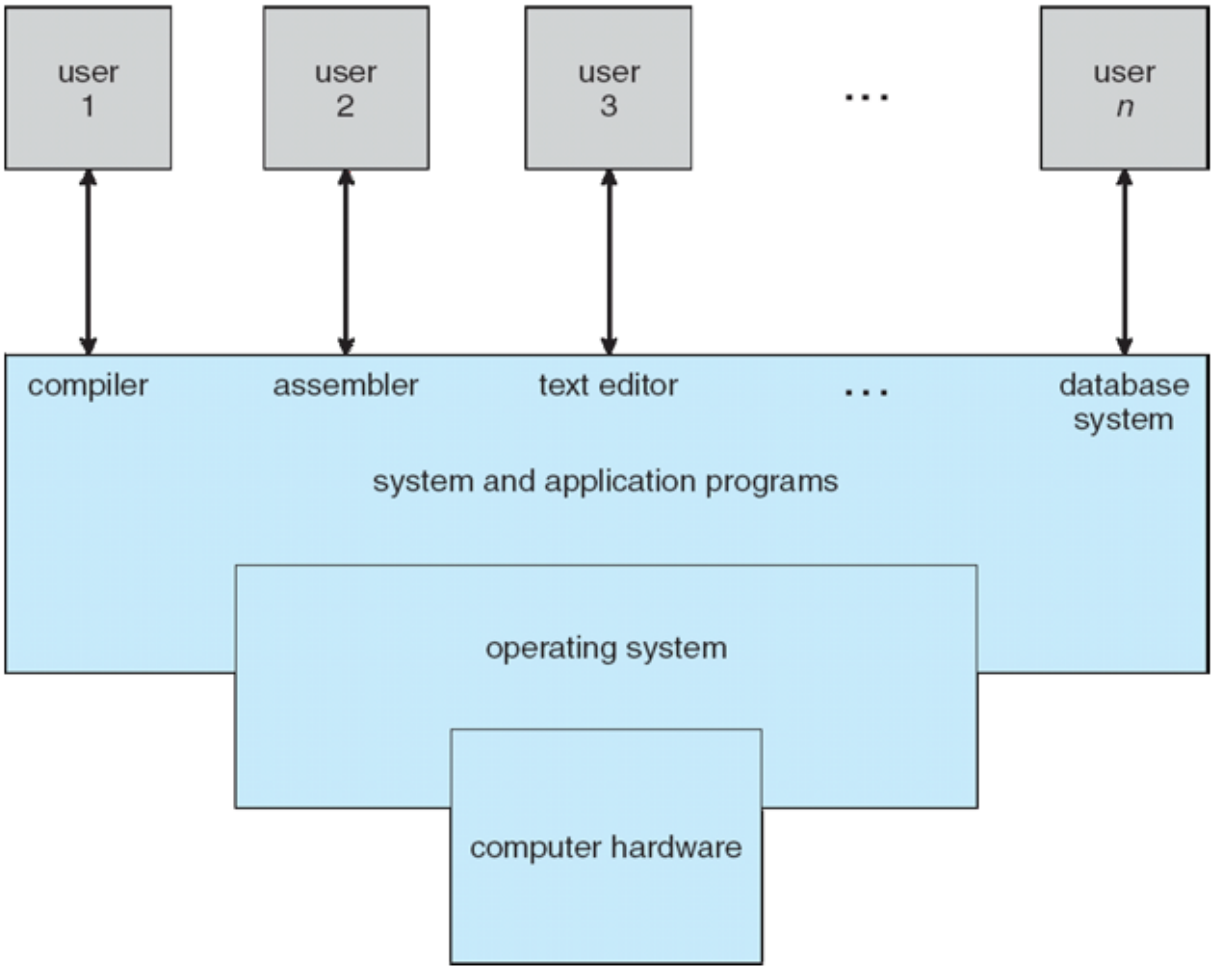
COS Module - Class Notes

Computer System: a computer system is a programmable electronic device that can accept input; store data; and retrieve, process and output information

Computer system can be divided into four components

- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
    Users

People, machines, other computers (Users)



Introduction to Operating System

→ Operating System: OS is a program that manages the computer hardware resources and provides a platform for application software to run.

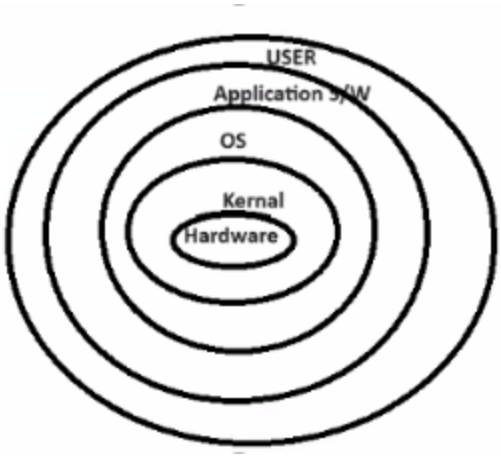
→ A program that acts as an intermediary between a user of a computer and the computer hardware

OS is a:

- Hardware Manager - It manage all the hardware resources of computer system
- Process Manager – It supervises all the tasks/processes that are ongoing in computer system

→ OS vs Application Software:

- OS manages system level tasks like Process Management, Memory Management & File Management while Applications are more focused on user centric tasks like creating documents.
- OS directly interacts with hardware of the computer while applications rely on OS to interact with the hardware.



History of OS:

→ When there was no operating system, there was no control of hardware and processes. All was done manually.

→ In old time punch cards are used for giving instructions to the machine and it was time lengthy process.

→ As OS got evolved, Efficiency (Throughput) got improved

→ Basic Computer Organizations required for OS:

1. CPU: executes instructions from program
2. Memory:

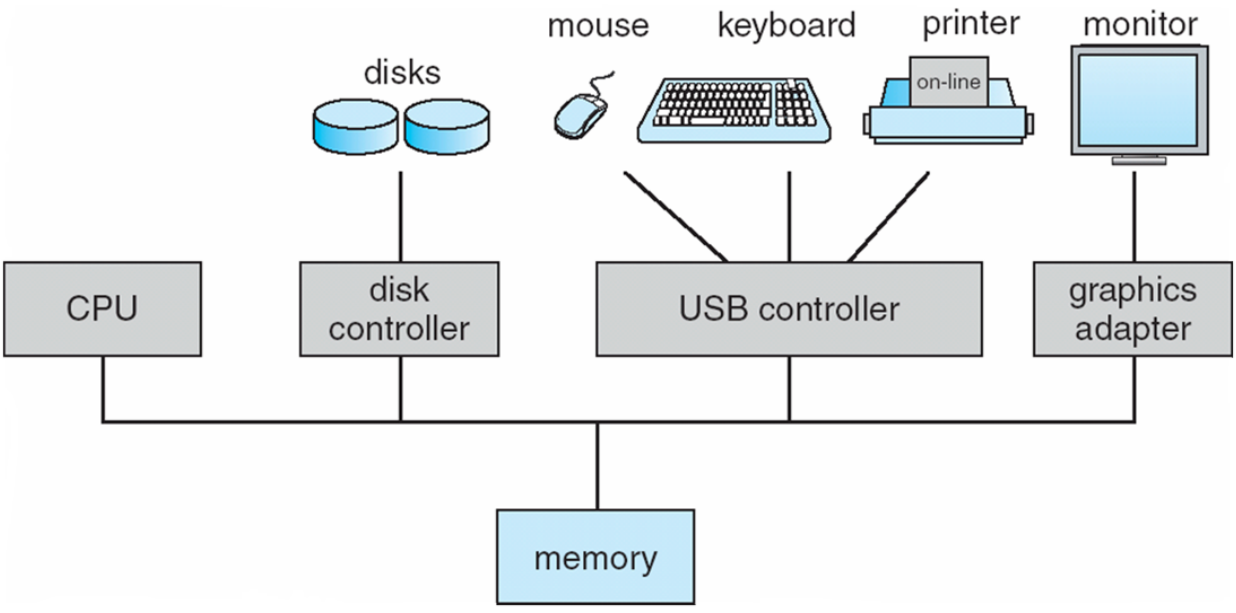
   To run OS, we need a physical memory which is high in speed than hard drive, RAM is a volatile memory which has high speed than hard drive

   So, we run the OS as well as programs/software on ram for quicker executions

   As OS is running, it may produce some intermediate results. Intermediate results are stored in registers (or Cache)

Speed of register is pretty much high (nearly same as processor)

3. Storage (HDD/SSD): Permanent Storage for Data, Files and OS itself.
4. I/O Devices
5. Common Bus:
    - One or more CPUs, device controllers connect through common bus providing access to shared memory
    - Concurrent execution of CPUs and devices competing for memory cycles
    - I/O devices and the CPU can execute concurrently
    - Each device controller is in charge of a particular device type
    - Each device controller has a local buffer
    - CPU moves data from/to main memory to/from local buffers
    - I/O is from the device to local buffer of controller
    - Device controller informs CPU that it has finished its operation by causing an *interrupt*



Booting of OS

1. Power On
2. CPU will move to BIOS in ROM

   BIOS: Basic Input Output System. it's a program that runs on a computer's microprocessor to start the computer when it's turned on
3. BIOS will be executed
4. Power On Self-Test (POST): All the Hardwares are tested.
5. Bios will load MBR (Master Boot Record) to RAM

   The Master Boot Record (MBR) is **a small program that's located at the beginning of a computer's hard disk or other partitioned memory hardware devices**. It's a vital part of a computer's storage device that contains the primary bootloader and partition table, which are used to start the operating system when the computer is turned on.
6. MBR will load Bootloader to RAM

   Bootloader is a program that loads OS to the RAM.
7. Bootloader will load OS to RAM

→ Cold booting is the mechanism of starting a system after it has been turned off.

→ Hot booting is the technique of restarting a system. It may be started up using the OS.

Example of OS

1. Mobile OS
2. Embedded System OS
3. Real Time OS (RTS)
4. Desktop OS
5. Server Machine OS

Types of OS: (https://www.geeksforgeeks.org/types-of-operating-systems/)

1. Batch Operating System

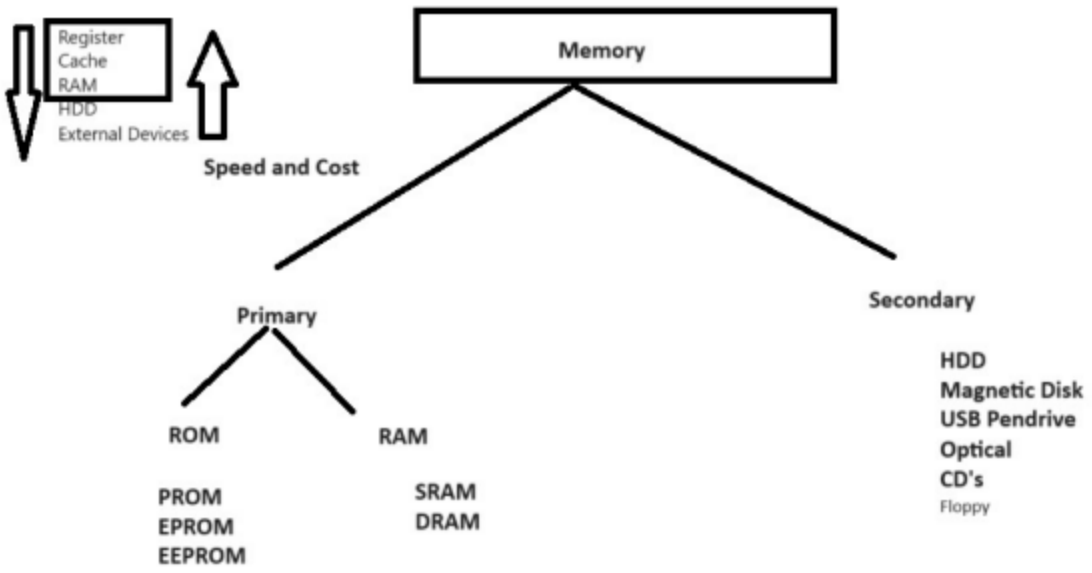
2. Multi-Programming System
3. Multi-Processor System
4. Multi-Tasking Operating System
5. Time-Sharing Operating System
6. Distributed Operating System
7. Network Operating System
8. Real-Time Operating System

Functions of OS:

1. Process Management
2. Memory Management
3. Device Management
4. Disk Management
5. Network Management
6. File Management
7. Security Management

Memory Hierarchy in Computer System

1. Primary Memory: RAM, ROM (PROM, EPROM, EEPROM): BIOS/Firmware/UEFI
2. Secondary Memory: HDD, SSD, Magnetic Tape, Pendrive, External HDD, CDs, Floppy



Process Management

User and Kernel Mode

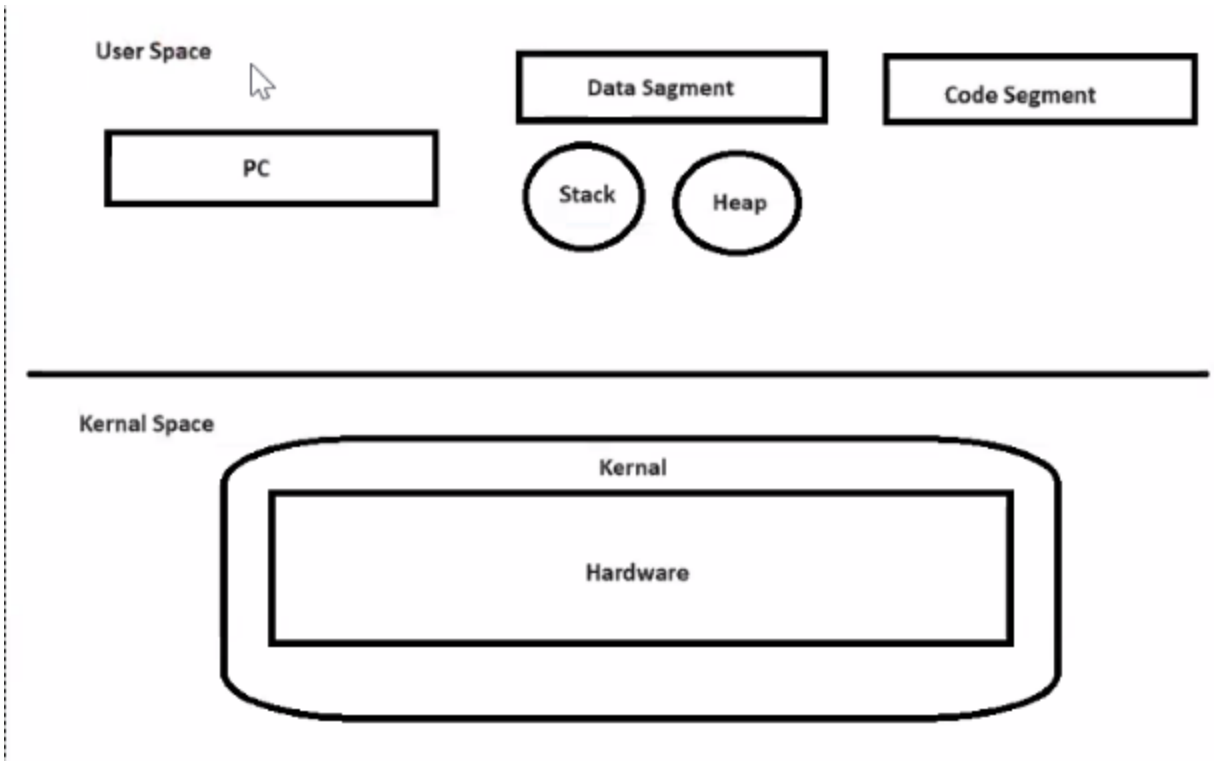
→ Kernel: The kernel is a computer program at the core of a computer's OS and generally has complete control over everything in the system.

> It is the portion of the OS code that is always resident in memory and facilitates interactions between hardware and software components. It is in always running state.

→ The critical code of the kernel is usually loaded into a separate area of memory (Kernel Space), which is protected from access by application software or other less critical parts of the operating system.

> The kernel performs its tasks, such as running processes, managing hardware devices such as the hard disk, and handling interrupts, in this protected kernel space

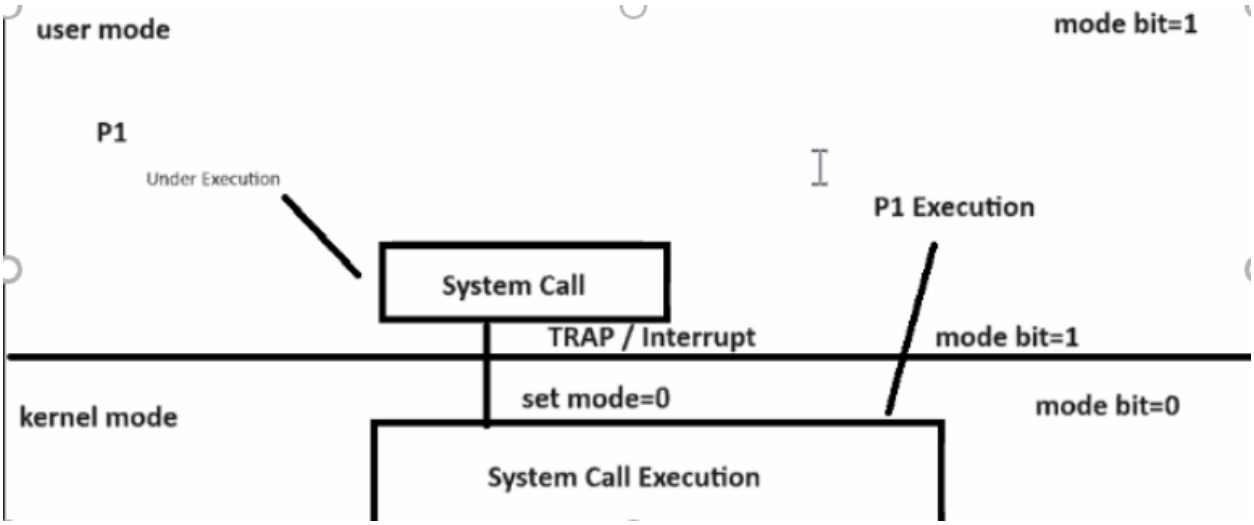
→ Application programs such as browsers, word processors, or audio or video players use a separate area of memory known as user space.



→ In User Mode, we do not have direct access to the hardware

→ the main functionality of the Kernel Mode is to execute privileged instructions by having access to the hardware

→ Mode bit is required to identify in which particular mode the current instruction is executing. If the mode bit is 1, it operates user mode, and if the mode bit is 0, it operates in kernel mode.



→ User mode and kernel mode communicate with each other using system calls.

Categories of system calls: (Read About these)

1. File related calls: Read(), Write(), Delete(), Open(), Close(), Create().
2. Process related calls: New(), Fork(), Exit(), Wait(), Running().
3. Device related calls: Read(), ioctl()
4. Information related: getpid(), getppid, gettime, sysdata
5. Communication Related: wait(), signal(), status()

Process Management

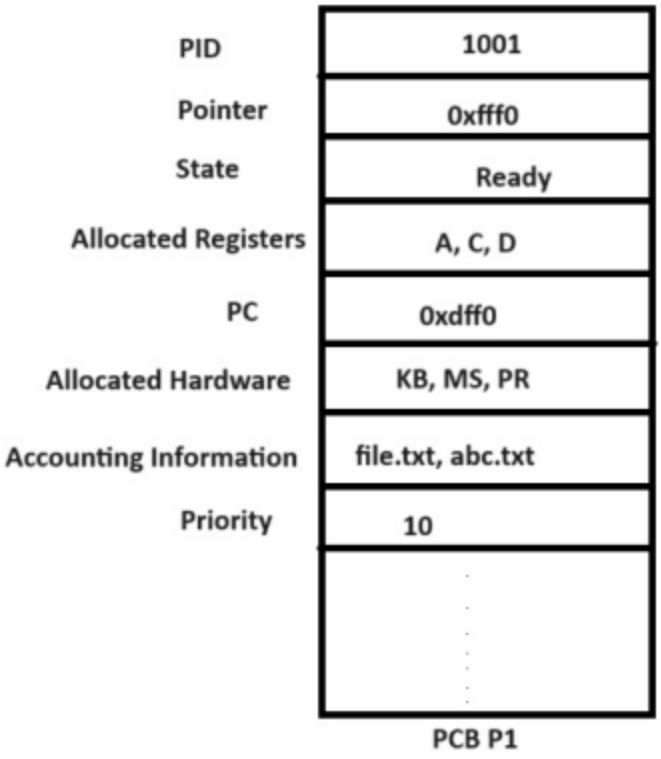
→ Process is an information/code/data which helps the processor to execute or complete the user task.

→ Different units of the process:

1. Code segment: It consist of compiled code or instructions to be executed
2. Data segment: It consist of data required for the execution
3. Information segment: It have metadata about the system variables or system which help in the process execution.
4. Memory segment: Heap and stack memory, PCB

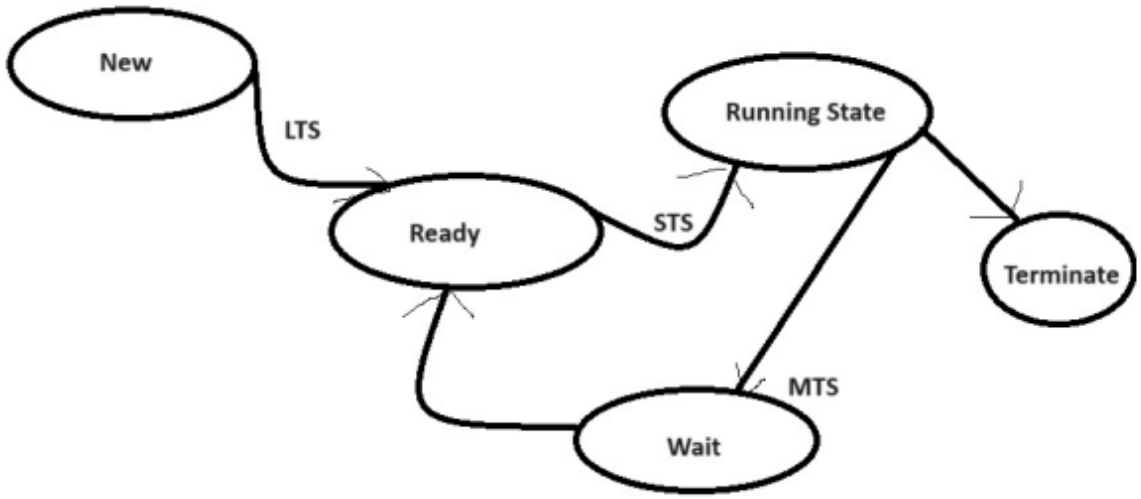
→ Process Control Block: it has full information about process.

read: https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/



→ Process Life Cycle:

1. New State: When your process is created
2. Ready State: When your process is scheduled for CPU. (Ready Queue)
3. Running: Processor is executing the process
4. Terminated State
4. Wait State: after i/o request generation (It will go back to ready state)



→ Process Schedulers : essential parts of operating systems that manage how the CPU handles multiple tasks or processes

1. Long Term Schedulers (LTS) : if process can wait for long time, it is done through LTS
2. Short Term Schedulers (STS): if the process is urgent and need to handled urgently, it is done through STS
3. Medium Term Schedulers (MTS)

→ **Process Starvation:** If the process is waiting for getting executed for long time

While execution of a process with higher priority, some processes had to wait to run indefinitely because of low priority

read more : https://www.geeksforgeeks.org/starvation-and-aging-in-operating-systems/

Process Scheduling Algorithms

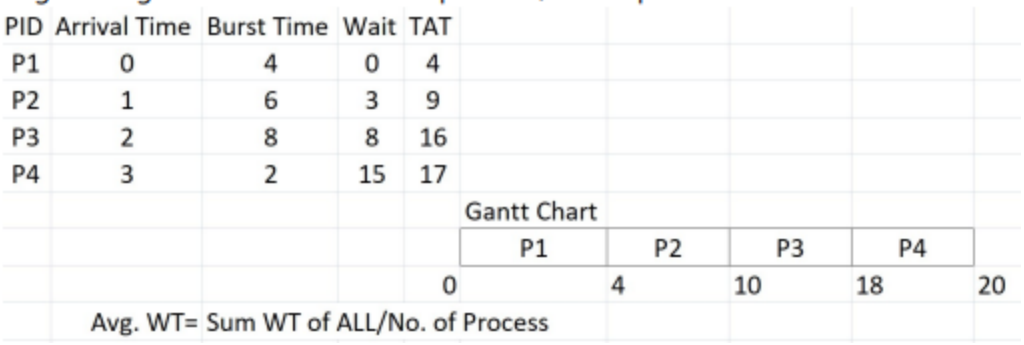
1. Preemptive Scheduling : Preemptive scheduling allows a running process to be interrupted by a high priority process (Context Switching)
2. Non preemptive scheduling: In non-preemptive scheduling, **any new process has to wait until the running process finishes its CPU cycle**.

- Waiting time of the process = CPU Allocation Time – Arrival Time
- Avg Waiting Time = Sum of waiting time of all process/ No. Of Process
- Turn Around Time (TAT) = Process end time – Process arrival time
- Avg TAT = Sum TAT/No. Of Process

Non Preemptive Scheduling:

1. First Come First Serve (FCFS) scheduling: FCFS Scheduling algorithm automatically executes the queued processes and requests in the order of their arrival.

   - Waiting Time of Process= CPU Allocation-Arrival Time
   - Avg Waiting Time = Sum WT of All process / no. of processes



PID	Arrival Time	Burst Time	Wait	TAT
P1	0	4	0	4
P2	1	6	3	9
P3	2	8	8	16
P4	3	2	15	17

   → Disadvantages: Convey Effect

   When the process that has arrived first uses a lot of burst time and other processes has to wait till it completes, known as Convey Effect.

2. Shortest Job First (SJF) scheduling: The shortest job first (SJF) selects the waiting process with the smallest execution time to execute next. (can be preemptive or non preemptive)

PID	Arrival Time	Burst Time	CT	Wait	TAT
P1	0	4	4	0	4
P2	1	6	12	5	11
P3	2	8	20	10	18
P4	3	2	6	1	3

Gantt Chart

P1	P4	P2	P3

0 4 6 12 20

→ Disadvantages: If smaller processes come in queue again and again, CPU would not attend larger processes, so starvation will occur

3. Round Robin Algorithm: In round robin algo a fixed slice of time is given to each an every process. That slice of time is know as quantum.

PID	Arrival Time	Burst Time	CT	Wait	TAT
P1	0	4	10	6	10
P2	1	2	4	1	3
P3	2	6	16	8	14
P4	3	8	20	9	17

P1	P2	P3	P4	P1	P3	P4	P3	P4

0 2 4 6 8 10 12 14 16 20

Round Robin Algo with Arrival time 0:

PID	Arrival Time	Burst Time	CT	Wait	TAT
P1	0	2	2	0	2
P2	0	4	10	6	10
P3	0	6	14	8	14
P4	0	2	8	6	8

P1	P2	P3	P4	P2	P3	P3		

0 2 4 6 8 10 12 14

→ disadvantages: Performance depends on time quantum size: If the quantum is too large, it behaves like FCFS, and if it's too small, it causes too many context switches.

→ This would lead to low CPU throughput

Preemptive Algorithms:

Processes are handled according to their priority

PID	Arrival Time	Burst Time	Priority	CT	Wait	TAT
P1	0	2	7	12	10	12
P2	0	4	5	10	6	10
P3	0	6	1	6	0	6
P4	0	2	9	14	12	14
					Avg WT= 7	Avg TAT=10.5

	P3	P2	P1	P4		
0		6	10	12	14	

PID	Arrival Time	Burst Time	Priority	CT	Wait	TAT
P1	0	2	7	12	10	12
P2	1	4	5	11	6	10
P3	2	6	1	8	0	6
P4	2	2	9	14	10	12
					Avg WT= 6.5	Avg TAT=10

	P1	P2	P3	P2	P1	P4	
0		1	2	8	11	12	14

→ disadvantages - A low-priority process may wait a long time or indefinitely before it can run due to the continuous arrival of higher-priority processes causing Process Starvation

---

Memory Management

→ The memory of computer systems can be divided into block of fixed size or variable size.

→ Fixed Size Partitioning: Here the memory is divided into fixed size blocks where all the blocks are of size like either 2 Bytes, 4 Bytes , etc.

- The Process which will get the memory may be small or equal to the partitioning block size.

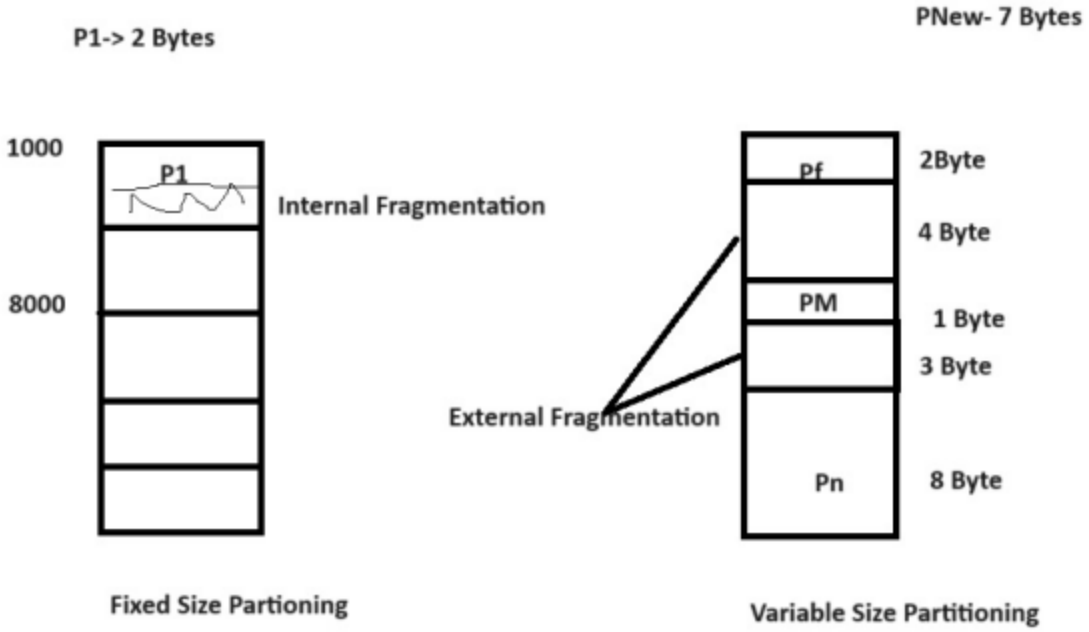
- As the process size may be smaller than block, it give rise to Internal Fragmentation

→ Variable Size Partitioning: Here the memory is divided into variable size blocks in which blocks may have size 2 Bytes, 3 Bytes or any size.

- In variable size partitioning during contiguous memory allocation some time total memory
available can not given to the new process which leads to External Fragmentation.

- To tackle external fragmentation, CPU deallocates and allocates memory again to the processes as per the requirement, it is known as compaction

- it is very time consuming and CPU heavy process.



→ First Fit Allocation: Allocates first memory which is capable of storing the data

→ Best Fit Allocation: Minimum required memory block is assigned for the data

→ Worst Fit Allocation: Maximum memory block is assigned for the data

Paging

→ Dividing the process into fixed sizes pages is know as paging.
→ Why Paging: Instead of loading the whole process which can't be loaded into main memory, OS loads few pages of the process into main memory according to frames available and other pages
loaded from virtual memory on demand of CPU

→ Only necessary part is loaded in main memory.

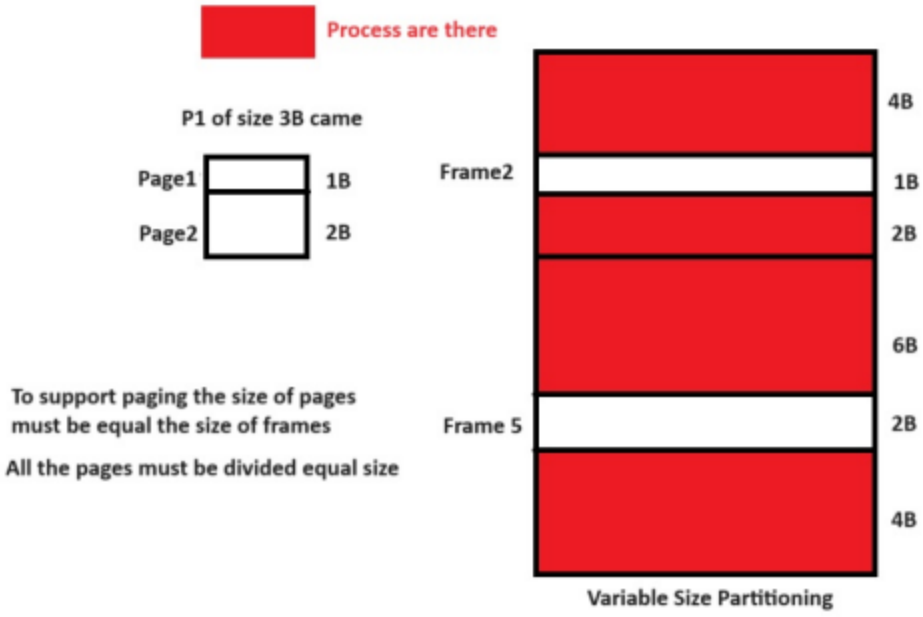
→ RAM Frames: Dividing ram in multiple memory locations of same size

→ To support paging, Size of pages must be equal to size of frames.

→ Size of the pages is always equivalent

→ Every frame in ram is of same size

→ Because of this, there would not be any internal and external fragmentation

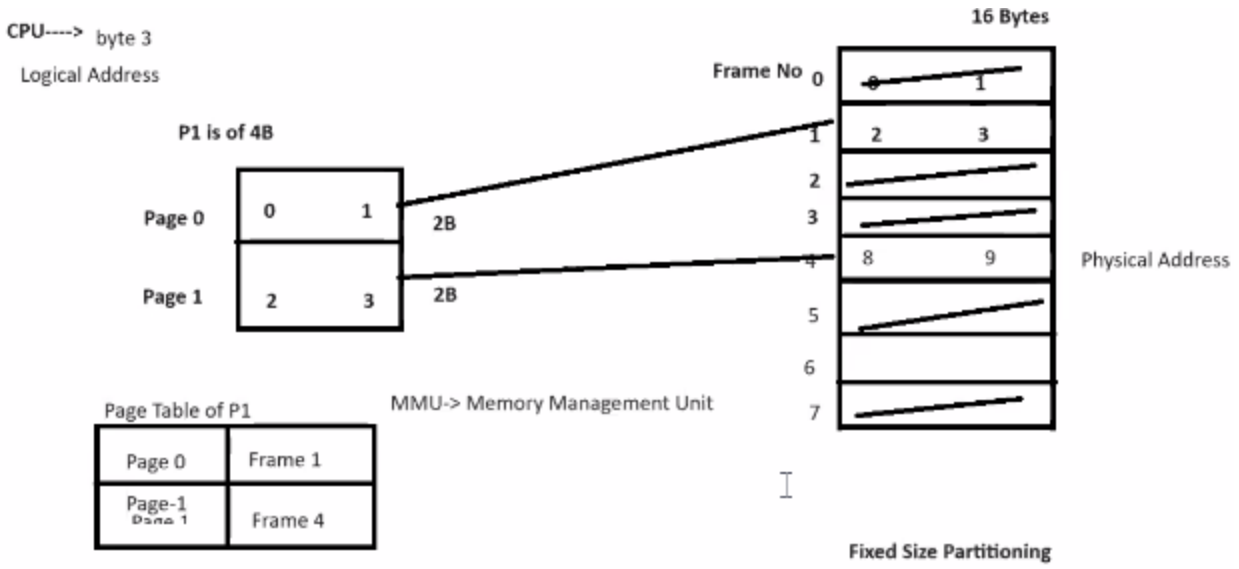


→ Program data is byte addressable

→ When CPU generates a request for byte of a page, it generates address according to the actual address of pages (Logical address) but the pages could be loaded in different physical address of ram

→ So the mapping of logical and physical address process is done by Memory Management Unit

→ It creates page table per process (read about aging in process handling)



→ Miss: when CPU demand the page and is not available in main memory, the it is known as miss. (Page fault)

→ Hit: If we get page that is required by CPU in main memory. Then it is hit.

→ Hit ratio and miss ratio decides paging efficiency

→ After a page fault, a interrupt is generated to load the page from hard drive (Virtual Memory)

→ Demand Paging: if the page is loaded into memory frame as per the demand of CPU i.e. known as demand paging

Page Replacement Algorithms

1. FIFO algorithm: First In First Out

→ removes the Page in the frame which is allotted long back

P1---->	0	2	1	2	0	3	5	3	2	1	0
FIFO											
Frame-3			1	1	1	1	1	1	2	2	2
Frame-2		2	2	2	2	2	5	5	5	5	0
Frame-1	0	0	0	0	0	3	3	3	3	1	1
	MISS	MISS	MISS	HIT	HIT	MISS	MISS	HIT	MISS	MISS	MISS
Frame-4						3	3	3	3	3	3
Frame-3			1	1	1	1	1	1	1	1	1
Frame-2		2	2	2	2	2	2	2	2	2	0
Frame-1	0	0	0	0	0	0	5	5	5	5	5
	MISS	MISS	MISS	HIT	HIT	MISS	MISS	HIT	HIT	HIT	MISS

→ In FIFO, for certain examples, increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern. this is known as Belady's Anomaly.

→ *Note – It is not necessary that every string reference pattern cause Belady anomaly in FIFO but there is certain kind of string references that worsen the FIFO performance on increasing the number of frames.*

2. LRU algorithm: Least Recently Used

→ whenever a page fault occurs, the least recently used page will be replaced with a new page in memory.

P1----->	2	3	2	4	3	5	4	3	2	7	
LRU											
		2	3	2	4	3	5	4	3	2	7
Frame-1	8	8	8	8	4	4	4	4	4	4	4
Frame-2	3	3	3	3	3	3	3	3	3	3	3
Frame-3	2	2	2	2	2	2	2	2	2	2	2
Frame-4	6	6	6	6	6	6	5	5	5	5	7
	Init	HIT	HIT	HIT	MISS	HIT	MISS	HIT	HIT	HIT	MISS
					Mis %	30					
					Hit %	70					

3. MRU algorithm: Most Recently Used

   → whenever a page fault occurs, the most recently used page will be replaced with a new page in memory.

P1---->	0	2	4	6	8	7	3	5	3	2	
MRU											
		0	2	4	6	8	7	3	5	3	2
Frame-1	0	0	2	4	6	8	7	7	7	7	7
Frame-2	3	3	3	3	3	3	3	3	3	3	2
Frame-3	5	5	5	5	5	5	5	5	5	5	5
	Initial	HIT	MISS	MISS	MISS	MISS	MISS	HIT	HIT	HIT	MISS

MISS Ration=60%

HIT= 40%

If the page is not in the frame, its known as miss of the mage and it's a page fault

4. OPR: Optimal Page Replacement Algorithm

   Replace that page which is required by CPU after long gap

   by replacing the page that is least likely to be used in the future

CPU--->6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6

	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame-3			8	9	9	9	1	1	1	1	1	1	1	1	6
Frame-2		7	7	7	7	7	7	7	7	7	7	7	7	7	7
Frame-1	6	6	6	6	6	6	6	6	6	8	9	9	9	9	9
	M	M	M	M	H	H	M	H	H	M	M	H	H	H	M

Miss % 8/15

Hit % 7/15

> Note – watch out for frame sizes in mcqs

→ **Virtual Memory** is region of hard drive which acts like ram

→ Virtual Memory is known as swap area

→ The virtual memory helps in paging and segmentation

→ Every program that needs to be executed or needed by CPU are stored in virtual memory in the form of pages.

→ OS loads the process pages from virtual to physical memory on demand of CPU i.e. known as Demand Paging

→ While working with virtual memory, pages are saved as per their logical address

The logical Address generated by CPU consists of:

1. Page No (P): The page number of the process.
2. Page Offset (D): The required byte number of page.

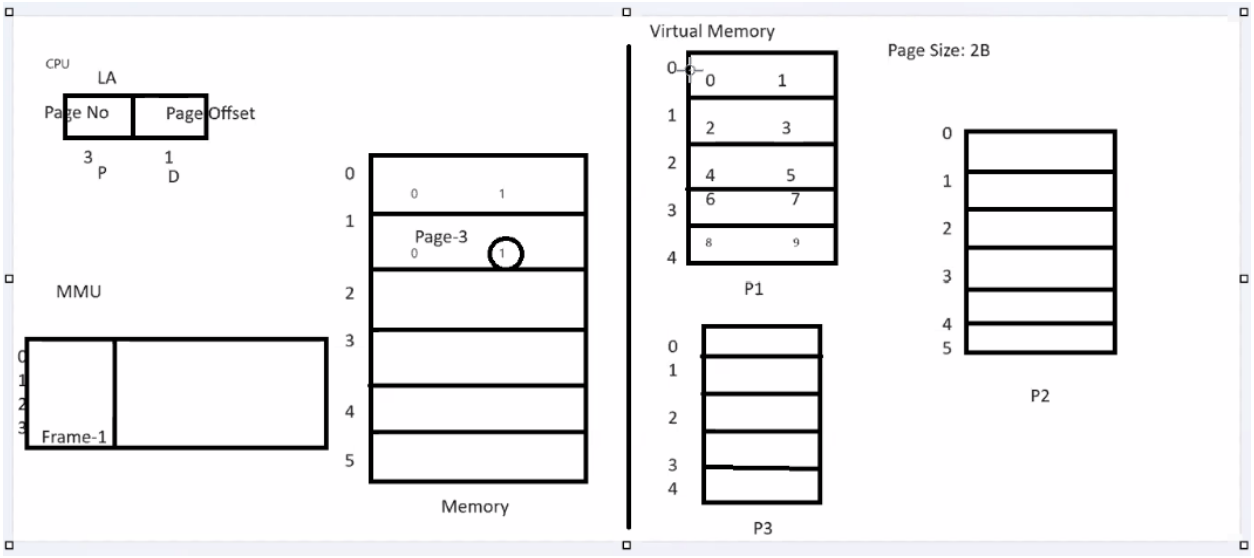
→ While loading pages from virtual to physical memory (Swap In), the pages get physical address.

→ If a page size is of 2 bytes, the number of pages can be generated in 4GB memory

$$\text{Total No of Pages} = \frac{4 \times 1024 \times 1024 \times 1024 \text{ bytes}}{2 \text{ bytes (Page Size)}}$$

→ To have addresses for pages in virtual memory, a architecture of CPU with required bits is selected. (Byte addressing)

```
No of bits to store address n = log2 P
//p: Number of Pages in virtual ram
```



→ Memory Management Unit (MMU): Pages to Frames Mapping

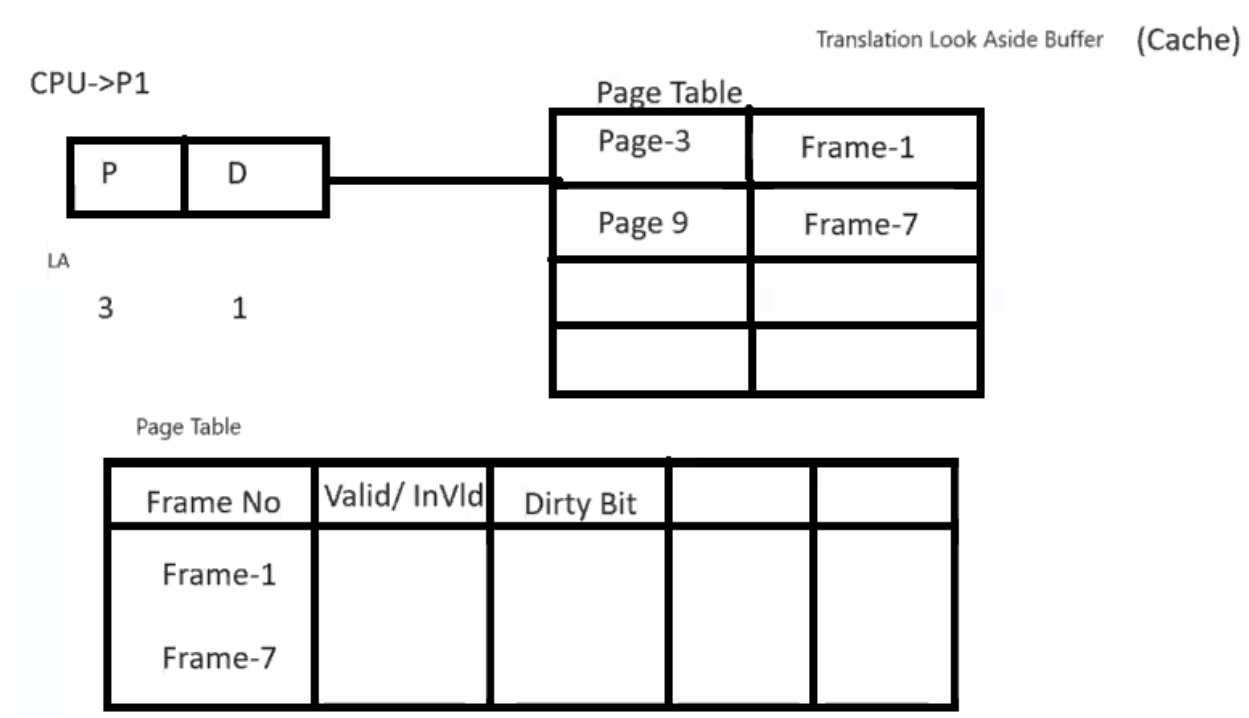
→ In paging, we load only few required pages in main memory. Entries of those pages is made in MMU Page Table.

→ Page table itself is also present in ram

→ If page is not present in page table, MMU will generate a TRAP (request) in virtual memory for page. Here, page replacement algorithms would come into picture

→ Transition Look Aside Buffer is basically a part of cache memory is used to temporarily save some of the physical and logical addresses of pages from page table to reduce the time.

→ While accessing the page, the CPU has to access RAM twice to look for and get the page. So, to make this process faster, entries of some of them is stored in cache as Transition Look Aside Buffer

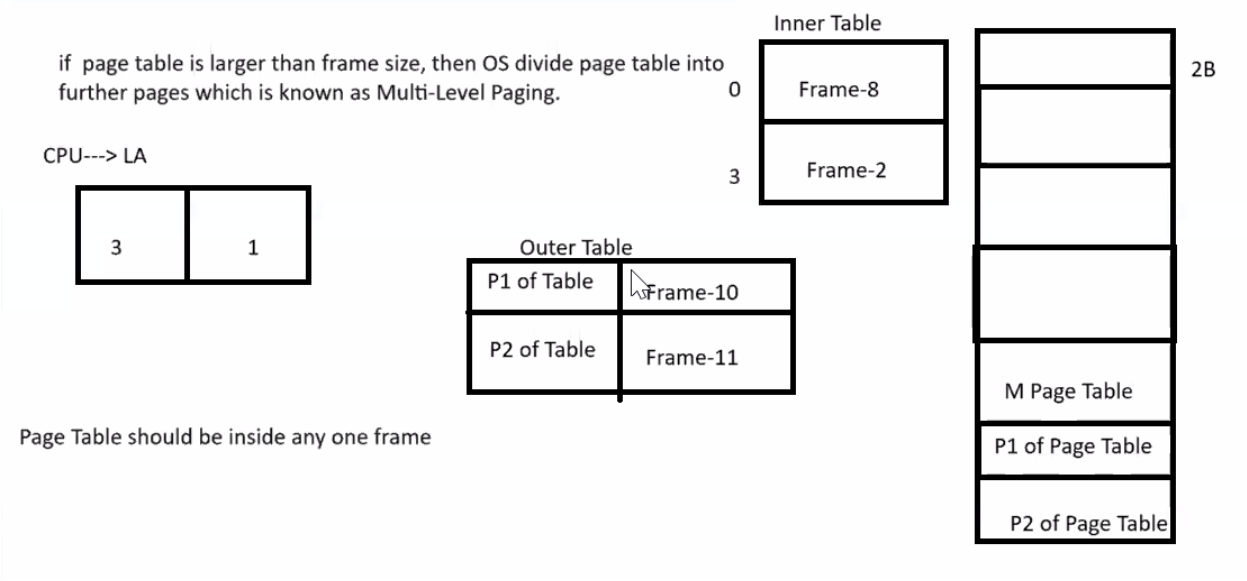


→ Valid/invalid: If the bit is set as valid, then it means that the page is valid and is present in main memory. If the bit is set as invalid, then either the page is not valid or the page is valid but not present in the main memory

→ Dirty Bit: If any changes made to the original page during execution, the dirty bit status would be changed and then updated page is stored in virtual memory.

Multilevel paging or Two level paging:

→ If page table is larger than frame size, then OS divides the page table into further pages which is known as Multi-Level Paging.



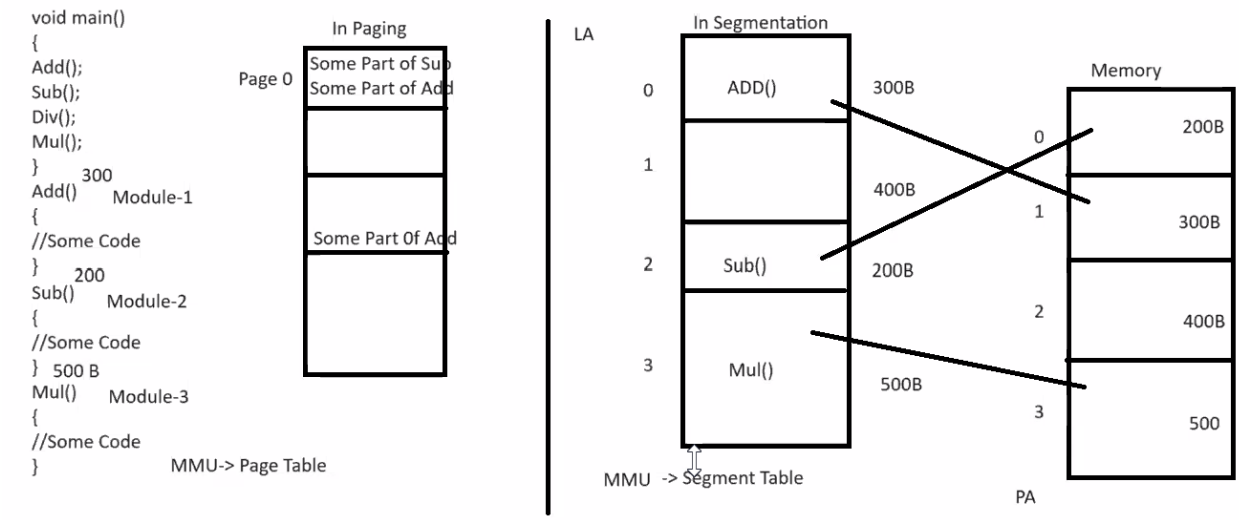
//concepts to read: Shared Pages and reentrant code, Throttling

Segmentation

→ In paging, the data is divided into fixed size pages where data of the process is stored. but the data of the one segment of the process can be divided into multiple pages and some of them might not have loaded into physical memory. this will create CPU overhead.

→ Segmentation is the method of dividing a process into variable size partitions known as Segments on the basis of modules/functions of the process to be executed

→ Segment Table



→ Hardware Requirement for Segmentation: Virtual Memory, Physical Memory, Cache

→ Difference between Pages and Segmentation:

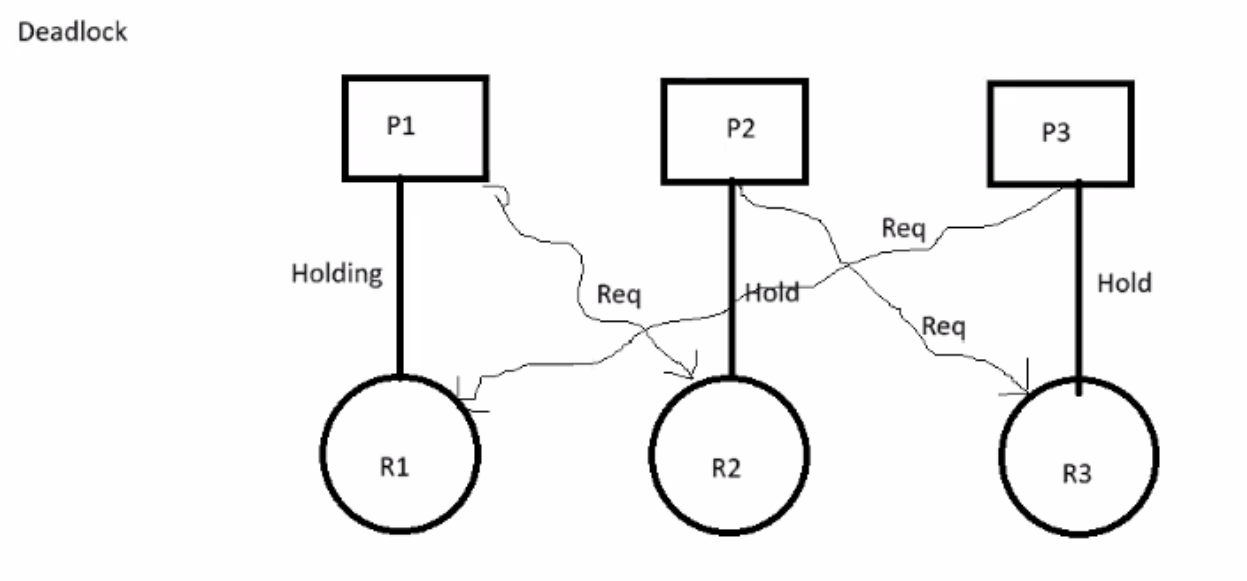
Paging is done with respect to the OS, Segmentation is done with respect to the User.

//read other points also

Deadlock

→ It is a condition where OS cannot continue the execution because of processes demanding those resources which are held by other processes hence deadlock occur

Deadlock



→ Required Reasons for Deadlock:

1. Mutual Exclusive of resources: other processes cannot interrupt the already held resources by a process.
2. No Preemption: If a process is holding a resource, no other process can prempt it.
3. Hold and Wait: Process are allowed to make request for other resource while holding one.
4. Circular wait: while holding and waiting for resources there should become a circular wait

Deadlock Handling Techniques

1. Deadlock Avoidance or Ignorance: As deadlock is very rare, it is ignored
2. Deadlock Prevention (Proactive): do one condition false
3. Deadlock Recovery (Reactive):
    a. kill a process
    b. Add multiple instances of resources
    c. Resource Allocation Graph (RAG)

Deadlock vs Starvation:

Starvation gets resolved after some time and Deadlock will not.

Process Synchronization:

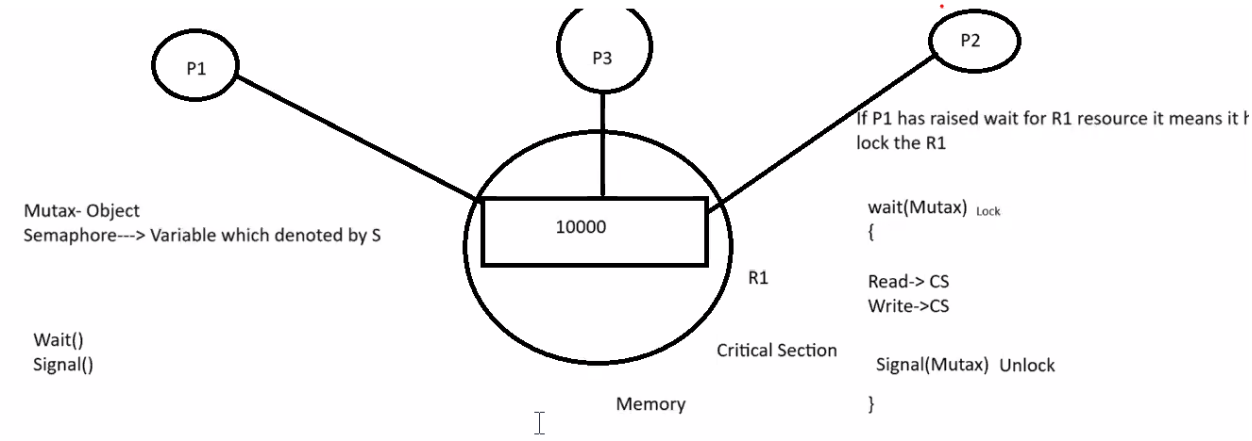
Processes Synchronization or Synchronization is the way by which processes that share the same memory space are managed in an operating system

Allowing Processes to work on or access (read/write) to a resource like memory one by one is called process synchronization.

Critical Section : A critical section is **a code segment that can be accessed by only one process at a time**. The critical section contains shared variables that need to be synchronized to maintain the consistency of data variables.

Mutex: A mutex is a locking mechanism used to synchronize access to a resource

Mutex is a specific kind of binary semaphore that is used to provide a locking mechanism



Semaphore (S) : To handle read processes in good manner

A semaphore is a non-negative integer variable that is shared between various threads. Semaphore works upon signaling mechanism, in this a thread can be signaled by another thread. Semaphore uses two atomic operations for process synchronisation:

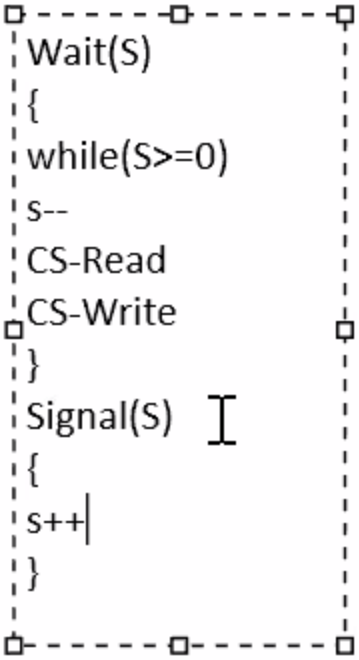
- Wait (P)

- Signal (V)

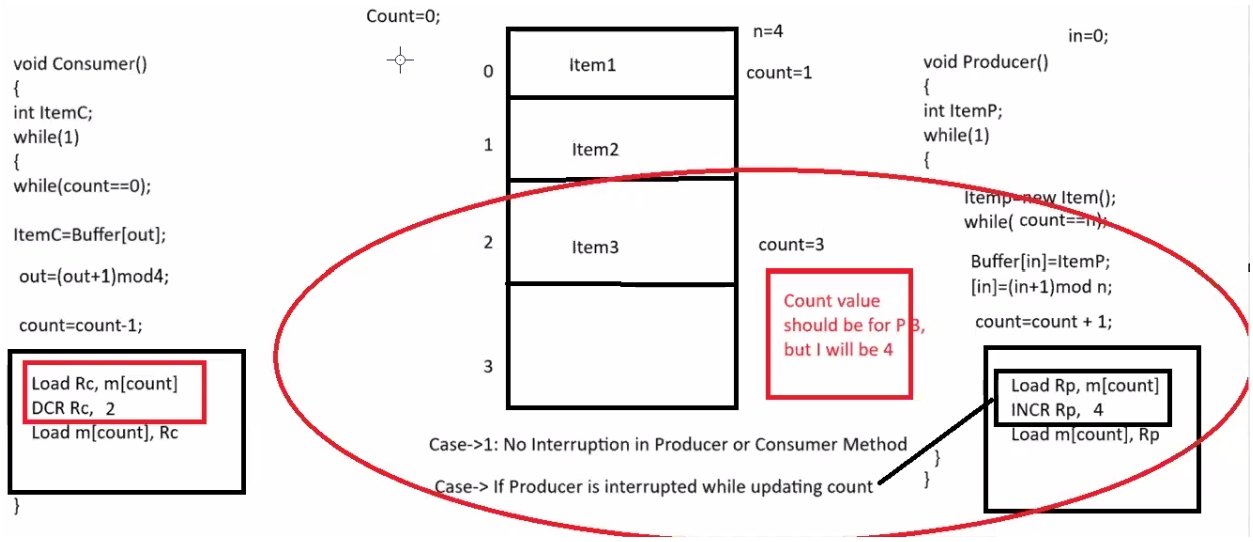
//read about mutex and semaphore difference, binary semaphor and normal semaphor (Important)

Semaphore

S



Producer Consumer Problem:



can be avoided by critical section and process synchronization