BALASORE SCHOOL OF ENGINEERING, BALASORE

STUDY MATERIAL

BRANCH – COMPUTER SCIENCE & ENGG.

SUBJECT :- OPERATING SYSTEM

SUBJECT NO-TH-1

SEMESTER :- 4TH

PREPARED BY: - SUVENDU NAYAK

CHAPTER – 1

2 MARKS

Q.1) Define Operating System? {2017-S(1 a),2018(w)-1-a}

Ans:- Operating system is a system software that installed in computer, that acts as an intermediary between a user of a computer and the computer hardware.

- ➢ The OS acts as an interface between the User, Application Programs, Hardware and the System Peripherals.
- ➢ It execute user programs and make solving user problems easier.
- ➢ It makes the computer system convenient to use.
- ➢ It use the computer hardware in an efficient manner.

Q.2) Name two Operating System. {2017-S (5 a),2015- S (1 a),2016-S (1 a),2018(s)-2-a,2019(s)-1-f}

Ans:- Following are two main operating system

- ➢ Single User:-If the single user Operating System is loaded in computer's memory; the computer would be able to handle one user at a time.
- ➢ Example:- MS-Dos, MS-Win 95-98, Win-ME
- ➢ Multi user:-If the multi-user Operating System is loaded in computer's memory; the computer would be able to handle more than one user at a time.
- ➢ Example:- UNIX, Linux, XENIX

Q.3) Difference between application and system software.{2018(s)-1-a,2019(s)-1-h}

- ➢ System software is used for operating computer hardware.
- ➢ Application software is used by user to perform specific task. ...
- ➢ System softwares are installed on the computer when operating system is installed.
- ➢ Application softwares are installed according to user's requirements.

5 MARKS

Q.1) Identify function of operating system and explain. {2016-S & 2017 –S (1 b), 2018(w)-1-b}

Ans:- Following are the main function of operating system

a) Process Management
b) Main Memory Management
c) File Management
d) I/O System Management
e) Secondary Storage Management

a) Process Management:-

A program does nothing unless their instructions are executed by a CPU.A process is a program in execution. A time shared user program such as a complier is a process. A word processing program being run by an individual user on a pc is a process.

- A system task such as sending output to a printer is also a process. A process needs certain
  resources including CPU time, memory files & I/O devices to accomplish its task.
- These resources are either given to the process when it is created or allocated to it while it is running. The OS is responsible for the following activities of process management.
- Creating & deleting both user & system processes.

b) Main Memory Management :-

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes ranging in size from hundreds of thousand to billions. Main memory stores the quickly accessible data shared by the CPU & I/O device. The central processor reads instruction from main memory during instruction fetch cycle & it both reads & writes data from main memory during the data fetch cycle.

- The main memory is generally the only large storage device that the CPU is able to address & access directly. For example, for the CPU to process data from disk.
- Keeping track of which parts of memory are currently being used & by whom.
- Allocating & deallocating memory space as needed.

c) File Management :-

File management is one of the most important components of an OS computer can store information on several different types of physical media magnetic tape, magnetic disk & optical disk are the most common media.

- Creating & deleting files & directories.
- Supporting primitives for manipulating files & directories.

d) I/O System Management:-

One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user. For example, in UNIX the peculiarities of I/O devices are hidden from the bulk of the OS itself by the I/O subsystem.

The I/O subsystem consists of a memory management component that includes buffering, catching & spooling

e) Secondary Storage Management :-

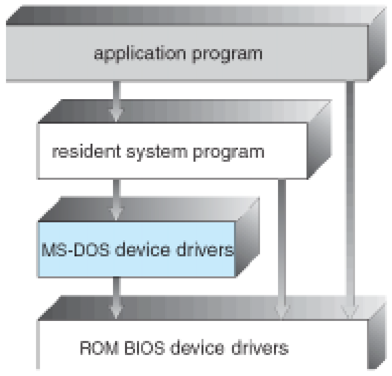
The main purpose of computer system is to execute programs. These programs with the data they access must be in main memory during execution. As the main memory is too small to accommodate all data & programs & because the data that it holds are lost when power is lost. The computer system must provide secondary storage to back-up main memory. Most modern computer systems are disks as the storage medium to store data & program.

Q.2) Explain structure of operating system with suitable diagram.

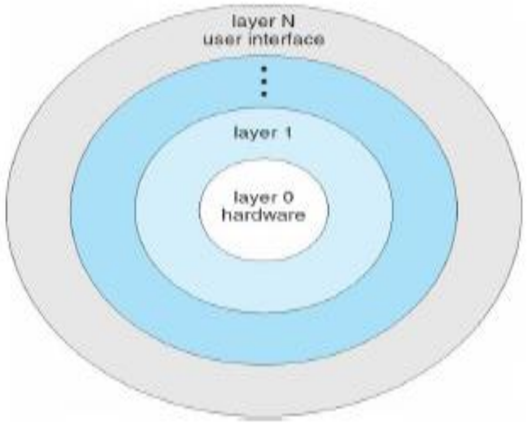
{2015-S (1 c), 2017-S (1 b),2018(w)-2-b}

Ans:- Three types of structure in operating system

a) Simple Structure
b) Layered Approach
c) Kernel Approach



Simple structure: There are several commercial system that don't have a well- defined structure such operating systems begins as small, simple & limited systems and then grow beyond their original scope. MS-DOS is an example of such system. It was not divided into modules carefully. Another example of limited structuring is the UNIX operating system.

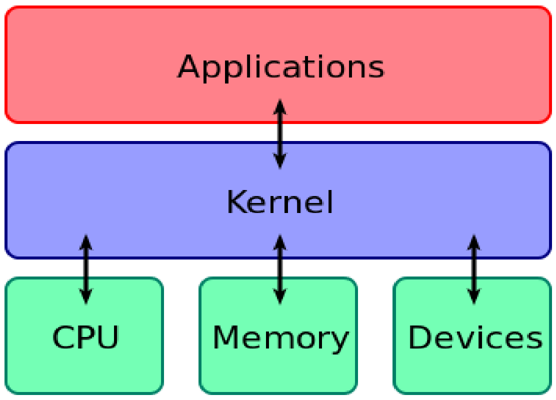


Layered approach: In the layered approach, the OS is broken into a number of layers (levels) each built on top of lower layers. The bottom layer (layer o) is the hardware & top most layer (layer N) is the user interface. The main advantage of the layered approach is modularity.

Layers	Functions
5	User Program
4	I/O Management
3	Operator Process Communication
2	Memory Management
1	CPU Scheduling
0	Hardware

- The layers are selected such that each users functions (or operations) & services of only lower layer.
- This approach simplifies debugging & system verification, i.e. the first layer can be debugged without concerning the rest of the system. Once the first layer is debugged.
- its correct functioning is assumed while the 2nd layer is debugged & so on.
- If an error is found during the debugging of a particular layer, the error must be on that layer because the layers below it are already

debugged. Thus the design & implementation of the system are simplified when the system is broken down into layers.



Kernel Approach:- The **kernel** is a computer program that is the core of a computer's operating system, with complete control over everything in the system. On most systems, it is one of the first programs loaded on start-up (after the boot loader).

➢ It handles the rest of start-up as well as input/output requests from software, translating them into data-processing instructions for the central processing unit. It handles memory and peripherals like keyboards, monitors, printers, and speakers.

7 MARKS

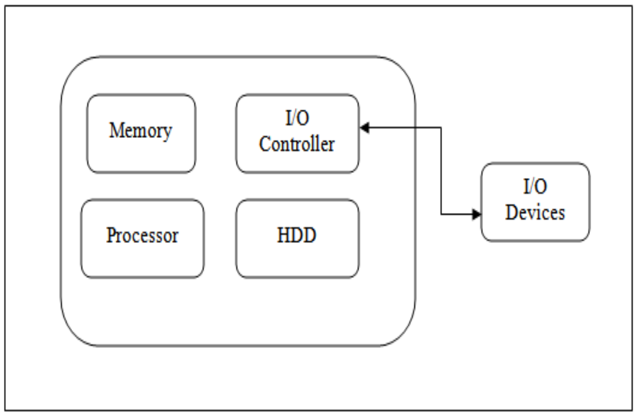
Q.1) State the objective of operating system. Explain the layered approach structure of operating system. **{2016-S(1 c)}**

Ans:- The operating system has two objectives such as

➢ Firstly, an operating system controls the computer's hardware.
➢ The second objective is to provide an interactive interface to the user and interpret commands so that it can communicate with the hardware.
➢ The operating system is very important part of almost every computer system.

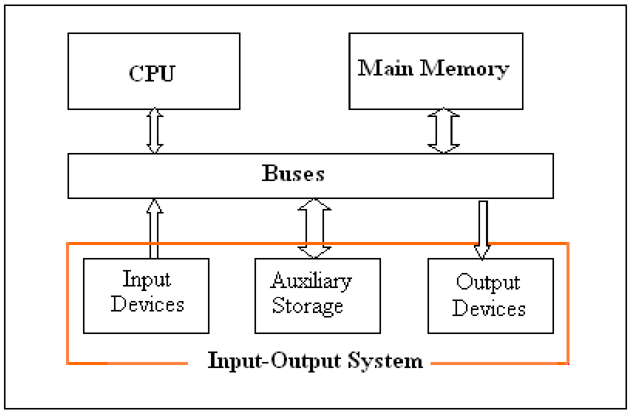
Managing Hardware

- These hardware resources include processer, memory, and disk space and so on.
- The output result was display in monitor. In addition to communicating with the hardware the operating system provides on error handling procedure and display an error notification.



Providing an Interface

- The operating system organizes application so that users can easily access, use and store them.
- It provides a stable and consistent way for applications to deal with the hardware without the user having known details of the hardware.



System goals

- The purpose of an operating system is to be provided an environment in which an user can execute programs.
- It's primary goals are to make the computer system convenience for the user.

Layered approach

In the layered approach, the OS is broken into a number of layers (levels) each built on top of lower layers. The bottom layer (layer o) is the hardware & top most layer (layer N) is the user interface. The main advantage of the layered approach is modularity.

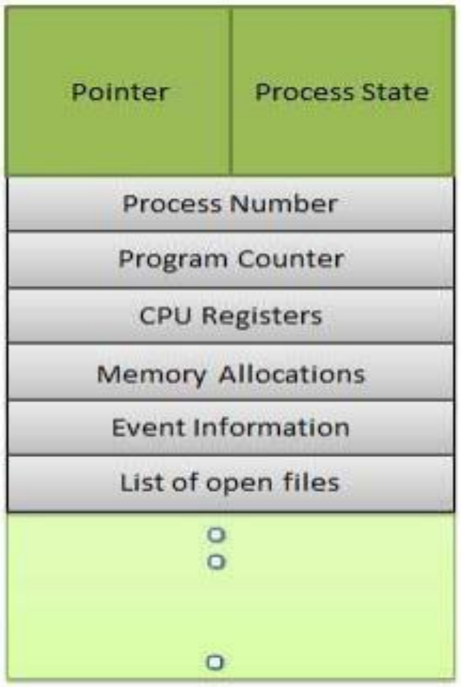
- The layers are selected such that each users functions (or operations) & services of only lower layer.
- This approach simplifies debugging & system verification, i.e. the first layer can be debugged without concerning the rest of the system. Once the first layer is debugged.
- Its correct functioning is assumed while the 2nd layer is debugged & so on.
- If an error is found during the debugging of a particular layer, the error must be on that layer because the layers below it are already debugged. Thus the design & implementation of the system are simplified when the system is broken down into layers.

Layers	Functions
5	User Program
4	I/O Management
3	Operator Process Communication
2	Memory Management
1	CPU Scheduling
0	Hardware

CHAPTER – 2
2 MARKS

Q.1) What is process control block ? {2016 –S (3 a) ,2015-S (4 a)}

Ans :- Process Control Block (PCB, also called Task Controlling Block, Entry of
the Process Table, Task Struct, or Switch frame) is a data structure in the operating system kernel containing the information needed to manage a particular process.



- When a process is created the O.S creates the corresponding process control block and when the process terminates the O.S release the process control block.
- PCBs defines the current state of the operating system. They are accessed and/or modified by most OS utilities, including those involved with scheduling, memory and I/O resource access and performance monitoring.

Q.2) What is process synchronization ? {2016-S(4 a), 2015 -S(8 b), 2018(s)-8-ii}

Ans:- Process Synchronization means sharing system resources by processes in a such a way that, Concurrent access to shared data is handled thereby minimizing the chance of inconsistent data.

- Maintaining data consistency demands mechanisms to ensure synchronized execution of cooperating processes
- It present both software and hardware solutions of the critical-section problem.
- Concurrent access to shared data may result in data inconsistency

Q.3) Define interacting process ? {2015-S (-7 a)}

Ans:- A process is a program in execution. For example, a running Lisp image is a single process of the operating system. The Multitasking Facility lets you divide the Lisp process into smaller independent processes that share the same address space. You can create new processes within the Lisp process with the function. The predicate identifies such processes. You can use the following functions to find out information about a process.

Q.4) What is job scheduling ? (2016-S-7 c)

Ans:- Job scheduling is the process of allocating system resources to many different tasks by an operating system (OS). The system handles prioritized job queues that are awaiting CPU time and it should determine which job to be taken from which queue and the amount of time to be allocated for the job.

- This type of scheduling makes sure that all jobs are carried out fairly and on time.
- Most OSs like Unix, Windows, etc., include standard job-scheduling abilities. A number of programs including database management systems (DBMS), backup, enterprise resource planning (ERP) and business process management (BPM) feature specific job-scheduling capabilities as well.

Q.5) State context switching ? {2017-S(3 a), 2019(s)-1-e }

Ans:- A context switch (also sometimes referred to as a process switch or a task switch) is the switching of the CPU (central processing unit) from one process or thread to another. A process (also sometimes referred to as a task) is an executing (i.e., running) instance of a program.

- Context switching can be described in slightly more detail as the *kernel* (i.e., the core of the operating system.
- It suspending the progression of one process and storing the CPU's *state* (i.e., the context) for that process somewhere in memory
- It retrieving the context of the next process from memory and restoring it in the CPU's registers.

Q.6) State principle of concurrency. {2017-S(4 a)}

Ans:- Concurrency is the interleaving of processes in time to give the appearance of simultaneous execution. Thus it differs from parallelism, which offers genuine simultaneous execution. However the issues and difficulties raised by the two overlap to a large extent.

- Concurrency can be implemented and is used a lot on single processing units, nonetheless it may benefit from multiple processing units with respect to speed.
- If an operating system is called a multi-tasking operating system, this is a synonym for supporting concurrency.
- If you can load multiple documents simultaneously in the tabs of your browser and you can still open menus and perform more actions, this is concurrency.

Q.7) Define semaphore ?

{2015-S(3 a) , 2016 –S(5 a), 2017-S(2 a), 2018(w)-3-a ,2019(s)-1-b }

Ans:- Semaphore is a simply a variable. This variable is used to solve critical section problem and to achieve process synchronization in the multi processing environment. The two most common kinds of semaphores are counting semaphores and binary semaphores. Counting semaphore can take non-negative integer values and Binary semaphore can take the value 0 & 1 only.

The variable is then used as a condition to control access to some system resource.

- For the solution to the critical section problem one synchronization tool is used which is known as semaphore
- A semaphore 'S' is an integer variable which is accessed through two standard operations such as wait and signal.
- It can only be accessed via two indivisible (atomic) operations.

5 MARKS

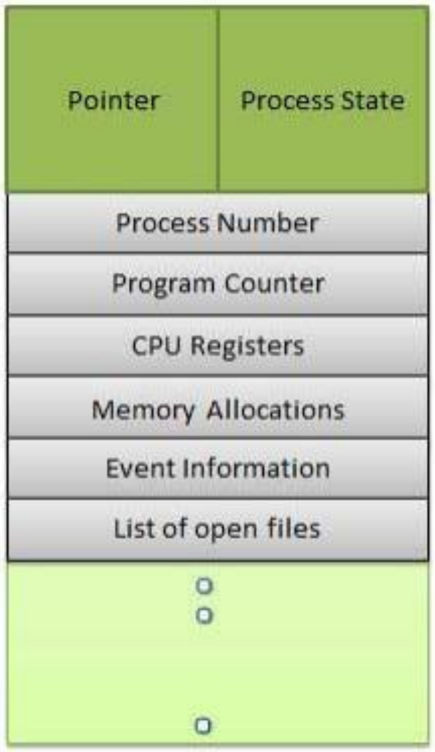
Q.1) Define PCB. Explain the different fields stored in PCB. {2017-S (2 b), **2018(s)-6-b**}

Ans:- Process Control Block (PCB, also called Task Controlling Block, Entry of the Process Table, Task Struct, or Switch frame) is a data structure in the operating system kernel containing the information needed to manage a particular process.

- When a process is created the O.S creates the corresponding process control block and when the process terminates the O.S release the process control block.

Each process is represented in the operating system by a process control block (PCB) also called a task control block. PCB is the data structure used by the operating system. Operating system groups all information that needs about particular process. PCB contains many pieces of information associated with a specific process which are described below.

1) Pointer Pointe:- points to another process control block. Pointer is used for maintaining the scheduling list.



2) Process State:- Process state may be new, ready, running, waiting and so on.

3) Program Counter:- Program Counter indicates the address of the next instruction to be executed for this process.

4) CPU registers :-CPU registers include general purpose register, stack pointers, index registers and accumulators etc. number of register and type of register totally depends upon the computer architecture.

5) Memory management information:- This information may include the value of base and limit registers, the page tables, or the segment tables depending on the memory system used by the operating system. This information is useful for deallocating the memory when the process terminates.

6) Accounting information:- This information includes the amount of CPU and real time used, time limits, job or process numbers, account numbers etc.

Q.2) What are interacting process? What is process Synchronization ? {2017-S (3 b,4 b)}

Ans:- Interacting process:

It contains the program code and its current activity. Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently. A computer program is a passive collection of instructions, while a process is the actual execution of those instructions.

- A computer program is a passive collection of instructions, while a process is the actual execution of those instructions. Several processes may be associated with the same program; for example, opening up several instances of the same program often means more than one process is being executed.

Process Synchronization:

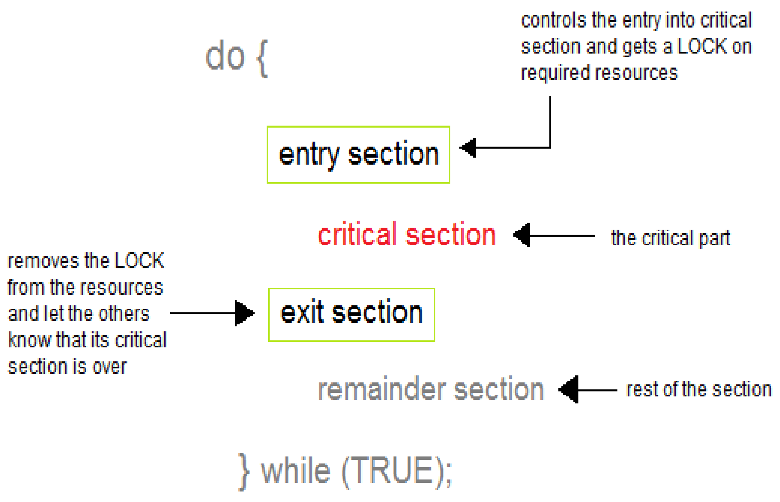
Process Synchronization means sharing system resources by processes in a such a way that, Concurrent access to shared data is handled thereby minimizing the chance of inconsistent data. Maintaining data consistency demands mechanisms to ensure synchronized execution of cooperating processes.

- Process Synchronization was introduced to handle problems that arose while multiple process executions.

Some of the problems are discussed below.

Critical Section Problem

A Critical Section is a code segment that accesses shared variables and has to be executed as an atomic action. It means that in a group of cooperating processes, at a given point of time, only one process must be executing its critical section. If any other process also wants to execute its critical section, it must wait until the first one finishes.


do {
controls the entry into critical section and gets a LOCK on required resources
entry section
critical section
the critical part
removes the LOCK from the resources and let the others know that its critical section is over
exit section
remainder section
rest of the section
} while (TRUE);

Example:

```
    While (1)
      {
    Entry Section;
    Critical Section;
    Exit Section;
    Remainder Section;
       }
```

Solution to Critical Section Problem

1. **Mutual Exclusion**

   Out of a group of cooperating processes, only one process can be in its critical section at a given point of time.

2. **Progress**

   If no process is in its critical section, and if one or more threads want to execute their critical section then any one of these threads must be allowed to get into its critical section.

3. **Bounded Waiting**

   After a process makes a request for getting into its critical section, there is a limit for how many other processes can get into their critical section, before this process's request is granted. So after the limit is reached, system must grant the process permission to get into its critical section.

3) What is process ? Draw a diagram to explain different state of a process with the state transition. {2017-S (1 c),2016-S (3 b),2015-S (3 b), 2018(w)-1-c, 2019(s)-2-b}

Ans:- Process:

A process is an instance of a computer program that is being executed. It contains the program code and its current activity. Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently

- A process is a program in execution. Process is not as same as program code but a lot more than it. A process is an 'active' entity as opposed to program which is considered to be a 'passive' entity. Attributes held by process include hardware state, memory, CPU etc.

Processes can be any of the following states :

When process executes, it changes state. Process state is defined as the current activity of the process. Fig. shows the general form of the process state transition diagram. Process state contains five states. Each process is in one of the states. The states are listed below

1. New
2. Ready
3. Running
4. Waiting
5. Terminated(exist)

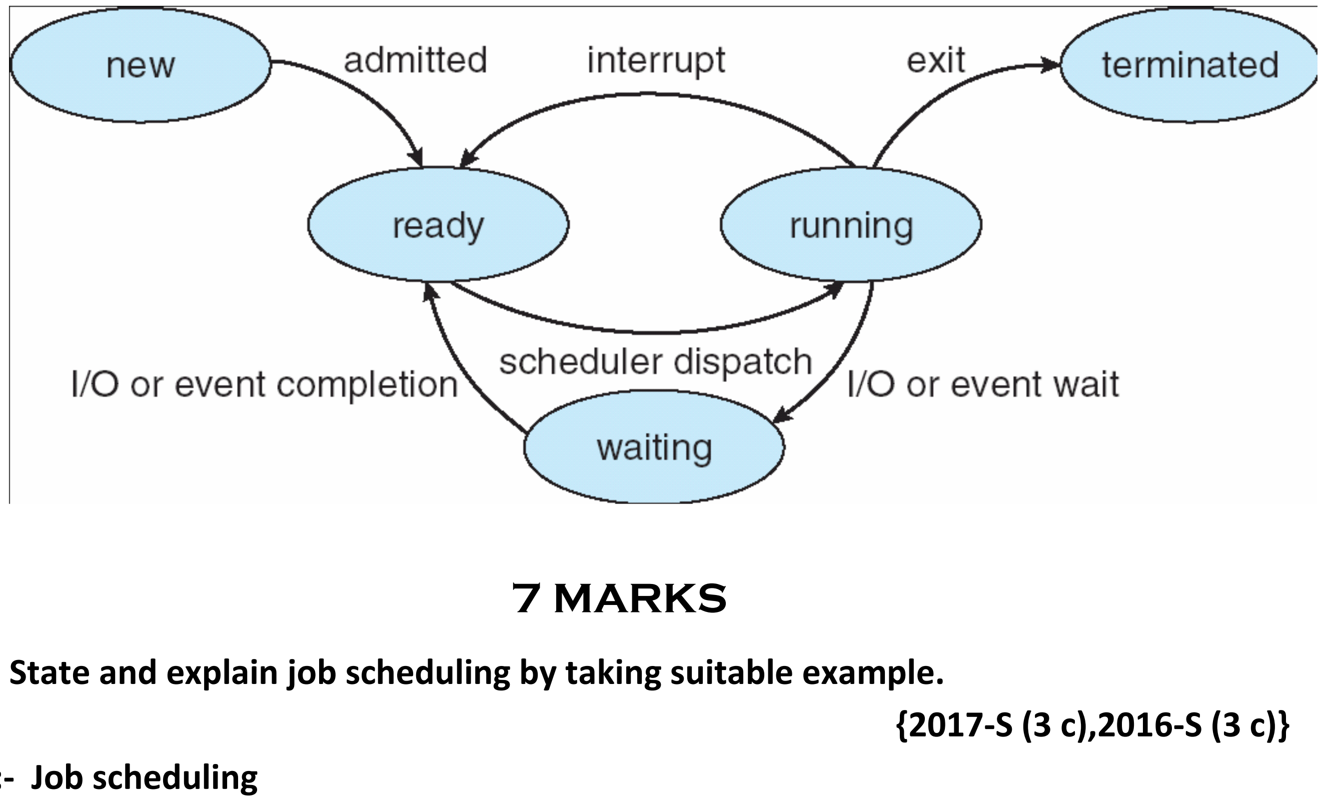
1. New : A process that just been created.

2. Ready : Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.

3. Running : The process that is currently being executed. A running process possesses all the resources needed for its execution, including the processor.

4. Waiting : A process that can not execute until some event occurs such as the completion of an I/O operation. The running process may become suspended by invoking an I/O module.

5. Terminated : A process that has been released from the pool of executable processes by the operating system.



7 MARKS

Q.1) State and explain job scheduling by taking suitable example.

{2017-S (3 c),2016-S (3 c)}

Ans:- Job scheduling

Job scheduling is the process of allocating system resources to many different tasks by an operating system (OS). The system handles prioritized job queues that are awaiting

CPU time and it should determine which job to be taken from which queue and the amount of time to be allocated for the job.

- This type of scheduling makes sure that all jobs are carried out fairly and on time.
- Most OSs like Unix, Windows, etc., include standard job-scheduling abilities. A number of programs including database management systems (DBMS), backup, enterprise resource planning (ERP) and business process management (BPM) feature specific job-scheduling capabilities as well.
- Real-time scheduling in accordance with external, unforeseen events
- Automated restart and recovery in case of failures
- Notifying the operations personnel
- Generating reports of incidents
- Audit trails meant for regulation compliance purposes
- Continuously automatic monitoring of jobs and completion notification
- Event-driven job scheduling
- Performance monitoring

Job scheduling schedules or decides which process will get to acquire the resource based on some parameters.

FIFO- The process that comes first is executed first.

Shortest job First- The time taken to complete the job of all processes is computed and the shortest length process is executed first.

Round Robin- Each process gets a time "share" for running and is later prevented to get the next process running. This process is known as time sharing, which provides the effect of all the processes running at the same time.

Priority scheduling- Here, the process on highest priority gets the resource.

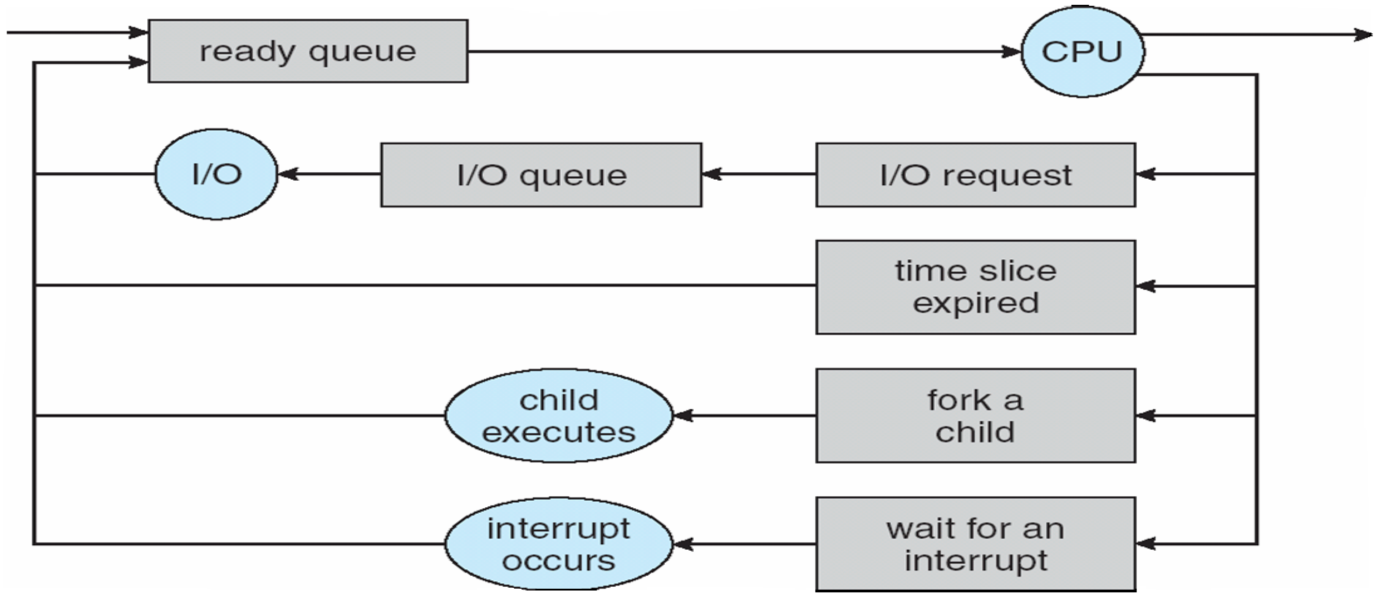
Q.2) Define process Scheduling. Explain type of scheduler . {2017-S (2 c)}

Ans:- Process scheduling:

The act of determining which process in the ready state should be moved to the running state is known as Process Scheduling.

The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs. For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of CPU

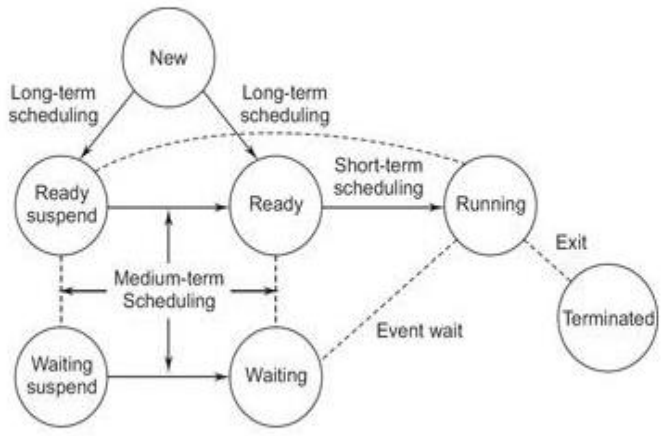
- Scheduling is a fundamental function of OS. When a computer is multi programmed, it has multiple processes completing for the CPU at the same time. If only one CPU is available, then a choice has to be made regarding which process to execute next. This decision making process is known as scheduling and the part of the OS that makes this choice is call in making this choice is called scheduling algorithm.



There are three types of schedulers available :

1. Long Term Scheduler :

Long term scheduler runs less frequently. Long Term Schedulers decide which program must get into the job queue. From the job queue, the Job Processor, selects processes and loads them into the memory for execution. Primary aim of the Job Scheduler is to maintain a good degree of Multiprogramming. An optimal degree of Multiprogramming means the average rate of process creation is equal to the average departure rate of processes from the execution memory.



- ➢ It is important that the long term scheduler should select a good mix of I/O bound & CPU bound processes
- ➢ It executes less frequently than other schedulers. If the degree of multiprogramming is stable than the average rate of process creation is equal to the average departure rate of processes leaving the system.

2. Short Term Scheduler :

This is also known as CPU Scheduler and runs very frequently. The primary aim of this scheduler is to enhance CPU performance and increase process execution rate.

- The short-term scheduler must select a new process for the CPU quite frequently. It must execute at least one in 100ms.

3. Medium Term Scheduler :

This scheduler removes the processes from memory (and from active contention for the CPU), and thus reduces the degree of multiprogramming. At some later time, the process can be reintroduced into memory and its execution van be continued where it left off. This scheme is called **swapping**. The process is swapped out, and is later swapped in, by the medium term scheduler.

Q.3) Define process Scheduling. Name different scheduling algorithms and explain.
{2016-S (3 c), 2018(s)-5-c}

Ans:- Process scheduling:

The act of determining which process in the ready state should be moved to the running state is known as Process Scheduling.

The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs. For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of CPU.

List of scheduling algorithms are as follows:

- First-come, first-served scheduling (FCFS) algorithm
- Shortest Job First Scheduling (SJF) algorithm
- Shortest Remaining time (SRT) algorithm
- Round-Robin Scheduling algorithm
- Multilevel Feedback Queue Scheduling algorithm
- Multilevel Queue Scheduling algorithm

a) First Come, First Served Scheduling (FCFS) Algorithm:

This is the simplest CPU scheduling algorithm. In this scheme, the process which requests the CPU first, that is allocated to the CPU first. The implementation of the FCFS algorithm is easily managed with a FIFO queue. When a process enters the ready queue its PCB is linked onto the rear of the queue. The average waiting time under FCFS policy is quiet long.

b) Shortest Job First Scheduling (SJF) Algorithm:

This algorithm associates with each process if the CPU is available. This scheduling is also known as shortest next CPU burst, because the scheduling is done by examining the length of the next CPU burst of the process rather than its total length

c) Priority Scheduling Algorithm:

In this scheduling a priority is associated with each process and the CPU is allocated to the process with the highest priority. Equal priority processes are scheduled in FCFS manner.

d) Round Robin Scheduling Algorithm:

This type of algorithm is designed only for the time sharing system. It is similar to FCFS scheduling with preemption condition to switch between processes. A small unit of time called quantum time or time slice is used to switch between the processes. The average waiting time under the round robin policy is quiet long.

e) Multilevel Queue Scheduling:

In this The Time of CPU is divided by using Some Process Categories. In this the Process those are executed on the Foreground or on the Screen, have a higher Priority and the Process those are running in the Background to fill the Request the user. When we Input the data into the Computer. Then the Data is displayed on the Screen after Processing.

CHAPTER –3

2 MARKS

Q.1) Define segmentation with one example ? {2015- S (2a), 2018(s)-4-c}

Ans:- Memory segmentation is the division of a computer's primary memory into segments or sections. In a computer system using segmentation, a reference to a memory location includes a value that identifies a segment and an offset (memory location) within that segment

Example:-

Q.2) Define multiple partitioning ? {2016-S(7 d)}

Ans:- Disk partitioning or **disk slicing** is the creation of one or more regions on a hard disk or other secondary storage, so that an operating system can manage information in each region separately. Partitioning is typically the first step of preparing a newly manufactured disk, before any files or directories have been created. The disk stores the information about the partitions' locations and sizes in an area known as the partition table that the operating system reads before any other part of the disk.

- Separation of the operating system (OS) and program files from user files. This allows image backups (or clones) to be made of only the operating system and installed software.
- Use of multi-boot setups, which allow users to have more than one operating system on a single computer. For example, one could Install BSD, Linux, mac OS, Microsoft Windows or other operating systems on different partitions of the same HDD and have a choice of booting into any compatible operating system at power-up.

- Protecting or isolating files, to make it easier to recover a corrupted file system or operating system installation. If one partition is corrupted, other file systems may not be affected.

5 MARKS

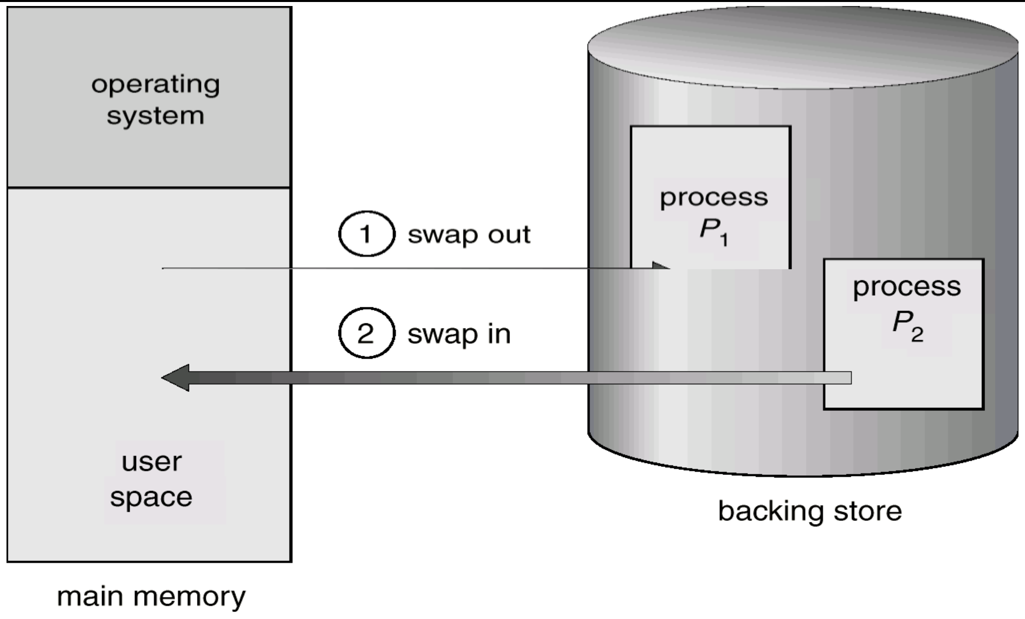
Q.1) What is swapping ? Explain the principle of swapping with diagram.
{2016 ,2017-S (5 b), 2018(s)-7-b, 2018(w)-3-b, 2019(s)-2-c}

Ans:- **Swapping**

- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution. For example, assume a multiprogramming environment with a round robin CPU scheduling algorithm. When a quantum expires, the memory manager will start to swap out the process that just finished, and to swap in another process to the memory space that has been freed. In the mean time, the CPU scheduler will allocate a time slice to some other process in memory. When each process finished its quantum, it will be swapped with another process. Ideally, the memory manager can swap processes fast enough that some processes will be in memory, ready to execute, when the CPU scheduler wants to reschedule the CPU. The quantum must also be sufficiently large that reasonable amounts of computing are done between swaps.
- To replace pages or segments of data in memory. Swapping is a useful technique that enables a computer to execute programs and manipulate data files larger than main memory. The operating system copies as much data as possible into main memory, and leaves the rest on the disk. When the operating system needs data from the disk, it exchanges a portion of data (called a *page* or *segment*) in main memory with a portion of data on the disk.

Schematic View of Swapping

- Main memory usually into two partitions:
- Resident operating system, usually held in low memory with interrupt vector
- User processes then held in high memory Relocation registers used to protect user processes from each other, and from changing operating-system code and data
- Base register contains value of smallest physical address
- Limit register contains range of logical addresses – each logical address must be less than the limit register
- MMU maps logical address dynamically



Q.2) Define swapping ? Explain the role of swapping in virtual memory management. {2015-S (2 b)}

Ans:- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution. For example, assume a multiprogramming environment with a round robin CPU scheduling algorithm. When a quantum expires, the memory manager will start to swap out the process that just finished, and to swap in another process to the memory space that has been freed. In the mean time, the CPU scheduler will allocate a time slice to some other process in memory. When each process finished its quantum, it will be swapped with another process. Ideally, the memory manager can swap processes fast enough that some processes will be in memory, ready to execute, when the CPU scheduler wants to reschedule the CPU. The quantum must also be sufficiently large that reasonable amounts of computing are done between swaps.

Role of swapping in virtual memory management

- Swap memory is a temporary memory and only used when RAM fails or full.
- Swap memory is simply a temporary storage space for the page.
- Virtual memory and Swap memory can be interchangeable with each other, but they work totally different from each other. But both the virtual memory and swap memory are equally important for the working of computer system properly.

Q. 3) What is page fault ? Explain the page fault handling techniques.

Ans:-

An interrupt that occurs when a program requests data that is not currently in real memory. The interrupt triggers the operating system to fetch the data from a virtual memory

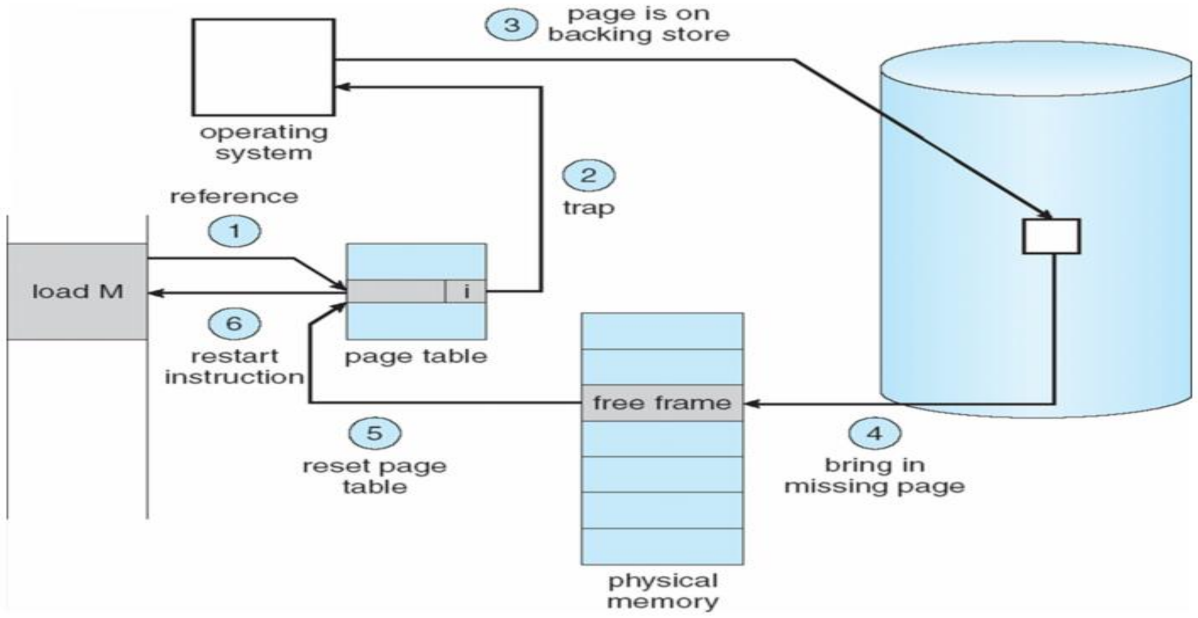
and load it into RAM. An invalid page fault or page fault error occurs when the operating system cannot find the data in virtual memory.
when the page was evicted, the OS set the PTE as invalid and noted the disk location of the page in a data structure (that looks like a page table but holds disk addresses)

- When a process tries to access the page, the invalid PTE will cause an exception (page fault) to be thrown
  - OK, it's actually an interrupt.
- The OS will run the page fault handler in response
  - Handler uses the "like a page table" data structure to locate the page on disk
  - Handler reads page into a physical frame, updates PTE to point to it and to be valid
  - OS restarts the faulting process

Procedure to handle page fault

If a process refers to a page that is not in physical memory then

- We check an internal table (page table) for this process to determine whether the reference was valid or invalid.
- If the reference was invalid, we terminate the process, if it was valid but not yet brought in, we have to bring that from main memory.
- Now we find a free frame in memory.
- Then we read the desired page into the newly allocated frame.
- When the disk read is complete, we modify the internal table to indicate that the page is now in memory.
- We restart the instruction that was interrupted by the illegal address trap. Now the process can access the page as if it had always been in memory.



7 MARKS

Q.1) **Define page. Explain memory management technique using paging .**

{2015-S (2 c)}

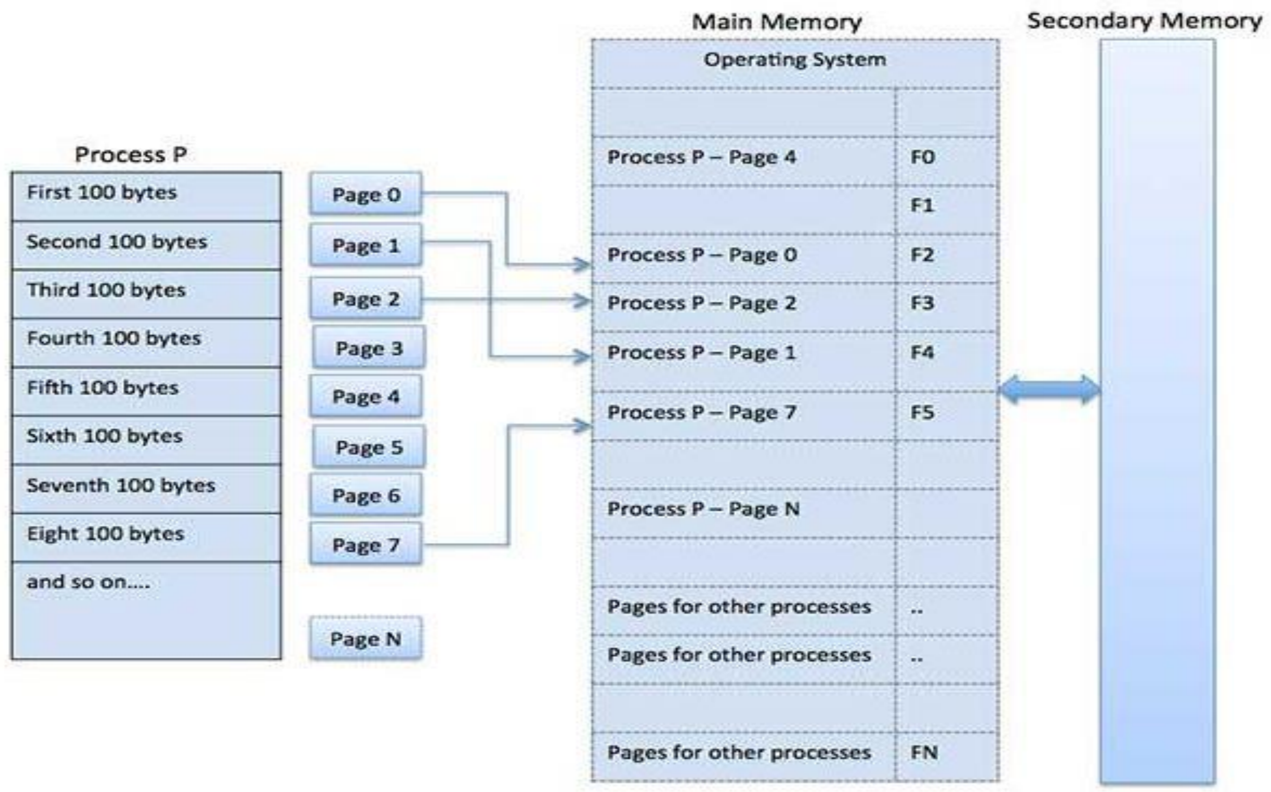
Ans:- Page

In computer operating systems, paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. In this scheme, the operating system retrieves data from secondary storage in same-size blocks called **pages**.

Paging

In computer operating systems, **paging** is a memory management scheme by which a computer stores and retrieves data from secondary storage[a] for use in main memory.

- Paging is a memory management scheme that permits the physical address space of a process to be non contiguous.
- Divide physical memory into fixed-sized blocks called frames (size is power of 2, for example 512 bytes).
- Divide logical memory into blocks of same size called pages Keep track of all free frames.



Memory management technique

Demand paging

When pure demand paging is used, pages are loaded only when they are referenced. A program begins execution with none of its pages in RAM. As the program commits page faults, the operating system copies the needed pages from the executable file into RAM. Pages of the executable file that are not executed during a particular run are never loaded into memory.

Anticipatory paging

This technique, sometimes also called swap pre fetch, predicts which pages will be referenced soon, to minimize future page faults. For example, after reading a page to service a page fault, the operating system may also read the next few pages even though they are not yet needed (a prediction using locality of reference). If a program ends, the operating system may delay freeing its pages, in case the user runs the same program again.

Free page queue, stealing, and reclamation

The free page queue is a list of page frames that are available for assignment. Preventing this queue from being empty minimizes the computing necessary to service a page fault. Some operating systems periodically look for pages that have not been recently referenced and perform *page stealing,* freeing the page frame and adding it to the free page queue. Some operating systems[b] support *page reclamation*; if a program commits a page fault by referencing a page that was stolen, the operating system detects this and restores the page frame without having to read the contents back into RAM.

Pre-cleaning

The operating system may periodically pre-clean dirty pages: write modified pages back to disk even though they might be further modified. This minimizes the amount of cleaning needed to obtain new page frames at the moment a new program starts or a new data file is opened, and improves responsiveness. (Unix operating systems periodically use sync to pre-clean all dirty pages; Windows operating systems use "modified page writer" threads.)

Q.2) Define page. Explain demand paging method of memory management. {2017-S(4 c), 2018(s)-5-b}

Ans:- Page

In computer operating systems, paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. In this scheme, the operating system retrieves data from secondary storage in same-size blocks called **pages**.

Paging

In computer operating systems, **paging** is a memory management scheme by which a computer stores and retrieves data from secondary storage[a] for use in main memory.

- Paging is a memory management scheme that permits the physical address space of a process to be non contiguous.
- Divide physical memory into fixed-sized blocks called frames (size is power of 2, for example 512 bytes).
- Divide logical memory into blocks of same size called pages Keep track of all free frames.

Demand Paging

In computer operating systems, demand paging(as opposed to anticipatory paging) is a method of virtual memory management. In a system that uses demand paging, the operating system copies a disk page into physical memory only if an attempt is made to access it and that page is not already in memory (i.e., if a page fault occurs). It follows that a process begins execution with none of its pages in physical memory, and many page faults will occur until most of a process's working set of pages is located in physical memory.

Demand paging follows that pages should only be brought into memory if the executing process demands them. This is often referred to as lazy evaluation as only those pages demanded by the process are swapped from secondary storage to main memory. Contrast this to pure swapping, where all memory for a process is swapped from secondary storage to main memory during the process startup.

Commonly, to achieve this process a page table implementation is used. The page table maps logical memory to physical memory. The page table uses a bitwise operator to mark if a page is valid or invalid. A valid page is one that currently resides in main memory. An invalid page is one that currently resides in secondary memory. When a process tries to access a page, the following steps are generally followed:

- Attempt to access page.
- If page is valid (in memory) then continue processing instruction as normal.
- If page is invalid then a page-fault trap occurs.

- Check if the memory reference is a valid reference to a location on secondary memory. If not, the process is terminated (illegal memory access). Otherwise, we have to page in the required page.
- Schedule disk operation to read the desired page into main memory.
- Restart the instruction that was interrupted by the operating system trap.

When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again. Instead of swapping in a whole process, the pager brings only those necessary pages into memory. The transfer of a paged memory to contiguous disk space is shown in below figure.

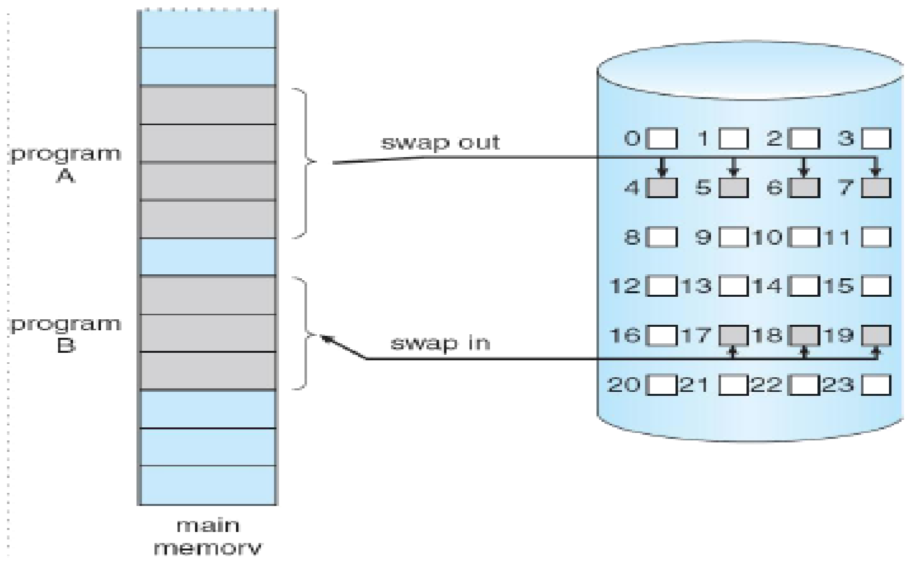


Fig. Transfer of a paged memory to continuous disk space

Q.3) **What is demand paging ? Explain the procedure to handle page fault with suitable block diagram.** **{2016-S (5 c)}**

Ans:- Demand paging

In computer operating systems, **demand paging** (as opposed to anticipatory paging) is a method of virtual memory management. In a system that uses demand paging, the operating system copies a disk page into physical memory only if an attempt is made to access it and that page is not already in memory (*i.e.*, if a page fault occurs). It follows that a process begins execution with none of its pages in physical memory, and many page faults will occur until most of a process's working set of pages is located in physical memory.

Page faults

Access to a page marked invalid causes a page-fault trap. This trap is the result of the operating system's failure to bring the desired page into memory. But page fault can be handled as following.

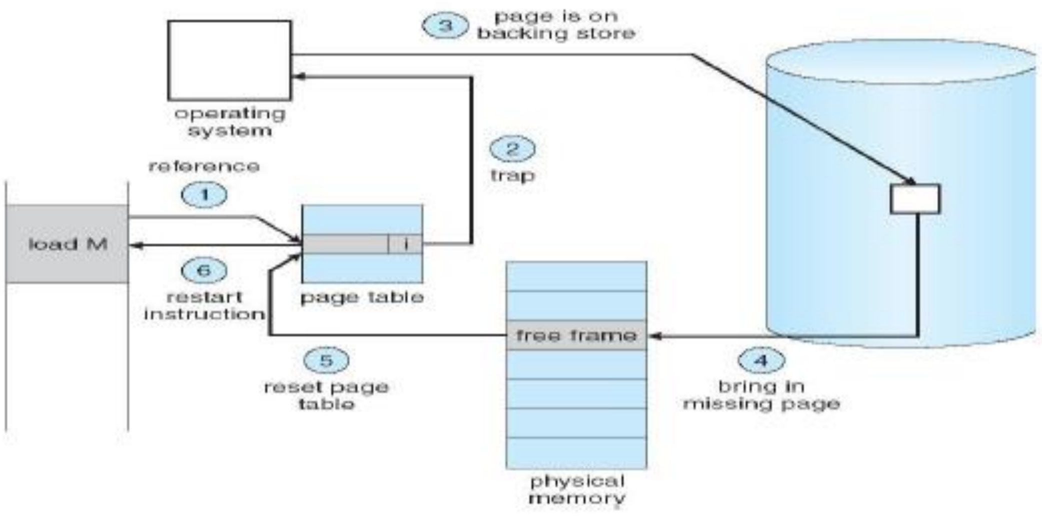


Fig. Steps in handling a page fault

If a process refers to a page that is not in physical memory then

- We check an internal table (page table) for this process to determine whether the reference was valid or invalid.
- If the reference was invalid, we terminate the process, if it was valid but not yet brought in, we have to bring that from main memory.
- Now we find a free frame in memory.
- Then we read the desired page into the newly allocated frame.
- When the disk read is complete, we modify the internal table to indicate that the page is now in memory.
- We restart the instruction that was interrupted by the illegal address trap. Now the process can access the page as if it had always been in memory.

Q.4) Define virtual memory. Explain virtual memory using segmentation method of memory management. {2017-S (5 c)}

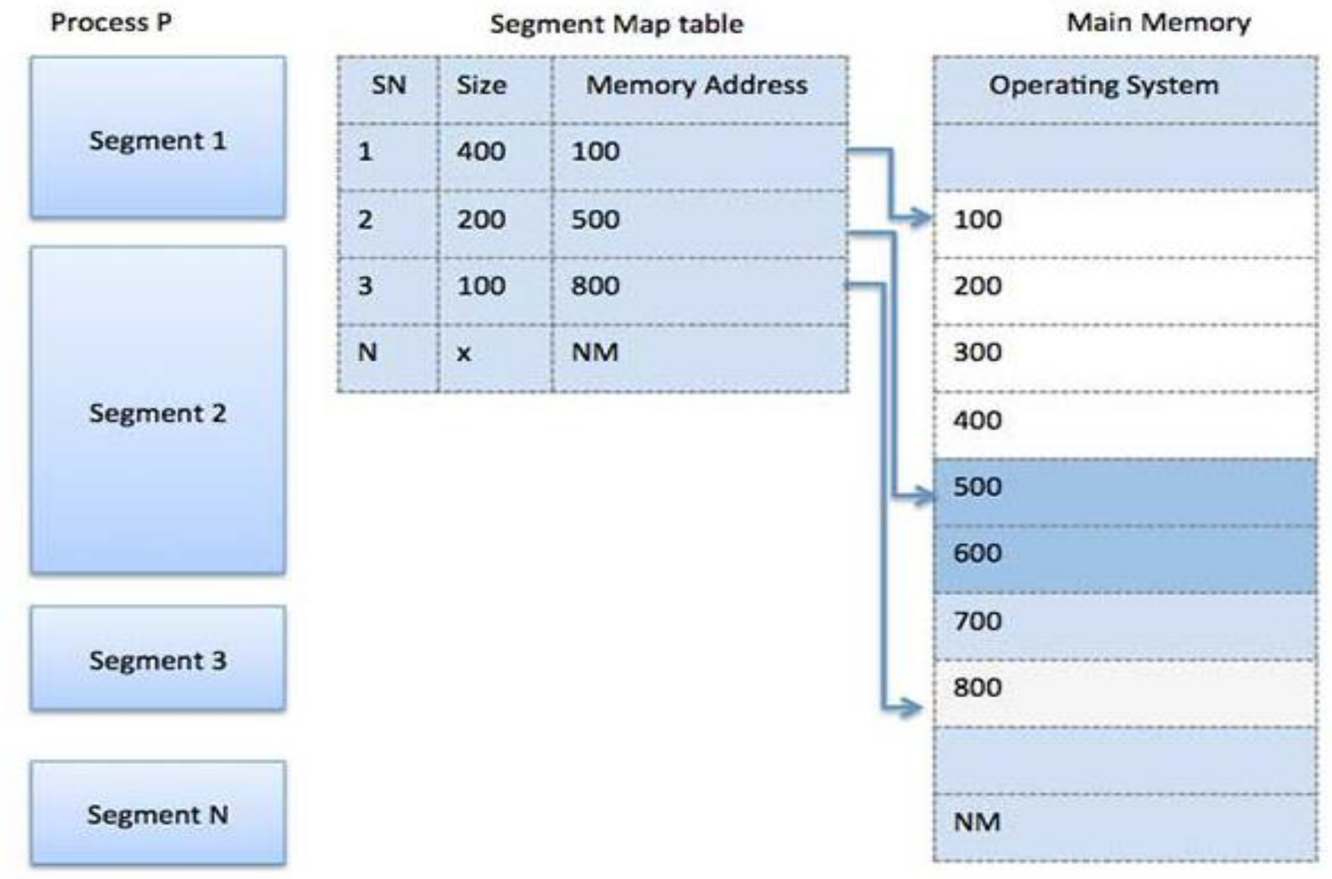
Ans :- Virtual Memory

- It is a technique which allows execution of process that may not be compiled within the primary memory.

- It separates the user logical memory from the physical memory. This separation allows an extremely large memory to be provided for program when only a small physical memory is available.
- Virtual memory makes the task of programming much easier because the programmer no longer needs to working about the amount of the physical memory is available or not.
- The virtual memory allows files and memory to be shared by different processes by page sharing.
- It is most commonly implemented by demand paging.

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. Each segment is actually a different logical address space of the program.
- When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.
- Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.
- Segmentation is a memory management scheme that supports this user view of memory.
- A logical address space is a collection of segments. Each segment has a name and a length.
- The addresses specify both the segment name and the offset within the segment.
- The user therefore specifies each address by two quantities such as segment name and an offset. For simplicity of implementation, segments are numbered and are referred to by a segment number, rather than by a segment name.



SN	Size	Memory Address
1	400	100
2	200	500
3	100	800
N	x	NM

CHAPTER – 4

2 MARKS

Q.1) Define SPOOLING ? {2015-(5 a), 2016 & 2017 (6 a),2018(s)-3-a,2019(s)-1-g}

Ans:- Spooling is a process in which data is temporarily held to be used and executed by a device, program or the system. Data is sent to and stored in memory or other volatile storage until the program or computer requests it for execution.

- "Spool" is technically an acronym for simultaneous peripheral operations online.
- Generally, the spool is maintained on the computer's physical memory, buffers or the I/O device-specific interrupts.
- The spool is processed in ascending order, working on the basis of a FIFO (first in, first out) algorithm.

5 MARKS

Q.1) Define device management ? Explain the techniques of device management.
{2015 (5 b), 2016(4 b), 2017(6),2019(s)-2-d}

Ans:- Device management is the process of managing the implementation, operation and maintenance of a physical and/or virtual device. It is a broad term that includes various administrative tools and processes for the maintenance and upkeep of a computing, network, mobile and/or virtual device.

- **Request device, release device:** If there are multiple users of the system, we first request

the device. After we finished with the device, we must release it.

- **Read, write, reposition:** Once the device has been requested & allocated to us, we can read,

write & reposition the device.

There are three basic techniques for implementing a device for policy.

1.**Dedicated** : A technique where a device is assign to a single process.
2.**Shared** : A technique where a device shared by many processes.
3.**Virtual** : A technique where one physical device this simulated on another physical device. Its combination of dedicated devices that have been transformed into shared devices.

CHAPTER – 5

2 MARKS

Q.1) What is Banker's Algorithm ? {2016-S (7 a)}

Ans:- The Banker's algorithm, sometimes referred to as the detection algorithm, is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and then makes an "s-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

- Several data structure must be maintained to implement the banker's algorithm.

5 MARKS

Q. 1) State and explain Banker's Algorithm ? {2015-S (4 b), 2018(s)-4-b ,2019(s)-2-e}

Ans:- Banker's Algorithm

The Banker's algorithm, sometimes referred to as the detection algorithm, is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and then makes an "s-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

- Several data structure must be maintained to implement the banker's algorithm.

Following **Data structures** are used to implement the Banker's Algorithm:

Let **'n'** be the number of processes in the system and **'m'** be the number of resources types.

Available :

- It is a 1-d array of size **'m'** indicating the number of available resources of each type.
- Available[j] = k means there are **'k'** instances of resource type $\mathbf{R_j}$

Max :

- It is a 2-d array of size '**n*m'** that defines the maximum demand of each process in a system.
- Max[i, j] = k means process $\mathbf{P_i}$ may request at most **'k'** instances of resource type $\mathbf{R_j}$.

Allocation :

- It is a 2-d array of size **'n*m'** that defines the number of resources of each type currently allocated to each process.
- Allocation[i, j] = k means process $\mathbf{P_i}$ is currently allocated **'k'** instances of resource type $\mathbf{R_j}$

Need :

- It is a 2-d array of size **'n*m'** that indicates the remaining resource need of each process.
- Need [i, j] = k means process $\mathbf{P_i}$ currently allocated **'k'** instances of resource type $\mathbf{R_j}$
- Need [i, j] = Max [i, j] – Allocation [i, j]

7 MARKS

Q. 1) How deadlock occurs?. Explain how dead lock is recovered and prevented.
{2016-S (6 c),2015 -S (4 c),2017-S (6 c), 2019(s)-4 }

Ans:- Deadlock

In a multiprogramming environment several processes may compete for a finite number of resources. A process request resources; if the resource is available at that time a process enters the wait state. Waiting process may never change its state because the resources requested are held by other waiting process. This situation is known as deadlock**.**

Recovery from Deadlock

When a detection algorithm determines that a deadlock exists, several alternatives exist. One possibility is to inform the operator that a deadlock has occurred, and to let the operator deal with the deadlock manually. The other possibility is to let the system recover from the deadlock automatically. There are two options for breaking a deadlock. One solution is simply to abort one or more processes to break the circular wait. The second option is to preempt some resources from one or more of the deadlocked processes.

- **Process Termination**:

To eliminate deadlocks by aborting a process, we use one of two methods. In both methods, the system reclaims all resources allocated to the terminated processes.

- **Abort all deadlocked processes**:
  This method clearly will break the deadlock cycle, but at a great expense; these processes may have computed for a long time, and the results of these partial computations must be discarded and probably recomputed later.
- **Resource Preemption**:
  Resources are preempted from the processes involved in deadlock, preempted resources are allocated to other processes, so that their is a possibility of recovering the system from deadlock. In this case system go into starvation.

Deadlock Prevention

Deadlocks can be prevented by preventing at least one of the four required conditions:

1) Mutual Exclusion

- Shared resources such as read-only files do not lead to deadlocks.
- Unfortunately some resources, such as printers and tape drives, require exclusive access by a single process.

2) Hold and Wait

- To prevent this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others. There are several possibilities for this:
  - Require that all processes request all resources at one time. This can be wasteful of system resources if a process needs one resource early in its execution and doesn't need some other resource until much later.

3) No Preemption

- Preemption of process resource allocations can prevent this condition of deadlocks, when it is possible.
  - One approach is that if a process is forced to wait when requesting a new resource, then all other resources previously held by this process are implicitly

released, (preempted), forcing this process to re-acquire the old resources along with the new resources in a single request, similar to the previous discussion.

4) Circular Wait

- One way to avoid circular wait is to number all resources, and to require that processes request resources only in strictly increasing (or decreasing) order.
- In other words, in order to request resource Rj, a process must first release all Ri such that i >= j.
- One big challenge in this scheme is determining the relative ordering of the different resources

CHAPTER – 6

2 MARKS

Q.1) Write file access method ? **{2016-S(7 e), 2019(s)-1-j }**

Ans:- An **access method** is a function of a mainframe operating system that enables access to data on disk, tape or other external devices.

It is of 3 types

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly

CHAPTER – 7

2 MARKS

Q.1) What is compiler ? **{2015-S (8 a)}**

Ans:- A **compiler** is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. The programmer then runs the appropriate language compiler, specifying the name of the file that contains the source statements.

- The entire program is verified so there are no syntax or semantic errors.
- The executable file is optimized by the compiler so it execute faster.
- A compiler is likely to perform many or all of the following operations: preprocessing, lexical analysis, parsing, semantic analysis

Q.2) What is Function of loader ? {2016-S (7 b),2018(s)-1-b}

Ans:- In computer **systems** a **loader** is the part of an **operating system** that is responsible for loading programs and libraries. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution.

- 1) It allocates the space for program in the memory, by calculating the size of the program. This activity is called allocation.
- 2) It resolves the symbolic references (code/data) between the object modules by assigning all the user subroutine and library subroutine addresses. This activity is called linking.
- 3) There are some address dependent locations in the program, such address constants must be adjusted according to allocated space, such activity done by loader is called relocation.
- 4) Finally it places all the machine instructions and data of corresponding programs and subroutines into the memory. Thus program now becomes ready for execution, this activity is called loading.

Q.3) Define System Programming ? {2016-S(2 a) , 2017-S (7 a)}

Ans:- System programming is an essential and important foundation in any computer's application development, and always evolving to accommodate changes in the computer hardware.

- The **system programming** enhances or extends the functions of an **operating system** and may comprise components such as drivers, utilities and updates.
- Such programming are designed for writing **system** software, which usually requires different development approaches when compared with application software.

Q.4) Define System's software ? {2015-S(6 a)}

Ans:- System software is a type of computer program that is designed to run a computer's hardware and application programs. If we think of the computer system as a layered model, the system software is the interface between the hardware and user applications.

- The operating system (OS) is the best-known example of system software. The OS manages all the other programs in a computer.

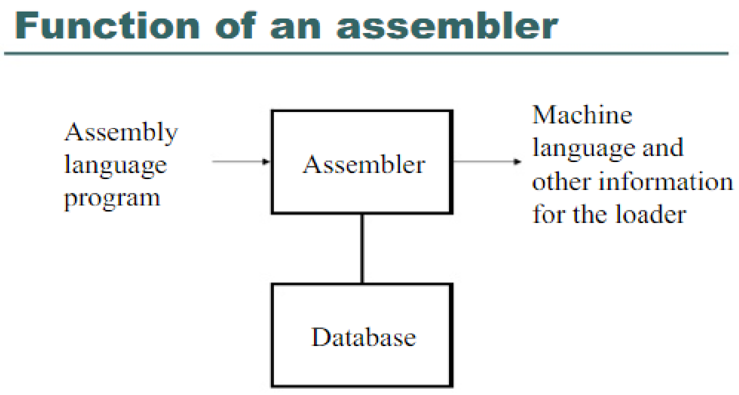
5 MARKS

Q.1) What is Assembler ? Explain function of an Assembler.

{2017-S (7 b),2015-S (6 b), 2018(s)-2-b ,2019(s)-2-f}

Ans:- Assembler :

An **assembler** is a program that takes basic computer instructions and converts them into a pattern of bits that the computer's processor can use to perform its basic operations. Some people call these instructions assembler language and others use the term assembly language.


Function of an assembler
Assembly language program
Assembler
Machine language and other information for the loader
Database

Function of an Assembler

- An assembler enables software and application developers to access, operate and manage a computer's hardware architecture and components.
- An assembler is sometimes referred to as the compiler of assembly language. It also provides the services of an interpreter.
- An assembler primarily serves as the bridge between symbolically coded instructions written in assembly language and the computer processor, memory and other computational components.
- Assigning machine addresses to symbolic labels

Q.2) What is the function of compiler ? Explain the seven phases.

{2017-S (7 c),2015-S (8 a), 2018(s)-1-b,2019(s)-3 }

Ans:- A **compiler** is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. ... The programmer then runs the appropriate language compiler, specifying the name of the file that contains the source statements.

- The entire program is verified so there are no syntax or semantic errors.
- The executable file is optimized by the compiler so it execute faster.
- A compiler is likely to perform many or all of the following operations: preprocessing, lexical analysis, parsing, semantic analysis

Phases of a Compiler

The phases of a compiler are as follows

1) Lexical Analysis

2) Syntax Analysis

3) Semantic Analysis

4)Intermediate Code Generation

5)Code Optimization

6) Code Generation

7) All of the aforementioned phases involve the following tasks:

• Symbol table management

• Error handling

Lexical Analysis

• Lexical analysis is the first phase of compiler which is also termed as scanning.

• Source program is scanned to read the stream of characters and those characters are grouped to form a sequence called lexemes which produces token as output

•**Token:** Token is a sequence of characters that represent lexical unit, which matches with the pattern, such as keywords, operators, identifiers etc.

Syntax Analysis

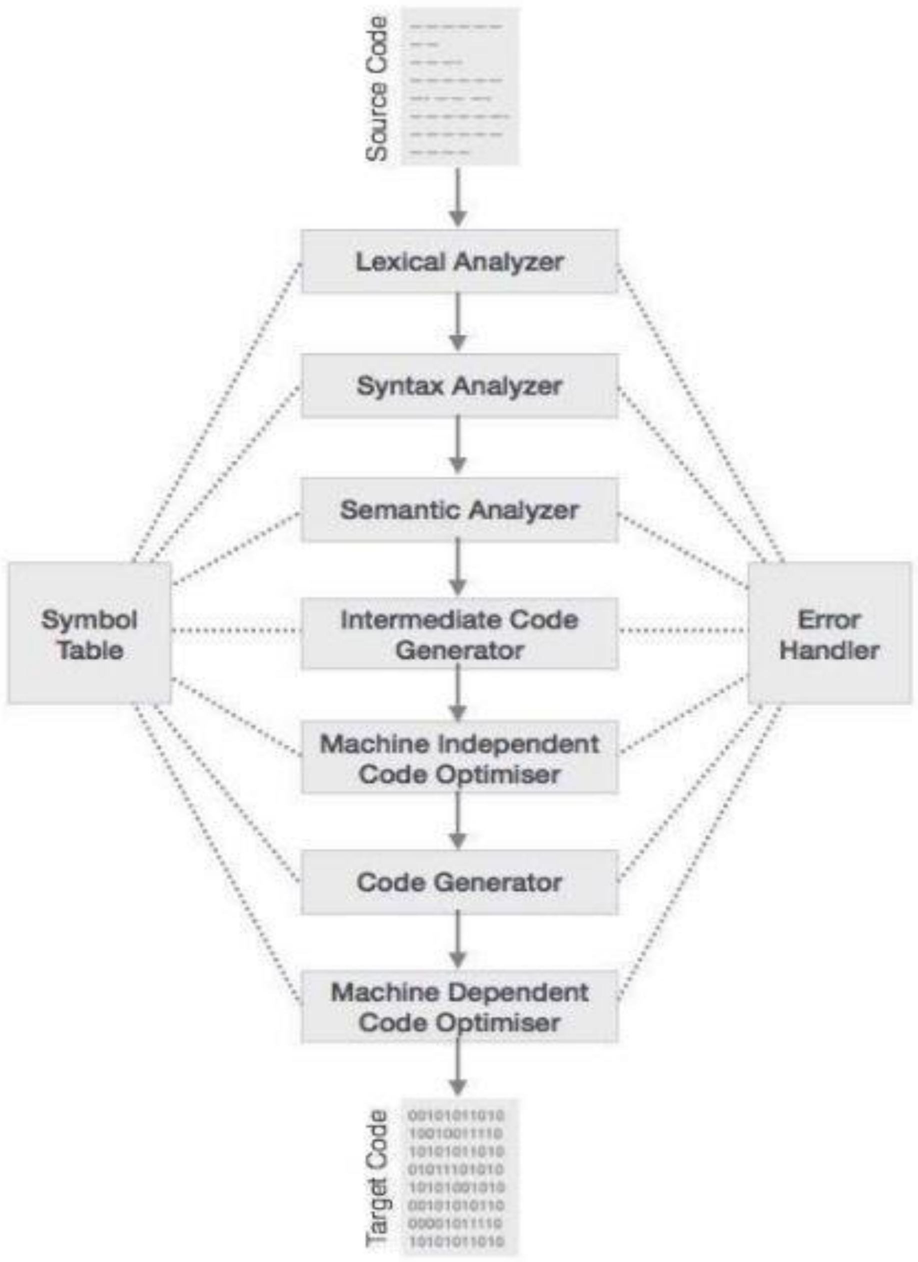
• Syntax analysis is the second phase of compiler which is also called as parsing.

• Parser converts the tokens produced by lexical analyzer into a tree like representation called parse tree.

• A parse tree describes the syntactic structure of the input.

• Syntax tree is a compressed representation of the parse tree in which the operators appear as interior nodes and the operands of the operator are the children of the node for that operator.

Input: *Tokens*



Semantic Analysis

- Semantic analysis is the third phase of compiler.
- It checks for the semantic consistency.
- Type information is gathered and stored in symbol table or in syntax tree.
- Performs type checking.

Intermediate Code Generation

- After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

Code Optimization

• Code optimization phase gets the intermediate code as input and produces optimized intermediate code as output.

• It results in faster running machine code.

• It can be done by reducing the number of lines of code for a program.

• This phase reduces the redundant code and attempts to improve the intermediate code so that faster-running machine code will result.

• During the code optimization, the result of the program is not affected.

• To improve the code generation, the optimization involves

- ➢ Deduction and removal of dead code (unreachable code).
- ➢ Calculation of constants in expressions and terms.
- ➢ Collapsing of repeated expression into temporary string.

Code Generation

• Code generation is the final phase of a compiler.

• It gets input from code optimization phase and produces the target code or object code as result.

• Intermediate instructions are translated into a sequence of machine instructions that perform the same task.

Symbol Table Management

• Symbol table is used to store all the information about identifiers used in the program.

• It is a data structure containing a record for each identifier, with fields for the attributes of the identifier.

• It allows finding the record for each identifier quickly and to store or retrieve data from that record.

Error Handling

• Each phase can encounter errors. After detecting an error, a phase must handle the error so that compilation can proceed.

• In lexical analysis, errors occur in separation of tokens.

• In syntax analysis, errors occur during construction of syntax tree.

Q. 3) Difference between interpreter and compiler. Name the seven phase of compiler.

{2016-S (2 c), 2019(s)-1-d }

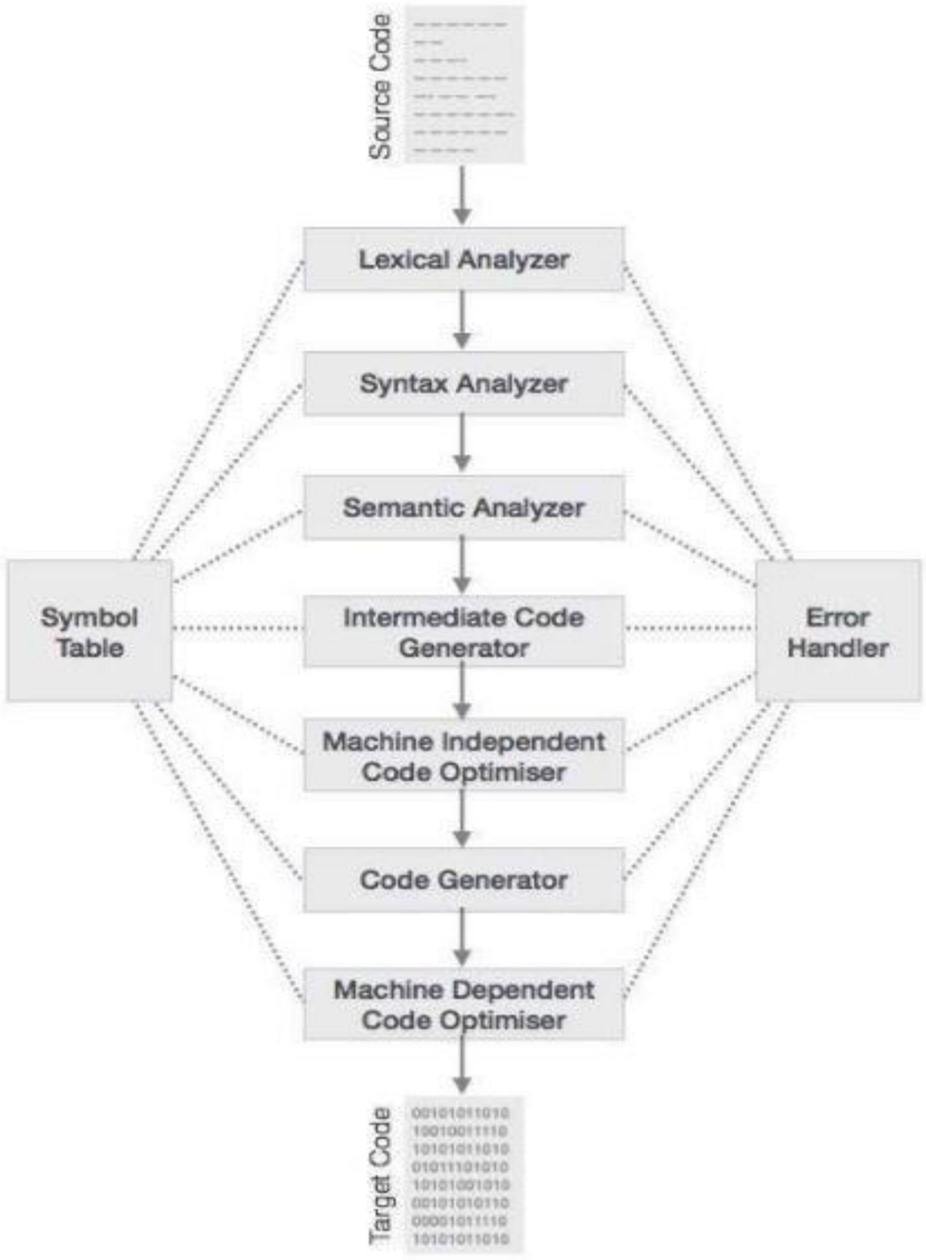
Interpreter	**Compiler**
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

Name of 7 Phases of a Compiler

The phases of a compiler are as follows

1) Lexical Analysis
2) Syntax Analysis
3) Semantic Analysis
4)Intermediate Code Generation
5)Code Optimization
6) Code Generation
7) All of the aforementioned phases involve the following tasks:

- Symbol table management
- Error handling



7 MARKS

Q. 1) What is Function of loader ? Explain different types of loader. {2015 -S (6 c)}

Ans:- In computer systems a **loader** is the part of an **operating system** that is responsible for loading programs and libraries. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution.

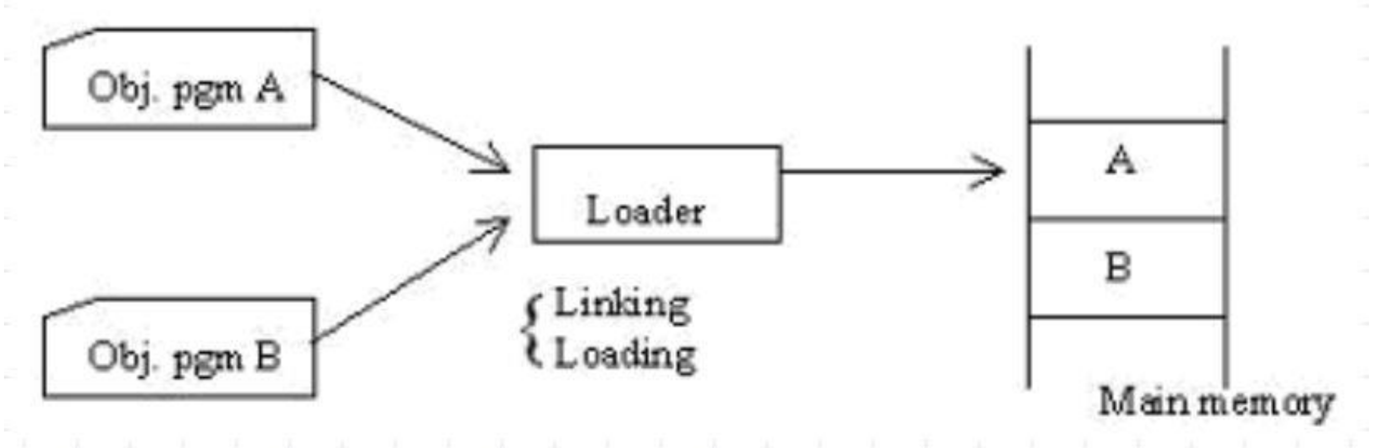
Function of loader

1) It allocates the space for program in the memory, by calculating the size of the program. This activity is called allocation.

2) It resolves the symbolic references (code/data) between the object modules by assigning all the user subroutine and library subroutine addresses. This activity is called linking.

3) There are some address dependent locations in the program, such address constants must be adjusted according to allocated space, such activity done by loader is called relocation.

4) Finally it places all the machine instructions and data of corresponding programs and subroutines into the memory. Thus program now becomes ready for execution, this activity is called loading.



Types of loader

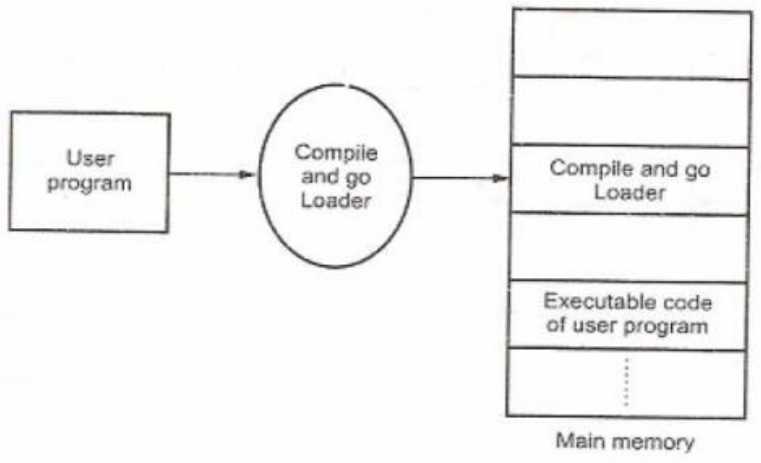
Based on the various functionalities of loader, there are various types of loaders:

1) "compile and go" loader

2) General Loader Scheme

3) Absolute Loader

1) "compile and go" loader:

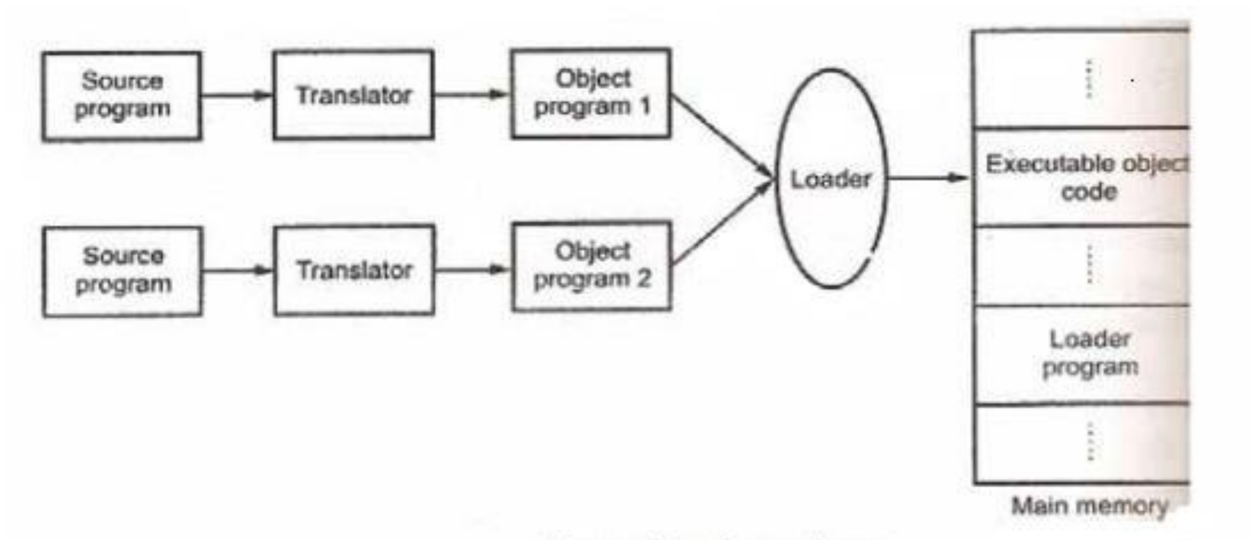


Compile and go loading scheme

In this type of loader, the instruction is read line by line, its machine code is obtained and it is directly put in the main memory at some known address. That means the assembler runs in one part of memory and the assembled machine instructions and data is directly put into their assigned memory locations. After completion of assembly process, assign starting address of the program to the location counter. The typical example is WATFOR-77, it's a FORTRAN compiler which uses such "load and go" scheme. This loading scheme is also called as "assemble and go".

2) General Loader Scheme:

In this loader scheme, the source program is converted to object program by some translator (assembler). The loader accepts these object modules and puts machine instruction and data in an executable form at their assigned memory. The loader occupies some portion of main memory.

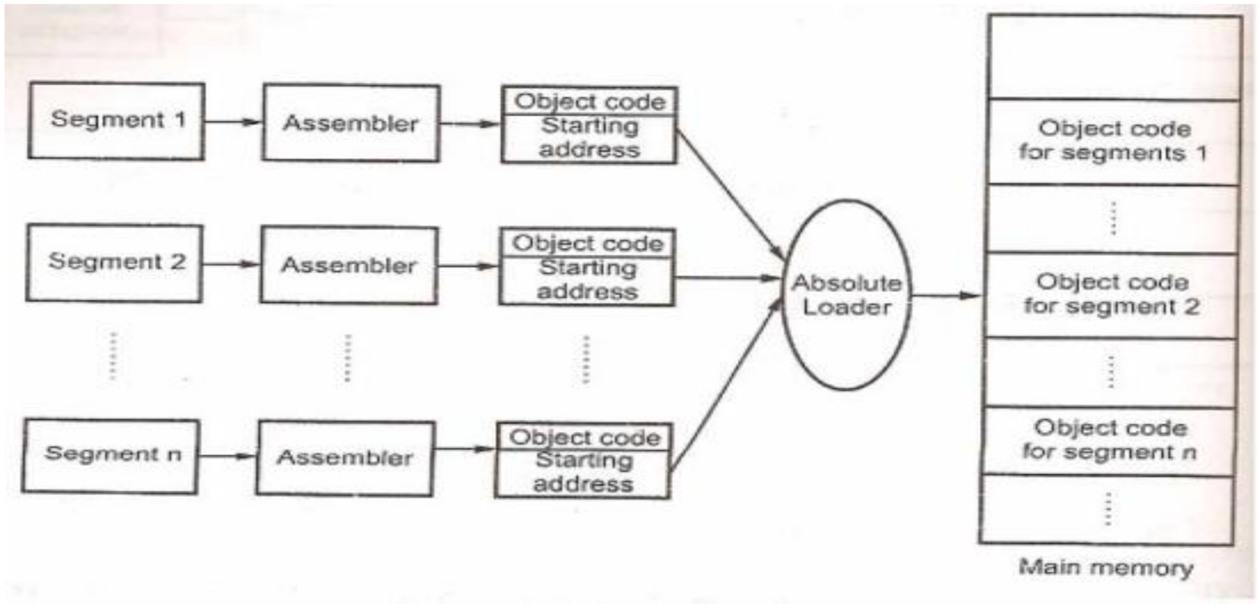


General loader scheme

3)

Absolute Loader:

Absolute loader is a kind of loader in which relocated object files are created, loader accepts these files and places them at specified locations in the memory. This type of loader is called absolute because no relocation information is needed; rather it is obtained from the programmer or assembler. The starting address of every module is known to the programmer, this corresponding starting address is stored in the object file, then task of loader becomes very simple and that is to simply place the executable form of the machine instructions at the locations mentioned in the object file


Segment 1
Assembler
Object code Starting address
Segment 2
Assembler
Object code Starting address
Segment n
Assembler
Object code Starting address
Absolute Loader
Object code for segments 1
Object code for segment 2
Object code for segment n
Main memory

Process of absolute loading

Q. 13) Explain I/O traffic control and I/O device handlers. {2015-S (7 c)}

Ans:- I/O Traffic Controller

It monitors the status of every device, control unit, and channel.

- Three main tasks:

  1. determine if there's at least one path available

  2. if more than one path, determines which to select

  3. if paths are all busy, determines when one will become available

- Maintains a database containing status and connections for each unit in the I/O subsystem grouped as:

  1. Channel Control Blocks

  2. Control Unit Control Blocks

  3. Device Control Blocks

- To choose a free path; traces backward from the control block of the requested device through the control units to the channels
- If no path available the Process is linked to the queues kept in the Control Blocks
- Creates multiple wait queues with one queue per path

Device Controllers

Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

- There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.
- Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.

IO device handler

ˆ IO device handler processes the IO interrupts, handles error conditions, and provides detailed scheduling algorithms

Q. 14) What is Device allocation ? Explain the function of I/O traffic controller and I/O scheduler. {2016-S (4 c), 2019(s)-5}

Ans:- Device allocation is the assignment of input/output devices and volumes to job steps. Requests for device allocation come from data definition (DD) statements and dynamic device allocation requests.

- Installation programs that run on the system can specify dynamic device allocation/unallocation requests.
- To control the amount of work needed for device allocation, you might want to restrict device allocation requests. You can define default values for allocation processing in ALLOCxx of the parmlib concatenation. ALLOCxx allows your installation to specify space, data set, and other allocation parameters for dynamic allocation requests.

While allocating devices, the system might ask operators to:

- Mount or dismount volumes
- Make decisions (for example, to bring a device online immediately or to wait)

Device Controllers

Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

- There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.
- Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.

Q. 15) Short notes on Application programs. {2015-S (8 d)}

Ans:- Application programs

An application program is a comprehensive, self-contained program that performs a particular function directly for the user. Among many others, application programs include:

- Email
- Web browsers
- Games
- Word processors
- Enterprise software
- Accounting software
- Graphics software
- Media players
- Database management

Because every program has a particular application for the end user, the term "application" is used. For instance, a word processor can help the user create an article, whereas a game application can be used for entertainment.

An application program is also known as an application or application software

Some examples of application programs include:

- Application suite: Includes various applications packaged together.
- Enterprise software: Addresses the data flow and process requirements of an organization, covering entire departments.
- Information worker software: Permits users to create and administer information.
- Content access software: Used mainly to gain access to content without editing.
- Media development software: Creates electronic and print media.
- Educational software: Includes content and/or features intended for students or educators.
- Product engineering software: Develops software and hardware products.