Tanvir Ahmed_IT18043_OS_CT - 02

1.a) What is Deadlock?Give an example.
b) Difference Between Starvation and Deadlock.
c) Necessary conditions for Deadlocks. What are the Advantages and Disadvantages of Deadlock?

2.a) What is Deadlock Detection? What are the Strategies for handling Deadlock?
b) How does Deadlock Prevention work? Explain the Recovery from Deadlock.
c) Describe data structures of the banker's algorithm?

3.a) What is Memory Management?Why Use Memory Management?
b) Memory Management Techniques.
c) What is Swapping?Benefits of Swapping

4.a) What is Memory allocation? Explain Partition Allocation.
b) What is Paging?What is Fragmentation?Difference Between Static and Dynamic Linking
c) What is Segmentation?Difference Between Static and Dynamic Loading.

5.a) What is virtual memory? What is the page fault? Write the causes of Thrashing?
b) What are the aspects of demand paging?
c) Paging VS Segmentation.

6.a) Write about Page Replacement Methods.
b) What is disk scheduling? Write various disk scheduling algorithms.
c) What is Physical and Logical Address Space? What is the Word and Page Table?

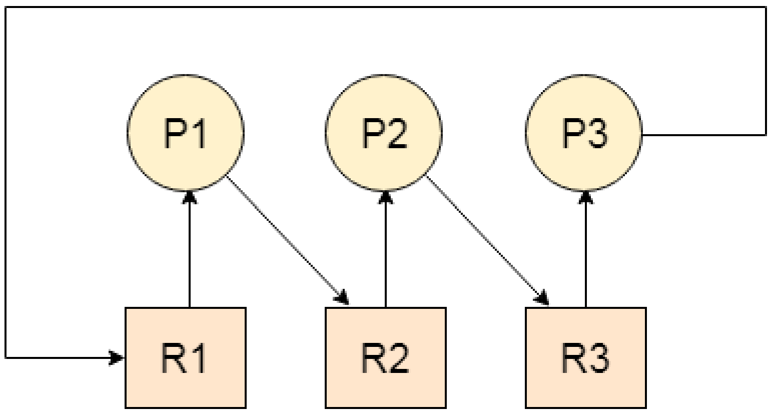
7.a) Describe the two types of input/output devices.What are the two parts of I/O devices?
b) Write about Direct Memory Access (DMA).
c) What is Interrupt? Describe three types of interrupts.

8.a) What is File System? Properties of a File System?Objective of File management System?
b) Explain File structure and File Attributes.Describe the types of file.What are the Functions of File?
c) What are the File Access Methods?Three types of space allocation methods of file.
d) What is File Directory?What are the information which is maintained in a directory?Give the chart containing file types, name and extension.

Ans to the Question NO - 1(a)

Deadlock is a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software.

Example: Let us assume that there are three processes P1, P2 and P3. There are three different resources R1, R2 and R3. R1 is assigned to P1, R2 is assigned to P2 and R3 is assigned to P3.
After some time, P1 demands for R1 which is being used by P2. P1 halts its execution since it can't complete without R2. P2 also demands for R3 which is being used by P3. P2 also stops its execution because it can't continue without R3. P3 also demands for R1 which is being used by P1 therefore P3 also stops its execution.
In this scenario, a cycle is being formed among the three processes. None of the process is progressing and they are all waiting. The computer becomes unresponsive since all the processes got blocked.



Ans to the Question NO - 1(b)

Difference between Deadlock and Starvation:

S.NO	Deadlock	Starvation

1.	All processes keep waiting for each other to complete and none get executed	High priority processes keep executing and low priority processes are blocked
2.	Resources are blocked by the processes	Resources are continuously utilized by high priority processes
3.	Necessary conditions Mutual Exclusion, Hold and Wait, No preemption, Circular Wait	Priorities are assigned to the processes
4.	Also known as Circular wait	Also know as lived lock
5.	It can be prevented by avoiding the necessary conditions for deadlock	It can be prevented by Aging

Ans to the Question NO - 1(c)

Difference Between Starvation and Deadlock

Deadlock	**Starvation**
The deadlock situation occurs when one of the processes got blocked.	Starvation is a situation where all the low priority processes got blocked, and the high priority processes execute.
Deadlock is an infinite process.	Starvation is a long waiting but not an infinite process.
Every Deadlock always has starvation.	Every starvation does n't necessarily have a deadlock.

Deadlock happens then Mutual exclusion, hold and wait. Here, preemption and circular wait do not occur simultaneously.	It happens due to uncontrolled priority and resource management.

Ans to the Question NO - 1(c)

Necessary conditions for Deadlocks

1. **Mutual Exclusion**
2. A resource can only be shared in mutually exclusive manner. It implies, if two process cannot use the same resource at the same time.**Hold and Wait**
3. A process waits for some resources while holding another resource at the same time.**No preemption**
4. The process which once scheduled will be executed till the completion. No other process can be scheduled by the scheduler meanwhile.**Circular Wait**

All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

Advantages of Deadlock

Here, are pros/benefits of using Deadlock method

- This situation works well for processes which perform a single burst of activity
- No preemption needed for Deadlock.
- Convenient method when applied to resources whose state can be saved and restored easily
- Feasible to enforce via compile-time checks
- Needs no run-time computation since the problem is solved in system design

Disadvantages of Deadlock method

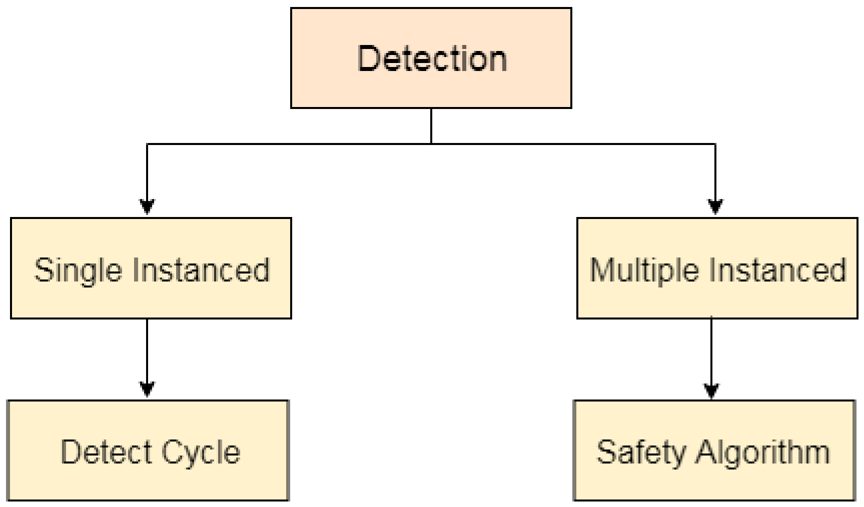
Here, are cons/ drawback of using deadlock method

- Delays process initiation
- Processes must know future resource need
- Pre-empts more often than necessary
- Dis-allows incremental resource requests
- Inherent preemption losses.

Ans to the Question NO - 2(a)

A deadlock occurrence can be detected by the resource scheduler. A resource scheduler helps OS to keep track of all the resources which are allocated to different processes.

The OS can detect the deadlocks with the help of Resource allocation graph.



In single instanced resource types, if a cycle is being formed in the system then there will definitely be a deadlock. On the other hand, in multiple instanced resource type graph, detecting a cycle is not just enough. We have to apply the safety algorithm on the system by converting the resource allocation graph into the allocation matrix and request matrix.

Strategies for handling Deadlock

1. Deadlock Ignorance

Deadlock Ignorance is the most widely used approach among all the mechanism. This is being used by many operating systems mainly for end user uses. In this approach, the Operating system assumes that deadlock never occurs. It simply ignores deadlock. This approach is best suitable for a single end user system where User uses the system only for browsing and all other normal stuff.

There is always a tradeoff between Correctness and performance. The operating systems like Windows and Linux mainly focus upon performance. However, the performance of the system decreases if it uses deadlock handling mechanism all the time if deadlock happens 1 out of 100 times then it is completely unnecessary to use the deadlock handling mechanism all the time.

In these types of systems, the user has to simply restart the computer in the case of deadlock. Windows and Linux are mainly using this approach.

2. Deadlock prevention

Deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously. If it is possible to violate one of the four conditions at any time then the deadlock can never occur in the system.

The idea behind the approach is very simple that we have to fail one of the four conditions but there can be a big argument on its physical implementation in the system.

We will discuss it later in detail.

3. Deadlock avoidance

In deadlock avoidance, the operating system checks whether the system is in safe state or in unsafe state at every step which the operating system performs. The process continues until the system is in safe state. Once the system moves to unsafe state, the OS has to backtrack one step.

In simple words, The OS reviews each allocation so that the allocation doesn't cause the deadlock in the system.

We will discuss Deadlock avoidance later in detail.

4. Deadlock detection and recovery

This approach let the processes fall in deadlock and then periodically check whether deadlock occur in the system or not. If it occurs then it applies some of the recovery methods to the system to get rid of deadlock.

Ans to the Question NO - 2(b)

Deadlock Prevention: If we simulate deadlock with a table which is standing on its four legs then we can also simulate four legs with the four conditions which when occurs simultaneously, cause the deadlock.

However, if we break one of the legs of the table then the table will fall definitely. The same happens with deadlock, if we can be able to violate one of the four necessary conditions and don't let them occur together then we can prevent the deadlock.

Let's see how we can prevent each of the conditions.

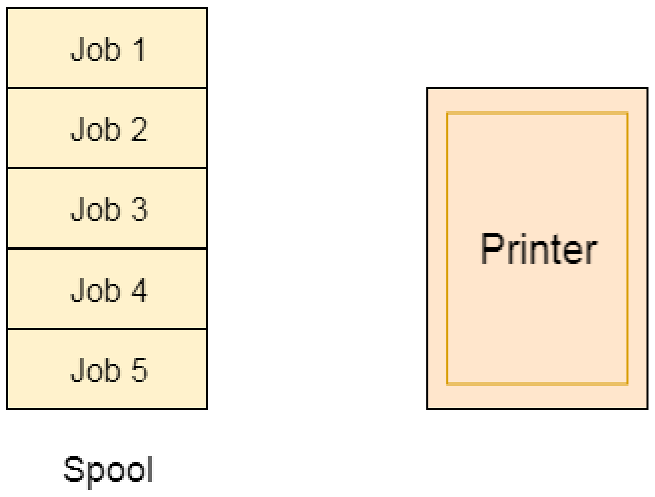
1. Mutual Exclusion

Mutual section from the resource point of view is the fact that a resource can never be used by more than one process simultaneously which is fair enough but that is the main reason behind the deadlock. If a resource could have been used by more than one process at the same time then the process would have never been waiting for any resource.

However, if we can be able to violate resources behaving in the mutually exclusive manner then the deadlock can be prevented.

Spooling

For a device like printer, spooling can work. There is a memory associated with the printer which stores jobs from each of the process into it. Later, Printer collects all the jobs and print each one of them according to FCFS. By using this mechanism, the process doesn't have to wait for the printer and it can continue whatever it was doing. Later, it collects the output when it is produced.



Although, Spooling can be an effective approach to violate mutual exclusion but it suffers from two kinds of problems.

1. This cannot be applied to every resource.
2. After some point of time, there may arise a race condition between the processes to get space in that spool.

We cannot force a resource to be used by more than one process at the same time since it will not be fair enough and some serious problems may arise in the performance. Therefore, we cannot violate mutual exclusion for a process practically.

2. Hold and Wait

Hold and wait condition lies when a process holds a resource and waiting for some other resource to complete its task. Deadlock occurs because there can be more than one process which are holding one resource and waiting for other in the cyclic order.

However, we have to find out some mechanism by which a process either doesn't hold any resource or doesn't wait. That means, a process must be assigned all the necessary resources before the execution starts. A process must not wait for any resource once the execution has been started.

!(Hold and wait) = !hold or !wait (negation of hold and wait is, either you don't hold or you don't wait)

This can be implemented practically if a process declares all the resources initially. However, this sounds very practical but can't be done in the computer system because a process can't determine necessary resources initially.

Process is the set of instructions which are executed by the CPU. Each of the instruction may demand multiple resources at the multiple times. The need cannot be fixed by the OS.

The problem with the approach is:

1. Practically not possible.
2. Possibility of getting starved will be increases due to the fact that some process may hold a resource for a very long time.

3. No Preemption

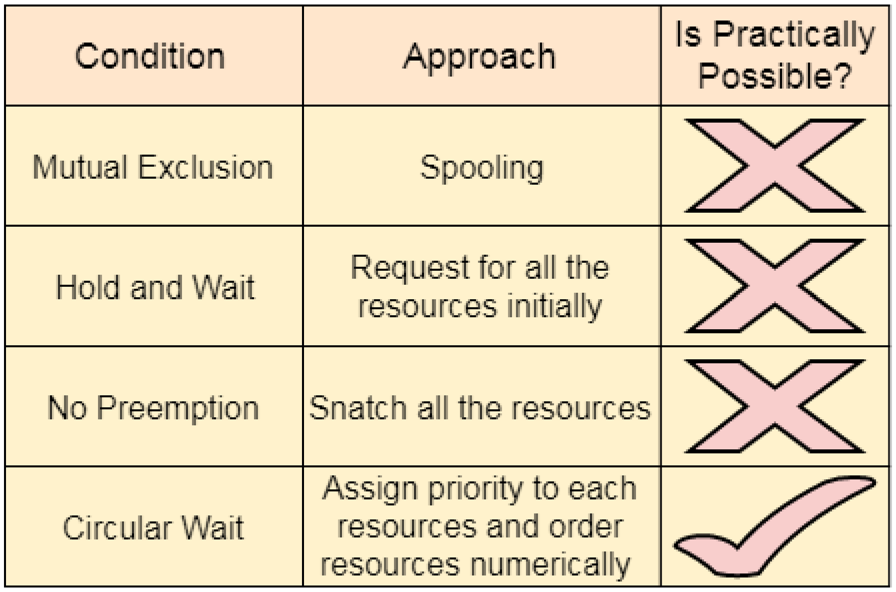
Deadlock arises due to the fact that a process can't be stopped once it starts. However, if we take the resource away from the process which is causing deadlock then we can prevent deadlock.

This is not a good approach at all since if we take a resource away which is being used by the process then all the work which it has done till now can become inconsistent.

Consider a printer is being used by any process. If we take the printer away from that process and assign it to some other process then all the data which has been printed can become inconsistent and ineffective and also the fact that the process can't start printing again from where it has left which causes performance inefficiency.

4. Circular Wait

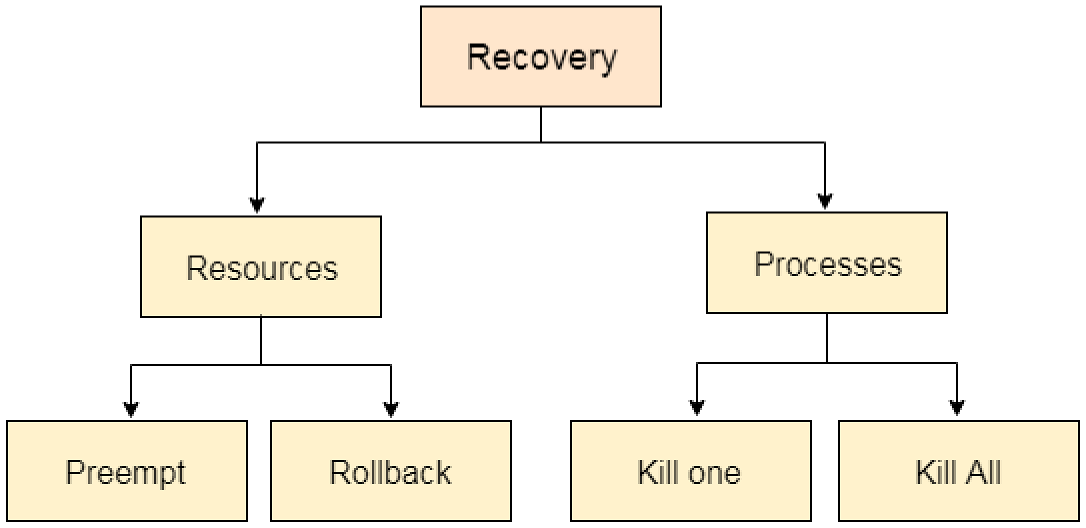
To violate circular wait, we can assign a priority number to each of the resource. A process can't request for a lesser priority resource. This ensures that not a single process can request a resource which is being utilized by some other process and no cycle will be formed.



Condition	Approach	Is Practically Possible?
Mutual Exclusion	Spooling	✗
Hold and Wait	Request for all the resources initially	✗
No Preemption	Snatch all the resources	✗
Circular Wait	Assign priority to each resources and order resources numerically	✓

Among all the methods, violating Circular wait is the only approach that can be implemented practically.

Deadlock Recovery : In order to recover the system from deadlocks, either OS considers resources or processes.



For Resource

Preempt the resource

We can snatch one of the resources from the owner of the resource (process) and give it to the other process with the expectation that it will complete the execution and will release this resource sooner. Well, choosing a resource which will be snatched is going to be a bit difficult.

Rollback to a safe state

System passes through various states to get into the deadlock state. The operating system canrollback the system to the previous safe state. For this purpose, OS needs to implement check pointing at every state.

The moment, we get into deadlock, we will rollback all the allocations to get into the previous safe state.

For Process

Kill a process

Killing a process can solve our problem but the bigger concern is to decide which process to kill. Generally, Operating system kills a process which has done least amount of work until now.

Kill all process

This is not a suggestible approach but can be implemented if the problem becomes very serious. Killing all process will lead to inefficiency in the system because all the processes will execute again from starting.

Ans to the Question NO - 2(c)

Banker's Algorithm Notations

Here is an important notation used in Banker's algorithm:

- X: Indicates the total number of processes of the system.
- Y: Indicates the total number of resources present in the system.

Available

[I: Y] indicate which resource is available.

Max

[I:X,I: Y]: Expression of the maximum number of resources of type j or process i

Allocation

[I:X,I:Y]. Indicate where process you have received a resource of type j

Need

Express how many more resources can be allocated in the future

Example of Banker's algorithm

Assume that we have the following resources:

- 5 Pen drives
- 2 Printers
- 4 Scanners
- 3 Hard disks

Here, we have created a vector representing total resources: Available = (5, 2, 4, 3).

Assume there are four processes. The available resources are already allocated as per the matrix table below.

Process Name	**Pen Drives**	**Printer**	**Scanner**	**Hard disk**
P	2	0	1	1
Q	0	1	0	0
R	1	0	1	1
S	1	1	0	1
Total	4	2	2	3

Here, the allocated resources is the total of these columns:

Allocated = (4, 2, 2, 3).

We also create a Matrix to display the number of each resource required for all the processes. This matrix is called **Need**=(3,0,2,2)

Process Name	Pen Drives	Printer	Scanner	Hard disk
P	1	1	0	0
Q	0	1	1	2
R	2	1	0	0
S	0	0	1	0

The available vector will be :

Available=Available- Allocated

= (5, 2, 4, 3) -(4, 2, 2, 3)

=(1, 0, 2, 0)

Resource Request Algorithm

Resource request algorithm enables you to represent the system behavior when a specific process makes a resource request.

Let understand this by the following steps:

Step 1) When a total requested instance of all resources is lesser than the process, move to step 2.

Step 2) When a requested instance of each and every resource type is lesser compared to the available resources of each type, it will be processed to the next step. Otherwise, the process requires to wait because of the unavailability of sufficient resources.

Step 3) Resource is allocated as shown in the below given Pseudocode.

```
Available = Available – Request (y)
Allocation(x) = Allocation(x) + Request(x)
```

```
Need(x) = Need(x) - Request(x)
```

This final step is performed because the system needs to assume that resources have been allocated. So that there should be less resources available after allocation.

Characteristics of Banker's Algorithm

Here are important characteristics of banker's algorithm:

- Keep many resources that satisfy the requirement of at least one client
- Whenever a process gets all its resources, it needs to return them in a restricted period.
- When a process requests a resource, it needs to wait
- The system has a limited number of resources
- Advance feature for max resource allocation

Disadvantage of Banker's algorithm

Here, are cons/drawbacks of using banker's algorithm

- Does not allow the process to change its Maximum need while processing
- It allows all requests to be granted in restricted time, but one year is a fixed period for that.
- All processes must know and state their maximum resource needs in advance.

Ans to the Question NO - 3(a)

Memory Management is the process of controlling and coordinating computer memory, assigning portions known as blocks to various running programs to optimize the overall performance of the system.

It is the most important function of an operating system that manages primary memory. It helps processes to move back and forward between the main memory and execution disk. It helps OS to keep track of every memory location, irrespective of whether it is allocated to some process or it remains free.

Here, are reasons for using memory management:

1. It allows you to check how much memory needs to be allocated to processes that decide which processor should get memory at what time.
2. Tracks whenever inventory gets freed or unallocated. According to it will update the status.
3. It allocates the space to application routines.
4. It also makes sure that these applications do not interfere with each other.
5. Helps protect different processes from each other
6. It places the programs in memory so that memory is utilized to its full extent.

Ans to the Question NO - 3(b)

Here, are some most crucial memory management techniques:

Single Contiguous Allocation

It is the easiest memory management technique. In this method, all types of computer's memory except a small portion which is reserved for the OS is available for one application. For example, MS-DOS operating system allocates memory in this way. An embedded system also runs on a single application.

Partitioned Allocation

It divides primary memory into various memory partitions, which is mostly contiguous areas of memory. Every partition stores all the information for a specific task or job. This method consists of allotting a partition to a job when it starts & unallocate when it ends.

Paged Memory Management

This method divides the computer's main memory into fixed-size units known as page frames. This hardware memory management unit maps pages into frames which should be allocated on a page basis.

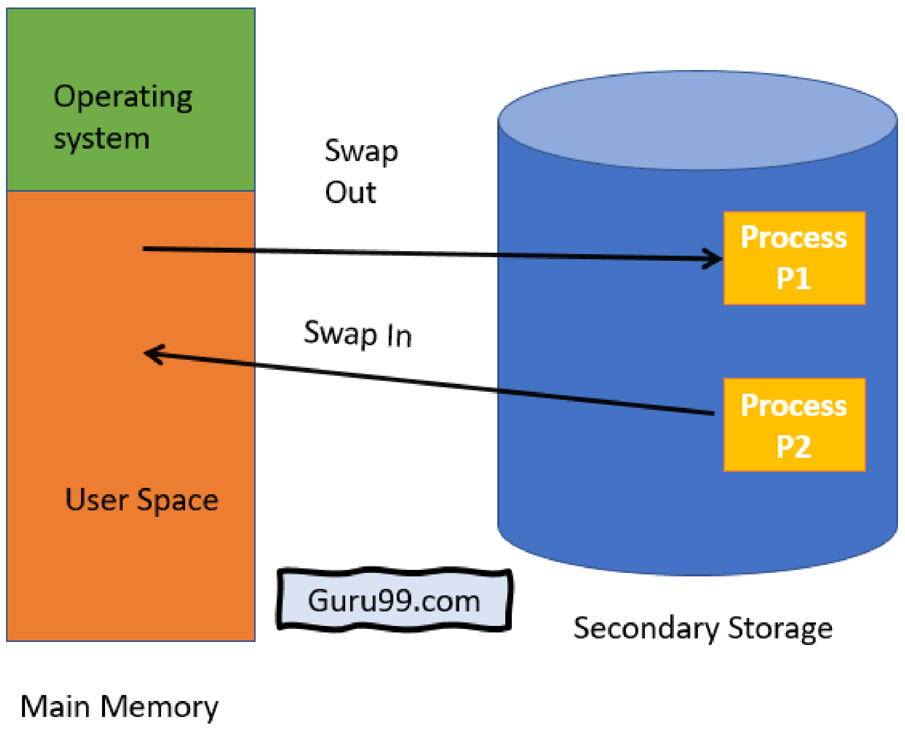
Segmented Memory Management

Segmented memory is the only memory management method that does not provide the user's program with a linear and contiguous address space.

Segments need hardware support in the form of a segment table. It contains the physical address of the section in memory, size, and other data like access protection bits and status.

Ans to the Question NO - 3(c)

Swapping is a method in which the process should be swapped temporarily from the main memory to the backing store. It will be later brought back into the memory for continue execution.
Backing store is a hard disk or some other secondary storage device that should be big enough inorder to accommodate copies of all memory images for all users. It is also capable of offering direct access to these memory images.



Benefits of Swapping

Here, are major benefits/pros of swapping:

- It offers a higher degree of multiprogramming.
- Allows dynamic relocation. For example, if address binding at execution time is being used, then processes can be swap in different locations. Else in case of compile and load time bindings, processes should be moved to the same location.
- It helps to get better utilization of memory.
- Minimum wastage of CPU time on completion so it can easily be applied to a priority-based scheduling method to improve its performance.

Ans to the Question NO - 4(a)

Memory allocation is a process by which computer programs are assigned memory or space.

Here, main memory is divided into two types of partitions

1. **Low Memory** - Operating system resides in this type of memory.
2. **High Memory**- User processes are held in high memory.

Partition Allocation

Memory is divided into different blocks or partitions. Each process is allocated according to the requirement. Partition allocation is an ideal method to avoid internal fragmentation.

Below are the various partition allocation schemes :

- **First Fit**: In this type fit, the partition is allocated, which is the first sufficient block from the beginning of the main memory.
- **Best Fit:** It allocates the process to the partition that is the first smallest partition among the free partitions.
- **Worst Fit:** It allocates the process to the partition, which is the largest sufficient freely available partition in the main memory.
- **Next Fit:** It is mostly similar to the first Fit, but this Fit, searches for the first sufficient partition from the last allocation point.

Ans to the Question NO - 4(b)

Paging is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages. In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept.

Fragmentation

Processes are stored and removed from memory, which creates free memory space, which are too small to use by other processes.

After sometimes, that processes not able to allocate to memory blocks because its small size and memory blocks always remain unused is called **fragmentation**. This type of problem happens during a dynamic memory allocation system when free blocks are quite small, so it is not able to fulfill any request.

Two types of Fragmentation methods are:

1. External fragmentation
2. Internal fragmentation

- **External fragmentation** can be reduced by rearranging memory contents to place all free memory together in a single block.
- The **internal fragmentation** can be reduced by assigning the smallest partition, which is still good enough to carry the entire process.

Here, are main difference between Static vs. Dynamic Linking:

Static Linking	**Dynamic Linking**
Static linking is used to combine all other modules, which are required by a program into a single executable code. This helps OS prevent any runtime dependency.	When dynamic linking is used, it does not need to link the actual module or library with the program. Instead of it use a reference to the dynamic module provided at the time of compilation and linking.

Ans to the Question NO - 4(c)

Segmentation method works almost similarly to paging. The only difference between the two is that segments are of variable-length, whereas, in the paging method, pages are always of fixed size.

A program segment includes the program's main function, data structures, utility functions, etc. The OS maintains a segment map table for all the processes. It also includes a list of free memory blocks along with its size, segment numbers, and its memory locations in the main memory or virtual memory.

Difference Between Static and Dynamic Loading

Static Loading	Dynamic Loading
Static loading is used when you want to load your program statically. Then at the time of compilation, the entire program will be linked and compiled without need of any external module or program dependency.	In a Dynamically loaded program, references will be provided and the loading will be done at the time of execution.
At loading time, the entire program is loaded into memory and starts its execution.	Routines of the library are loaded into memory only when they are required in the program.

Ans to the Question NO - 5(a)

Virtual Memory is a storage mechanism which offers user an illusion of having a very big main memory. It is done by treating a part of secondary memory as the main memory. In Virtual memory, the user can store processes with a bigger size than the available main memory.

Therefore, instead of loading one long process in the main memory, the OS loads the various parts of more than one process in the main memory. Virtual memory is mostly implemented with demand paging and demand segmentation.

A **page fault** occurs when a program attempts to access data or code that is in its address space, but is not currently located in the system RAM.

Thrashing happen when your system starts spending more time in doing paging rather than performing computation. Thrashing results in severe performance problems.
Thrashing is the state of a process where there is high paging activity. A process that is spending more time paging than executing is said to be thrashing.

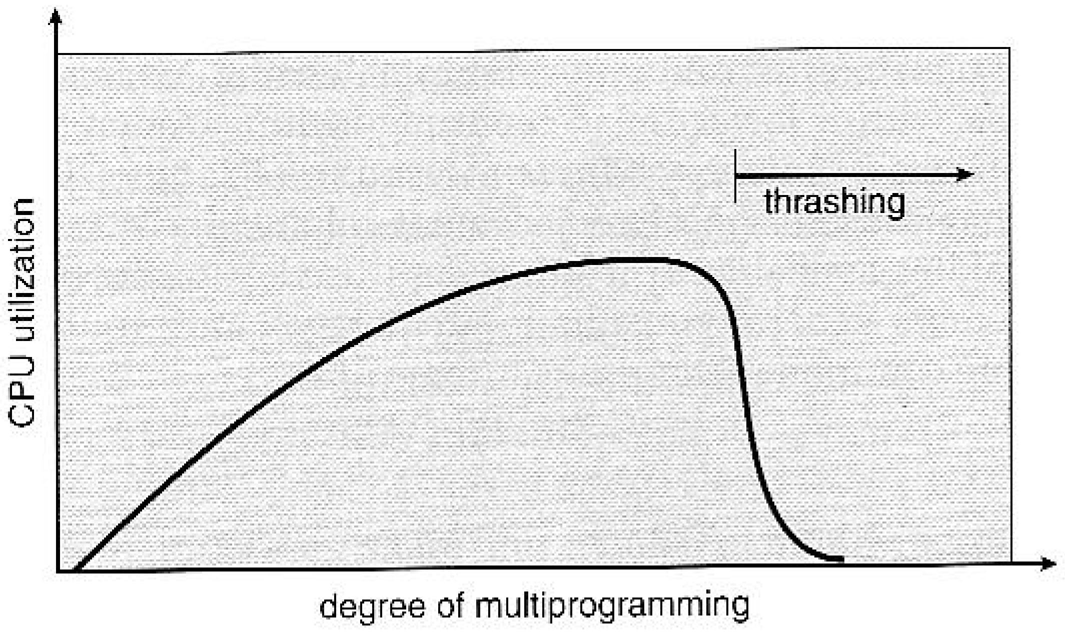


Figure 9.18 Thrashing.

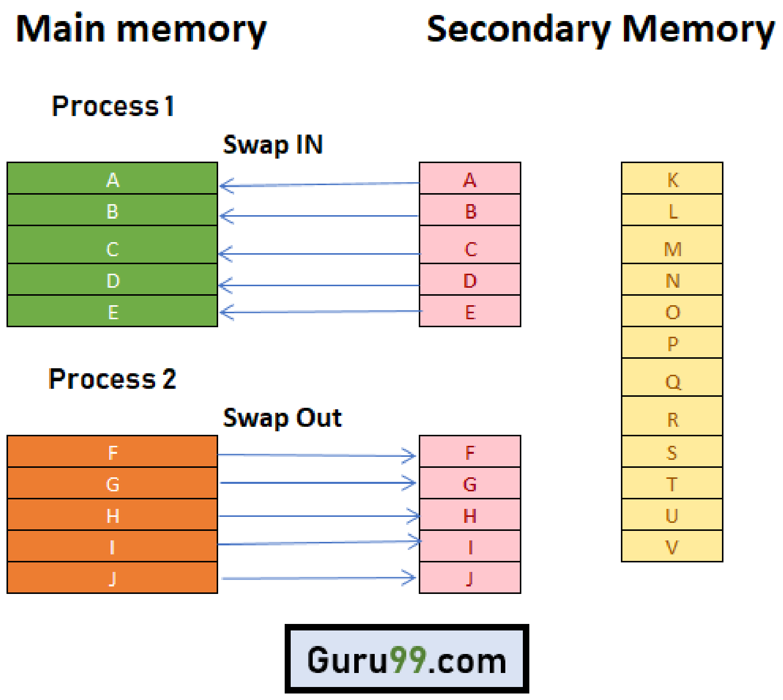
Cause of Thrashing

1. When memory is filled up and processes starts spending lots of time waiting for their pages to page in, then CPU utilization decreases(Processes are not executed as they are waiting for some pages), causing the scheduler to add in even more processes and increase the degree of multiprogramming even more. Thrashing has occurred, and system throughput plunges. No work is getting done, because the processes are spending all their time paging.

2. In the graph given below , CPU utilization is plotted against the degree of multiprogramming. As the degree of multiprogramming increases, CPU utilization also increases, although more slowly, until a maximum is reached. If the degree of multiprogramming is increased even further, thrashing sets in, and CPU utilization drops sharply. At this point, to increase CPU utilization and stop thrashing, we must decrease the degree of multiprogramming.

3. Local page replacement policies can prevent thrashing process from taking pages away from other processes, but it still tends to clog up the I/O queue.

Ans to the Question NO - 5(b)



A **demand paging** mechanism is very much similar to a paging system with swapping where processes stored in the secondary memory and pages are loaded only on demand, not in advance.
So, when a context switch occurs, the OS never copy any of the old program's pages from the disk or any of the new program's pages into the main memory. Instead, it will start executing the new program after loading the first page and fetches the program's pages, which are referenced.
During the program execution, if the program references a page that may not be available in the main memory because it was swapped, then the processor considers it as an invalid memory reference. That's because the page fault and transfers send control back from the program to the OS, which demands to store page back into the memory.

Ans to the Question NO - 5(c)

Here, are differences between Paging and Segmentation method:

Paging	**Segmentation**
A page is of the fixed block size.	A segment is of variable size.
It may lead to internal fragmentation.	It may lead to external fragmentation.
In Paging, the hardware decides the page size.	The segment size is specified by the user.
A process address space is broken into fixed-sized blocks, which is called pages.	A process address space Is broken in differing sized blocks called sections.
The paging technique is faster for memory access.	Segmentation is slower than paging method.
Page table stores the page data	Segmentation table stores the segmentation data.
Paging does not facilitate any sharing of procedures.	Segmentation allows for the sharing of procedures.
Paging fails to distinguish and secure procedures and data separately.	Segmentation can be able to separate secure procedures and data.
Paging address space is one dimensional	In segmentation, there is the availability of many independent address spaces
In paging, the user just provides a single integer as the address, that is divided by the hardware into a page number and offset.	In the segmentation method, the user specifies the address in two quantities 1)segment number 2)offset.

Ans to the Question NO - 6(a)

In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms :

FIFO Page Replacement

FIFO (First-in-first-out) is a simple implementation method. In this method, memory selects the page for a replacement that has been in the virtual address of the memory for the longest time.

Features:

- Whenever a new page loaded, the page recently comes in the memory is removed. So, it is easy to decide which page requires to be removed as its identification number is always at the FIFO stack.
- The oldest page in the main memory is one that should be selected for replacement first.

Optimal Algorithm

The optimal page replacement method selects that page for a replacement for which the time to the next reference is the longest.

Features:

- Optimal algorithm results in the fewest number of page faults. This algorithm is difficult to implement.
- An optimal page-replacement algorithm method has the lowest page-fault rate of all algorithms. This algorithm exists and which should be called MIN or OPT.
- Replace the page which unlike to use for a longer period of time. It only uses the time when a page needs to be used.

LRU Page Replacement

The full form of LRU is the Least Recently Used page. This method helps OS to find page usage over a short period of time. This algorithm should be implemented by associating a counter with an even- page.

How does it work?

- Page, which has not been used for the longest time in the main memory, is the one that will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

Features:

- The LRU replacement method has the highest count. This counter is also called aging registers, which specify their age and how much their associated pages should also be referenced.
- The page which hasn't been used for the longest time in the main memory is the one that should be selected for replacement.
- It also keeps a list and replaces pages by looking back into time.

Fault rate

Fault rate is a frequency with which a designed system or component fails. It is expressed in failures per unit of time. It is denoted by the Greek letter ? (lambda).

Ans to the Question NO - 6(b)

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

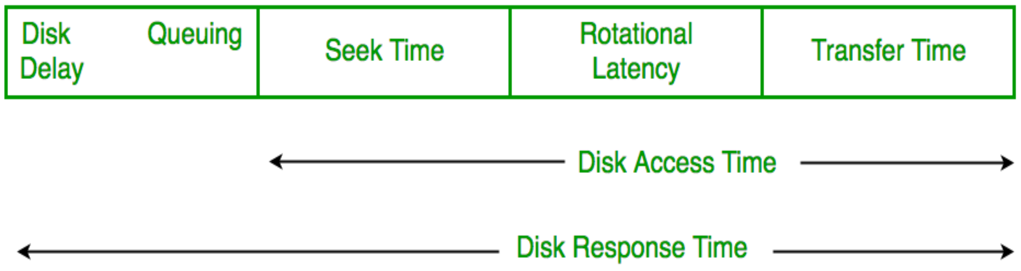
- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.

- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **Seek Time:**Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:** Disk Access Time is:

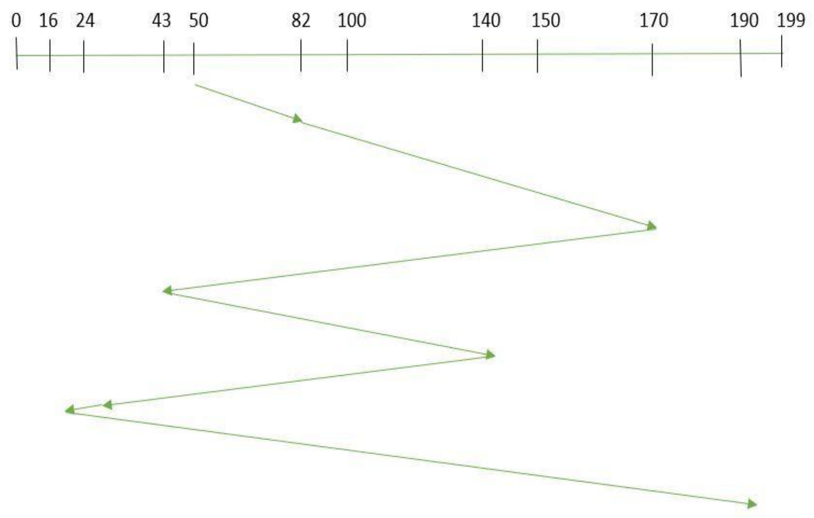
```
Disk Access Time = Seek Time +
                   Rotational Latency +
                   Transfer Time
```



- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

Disk Scheduling Algorithms

1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.Let us understand this with the help of an example.
   Example:
   Suppose the order of request is- (82,170,43,140,24,16,190)
   And current position of Read/Write head is : 50



So, total seek time:

=(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16)

=642

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time
- May not provide the best possible service

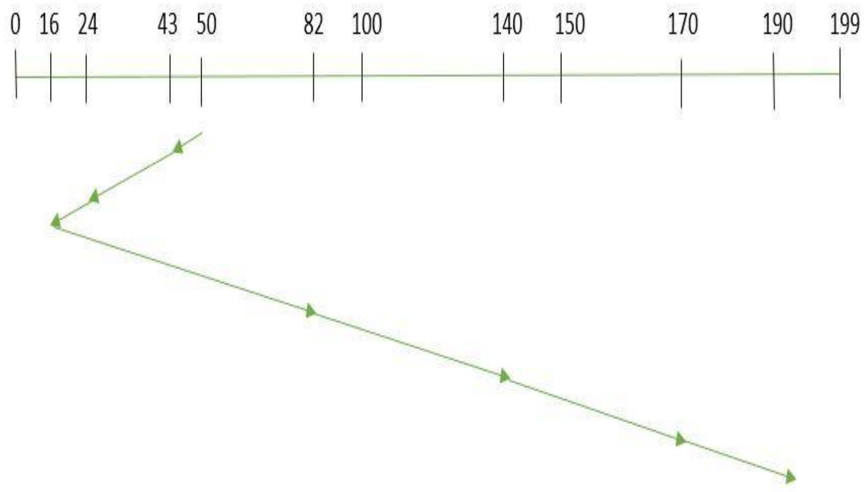
2. **SSTF:** In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in the queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over

FCFS as it decreases the average response time and increases the throughput of system.Let us understand this with the help of an example.

Example:

Suppose the order of request is- (82,170,43,140,24,16,190)

And current position of Read/Write head is : 50



So, total seek time:

=(50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-40)+(190-170)

=208

Advantages:

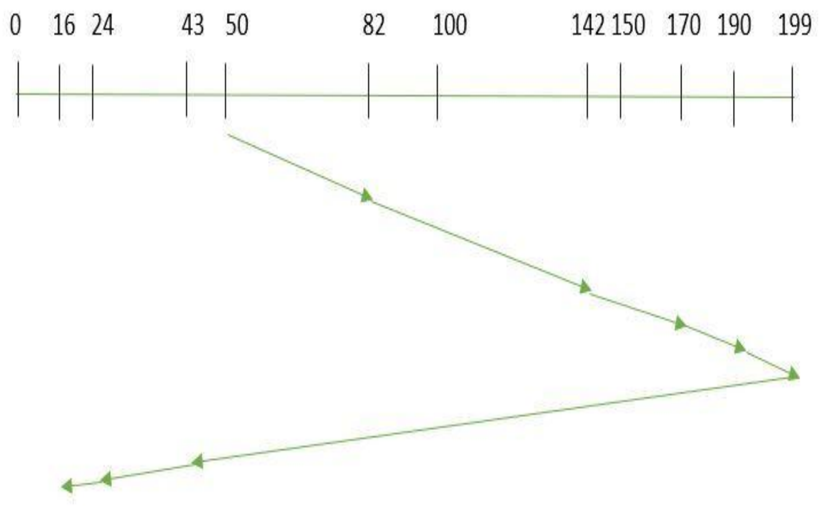
- Average Response Time decreases
- Throughput increases

Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests

3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as **elevator algorithm.** As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

   Example:
   Suppose the requests to be addressed are-82,170,43,140,24,16,190.
   And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value".**



Therefore, the seek time is calculated as:

=(199-50)+(199-16)

=332

Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

- Long waiting time for requests for locations just visited by disk arm

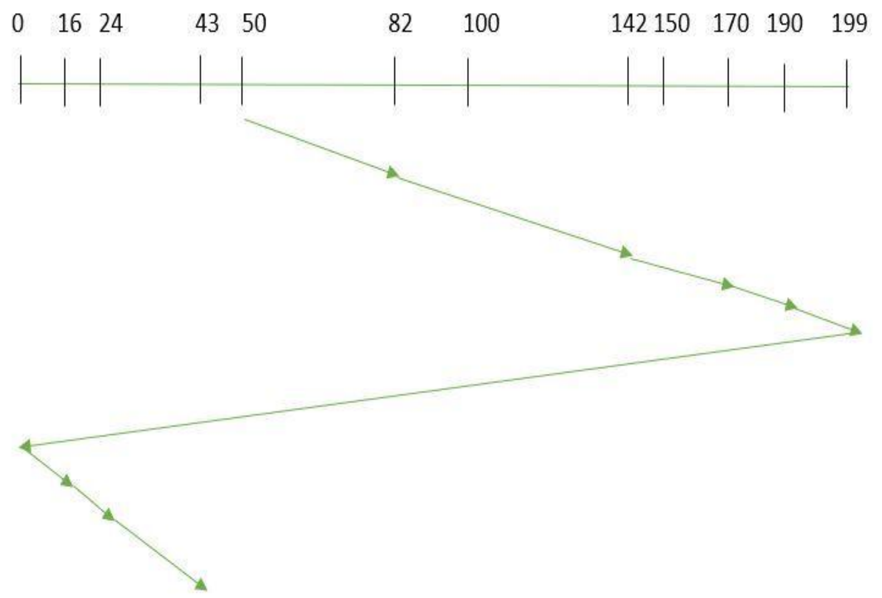
4. **CSCAN**: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that

too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in *CSCAN* algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value".**



Seek time is calculated as:

=(199-50)+(199-0)+(43-0)

=391

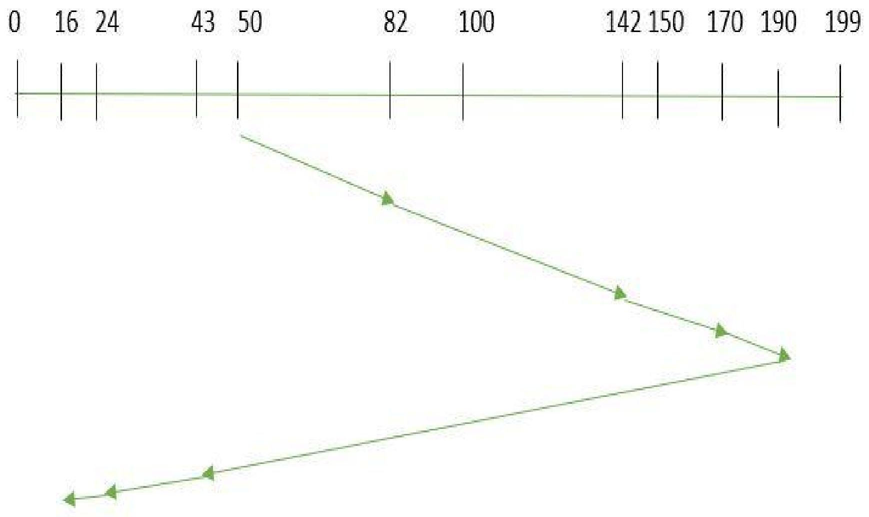
Advantages:

- Provides more uniform wait time compared to SCAN

5. **LOOK:** It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

   Example:

   Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value".**

   So, the seek time is calculated as:

   =(190-50)+(190-16)

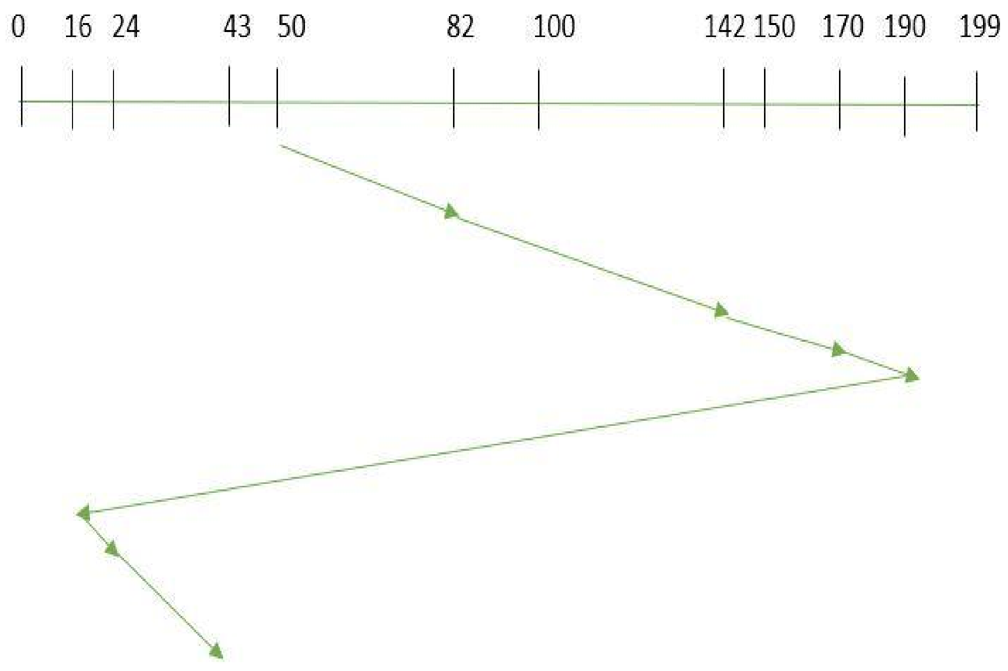
   =314

6. **CLOOK:** As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced

in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Example:

Suppose the requests to be addressed are-82,170,43,140,24,16,190. And the Read/Write arm is at 50, and it is also given that the disk arm should move **"towards the larger value"**



So, the seek time is calculated as:

=(190-50)+(190-16)+(43-16)

=341

7. **RSS**– It stands for random scheduling and just like its name it is nature. It is used in situations where scheduling involves random attributes such as random processing time, random due dates, random weights, and stochastic machine breakdowns this algorithm sits perfect. Which is why it is usually used for and analysis and simulation.
8. **LIFO**– In LIFO (Last In, First Out) algorithm, newest jobs are serviced before the existing ones i.e. in order of requests that get serviced the job that is newest or last entered is serviced first and then the rest in the same order.

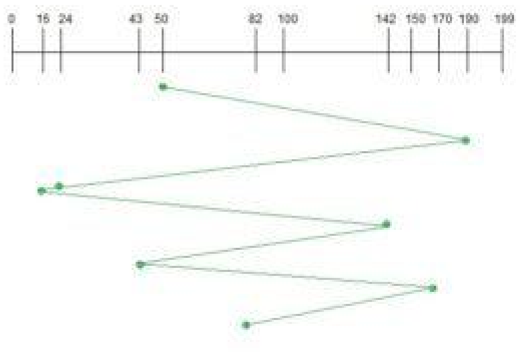
   Advantages

      ○ Maximizes locality and resource utilization
9. **Disadvantages**

      ○ Can seem a little unfair to other requests and if new requests keep coming in, it cause starvation to the old and existing ones.
10. **Example**

   Suppose the order of request is- (82,170,43,142,24,16,190)

   And current position of Read/Write head is : 50



11. **N-STEP SCAN** – It is also known as N-STEP LOOK algorithm. In this a buffer is created for N requests. All requests belonging to a buffer will be serviced in one go. Also once the buffer is full no new requests are kept in this buffer and are sent to another one. Now, when these N requests are serviced, the time comes for another top N requests and this way all get requests get a guaranteed service

    Advantages

    - It eliminates starvation of requests completely

12. **FSCAN**– This algorithm uses two sub-queues. During the scan all requests in the first queue are serviced and the new incoming requests are added to the second queue. All new requests are kept on halt until the existing requests in the first queue are serviced.

    Advantages

    - FSCAN along with N-Step-SCAN prevents “arm stickiness” (phenomena in I/O scheduling where the scheduling algorithm continues to service requests at or near the current sector and thus prevents any seeking)

Each algorithm is unique in its own way. Overall Performance depends on the number and type of requests.

Note: Average Rotational latency is generally taken as 1/2(Rotational latency).

Ans to the Question NO - 6(b)

Physical Address Space

Physical address space in a system can be defined as the size of the main memory. It is really important to compare the process size with the physical address space. The process size must be less than the physical address space.

Physical Address Space = Size of the Main Memory

If, physical address space = 64 KB = 2 ^ 6 KB = 2 ^ 6 X 2 ^ 10 Bytes = 2 ^ 16 bytes

Let us consider,

word size = 8 Bytes = 2 ^ 3 Bytes

Hence,

Physical address space (in words) = (2 ^ 16) / (2 ^ 3) = 2 ^ 13 Words

Therefore,

Physical Address = 13 bits

In General,

If, Physical Address Space = N Words

then, Physical Address = $\log_2 N$

Logical Address Space

Logical address space can be defined as the size of the process. The size of the process should be less enough so that it can reside in the main memory.

Let's say,

Logical Address Space = 128 MB = (2 ^ 7 X 2 ^ 20) Bytes = 2 ^ 27 Bytes

Word size = 4 Bytes = 2 ^ 2 Bytes

Logical Address Space (in words) = (2 ^ 27) / (2 ^ 2) = 2 ^ 25 Words

Logical Address = 25 Bits

In general,

If, logical address space = L words

Then, Logical Address = $\text{Log}_2 L$ bits

The **Word** is the smallest unit of the memory. It is the collection of bytes. Every operating system defines different word sizes after analyzing the n-bit address that is inputted to the decoder and the 2 ^ n memory locations that are produced from the decoder.

Page Table

Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.

Logical addresses are generated by the CPU for the pages of the processes therefore they are generally used by the processes.

Physical addresses are the actual frame address of the memory. They are generally used by the hardware or more specifically by RAM subsystems.

The image given below considers,

Physical Address Space = M words

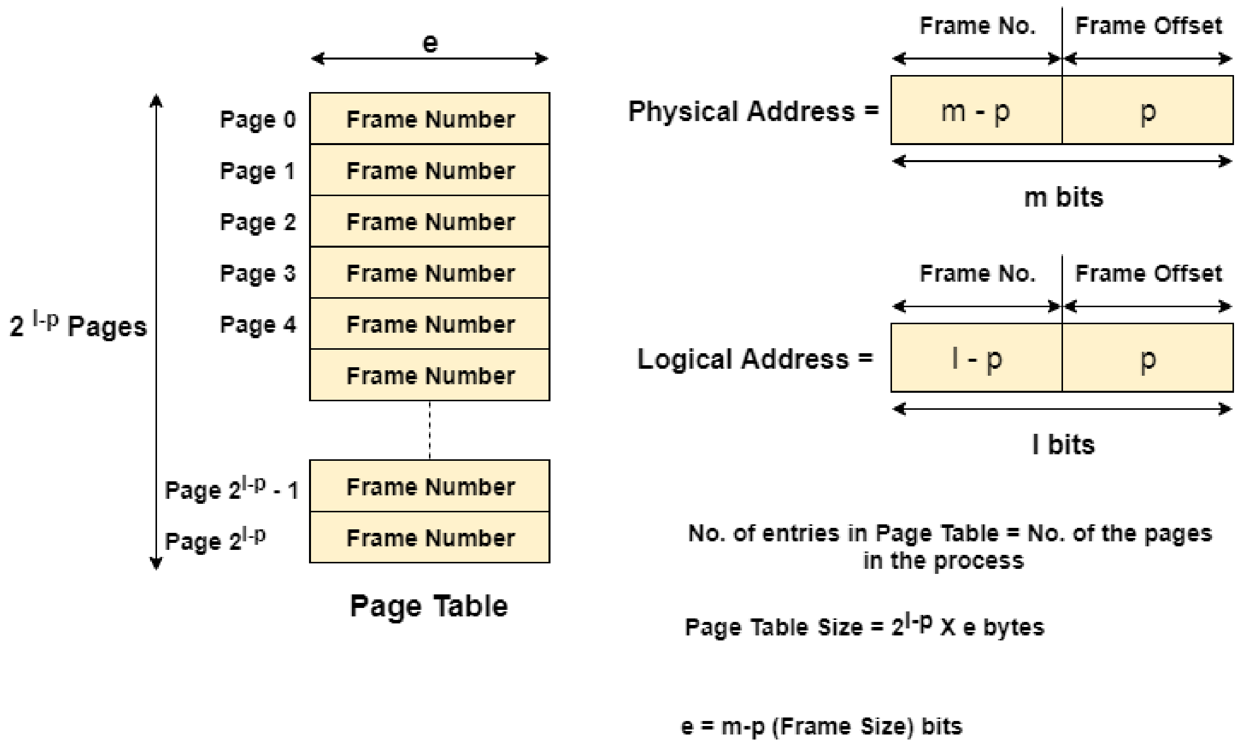
Logical Address Space = L words

Page Size = P words

Physical Address = $\log_2 M$ = m bits

Logical Address = $\log_2 L$ = l bits

page offset = $\log_2 P$ = p bits



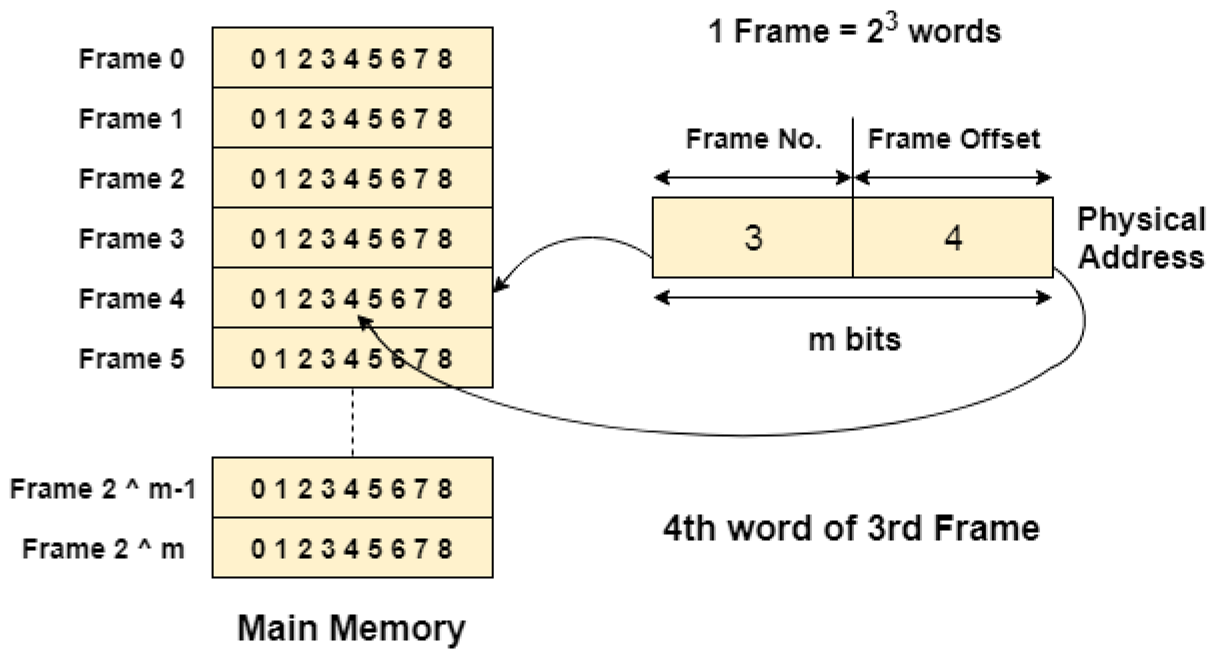
The CPU always accesses the processes through their logical addresses. However, the main memory recognizes physical address only.

In this situation, a unit named as Memory Management Unit comes into the picture. It converts the page number of the logical address to the frame number of the physical address. The offset remains same in both the addresses.

To perform this task, Memory Management unit needs a special kind of mapping which is done by page table. The page table stores all the Frame numbers corresponding to the page numbers of the page table.

In other words, the page table maps the page number to its actual location (frame number) in the memory.

In the image given below shows, how the required word of the frame is accessed with the help of offset.



Ans to the Question NO - 7(a)

Input/Output devices are the devices that are responsible for the input/output operations in a computer system.

Basically there are following two types of input/output devices:

- Block devices
- Character devices

Block Devices

A block device stores information in block with fixed-size and own-address.

It is possible to read/write each and every block independently in case of block device.

In case of disk, it is always possible to seek another cylinder and then wait for required block to rotate under head without mattering where the arm currently is. Therefore, disk is a block addressable device.

Character Devices

A character device accepts/delivers a stream of characters without regarding to any block structure.

Character device isn't addressable.

Character device doesn't have any seek operation.

There are too many character devices present in a computer system such as printer, mice, rats, network interfaces etc. These four are the common character devices.

The input/output units consists of the following two parts:

- Mechanical component
- Electronic component (device controller)

Basically, device controllers are the electronic components of input/output units.

Device controller can also called as adapter.

The mechanical component of the input/output unit is the device itself whereas the electronic component or device controller or adapter takes the form of a printed circuit card that can be inserted into an expansion slot.

Usually, the controller card has a connector on it, just to help in plugging the cable leading to the device controllers can handle many identical devices.

<u>Ans to the Question NO - 7(b)</u>

Direct Memory Access (DMA) transfers the block of data between the *memory* and *peripheral devices* of the system, **without the participation** of the **processor**. The unit that controls the activity of accessing memory directly is called a **DMA controller**.

Direct memory access (DMA) is a **mode of data transfer** between the memory and I/O devices. This happens **without the involvement** of the processor. We have two other methods of data transfer, **programmed I/O** and **Interrupt driven I/O**. Let's revise each and get acknowledge with their drawbacks.

In **programmed I/O**, the processor keeps on scanning whether any device is ready for data transfer. If an I/O device is ready, the processor **fully dedicates** itself in transferring the data between I/O and memory. It transfers data at a **high rate, but it can't get involved in any other activity** during data transfer. This is the major **drawback** of programmed I/O.

In **Interrupt driven I/O**, whenever the device is ready for data transfer, then it raises an **interrupt to processor**. Processor completes executing its ongoing instruction and saves its current state. It then switches to data transfer which causes a **delay**. Here, the processor doesn't keep scanning for peripherals ready for data transfer. But, it is **fully involved** in the data transfer process. So, it is also not an effective way of data transfer.

The above two modes of data transfer are not useful for transferring a large block of data. But, the DMA controller completes this task at a faster rate and is also effective for transfer of large data block.

The DMA controller transfers the data in three modes:

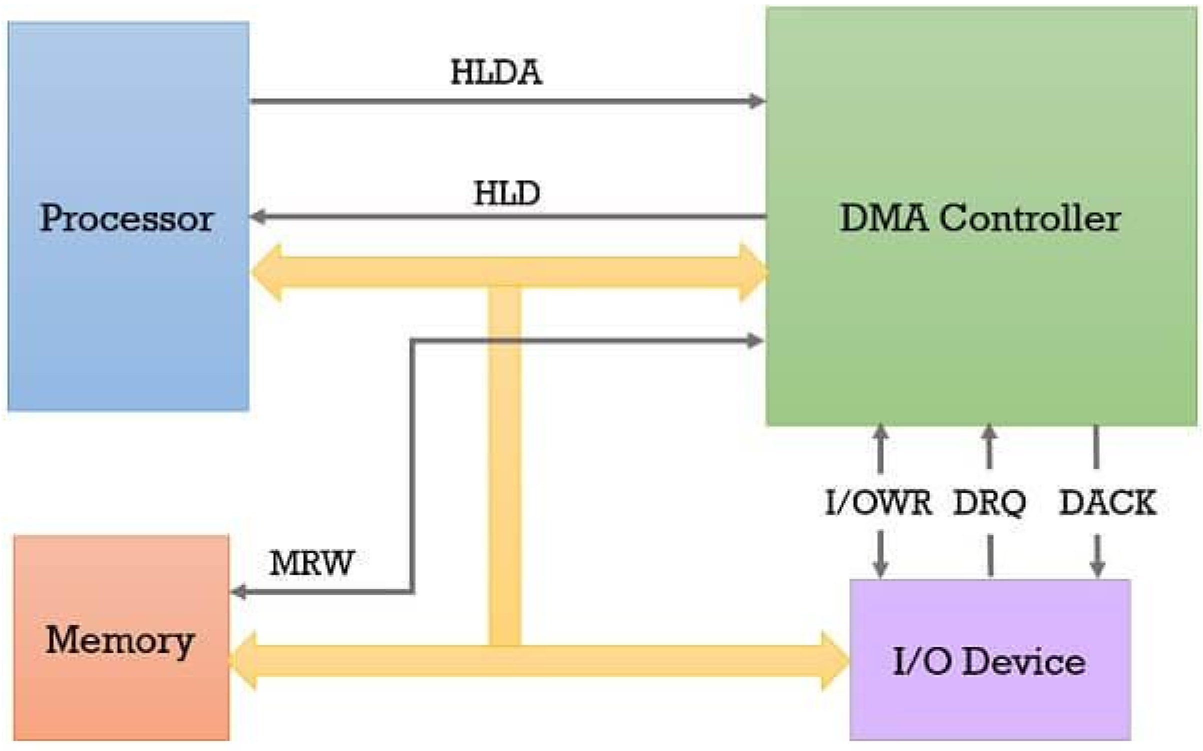
1. **Burst Mode:** Here, once the DMA controller gains the charge of the system bus, then it releases the system bus only after **completion** of data transfer. Till then the CPU has to wait for the system buses.
2. **Cycle Stealing Mode:** In this mode, the DMA controller **forces** the CPU to stop its operation and **relinquish the control over the bus** for a

short term to DMA controller. After the **transfer of every byte**, the DMA controller **releases** the **bus** and then again requests for the system bus. In this way, the DMA controller steals the clock cycle for transferring every byte.

3. **Transparent Mode:** Here, the DMA controller takes the charge of system bus only if the **processor does not require the system bus**.

Direct Memory Access Controller & it's Working

DMA controller is a **hardware unit** that allows I/O devices to access memory directly without the participation of the processor. Here, we will discuss the working of the DMA controller. Below we have the diagram of DMA controller that explains its working:



DMA Controller Data Transfer

1. Whenever an I/O device wants to transfer the data to or from memory, it sends the DMA request (**DRQ**) to the DMA controller. DMA controller

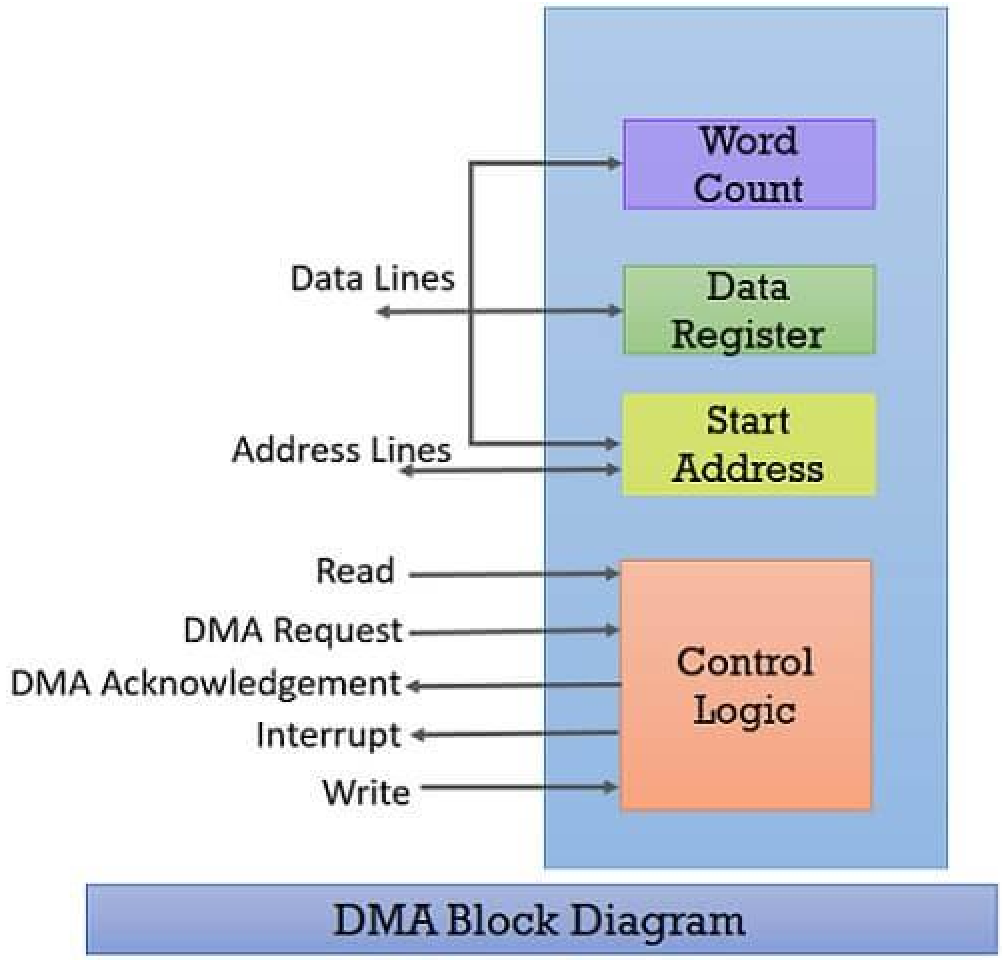
accepts this DRQ and asks the CPU to hold for a few clock cycles by sending it the Hold request (**HLD**).

2. CPU receives the Hold request (HLD) from DMA controller and relinquishes the bus and sends the Hold acknowledgement (**HLDA**) to DMA controller.
3. After receiving the Hold acknowledgement (HLDA), DMA controller acknowledges I/O device **(DACK)** that the data transfer can be performed and DMA controller takes the charge of the system bus and transfers the data to or from memory.
4. When the data transfer is accomplished, the DMA raise an **interrupt** to let know the processor that the task of data transfer is finished and the processor can take control over the bus again and start processing where it has left.

Now the DMA controller can be a separate unit that is shared by various I/O devices, or it can also be a part of the I/O device interface.

Direct Memory Access Diagram

After exploring the working of DMA controller, let us discuss the block diagram of the DMA controller. Below we have a block diagram of DMA controller.



Whenever a processor is requested to read or write a block of data, i.e. transfer a block of data, it instructs the DMA controller by sending the following information.

1. The first information is whether the data has to be read from memory or the data has to be written to the memory. It passes this information via **read or write control lines** that is between the processor and DMA controllers **control logic unit**.
2. The processor also provides the **starting address** of/ for the data block in the memory, from where the data block in memory has to be read or where the data block has to be written in memory. DMA controller stores this in its **address register**. It is also called the **starting address register**.

3. The processor also sends the **word count**, i.e. how many words are to be read or written. It stores this information in the **data count** or the **word count** register.
4. The most important is the **address of I/O device** that wants to read or write data. This information is stored in the **data register.**

Direct Memory Access Advantages and Disadvantages

Advantages:

1. Transferring the data without the involvement of the processor will **speed up** the read-write task.
2. DMA **reduces the clock cycle** requires to read or write a block of data.
3. Implementing DMA also **reduces the overhead** of the processor.

Disadvantages

1. As it is a hardware unit, it would **cost** to implement a DMA controller in the system.
2. Cache **coherence** problem can occur while using DMA controller.

Ans to the Question NO - 7(c)

Interrupts are signals sent to the CPU by external devices, normally I/O devices. They tell the CPU to stop its current activities and execute the appropriate part of the operating system.

There are three types of interrupts:

1. **Hardware Interupts** are generated by hardware devices to signal that they need some attention from the OS. They may have just received some data (e.g., keystrokes on the keyboard or an data on the ethernet card); or they have just completed a task which the

operating system previous requested, such as transfering data between the hard drive and memory.
2. **Software Interupts** are generated by programs when they want to request a system call to be performed by the operating system.
3. **Traps** are generated by the CPU itself to indicate that some error or condition occured for which assistance from the operating system is needed.

Interrupts are important because they give the user better control over the computer. Without interrupts, a user may have to wait for a given application to have a higher priority over the CPU to be ran. This ensures that the CPU will deal with the process immediately.

Ans to the Question NO - 8(a)

A **file** is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes. It is a method of data collection that is used as a medium for giving input and receiving output from that program.

In general, a **file** is a sequence of bits, bytes, or records whose meaning is defined by the file creator and user. Every File has a logical location where they are located for storage and retrieval.

Properties of a File System

Here, are important properties of a file system:

- Files are stored on disk or other storage and do not disappear when a user logs off.
- Files have names and are associated with access permission that permits controlled sharing.
- Files could be arranged or more complex structures to reflect the relationship between them.

Objective of File management System

Here are the main objectives of the file management system:

- It provides I/O support for a variety of storage device types.
- Minimizes the chances of lost or destroyed data
- Helps OS to standardized I/O interface routines for user processes.
- It provides I/O support for multiple users in a multiuser systems environment.

Ans to the Question NO - 8(b)

File structure

A File Structure needs to be predefined format in such a way that an operating system understands . It has an exclusively defined structure, which is based on its type.

Three types of files structure in OS:

- A text file: It is a series of characters that is organized in lines.
- An object file: It is a series of bytes that is organized into blocks.
- A source file: It is a series of functions and processes.

File Attributes

A file has a name and data. Moreover, it also stores meta information like file creation date and time, current size, last modified date, etc. All this information is called the attributes of a file system.

Here, are some important File attributes used in OS:

- **Name:** It is the only information stored in a human-readable form.

- **Identifier**: Every file is identified by a unique tag number within a file system known as an identifier.
- **Location:** Points to file location on device.
- **Type:** This attribute is required for systems that support various types of files.
- **Size**. Attribute used to display the current file size.
- **Protection**. This attribute assigns and controls the access rights of reading, writing, and executing the file.
- **Time, date and security:** It is used for protection, security, and also used for monitoring

File Type

It refers to the ability of the operating system to differentiate various types of files like text files, binary, and source files. However, Operating systems like MS_DOS and UNIX has the following type of files:

Character Special File

It is a hardware file that reads or writes data character by character, like mouse, printer, and more.

Ordinary files

- These types of files stores user information.
- It may be text, executable programs, and databases.
- It allows the user to perform operations like add, delete, and modify.

Directory Files

- Directory contains files and other related information about those files. Its basically a folder to hold and organize multiple files.

Special Files

- These files are also called device files. It represents physical devices like printers, disks, networks, flash drive, etc.

Functions of File

- Create file, find space on disk, and make an entry in the directory.
- Write to file, requires positioning within the file
- Read from file involves positioning within the file
- Delete directory entry, regain disk space.
- Reposition: move read/write position.

Ans to the Question NO - 8(c)

File Access Methods

File access is a process that determines the way that files are accessed and read into memory. Generally, a single access method is always supported by operating systems. Though there are some operating system which also supports multiple access methods.

Three file access methods are:

- Sequential access
- Direct random access
- Index sequential access

Sequential Access

In this type of file access method, records are accessed in a certain pre-defined sequence. In the sequential access method, information stored in the file is also processed one by one. Most compilers access files using this access method.

Random Access

The random access method is also called direct random access. This method allow accessing the record directly. Each record has its own address on which can be directly accessed for reading and writing.

Sequential Access

This type of accessing method is based on simple sequential access. In this access method, an index is built for every file, with a direct pointer to different memory blocks. In this method, the Index is searched sequentially, and its pointer can access the file directly. Multiple levels of indexing can be used to offer greater efficiency in access. It also reduces the time needed to access a single record.

Three types of space allocation methods are:

- Linked Allocation
- Indexed Allocation
- Contiguous Allocation

Contiguous Allocation

In this method,

- Every file users a contiguous address space on memory.
- Here, the OS assigns disk address is in linear order.
- In the contiguous allocation method, external fragmentation is the biggest issue.

Linked Allocation

In this method,

- Every file includes a list of links.
- The directory contains a link or pointer in the first block of a file.
- With this method, there is no external fragmentation
- This File allocation method is used for sequential access files.
- This method is not ideal for a direct access file.

Indexed Allocation

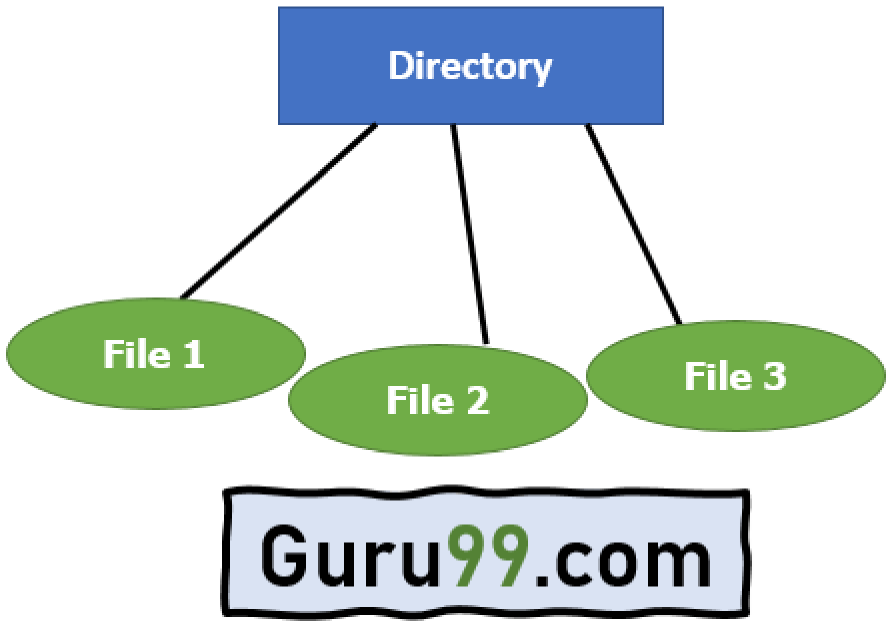
In this method,

- Directory comprises the addresses of index blocks of the specific files.
- An index block is created, having all the pointers for specific files.
- All files should have individual index blocks to store the addresses for disk space.

Ans to the Question NO - 8(d)

File Directories

A single directory may or may not contain multiple files. It can also have sub-directories inside the main directory. Information about files is maintained by Directories. In Windows OS, it is called folders.



Single Level Directory

Following is the information which is maintained in a directory:

- **Name** The name which is displayed to the user.
- **Type**: Type of the directory.
- **Position**: Current next-read/write pointers.
- **Location**: Location on the device where the file header is stored.
- **Size** : Number of bytes, block, and words in the file.
- **Protection**: Access control on read/write/execute/delete.
- **Usage**: Time of creation, access, modification

File types- name, extension

File Type	Usual extension	Function
Executable	exe, com, bin or none	ready-to-run machine- language program
Object	obj, o	complied, machine language, not linked
Source code	c. p, pas, 177, asm, a	source code in various languages
Batch	bat, sh	Series of commands to be executed
Text	txt, doc	textual data documents
Word processor	doc,docs, tex, rrf, etc.	various word-processor formats
Library	lib, h	libraries of routines
Archive	arc, zip, tar	related files grouped into one file, sometimes compressed.

The End