OPERATING SYSTEMS

ASSIGNMENT-3

NAME: SUBHAPREET PATRO
ROLL NO.: 2211CS010547
GROUP: 7A

Q1) Explain any 5 CPU Scheduling Algorithms with example

Ans.

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.

Some of the popular Scheduling Algorithms are:
1) First Come First Served (FCFS) Scheduling

2) Shortest Job Next (SJN) Scheduling

3) Priority Scheduling

4) Shortest Remaining Time

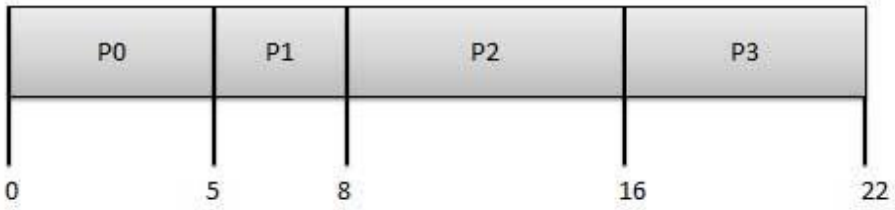
5) Round Robin(RR) Scheduling

These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows –

Process	Wait Time: Service Time – Arrival Time
P0	0-0=0
P1	5-1=4
P2	8-2=6
P3	16-3=13

Average Wait Time: (0+4+6+13) / 4 = 5.75

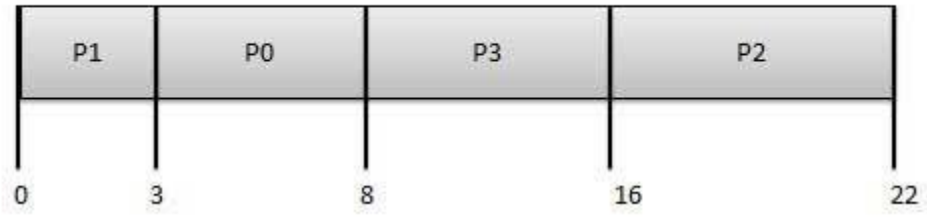
Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Process	Arrival Time	Execute Time	Service Time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8



Waiting time of each process is as follows –

Process	Waiting Time
P0	0-0=0
P1	5-1=4
P2	14-2=12
P3	8-3=5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

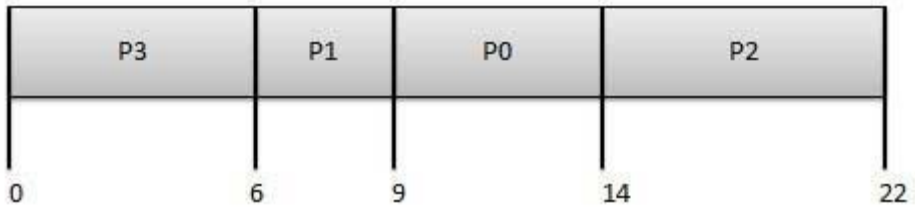
Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Waiting time of each process is as follows –

Process	Waiting Time
P0	0-0=0
P1	11-1=10
P2	14-2=12
P3	5-3=2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Shortest Remaining Time

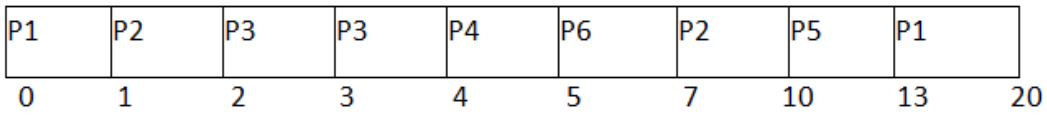
- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.

- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Example

In this Example, there are five jobs P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table.

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time
1	0	8	20	20	12	0
2	1	4	10	9	5	1
3	2	2	4	2	0	2
4	3	1	5	2	1	4
5	4	3	13	9	6	10
6	5	2	7	2	0	5

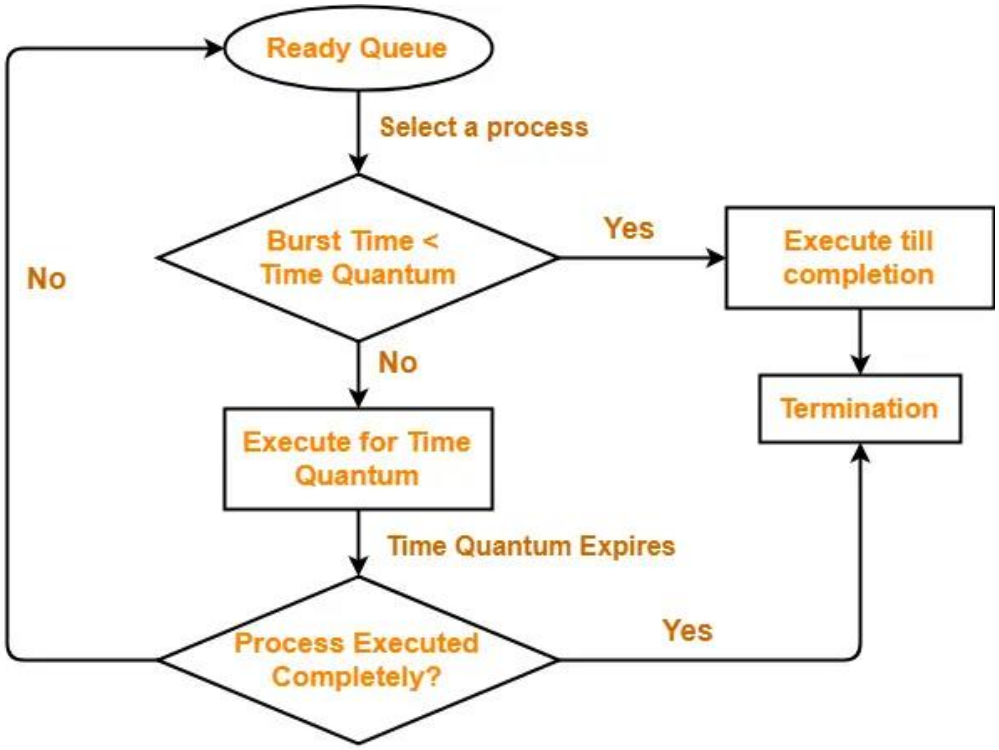


Average Waiting Time = 24/6

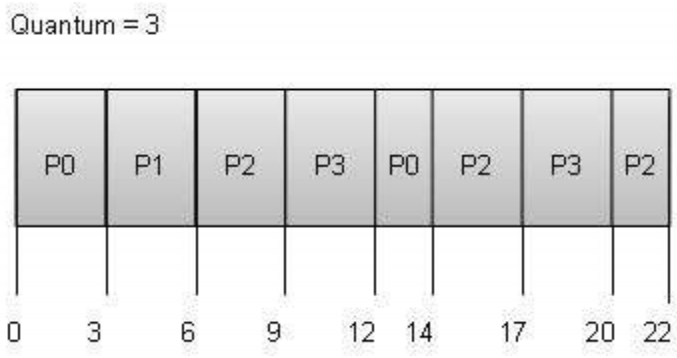
Round Robin Scheduling

- Round Robin is the pre-emptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.

- Context switching is used to save states of preempted processes.



Round Robin Scheduling



Wait time of each process is as follows –

Process	Wait Time: Service Time – Arrival Time
P0	(0-0)+(12-3)=9
P1	(3-1)=2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Q2) Explain (a) Paging (b) Segmentation (c) Fragmentation (d) Swapping

Ans.

Paging

Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. Additionally, frames will be used to split the main memory. This scheme permits the physical address space of a process to be non – contiguous.

In paging, the physical memory is divided into fixed-size blocks called page frames, which are the same size as the pages used by the process. The process's logical address space is also divided into fixed-size blocks called pages, which are the same size as the page frames. When a process requests memory, the operating system allocates one or more page frames to the process and maps the process's logical pages to the physical page frames.

The mapping between logical pages and physical page frames is maintained by the page table, which is used by the memory management unit to translate logical addresses into physical addresses. The page table maps each logical page number to a physical page frame number.

Example:

- If Logical Address = 31 bit, then Logical Address Space = 231 words = 2 G words (1 G = 230)
- If Logical Address Space = 128 M words = 27 * 220 words, then Logical Address = log2 227 = 27 bits
- If Physical Address = 22 bit, then Physical Address Space = 222 words = 4 M words (1 M = 220)
- If Physical Address Space = 16 M words = 24 * 220 words, then Physical Address = log2 224 = 24 bits

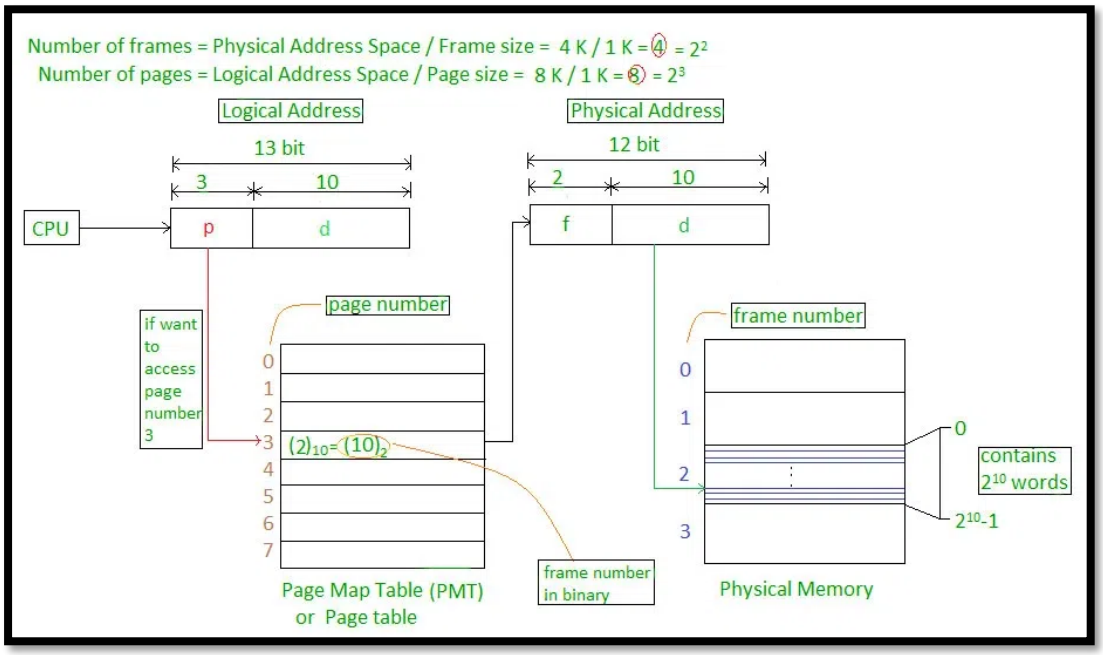
The mapping from virtual to physical address is done by the Memory Management Unit (MMU) which is a hardware device and this mapping is known as the paging technique.

- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called frames.

- The Logical Address Space is also split into fixed-size blocks, called pages.
- Page Size = Frame Size

Let us consider an example:

- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)



The address generated by the CPU is divided into:

- Page number(p): Number of bits required to represent the pages in Logical Address Space or Page number
- Page offset(d): Number of bits required to represent a particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Segmentation

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user's view of the process which paging does not provide. Here the user's view is mapped to physical memory.

Types of Segmentation in Operating System:

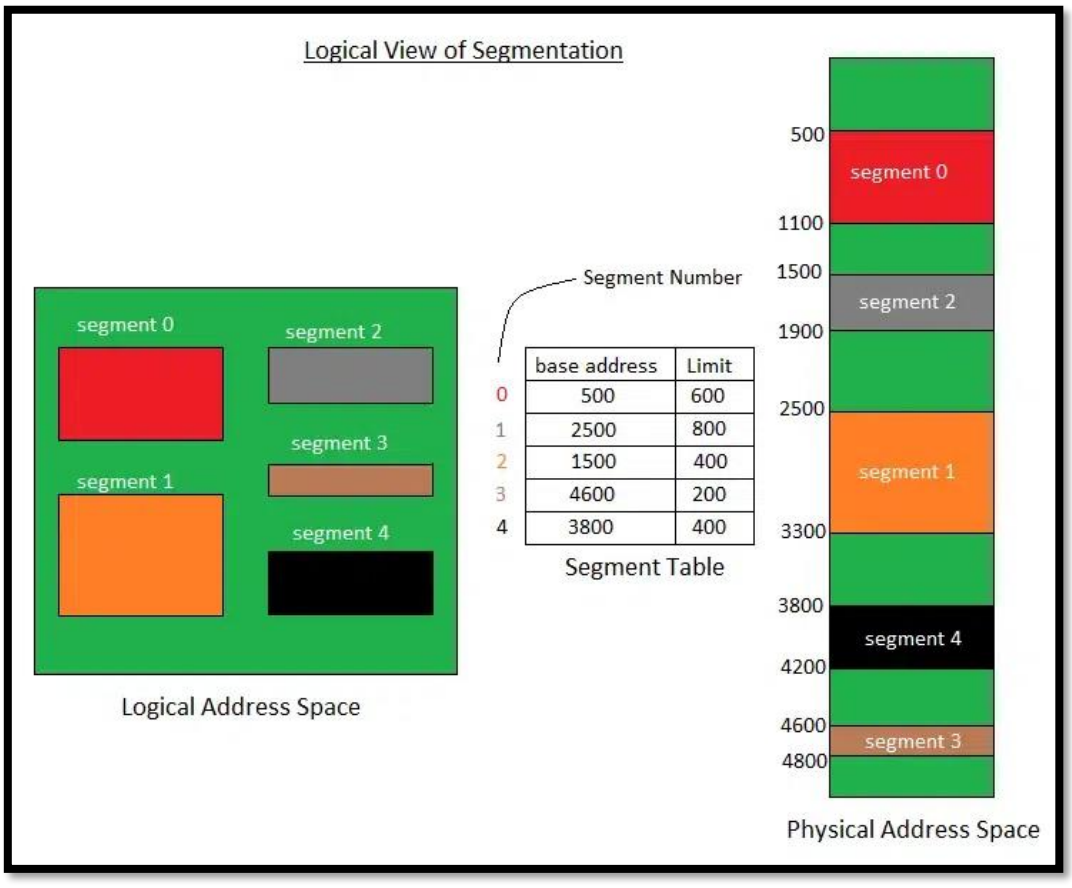
- **Virtual Memory Segmentation**: Each process is divided into a number of segments, but the segmentation is not done all at once. This segmentation may or may not take place at the run time of the program.
- **Simple Segmentation**: Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called Segment Table.

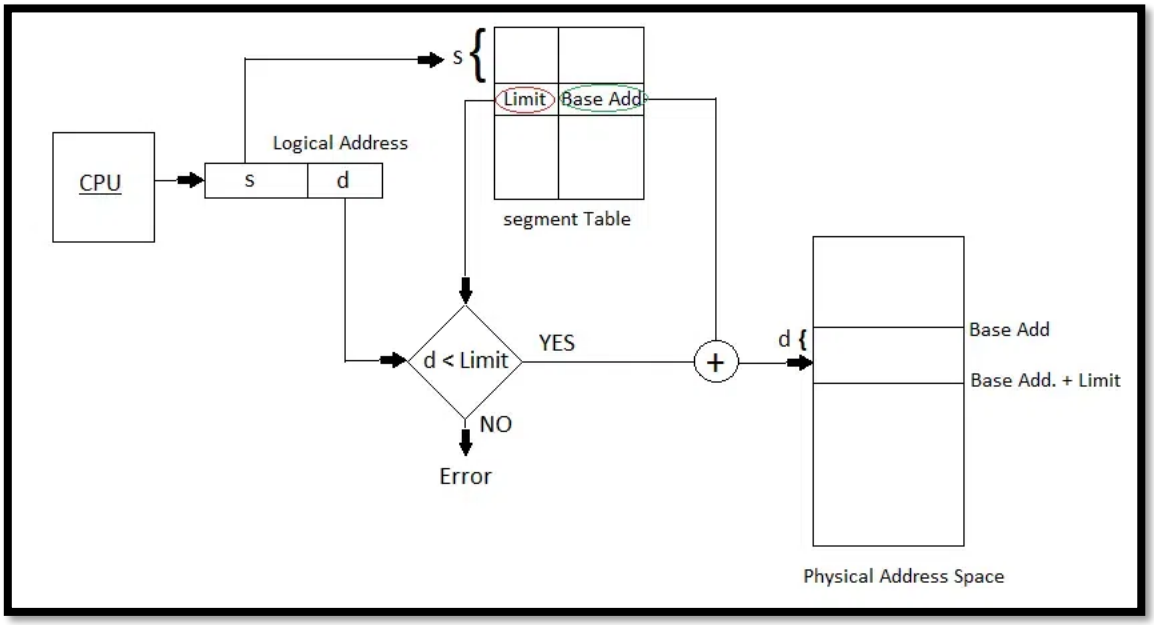
What is Segment Table?

It maps a two-dimensional Logical address into a one-dimensional Physical address. It's each table entry has:

- Base Address: It contains the starting physical address where the segments reside in memory.
- Segment Limit: Also known as segment offset. It specifies the length of the segment.



Translation of Two-dimensional Logical Address to Dimensional Physical Address.



The address generated by the CPU is divided into:

- **Segment number (s):** Number of bits required to represent the segment.
- **Segment offset (d):** Number of bits required to represent the size of the segment.

Fragmentation

The process of dividing a computer file, such as a data file or an executable program file, into fragments that are stored in different parts of a computer's storage medium, such as its hard disc or RAM, is known as fragmentation in computing. When a file is fragmented, it is stored on the storage medium in non-contiguous blocks, which means that the blocks are not stored next to each other.

What is Fragmentation in an Operating System?

An unwanted problem with operating systems is fragmentation, which occurs when processes load and unload from memory and divide available memory. Because memory blocks are so small, they cannot be assigned to processes, and thus remain idle. It's also important to realize that programs create free space or holes in memory when they are loaded and unloaded. Because additional processes cannot be assigned to these little pieces, memory is used inefficiently.

The memory allocation scheme determines the fragmentation circumstances. These regions of memory become fragmented when the process loads and unloads from it, making it unusable for incoming processes. We refer to it as fragmentation.

Cause of Fragmentation

This can happen when a file is too large to fit into a single contiguous block of free space on the storage medium, or when the blocks of free space on the medium are insufficient to hold the file. Because the system must search for and retrieve individual fragments from different locations in order to open the file, fragmentation can cause problems when reading or accessing the file.

Effect of Fragmentation

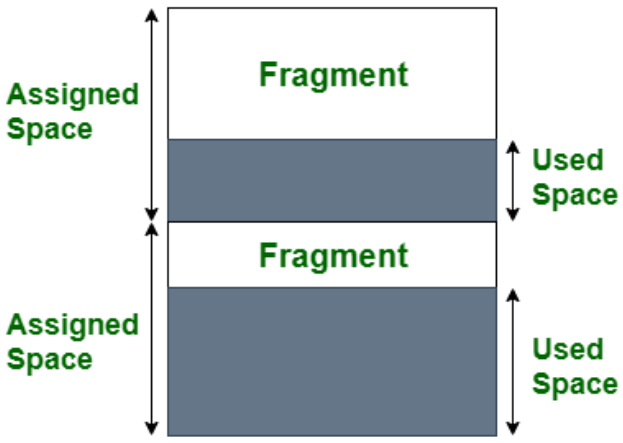
This can reduce system performance and make it more difficult to access the file. It is generally best to defragment your hard disc on a regular basis to avoid fragmentation, which is a process that rearranges the blocks of data on the disc so that files are stored in contiguous blocks and can be accessed more quickly.

Types of Fragmentation

There are two main types of fragmentation:

1. Internal fragmentation

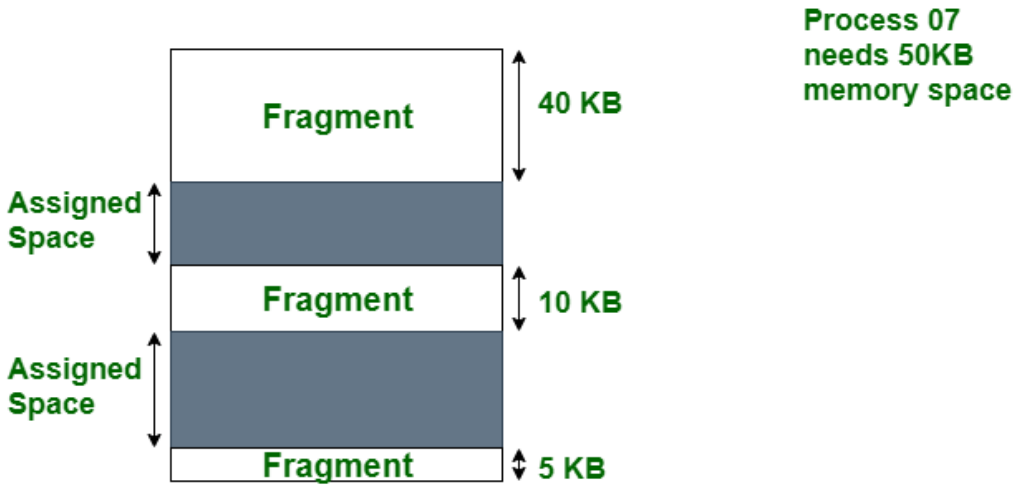
Internal fragmentation occurs when there is unused space within a memory block. For example, if a system allocates a 64KB block of memory to store a file that is only 40KB in size, that block will contain 24KB of internal fragmentation. When the system employs a fixed-size block allocation method, such as a memory allocator with a fixed block size, this can occur.



Internal Fragmentation

2. External fragmentation

External fragmentation occurs when a storage medium, such as a hard disc or solid-state drive, has many small blocks of free space scattered throughout it. This can happen when a system creates and deletes files frequently, leaving many small blocks of free space on the medium. When a system needs to store a new file, it may be unable to find a single contiguous block of free space large enough to store the file and must instead store the file in multiple smaller blocks. This can cause external fragmentation and performance problems when accessing the file.

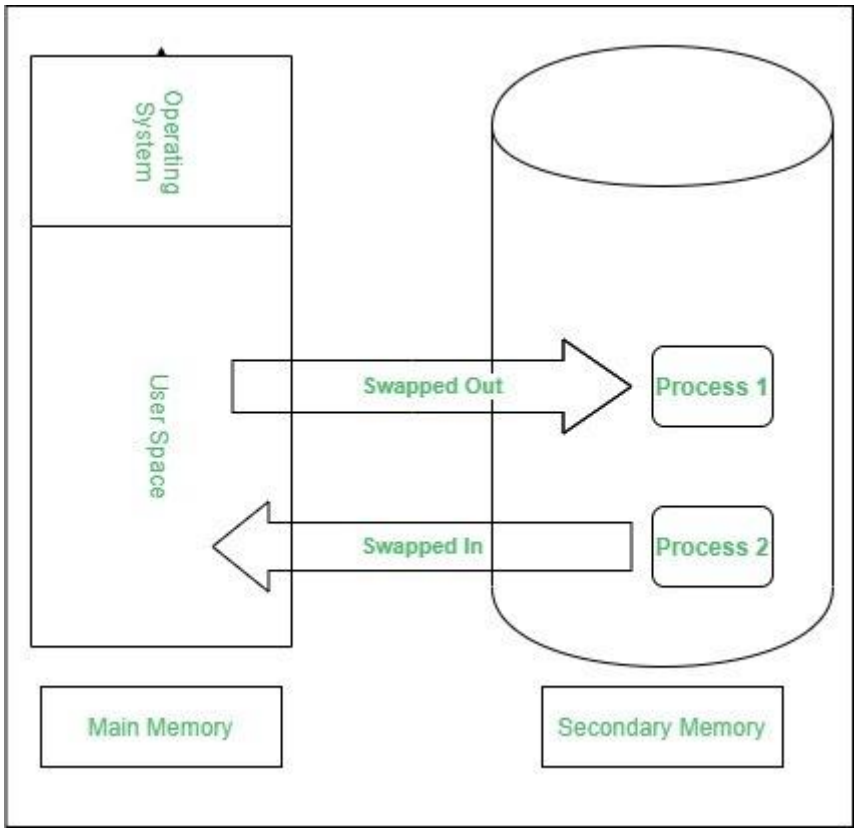


Fragmentation can also occur at various levels within a system. File fragmentation, for example, can occur at the file system level, in which a file is divided into multiple non-contiguous blocks and stored on a storage medium. Memory fragmentation can occur at the memory management level, where the system allocates and deallocated memory blocks dynamically. Network fragmentation occurs when a packet of data is divided into smaller fragments for transmission over a network.

Swapping

To increase CPU utilization in multiprogramming, a memory management scheme known as swapping can be used. Swapping is the process of bringing a process into memory and then temporarily copying it to the disc after it has run for a while. The purpose of swapping in an operating system is to access data on a hard disc and move it to RAM so that application programs can use it. It's important to remember that swapping is only used when data isn't available in RAM. Although the swapping process degrades system performance, it allows larger and multiple processes to run concurrently. Because of this, swapping is also known as memory compaction. The CPU scheduler determines which processes are swapped in and which are swapped out. Consider a multiprogramming environment that employs a priority-

based scheduling algorithm. When a high-priority process enters the input queue, a low-priority process is swapped out so the high-priority process can be loaded and executed. When this process terminates, the low priority process is swapped back into memory to continue its execution. Below figure shows the swapping process in operating system:



Swapping has been subdivided into two concepts: **swap-in** and **swap-out**.

- Swap-out is a technique for moving a process from RAM to the hard disc.
- Swap-in is a method of transferring a program from a hard disc to main memory, or RAM.

Advantages

- If there is low main memory so some processes may has to wait for much long but by using swapping process do not have to wait long for execution on CPU.
- It utilize the main memory.
- Using only single main memory, multiple process can be run by CPU using swap partition.
- The concept of virtual memory start from here and it utilize it in better way.

- This concept can be useful in priority based scheduling to optimize the swapping process.

Disadvantages

- If there is low main memory resource and user is executing too many processes and suddenly the power of system goes off there might be a scenario where data get erase of the processes which are took parts in swapping.
- Chances of number of page faults occur
- Low processing performance

Q3) Explain about Logical and Physical Address space with a diagram.

Ans.

A logical address is generated by the CPU while a program is running. The logical address is a virtual address as it does not exist physically, therefore, it is also known as a Virtual Address. The physical address describes the precise position of necessary data in a memory. Before they are used, the MMU must map the logical address to the physical address. In operating systems, logical and physical addresses are used to manage and access memory. Here is an overview of each in detail.

What is a Logical Address?

A logical address, also known as a virtual address, is an address generated by the CPU during program execution. It is the address seen by the process and is relative to the program's address space. The process accesses memory using logical addresses, which are translated by the operating system into physical addresses. An address that is created by the CPU while a program is running is known as a logical address. Because the logical address is virtual—that is, it doesn't exist physically—it is also referred to as such. The CPU uses this address as a reference to go to the actual memory location. All logical addresses created from a program's perspective are referred to as being in the "logical address space". This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective.

What is a Physical Address?

A physical address is the actual address in the main memory where data is stored. It is a location in physical memory, as opposed to a virtual address. Physical addresses are used by the Memory Management Unit (MMU) to translate logical addresses into physical addresses. The user must use the corresponding logical address to go to the physical address rather than

directly accessing the physical address. For a computer program to function, physical memory space is required. Therefore, the logical address and physical address need to be mapped before the program is run.

The term “physical address” describes the precise position of necessary data in a memory. Before they are used, the MMU must map the logical address to the physical address. This is because the user program creates the logical address and believes that the program is operating in this logical address. However, the program requires physical memory to execute. All physical addresses that match the logical addresses in a logical address space are collectively referred to as the “physical address space”

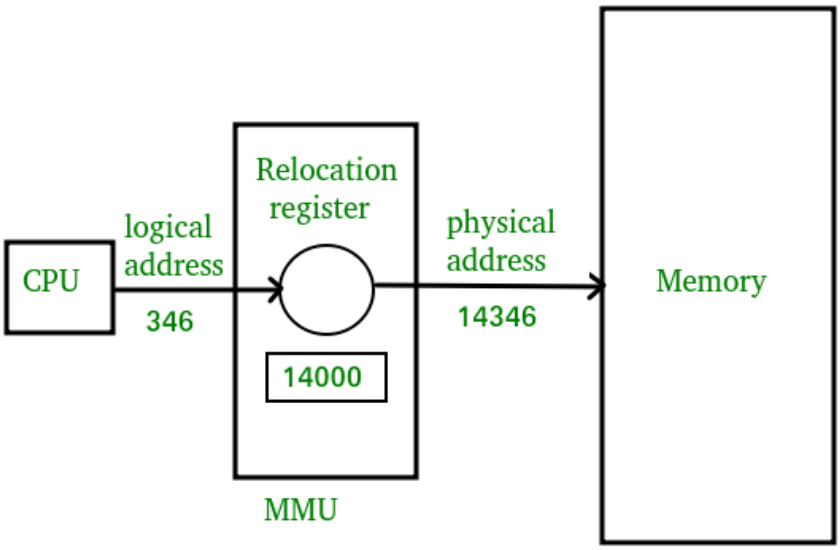
The translation from logical to physical addresses is performed by the operating system’s memory management unit. The MMU uses a page table to translate logical addresses into physical addresses. The page table maps each logical page number to a physical frame number.

Similarities Between Logical and Physical Addresses in the Operating System

- Both logical and physical addresses are used to identify a specific location in memory.
- Both logical and physical addresses can be represented in different formats, such as binary, hexadecimal, or decimal.
- Both logical and physical addresses have a finite range, which is determined by the number of bits used to represent them.

Important Points about Logical and Physical Addresses in Operating Systems

- The use of logical addresses provides a layer of abstraction that allows processes to access memory without knowing the physical memory location.
- Logical addresses are mapped to physical addresses using a page table. The page table contains information about the mapping between logical and physical addresses.
- The MMU translates logical addresses into physical addresses using the page table. This translation is transparent to the process and is performed by hardware.
- The use of logical and physical addresses allows the operating system to manage memory more efficiently by using techniques such as paging and segmentation.



What is Memory Management Unit?

The physical hardware of a computer that manages its virtual memory and caching functions is called the memory management unit (MMU). The MMU is sometimes housed in a separate Integrated Chip (IC), but it is typically found inside the central processing unit (CPU) of the computer. The MMU receives all inputs for data requests and decides whether to retrieve the data from ROM or RAM storage.

Difference Between Logical address and Physical Address

Parameter	LOGICAL ADDRESS	PHYSICAL ADDRESS
Basic	generated by CPU	location in a memory unit
Address Space	Logical Address Space is set of all logical addresses generated by CPU in reference to a program.	Physical Address is set of all physical addresses mapped to the corresponding logical addresses.
Visibility	User can view the logical address of a program.	User can never view physical address of program.
Generation	generated by the CPU	Computed by MMU
Access	The user can use the logical address to access the physical address.	The user can indirectly access physical address but not directly.
Editable	Logical address can be change.	Physical address will not change.
Also called	virtual address.	real address.