system with CPU, ROM, RAM, clock, I/O ports all in single package. Ex. Intel 8048, Motorola MC 6801 etc.

Third Generation:
Microprocessor evolution has been towards one which performs all function of a minicomputer, which can work with bytes, strings of characters and instruction cycle.
In 1978 – Intel introduced 16 bit microprocessor 8086 now called as APX 86.
Ex. Intel 8086, 8088, zylog z8000 – all are16 bit microprocessor.

Fourth Generation:
In 1981 – Intel introduced 32 bit microprocessor 80386 which can address a physical memory of 4 giga bytes.
Pentium I, Pentium II, Pentium III & Pentium IV are recently introduced microprocessor by Intel, Intel Core 2 Duo, Intel Xeon are most advanced microprocessors.

Qs 3. Explain function of the following registers of 8085 Microprocessor(1 M each)

i. Stack pointer

Ans: (a) Stack pointer is a 16-bit register, which contains the address of stack top. i.e. the memory address of last byte entered in stack.

(b) With the help of incrementer/decrementer, the stack pointer is decremented each time data is pushed onto stack and incremented each time data is popped off the stack.

ii. Program counter

Ans: (a) The program counter is 16-bit register acting as a pointer to next executable instruction.

(b) It always contains the 16-bit address of the memory location where next executable instruction is stored.

(c) The microprocessor uses this register to sequence the execution of instruction.

(d) The PC is autoincremented after a particular instruction has been fetched by the microprocessor.

iii. Instruction Decoder

Ans: (1) This interprets the content of instruction register and determines exact steps to be followed in executing the entire instruction.

(2) It directs the control section accordingly.

iv. Instruction register

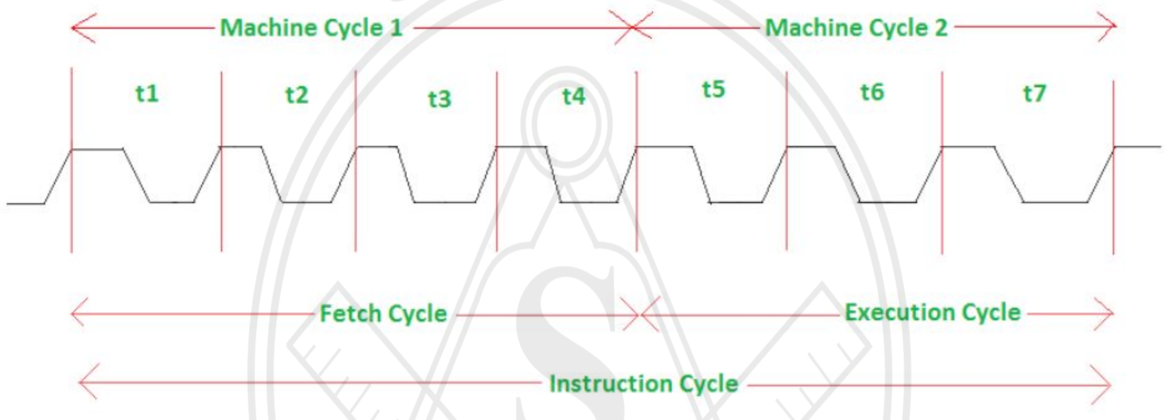
Ans: (a) This is an 8-bit register.

(b) The first byte of an instruction is stored in this register.

Qs 4. Define the following terms with suitable diagram:(3M)

i. T-state

Ans: The subdivision of an operation, which is performed in one clock period is called as T-state.



Instruction cycle in 8085 microprocessor

ii. Machine cycle

Ans: Machine cycle is defined as the time required to complete any operation of accessing wither memory or I/O which is the subpart of an instruction.

iii. Instruction cycle

Ans: An instruction cycle is defined as the time required to complete the execution of an instruction. The 8085 instruction cycle consists of one to five machine cycles.

Qs 5. Explain the flag register of 8085 Microprocessor with suitable example.

<u>OR</u>

Qs 5. What is flag? Explain Flag in 8085 with diagram.(4M)

Ans:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	-	AC	-	P	-	CY

Flag register of 8085

8085 has five flags: Sign flag, zero flag, Auxiliary carry flag, Parity flag and Carry flag.

A 8-bit register is used to represent five flags as shown in above figure where S - Sign flag, Z- Zero flag, Ac- Auxiliary carry flag, P - Parity flag, Cy- Carry flag.

(1) Sign flag (S) :After the execution of arithmetic and logic operation, if the most significant bit(MSB) of the result is 1, then the flag is set to 1 otherwise 0. This flag is used with signed number. If MSB is 1, the number will be negative and if it is 0, the number will be positive.

(2) Zero flag(Z) :After performing an arithmetic or logic operations, if the result is zero, then zero flag is set to 1, else it is reset(0). This flag is modified by the results in accumulator as well as in other registers.

(3) Auxiliary carry flag(AC) :In an arithmetic operation, when carry is generated from bit D3 to D4, the auxiliary carry flag is set to 1. This flag is only available internally and used for B.C.D. operations and not available for programmer.

(4) Parity flag (P) :Parity flag is set to 1, if the result stored in accumulator contains even parity, i.e., even number of 1' s. If accumulator contains odd number of 1' s, the flag is 0.

(5) Carry flag(CY) :In an arithmetic operation, when carry is generated, the carry flag is set to 1, else it is reset. This flag also serves as borrow flag for subtraction.

Qs 6. Write down the functions of Accumulator(Register A).(3M)

Ans: (a) Accumulator is 8-bit main register in 8085, used to perform the arithmetical and logical operations. In such operations, one of the operand is always stored in accumulator.

(b) It can be used as both primary source and destination register. The final result of operation is also stored in accumulator.

(c) All data transfer between the CPU and I/O devices are performed through accumulator.

(d) Many memory reference instructions move data between the accumulator and memory

Qs 7. What is are Hardware Interrupts? What are Maskable and Non-maskable interrupts? State all Hardware Interrupts of 8085 microprocessor with their priorities and branching or vector addresses.(4M)

Ans: 8085 provides 5 hardware interrupts:

(i) TRAP

(ii) RST 7.5

(iii) RST 6.5

(iv) RST 5.5

(v) INTR

(1) These interrupts are vectored interrupts. It means that when these interrupts are given , it is directed (or vectored) to transfer the control to specific memory location given by

TRAP = 4.5 x8= 0024H	RST7.5= 7.5 x 8 = 003C H
RST 6.5=6.5x8 = 0034 H	RST 5.5 = 5.5 x 8 = 002C H

(2) Among these interrupts, TRAP is non- maskable interrupt which cannot be disabled. But the other four interrupts are maskable interrupts, which can be disabled.

(3) The TRAP has highest priority and the INTR has lowest priority among the hardware interrupts. The hardware interrupts in descending order of priority are listed below: (i) TRAP - highest priority (ii) RST 7.5 (iii) RST 6.5 (iv) RST 5.5 (v) INTR - lowest priority.

Qs 8. Give functions of the following pins: (1M each)

i. CLK(OUT)

Ans: (1) The whole circuitry is synchronized with clock.
(2) The speed of the system depends on the clock frequency.

ii. $IO/\overline{M}$

Ans: (1) It is a status signal indicates whether the address bus is for I/O device or for memory.
(2) When it goes high, the address on the address bus referring I/O device and when it goes low, the address on the address bus referring memory.
(3) It is tristated during HOLD and HALT.

iii. $\overline{RD}$

Ans: (1) This is read control signal. This is active low signal.
(2) This signal indicates that selected I/O or memory device is to be read and data is available on data bus.
(3) It is tristated during HOLD and HALT.

iv. $\overline{WR}$

Ans: 1)This is write control signal.This is also active low signal.
2)This signal indicates that the data on the data bus is to be written into selected memory or I/O locations.
3)It is tristated during HOLD and HALT.

v. ALE

Ans:1) This is address latch enable.
2)This is positive going pulse generated every time the 8085 begins an operation (machine cycle). This indicates that the bits on AD_0-AD_7 are address bits.
3)This signal is used to separate the address bits.

vi. SID

Ans: (a) Serial input data. It is a data line for serial input.
(b) For serial data transmission SID pin is used. For this type of transmission RIM and SIM instructions are used.

vii. HLDA

Ans: (a) It is signal for HOLD ACKNOWLEDGEMENT.

(b) A HLDA output indicates to a peripheral that a HOLD request has been received and that the microprocessor will relinquish control of buses in the next clock cycle.
(c) After the removal of HOLD request, HLDA goes low.

viii. RESET OUT

Ans: 1) This signal indicates that mpu is reset.
2) The signal can be used to reset other devices.

ix. READY

Ans: (a) It is a input signal used by the microprocessor to sense whether a peripheral is ready to transfer data or not.
(b) This signal is used to delay the microprocessor until a slow responding peripheral is ready to send or accept data.
(c) If READY is high, the peripheral is ready. If it is low, the microprocessor waits for an integral number of clock cycles until it goes high.
(d) It is used to synchronize slower peripheral to faster microprocessor.

x. RST 6.5

Ans: (1) RESTART INTERRUPT : This signal is used to interrupt the microprocessor.
(2) When an interrupt is recognized the next instruction is executed from a fixed location is the memory
i.e.6.5*8 = 0034H.
(3) It is maskable interrupt.
(4) They cause an internal restart to be automatically insert.

xi. AD_0-AD_7

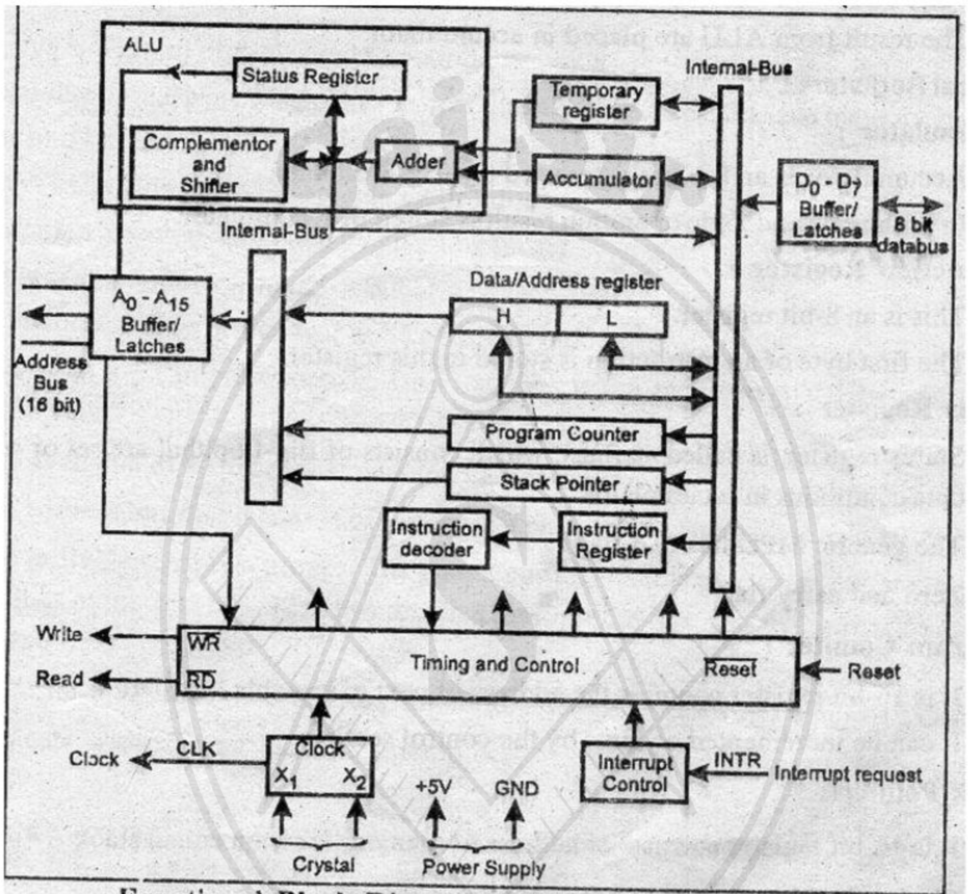
Ans: (1)Microprocessor 8085 has 8-bit data bus and 16-bit address bus.
(2) The least significant 8-bits of address bus are passed on the same eight lines as that of data bus i.e. on the signal lines AD7 - AD0.
(3) These signal lines are bi-directional.
(4) They are used for dual purpose for lower order 8-bit of address and as well as 8-bit of data. This is known as multiplexing and such bus is known as multiplexed bus.
(5) In multiplexed means, first to select one and then other.
(6) In executing an instruction, during earlier part of cycle these lines are used as the lower order address bus. During label part of cycle, these lines are used as data bus.

xii. $\overline{\text{INTA}}$

Ans1: (a) INTA is an Abbrevation for Interrupt Acknowledgement.

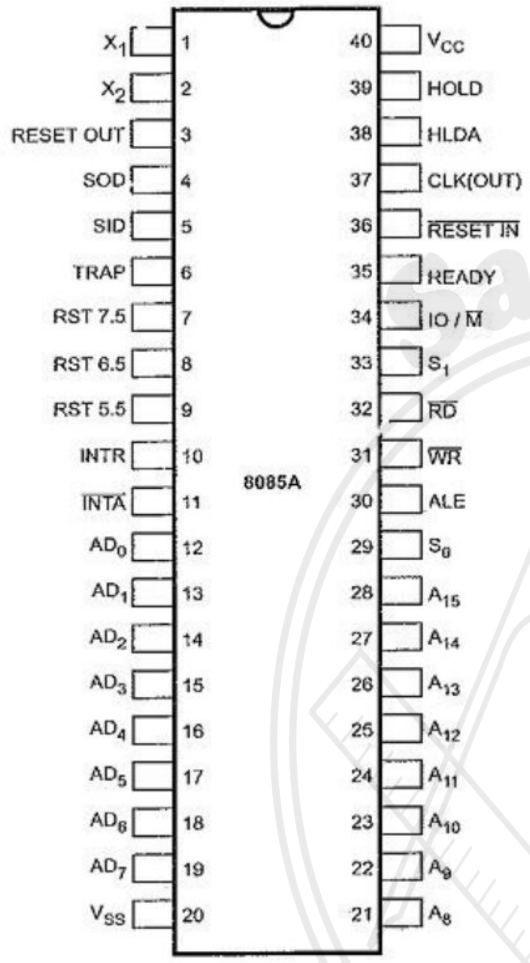
(b) A low an INTA indicates that the processor has acknowledged an INTR interrupt.

Qs 9. Draw the block diagram of generic microprocessor. (4M)
Ans:



Functional Block Diagram of Generic Microprocessor

Qs 10. Draw the pin diagram of 8085 microprocessor.(4M)
Ans:



8085A
X1 1, X2 2, RESET OUT 3, SOD 4, SID 5, TRAP 6, RST 7.5 7, RST 6.5 8, RST 5.5 9, INTR 10, INTA 11, AD0 12, AD1 13, AD2 14, AD3 15, AD4 16, AD5 17, AD6 18, AD7 19, VSS 20
40 VCC, 39 HOLD, 38 HLDA, 37 CLK(OUT), 36 RESET IN, 35 READY, 34 IO/M, 33 S1, 32 RD, 31 WR, 30 ALE, 29 S0, 28 A15, 27 A14, 26 A13, 25 A12, 24 A11, 23 A10, 22 A9, 21 A8

Fig. 1.3 (a) Pin configuration

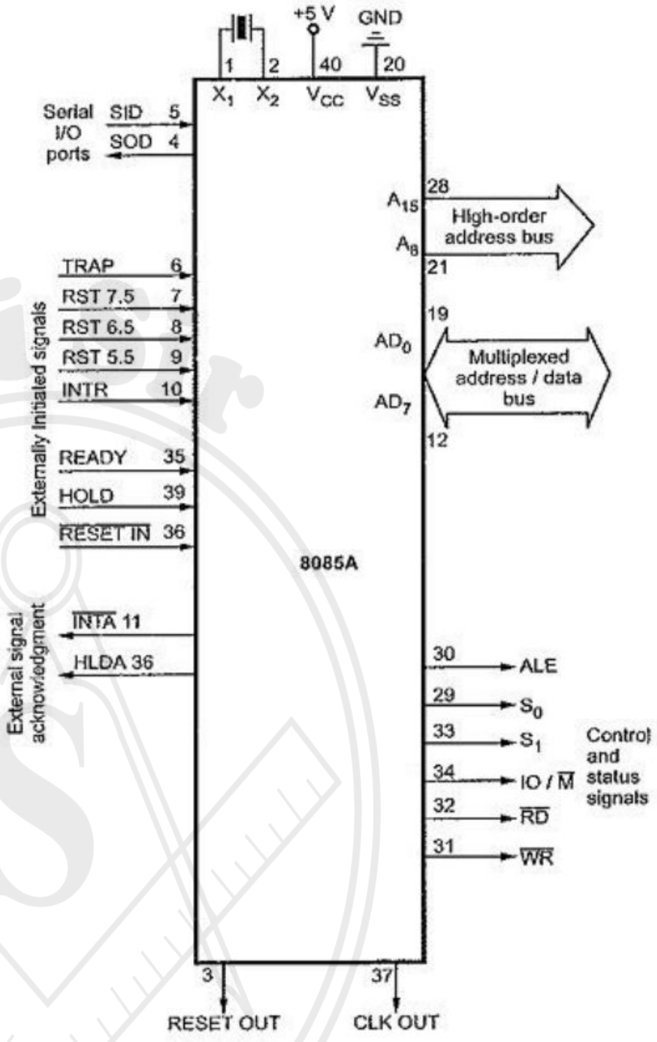


Fig. 1.3 (b) Functional pin diagram

Qs 11. In case of a microprocessor architecture, explain the following terms in brief:

i. Address bus

Ans: 1)This is group of 16 lines A_0-$A_{15.}$
2)It is unidirectional i.e. bits flowing in one direction- from mpu to peripheral devices.

3)It can carry 16-bit address and thus 64k locations can be addressed.

4)Mpu uses the address bus to identify a peripheral or memory location.

ii. Data bus

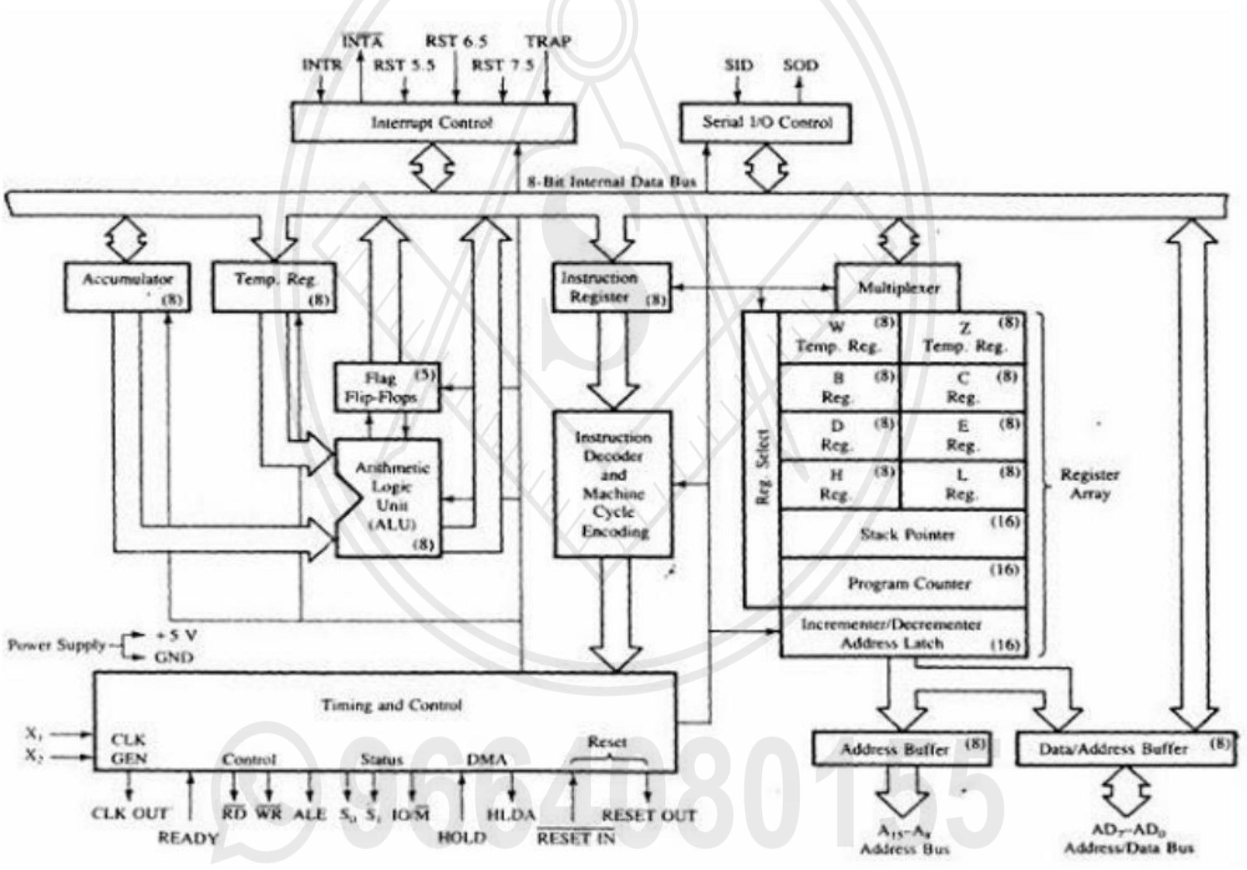
Ans: 1)This is group of 8 lines AD_0-AD_7.
2)It is bidirectional i.e. bits flowing in both direction between from mpu and peripheral devices.

3)It can carry 8-bit data ranging from 00 to FF.

4)Mpu having 8-bit data bus are called 8 bit microprocessors.

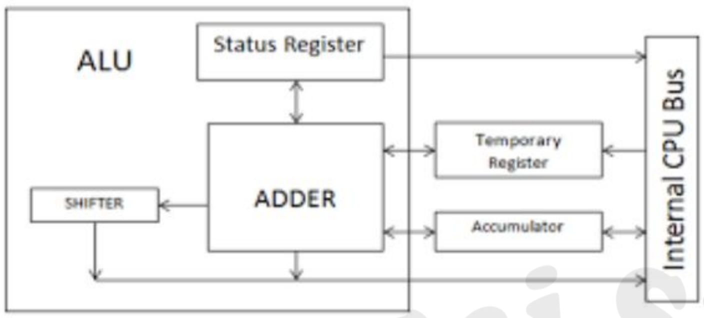
Qs 12. Draw the functional block diagram of 8085 microprocessor. (4M)

Ans:



Qs 13. Explain the function of ALU with simple block diagram.(3M)

Ans: The organization of arithmetic and logic unit is shown in figure.



1) The arithmetic and logic unit is 8-bit unit.

2) It performs arithmetic, logic and rotate operations.

3) It consists of binary adder to perform addition and subtraction by 2' s complement method.

4) The result is typically stored in accumulator.

5) Accumulator, temporary register and flag register are closely associated with A.L.U.

6) The temporary register is used to hold data during an arithmetic/ logic operation.

7) The flags are set or reset according to the result of operations in status register.

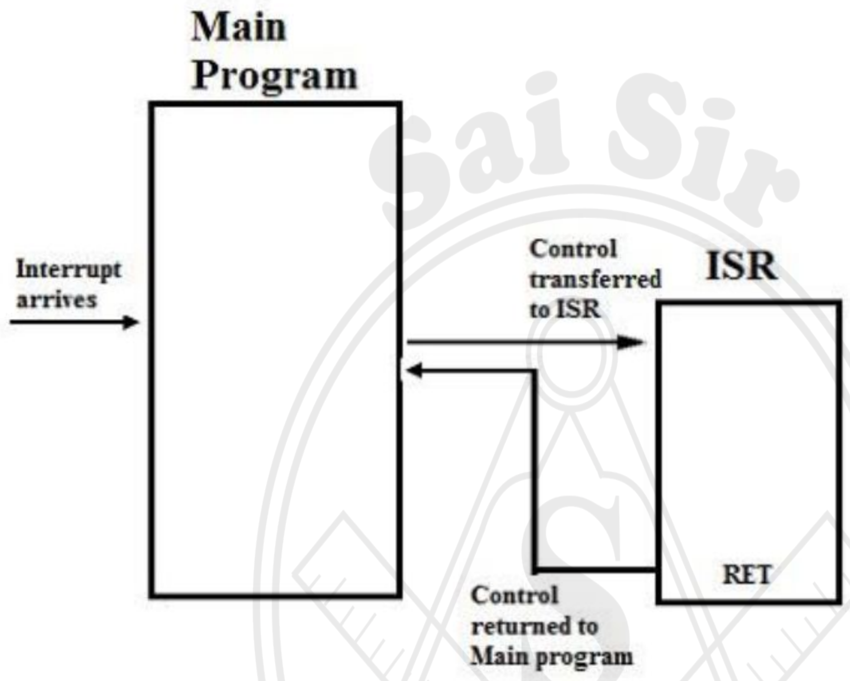
Qs 14. What do you mean by interrupt?

Ans: 1) An interrupt is a subroutine called, initiated by the external device through hardware (hardware interrupt) or microprocessor itself (software interrupt).

2) An interrupt can also be viewed as a signal, which suspends the normal sequence of the microprocessor and then microprocessor gives service to that device which has given the signal. After completing the service, the microprocessor again returns to the main program.

3) A microprocessor is connected to different peripheral devices. To communicate with these devices, microprocessor 8085 uses interrupt method.

4) An interrupt is an input signal, which transfers control to specific routine known as Interrupt Service Routine (ISR). After executing ISR, control is again transferred to the main program.



Qs 15. List all Software interrupts of Intel 8085

Ans:

1) The normal operation of a microprocessor can be interrupted by special instruction. Such an interrupt is called a Software interrupt.
2) 8085 provides 8 user-defined software interrupts RST 0 to RST 7 where RST means the restart.
3) These interrupts are vectored interrupts and when these interrupts are called the control is transferred to the memory location as shown below:

Interrupt	Mnemonics	Call Location (Hex)
RST	0	0000H
RST	1	0008H
RST	2	0010H

RST	3	0018H
RST	4	0020H
RST	5	0028H
RST	6	0030H
RST	7	0038H

4) Software interrupts are not used to handle asynchronous events. They are used to call software routines like a single step, breakpoint etc.
5) These interrupts are requested by executing interrupt instructions. They can also be requested due to arithmetic errors.
6) After execution of these interrupts, the program counter is incremented. The microprocessor does not execute any interrupt acknowledge cycle. The microprocessor executes normal instruction cycle.
7) These interrupts cannot be ignored or masked. They have more priority than any hardware interrupt.
8) They are not used to interface peripherals. That means, they don't improve the throughput of the system. They are used in program debugging.

CHAPTER 2: INSTRUCTION SET & PROGRAMMING OF 8085

Addressing modes:

Addressing modes of microprocessor is the various formats of specifying the operands. Every microprocessor has its own set of instructions. Each of these instruction uses one of the addressing modes.

8085 microprocessor has FIVE addressing modes.

1. Implied addressing
2. Register addressing
3. Immediate addressing
4. Direct addressing
5. Register indirect addressing

1.. IMPLIED ADDRESSING:

The addressing mode of certain 8085 instructions is implied by the instructions functions.

E.g.1. The STC (set carry flag) instruction deals with the carry flag only and no other register or memory locations.

E.g.2.CMA instruction complements the content of the accumulator. No specific data or operand is mentioned in the instruction.

2. REGISTER ADDRESSING MODE :- register addressing mode is used to move the data between the internal registers and hence the instruction source and destination address specifies which of these registers are involved in the transfer.

E.g. 1

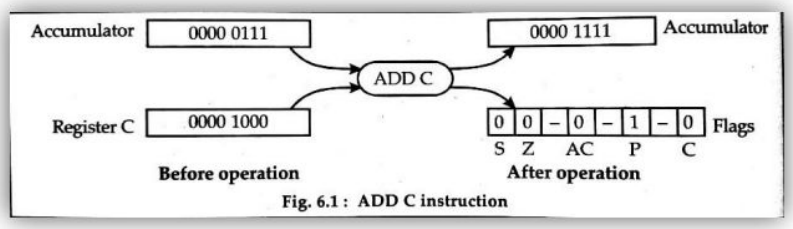


Fig. 6.1 : ADD C instruction

Here After ADC C instruction the result is stored in accumulator.

Instructions using in register addressing are very efficient in that they only use 1 byte of program memory space. They are also executed quickly because they do not have to fetch operands from memory

3.. IMMEDIATE ADDRESSING:

In immediate addressing mode the data appears immediately after opcode of instruction in program memory. 8 0r 16 bit data can be specified as a part of instruction.

E.g. ADI (add immediate) instruction.

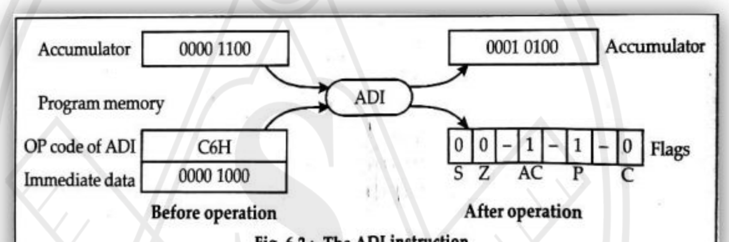


Fig. 6.2 : The ADI instruction

E.g.2 MVI A, 02 – Where A is the destination and 02 is the source data and MVI is the operation. In such type of instruction, the value of the source data is transferred to register A.

4.. DIRECT ADDRESSING:

In direct addressing, the address appears after the OP code of instruction in program memory. Instruction using direct addressing are 3 byte instructions.

Byte - 1 is OP code of instruction, byte - 2 is lower order address and byte – 3 is high order address.

E.g. LDA(load accumulator direct) instruction.

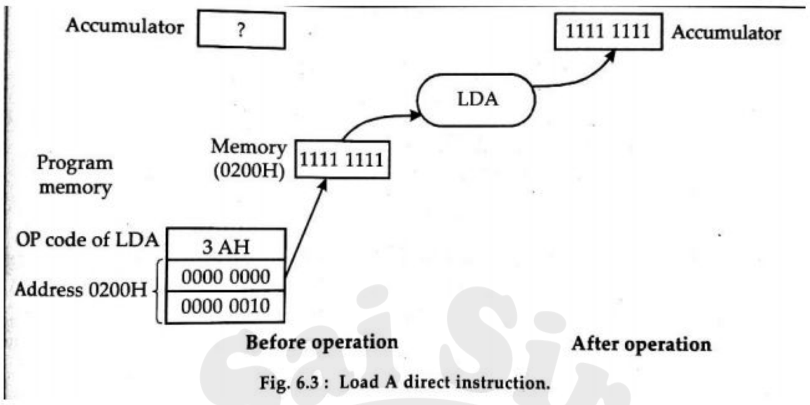


Fig. 6.3 : Load A direct instruction.

When LDA 0200 H instruction is executed, the content of memory address 0200 H (i.e. 1111 1111) are loaded in accumulator.

Direct address mode requires a lot of program memory space

5.. REGISTER INDIRECT ADDRESSING:

In register addressing mode, the contents of register pair points to the address of the operand.

E.g. ADD M instruction adds the contents of the accumulator to the contents of memory pointed by the address in HL pair.

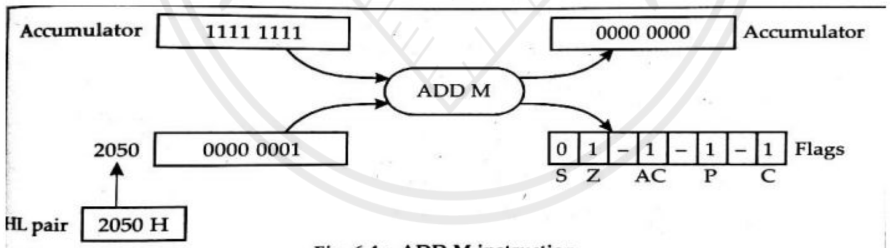


Fig. 6.4 : ADD M instruction

Some of the instructions in 8085 uses combined addressing modes.

E.g. CALL instruction combines direct addressing and register indirect addressing

➢ <u>**Data Transfer Group :**</u>

1. <u>MOV r1, r2 [Move register]</u>

Format: (r1) ← (r2)

<u>Addressing:</u> Register addressing.

The content of register r2 is moved to register r1. r1 & r2 can be one of the registers A, B, C, D, E, H, L.This is 1 byte instruction.

E.g. Let [A] = 05H [B] = 55H

Instruction: MOV A, B

After execution: [A] = 55H

2. <u>MOV r, M [Move from Memory]</u>

Format: (r)← [(H)(L)]

<u>Addressing:</u> Register indirect. This is 1 byte instruction.

The contents of memory location, whose address is in register H & L, is load to destination register r. r can be one of the registers A, B, C, D, E, H, L.

E.g. let [H-L] A300 H, [A300] = 35 H, [B] = 82H

Instruction: MOV B, M

After execution: [B] = 35H

<u>3. **MOV M, r [Move to memory]**</u>

Format: [(H) (L)]← (r), <u>Addressing</u>: Register Indirect

This is 1 byte instruction.

The content of register r is moved to the memory location whose address is in registers H & L pair register. r can be one of the registers A, B, C, D, E, H, L.

E.g. let [HL] = F000 H & F000=40H, [C] = FA H

Instruction: MOV M, C

After execution: [F000] = FA H

4. MVI r, data [Move Immediate]

Format: (r)← (byte 2), Addressing : Immediate addressing

The contents of byte 2 (data) of instruction is moved to register r. This is two byte instruction.

E.g. MVI A, 82H

The instruction will load accumulator with immediate data 82 H

5. MVI M, data [move to memory immediate]

[(H) (L)] ← (byte 2), Addressing : Immediate / reg, indirect addressing.

The content of byte 2 of the instruction is moved to memory location whose address is in registers H & L . This is two byte instruction.

E.g. let [H][L] = D000 H

MVI M, 12H

The data 12 is moved to memory pointed by HL pair i.e. [D000] = 12H.

6. LXI rp, 16- bit data [load register pair immediate]

Format: (rp) ← 16 bit data (rh) ← (byte 3) (rl) ← (byte 2)

addressing : immediate addressing

Byte 3 of instruction is moved into the high-order register (rh) of register pair rp. Byte 2 of the instruction is moved into the low order register (rl) of register pair rp. This is 3 byte instruction.

The register pair rp can be one of the register pairs BC, DE, HL or SP.

rh is first register(high order) in pair and rl is second(low order) register in pair.

E.g. LXI H, 3500H

This instruction will load H-L pair with 3500H. 35 h will be loaded in high order register(H) & (00) will be loaded in low order register (L).

7. LDA addr (load accumulator direct)

Format: (A) ← [(byte 3) (byte 2)], Addressing: Direct addressing.

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A. This is byte 3 instruction.

E.g.[C000] = 26H

Instruction: LDA C000H

After execution: [A] = 26H

The content of memory location C000H is moved to register A.

<u>8. STA addr [store Accumulator direct]</u>

Format : [(byte 3)(byte 2)] ← (A) , Addressing : Direct addressing

This is 3 byte instruction

The content of accumulator is moved to the memory location whose address specified in byte 2 and byte 3 of instruction.

E.g. [A]= 35H

Instruction: STA D000H

[D000] = 35H . Contents of accumulator are moved to D000 H

<u>9. LHLD addr (Load H & L direct)</u>

Format : (L) ← [(byte 3) (byte 2)]

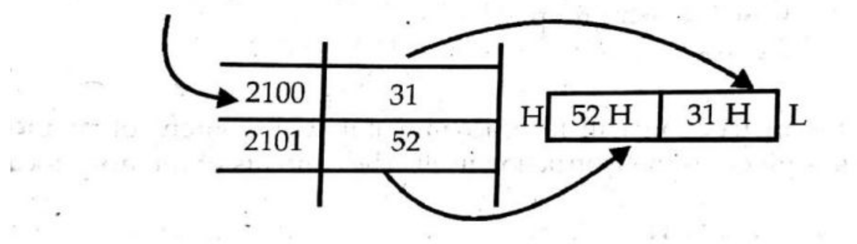
(H) ← [(byte 3) (byte 2)+1],

Addressing: Direct Addressing

This is 3 byte instruction. The content of memory location whose address is specified in the byte 2 and byte 3 of the instruction, is moved to register L. the content of memory location at the succeeding address is moved to register H.

E.g. let memory 2100 H contains 31H & 2101 H contains 52H, then after execution of instructions-

Register will contain 52H & register L will contain 31 H



10. SHLD addr (Store H & L direct)

Format: [(byte 3) (byte 2)] ← (L)

[(byte 3) (byte 2)+1] ← (H),

Addressing : Direct addressing

The content of register L is moved to the memory location whose address is specified in byte 2 & byte 3. the content of register H is moved to the succeeding memory location.

E.g. let [H] = 32H , [L]= 35H

Instruction: SHLD 2100 H

After execution: [2100] = 35H , [2101] = 32H

11. LDAX rp [load accumulator indirect]

Format: (A)← [(rp)], Addressing: Register indirect addressing

The content of memory location, whose address is in register pair rp, is moved to register A.

rp can be B(i.e. B & C) or (D & E). This is one byte instruction.

E.g. let [B] = 25H, [C] = 25H & [2525] = 33 H

Instruction: LDAX B

After execution: [A] = 33H

12. STAX rp [store accumulator indirect]

Format: [(rp)] ← (A), Addressing : Reg. indirect

The content of register A is moved to the memory location whose address is in register pair rp.

rp can be B(i.e. B & C) or (D & E). This is one byte instruction.

E.g. let [D] =25, [E]= 25 H [A] = 55H

Instruction: STAX D

After execution: [2525] = 55H

13. XCHG [exchange H & L with D & E)

(H) ← → (D) (L) ← → (E)

Addressing : register

The content of register H & L are exchanged with the contents of registers D & E. this is one byte instruction.

E.g. let [H] = 23H [L] = 32H, [D] = 53H [E] = 55H

Instruction: XCHG

After execution: [H] =53H [L] = 55H

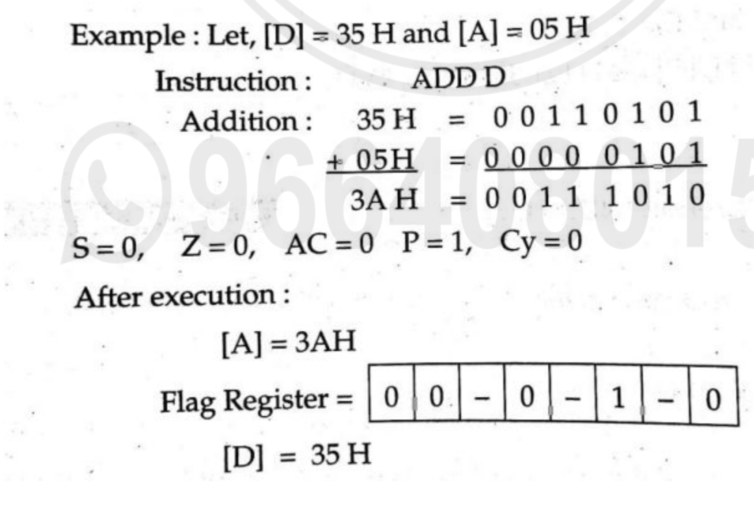
[D] = 23H [E] = 32H

➢ **ARITHMETIC GROUP:**

1. ADD r [Add register] (without Carry)

Fromat: (A)← (A) + (r), Addressing : Register addressing. Flag : All affected

The content of register r is added to the contents of the accumulator. The result is placed in accumulator. This is one byte instruction.


Example : Let, [D] = 35 H and [A] = 05 H

Instruction : ADD D

```
Addition :  35 H  =  0 0 1 1 0 1 0 1
          + 05H   =  0 0 0 0 0 1 0 1
            3A H  =  0 0 1 1 1 0 1 0
```

S = 0, Z = 0, AC = 0 P = 1, Cy = 0

After execution :

[A] = 3AH

Flag Register =

0	0	–	0	–	1	–	0

[D] = 35 H

2. ADD M [Add memory] (without Carry)

Format: (A) ← (A) + ((H)(L)), Addressing : Reg. indirect addressing. Flags: All

This is one byte instruction. The content of memory location whose address is contained in the H & L registers is added to the contents of accumulator.

The result is placed in accumulator.

E.g. let [H-L] = D000 H, [D000] = 51 H, & [A] = 35 H

Instruction: ADD M

After execution: [A] = 86H [D000] = 51H

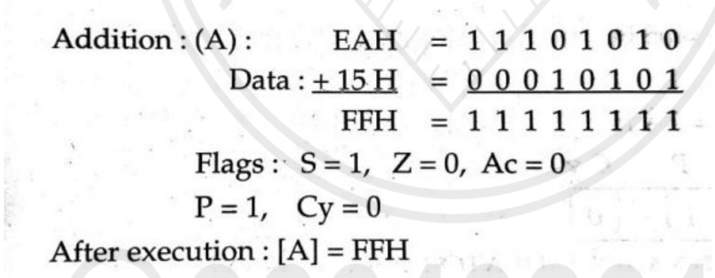
3. ADI data [Add immediate] (without Carry)

(A) ← (A) + (byte 2), Addressing : Immediate, Flags: All

The content of second byte of the instruction is added to the contents of the accumulator. The result is placed in accumulator. This is 2 byte instruction.

E.g. let [A] = EA H

Instruction: ADI 15 H

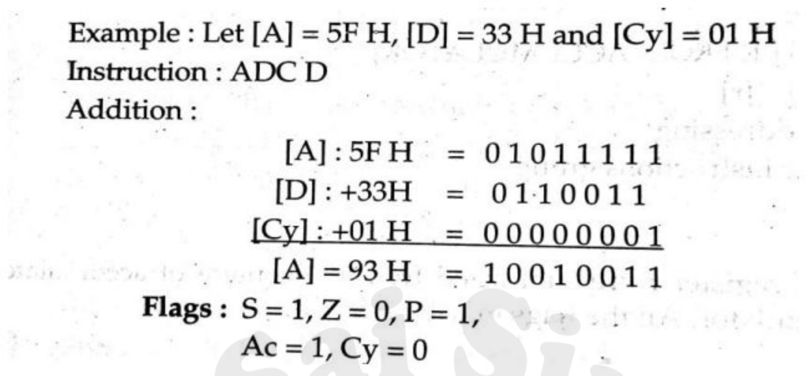


```
Addition : (A) :      EAH   = 1 1 1 0 1 0 1 0
            Data : + 15 H   = 0 0 0 1 0 1 0 1
                      FFH   = 1 1 1 1 1 1 1 1
          Flags :  S = 1,  Z = 0,  Ac = 0
          P = 1,   Cy = 0
After execution : [A] = FFH
```

4. ADC r   [Add register with carry]

Format: (A) ← (A) + (r) + (cy), Addressing : Register, Flags: All

This is one byte instruction. The content of register r & the content of the carry bit are added to the content of the accumulator. The result is placed in accumulator.


Example : Let [A] = 5F H, [D] = 33 H and [Cy] = 01 H
Instruction : ADC D
Addition :

```
           [A] : 5F H   =  0 1 0 1 1 1 1 1
          [D] : +33H    =  0 1 1 0 0 1 1
         [Cy] : +01 H   =  0 0 0 0 0 0 0 1
          [A] = 93 H    =  1 0 0 1 0 0 1 1
Flags :  S = 1, Z = 0, P = 1,
         Ac = 1, Cy = 0
```

5. ADC M [add memory with carry] - (with Carry)

Format: (A) ← (A) + ((H)(L)) +n(cy), Addressing : Immediate, Flags: All

This is 1 byte instruction. The content of memory location whose address is contained in H & L registers & content of the CY flag are added to accumulator. The result is placed in accumulator.

E.g. let [HL] = F000 H = [F000]= 05 h, [A] = 35H, [Cy] = 00H ,

Instruction: ADC M

After execution: [A] = 3A H

Flag: S=0 , Z= 0 , Ac= 0, P=1, Cy= 0

6. ACI data [add Immediate with carry] - (with Carry)

Format: (A) ← (A) + (byte 2) + (cy), Addressing : Immediate, Flag: All

This is 2 byte instruction. The content of second byte of the instruction and the content of cy flag are added to the contents of accumulator. The result is placed in accumulator.

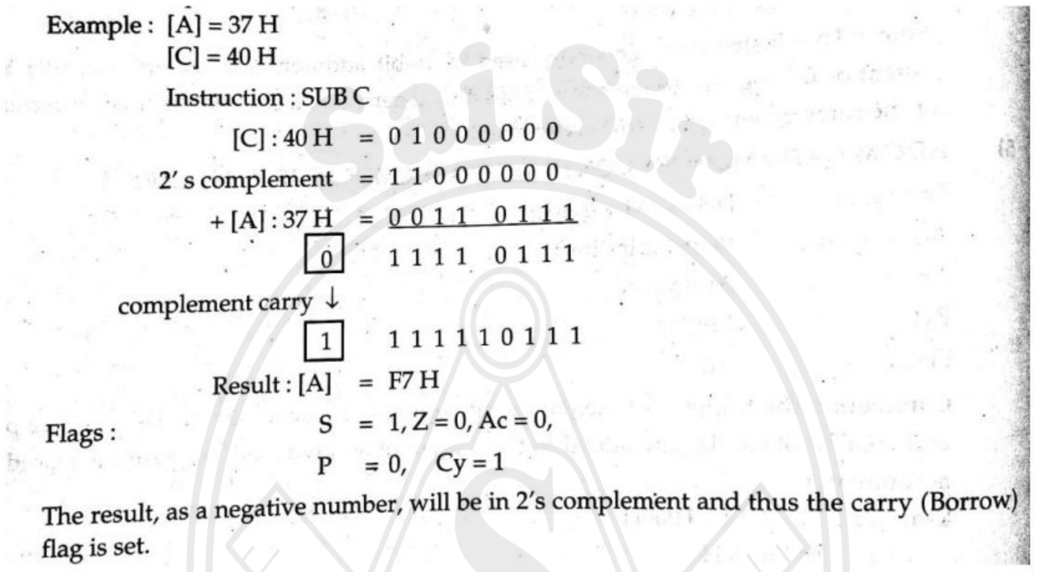
E.g. let [Cy] = 1 H, [A] = 05 H

Instruction: ACI 55H
After execution: [A] = 5B H

7. SUB r [subtract register]

Format: (A) ← (A) – (r), Addressing: Register, Flags: All

This is one byte instruction. The content of register r is subtracted from the content of accumulator. The result is placed in accumulator.



```
Example : [A] = 37 H
          [C] = 40 H
          Instruction : SUB C
                [C] : 40 H  = 0 1 0 0 0 0 0 0
          2' s complement   = 1 1 0 0 0 0 0 0
              + [A] : 37 H  = 0 0 1 1  0 1 1 1
                      [0]     1 1 1 1  0 1 1 1
       complement carry ↓
                      [1]     1 1 1 1 1 0 1 1 1
            Result : [A]    = F7 H
Flags :                S    = 1, Z = 0, Ac = 0,
                       P    = 0,  Cy = 1
```

The result, as a negative number, will be in 2's complement and thus the carry (Borrow) flag is set.

8. SUB M [subtract memory from accumulator]

Format: (A) ← (A) – ((H)(L)), Addressing : reg. indiret, Flags: All

The content of memory whose address is contained in the H & L registers is subtracted from the content of accumulator. The result is placed in accumulator.

E.g. [HL] = 2500 H, [2500] = 05 H, [A] = 07H

Instruction: SUB M

After execution: [A] = 02 H

9. SUI data [subtract immediate]

Format : (A) ← (A) – (data), Addressing : Immediate, Flags: All

The content of second byte of the instruction is subtracted from the content of the accumulator. The result is placed in accumulator.

E.g. let [A] = 1F H

Instruction: SUI 1F H

After execution: [A] = 00H

10. SBB r [subtract register with borrow]

Format: (A) ← (A) – (r) – (cy), Addressing : Register, Flags: All

The content of register r and carry bit are subtracted from content of accumulator. This is one byte instruction.

Example : [A] = 37 H
[B] = 3F H
[Cy] = 01 H
Instruction : SBB B

```
              [B] = 3 F
          Borrow :      + 1
                 ______________
             40 H = 0 1 0 0 0 0 0 0
2 ' s complement of 40 H

                  = 1 1 0 0 0 0 0 0
            + [A] = 0 0 1 1 0 1 1 1
           ______________________
              [0]   1 1 1 1 0 1 1 1
  Complement carry :
              [1]   1 1 1 1 0 1 1 1
  Result :    [A]   = F7H
```

11. SBB M [subtract memory with borrow]

Format: (A) ← (A) – [(H)(L)] – (cy), Addressing: reg.indirect, Flags: All

The content of memory location pointed to by HL register is subtracted from content of accumulator. The carry bit cy is also subtracted from the contents of accumulator. The result is placed in accumulator.

This is 1 byte instruction.

E.g. let [H-L] = 2500 H, [2500] = 05H, [A] = 07H & [Cy] = 0

Instruction: SBB M

After execution: [A] = 02 H

12. SBI data [subtract immediate with borrow]

(A) ← (A) – (byte 2) – (cy), addressing : immediate.

The content of the second byte of the instruction and the contents of cy flag are both subtracted from the accumulator. The result is placed in accumulator.

E.g. let [A] = 32 H, [Cy] = 1H

Instruction: SBI 31H

After execution: [A] = 0

13. INR r [increment register]

Format: (r) ← (r) + 1, addressing : Register, Flags: S, Z, P, AC

The content of register r are incremented by one. This is one byte instruction.the register r can be A, B, C, D, E, H, L.

All flags except cy are affected

E.g. let [B] = FF H

Instruction: INR B

After Execution: [B] = 00H Flags: S= 0, P=1, Ac= 0, Cy=0, Z=1

14. INR M [increment register]

Format: [(H) (L)] ← [(H)(L)] + 1 , addressing : reg. indirect,

The content of memory location whose address is contained in H & L register is incremented by one.

All flag except carry flag are affected.

E.g. [H-L] = 2500 H- [2500] = 04H

Instruction: INR M , After execution: [2500] = 05H

15. <u>INX rp   [increment register pair]</u>

Format: [(H) (L)] ← [(H) (L)] + 1,   addressing : register, Flags: None

The content of register pair rp is incremented by 1. No condition flags are affected.

Possible Combination : BC, DE, HL, SP

E.g. [H-L] = D000 H

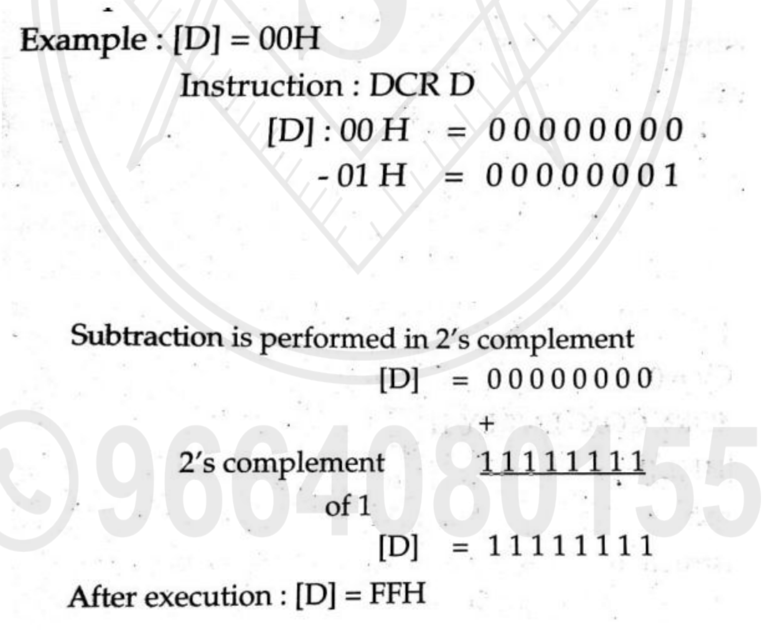
Instruction: INR M

After execution: [HL] = D001H

16. <u>DCR r   [Decrement register]</u>

Format: (r) ← (r) - 1 , addressing : register, Flags: S,Z,P,Ac except Cy

the content of register r is decremented by 1. All flags except cy are affected.



```
Example : [D] = 00H
            Instruction : DCR D
                   [D] : 00 H  =  0 0 0 0 0 0 0 0
                       - 01 H  =  0 0 0 0 0 0 0 1

    Subtraction is performed in 2's complement
                          [D]  =  0 0 0 0 0 0 0 0
                                   +
            2's complement        1 1 1 1 1 1 1 1
                   of 1
                          [D]  =  1 1 1 1 1 1 1 1
    After execution : [D] = FFH
```

17. DCR M [Decrement memory]

Format: [(H) (L)] ← [(H) (L)] – 1, addressing : reg. indirect, flags: S, Z, Ac, P except Cy

The content of memory location whose address is contained in the H & L registers is decremented by 1

All condition flag except cy are affected.

E.g. [H-L] = D000 H- [D000] = 2AH

Instruction: INR M

After execution: [D000] = 29H

18. DCX rp [decrement register pair]

Format: [(rh) (rl)] ← [(rh) (rl)] - 1, addressing : register. Flags: None

The content of register pair rp is decremented by 1. No condition flags are affected.

Possible Combination : BC, DE, HL, SP

E.g. [DE] = D000 H

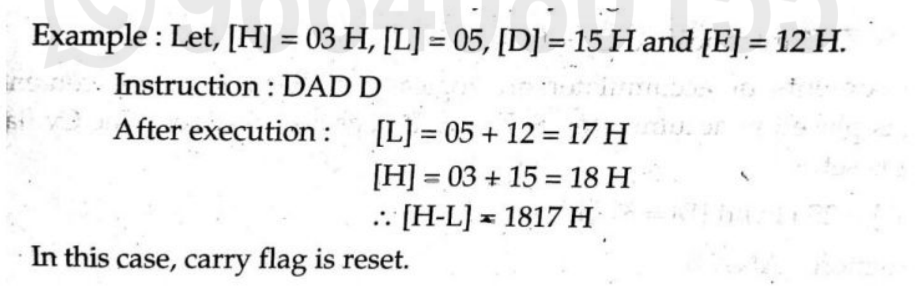
Instruction: DCR D

After execution: [DE] = CFFFH

19. DAD rp [add register pair to H & L]

Format: (H) (L) ← (H)(L) + (rh)(rl), addressing : register Flags: Cy

The content of register pair rp is added to the content of the register pair H & L. the result is placed in register H & L. only cy flag is affected.



Example : Let, [H] = 03 H, [L] = 05, [D] = 15 H and [E] = 12 H.

Instruction : DAD D

After execution : [L] = 05 + 12 = 17 H

[H] = 03 + 15 = 18 H

∴ [H-L] = 1817 H

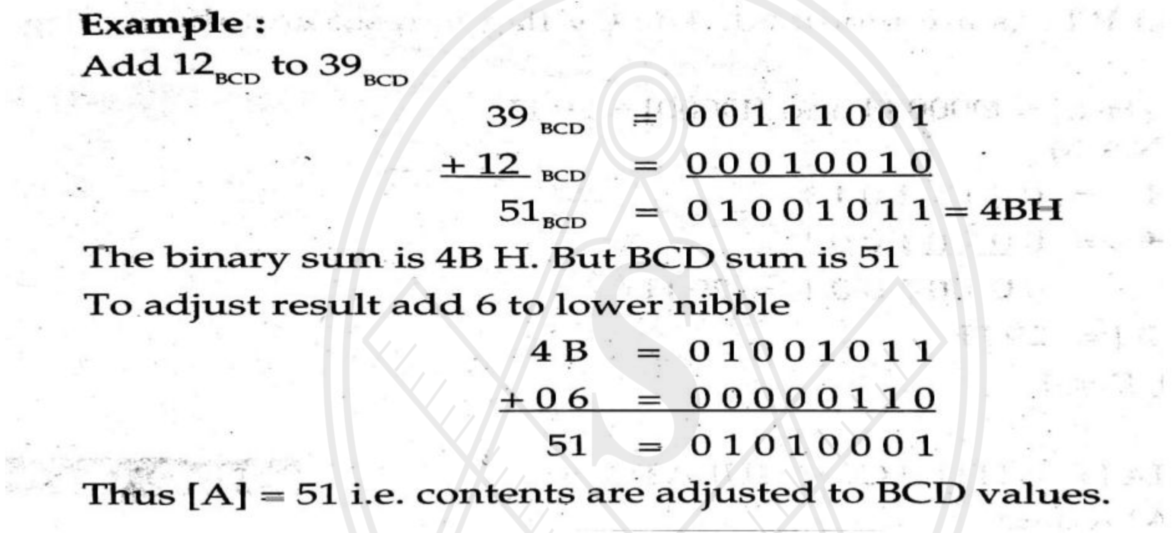
In this case, carry flag is reset.

20. DAA [decimal adjust accumulator]

The eight bit number in the accumulator is adjusted to form two four – bit binary-coded-decimal (BCD) digits by the following process.

If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.

If the value of the most significant 4 bits of the accumulator is now greater than 9, or of the cy flag is set, 6 is added to the most significant 4 bits of the accumulator. All flags are affected.



Example :

Add 12_{BCD} to 39_{BCD}

$$\begin{array}{rcl} 39_{BCD} & = & 0\,0\,1\,1\,1\,0\,0\,1 \\ \underline{+12_{BCD}} & = & \underline{0\,0\,0\,1\,0\,0\,1\,0} \\ 51_{BCD} & = & 0\,1\,0\,0\,1\,0\,1\,1 = 4BH \end{array}$$

The binary sum is 4B H. But BCD sum is 51

To adjust result add 6 to lower nibble

$$\begin{array}{rcl} 4B & = & 0\,1\,0\,0\,1\,0\,1\,1 \\ \underline{+06} & = & \underline{0\,0\,0\,0\,0\,1\,1\,0} \\ 51 & = & 0\,1\,0\,1\,0\,0\,0\,1 \end{array}$$

Thus [A] = 51 i.e. contents are adjusted to BCD values.

➢ LOGICAL GROUP:

1. ANA r [AND register]

Format: (A) ← (A) ^ (r), Addressing : register. Flags: S, Z, P are modified Cy= 0 , Ac = 1

The content of register r is logically AND'ed with the content of accumulator. The result is placed in accumulator. The cy flag is cleared and AC is set. This is one byte instruction.

Example : Let, [A] = 25 H and [B] = 31 H

```
        Instruction :  ANA B
                 [A] : 25 H   =  0 0 1 0 0 1 0 1
           AND [B] : 31 H     =  0 0 1 1 0 0 0 1
                                 0 0 1 0 0 0 0 1  =  21 H
    After execution : [A] =  21 H
              Flags :   S  =  0, Z = 0, P = 1,
                       Ac  =  1, Cy = 0
```

2. ANA M [AND Memory]

Format: (A) ← (A) ^ ((H) (L)), Addressing : reg. indirect, Flags: S, Z, P are modified Cy= 0 , Ac = 1

The content of memory location whose address is contained in the H & L registers is logically ANDed with the content of accumulator. The result is placed in the accumulator.

The cy flag is cleared and AC is set. The is one byte instruction.

Example : Let [A] = 3B H, [H-L] = D000 H and [D000] = 29 H

```
        Instruction : ANA M
                 [A] : 3B H   =  0 0 1 1 1 0 1 1
                AND 29 H      =  0 0 1 0 1 0 0 1
                                 0 0 1 0 1 0 0 1  =  29 H
    After execution : [A] =  29 H
    Flags : S = 0, Z = 0, P = 0
    Ac = 1, Cy = 0
```

3. ANI data [AND immediate]

Format: (A) ← (A) ^ (byte 2), addressing : immediate, , Flags: S, Z, P are modified Cy= 0 , Ac = 1

The content of the second byte of the instruction is logically ANDed with the contents of the accumulator. The result is placed in accumulator. The cy flag is cleared and AC is set.

E.g. let [A] = 11H

Instruction: ANI 11H , After execution: [A] = 11H

4. ORA r [OR register]

Format: (A) ← (A) V [r], addressing : register, flags: Z, S, P are modified, Ac & Cy are reset

The content of the register r is inclusive OR'd with the content of accumulator. The result is placed in accumulator. The CY & AC flags are cleared.

```
Example : Let [A] = 29 H and [B] = 35 H
          Instruction : ORA B
                 [A] : 29 H  =  0 0 1 0 1 0 0 1
              OR [B] : 35 H  =  0 0 1 1 0 1 0 1
                                0 0 1 1 1 1 0 1 = 3D H
      After execution : [A] =   3D H
Flags : S = 0, Z = 0, P = 0, Ac = 0, Cy = 0
```

5. ORA M [OR memory]

Format: (A) ← (A) V [(H) (L)], addressing : reg. indirect, Flags: Z, S, P are modified, Ac & Cy are reset

The content of memory location whose address is contained in the H & L registers in inclusive-OR'd with the content of the accumulator. The result is placed in accumulator. The CY & AC flags are cleared. This is one byte instruction.

Example : [A] = 03 H

[H-L] = D000H

[D000] = 81 H

Instruction : ORA M

```
       03 H  =  00000011
OR     81 H  =  10000001
       83 H  =  10000011
```

After execution: [A] = 83H, Flags: S= 0, Z=0, P=0, Cy=Ac=0

6. ORI data [OR immediate]

Format: (A) ← (A) V (byte 2), addressing : immediate, Flags: Z, S, P are modified, Ac & Cy are reset

The content of second byte of the instruction is inclusive OR'd with the content of the accumulator. The result is placed in accumulator. The CY & AC flags & cleared. This is 2 byte instruction.

Example : Let, [A] = 35 H

Instruction : ORI 99H

```
      [A] = 35 H  =  00110101
OR        99 H    =  10011001
                     10111101 = BD H
```

After execution : [A] = BDH

Flags : S = 1, Z = 0, P = 1, Ac = 0, Cy = 0

7. XRA r [Exclusive OR register]

Format: (A) ← (A) V□ (r), addressing : register,

The content of the register r is exclusive OR'ed with the content of accumulator. The result is placed in accumulator. The CY & AC flags are cleared. This is one byte instruction.

Example : Let [A] = 25 H and [B] = 39 H

Instruction : XRA B

```
[A] : 25 H  =  0010  0101
[B] : 39 H  =  0011  1001
               ----------
               00011100 = 1C H
```

After execution : [A] = 1C H

Flags : S = 0, Z = 0, P = 0, Ac = 0, Cy = 0

Flags: Z, S, P are modified, Ac =0 , Cy=0

8. <u>XRA M [exclusive OR memory]</u>

Format: (A) ← (A) V [(H) (L)], Addressing : reg. indirect. Flags: Z, S, P are modified, Ac ,Cy are reset

The contents of memory location whose address is contained in the H & L registers is exclusive OR'ed with the content of the accumulator. The result is placed in accumulator. The CY & AC Flags are cleared. This is one byte of instruction.

Example : Let [A] = 77 H

[H-L] = D000H

[D000] = 56 H

Instruction : XRA M

```
[A] : 77 H     =  0111  0111
[D000] : 56 H  =  0101  0110
                  ----------
                  0010  0001
```

After execution : [A] = 21 H

Flags : S = 0, Z = 0, P = 1, Cy = 0, Ac = 0

9. <u>XRI data [Exclusive OR immediate]</u>

Format: (A) ← (A) V (byte 2), addressing : immediate. Flags: Z, S, P are modified, Ac =0 , Cy=0

The content of the second byte of the instruction is exclusive OR'ed with content of the accumulator. The result is placed in the accumulator. The CY & AC flags are cleared. This is two byte instruction.

Example : Let [A] = 5B H

Instruction XRI 35 H

```
[A] : 5B H   = 0 1 0 1 1 0 1 1
Data 35 H    = 0 0 1 1 0 1 0 1
               ---------------
               0 1 1 0 1 1 1 0 = 6E H
```

After execution : [A] = 6E H

Flags : S = 0, Z = 0, P = 0, Cy = 0, Ac = 0

10. <u>CMP r [Compare register with Accumulator]</u>

Format: (A) - (r) , addressing : register Flags: All

The content of a register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of subtraction.

The result of comparison is shown by setting flags as:

1. If A < R/ M/ 8 bit data → Cy = 1, Z=0
2. If A = R/ M/ 8 bit data → Z = 1, CY=0
3. If A > R/ M/ 8 bit data → Cy = 0; Z = 0

E.g. [A] = 15H & [H] = 57H

Instruction: CMP H

After execution: Cy=1, Z=0

11. <u>CMP M [Compare memory]</u>

Format: (A) – [(H) (L)] Addressing: reg indirect addressing, Flags: All

The content of memory location whose address is contained in the H & L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction.

The result of comparison is shown by setting flags as:

1. If A < R/ M/ 8 bit data → Cy = 1, Z=0
2. If A = R/ M/ 8 bit data → Z = 1, CY=0
3. If A > R/ M/ 8 bit data → Cy = 0; Z = 0

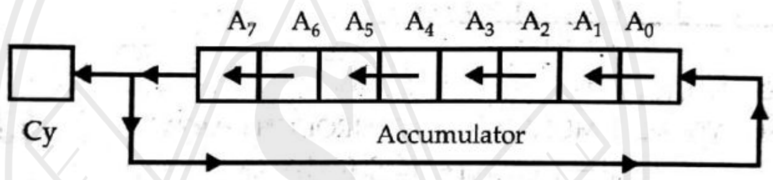
12. RLC [Rotate Accumulator Left]

Format: $(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$, Addressing: Implied , Flags: only Cy

(CY) ← (A_7)

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position.

Only the CY flag is affected. This is 1 byte instruction



Example : Let [A] = 93 H and [Cy] = 0

Instruction : RLC

Before instruction :

Cy	A_7							A_0
0	1	0	0	1	0	0	1	1

Accumulator

After execution : RLC

Cy	A_7							A_0
1	0	0	1	0	0	1	1	1

Accumulator

Thus [A] = 27 H and Cy = 1

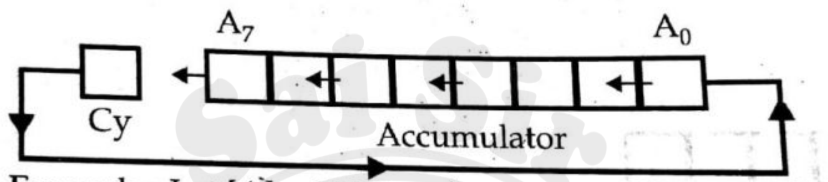
13. RAL [Rotate left through carry]

Format: $(A_{n+1}) \leftarrow (A_n)$; (CY) ← (A_7)

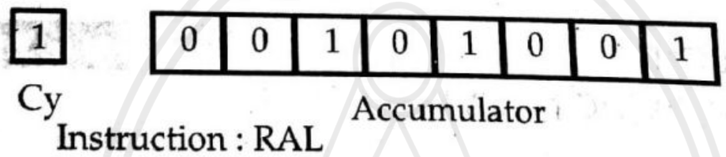
(A_0) ← (CY)

Addressing: Implied , Flags: only Cy

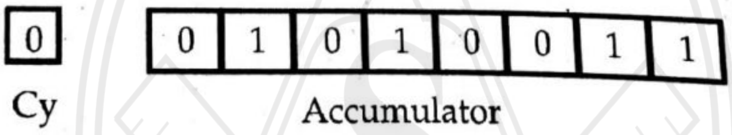
The content of accumulator is rotated left one position through the CY flag. The low – order bit is set equal to the CY flag and the CY flag is set to the value of shifted out of the high order bit. Only the CY flag is affected.



Example : Let [A] = 29 H,
[Cy] = 1 H



Instruction : RAL



Thus [A] = 53 H
[Cy] = 0

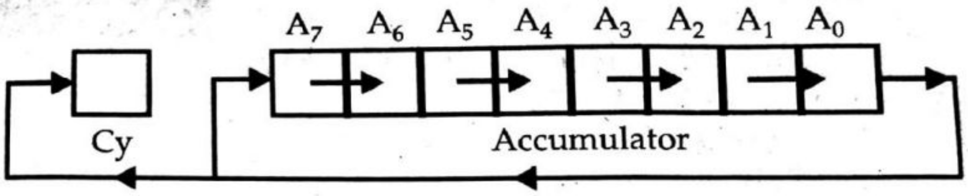
14. RRC [Rotate Right]

Format: $(A_n) \leftarrow (A_{n+1})$; $(A_7) \leftarrow (A_0)$

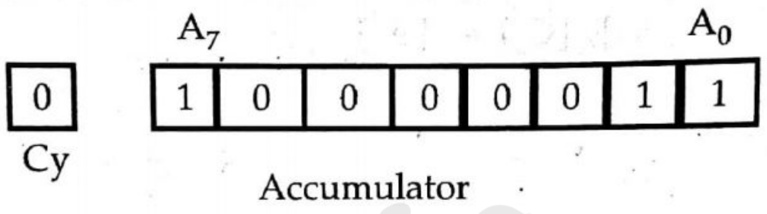
(CY) $\leftarrow (A_0)$, Addressing: Implied , Flags: only Cy

The content of accumulator is rotated right one position. The higher order bit and the CY flag are both set to the value shifted out of the low order bit position.

Only the CY flag is affected. This is 1 byte instruction.

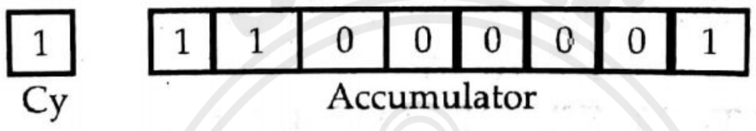


Example : [A] = 83 H [Cy] = 0



Instruction : RRC

After execution :



Thus [A] = C1H, [Cy] = 1

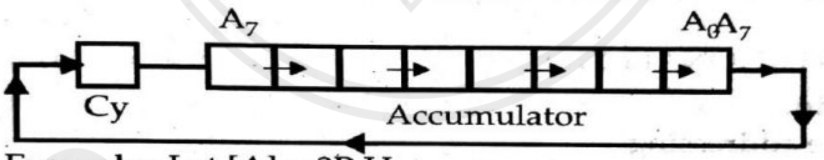
15. <u>RAR [rotate right through carry]</u>

$(A_n) \leftarrow (A_{n+1}); (cy) \leftarrow (A_0)$

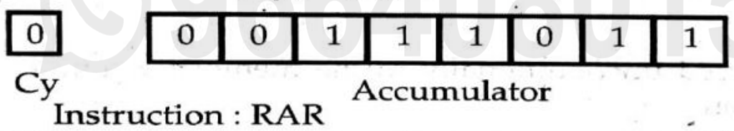
$(A_7) \leftarrow$ (cy) , Addressing: Implied , Flags: only Cy

The content of the accumulator is rotated right on position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit.

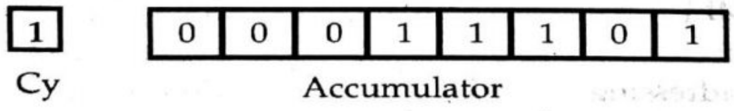
Only the cy flag is affected. This is one byte instruction.



Example : Let [A] = 3B H,
[Cy] = 0 H



Instruction : RAR



Thus [A] = 1D H
[Cy] = 1

16. <u>CMA (complement accumulator)</u>

Format: (A) ← ($\overline{A}$), addressing: Implied

The contents of accumulator are complemented (zero bit becomes 1, one bits become 0). No flags are affected.

This is one byte instruction.

E.g. If A = 0001 0101 (15 H)

Instruction: CMA , After execution: [A] = 1110 1010 (EA H)

17. <u>CMC [Complement carry]</u>

Format: (cy) ← ($\overline{cy}$) -

the carry flag is complimented, no other flags are affected.

E.g. [CY] = 1 H

Instruction: CMC. After execution: [CY] = 0H

18: <u>STC [set carry] →</u>

Format: [cy] ← 1 - Instruction sets carry flag to 1. No other flags are affected.

Addressing : implied

Branch Instructions

- There are two types of branching instructions. UNCONDITIONAL & CONDITIONAL.
- **In unconditional transfer no condition is tested.**
- In Conditional transfer the status of one of the flags is tested to determine where the branch is to be executed.

1. JMP addr [JUMP UNCONDITIONALLY]

- Format: (PC)← (BYTE 3)(BYTE 2)
- Addressing: Immediate addressing.   Flags: NONE
- This is 3 byte of instruction.
- Control is transferred to the instruction unconditionally whose address is specified in byte 3 and byte 2 of current instruction.
- E.g. JMP 2000H

2. Jcondition addr [Conditional JUMP]

- Format: (PC)<-(BYTE 3)(BYTE 2)
- Addressing: immediate addressing,   Flags: NONE
- This is 3 byte instruction.
- Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of current instruction based on the specified flag of the status register.
- The jump is taken only if the condition is true. The conditional jump instructions & conditions are given below:

Opcode	Address	Description	Flags
JC	16 bit address	Jump on carry	Cy = 1
JNC	16 bit address	Jump on no carry	Cy = 0
JP	16 bit address	Jump on positive	S = 0
JM	16 bit address	Jump on minus	S = 1
JZ	16 bit address	Jump on zero	Z = 1
JNZ	16 bit address	Jump on no zero	Z = 0
JPE	16 bit address	Jump on parity even	P = 1
JPO	16 bit address	Jump on parity odd	P = 0

3.CALL addr [UNCONDITIONAL SUBROUTINE CALL]

- Format: [(SP)-1]<-(PCH)
    - [(SP)-2]<-(PCL)
    - (SP)<-(SP)-2
    - (PC)<-(BYTE 3)(BYTE 2)
- Addressing: Immediate addressing/ reg. indirect addressing,
- This is 3 bytes of instruction.
- The program sequence is transferred to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack.
- Higher order byte of next instruction (PCH) is moved to memory location whose address is 1 less than the content of register SP
- Lower order byte of next instruction (PCL) is moved to memory location whose address is 2 less than the content of register SP. Content of SP is decremented by 2
- E.g. CALL 2000H

4. Condition addr

Format: [(SP)-1]<-(PCH)

[(SP)-2]<-(PCL)

(SP)<-(SP)-2

(PC)<-(BYTE 3)(BYTE 2)

The Conditional call instructions & conditions are listed below.

Opcode	Address	Description	Flags
CC	16 bit address	Call if carry	Cy = 1
CNC	16 bit address	Call if no carry	Cy = 0
CP	16 bit address	Call if positive	S = 0
CM	16 bit address	Call if minus	S = 1
CZ	16 bit address	Call if zero	Z = 1
CNZ	16 bit address	Call if no zero	Z = 0
CPE	16 bit address	Call if parity even	P = 1
CPO	16 bit address	Call if parity odd	P = 0

5. RET [Return]

Format: (PCL)<-[(SP)]

(PCH)<-[(SP)+1]

(SP)<-[(SP)+2]

Addressing: Register indirect., This is one byte instruction.

The program sequence is transferred from the subroutine to the calling program.

The program sequence is transferred from the subroutine to the calling program.

Content of memory location whose address is specified in register SP is moved to lower order 8 bits of register PC.

Content of memory location whose address is 1 more than content of register SP is moved to higher order 8 bits of register PC. Content of SP is incremented by 2

6. Rcondition [Conditional return]

Format: (PCL)<-[(SP)]

(PCH)<-[(SP)+1]

(SP)<-[(SP)+2]

Addressing: Reg. indirect, This is one byte instruction.

If the specified condition is true, the actions specified in RET are performed, otherwise the control continues sequentially.

RC	Return on Carry	CY=1
RNC	Return on no Carry	CY=0
RP	Return on positive	S=0
RM	Return on minus	S=1
RZ	Return on zero	Z=1
RNZ	Return on no zero	Z=0
RPE	Return on parity even	P=1
RPO	Return on parity odd	P=0

7. RST n(0-7) : [Restart]

Format: [(SP)-1]← (PCH)

[(SP)-2]← (PCL)

(SP)← SP)-2

(PC)← 8*(NNN)

Addressing: reg. indirect This is one byte instruction.

Higher order byte of next instruction (PCH) is moved to memory location whose address is 1 less than the content of register SP

Lower order byte of next instruction (PCL) is moved to memory location whose address is 2 less than the content of register SP. Content of SP is decremented by 2

The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations.

Instruction	Restart Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

8. PCHL [jump H & L indirect – move H & L to PC]

Format: (PCL)<-(L)

(PCH)<-(H)

Addressing : register. This is one byte instruction.

The contents of registers H & L are copied into the program counter. The contents of H are placed as the high-order byte of register PC and the contents of L as the low order byte of register PC.

E.g. let H = 25H & L = 39H

Instruction: PCHL

After execution: [PC] = 2539 H

After execution of PCHL instruction the control will be transferred to memory location 2539 H

Machine Control Group of Instructions:

These include:

A. Interrupt control and general machine operations

B. Stack Operations.

A. Interrupt control and general machine operations

Following is the table showing the list of Control instructions with their meanings.

Opcode	Operand	Meaning	Explanation
NOP	None	No operation	No operation is performed, i.e., the instruction is fetched and decoded.
HLT	None	Halt and enter wait state	The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.
DI	None	Disable interrupts	The interrupt enable flip-flop is reset and all the interrupts are disabled except TRAP.
EI	None	Enable interrupts	The interrupt enable flip-flop is set and all the interrupts are enabled.
RIM	None	Read interrupt mask	This instruction is used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
SIM	None	Set interrupt mask	This instruction is used to implement the interrupts 7.5, 6.5, 5.5, and serial data output.

B. Stack Operations.

These instructions allows for data storage and retrieval from the stack.

I. PUSH: Push register pair on stack

In 8085 Instruction set, **PUSH rp** instruction stores contents of register pair **rp** by pushing it into two locations above the top of the stack. rp stands for one of the following register pairs.

```
rp = BC, DE, HL, or PSW
```

It occupies only 1-Byte in memory.

PSW: Program Status Word

Mnemonics, Operand	Bytes
PUSH B	1
PUSH D	1
PUSH H	1
PUSH PSW	1

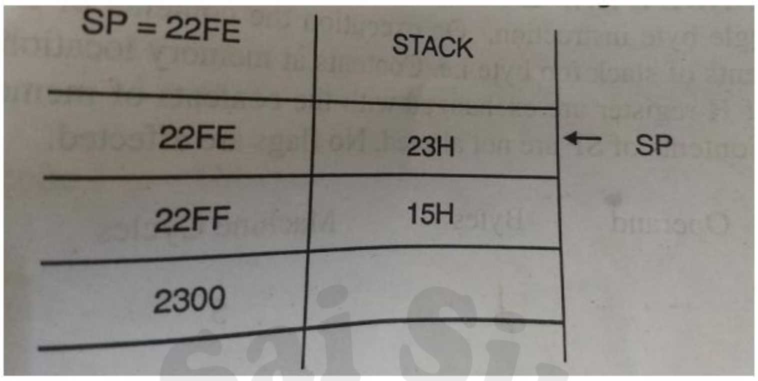
Example:

PUSH D

Will push contents of DE pair.

Let, D= 15H and E= 23H and SP=2300H

After execution of PUSH D, we have the following contents in SP and stack.



2. POP:

In 8085 Instruction set, with the mnemonic **POP**, we can pop out 2-Bytes from the top of the stack through **rp** i.e. register pair e.g. BC, DE, HL or AF. Here AF is a register pair formed with Flag and Accumulator registers and also known as PSW (Processor Status Word). In PSW, Accumulator is the MS Byte, and Flags register is the LS Byte.

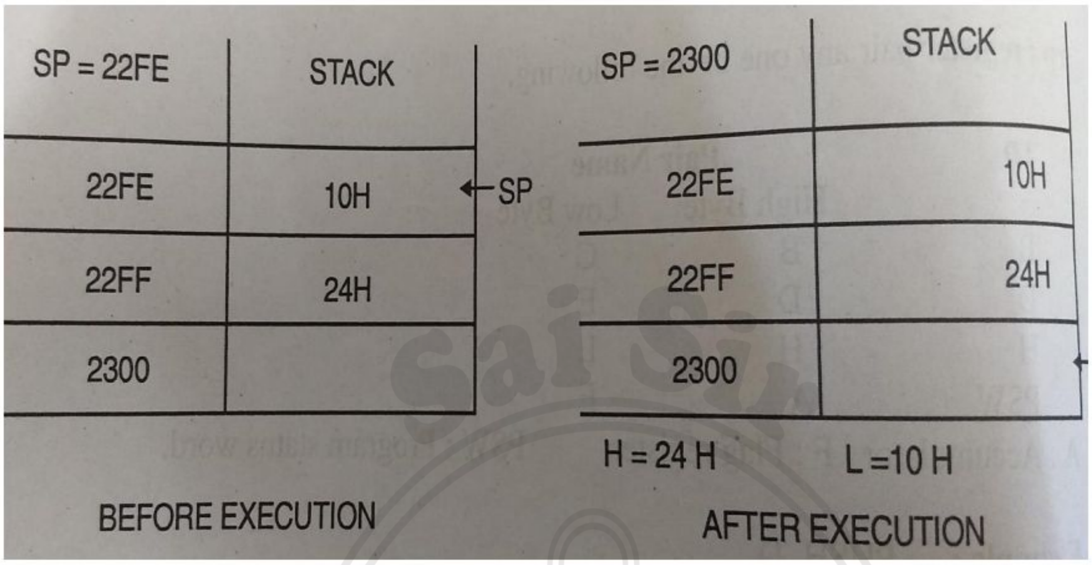
Mnemonics, Operand	**Bytes**
POP B	1
POP D	1
POP H	1
POP PSW	1

In the above mention Opcodes, 2-bits are occupied to mention the register pair. 2-bits can have 4 combinations. So 4 register pairs can be mentioned with POP. As mentioned earlier, they are BC, DE, HL and AF or PSW.

Example:

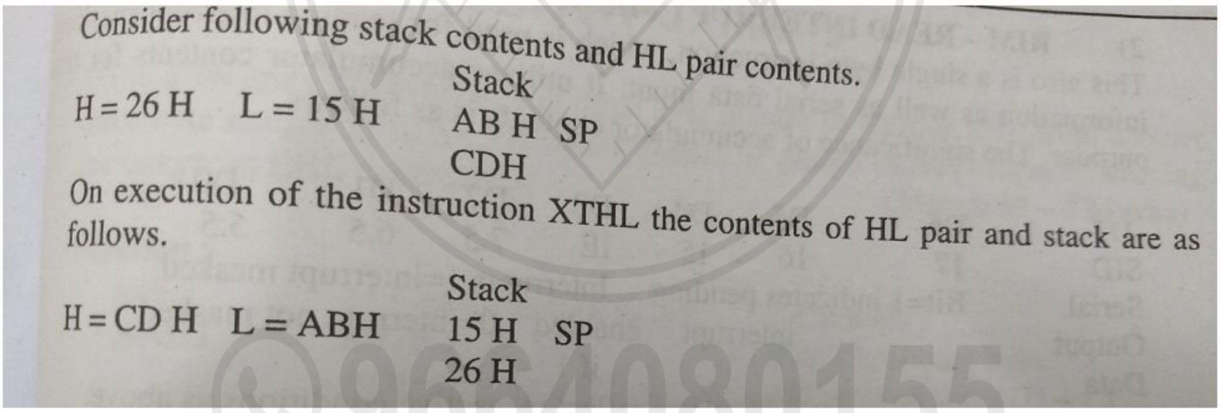
POP H

After execution of POP H, we have the following contents in SP and stack.



3. XTHL: Exchange H and L register with top of stack

In 8085 Instruction set, XTHL is a mnemonic that stands for "eXchange Top of stack with HL". This instruction exchanges the contents of the top two locations of the stack with the contents of register pair HL.



Consider following stack contents and HL pair contents.

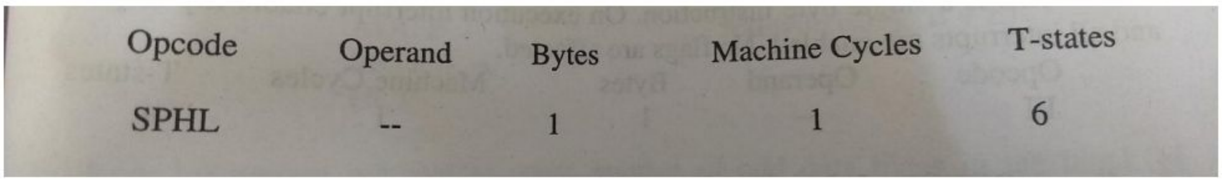
```
                          Stack
H = 26 H    L = 15 H      AB H  SP
                          CDH
```

On execution of the instruction XTHL the contents of HL pair and stack are as follows.

```
                          Stack
H = CD H   L = ABH        15 H   SP
                          26 H
```

4. SPHL: Copy H and L register to SP.

In 8085 Instruction set, SPHL is an instruction with the help of which Stack Pointer will get initialized with the contents of register pair HL. It is an indirect way of initializing the stack pointer. But it is not a very common and regularly usable instruction as well.



Opcode	Operand	Bytes	Machine Cycles	T-states
SPHL	--	1	1	6

CS – II, Chapter 3

Introduction to INTEL X – 86 family

Q. Explain in brief 8086 architecture.

A: The 8086 C.P.U. is divided into two independent functional parts:

I) Bus Interface Unit (B.L.U.)

I) Execution Unit (E.U.)

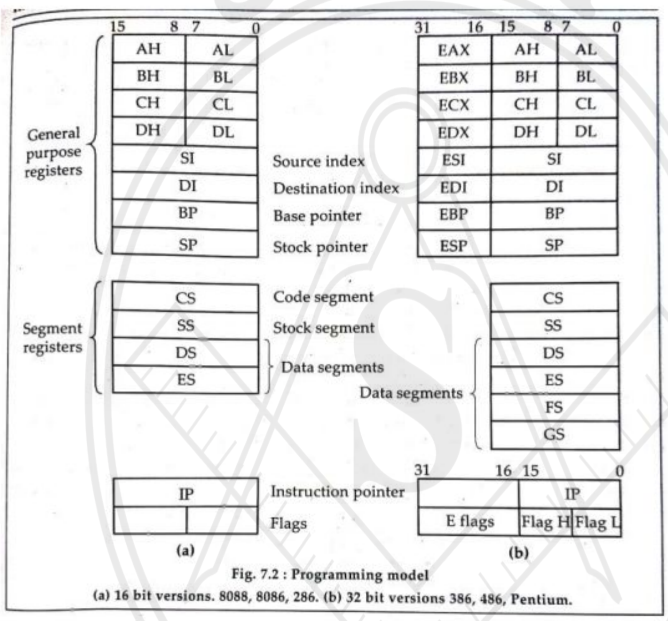


Fig. 7.2 : Programming model
(a) 16 bit versions. 8088, 8086, 286. (b) 32 bit versions 386, 486, Pentium.

I. **Bus Interface Unit (B.I.U.)** : The BIU sends address, fetches instruction from memory, reads data from ports/memory and writes data to ports/memory. In other word, B.I.U. handles transfer of data and address on buses for execution unit. It consists of following parts:

(a) **Queue:** To speed up execution of program, B.I.U. fetches 6 instruction bytes ahead of time from memory and stores them for E.U. in the FIFO register, called queue.

(b) **Segment registers:** B.I.U. contains following 16-bit segment registers:

i) Extra segment (E.S.) ii) Code segment (CS.) iii) Stack segment (S.S.) iv) Data segment (D.S.)

These 16-bit segment registers are used to store 16-bit starting address of memory segment.

(c) **Instruction pointer (I.P.):** The code segment (C.S.) register holds 16-bit starting address of segment, from which B.I.U. is fetching instruction code bytes. The. IP register holds 16-bit address of next code byte within code segment.

II. Execution Unit (E.U.): Execution unit of 8086 tells B.I.U. where to fetch instructions or data, decodes instruction and executes them. The following sections describe functional part of EU

(a) Flag register: Flag is a flip-flop, which indicate some condition. The 8086 has 16-bit flag register with 9-active flags.

(b) General purpose registers: 8086 has 8 general purpose registers, labelled AH, AL,

BH, BL, CH, CL, DH and DL. These registers can be used individually for temporary storage of 8-bit data. The AL register. is also called as accumulator.

These registers in certain pairs can be used as 16-bit registers. These pairs are AH and AL (AX), BH and BL (BX), CH and CL (CX), DH and DL (DX).

(c) Stack pointer: The 8086 allows us to set 64KB of memory as stack. The 16-bit Starting address of stack is stored in stack pointer.

Q. State advance features of X-86 microprocessor family

A: 1. it is capable of performing various computing functions and making decisions to change the sequence of program execution.

2. It is very powerful computing device.
3. Coprocessor – advanced microprocessor are supported by numeric coprocessor. It is separate CPU which perform arithmetic & trigonometric functions.
4. Operating system: microprocessor works with multiuser & multitasking operating system.
5. Virtual memory: advanced microprocessor supports virtual memory technique for storing large value of data.
6. Microprocessor includes special instructions & internal hardware which allow a programmer or to write software without knowing how much memory is available.

Q. Explain the following microprocessor in Intel X-86 family.

- ➢ **<u>8086:</u>** 8086 is a 16-bit microprocessor introduced by Intel in 1978
- It was designed by the desire to be e used as a CPU in microcomputer system. It's ALU, internal register can work with 16 binary bits at a time
- 8086 has 16-bit data bus and 20-bit address bus, so that it can address of physical memory of tourist 2^{20}= 1048576 = 1 Mbyte memory location
- The least significant 8 bits of the address bus are passed on same 8 lines as that of data bus this bus is known as a multiplexed bus

- In 8086 words are stored in two consecutive bytes. If the first bite of the word has even address then 8086 can read it in a single operation else it requires two operations
- This processor supports multiplication and division operations
- The 80186 is an improved version of 8046. In addition to 16 bit CPU, it has Programmable peripheral I/O devices integrated in the same package. Instruction set of 80186 is a superset of an instruction set of 8086

➢ **80286**: 80286 is a 16 bit microprocessor introduced in 1982 this advanced version of 8086 is specially designed to be used as CPU in multi user /multitasking Operating System

- 80286 has 16 bit data bus and 24 bit address bus
- In 1984, IBM introduced PC /AT (personal computer /advanced technology) version of its PC using 80286
- 286 was having real and protected modes of operation, in real mode the processor can address only one Mbyte of memory whereas in protected mode It can address 16 Mbytes of memory
- Another new feature was the ability to work up to 1 GB out of virtual memory and yet another feature was and Hardware multitasking
- The program written for 8086 can run on 80286 operating in its real address mode

➢ **80386:** the Intel 80386 is a 32-bit microprocessor introduced in 1985

- 80386 is a logical extension of 80286. It is more highly pipelined.
- The instruction set of 80386 is a superset of other members of the 8086 family.
- It has a 32-bit data bus and 32 Bit non multiplexed address bus. It can address of physical memory of 2^{32} that is 4 G Bytes. The 80386 memory management allows it to address to rest 2^{46} or 64 T bytes
- The 386 can be operated in one of the following memory management modes
- Page mode
- Non page mode
- When operated in page mode the 386 which is the paging unit that after the segment unit. The page units allows memory pages of 4K Kbytes each to be swapped in and out from the disk. In non-paged mode memory management unit operates very similar to the 286.
- Virtual address are represented with selected components and offset components as they are with 80286

- **80486:** Intel 80486 is a 32 bit microprocessor it was introduced in 1989.
  - It has 32 bit address bus and 32 Bit data bus
  - The 486 is basically a large integral circuit which contains a fast built in a Math Coprocessor, a memory management unit (MMU) and an 8 Kbyte cache memory
  - All 486 processor have 32 bit data bus SX version does not have any cheap numeric coprocessor
  - The 486 achieves its high speed all operations from its faster clock speeds internal pipeline architecture and the use of reduced instruction set computing (RISC) to speed up the internal microcode
  - 486 also has 486 DX2 and 486 DX4 versions with double and triple clock speed

Q. Explain main features of Pentium processor.

A: 1. Pentium is a 64 bit microprocessor introduced in 1993

2. It has 64 bit data bus and 32 Bit address bus. The use of super scalar architecture incorporate a duel pipe lined processor, which lets the Pentium process more than one instruction per clock cycle

3. The addition both of data and cold cache on chip is also a feature designed to improve processing speed

4. A new advanced computing technique used in Pentium is called the branch prediction, the Pentium makes an educated guess where the next instruction following a condition instruction will be. This prevents instruction cache from running dry during conditional instruction.

5. The Pentium has 64 bit data bus this means that it can perform data transfer with an external device twice as fast as processor with a 32 bit data bus

Q. Compare any four attributes of 80286 & Pentium microprocessor

Attributes	**80286**	**Pentium**
Data bus	16 bit microprocessor	64 bit microprocessor
Address bus	Address bus is 24 – bit hence can access 16 Mb memory	Address bus is 32 – bit hence can access 4 GB memory
Operating speed(MHz)	Clock frequencies 6 MHz – 20 MHz	Clock frequencies 50 MHz – 100 MHz
Memory management	External	Internal
Math co-processor	External	Internal

Q. Compare microprocessor 8085 & 8086.

Basis for Comparison	8085	8086
Microprocessor type	8-bit	16 - bit
Size of data bus	8 - bit	16 - bit
Size of address bus	16 - bit	20 - bit
Number of flags present	5 flags	9 flags
Cost	low	Comparatively high
Addressing mode	5	9

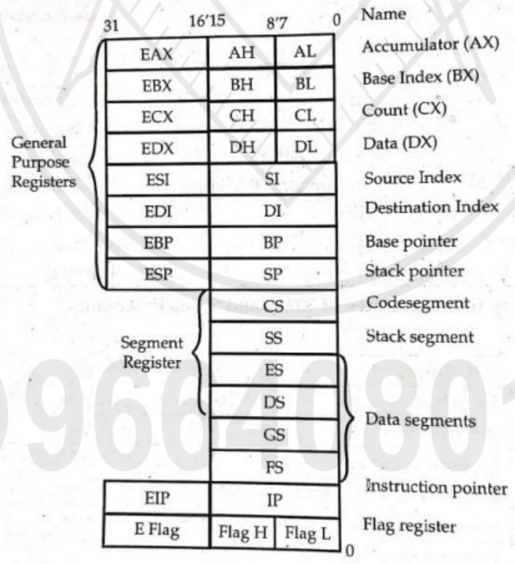
Q. Explain the programming model for 32 bit version of x-86 family with suitable diagram.

A: 1. The 8088 & 8086 defines the base programming model for the entire X – 86 family of advanced microprocessors.

2. The newer members of x-86 family have greater computing power because they are faster, they use 32-bit registers instead of 16-bit registers and they have advanced addressing techniques.

3. Following figure shows programming model for 32-bit version of X-86 family used in 32 bit microprocessor family i.e. 386, 486 & Pentium.

4. The programming of 32 bit version of X-86 family consist of 3 register groups.



5. The first group contains eight general purpose registers called EAX, EBX, ECX, EDX, ESI, EDI, ESP & EBP registers. Where E tells us that these registers have extended length. Each register can be addresses in 1, 8, 16 or 32 bit models. These register are used to store data during computations.

6. The second group of registers is the segment group. This group consists of code segment, stack segment & four data segment registers. The data segment registers are DS, ES, FS, and GS. These are 16 bit registers. These register manage operation with external memory. Address computations & data movement are performed here.

7. The third set of registers consist of instruction pointer and flag.

Q. explain advantages of the Pentium processor with respect to the following features:

A: **1. Dual pipelining:** The use of superscalar architecture incorporates a dual pipelining in Pentium t processor which lets Pentium to process more than one instruction per clock cycle and achieve a high-level performance.

2. On-chip caches: the data and code on-chip caches improve the processing speed of the Pentium processor.

3. Branch prediction: the advantage of branch prediction is that using it, the Pentium makes an educated guess where the next instruction following a conditional instruction will be this prevents instruction cache from running dry during conditional instruction

4. 64-bit data bus: Pentium has a 64-bit data bus which allows a higher speed of data transfer to it. The data transfer speed of Pentium is twice as fast as processor with 32 bit data bus

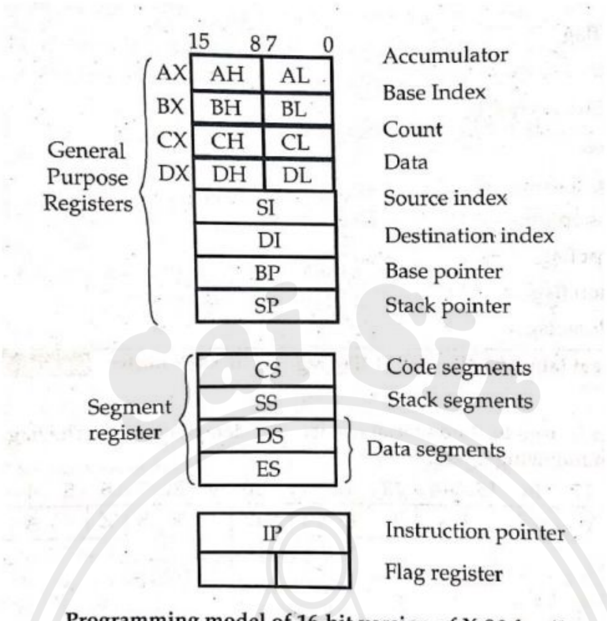
Q.. Draw neat labelled diagram and explain the programming model of 16-bit version of X-86 family with register set.

A. 1. The 8088 & 8086 defines basic programming model for X-86 family.

2. The 16 bit version for programming model is used in 16-bit microprocessors of X-86 family. i.e. in 8088, 8086 & 80286.

3. The 16-bit versions of programming model of X-86 family.

4. X-86 family consist of three register groups.



Programming model of 16-bit version of X-86 family

5. First group contains 8- general purpose registers called A, B, C, D, SI, Di, SP & BP registers. AL, BL, ………… Indicates lower bytes & AH, BH, …. Indicates higher bytes. The full 16 bit registers are referred as AX, BX, CX & DX where X stands for extended SI, DI, BP, SP registers are always treated as 16-bit registers. These are pointer registers because they are used to point locations within a segment.

6. The second group of registers. This group consist of code segment, stack segment & two data segment registers. Operation with external memory. Address computations & data movement are performed here.

7. The third set of registers consist of instruction pointer and flag.

Q. Draw a neat labelled diagram of flag register of X-86 family.

A: Flag register is used to store special results from data operations.

31…	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	**Ac**	**Vm**	**Rf**	**O**	**Nt**	**IO**	**PL**	**O**	**D**	**I**	**T**	**S**	**Z**	**-**	**A**	**-**	**P**	**-**	**C**

C – Carry flag (CF)
P – Parity flag (PF)
A – Auxilliary carry flag (AF)
D – String Direction flag (DF)
O – Overflow flag (OF)
IOPL – I/O privilege level

Z – Zero flag (ZF)	RF – Resume flag
S – Sign Flag (SF)	Nt – Nested task flag
T – Trap flag (TF)	AC – alignment check
I – Interrupt flag (IF)	Vm – virtual mode

a) 8085 microprocessor has 8-bit flag register (0 - 7) with 5 flags are active.

b) 8086/8088 microprocessor has 16-bit flag register, 8 to 11 bits are introduced and remaining bits are under fined. |

c) 80286 has also 16-bit flag register but 12 to 14 bits are added.

d) 80386 microprocessor has 32 bit flag register with 16 and 17th bits were introduced.

e) 80486 has 32 bit flag register with new 18th bit.

Remaining all positions in flag register are reserved

Introduction to Microcontroller

What is Microcontroller?

Ans: 1) A microcontroller is a complete microprocessor system, consisting of the microprocessor, the limited amount of ROM or EPROM, RAM and I/O ports, built on a single integrated circuit.

(2) A microcontroller is, in fact, a microcomputer, but it is called so because it is used to perform control functions.

(3) The designer of a microcontroller identify all the needs to build simple microprocessor system and puts as many as possible in single IC. e.g. Intel's 8048, 8051

State Any Six Features of 8051 Microcontroller.

Ans: 1) The 8051 microcontroller has an 8-bit ALU

2) The 8051 has a 4K byte ROM or EPROM.

3) The 8051 has 128 byte RAM

4) It has dual 16-bit timer event counter.

5) It has 32 I/O lines for four 8-bit I/O ports.

6) It can address 64 kB of program memory

7) It can address 64 kB of data memory.

8) It has powerful instruction set, consisting of 111 instructions

9) It has two external interrupts

10) The 8051 has clock up to 12-MHz frequency.

11) Full-featured serial port

Introduction to Microcontroller

Differentiate Between Micro-controller and a Micro-processor

Ans:

Microprocessor 8085	Microcontroller 8051
a) It is an 8-bit microprocessor	a) It is a 8-bit microcontroller
b) It provides seven 8-bit registers	b) It provides 34 &ndash;8 bit registers-A, B and 32 general purpose registers
c) 8-bit of data bus but ports are not available.	c) It has four ports P0-P3 for I/O.
d) Flag register is 8-bit and contains 5 flags.	d) Flag register is 8-bit and contains 9 flags.
e) Peripheral chips are required	e) Peripheral chips are not required

Explain the memory register map of Micro-controller 8051.

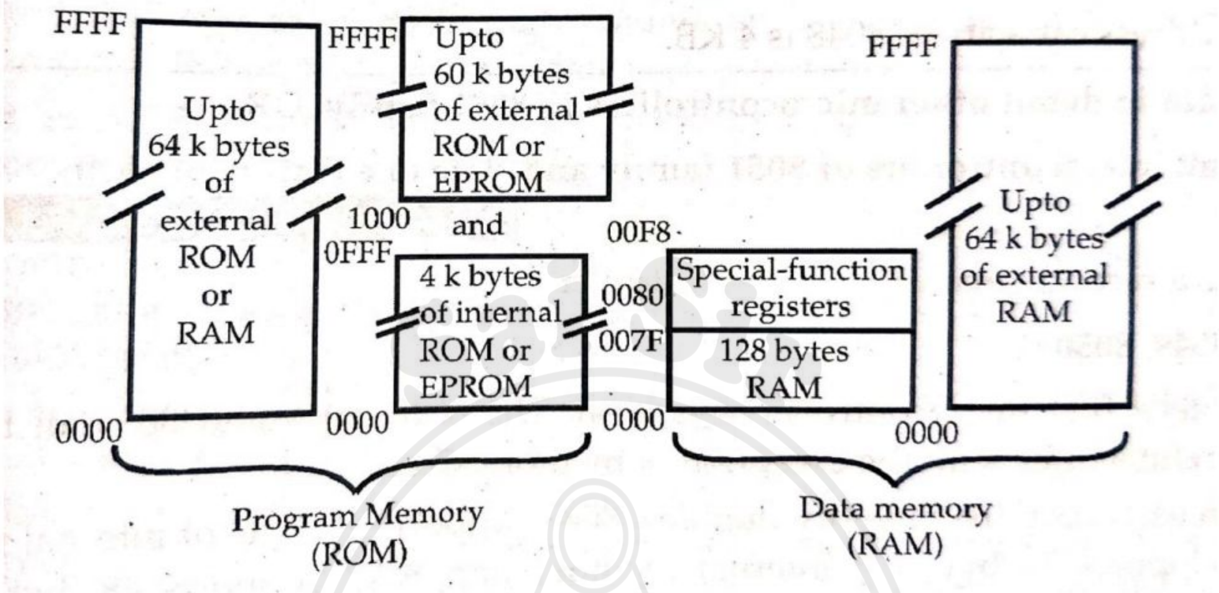
Ans: (a) The 8051 has two separate memory spaces:

i. Program memory space.

ii. Data memory space.

(b) The memory map for 8051 is shown in the following figure.

Introduction to Microcontroller


FFFF
Upto 64 k bytes of external ROM or RAM
0000
FFFF
Upto 60 k bytes of external ROM or EPROM
1000
and
0FFF
4 k bytes of internal ROM or EPROM
0000
Program Memory (ROM)
00F8
0080
Special-function registers
007F
128 bytes RAM
0000
FFFF
Upto 64 k bytes of external RAM
0000
Data memory (RAM)

(c) The program memory space is Read- only memory (ROM) space.

(d) This memory space is used for storing programs and variable data.

(e) It is possible to read program instructions from this space, but the processor cannot write data into this memory.

(f) All instruction fetches are taken from program memory space.

(g) The data memory space is read/write memory space.

(h) The processor can read data from this memory space and can write data to this memory space.

(i) It cannot execute instructions from this memory space. The 8051 internal RAM is in this memory space.

(j) The 128 bytes of internal RAM provide general read/write data storage. Some part of this is memory space.

(k) The 8051 has 22 special function registers which are not part of 128 bytes of RAM. They occupy memory space from 80H to F8H. These registers are used for their intended purpose.

Introduction to Microcontroller

(l) The 4K byte program memory can be expanded by additional 60k bytes, making it 64K bytes program memory. The data memory can also be expanded to 64K bytes.

(m) The 8051 can also be operated with common memory. In this case, 8051 only has 64K bytes of total external memory. In this configuration, 8051 can input block of data through its serial communications port load that data in memory and then execute that data as a program. This is called downloaded program. It is a very common technique used to change the program operating in a remote microprocessor- based controller.

Discuss the Micro-controllers in 8051 family.

Ans: 8051 is a second generation microcontroller.

(1) 8048, 8049, 8050

(a) Intel's first microcontroller was 8048. The 8048, 8049 and 8050 all have indentical architectures with the exception of memory size.

(b) In each case, the memory doubles. 8048 supports 1K byte of internal memory. 8049 supports 2k bytes of internal memory and 8050 supports 4K bytes of internal memory.

(c) 8048 has 64 bytes internal RAM, including 32 bytes of register/memory location. The 8049 and 8050 have a total of 128 and 256 bytes of RAM respectively.

(d) The microcontrollers are low cost products and hence are very popular.

(2) 8052

(a) 8052 is a simple expansion of 8051

(b) 8052 has 8K bytes of onboard ROM and 256 bytes of onboard RAM.

(c) 8052 allows programmers to write larger programs and that can use more data.

(d) The cost of 8052 is more than that of 8051.

(e) The 8052 also has one extra 16-bit counter-time. This counter-time gives more

Introduction to Microcontroller

(3) 8031 and 8032

(a) The alternative versions of 8051 and 8052 are 8031 and 8032.

(b) These devices do not have any on board ROM. It may use external ROM for program memory.

(c) These are excellent devices for prototyping and low-volume products

(4)8052 AH-BASIC

(a) Another form of 8052 is 8052 AH-BASIC. This special 8052 has BASIC programming language in ROM.

(b) Using BASIC instructions, a programmer can write instructions for this 8052 rather than assembly language.

State any three advanced features of 8052 microcontroller over 8051 microcontroller.

Ans: Advanced Features of 8052 Microcontroller over 8051 Microcontroller

(1) Microcontroller 8051 has 4k bytes of ROM, whereas 8052 has 8k bytes of onboard ROM or EPROM.

(2) Microcontroller 8051 has 128 bytes of RAM, whereas 8052 has 256 bytes of onboard RAM.

(3) Microcontroller 8051 has a dual 16-bit timer event counter whereas 8052 has an extra 16-bit timer event counter.

(4) The cost of microcontroller 8051 is less than that of microcontroller 8052.

(5) Both 8051 and 8052 are used in high volume applications and both allow us to write large program. But in 8052 we can write larger programs than that in 8051.

Write any two features of following Microcontrollers : 8048

Ans:(1) Intel's first microcontroller was 8048. The 8048, 8049 and 8050 all have identical architectures with the exception of memory size.

(2) In each case, the memory doubles. 8048 supports 1K byte of internal memory.

Introduction to Microcontroller

(3) 8048 has 64 bytes internal RAM, including 32 bytes of register/memory location.

(4) The microcontrollers are low cost products and hence are very popular.

Write any two features of following Microcontrollers : 8052

Ans:(1) 8052 is a simple expansion of 8051

(2) 8052 has 8K bytes of onboard ROM and 256 bytes of onboard RAM.

(3) 8052 allows programmers to write larger programs and that can use more data.

(4) The cost of 8052 is more than that of 8051.

(5) The 8052 also has one extra 16-bit counter-time. This counter- time gives more flexibility.

Write any two features of following Microcontrollers : 8031

Ans:(1) The alternative versions of 8051 and 8052 are 8031 and 8032.

(2) These devices do not have any on board ROM. It may use external ROM for program memory.

(3) These are excellent devices for prototyping and low-volume products.

Write any two features of following Microcontrollers : 8050

Ans: (1) Intel's first microcontroller was 8048. The 8048, 8049 and 8050 all have identical architectures with the exception of memory size.

(2) In each case, the memory gets doubles. 8050 supports 4K bytes of internal memory.

(3) It has 256 bytes of RAM.

(4) The microcontrollers are low-cost products and hence are very popular.

What is a Single Chip Computer ? State its advantages

Ans: Microcontroller is a Single Chip Computer. This is used for dedicated functions.

Introduction to Microcontroller

Advantages:

(1) It can be used as independent controllers in various machines.

(2) It includes all the essentials of a computer on a single chip like CPU, R/W memory, ROM it can be used as a microcomputer.

(3) It reduces the cost of a system than microprocessor based system. So it can be used in low cost products like toys.

State any six applications of Microcontrollers

Ans: (1) Microcontroller is a single chip microcomputer.

(2) They are used as independent controllers in machined or as slaves in distributed processing.

(3) They are used as machine tools, chemical processors, medical instrumentation and sophisticated guidance control.

(4) Some applications require simple timing and bit set/ reset functions; other requires high speed data processing capability.

(5) Many low cost products such as electronic toys, microwave ovens, VCRs are based on microcontrollers.

(6) A home security system, a tape deck and intelligent multimeter can also be built by using microcontroller.

Networking Technology

Explain in short six characteristics of Transmission Media.

Ans: Characteristics of Transmission Media:

(1) Bandwidth

(i) Bandwidth is the measure of the capacity of a medium to transmit data.

(ii) Data transmission rate is number of bits transmitted per second.

(iii) Bandwidth of a cable depends on cable length.

(2)Band usage

(i) The bandwidth is shared so that maximum usage is obtained.

(ii) There are two transmission modes, base band and broad band transmissions.

(iii) Base band devotes the entire capacity of the medium to one communication channel.

(iv) Broad band enables two or more communication channels to share the bandwidth of communication medium.

(v) The base band and broad band transmission modes are shown in following figure

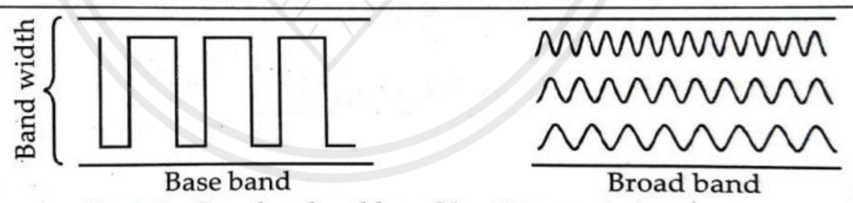


Fig. 9.2 : Base band and broad band transmission modes.

(3) Attenuation

(i) Attenuation is a measure of how much a signal weakens as it travels through a medium.

(ii) As signals pass through the medium, part of the signal is absorbed and makes the signal weak.

(4) Immunity from electromagnetic interference (EMI)

Networking Technology

(i) Electromagnetic interference consist of outside electromagnetic noise that distorts the signal in a medium.

(ii) EMI is interfering the signals and makes difficult for computers to decode the signal.

(5) Cost of media

(i) One major factor in purchase decision of any networking component is its cost.

(ii) For a new fast technology, cost is also more expensive.

(iii) Decision depends upon application and standard of the resources.

(iv) Therefore, the network designer must settle for something, which is cheaper and robust.

(6) Installation requirement

(i) Some transmission media requires skilled labor to install. This increases cost of network and it may cause certain delay.

(ii) Before installation we need to prepare actual physical layout of network.

Compare Twisted Pair Cable and Coaxial Cable

Ans: Twisted Pair Cable:

(1) It consists of a pair of wires or one or more pairs of two twisted copper wires insulation.

(2) This is inexpensive medium.

(3) EMI effect is maximum.

Coaxial Cable:

(1) It is a hallow cable with a solid copper at the center of the cable surrounded by plastic.

(2) Relatively expensive i.e. twice or thrice than twisted pair.

Networking Technology

Explain the Structure of Coaxial cable.

Ans: In this cable there are two conductors sharing a common axis.

The TV cable is a coaxial cable. These cables are used in LANs.

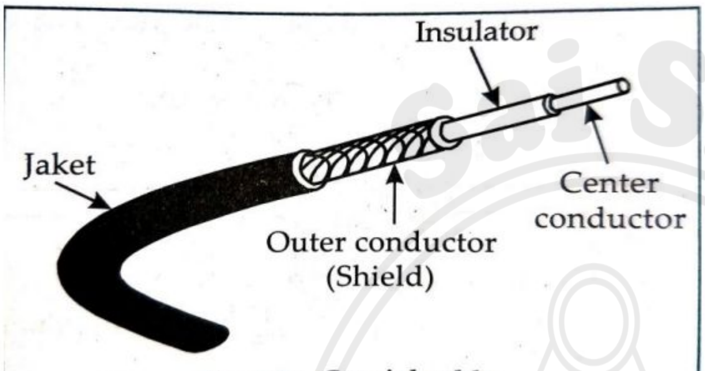


Fig. 9.5 : Coaxial cable

Types of coaxial cable:

1. Thinnet : thinnet is light and flexible cabling medium that is inexpensive & easy to install. Thinnet cable falls under the RG – 58 family, which has a 50-ohm impedance. Thinnet is 0.25 inches (6mm) in thickness.
2. Thicknet : thicknet is thicker than thin net. It is about 0.5 inches(13mm) in diameter. Since its thick it does not bend. It is difficult to work with thicknet. It can transmit a signal approximately 500 meters(1650 feet)

Two or more thinnet LANs can be connected by thicknet. It is also called as standard Ethernet cable. This cable is more expensive.

Coaxial cable is easy to install . This is robust & difficult to damage.

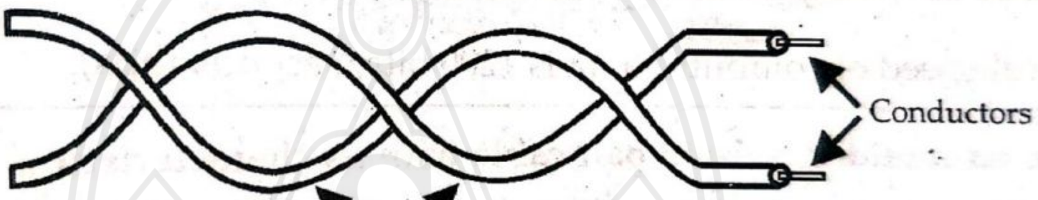
1. **Cost :** Thinnet cable is low cost cable. Thicknet is more costly.
2. **Capacity:** LANs employing coaxial cable have bandwidth in between 2.5Mbps (ARC net) to 10Mbps(Ethernet). Thicker coaxial cable offers higher bandwidth.
3. **EMI:** All copper media are sensitive to EMI. Shield increases resistance of cable to EMI. Coaxial cable radiate portion of their signal.

Networking Technology

Explain the Structure of twisted pair cable.

Ans:

- This is popular cable for all new networks. This is low cost cable.
- The telephone cable is an example of twisted pair cable
- Twisted pair cable consist of two strands of copper wire twisted together



Twisted pair cable

- There are two types of twisted pair cable – Shielded twisted- pair(STP)& unshielded twisted- pair(UTP).

Shielded twisted- pair(STP) cable:

1. Cost: the cost of coaxial cable is more than that of coaxial or unshielded twisted pair cable. Less than that of thick coaxial & fiber – optic
2. Installation : different network types have different installations.
3. Capacity : theoretical capacity 500Mbps, practically 155 mbps. The most common data rate for STP cable is 16 Mpbs.
4. Attenuation : all twisted pair cable have attenuation. This limits the length of cable. 100 meter limit is most common.

5. EMI characteristics : the shield in STP cable results in good EMI characteristics for copper cable, than that of coaxial cable.

UTP cable is available in 5 different categories.

- Categories 1 & 2: low data rates (below 4 mbps)
- Categories 3: data rates upto 10 mbps
- Categories 4: data rates upto 16 mbps
- Categories 5: data rates upto 100 mbps

1. Cost: UTP cable is least costly than any cable. The cost of category 5 cable is high
2. Installation: UTP cable is easy to install.
3. Capacity: up to 100mbps data can be achieved.
4. Attenuation: similar attenuation characteristics of that of other characteristics.
5. EMI: UTP cable does not have shield. It is more sensitive to EMI than coaxial or STP.

Explain the Structure of Fiber Optic Cable.

Ans: (1) The light wave can be efficiently conducted through transparent glass fiber cables known as optic fiber cables.

(2) The center conductor of this cable is a fibre that consists of highly refined glass or plastic.

(3) It is designed to transmit light signals with little loss.

(4) The fibre is coated with cladding or gel that reflects signals back into fibre to reduce signal loss. A plastic sheet protects the fibre from damage.

(5) The fibre optic cable is shown in the following figure.

Networking Technology

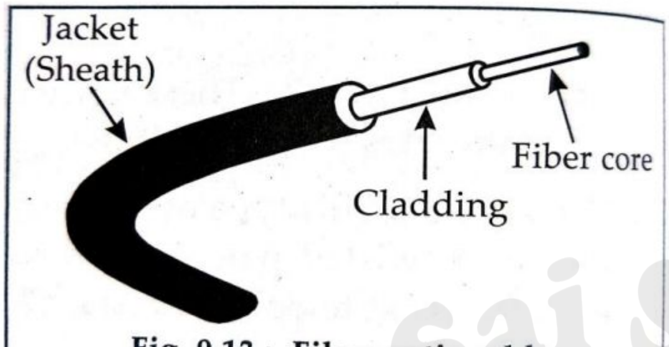


Fig. 9.13 : Fiber optic cable.

(6) The fibre optic cable is used in the optical transmission system.

(7) This cable has the extremely high bandwidth. It has zero sensitivity to EMI and runs over several kilometres.

(8) The characteristics of fibre optic cable are given below

Cost: The cost of fibre optic cable is more than that of coaxial cable and Twisted pair cable.

Installation: Fibre optic cable requires skilled installation. Every cable has minimum bend radius. They may get damaged if bent sharply Fibre optic cable cannot be stretched.

Capacity: Fibre optic cable supports high data rates (up to 2,00,000 MBPS), even with long-run cables. Fibre optic cable can transmit 100 MBPS for several kilometres.

Attenuation: Attenuation for fibre optic cable is much lower than co-axial cable and twisted pair cable. It can run to the larger distance.

EMI: Fibre optic cable does not use electrical signals to transmit data, therefore they are free from EMI. The data transfer in fibre optic cable has high security, as it cannot be detected by electronic wave dropping equipment.

What is Wireless Media ?

Ans: Transmission of waves take place in the electromagnetic (EM) spectrum. The carrier frequency of the data is expressed in cycles per second called hertz (Hz). Low frequency signals can travel for long distances through many obstacles but cannot carry a high bandwidth of date while high frequency signals can travel for shorter distances through few obstacles and carry a narrow bandwidth. Also the noise effect on the signal is inversely proportional to the power of the radio transmitter.

Discuss different types of Ethernet.

Ans: Ethernet Types :

10 BASE 5 : Speed 10 Mbps (10) and thick co-axial known as thicknet.Maximum cable net 500 mtrs.(5).

10 BASE 2 : Speed 10 Mbps (10) and thin co-axial cable known as thinnet.Maximum length 200 mtrs.(2)

10 BASE T : Speed 10 Mbps (10) and uses UTP twisted pair 100 mtrs.

10 BASE FL : Speed 10 Mbps(10) uses (FL).

100 BASE VG : Speed 100 Mbps (10) twisted pair. (VG-Voice Grade)

What is Transmission Media ?

Ans: (1) The pathways through which individual systems are connected in a network are called as transmission media.

(2) Transmission media makes transmission of electronic signals possible from one computer to another. These electronic signals are nothing but binary pulse (I/O).

Explain in Brief the Following Access Method : Contention

Ans: In contention, any computer in the network can transmit at any time (first come first served).

This system breaks down when two computers attempt to transmit at the same time. This is a case of the collision.

To avoid the collision, carrier sensing mechanism is used. Here each computer listens to the network before attempting to transmit. If a network is busy, it waits until network quits down. In carrier detection, computers continue to listen to the network as they transmit. It computer detects another signal that interferes with the signal it is sending, it stops transmitting. Both computers then wait the random amount of time and attempt to transmit.

Contention methods are most popular media access control method on LANs

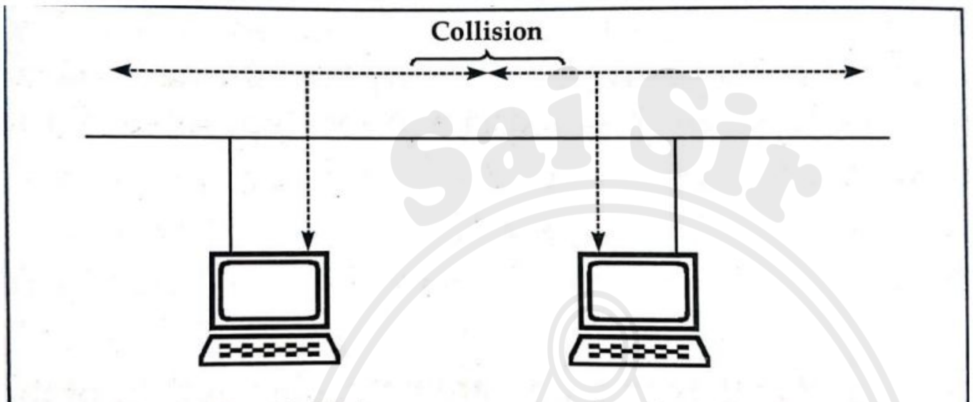


Fig. 9.14 : Collision on a contention based network.

Explain in Brief the Following Access Method: Token Passing

Ans: Token passing Token passing utilizes a frame called a token, which circulates around the network.

A computer that needs to transmit must wait until it receives the token. When the computer receives token, it is permitted to transmit.

When computer completes transmitting, it passes the token frame to the next station or token ring network.

Explain in Brief the Following Connectivity Devices : Repeater

Ans: A repeater is a hardware unit mostly used in Ethernet to extend.

A repeater reshapes and amplifies the signal from one Ethernet segment to another.

The figure shows to network with repeaters

Networking Technology

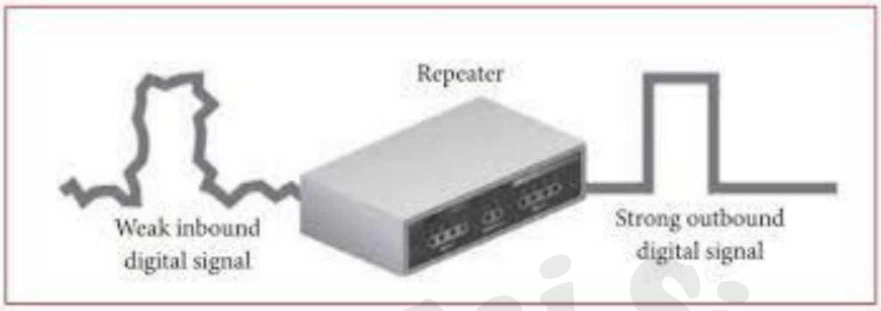


Figure 14.19 Repeater

A backbone cable runs vertically up in the building and a repeater is used to attach an Ethernet segment running on each floor of the office to the backbone cable.

No two Ethernet workstations can have more than two repeaters between them if they have to communicate reliably.

The main disadvantage of repeaters is that they repeat noise in the system.

A separate power supply is needed for repeaters.

Explain in brief the following connectivity devices: Router

Ans: Routers are internetwork connectivity devices. They are used to connect two topologically similar or dissimilar LANs. i.e. the LANs can be different e.g. they can be ethernet and token ring. Each LAN is logically separate and is assigned an address.

A router can use a network address to assist efficient delivery of a message. Delivering packets according to the logical network address to assist is called as routing. Routers perform routing.

Routers are intelligent. They can use algorithms to determine most efficient path for sending a packet to any given network.

Router are also be used to divide large, busy LANs into smaller segments.

Router is also employed to connect LAN to wide area network (WAN).

Routers are of two types:

1. Static routers

Networking Technology

2. Dynamic routers

Static routers do not determine paths, but you need to specify them. Dynamic routers have capacity to determine paths (routers).

Give Advantages of Fiber Optic Cable over an Electrical Cable

Ans:

(i) Fiber optic cable if cable is broke, still Data Transfer is possible, while in electrical cable it's not possible.

(ii) In fiber optics cable, Data transfer is optically using light rays while in electrical cable, data transfer is done by electricity.

(iii) In fiber optic cable, data rate is very high in terms of gbps, while in electrical cable, its data rate is very low in terms of mbps.

(iv) Cost of fiber optic cable is very high while, in electric at cable it's very low.

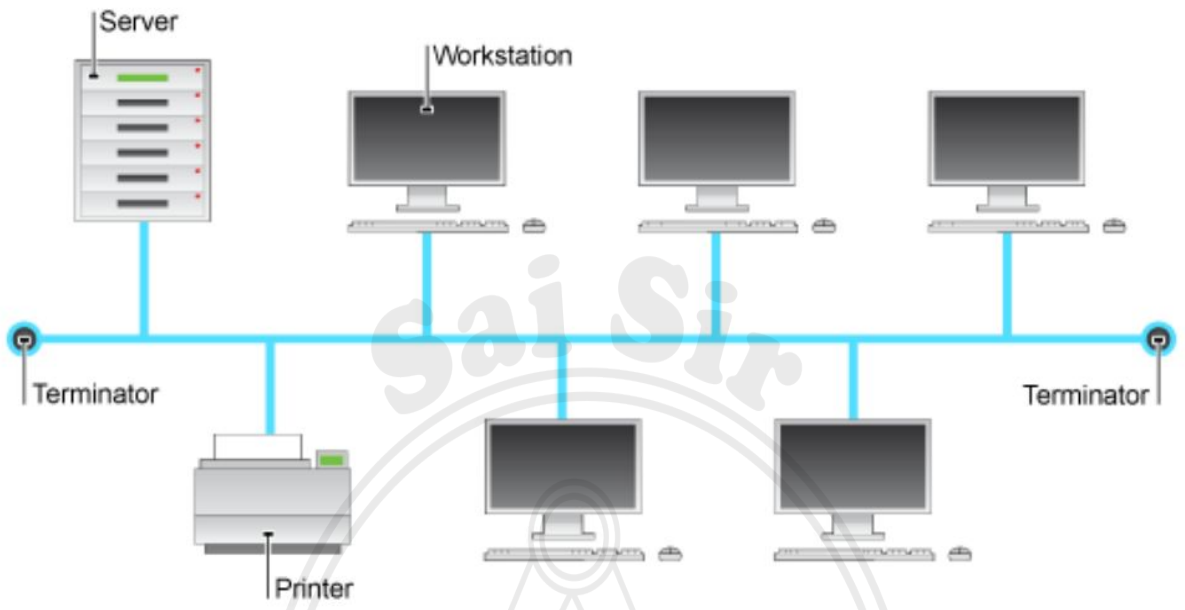
Explain in Short Bus Topology

Ans: BUS Topology :

(1) In a BUS physical topology, all the devices are connected to a common shared cable, called as backbone of the network.

(2) A BUS physical topology is shown in following figure

Networking Technology



(3) The bus is available for each node to send its data to each and every computer node.

Advantages:

(1) The bus system is much faster.

(2) The bus topology can be extended with sub branches to form another topology.

(3) Breakdown of any failure node does not affect other node's communication.

(4) Bus topology is widely used in LAN network.

Explain in short Ring Topology.

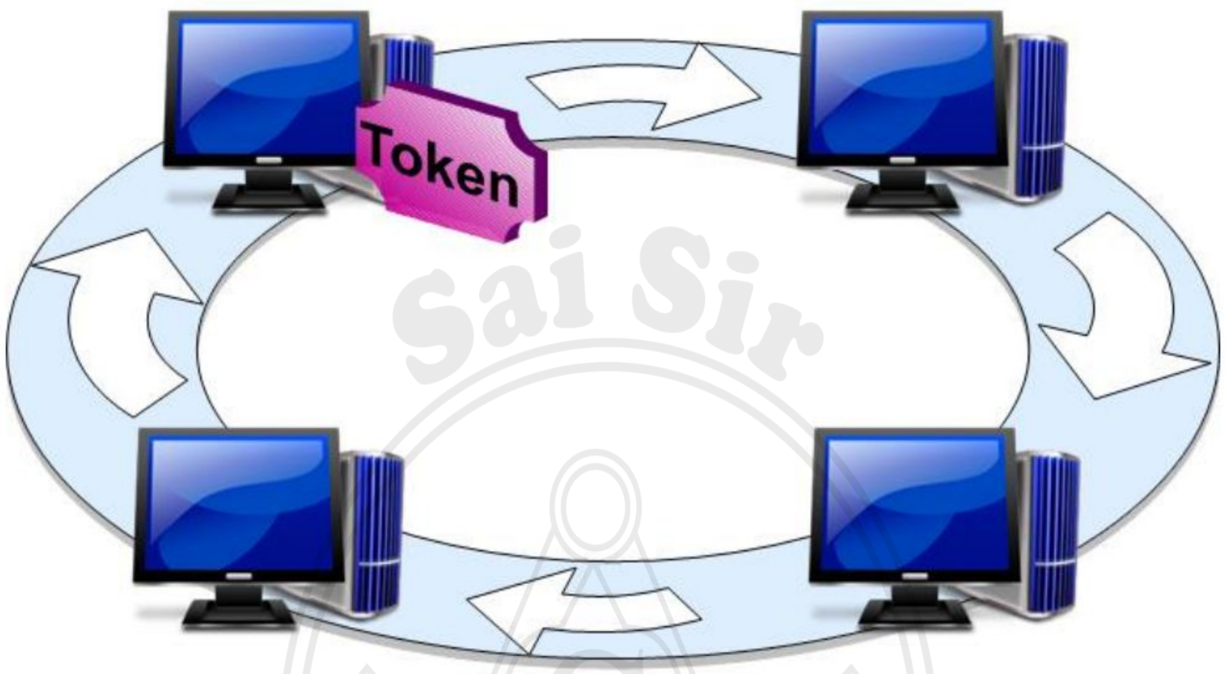
Ans: RING Topology:

(1) RING topologies are wired in a circle. Each node is connected to its neighbor on either side, and the data transmits along the ring in one direction only.

(2) Each device incorporates a receiver and a transmitter and serves as a repeater that passes the signal to the next device in the ring.

(3) The RING topology is as shown in the following figure:

Networking Technology



(4) RING topologies are suited for networks that use token passing access methods. The token passes around the ring, and the only node that holds the token can transmit data.

(5) This topology is always implemented as a logical topology. E.g. In a token ring network, the topology is physically a STAR topology. But logical topology is RING topology.

(6) The commonly used implementation for RING topology is a token ring at 4-16 MBPS.

Advantages:

(1) Cable failure affects limited users.

(2) Each node has equal access speed to the ring.

(3) Equal access for all users.

Disadvantages:

(1) Costly wiring is required for a RING topology.

Networking Technology

(2) Expensive adapter cards.

(3) Difficult connections

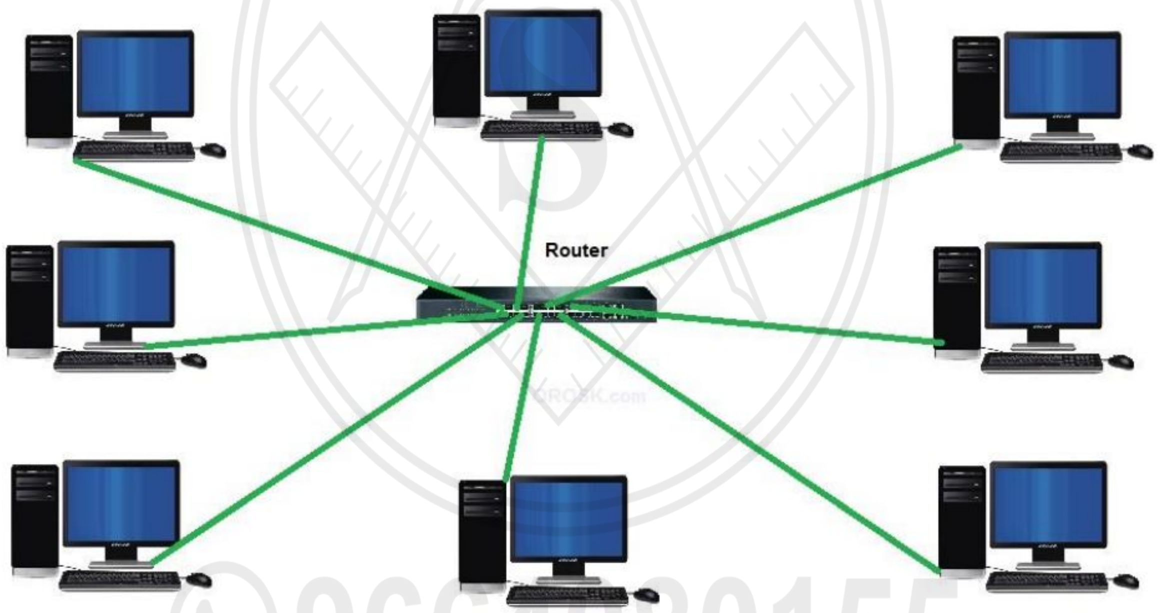
Explain STAR topology with diagram. Also give two advantages and disadvantages.

Ans: (1) In a STAR topology, all the workstations are connected to a central hub.

(2) The hub receives a signal from a workstation and routes it to the proper destination.

(3) STAR physical topology is often implemented to implement BUS or RING logical topology.

(4) A STAR topology is shown in the following figure:



Advantages:

(1) Adding a new workstation is easier than that in BUS or RING topology.

(2) The control is centralised due to use of a hub.

Disadvantages:

Networking Technology

(1) Hub failure affects all users.

(2) Hubs are slightly expensive.

(3) STAR topology requires more cabling than BUS or RING topology. Hence, it costs more.

Distinguish between LAN and WAN.

Ans:

WAN	LAN
a) A WAN (Wide Area Network) is the interconnection of LAN or MAN can be located entirely within a state, country or around the world.	A LAN (Local Area Network) is a group of computers interconnected within a small area such as room, building
b) Data transfer rate is comparatively slower such as 150 mbps.	Data transfer speed is comparatively high such as thousand mbps.
c) WAN, links may be established by using telephone cable or microwave towers or satellite.	Co- axial cables are generally used to connect the computer and other devices.
d) In this network, short circuit errors, noise errors, atmospheric errors are higher than any other networks.	Due to short distance, short circuit errors or other noise errors are minimum.
e) For example : Pager.	For example : A computer lab in a college.

What is a Protocol ? Explain the concept of TCP/IP Protocol.

Ans:

(1) A protocol is defined as an agreement between communication particle for how communication should proceed.

Networking Technology

OR

Protocols are rules by which computers communicate i.e. protocol is set of rules and formats for sending and receiving data.

(2) Internet protocol are called TCP/IP (Transmission Control Protocol/Internet Protocol) protocols. This protocol do not belong any one company and technology is available to everybody

(3) TCP/IP protocol use three types of addresses for network addressing

- Hardware or physical address is used by the data link and physical layers.
- Internet protocol address provides logical node identification. This address is the unique address assigned by administratory expressed in four parts dotted notation.e.g. 123.144.131.21
- Logical node names are easier to remember than an IP address.

Write a note on Ethernet.

Ans: (1) Ethernet is a local area network technology, with networks traditionally operating within single building.

(2) Atmost, Ethernet devices can have a few hundred meters of cable between them. Modern technology allows Ethernet to span upto 10 kms.

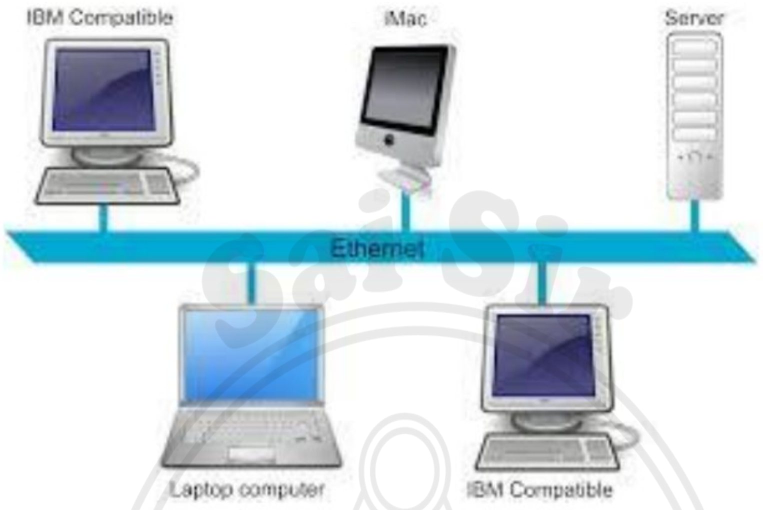
(3) Ethernet devices are connected to a common shared medium that provides the path along which the electronic signals will travel. Historically, this medium was co-axial cable. But, now-a-days twisted pair cable or fibre optic cable are also used.

(4) Ethernet network transmit data in small units called frames.

(5) Each frame must contain source address as well as destination address, which identifies recipient and sender of message. The address will uniquely identify node. No two Ethernet devices can have same address.

Networking Technology

(6) Ethernet network is as shown in following in figure.



In above figure when sender computer sends message to recipient computer, other computers will also get the message and check whether the destinations address matches to its own address or not, it not, it will discard the frame.

Write a short note on MODEM.

Ans: (1) Computer store digital data, while telephone lines can only transfer analog data. If a computer is to be connected to internet through telephone, then it must convert digital data to analog data before transmitting the computer signals.

(2) Converting one signal form to another form is called modulation and reconverting it to original form is called as demodulation.

(3) Modem is modulator/demodulator. Modem is used to connect computer to internet. Modems convert digital data to analog data and vice-versa.

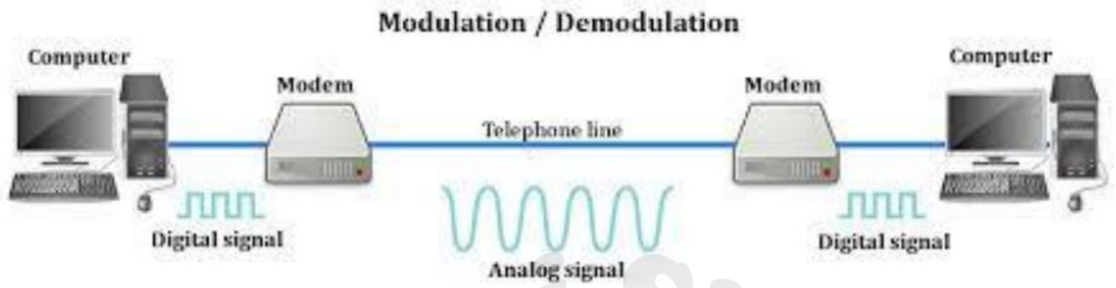
(4) They have two advantages:

(a) Modem allows higher speed of transmission on any given analog line.

(b) Modem reduce effect of noise and distortion.

(5) The function of modem is described by following figure.

Networking Technology



(6) Modems are classified into two categories according to transmission method :

(a) Asynchronous modems (b) Synchronous modems

What is HUB ? Explain all the types of HUB.

Ans: Hubs:

(1) In some network topologies, mostly star topologies, a device called hub is used.

(2) Hub is a connecting device in which cables can be connected without soldering wires to centralize network traffic through a single connecting point.

There are three types: (1) Active hub (2) Passive hub (3) Switching hub

Active hub : It interconnects the network and also amplifies the signal received apart from splitting and retransmitting it to the destination.

Passive hub : It only splits and transmits signal received and it cannot amplify it. It does not contain any electronic component.

Switching hub : This quickly routes the signals between ports of hubs. It can be used in place of router.