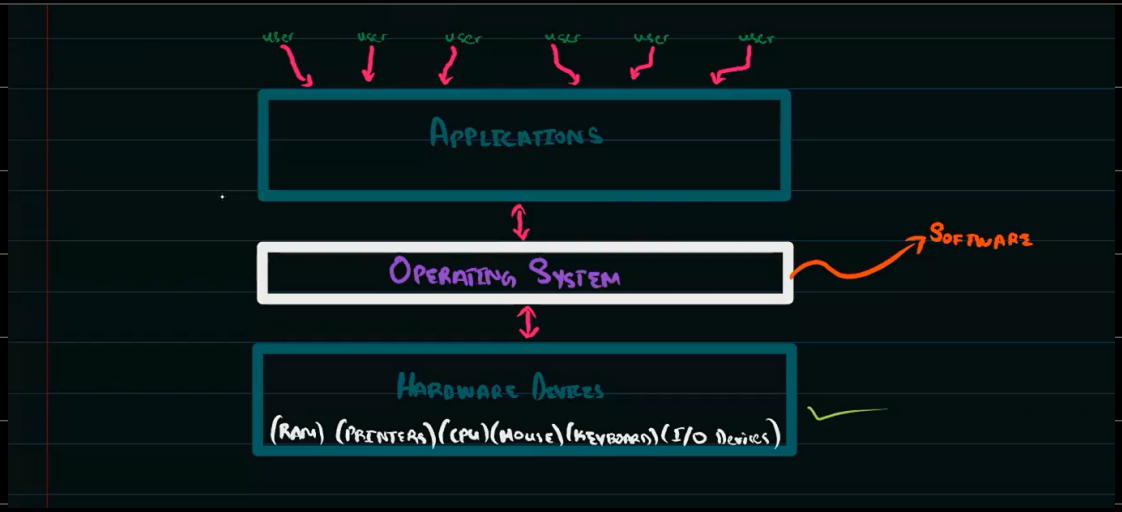
Purpose of an Operating System

Q- What is an operating system?

- Provides interface between users and Hardware



Tasks Performed By An Operating System

Resource Management

Q- What are resources?

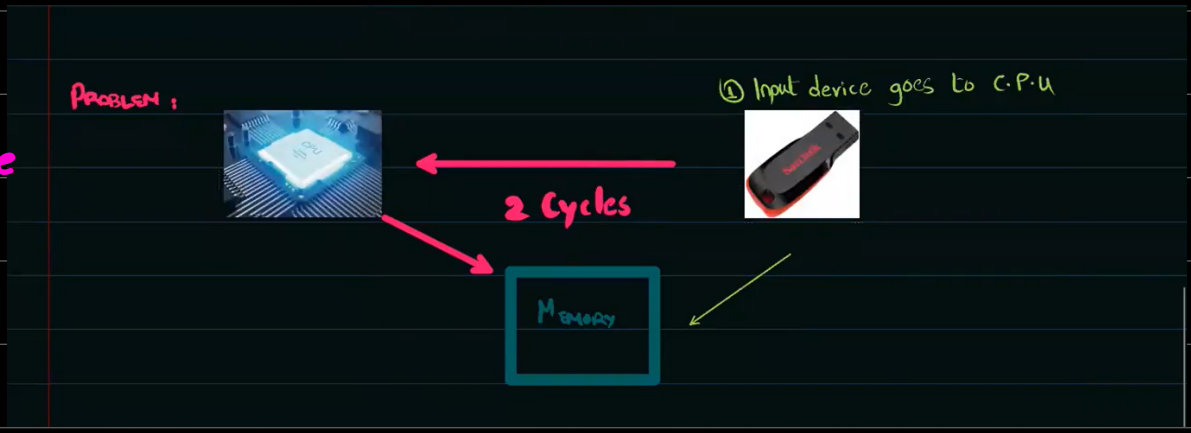
- CPU
- I/O device
- Memory

- Resource management focuses on utilizing the resources and maximise the use of resources.
- Deals with I/O operations

Direct Memory Access

Controller:

A physical device



- We use DMA controller to give access to memory directly. It allows the hardware to access the main memory independantly of the CPU.
- It frees up CPU to allow it to carry out other tasks.

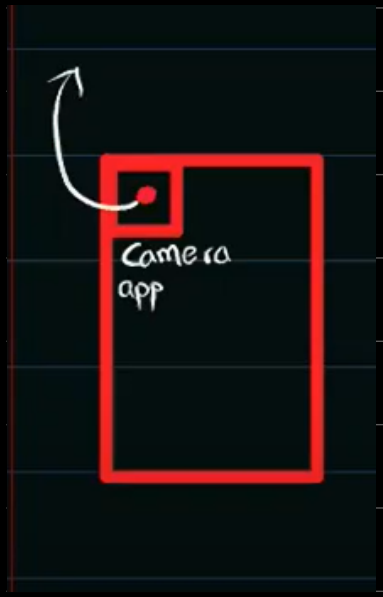
Steps:

1- DMA initiates the data transfer

2- While CPU carries out the other tasks

3- Once the data transfer is complete, an interrupt signal is sent to the CPU from the DMA.

KERNEL

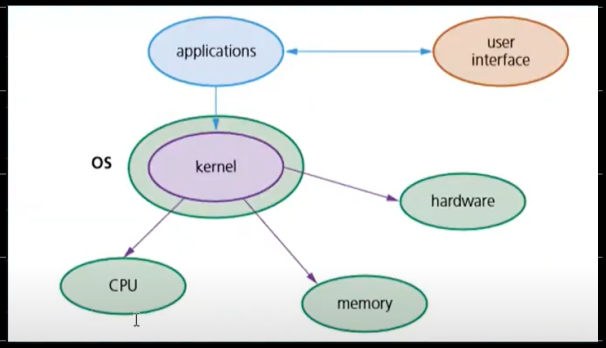


- If an application wants to access a hardware component such as flash light, the app first goes to kernel and seek permission to use it.

Q- What is kernel?

- It is part of an operating system
- Responsible for communicating b/w hardware, software, and memory
- Responsible for process management, device management, memory management.

Q- How the operating system hides the complexities of the hardware from the user?

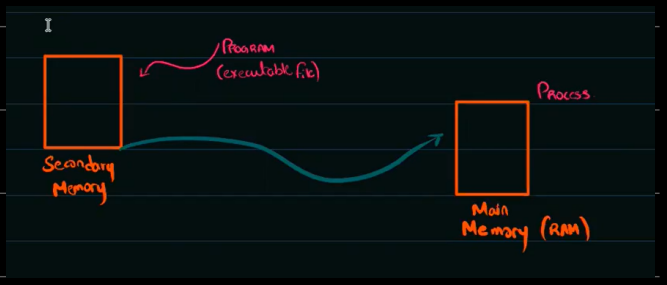


- Operating system provides interface e.g: GUI which helps to use the hardware
- Operating system uses device drivers to synchronize the hardware.

Process Management

Q- Explain the difference b/w a program and a process.

- Program is written code
- Process is the executing code.



Multi tasking

- Multitasking in an Operating System (OS) allows a user to perform more than one task at a time:

- To ensure multitasking operates correctly, scheduling is used to decide which process should be carried out.
- Multitasking ensures the best use of computer resources by monitoring each state of the process.
- It should seem that many processes are executed at the same time

- In fact, kernel overlaps the execution of each process based on scheduling algorithm

Process manager which handles the removal of running programs from the CPU and selection of new processes.

Types of Scheduling Algorithms

Pre-emptive: (To take action // steal). When CPU is allocated to a particular process and if at that time a higher priority process comes then CPU is allocated to that process.

Non-Preemptive: Does not take action until the currently running process is terminated

Pre emptive	Non-Preempt've
· Resources are allocated for a limited time	· Once the resources are allocated to a process, the process retains them until it has completed its burst time (amount of time required by a process for execution)
· The process can be interrupted while it is running	· The process can not be interrupted while running (It must finish or switch to waiting state)
· More flexible form of scheduling	· more rigid form of scheduling

Q- Explain why an operating system needs to use scheduling algorithms?

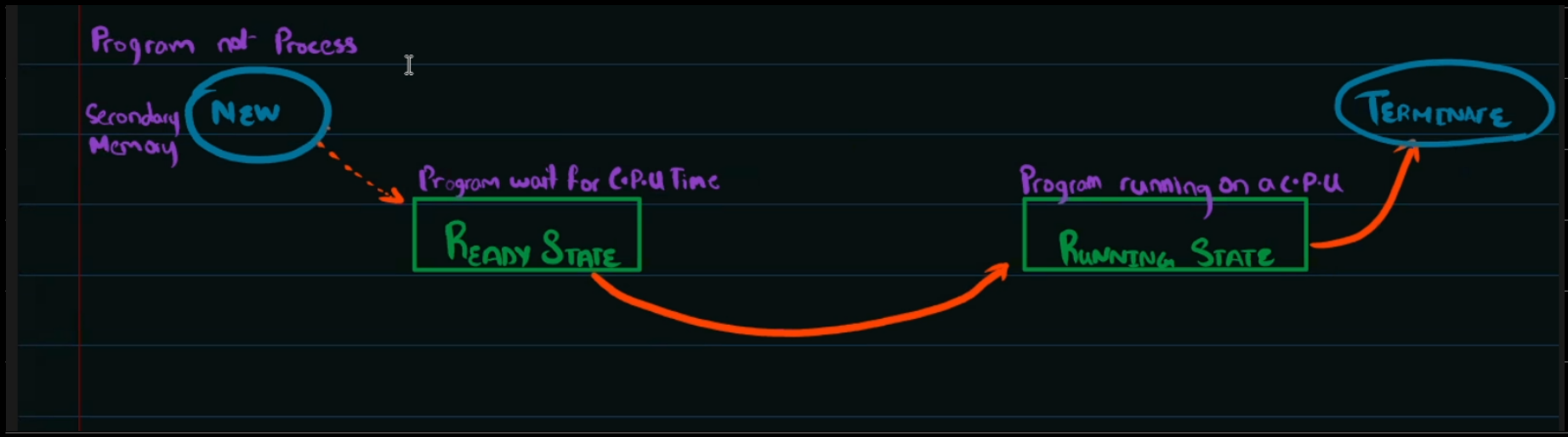
- To allow multi tasking to take place.
- To ensure fair usage of processor
- To ensure fair usage of peripherals (Hardware device)
- To ensure fair usage of memory
- To ensure higher priority tasks are executed sooner
- To ensure all the tasks have an opportunity to finish.
- To minimize the amount of time users wait for their results. I/O operation.
- To keep CPU busy at all times
- To service the largest possible number of jobs in the given amount of time.

* Higher priorities are taken into account in both: Preemptive and Non Preemptive. In Non-Preemptive, the current on going task will be finished first, then higher priority will be processed. (Will link ahead)

Process States

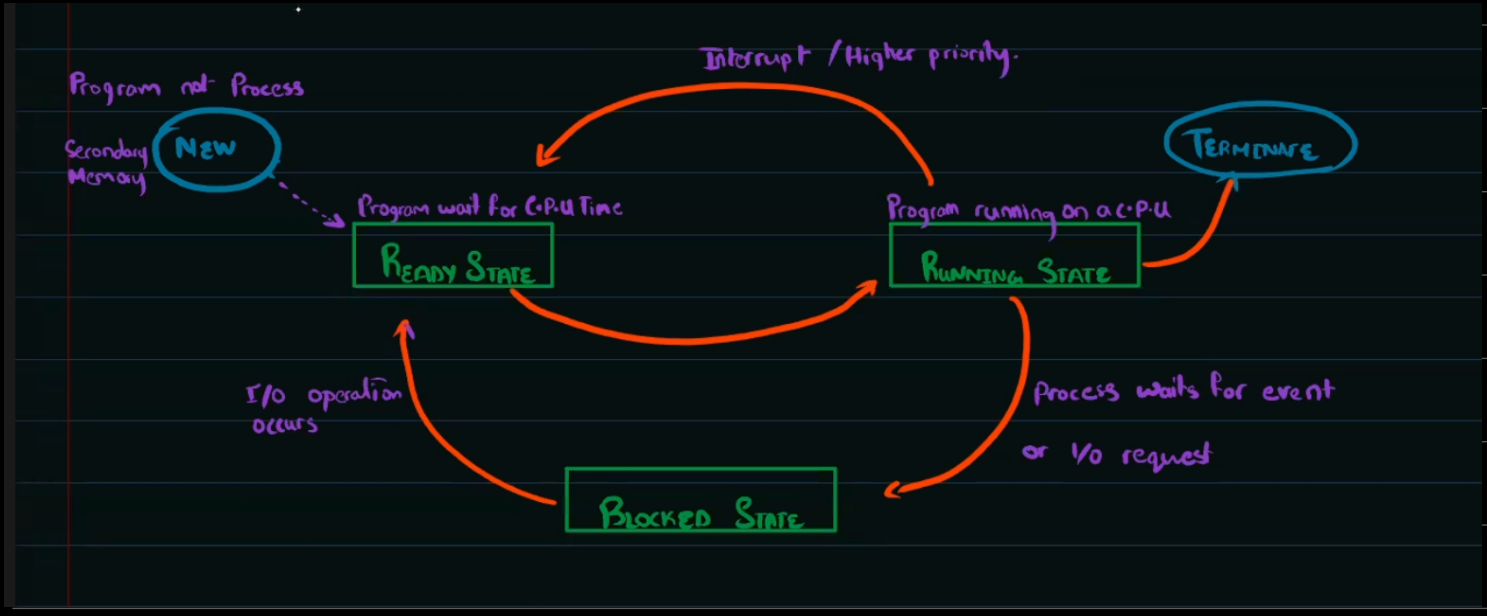
1. Running
2. Ready
3. Blocked

CASE 1: No Interrupts OR I/O Request



CASE 2 : I/O Request

CASE 3: Preemptive



① Ready State

Description:

- The process is not being executed
- The process is in the queue.
- Waiting for processor's attention.

Time slice: Random time given to a process for its partial completion

② Running State

Description:

- The process is being executed
- The process is currently using its allocated processor time / time slice

③ Blocked state

Description:

- The process is waiting for an event, so it can not be executed at the momment
- e.g: input/output

Conditions For Transition Between States

Ready to Running

- Processor is available, current process is no longer running
- Process was at the head of the ready queue. // Process has highest priority
- OS allocates processor to process so that process can execute.

Running to Ready

- When process is executing, it is allocated a time slice
- When time slice is completed, interrupts occur and process can no longer use processor even though it is capable for further processing

Running to Blocked

- Process is executing (running state) and when it needs to perform I/O operation, it is placed in blocked state until I/O operation is completed.

Q- Explain why a process can not be moved from blocked state to running state?

- When I/O operation completed for process in blocked state
- Process is transferred to ready state

- OS decides which process to allocate to processor.

Q- Explain why a process can not move directly from ready state to blocked state?

- To be in blocked state, process must initiate some I/O operation.
- To initiate operation, process must be executing
- If the process is in ready state, it can not be executing

Scheduler

High-level Scheduler

- Decides which processes are to be loaded from backing store.
- Into ready queue.

Secondary memory

Low-level Scheduler

- Decides which of the processes in ready state
- should get use of processor / or which process is put into running queue
- Based on position or priority

Scheduling Routine Algorithms

- First come, first serve scheduling
- Shortest job first scheduling
- Shortest - remaining time first scheduling
- Round Robbin

First Come First Serve Scheduling

- Non - Preemptive
- Based on arrival time
- Uses first in first out principle. (FIFO)

Process	Arrive Sequence	Burst Time
P1	1	23 ms
P2	2	4 ms
P3	3	1 ms
P4	4	3 ms

So the queue will be :

Gantt Chart:

Average waiting time for each process = $\frac{0+23+27+36}{4}$ = 21.5 ms

Shortest Job First Scheduling

- Non-Preemptive
- The burst time of the process should be known in advance.

Process	Arrive Sequence	Burst Time
P1	1	23 ms
P2	2	4ms
P3	3	9ms
P4	4	3ms

- With SJF, the process requiring the least CPU time is executed first.

So the ready queue will be :

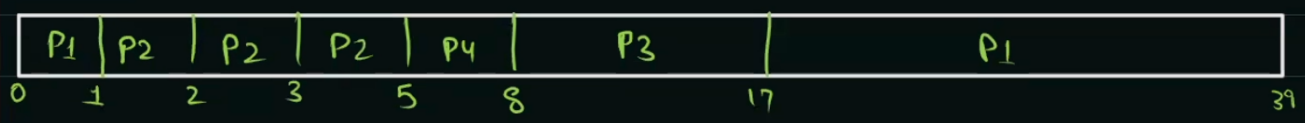
3

Average waiting time = $\frac{0+3+7+16}{4}$ = 6.5 ms

Shortest Remaining Time First

- Preemptive
- The processes are placed in ready queue as they arrive
- but when a process with a shortest burst time arrives
- the existing process is removed
- The shorter process is then executed first.

Process	Burst time (ms)	Arrival time of process (ms)
P1	23	0
P2	4 100%	1
P3	9	2
P4	3 100%	3



at time 0 ms: P1 (23 ms)

at time 1 ms: P1 (22) P2 (4 ms)

at time 2ms: P1 (22ms) P2 (3ms) P3 (9ms)

at time 3ms: P1 (22ms) P2 (2ms) P3 (9) P4 (3ms)

Round Robbin

- Preemptive
- A fixed time slice is given to each process, this is known as time quantum
- The running queue is worked out by giving each process its time slice in the correct order (if a process completes before the end of its time slice then the next process is brought into ready queue for its time slice.)

Process	Burst time (ms)	Arrival time of process (ms)
P1	~~23~~ ~~18~~ 13	0
P2	~~4~~ 100%	1
P3	~~9~~ 4 100%	2
P4	3 100%	3

Time Quantum = 5ms

Ready: ~~P1~~ | ~~P2~~ | ~~P3~~ | ~~P4~~ | ~~P1~~ | ~~P3~~ | P1

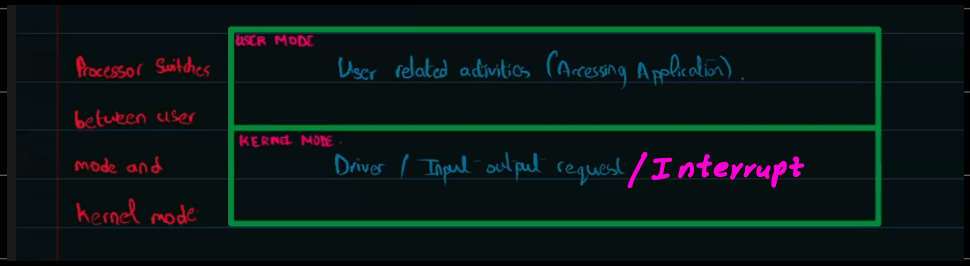
Running: P1 | P2 | P3 | P4 | P1 | P3 | P1 | P1 | P1

0 5 9 14 17 22 26 31 34 39

Interrupt Handling

User Mode and kernel Mode

- Driver enables communication b/w hardware and operating system



Interrupt

- Stop the continuous progress of an activity

- Interrupt is a kind of signal to OS from the device which is connected to computer. Sometimes interrupts are within the computer

- The processor will check for interrupt signals and will switch to kernel mode if any of the following type of interrupt signals are sent:

- **Device Interrupt:** e.g: Printer out of Paper
- **Exception:** e.g: Instruction faults such as division by zero
- **Traps / Software Interrupts:** e.g: Process requesting a resource

IDT

- Interrupt Dispatch Table
- To determine the current response to interrupt

IPL

- Interrupt Priority Level numbered (0-31)

Note: First Interrupt's priority is checked then IDT is referred to as accordingly.

- When an interrupt is received, other interrupts are disabled, so that the process that deals with the interrupt can not itself be interrupted
- The state of the current task / process is saved on the kernel Stack
- The system now jumps to the interrupt service routine (using IDT)

- Once completed, the state of the interrupted process is restored using the values stored on the kernel stack, the process then continues
- After an interrupt has been handled, other interrupts need to be restored so that any further interrupts can be dealt with.

Memory Management

- Frames and pages have same size
- A single frame will store a single page

- Rest of pages which are not present in main memory are stored in Hard Drive.

Page Replacement

- Page Replacement occurs when a requested page is not in main memory (flag=0). When paging in/out from memory, It is necessary to consider how the computer can decide which page(s) to replace to allow the requested page to be loaded. When a new page is requested but it is not in memory, a page fault occurs.

Page Replacement Algorithms

- First In First Out : The first page that is loaded will leave RAM first

- Optimal Page Replacement: Looks forward in time to see which frame it can replace in the event of page fault.

- Longest Resident: A particular page which is present for longest time is swapped. (Time of entry should be present in page table)

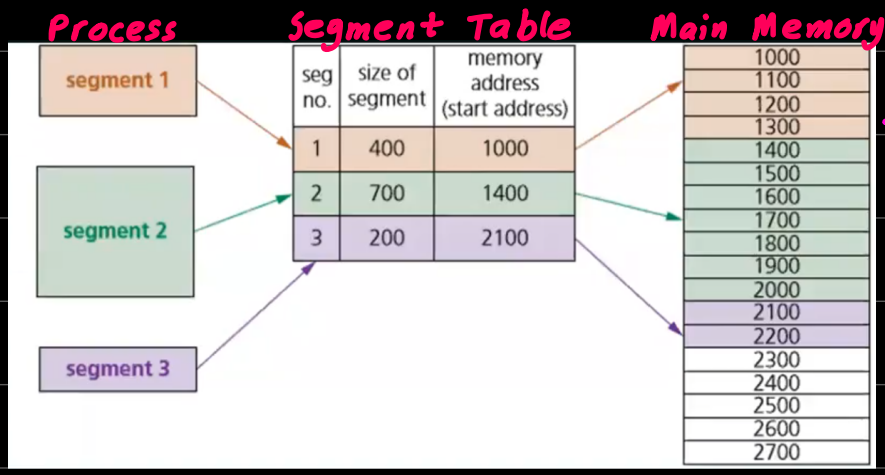
- **Least Used:** A particular page which is used less is swapped. (Number of times the page has been accessed should be present in page table)

Q- Explain why the algorithms (Longest resident / least used) may not be the best choice for memory management?

- **Longest Resident:** Page in for a lengthy period of time may be accessed often... so not good candidate for being removed.

- **Least Used:** A page just entered has a low least value, so likely to be a candidate for immediately being swapped out.

Segmentation



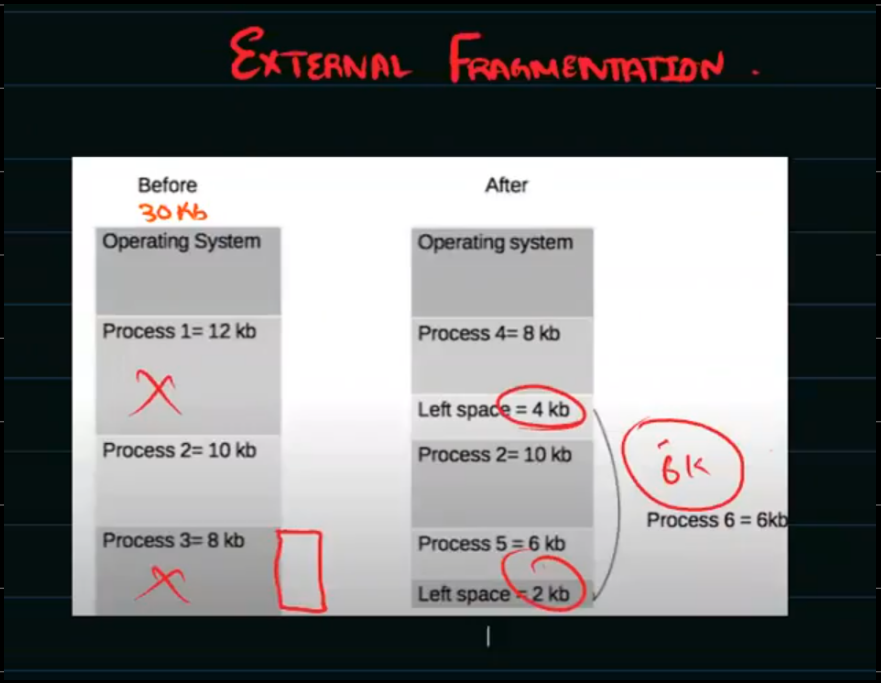
- Segments need to be stored in contiguous (Next to each other) for processing to occur

Internal Fragmentation

- Occurs when using paging method.

External Fragmentation

- Occurs when using segmentation.



Differences b/w paging and segmentation

Paging

- A page is a fixed size block of memory

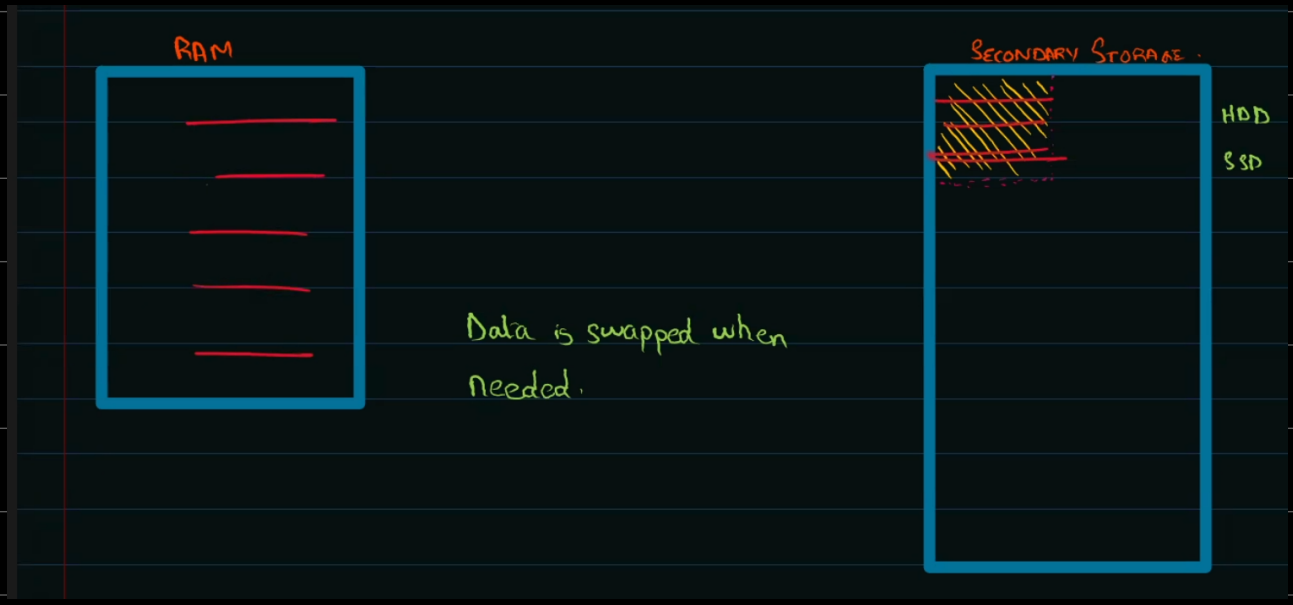
Segmentation

- A segment is a variable-size block of memory

· Since the block size is fixed, it is possible that all blocks may not be fully used. This can lead to internal fragmentation	· Memory blocks are variable-size, this increases the risk of external fragmentation
· The user provides a single value which means the hardware decides the actual page size.	· The user will supply the segment number and segment size
· Procedures (Modules) can not be seperated when using paging.	· Procedures can be seperated using segmentation.

Virtual Memory

- Works on the concept of paging.



Q- Describe what is meant by virtual Memory?

- Secondary storage is used to extend RAM.
- So CPU can access more memory space than available RAM.
- Only Part of program / data in use needs to be in RAM
- Data is swapped b/w RAM and disk

Q- Explain how paging is used to manage virtual memory?

- Divides RAM into Frames
- Divides virtual memory into blocks of same-size called pages
- Frames / pages are of a fixed and equal size.
- Set up a page table to translate a logical address to physical address
- keep track of all free frames
- Swap pages in memory with new pages from disk when needed.

Q- One drawback of using virtual memory is disk thrashing. Describe what is meant by disk thrashing.

- Pages are required back in RAM as soon as they are moved to disk
- There is continuous swapping (of same pages)
- No useful processing happens
- Because pages that are in RAM and on disk are inter-dependant
- Nearly all of the processing time is used for swapping pages.