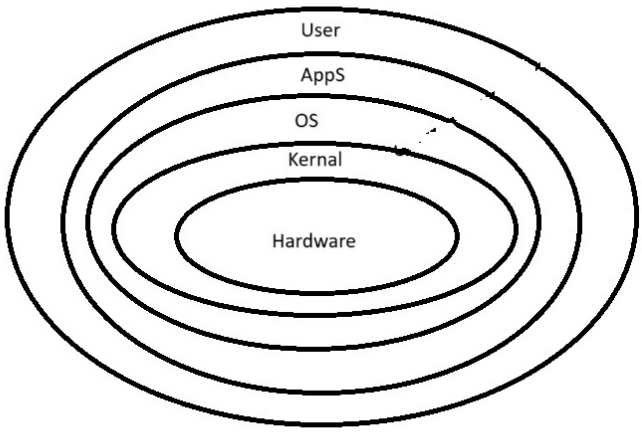
OS Notes Day-1 Date: 24-02-2025

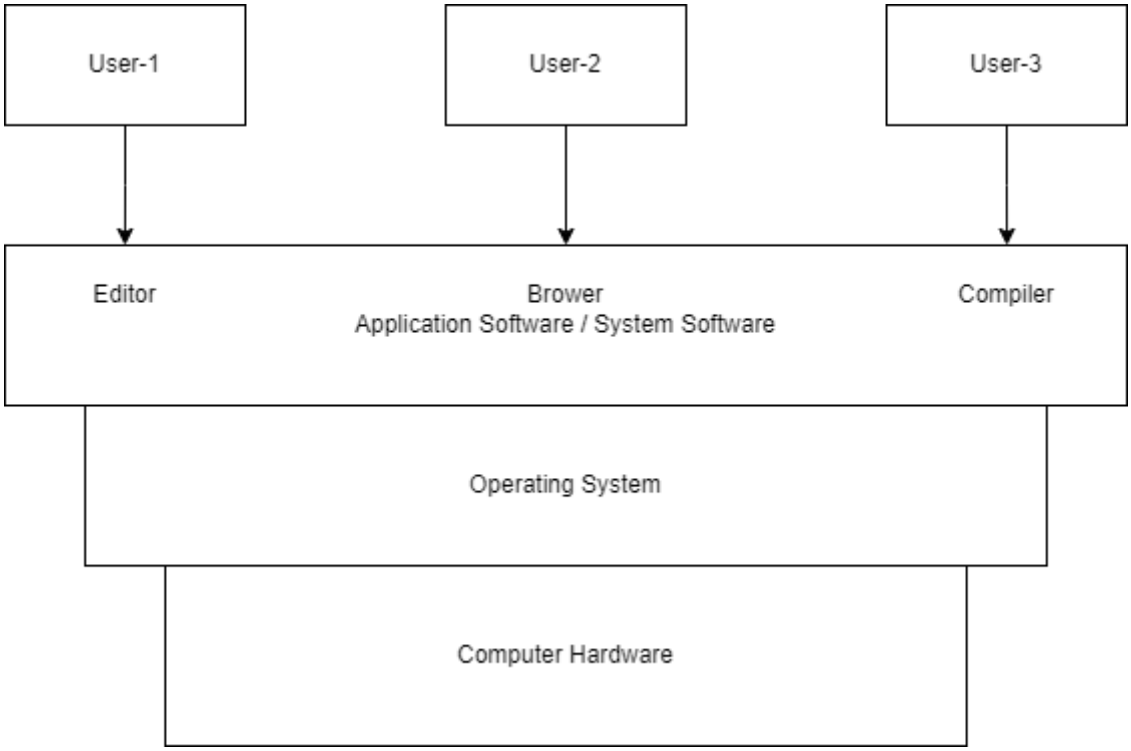
Session-1: Introduction to Operating System

- What is OS?
    - Hardware Manager: It manage all the hardware resources or components of computer.
    - Process Manager: It supervise all the task/process/job which is being executed by processor.
    - It is an Interface between Hardware and User.

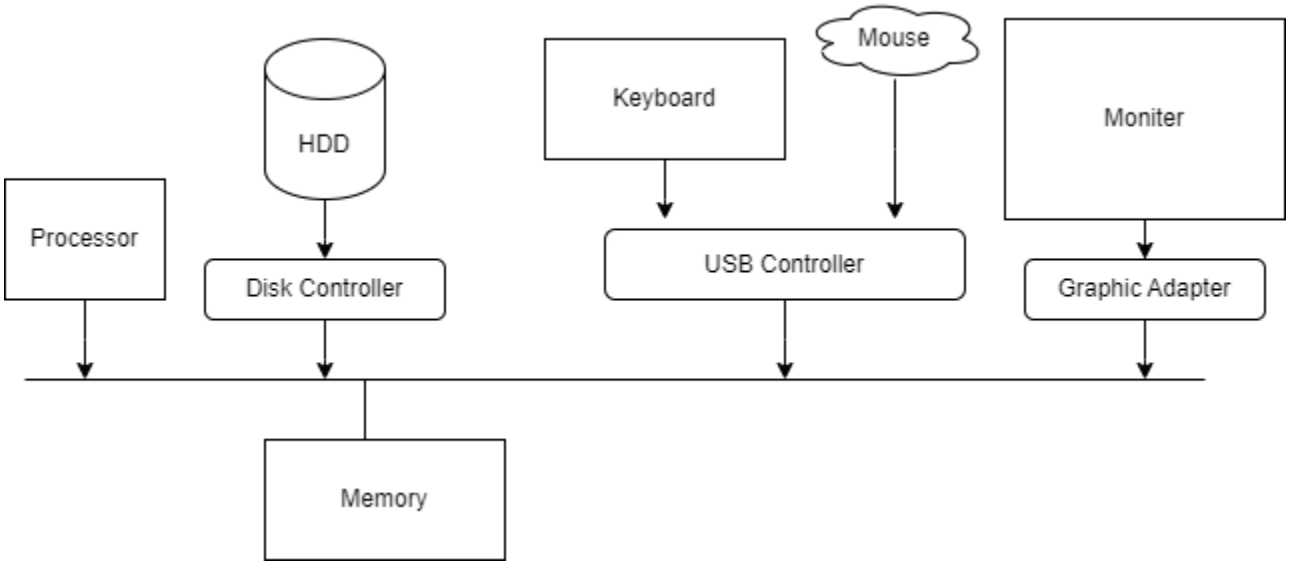


- How is it different from other application software?
    - OS is installed over hard-drive.
    - Applications are also installed over hard-drive but under the layer of OS.
    - OS runs over computer system, and Applications runs over OS.
- Booting: Loading of OS from hard drive to main memory is known as Booting
    - Cold Booting: When we are starting computer system from totally power off condition and OS code loaded into main memory.
    - Hot Booting: When the system is already running and we re-start the system, then firstly RAM is totally cleared and OS code is loaded again into RAM.
- Why OS is hardware dependent?
    - TODO

- Different components of OS

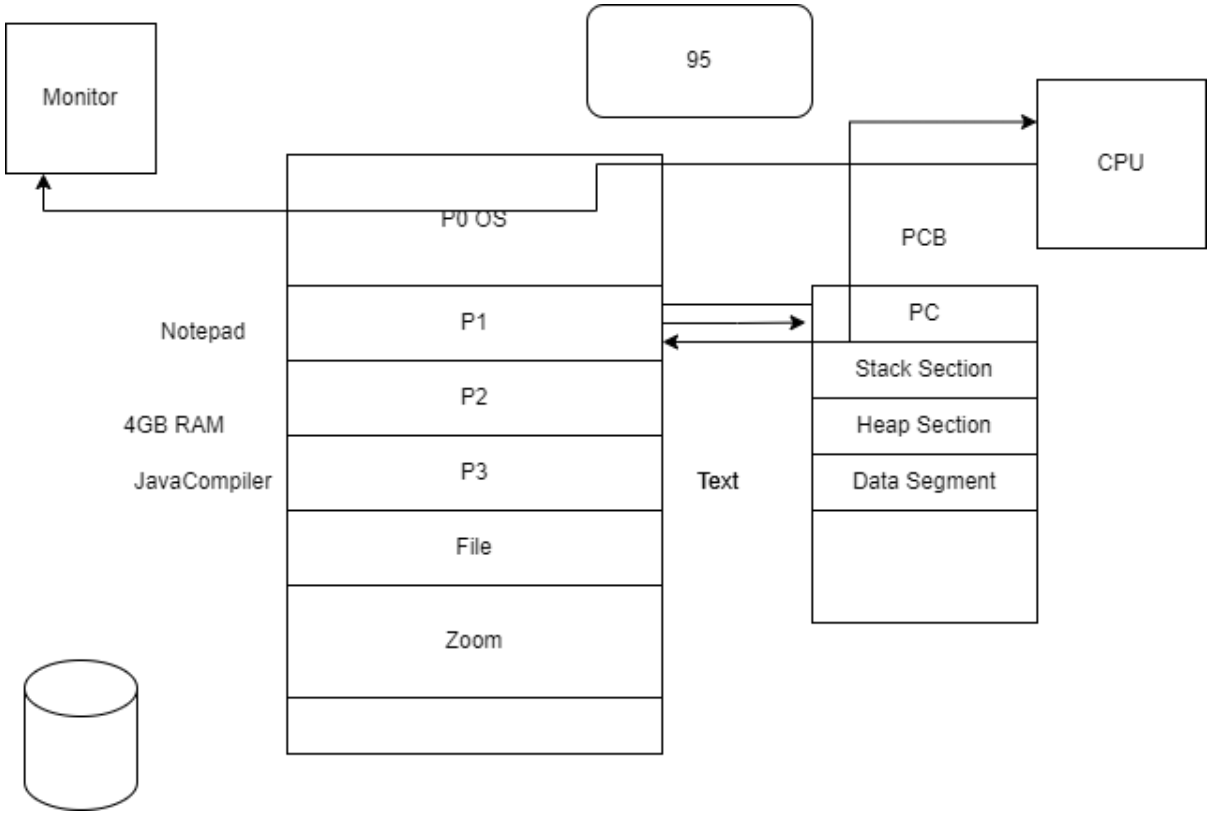


- Basic computer organization required for OS.



- Examples of well-known OS
  - Mobile OS: Android, iOS, Windows
  - Embedded OS:
  - Real Time OS: HRT, SRT
  - Desktop OS: Windosq, MacOS, Chrome OS
  - Sever OS: Cent7 OS, Ubuntu Server
- TO DO:How are these different from each other and why?
- Functions of OS:

- Process Management (Process Scheduling Algo)



- Device Management
- Disk Management (Disk Scheduling Algos)
- File Management
- Network Management
- Security Management (Firewall, Anti Virus, BIOS)

Will be discussed tomorrow

- User and Kernel space and mode;
- Interrupts and system calls
- Memory Hierarchy in Computer System
- Types of Operating System

Session-2: Introduction to Linux

- It is an open source operating system. It is available for the user for free of cost. User can modify its code w.r.t. their need.
- The founder of linux OS is Linus Torvards. It is launched in year 1991.
- An Open Source Community is working behind its continous support and upgradation.
- Feature:
  - No Cost/ Low Cost
  - Multi-Tasking
  - Security
  - Customizable
  - Multi-User
  - Better File System
  - CLI and GUI

- Working basics of file system
    - / it is known root directory
        - /bin: User bineries
        - /sbin: System Bineries
        - /etc: Configuaration files
        - /dev: Device Files
        - /proc: Process Information
        - /var: Variables Files
        - /tmp: Temporaray files
        - /usr: User Directory
        - /Home: Parent directory for user
        - /log:
- Commands associated with files/directories & other basic commands.
- Operators like redirection, pipe

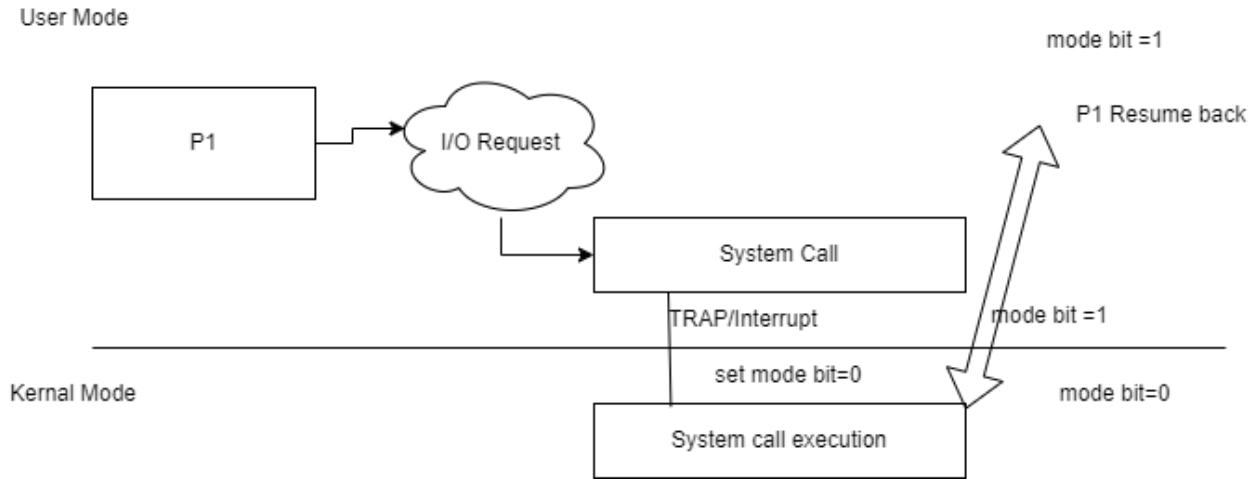
OS Notes Day-2 Date: 26-02-2025

Lets revise Session-1: Introduction to Operating System

- What is OS?
- How is it different from other application software?
- Booting
- Why OS is hardware dependent?
- Different components of OS
- Basic computer organization required for OS.
- Examples of well-known OS
- TO DO:How are these different from each other and why?
- Functions of OS
- Introduction to Linux, Features and Directories Structure

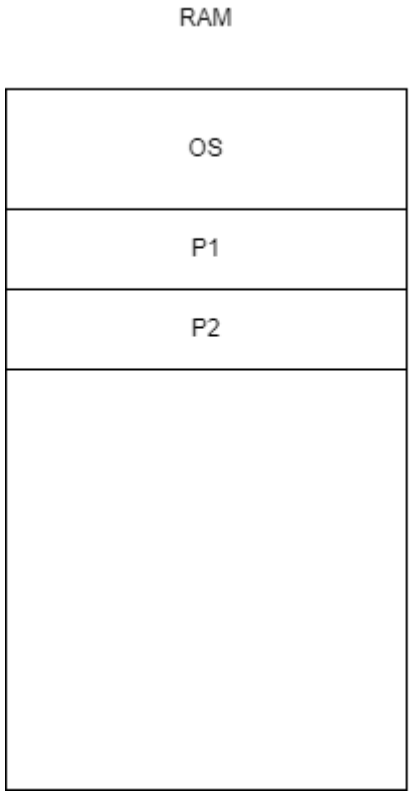
OS Introduction and Basic Functions

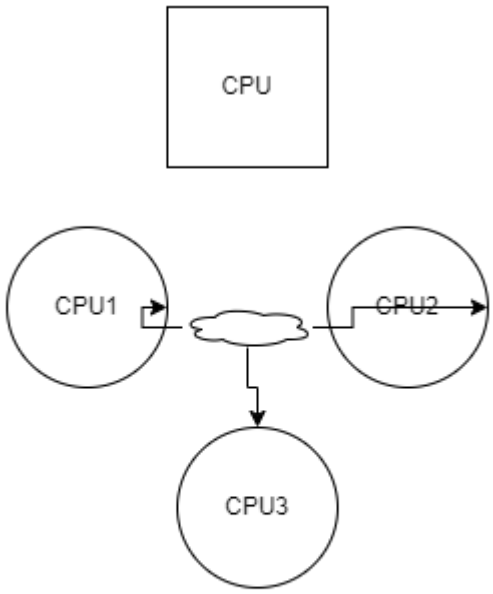
- User and Kernel space and mode



- Interrupts and system calls
    - File Related: Open(), Close(). Read(), Write(), Execute, Delete(), Create()
    - Process Related: New(), fork(), wait(), running()
    - Device Related: read(), uptime, getime etc.
    - Information Related: getpid, getppid, sysdata
    - Communication Related (IPC): wait(), notify(), notifyall()
- Types of Operating System
    - Bathc OS
    - Multi-Programming OS
    - Multi-Tasking OS
    - Multi Processing OS

- Clustered OS



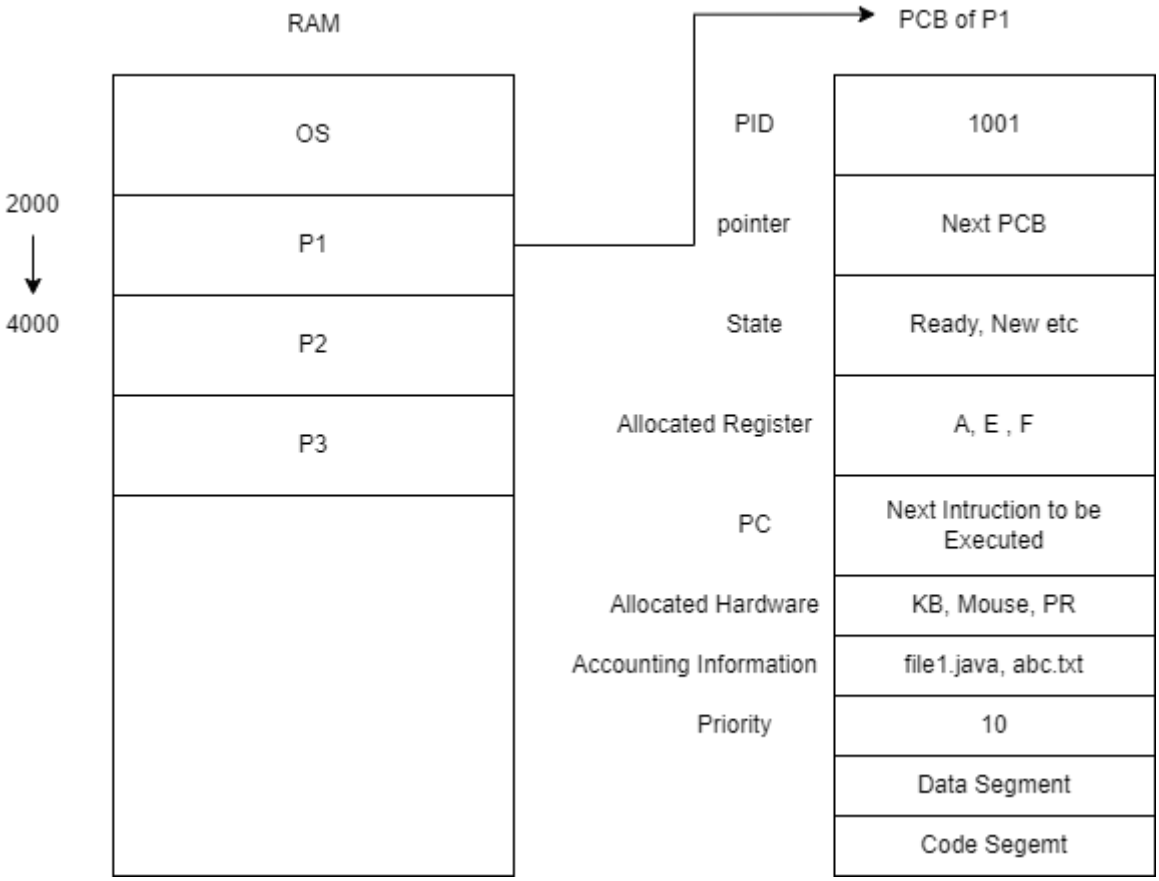


- Distributed OS
- Embedded OS

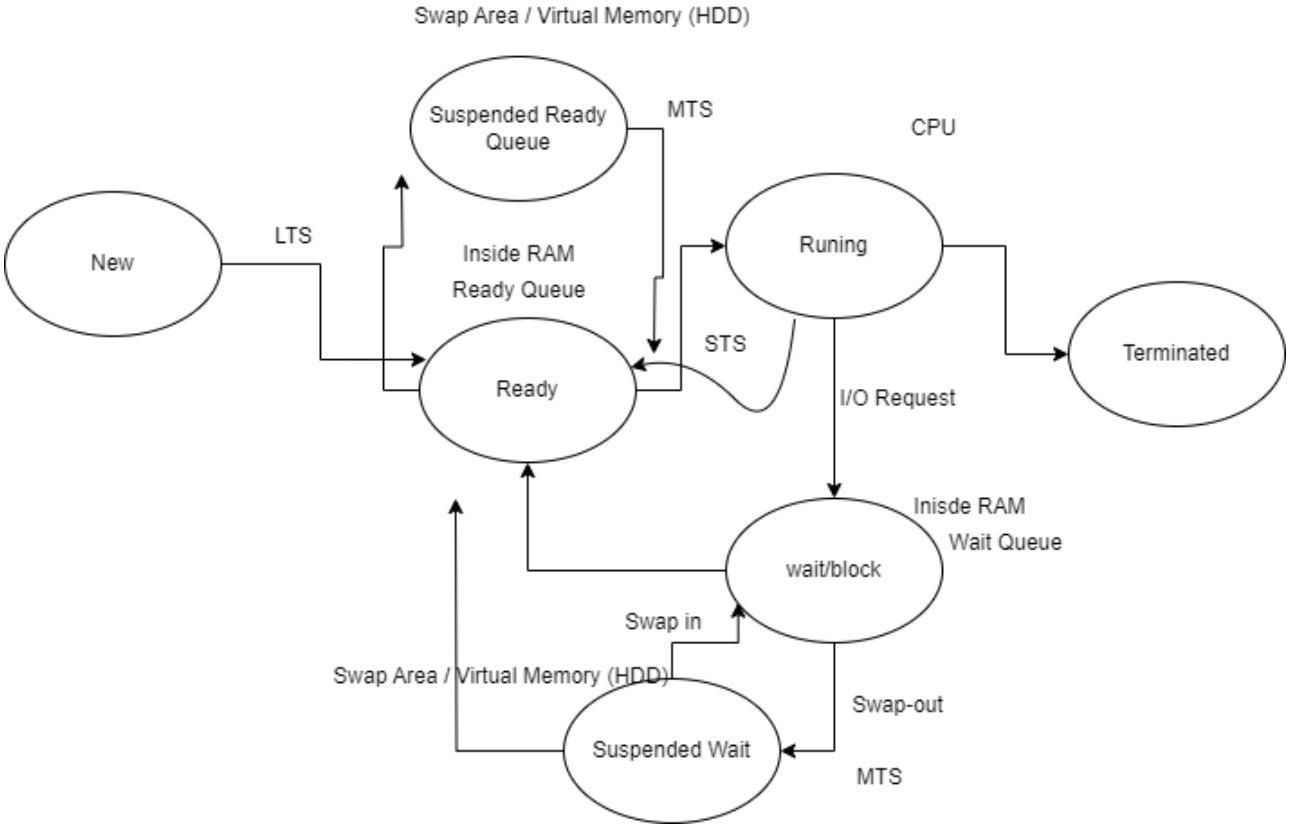
Process Magagement

- What is process: A program / application loaded in RAM is known as process.
  - preemptive: If a process can be interupted while its execution and resumed back later that process is known as pre-emptive process.
  - non-preemptive processes: If a process can not be interupted while its execution that process is known as non pre-emptive process.

- Every process have its control block which is known as Process Control Block



- Process life cycle



- What are schedulers (Please briefly read them)
  - Short Term
  - Medium Term
  - Long Term
- Process scheduling algorithms

- FCFS (First Come First Serve)
  - processes are scheduled according to their arrival time

PID	Arrival Time	Burst Time	Response Time	Waiting Time	TAT			
P1	0	4	0	0	4			
P2	1	6	4	3	9			
P3	2	2	10	8	10			
P4	3	8	12	9	17			
P5	4	10	20	16	26			
			Avg RT	Avg WT	Avg TAT			
		Gantt Chart	P1	P2	P3	P4	P5	
			0	4	10	12	20	30
		FCFS						

To be discussed tomorrow morning (27-02-2025)

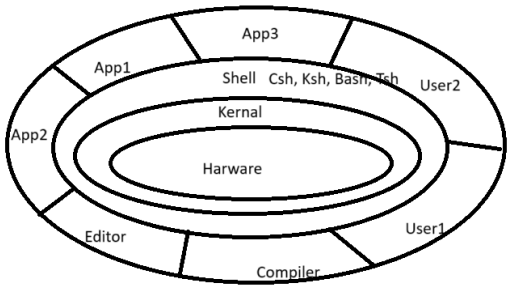
- Memory Hierarchy in Computer System
- Shortest Job First (Shortest Job First)
- Priority ()
- RR (Round Robin):
- Queue.
- Belady's Anomaly
- Examples associated with scheduling algorithms to find turnaround time to find the better performing scheduler.
- Process creation using fork; waitpid and exec system calls; Examples on process creation;
  - Parent and child processes
  - Orphan and zombie processes

Linux and some Usefull commands

Let us revise

- It is an Open Source operating system. It is available free to use and user can modify it according their need.
- The founder or linux is Linun Torvards. It available since 1991.
- An Open Source Community is woking behind the updation and upgradation of the linux code.
- Feature
  1. No Cost / Low Cost
  2. Multi-Tasking
  3. Security
  4. Multi-User
  5. Stable and Scalable
  6. Networking
  7. CLI as well as GUI
  8. Better File System

- Unix/Liux OS Architecture



Layered Archi of Unix/Linux OS

- Working with basics file system of Linux
- / is root directory
    1. /bin: User Bineries
    2. /sbin: System Bineries
    3. /etc: Configuration Files
    4. /dev: Device Files
    5. /proc: Process Information
    6. /var: Variables Files
    7. /tmp: Temporary Files
    8. /usr: User Programs
    9. /home: Parent directory of user friendly directory
    10. /boot: Boot Loader Files
    11. /opt:Apps
    12. /lib: System Libraries
- Commands associated with files/directories

Linux command Format

Command Name	commnad options	Command Argument

Example: cat -n abc.txt

1. pwd: Present Working Directory
2. ls: it list out all the files and directory of current working directory
3. nano: it actually run the nano editor and open the specified file.
4. touch: It is used to create a new file
5. mkdir: To create a new directory.
6. chmod: to give and revoke the file or directory permissions
7. rm: to remove file and recursive directory
8. rmdir: to remove a prticuler directory
9. cd: to change directory

- Ref: https://ubuntu.com/tutorials/command-line-for-beginners#1-overview

- What is shell?
  - Shell is interface b/w user and kernal.
  - It take input from user and pass it on to the kernal.
  - An user can interact with kernal by using shell commands or shell script / program.
- What are different shells in Linux?
  - /bin/sh
  - /bin/bash
  - /usr/bin/bash
  - /bin/rbash
  - /usr/bin/rbash
  - /usr/bin/sh
  - /bin/dash
  - /usr/bin/dash
  - /usr/bin/tmux
  - /usr/bin/screen
- Shell variables
  - Shell varibale can be defined using any name without the type of the variable
- Example:

```
X=100;          //X will store value 100
Y="Malkeet"     //Y will store the value "Malkeet"
echo $X         //To access the variable you should use $ sign before the variable
name
```

- Read: Its is used to read input from keyboard
- Echo: It used to print output something in screen.

To be discussed tomorrow evening (27-02-2025)

- Operators like redirection, pipe
- What are file permissions and how to set them?
- Permissions (chmod, chown, etc)
- access control list

Shell Programming

- Shell variables
- Wildcard symbols
- Shell meta characters
- Command line arguments
- more use of Read, Echo
- Decision loops (if else, test, nested if else, case controls, while…until, for)
- Regular expressions
- Arithmetic expressions
- More examples in Shell Programming

OS Notes Day-3 Date: 27-02-2025

Lets revise

- User and Kernel space and mode
- Interrupts and system calls
    - Hardware Interrupt
        - RST 7.5
    - Software Interrupt (System Calls)
- Types of Operating System
- Process life cycle
- What are schedulers
- Process scheduling algorithms

Process scheduling algorithms

Memory Hierarchy

- Memory hierarchy is a structured arrangement of different storage types used in a computer system. It is designed to balance cost, capacity, and access speed by placing small, fast, and expensive memory close to the CPU and larger, slower, and cheaper memory further away.
- Levels of Memory Hierarchy:
    - Registers:
        - Location: Inside the CPU.
        - Characteristics: Extremely fast and very limited in number.
        - Use: Holds the operands and results of arithmetic operations.
    - Cache Memory:
        - Types: Typically divided into L1 (fastest and smallest), L2 (larger but slightly slower), and sometimes L3 (even larger and shared among cores).
        - Characteristics: Provides a temporary storage area for frequently accessed data to reduce the average time to access data from the main memory.
        - Principle: Operates based on the principles of temporal and spatial locality.
    - Main Memory (RAM):
        - Characteristics: Larger capacity compared to cache but slower in terms of access speed.
        - Use: Holds the bulk of the data and code that the CPU is actively using.
    - Secondary Storage:
        - Examples: Hard Disk Drives (HDDs), Solid-State Drives (SSDs).
        - Characteristics: Much larger in capacity, non-volatile, but significantly slower than RAM.
        - Use: Used for long-term data storage and retrieval.
    - Tertiary/External Storage:
        - Examples: Optical disks, magnetic tapes, and cloud storage systems.
        - Characteristics: Often used for backup, archival purposes, or rarely accessed data.
- Key Takeaways:
    - Cost vs. Speed Trade-off: Faster memories are more expensive per bit, so systems use a combination to optimize performance.

- Locality Principles: Temporal locality (recently accessed data is likely to be accessed again) and spatial locality (data near recently accessed data is likely to be accessed soon) drive efficient memory hierarchy designs.

Process Scheduling Algo

- Shortest Job First (SJF) Scheduling
  - Shortest Job First (SJF) is a scheduling algorithm that selects the process with the smallest execution time (or CPU burst) to run next. It is designed to minimize the average waiting time of processes in the queue.
  - Key Characteristics:
    - Optimality: SJF is proven to be optimal in terms of average waiting time if all processes arrive simultaneously.
  - Types:
    - Non-preemptive SJF: Once a process starts executing, it runs to completion.
    - Preemptive SJF (Shortest Remaining Time First - SRTF): A newly arriving process can preempt the current one if its remaining time is shorter.
  - Drawback:
    - Starvation: Longer processes may wait indefinitely if shorter processes continuously arrive.
- Example-1: Non Preemptive

PID	Arrival Time	Burst Time	Response Time	Waiting Time	TAT		
P1	0	4	5	5	9		
P2	0	2	0	0	2		
P3	0	6	9	9	15		
P4	0	3	2	2	5		
		Gantt Chart	P2	P4	P1	P3	
			0	2	5	9	15

- Example-2: Preemtive

PID	Arrival Time	Burst Time	2nd Second	3rd Second			Response Time	Waiting Time	TAT
P1	0	4	3	3			0	2	6
P2	0	5	5	5			6	6	11
P3	1	2	1	0			1	0	2
P4	2	9	9	9			11	9	18
	SJF with Prem								
		Gantt Chart	P1	P3	P3	P1	P2	P4	
			0	1	2	3	6	11	20

- Priority Scheduling

- Priority scheduling assigns a priority to each process. The CPU is allocated to the process with the highest priority. Priorities can be either static (assigned at creation) or dynamic (changed over time).
- Types:
  - Preemptive Priority Scheduling: A running process can be interrupted if a new process with a higher priority arrives.
  - Non-preemptive Priority Scheduling: Once a process starts executing, it continues until completion even if a higher priority process arrives.
- Potential Issue:
  - Starvation: Lower priority processes may never execute if higher priority processes continue to arrive. This can be mitigated by using aging (gradually increasing the priority of waiting processes).

- Example-1:

PID	Arrival Time	Priority	Burst Time	Response Time	Waiting Time	TAT			
P1	0	7	4	0	10	14			
P2	0	5	6	14	14	20			
P3	1	2	7	20	19	26			
P4	2	10	10	2	0	10			
	Priority	Gantt Chart	P1	P1	P4	P1	P2	P3	
			0	1	2	12	14	20	27

- Round Robin (RR) Scheduling
  - Round Robin (RR) is a pre\emptive scheduling algorithm particularly suitable for time-sharing systems. Each process is assigned a fixed time quantum (or time slice), and processes are cycled through in a circular queue.
  - Key Characteristics:
    - Fairness: Every process gets an equal share of the CPU.
    - Time Quantum: The duration of the time slice is crucial:
    - Too Small: Can lead to high context switching overhead.
    - Too Large: Can make the algorithm behave like First-Come, First-Served (FCFS).
    - Preemptiveness: Processes are preempted if they do not complete within their allotted quantum, and are placed at the end of the queue.
    - Responsiveness: Especially important in interactive systems where users expect a responsive system.

- Example-1:

PID	Arrival Time	Burst Time	Response Time	Waiting Time		TAT		u= 2ms							
P1	0	4	0	6		10									
P2	0	6	2	10		16									
P3	0	2	4	4		6									
P4	0	10	6	12		22									
		Gantt Chart	P1	P2	P3	P4	P1	P2	P4	P2	P4	P4	P4		
			0	2	4	6	8	10	12	14	16	18	20	22	

- Example-2: Reduced Quantum

PID	Arrival Time	Burst Time	Response Time	Waiting Time		TAT		u= 2ms
P1	0	4	0	6		10		
P2	0	5	2	10		15		
P3	0	2	4	4		6		
P4	0	10	6	11		21		

Gantt Chart	P1		P2		P3	P4	P1		P2	P4	P2		P4	P4	P4	
	0		2		4	6	8		10	12	14	15		17	19	21

Linux's Usefull commands, Shell Scripting

- Decision loops in shell scripting / programming
    - if else -Syntax:

```
if [ condition ]
then
    statement
else
    statement
fi (end of if)
```

- Example:

```
echo "Enter Num1"
read Num1
if [ $Num1 -eq 5 ]
then
        echo "Number is equal to 5"
else
        echo "Number is not equal to 5"
fi
```

```
- nested if- else
```

- syntax:

```
if [ condition ]  //Outer-if
then
    if [ condition ]   //Inner-if in if part
    then
        statement
    else //else part of inner-if
        statement
```

```
    fi (end of inner if)
else //else of outer if
    if [ condition ]
    then
        statement
    else //else of inner if
    fi (end of inner if)
fi (end of outer if)
```

- Example: Nested if-else

```
echo Enter Num1
read Num1
echo Enter Num2
read Num2
echo Enter Num3
read Num3
if [ $Num1 -gt $Num2 ]
then
if [ $Num1 -gt $Num3 ]
then
        echo Num1 is greatest
else
        echo Num3 is greatest
fi
else
        if [ $Num2 -gt $Num3 ]
then
        echo Num2 is greatest
else
        echo Num3 is greatest
fi
fi
```

- test command
- Example-1:

```
#!/bin/bash
x=100
y=200
if test $x -eq $y
then
echo x and y are equal
else
echo x and y are not equal
fi
```

- Please try: -eq, -gt, -le, -ge, -lt these operators with test command

- Example-2

```
#!/bin/bash

x=Malkeet
y=Malkeet
if test $x == $y
then
echo x and y are equal
else
echo x and y are not equal
fi
```

- Please try: >, <, ==, empty these operators with test command
- loops in shell programming
    1. for: It is used to repeat certain code upto n no. of time.
- Syntax:

```
a=0
for a in 1 2 3 4 5 6
do
echo $a
done
```

- Example-1

```
#!/bin/bash
a=0
for a in 1 2 3 4 5
do
echo $a
done
```

- Example-2

```
#!/bin/bash
a=0
sum=0
for a in 1 2 3 4 5
do
echo $a
sum=`expr $sum + $a`
done
echo Sum is, $sum
```

To do in the Lab

1. Sum of n Odd numbers
2. Sum of n even numbers
3. To find whether number is a prime or not
4. Fibbonacci series upto n numbers
5. Factorial of number
6. Table of a number

To be discussed tomorrow (28-02-2025)

Memory Management

- Continuous and Dynamic allocation
- First Fit, Best Fit, worst Fit
- Compaction
- Internal and external fragmentation
- Paging:
    - What is paging?
    - hardware required for paging
    - paging table
    - Translation look aside buffer
- Concept of dirty bit
- Shared pages and reentrant code
- Throttling
- IO management
- Virtual Memory
    - What is virtual memory
    - Demand paging
    - Page faults
    - Page replacement algorithms
    - Belady's Anomly
- Segmentation:
    - What is segmentation?
    - Hardware requirement for segmentation.
    - segmentation table and its interpretation

Linux and Shell Programming

- Permissions (chmod, chown, etc)
- access control list
- Shell variables
- Wildcard symbols
- Shell meta characters
- Command line arguments

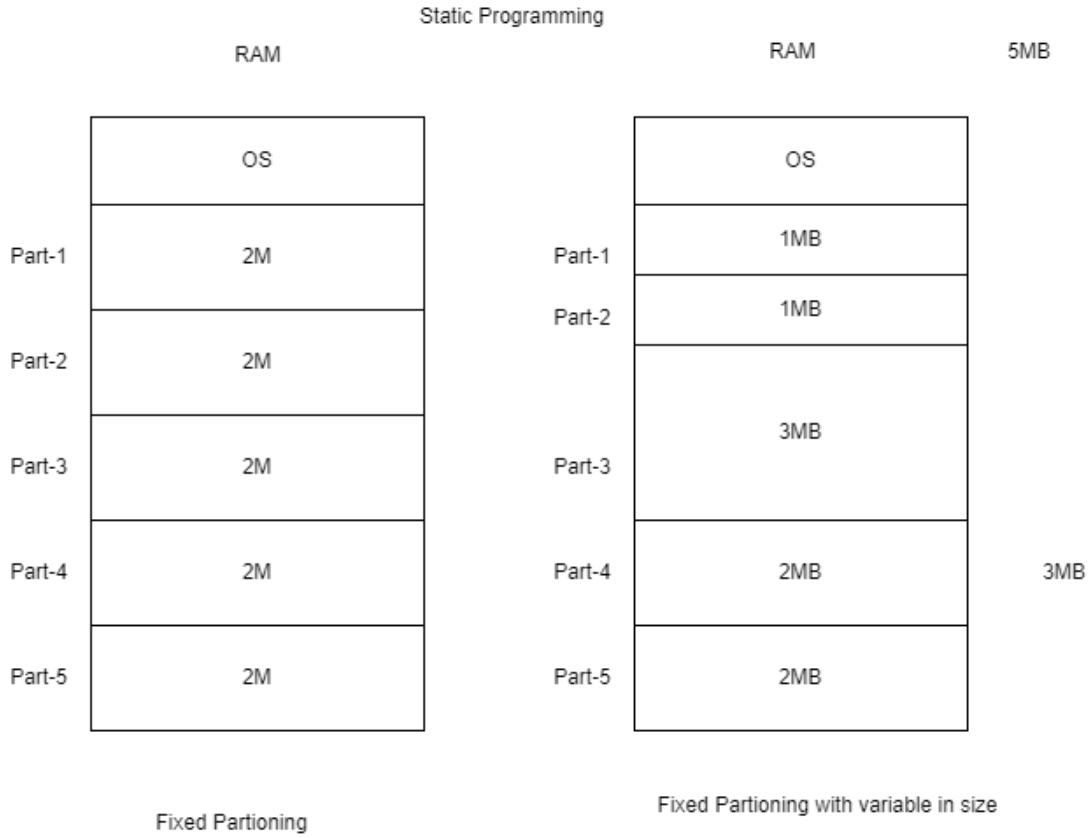
OS Notes Day-4 Date: 28-02-2025

Lets revise

- Memory Hierarchy
- Process Scheduling Algo
    - Shortest Job First (SJF) Scheduling (Preemptive and Non Preemptive)
    - Priority Scheduling
    - Round Robin (RR) Scheduling
    - Redirection (>,>>) and Piping (|) in linux shell commands
    - Control statements in Shell Scripting e.g. if-else, nested if-else, for loop

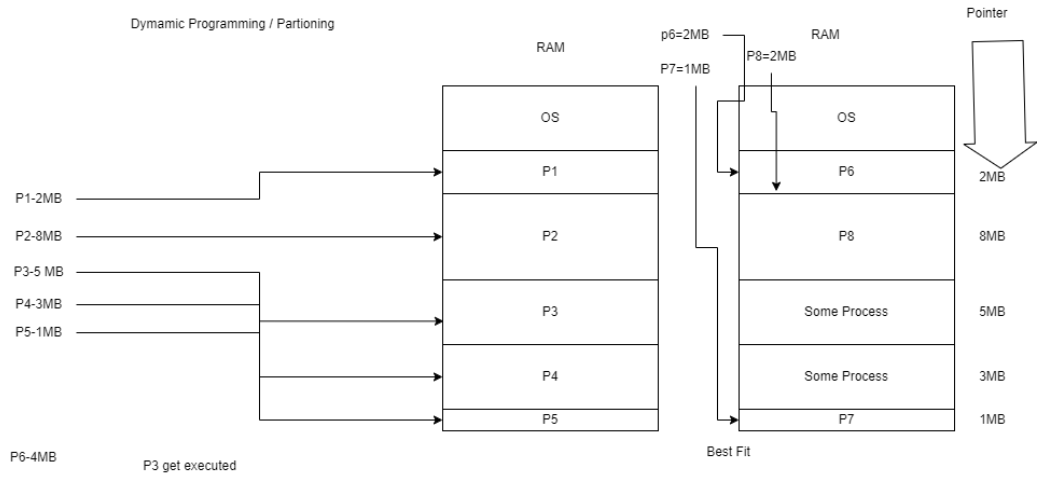
Memory Management

- Static Partitioning: Early systems used fixed-sized partitions, where each process occupied one contiguous block.
    - Drawbacks:
        - Can lead to external fragmentation.
        - Processes must be loaded into contiguous spaces, which can be inefficient if free memory is scattered.



- First Fit, Best Fit, Worst Fit
    - These are strategies used by the operating system to allocate free memory blocks to processes:
    - First Fit:
        - Mechanism: The allocator scans the memory from the beginning and chooses the first free block that is large enough for the process.
        - Pros:

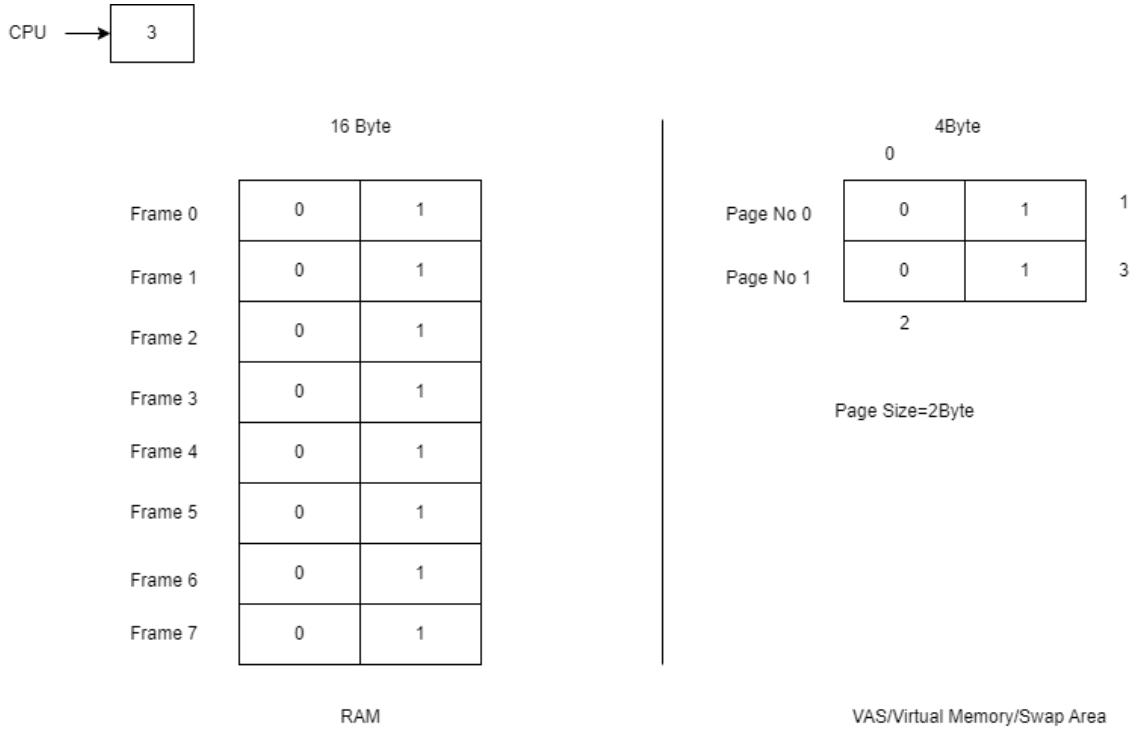
        - Simple and fast allocation.
    - Cons:
        - May leave small unusable gaps (external fragmentation) near the beginning of memory.
    - Example:
        - Suppose free blocks are of sizes 50, 200, 100, and a process requires 75 units. First Fit would allocate the 200-unit block if it is the first encountered that fits (if the 50-unit block is too small).



- Best Fit:
    - Mechanism: The allocator searches the entire free memory list and chooses the smallest block that is large enough to satisfy the request.
    - Pros:
        - Minimizes wasted space within the allocated block.
    - Cons:
        - Can be slower due to searching the entire list.
        - May leave many very small fragments.
    - Example:
        - With free blocks of sizes 50, 200, 100 and a request of 75 units, Best Fit would choose the 100-unit block, as it's the smallest block that fits.
- Worst Fit:
    - Mechanism: The allocator chooses the largest available block, under the assumption that splitting a large block leaves a large remainder that might be more useful later.
    - Pros:
        - Tends to leave a larger remainder, which could be useful for future requests.
    - Cons:
        - Can quickly reduce the size of the largest block available.
        - Not as commonly used in modern systems.
    - Example:
        - Given free blocks of sizes 50, 200, 100 for a 75-unit request, Worst Fit would choose the 200-unit block, splitting it into a 75-unit allocation and leaving a 125-unit free block.

- Internal and External Fragmentation
    - Internal Fragmentation:
        - Definition: Occurs when allocated memory may have some unused space within it, often because memory is allocated in fixed-sized blocks.

        - Example:
            - If a process requires 70 bytes and the allocation unit is 100 bytes, 30 bytes are wasted within that block.
    - External Fragmentation:
        - Definition: Occurs when free memory is split into small blocks scattered throughout memory. Even if the total free memory is sufficient, there may be no single contiguous block large enough to satisfy a request.
        - Example:
            - Memory has free blocks of sizes 30, 20, and 40 bytes scattered between processes. A request for 50 bytes cannot be satisfied even though 90 bytes are free in total.
- Compaction/Defragmentation
    - Compaction is the process of rearranging the contents of memory to place all free memory together in one large block. This is typically done to overcome external fragmentation.
    - Process:
        - The operating system shifts processes in memory so that all allocated memory becomes contiguous.
        - The free spaces merge into a single block that can be used to satisfy larger allocation requests.
    - Compaction is time-consuming and may require the relocation of processes (updating pointers, tables, etc.), so it is used sparingly.
- 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 accoring to frames available and other pages loaded from storage device on demand of CPU.



- Paging table: It is a special table maintained by MMU (Memory Management Unit) to map the logical address of the demanded page with the physical address of the page in main memory where actualy the page is placed.
- Demand paging

    - If the page is loaded in memory frame as per the demand of CPU i.e. known as demand paging.
- Page faults: If CPU demands a page and its not there in main memory. That time an interrupt generated to load the page from virtual memory / hard-drive.
- Page replacement algorithms
    1. FIFO (First in First Out)

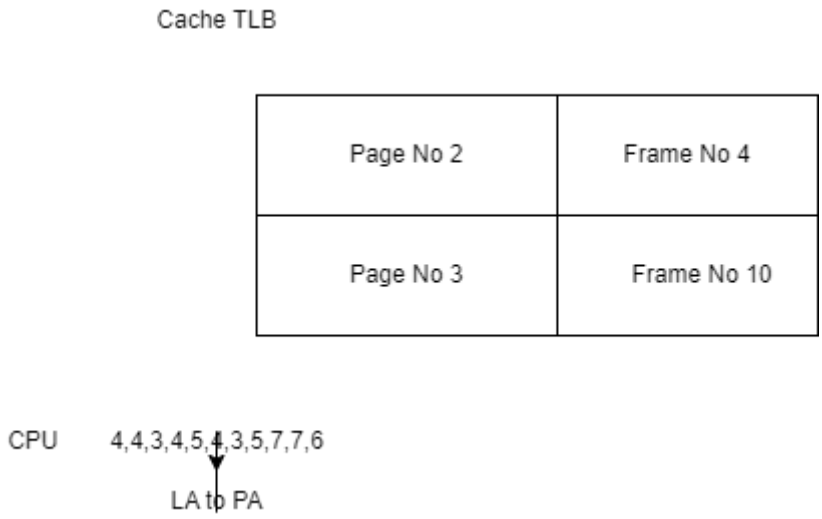
Frame no 2			2	2	2	2	2	0	0	0
Frame no 1		3	3	3	3	3	7	7	7	2
Frame no 0	0	0	0	0	0	5	5	5	3	3
	MISS	MISS	MISS	HIT	HIT	MISS	MISS	MISS	MISS	MISS
Input String	0	3	2	0	2	5	7	0	3	2
					MISS	8				
					HIT	2				

    - Belady's Anomly: It state that if we increase the number of frames, the HIT Ratio decreased.
    2. LRU (Optimal Page Replacement): to be disscussed
    3. MRU (LIFO)

Frame no 2			2	7	2	5	7	7	7	7
Frame no 1		3	3	3	3	3	3	3	3	3
Frame no 0	0	0	0	0	0	0	0	0	0	0
	MISS	MISS	MISS	MISS	MISS	MISS	MISS	HIT	HIT	HIT
nput String	0	3	2	7	2	5	7	0	3	0
					MISS	7				
					HIT	3				

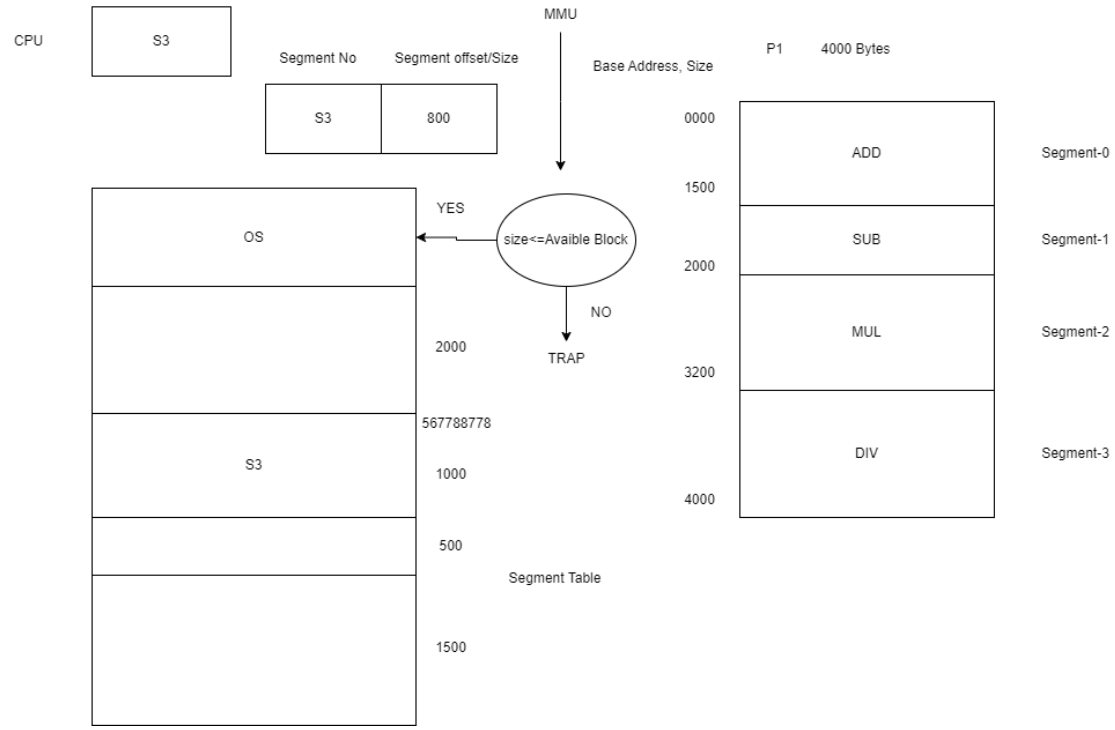
- Hardware required for paging (Virtual + Cache)
- What is virtual memory
    - Its a memory space in hard-drive, which work like physical memory to entertain large processes whose size is bigger than the RAM. It is an illusion memory.
    - In Virtual Memory process is divided into fixed sized partions know as pages.
    - OS loads the process's pages from virtual memory 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.
    - While loading the pages from virtual memory to physical memory the pages get physical address.
    - If a page loaded from virtual memory to physical memory i.e. know as swap-in process.
    - If the page is shifted from physical memory to logical memory by the replacement process i.e. know swap out.
- Translation look aside buffer

- It is a page table represented in cache memory to fasten the process of finding physical address



  for a given logical address.
  - In paging we load only few pages in main memory of a particuler process. Entries of that pages is made in MMU Page Table which is actually present at physical memory. To find a frame MMU first look into page table and then according page table entry it accress the given frame no. This process is actually access the RAM twice. So to reduce this access OS use Translation Look Aside Buffer in cache memory.
- Concept of dirty bit
  - It is a bit in page table which describe either the page content is updated during the execution or not. If for a specific frame the content is updated then its dirty is set to 1, other wise its always 0.
- What is Segmentation?
  - Segmentation divides a program's memory into logical segments such as code, data, and stack. Each segment is a variable-sized block and is treated as a separate unit.
  - Benefits:
    - Allows for logical organization and protection of different types of data.
    - Segments can be shared between processes if needed.
  - Hardware Requirements for Segmentation
    - Segmentation Registers:
      - The CPU includes special registers (such as the segment base and limit registers) that hold information about the segments.
    - Memory Management Unit (MMU):
      - The MMU uses the segmentation registers to translate logical addresses (segment:offset) into physical addresses.
    - Protection and Limit Checks:
      - Hardware supports boundary checking using the segment limit, ensuring that processes cannot access memory outside their assigned segments.
- Segmentation Table and Its Interpretation
  - A segmentation table (or segment descriptor table) contains entries for each segment, detailing the base address, length (limit), and access rights.
  - Typical Entry Contains:
    - Segment Number: Identifier for the segment.
    - Base Address: The starting physical address of the segment.
    - Limit: The size of the segment (defines the maximum offset within the segment).

- Access Rights/Attributes: Read, write, execute permissions.



- Example Interpretation:
  - Suppose a segmentation table entry for a code segment shows a base address of 0x1000, a limit of 0x0FFF, and execute permission. A logical address given as segment:offset is translated by adding the offset to 0x1000 after verifying that the offset does not exceed 0x0FFF.

Linux and Shell Programming

- TODO: Kindly explore more about Permissions (chmod, chown, etc)
- while loop in shell scripting
- Syntax:

```
while [ condition ]
do
statement
increment
done
```

- Example-1

```
#!/bin/bash
a=0
while [ $a -lt 10 ]
do
echo $a
a=`expr $a + 1`
done
```

- until loop in shell scripting
- Syntax:

```
a=0
until [ $a -gt 10 ]
do
statement
increment expresion
done
```

- Example-1

```
#!/bin/bash
a=0
until [ $a -gt 10 ]
do
echo $a
a=`expr $a + 1`
done
```

- Wildcard symbols
    1. Question Mark ?
    2. Asterisk *
    3. Square Braces []
    4. Curly Braces {}
    5. Sign of Exclamation [!]
    6. Backslash \
- Ref: https://tldp.org/LDP/GNU-Linux-Tools-Summary/html/x11655.htm
- Shell meta characters (> , >> , < , | , ; , && , '' , "" , /)
    1. Pipe (|): Piping allow us to filteartion among the result of the command. e.g. cat file2 | grep ^h
    2. Redirection (>): It allow you to use output of one command as an input of other command
    3. Asterisk (*)
    4. Tilde (~): Used for going back to the user's home directory
    5. Doller Symbol ($): It is aslo used to matching the end of the string for a line in a file.
    6. Caret (^): It is used to match the starting of the string withinggiven file
- Ref: https://opensource.com/article/22/2/metacharacters-linux
- Command line arguments
    - Special Parameters in Shell while passing command line arguments
        1. $#: No. of parameters passed
        2. $0: Script Name
        3. $i: where i can be any number, it will return that parameter
        4. $*: Give all parameters passed

```
#!/bin/bash
echo Number of Param, $#
```

```
echo Script Name, $0
echo Hello, $1
echo Hii, $2
echo Ok, $3
echo Bye, $*
```

- Regular expressions
    - | grep ^S: It will filter the pipeline and get the words which are starting with S
    - | grep ^n.*f$: It will filter the pipeline and get the lines which are starting with n and ending f
    - ls f*: It will return all the file starting with f
- Arithmetic expressions
    - $a + $b
    - $a - $b
    - $a / $b
    - $a * $b

```
#!/bin/bash
echo Enter a Number
read Num1
echo Enter a Number
read Num2
Res=`expr $Num1 \* $Num2`
echo Result, $Res
```

- Network Commands
    - telenet: It is used to take the control of remote server / machine
    - ftp: File Transfer Protocol, It allow user to send or recieve file from remote server / machine
    - ssh: It is also used to take control of remote server / machine
    - sftp: Secure FTP
    - finger:

Reading Assigment

- waitpid and exec system calls
- Shared pages and reentrant code
- Throttling
- access control list
- network commands (telenet, ftp, ssh, sftp, finger)
- System variables like – PS1, PS2 etc. How to set them

To be disscussed tomorrow (28-02-2025)

- Deadlock
    - Necessary conditions of deadlock
    - Deadlock prevention and avoidance
    - Semaphore

- Mutex
- Producer consumer problem
- Dead-lock vs Starvation