Date : / /
Page No.

UNIT-1

→ OS is an interface b/w hardware and user
→ It is a System program.
→ Manages System resources
→ Provides platform for other applications
→ Core part of OS is Kernel

Primary Goals : To make ~~make~~ User Sys. friendly

Secondary Goal : Make Sys. efficient

Functions :

→ Process Management
- Scheduling Processes & threads on CPU
- Creating & deleting both user & Sys. process
- Process Synch.
- Process Communication

→ Memory Management
- keeping track of memory usage
- Deciding which process & data move in/out of memory.
- Allocating / De-allocating memory space

→ Disk Management
- Free space mgmt. in disk
- Disk Scheduling
- Storage alloc.

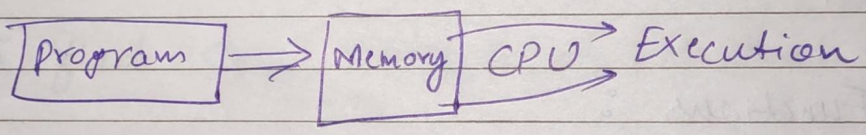
Date : / /
Page No.

→ File Mgmt.
- Create/Delete files
- Backing up files on stable storage media
- Create/Delete Directories

→ I/O Device Mgmt.
→ Network Mgmt.
→ Security and mgmt.

Services :

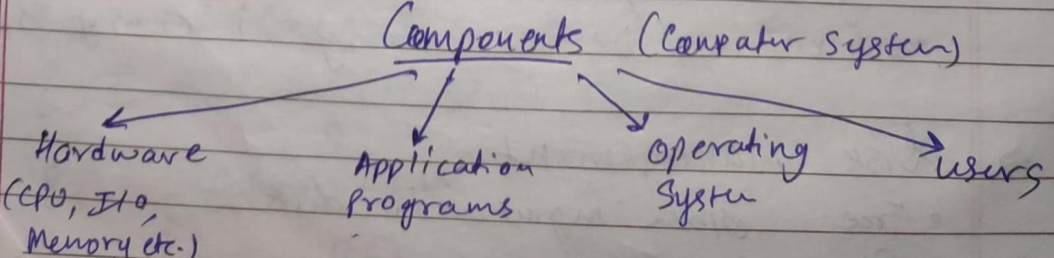
→ User Interface
→ Program Execution



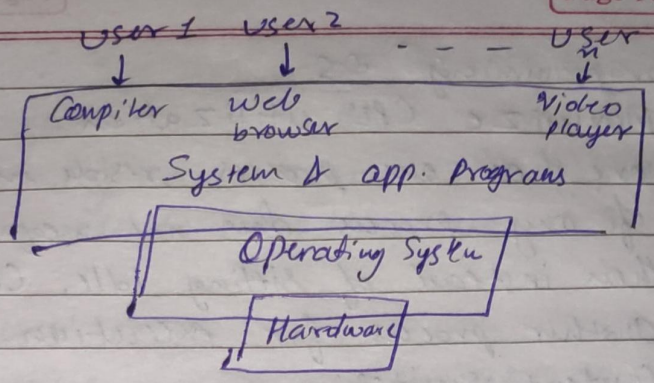
→ Access I/O Devices
→ File system access
- Read/write files & directories
- Search for a file in it
- Access control to files

→ Error Detection & Response
- Troubleshooting
- Error codes

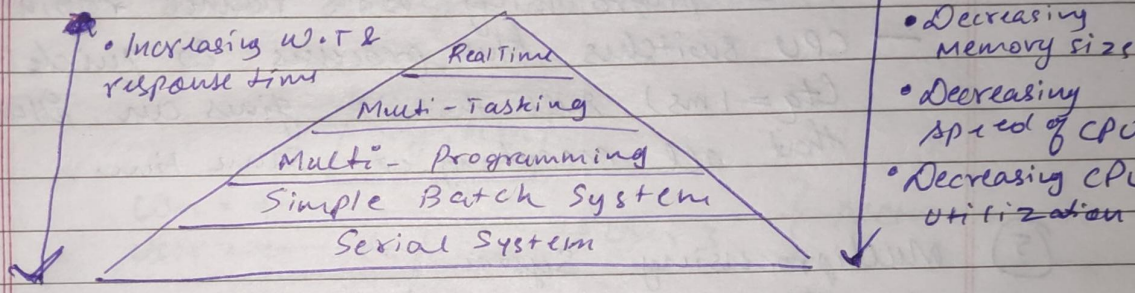
→ Communication



Date : / /
Page No.



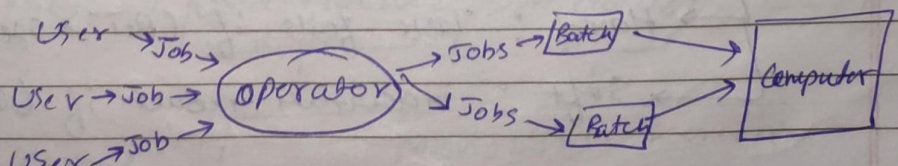
Classification of OS :



(1) Serial System
- No O.S needed
- Direct interaction of programs with hardware

(2) Batch System
- Set of JOBS with similar need.
  - Program
  - Input Data
  - Control Instructions

- Similar types of jobs batch together



Date : / /
Page No.

(3) Multiprogramming OS
- Maximize CPU utilization
- More than one process reside in M/M
- If any process does not require CPU then instead of sitting idle, CPU picks another process for execution.
- Context switching

(4) Time sharing / Multi-Tasking
- Multi-programming with round robin system.
- CPU switches b/w processes so quickly (tq = 1 ms) such that it gives an illusion that all executing at same time.

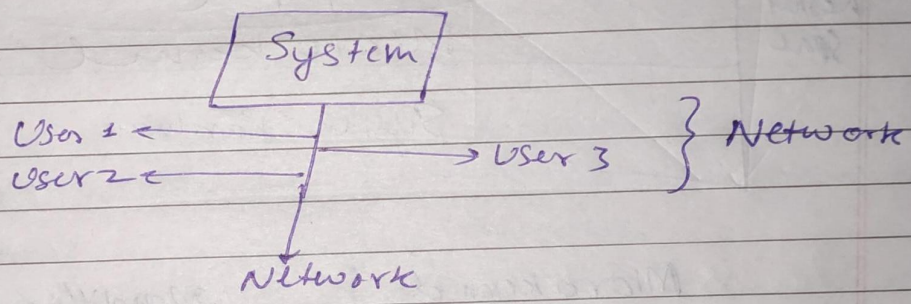
(5) Multiprocessing System
- Two or more CPU within a single comp. share system bus and I/O devices.
- True parallel execution of processes.
- Two types
  - Asymmetric : Master-slave
  - Symmetric : No master-slave

(6) Real Time System
- Well-defined fixed time constraints.
- Used when we require quick response against input.
- Two Types
  - Hard : Task fails if (R.T > specified time)
  - Soft : inaccurate result if (RT > specified time)

Date : / /
Page No.

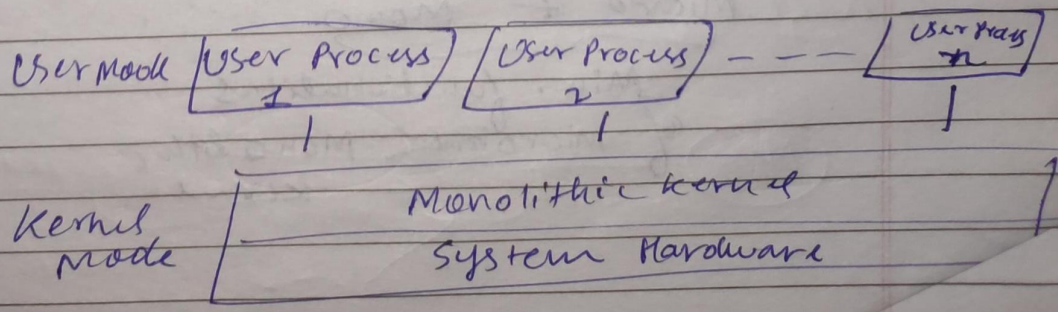
(7) Interactive System
- Direct comm. btw user & system.
- user gives ins. to OS or program (Input)
- Immediate results (output)

(8) Multi-user O.S
- Multiple users on different computers to access a single system with one OS on it.
- Access through a network



Kernels : Set of necessary software/programs modules which drives the hardware.
Types →

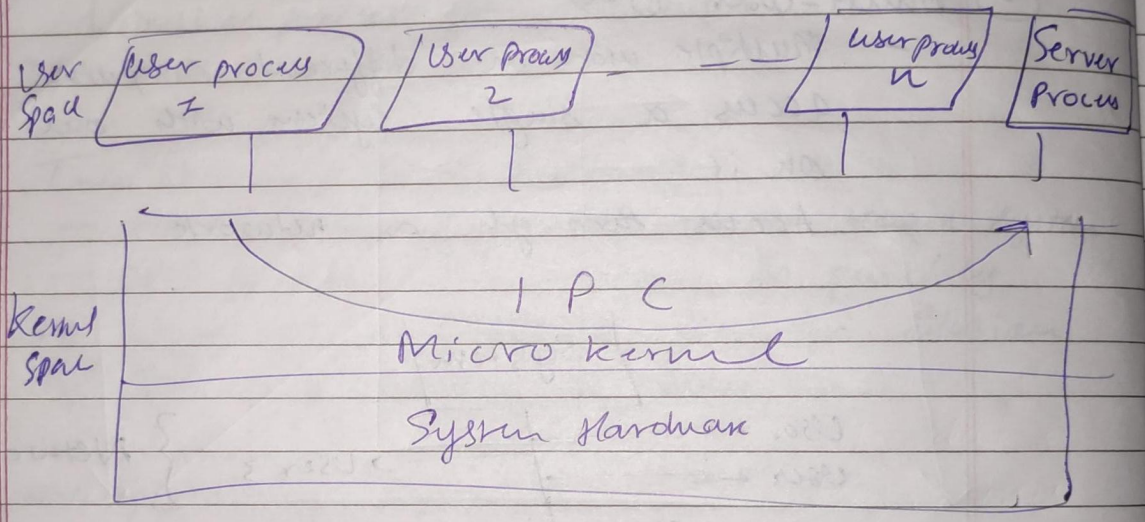
(1) Monolithic
- Literally means all in one place.
- User and kernel services under same address space.



Date : / /
Page No.

(2) Micro kernel

- Different address space for user and kernel services implementation
- Only most imp. services present in kernel



	Microkernel	Monolithic kernel
Size	Small	Large
Speed	Faster	Slower
Extendibility	Easy	Hard
Error prone	Less	More

(3) Hybrid kernel

Micro + Mono

Mix functionalities of microkernel & Monolithic kernel

Date : / /
Page No.

(4) Reentrant kernel

- Executable code stored in memory such that several processes can use this code at same time without hindering the working of other processes.
- Multiple processes executed in kernel mode at same time.
- Example : UNIX.

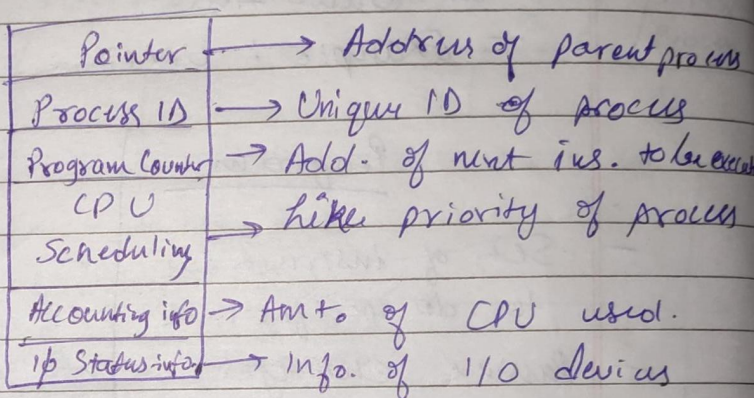
Program	Process
- Set of instructions to do specific task	- Instance of an executing program
- Passive entity	- Active entity
- Resides in secondary memory	- Resides in primary memory.
- continues to exist until deleted	- Limited span of time until completion of time.

Date : / /
Page No.

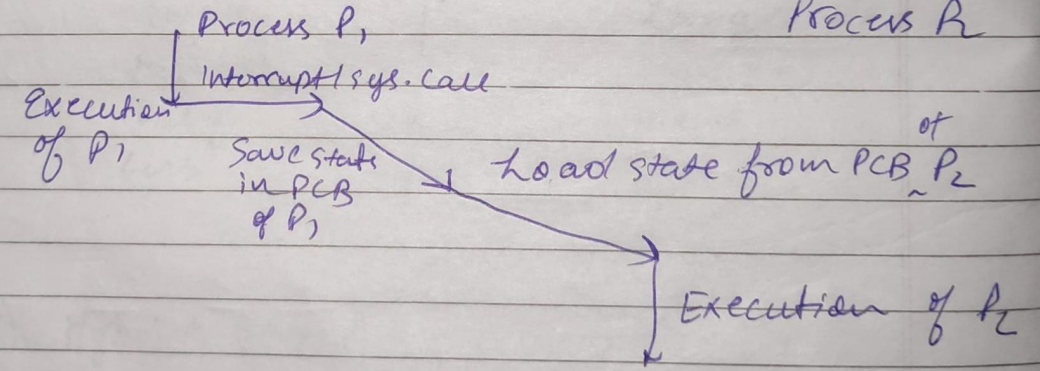
UNIT-3

Process Control Block :

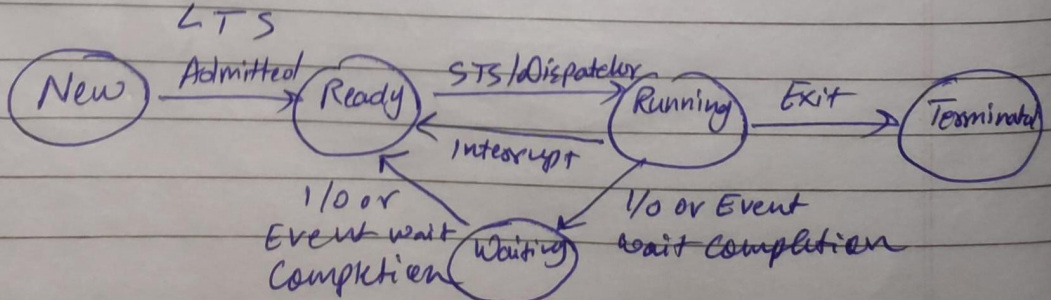
- Holds all info necessary to keep track of process
- OS creates PCB for every process



Process State Switching :



Life Cycle of Process :



Date : / /
Page No.

- New state: Process is created (Job queue)
- Ready state: Resides in Ready queue (Primary Memory)
- Running state: ~~Assigned~~ CPU is assigned to a process
- Terminated state: Process has completed its execution successfully
- Waiting state: Resides in waiting queue (Primary Memory) → I/O burst.

Schedulers:

- LTS
  - Selects process from Job Queue → Ready Queue
  - New state → Ready state
- STS
  - Selects process Ready Queue → CPU
  - Ready State → Running state
  - Dispatcher Saves content of PCB and loads content of another process. (Content Switching)
- MTS
  - Removes processes from MM.
  - ★ Reduces degree of multiprogramming

Multilevel Queue Scheduling:

- Partitions ready queue into separate queue
- Each queue has its own scheduling algorithm.

Date : / /
Page No.

* Aging :

Gradually increasing the priority of the process that wait in the System for long time

Multilevel Feedback Queue Scheduling :

- Allows a process to move between queues.
- If process uses too much CPU time it will be moved towards lower priority queue
- If it stayed too long in lower priority queue, it may be moved to higher priority queue.

Fork() System call :

→ used to create new process (child process)
→ If we call fork() n-times, $2^{(n-1)}$ child process will be generated.

Deadlock :

Def→ Sit Situation where two or more than 2 processes are waiting for each other to complete the execution and release the resource so that requesting processes will get resources for successful execution.

But, none of processes get executed.

Date : / /
Page No.

→ Mutual Exclusion
→ Hold & Wait
→ No Preemption
→ Circular Wait

Deadlock Recovery :

→ Abort all processes involved in deadlock
→ Abort processes one by one to make system deadlock free.

8.

Process	Allocated R_1 R_2 R_3	MAX R_1 R_2 R_3	AVAILABLE R_1 R_2 R_3	NEED R_1 R_2 R_3
P1	2 2 3	3 6 8	7 7 10	1 4 5
P2	2 0 3	4 3 3	9 9 13	2 3 0
P3	1 2 4	3 4 4	11 9 16	2 2 0
			12 11 20	

$[P_1 \longrightarrow P_2 \longrightarrow P_3]$

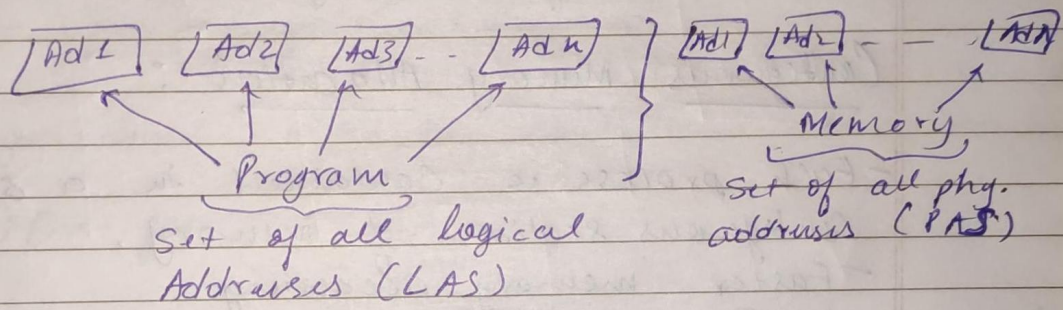
SAFE SEQUENCE ✓✓ Safe state

Date : / /
Page No.

UNIT - 4

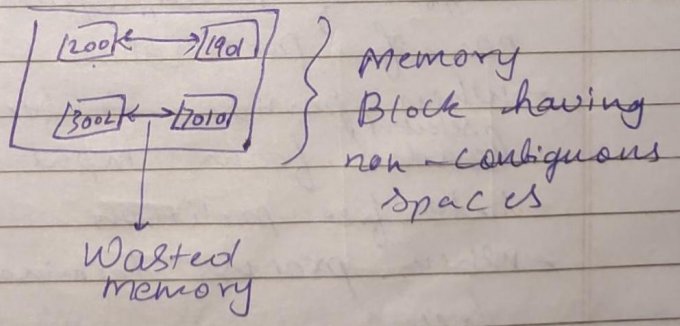
LAS V/s PAS :

→ Logical Address is generated by CPU
→ Physical Address is generated by Memory Unit.



Fragmentation :

① External
- Enough total memory to satisfy a req. but available spaces are not contiguous.



② Internal
- Memory allocated to a process may be larger than req. memory.
- Difference b/w these 2 nos. is Internal frag.

Date : / /
Page No.

Compaction →

Shuffle ~~all~~ Memory contents to place all free memory together in one large block.

↳ To solve external fragmentation

↳ we can move all holes to one side/ one end of memory.

Contiguous Memory Allocation :

- Each process is contained in a single contiguous section of memory.
- Faster memory accessing
- Two Categories :

① Multiprogramming with fixed no. of tasks

- Fixed sized partitions
- Each partition may contain one process
- Degree of multiprogramming is bound by no. of partitions
- when partion is free, a process is ~~loaded~~ selected from input queue and loaded into free partition.
- when process terminates, partition becomes free.
- Prone to Internal, External fragmentation.

Date : / /
Page No.

(2) Multiprogramming with variable no. of tasks

- OS keeps a table indicating which parts of memory are available / occupied.
- Initially all memory is one large hole
- As process enter, put into input queue and OS checks memory requirements of each process and allocates memory.
- Ext. fragmentation occur.

Algorithms used in Contigious memory --

(1) First fit

Allocate the first hole that is big enough

(2) Best fit

Allocate the ~~Smallest~~ smallest hole big enough

(3) Worst fit

Allocate the largest hole.

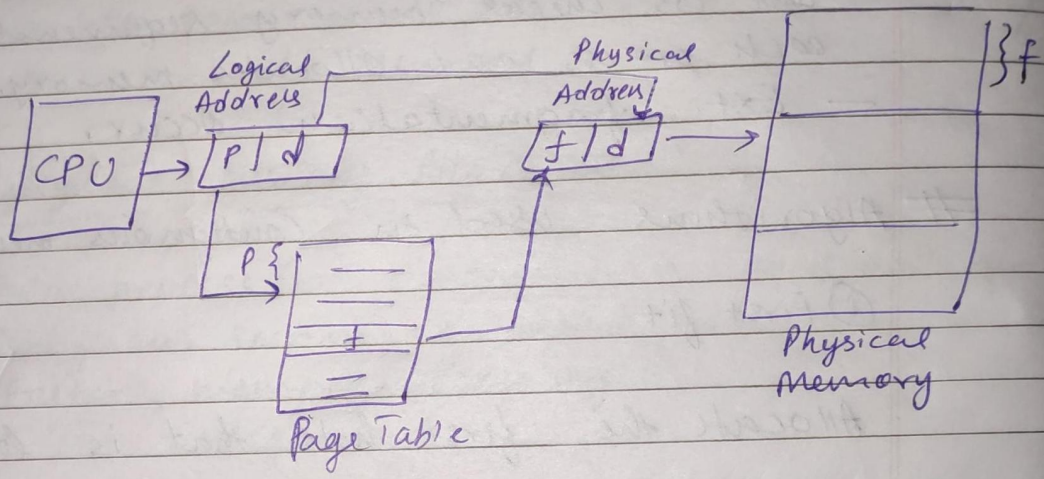
Memory Protection :

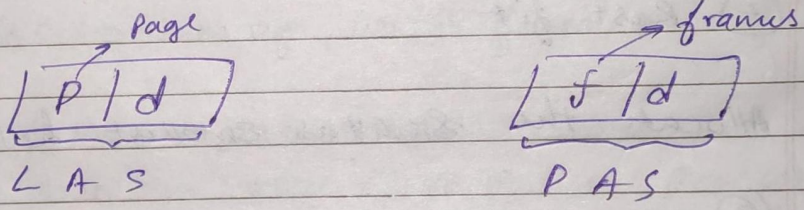
- We are allocating separate memory spaces to each process
- Obviously there should be a range for these addresses of memory spaces
- Base register → smallest phy. memory address
  Limit register → size of range

Date : / /
Page No.

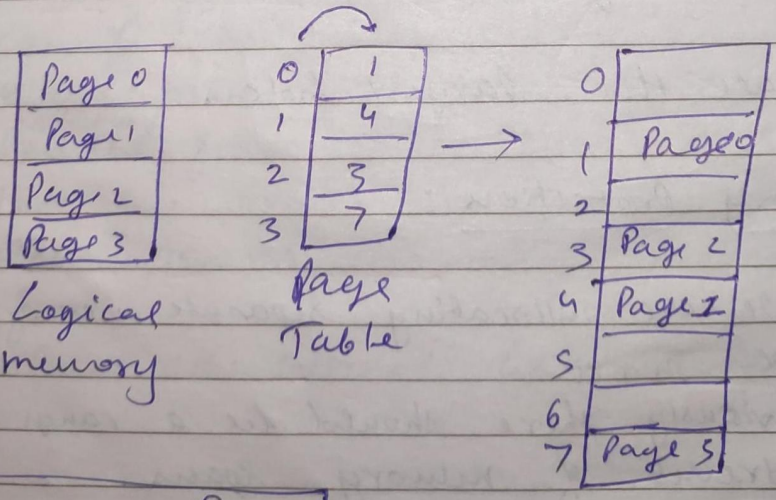
Paging :

- a scheme that permits phy. addrss space a process to be non-contiguous.
- Prone to internal fragmentation.





Example:



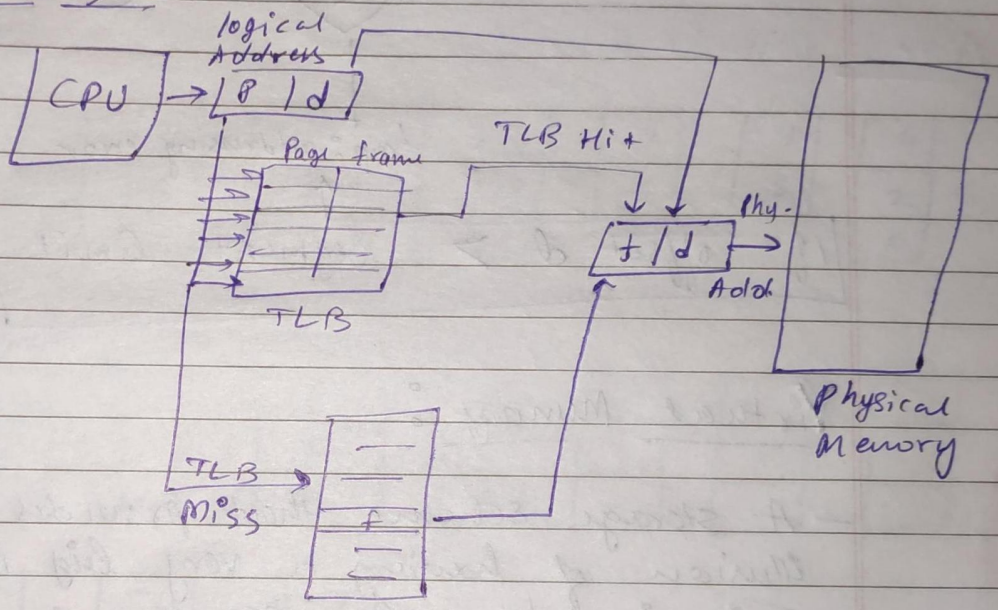
Frame no.	Page no.

Page table

Date : / /
Page No.

Paging with TLB :

- To increase speed of memory accss, TLB is used with page table.
- Small, fast
- Fixed sized blocks



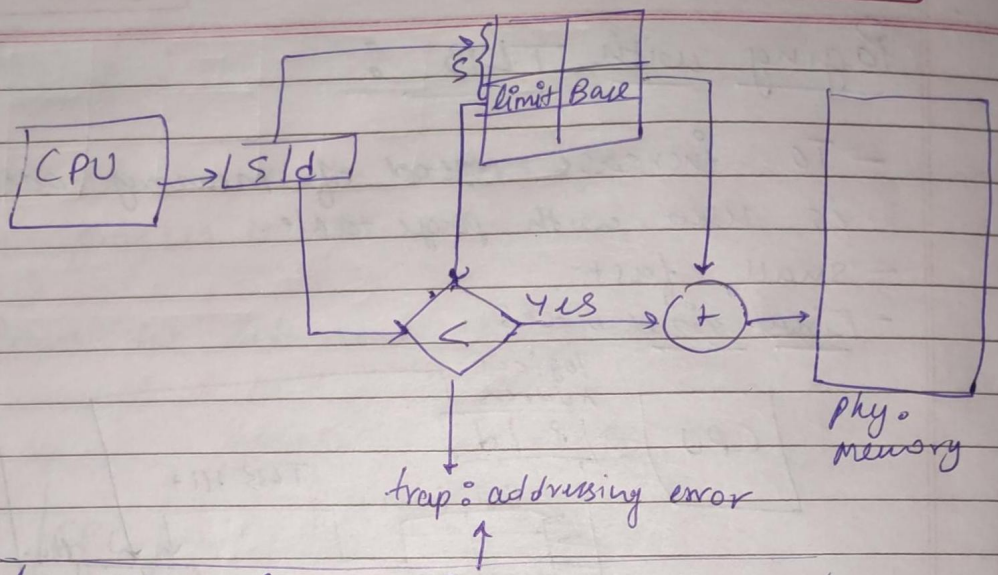
$$\text{Hit Ratio} = \frac{N_{Hit}}{(N_{Hit} + N_{miss})}$$

- Valid bit → 1 → Page is in LAS
  0 → Page is not in LAS

Segmentation :

- LAS has variable sized segments. Each segment has a name and a length
- Avoids both ext. and int. frag.
- Takes lesser space in phy. memory.

Date : / /
Page No.



If offset d > Segment limit

Virtual Memory :

- A storage scheme that provides user an illusion of having a very big main memory.
- This is done by treating a part of secondary memory as main memory.

Demand Paging :

- only ~~those~~ pages are loaded when they are demanded.
- Lazy swapper is used, it never swaps a page into memory unless that page will be needed.

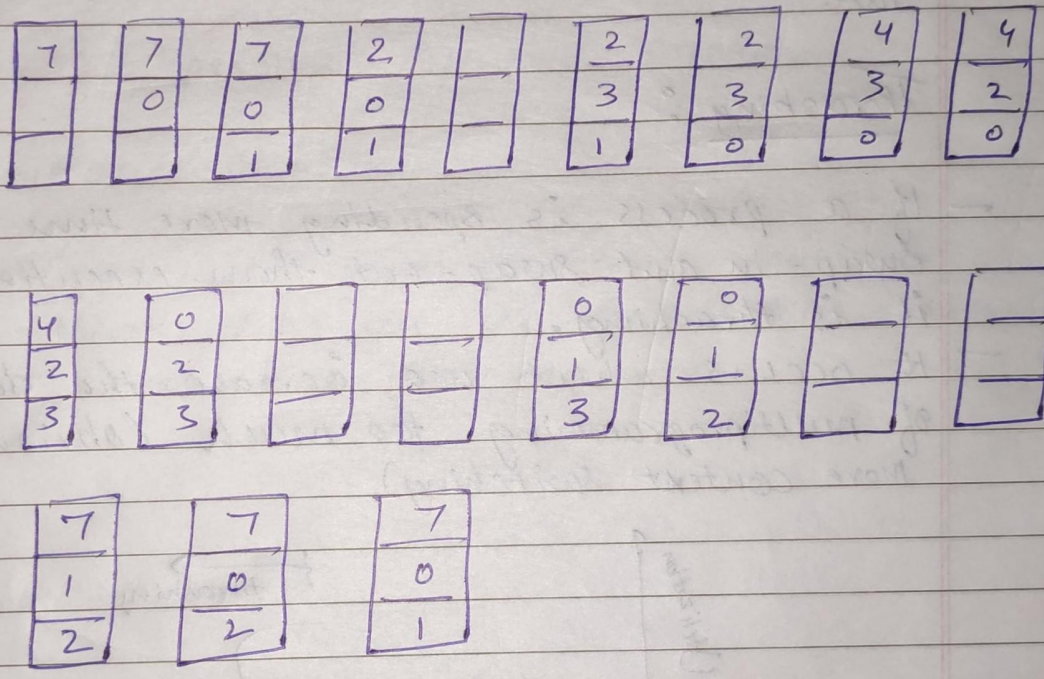
Effective Access Time = (1-p) * ma + p * page fault time

Date : / /
Page No.

Q. 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

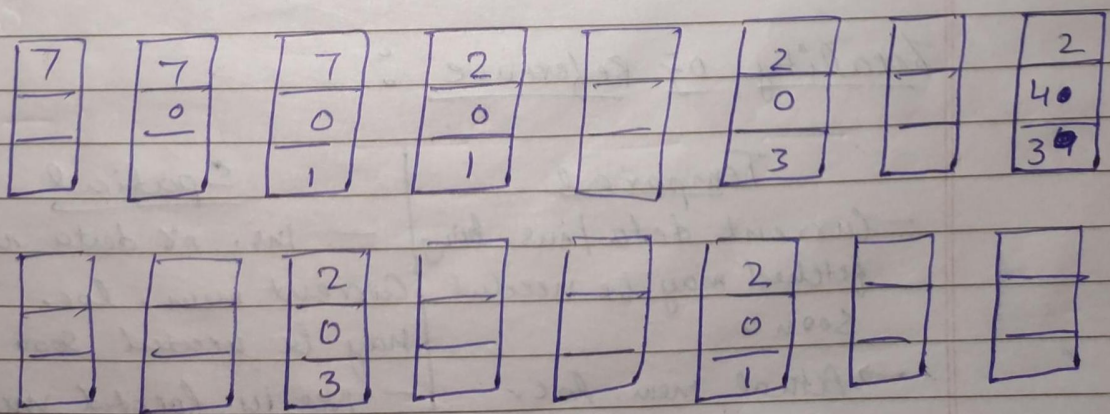
Apply : (3 frames)

(1) FIFO

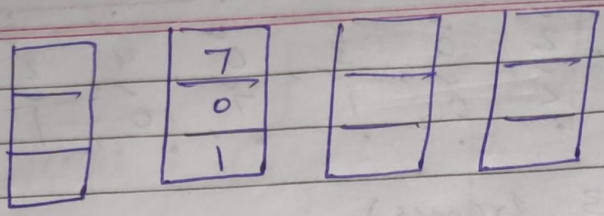


Page Faults = 15

(2) Optimal Page Replacement



Date : / /
Page No.

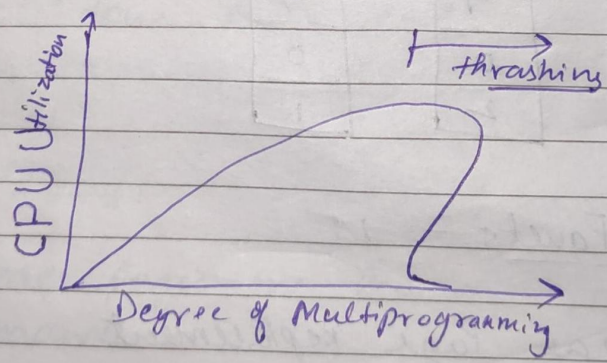


Page Faults = 9

Note :- Optimal Page Replacement has lowest page-fault rate.

Thrashing :

- If a process is spending more time in swap-in and swap-out than executing then it is thrashing.
- It occurs when we increase the degree of multiprogramming too much (obviously, more context switching).



Locality of Reference :

Temporal	Spatial
- Current data/ins. being fetched may be needed soon. → Actual mem. loc.	- Ins. or data near to current mem. loc. being fetched may be needed soon in future. - Nearly located mem. loc.

Date : / /
Page No.

Other useless Machines :

① Bare Machine

- Comp. with no OS
- Ins. directly on logical HW.
- uses machine language.
- Fast code
- Expensive Software creation

② Resident Monitor

- Code that runs on Bare Machine
- Works like an OS that controls instructions
- Works like job sequencer

Date : / /
Page No.

UNIT - 5

Disk Scheduling Algorithms :

1. FCFS
   - Scheduled in the order they arrive in disk queue

2. Shortest Seek Time First (SSTF)
   - Request is calculated in advance in the queue
   - Request near disk arm gets executed first.

3. SCAN
   - ELEVATOR Algorithm
   - Disk moves into a particular direction
   - Then reverses its direction and again services the request arriving in its path.
   - Touches the first limit of disk (upper or lower)

4. LOOK
   - Improved version of SCAN Algorithm.
   - Head does not touch limit, just goes to last request and reverses its direction

5. CSCAN
   - Direction is preserved
   - Both limits are touched.

Date : / /
Page No.

(6) CLOOK

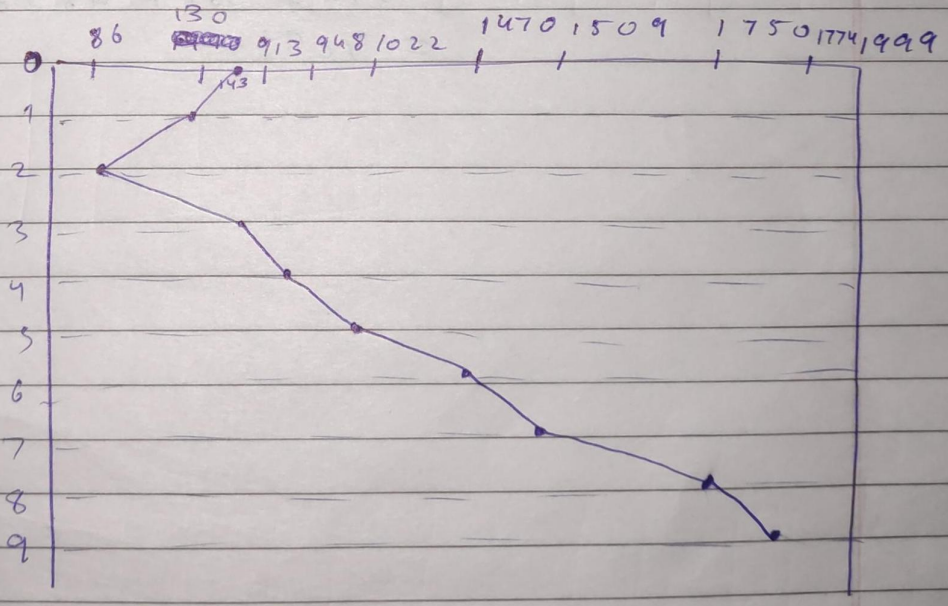
- Same as CSCAN
- No limits are touched
- Only first and last requests are touched to preserve direction

8. 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Initial Head = 143
Range = 0 to 1999

Apply :

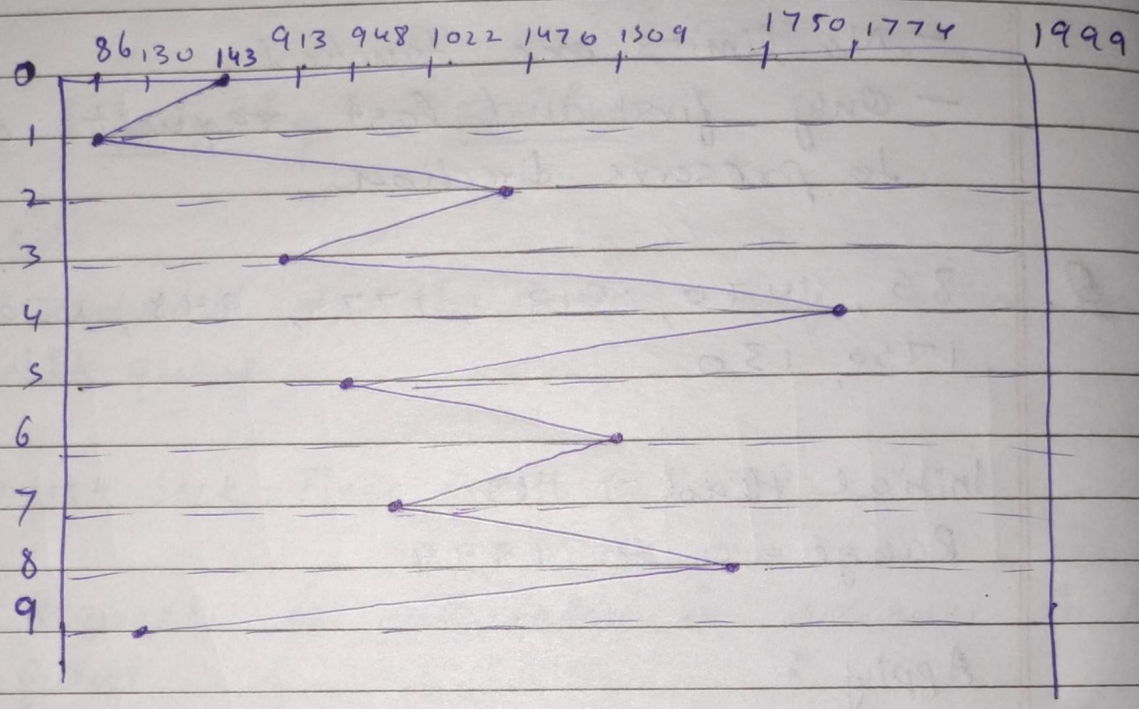
(1) ~~FCFS~~ SSTF



Total Seek Time = 13 + 44 + 827 + 35 + 74 + 448 + 39 + 241 + 24
= 1745

Date : / /
Page No.

(2) FCFS



Total Seek Time = 57 + 1384 + 557 + 861 +
826 + 561 + 487 + 728 + 1620
= 7081