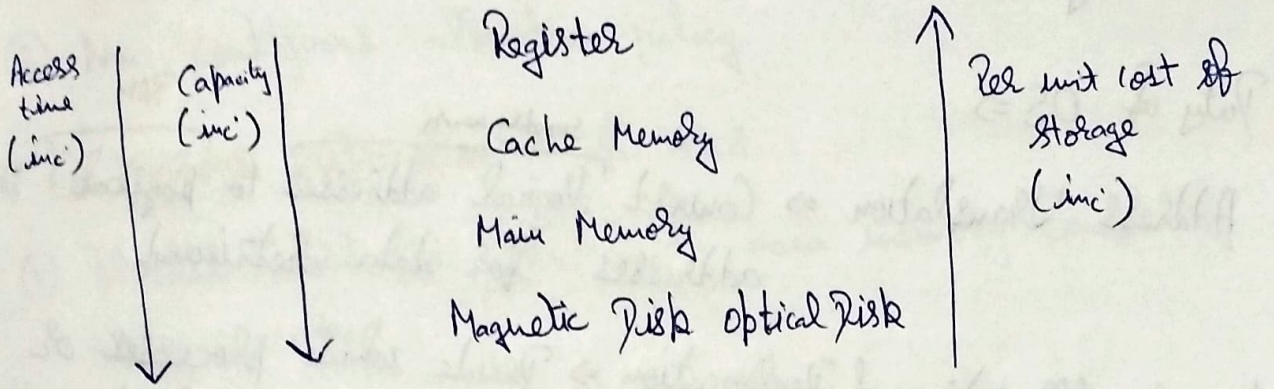
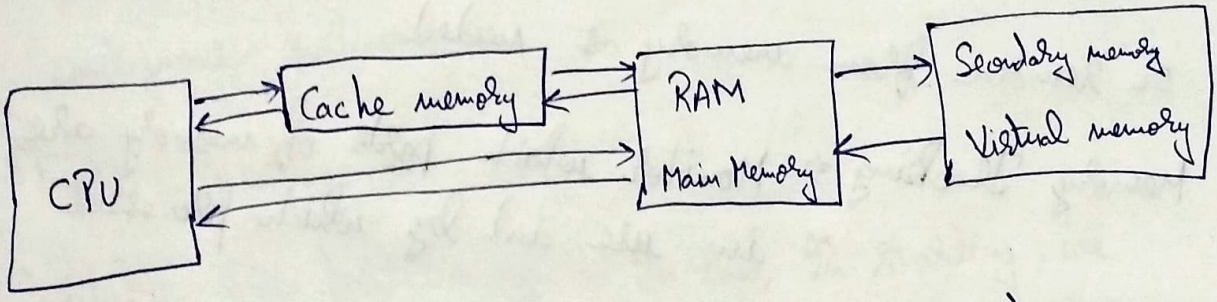
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Memory Hierarchy =>



★



★ Locality of Reference => The references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as property of locality of reference.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

① Spatial Locality ⇒ Use of data elements in nearby locations.

② Temporal Locality ⇒ It refers to the reuse of specific data or resources within a relatively small time duration i.e. Most recently used.

★ Duty of OS ⇒

① Address Translation ⇒ Convert logical addresses (Secondary memory) to physical addresses (RAM) for data retrieved.

② Memory Allocation & Reallocation ⇒ Decide which processes or data segment to load or remove from memory as needed.

③ Memory Tracking ⇒ Monitor which parts of memory are in use and by which processes.

④ Memory Protection ⇒ Implement safeguards to restrict unauthorized access to memory, ensuring both process isolation & data integrity.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ 2 Approach to store process in main memory ⇒

① Contiguous allocation policy

② Non- contiguous allocation policy

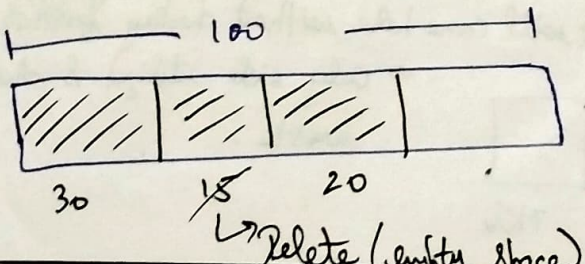
☆ Contiguous allocation ⇒ In this

① process must be loaded to main memory completely for execution.

② process must be stored in main memory in contiguous fashion.

. Space Allocation ⇒

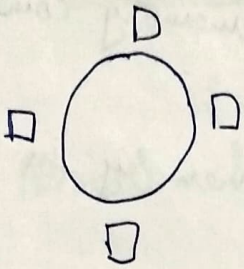
① Variable size partitioning ⇒ In this, In starting, we treat the memory as a whole or a single chunk & whenever a process request for some space, exactly same space is allocated if possible & the remaining space can be reused again.



Main Ideas, Questions & Summary:

Library / Website Ref.:-

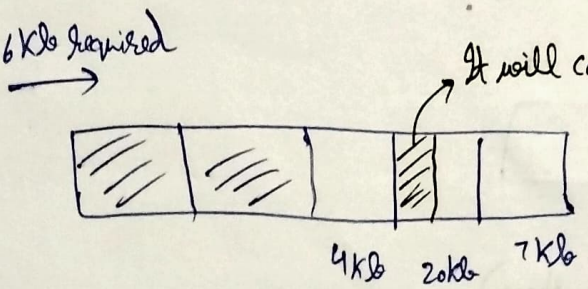
② Fixed size partitioning ⇒ Here, we divide memory into fixed size partitions, which may be of diff sizes, but here if a process request for some space, then a partition is allocated entirely if possible, & remaining space will be wasted internally.



→ only 3 come to sit then 1 seat is wasted

★ First fit policy ⇒ It starts searching the memory from the base & will allocate first partition which is capable enough.

- Advantage ⇒ Simple, easy to use, easy to understand
- Dis advantage ⇒ Poor performance, both in terms of time & space.



Worked in both Variable & Fixed size partitioning.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Best fit policy => We search the entire memory & will allocate the smallest partition which is capable enough.

- Advantage => Perform best in fixed size partitioning
- Disadvantage => Difficult to implement, perform worst in variable size partitioning as the remaing spaces which are of very small size.

☆ Worst fit policy => It also searches the entire memory & allocate the largest partition possible.

- Advantage => Perform best in variable size partitioning

Disadvantage => Worst in fix size partitiong, resulting into large internal fragmentation

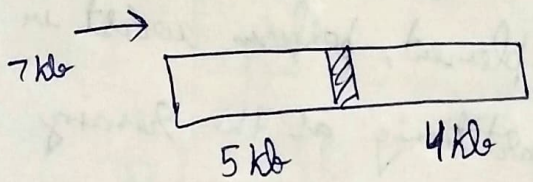
☆ Next fit policy=>

Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ External fragmentation ⇒ It is a function of Contiguous allocation policy.

The space requested by the process is available in memory but, as it is not being contiguous, cannot be allocated this wastage is called external fragmentation.



Space is available but can't use because it is contiguous.

☆ Internal fragmentation ⇒ It is a function of fixed size partition which means, when a partition is allocated to a process. Which is either the same size or larger than request then, the unused space by the process in the partition is called internal fragmentation.

☆ How to solve external fragmentation?

Ans • We should use non-contiguous allocation which means process can be divided into parts & diff parts can be allocated in different area.

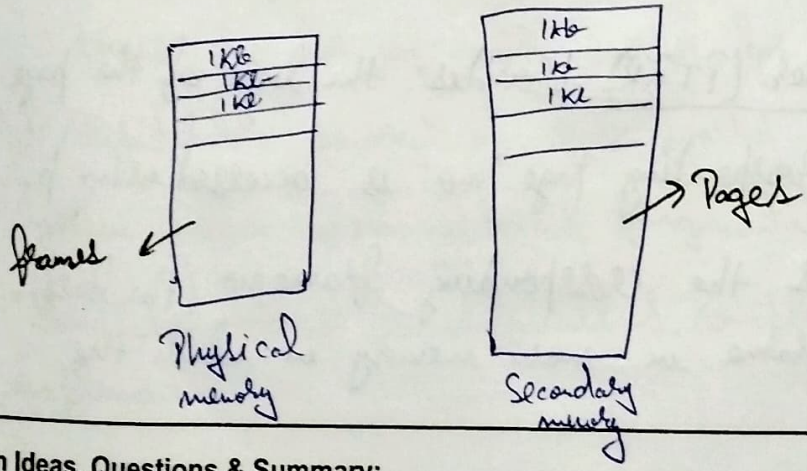
• We can also do defragmentation but this solution is very costly in respect to time. So we don't use this.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Non Contiguous memory allocation (Paging) ⇒

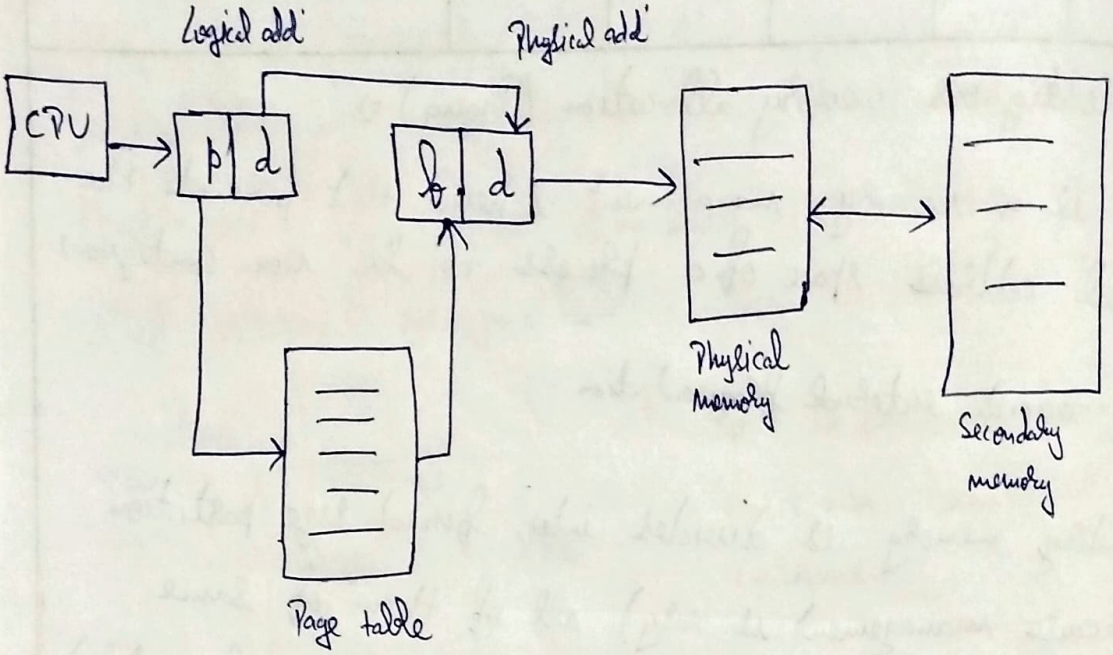
- Paging is a memory-management scheme that permits the physical address space of a process to be non-contiguous.
- Paging avoid external fragmentation.
- Secondary memory is divides into fixed size partition (because management is easy) all of them of same size called pages (easy swapping & no external fragmentation).
- Main memory is divided into fix size partition, each of them having same size called frames.
- Size of frame = Size of pages ~~(no of partition)~~
- In general no of pages are much more than no of frames



Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Translation process ⇒



- CPU generate a logical address which is divided into 2 parts ⇒ p & d

  p → page no.
  d → instruction offset

- Page number(p) is used as an index into a Page table
- Page table base register (PTBR) provides the base of the page table & then the corresponding page no. is accessed using p.
- Here we will finds the corresponding frame no. (the base address of that frame in main memory in which the page is stored).
- Combine corresponding frame no. ~~to~~ with the instruction offset & get physical address. Which is used to access main memory.

• Page table size = No of entries in Page table * size of each entry (b)

• Process size = No of pages * Size of each page

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Page Table ⇒

- It is a data structure not hardware.
- Every process have a seperate page table
- Number of entries a process have in page table is the number of pages a process have in secondary memory.
- Size of each entry in the page table is same it is corresponding frame number.
- Page table is a data structure which is itself stored in main memory.
- Advantage of Paging ⇒ Removal of internal fragmentation
- Disadvantage of Paging ⇒

① Translation process is slow as main memory is accessed 2 times (one for page table, another for actual access).

② A considerable amount of space wasted in storing page table.

③ System suffer from internal fragmentation (as it is fix size pol)

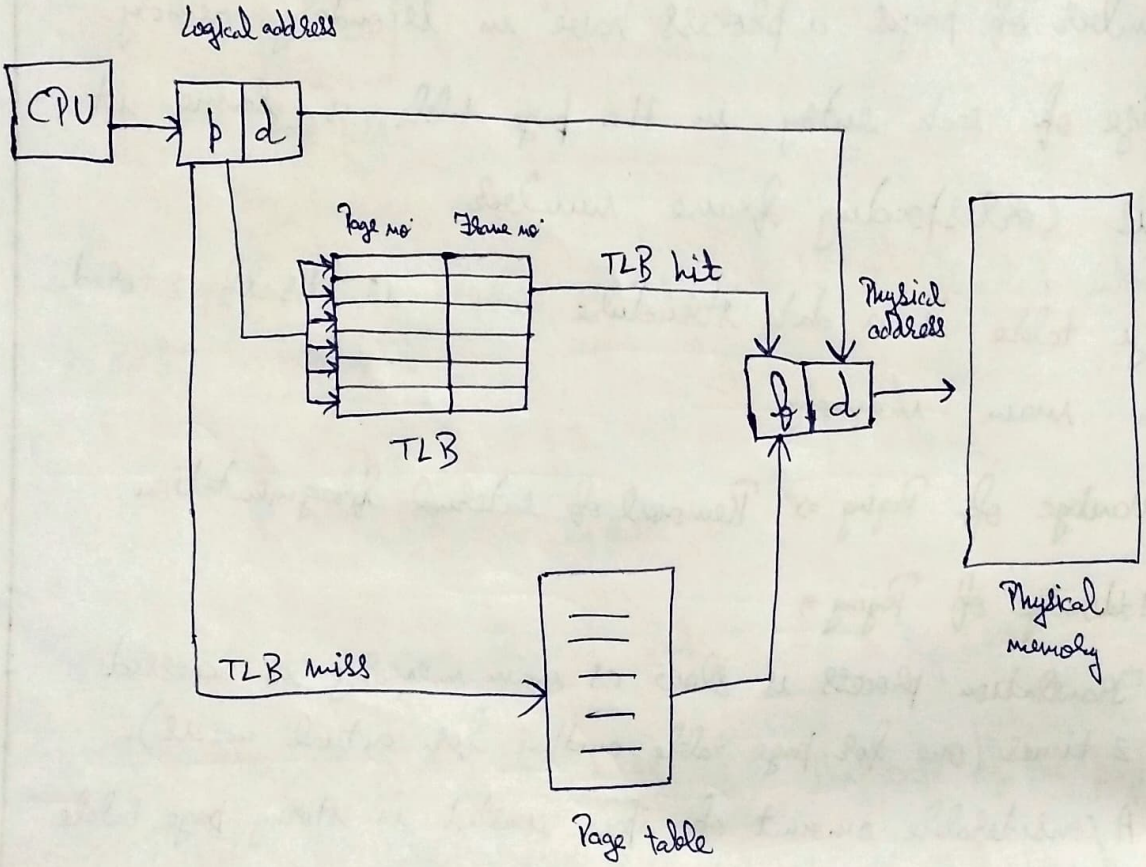
④ Translation process is difficult & complex to understand & implement.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✱ Resolving disadvantage of paging =>

- Translation process is slow. To resolve this we take help of TLB. The TLB is associative, high speed memory.
- Translation lookaside ~~fast~~ buffer (TLB)
- It is a hardware, not software



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

- Each entry in the TLB consists of 2 parts ⇒ a key (page no.) & a value (frame no.).

When the associative memory is search for page no, the page no. is compared with all page no. simultaneously. If the item is found, the corresponding frame no. field is returned.

- The search is fast; the hardware, however, is expensive, TLB contains the frequently referred page numbers & corresponding frame number.
- If the page number is not in TLB (known TLB miss) then memory reference to the page table must be made.
- Page size ⇒ ① Size of page table inc, then internal fragmentation increase but size of page table decrease.

② If we decrease the size of page then internal fragmentation dec but size of page table inc.

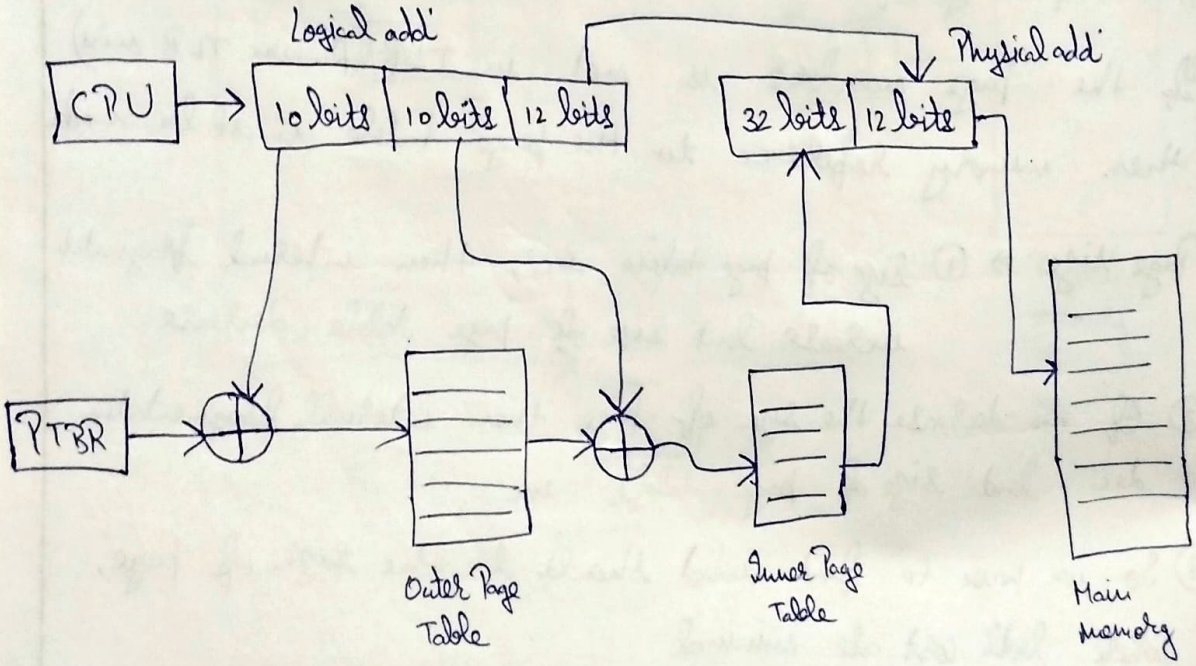
③ So we have to find what should be the size of page, where both cost are minimal.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Multilevel paging / Hierarchical paging =>

- Modern systems supports a large logical address space. In such cases, the page table itself becomes excessively large and can contain millions of entries. It can take a lot of space in memory, so cannot be accommodated into a single frame.
- A simple solution to this is to divide page table into smaller pieces.
- One way is to use a two-level paging algorithm, in which the page table itself is also paged.



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Segmentation ⇒ Paging is usable to separate the user's view of memory from the actual physical memory. Segmentation is a memory management scheme that supports this user view of memory.

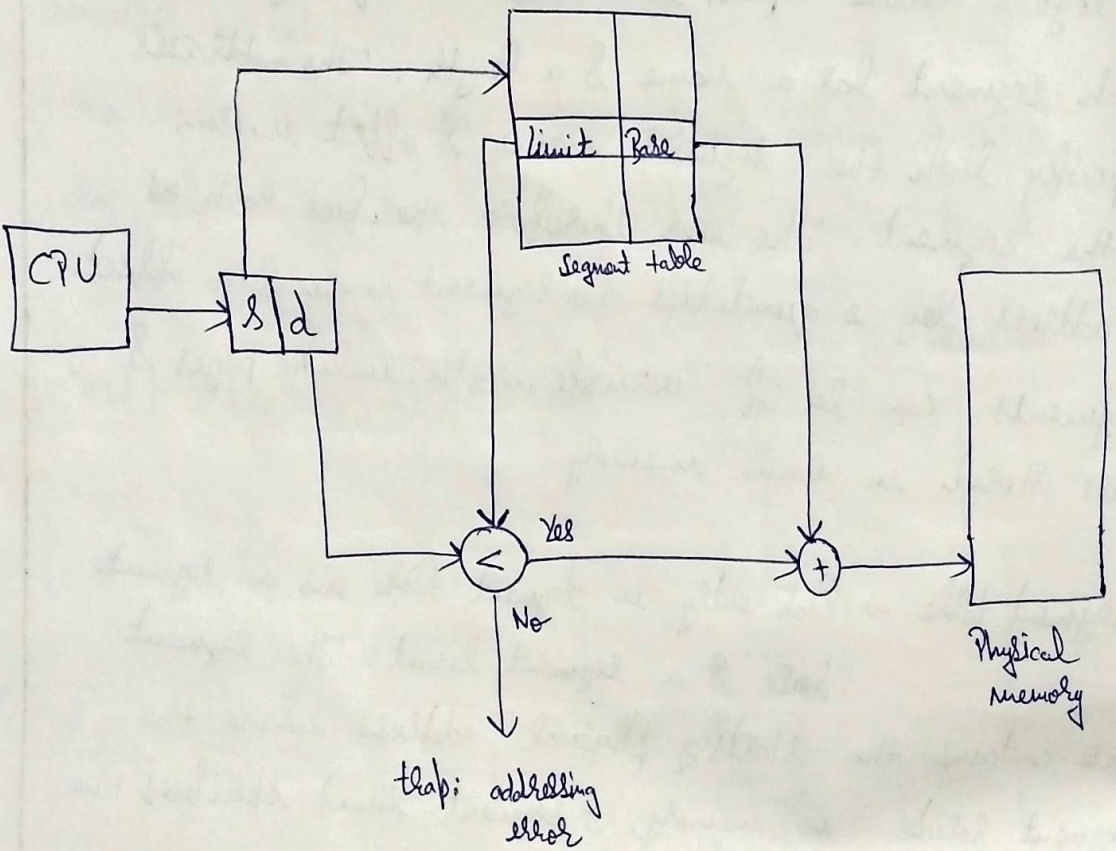
- A logical address space is a collection of segments. Each segment has a name & a length. The addresses specify both the segment name & offset within the segment. The user therefore specifies each address by 2 quantities ⇒ a segment name & an offset.
- Segments can be of variable length unlike pages & are stored in main memory.
- Segment table ⇒ Each entry in segment table has a segment base & a segment limit. The segment base contains the starting physical address where the segment resides in memory, & segment limit specifies the ~~limit~~ length of segment.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

- The segment no is used as an index to the segment table. The offset d of logical address must lie b/w 0 and segment limit. If it is not, use trap to the OS.

- When an offset is legal, it is added to segment base to produce the address in physical memory of the desired byte. Segmentation suffers from external fragmentation.



It is not good, so use segmentation with paging.

POORNIMA

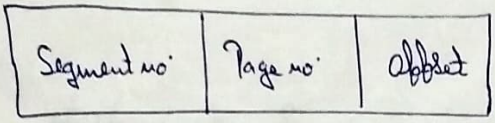
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

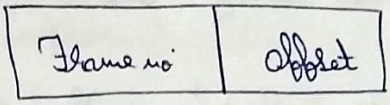
☆ Segmentation with paging =>

. Segmentation suffers from external fragmentation. It is better to divide segments into pages.

. In segmentation with paging a process is divided into segments & further segments are divided into pages.

. It is quite better than multilevel paging.





Same as previous

☆ Inverted page table =)

Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Virtual memory ⇒ It is a memory management technique used by OS to give the appearance of a large, continous block of memory to application, even if the physical memory (RAM) is limited. Enabling large applications to run on system with less RAM.

☆ Pure Demand paging ⇒

- A memory management scheme where a process starts with no pages in memory. Pages are only loaded when explicitly required during execution.

① Process starts with zero pages in memory. Immediate page fault occurs.

② Load the needed page into memory. Execution resumes after the required page is loaded into memory.

③ Addition page faults occurs as the process continues and requires new pages.

④ Execution proceeds without further faults once all necessary pages are in memory. The key principle is to only load pages when absolutely necessary.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

- Advantage =>
- A program would no longer be constrained by the amount of physical memory that is available, Allows the execution of processes that are not completely in main memory i.e. process can be larger than main memory.
- More programs could be run at the same time as use of main memory is less.

- Disadvantage =>
- Virtual memory is not easy to implement.
- It may substantially decrease performance if it is used ~~carefully~~ carelessly (Thrashing)

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✱ Implementation of virtual memory =>

- We add a new column page table, which have binary value 0 or Invalid which means page is not currently in main memory, 1 or valid which means page is currently in main memory.

✱ Page Fault => When a process tries to access a page that is not in main memory then a page fault occurs.

✱ Steps to handle page fault =>

① If the reference was invalid, it means there is a page fault and page is not currently in main memory, now we have to load this required page in main memory.

② We find a free frame if available we can brought in desired page, but if not we have to select page as a victim & swap it out from main memory to secondary memory & then swap in desired page.

Diagram

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

We can reduce this overhead by using a modify bit or Dirty bit as a new column in page table.

(3.1) The modify bit for page is set whenever the page has been modified. In this case, we must write the page to disk.

(3.2) If the modify bit is not set :⇒ It means the page has not been modified since it was read into main memory. We need not write the memory page to disk.

☆ Page Replacement algo ⇒ It will decide which page to replace in demand paging

Diag

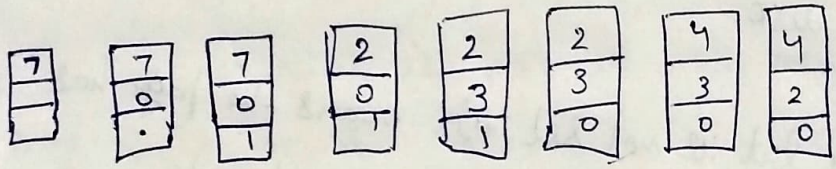
Main Ideas, Questions & Summary:

Library / Website Ref.:-

① First In First Out Algo ⇒

It associates with each page, the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen i.e. first page that came into memory will be replaced first.

Pr ⇒ 7 0 1 2 0 3 0 4 2 3 0

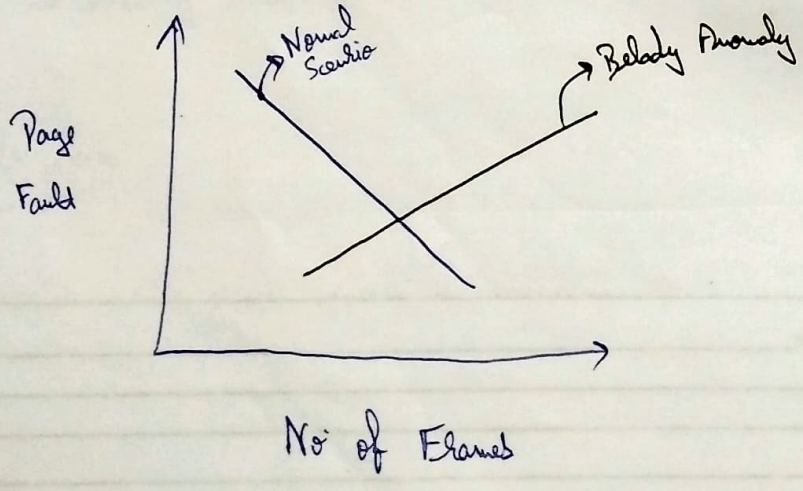


Hit / miss

No of page fault 8

- Easy to understand & program.
- Performance not always good.

★ Belady's Anomaly ⇒ For some page replacement algo, the page fault rate may increase as the no of allocated frames increase.



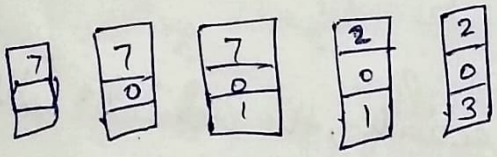
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(2) Optimal Page Replacement Algo ⇒

- Replace the page that will not be used for longest period of time.
- It has lowest page fault rate of all algo.
- Never suffer from Belady's anomaly
- Difficult to implement because it requires future knowledge of reference string.

Ex ⇒ 7 0 1 2 0 3 0 4 2 3 0 3 2 1



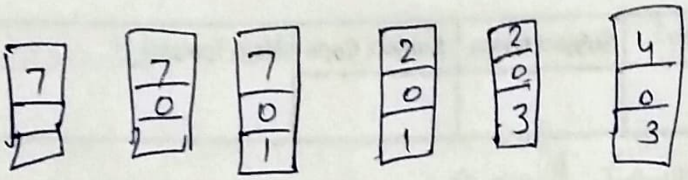
(3) Least Recently used (LRU) algo ⇒ This algo look backward in time rather than forward. Replace the page that has not been used for longest period of time.

- Much better than FIFO in term of page fault.
- Never suffer from Belady anomaly.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

Ex ⇒ 7 0 1 2 0 3 0 4 2 3

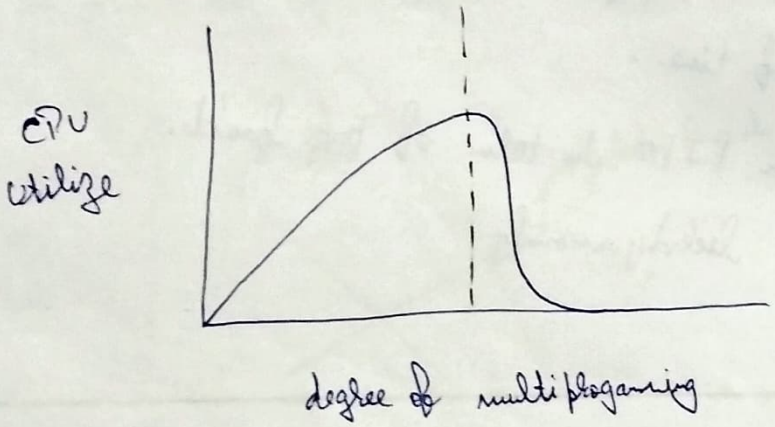


* Thrashing ⇒

① When a process spends more time swaping pages than executing, it's called Thrasing. Low CPU utilization prompts adding more processes to increase multiprograming.

② If a process needs more frames, it starts taking them from others, causing those processes to also fault & swap pages. This emptied the ready queue & lower CPU utilization.

③ Responding to decreased CPU activity, the scheduler adds more processes, worsening the issue. This lead to thrashing.



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Disk Management ⇒

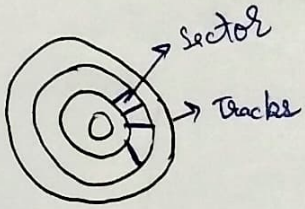
- Total Transfer time = Seek time + Rotational Latency + Transfer time

- Seek time ⇒ Time taken by Read/Write header to reach the correct track.

- Rotational Latency ⇒ It is the time taken by read/write header during the wait for the correct sector. In general, it's a random value, so for average analysis, we consider the time taken by disk to complete half rotation.

- Transfer time ⇒ Time taken by read/write header either to read or write on a disk.

total time = (File size / Track size) × time taken in one revolution



Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Disk scheduling ⇒ It is a technique, OS use to manage the order in which disk I/O request are processed. Also known as I/O scheduling.

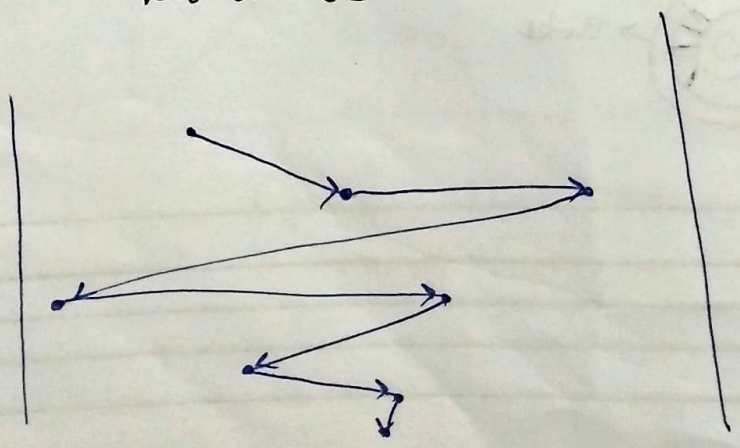
- Main goal is to optimize the performance of disk operation, reduces the time it takes to access data & improve overall system efficiency.

① FCFS (First come First serve) ⇒ In this, the requests are addressed in the order they arrive in disk queue.

- Advantage ⇒ Easy to use & understand
  ⇒ Every request gets a fair chance
  ⇒ No Starvation

- Disadvantage ⇒ Does not try to optimize seek time, or waiting time.

Ex ⇒ queue = 48, 183, 37, 122, 14
head at 53



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

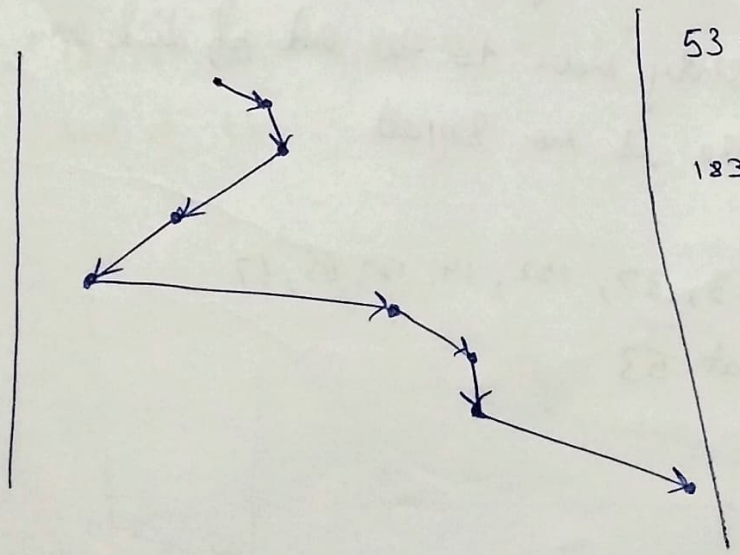
② SSTF (Shortest Seek Time First) scheduling ⇒

. In this, request nearest to the disk arm will get executed first i.e. requests having shortest seek time are executed first.

. Advantage ⇒ Seek movement decrease
⇒ Throughput increase

. Disadvantage ⇒ Overhead to calculate the closest request
⇒ Can cause starvation

Ex ⇒ Queue ⇒ 98, 183, 37, 122, 14, 124, 65, 67
head at 53



53 → 65 → 67 → 37 → 14
↓
183 ← 124 ← 122 ← 98

Main Ideas, Questions & Summary:

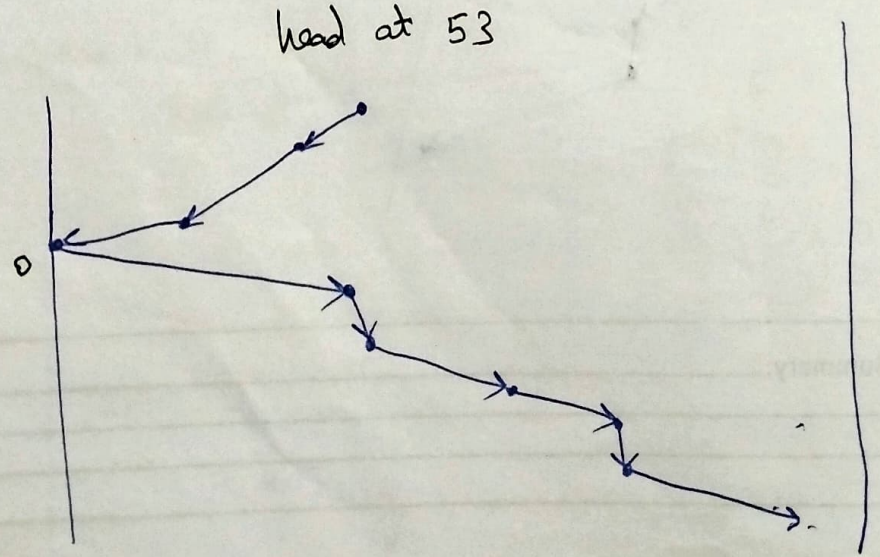
Library / Website Ref.:-

(3) SCAN / Elevator Algo ⇒

- Disk arm starts at one end of disk & move towards the other end, servicing requests as it reaches each track, until it gets to other end of disk.
- At the other end, the direction of head movement is reversed, & servicing continues. The head continuously scans back & forth across the disk.
- Advantage ⇒ Easy to use & understand
  ⇒ No starvation
  ⇒ Average response time
- Dis Advantage ⇒ Long waiting time for request for location just visited by disk arm.
  ⇒ Unnecessary move to the end of disk, even if there is no request

Ex ⇒ Queue ⇒ 48, 183, 37, 122, 14, 124, 65, 67

head at 53



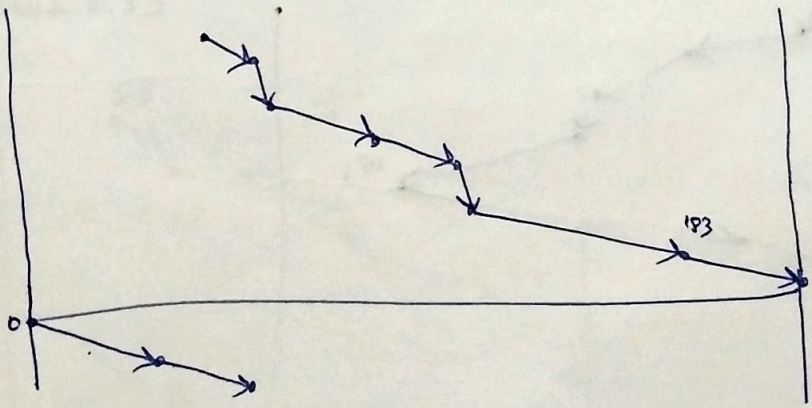
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(4) C-Scan Scheduling ⇒ In this, head moves from one end of disk to other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any request on return trip.

- Adv ⇒ Better response time compared to Scan
  ⇒ Provide more uniform wait time compared to SCAN.
- Dis Adv ⇒ More seeks movement in order to reach starting position.

Ex ⇒ 98, 183, 37, 122, 14, 124, 65, 67
head at 53



Main Ideas, Questions & Summary:

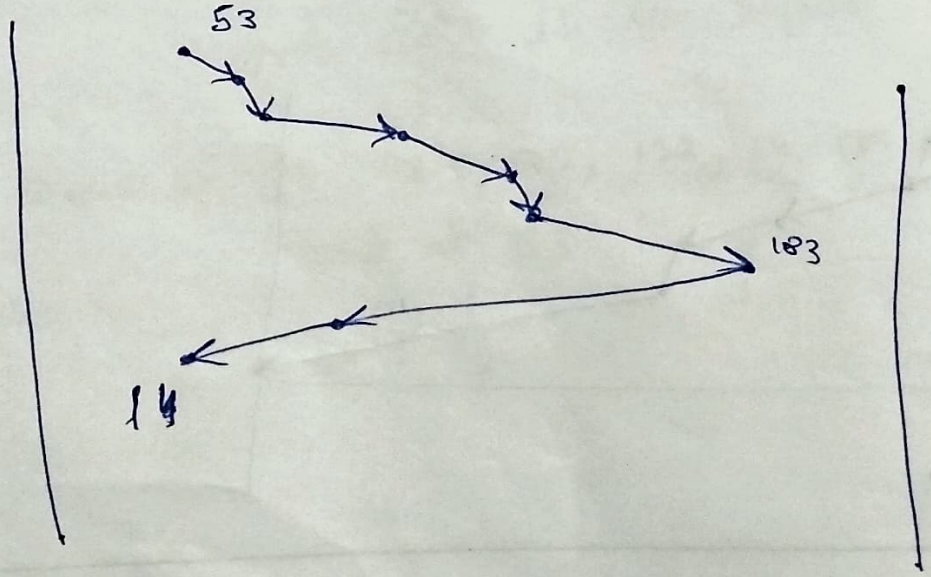
Library / Website Ref.:-

(5) Look scheduling ⇒ In this, disk arm inspite of going to end of disk goes only to the last request to be serviced in front of head and then reverses it direction from there only. Thus, it prevents the extra delay which occured due to unnecessary traversal to end of disks.

- Adv ⇒ Better performace than SCAN
  ⇒ Should be used in case of less load

- DisAdv ⇒ Overhead to find last request
  ⇒ Should not be used in case of more load

Q ⇒ 98, 183, 37, 122, 14, 124, 65, 67
Head = 53



POORNIMA

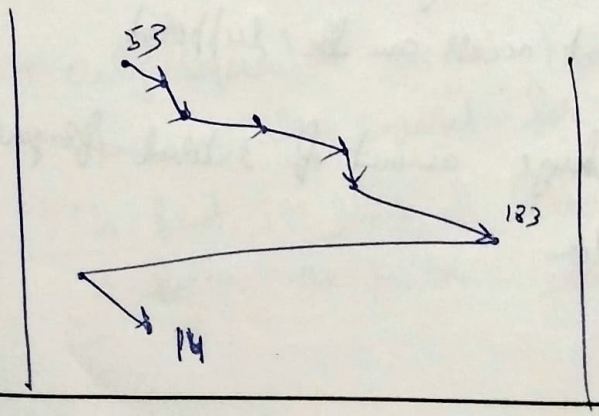
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(6) Clook ⇒ In this, the disk arm in spite of going to end, goes only to the last request to serviced in front of head & then from there goes to other end's last request. Thus, it also prevents the extra delay which occured due to unnecessary traversal to the end of disk.

- Adv ⇒ Better respons time than Look
  ⇒ Provides more uniform wait time compared to Look.
- Dis Adv ⇒ Overhead to find the last request & go to initial position is more
  ⇒ Should not used in case of more load

Ex ⇒ 98, 183, 37, 122, 14, 124, 65, 67

Head = 53



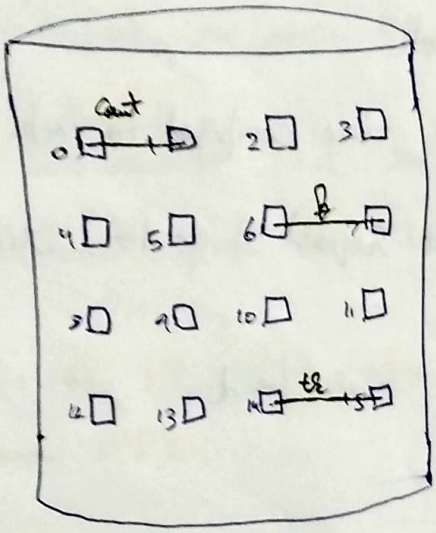
Main Ideas, Questions & Summary:

Library / Website Ref.:-

✱ File allocation methods ⇒

① Contiguous Allocation ⇒

- Each file occupy a set of contiguous blocks on disk.
- In directory usually we store 3 column file name, start addr, & length of file in number of blocks



directory

file	start	length
count	0	2
tr	14	3
mail	6	2

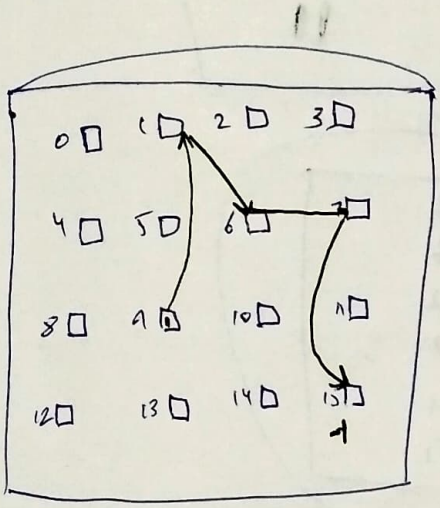
- Adv ⇒ Accessing file is easy.
  ⇒ Sequential & direct access can be supported.
- Disadv ⇒ Suffer from huge amount of internal fragmentation
  ⇒ File modification

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(2) Linked allocation =>

- Each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on disk. The directory contains a pointer to the first and last block of file.



Directory

file	start	end
jeep	9	15

- Adv => Create, read, write a new file easy.
  => Size of file need not to be declared when file created.
  => No external fragmentation.

- Disadv => Only sequential access possible.
  => Space is also required for pointers
  => To find i^{th} block of file, we must start a begining & follow the pointer until we get to i^{th} block.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

③ Indexed Allocation ⇒ This solves problems of contiguous and linked allocation, by bringing all the pointers together into one location : index block

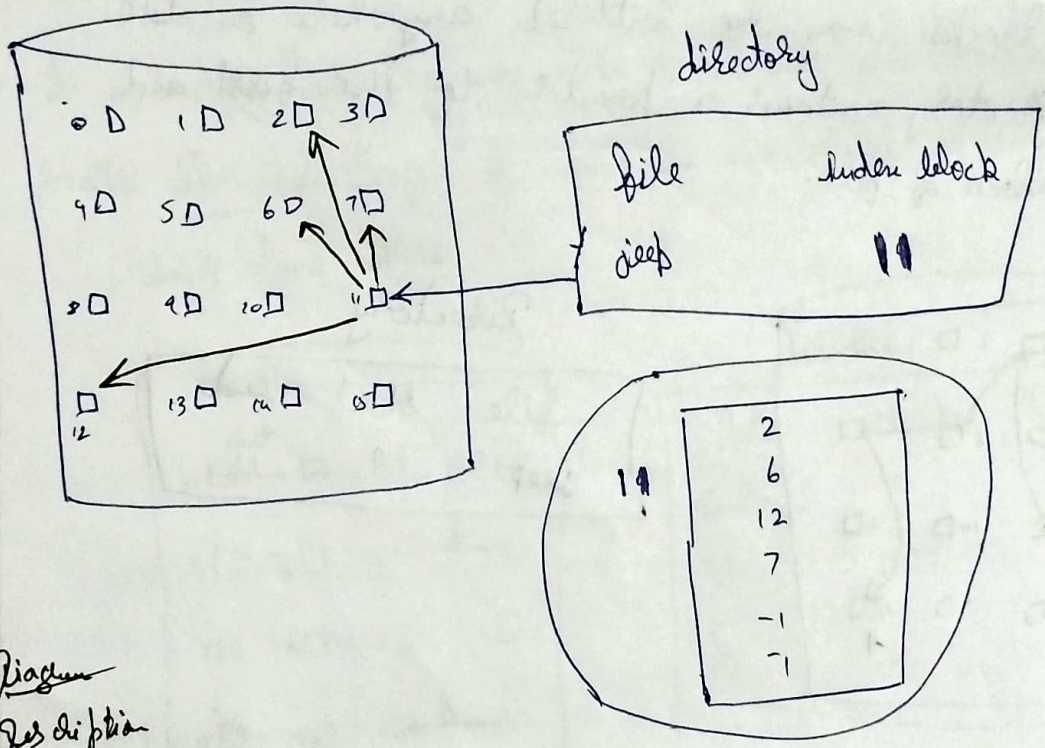
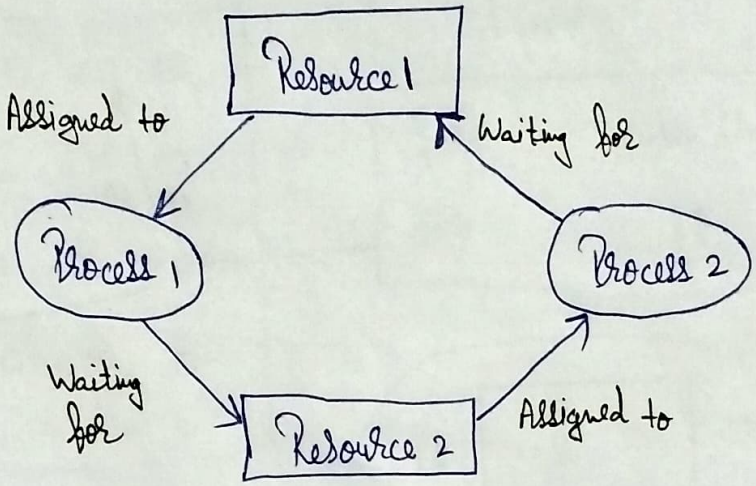


Diagram
Description

- Adv ⇒ Indexed allocation supports direct access,
  ⇒ No external fragmentation

- Disadv ⇒ Can suffer from wasted space in indexed allocation
  ⇒ pointer overhead of index block is generally greater than pointer overhead of linked allocation.

☆ Deadlock ⇒ It is a situation where a set of process are blocked because each process is holding a resource & waiting for another resource acquired by some other process.



☆ Necessary conditions for deadlock ⇒

- Deadlock can occur if all these 4 condition occur in system simultaneously.

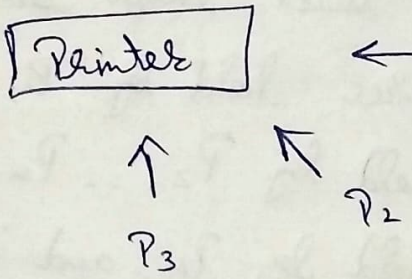
① Mutual exclusion
② Hold & wait
③ No pre-emption
④ Circular wait

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

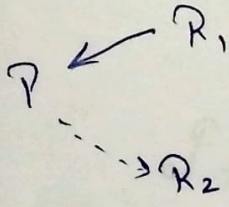
① Mutual exclusion ⇒ In this, at least one resource must be held in a non-sharable mode, i.e. only one process at a time can use the resource.

- If another process request that resource, the requesting process must be delayed until the resource has been released.

Ex ⇒ 

Every one sending printing operation together.

② Hold & wait ⇒ A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

Ex ⇒ 

P is having R1 & waiting for R2

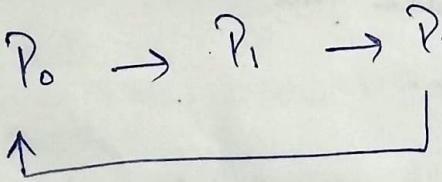
Main Ideas, Questions & Summary:

Library / Website Ref.:-

(3) No pre-emption ⇒

- Resources cannot be pre-empted.
- A resource can be released only voluntarily by the process holding it, after that process has completed its task.
- Process cannot take resource forcefully that is holding by another process.

(4) Circular wait ⇒ A set $P_0, P_1, \ldots P_n$ of waiting processes must exist such that P_0 is waiting for a resource held by P_1, P_2 is waiting for a resource held by $P_2, \ldots P_{n-1}$ is waiting for a resource held by P_n, and P_n is waiting for a resource held by P_0.



P_0 waiting for P_1, P_1 for P_2, P_2 for P_0.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Deadlock Handling methods ⇒

(1) Prevention ⇒ Design such protocol that there is no possibility of deadlock

(2) Avoidance ⇒ Try to avoid deadlock in run time by ensuring that the system will never enter a deadlock state.

(3) Detection ⇒ We can allow the system to enter a deadlock state, then detect it, and recover.

(4) Ignorance ⇒ We can ignore the problem altogether and pretended that deadlocks never occur in system.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

① Prevention ⇒ It means designing such system where there is no possibility of existence of deadlock. For that we have to remove one of the 4 necessary condition of deadlock. Eg ⇒ Polio vaccine

(1) Mutual exclusion ⇒ In this approach, there is no solution for mutual exclusion as resource can't be made shareable.

(2) Hold & wait ⇒

(i) Conservative approach ⇒ Process is allowed to run if & only if it has acquired all the resources.

(ii) Alternative protocol ⇒ A process may request some resources and use them. Before it can request any additional resources, it must release all the resources that it is currently allocated.

(iii) Wait time out ⇒ We place a max time outs up to which a process can wait. After which process must release all holding resources & exit.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(3) No pre-emption ⇒ PDF

(4) Circular wait ⇒ PDF

. Problem of prevention ⇒ System becomes slow and resources utilization & reduced system throughput.

. Costly

(2) Avoidance ⇒ To avoiding deadlocks we require additional information about how resources are to be requested. Which resources a process will request during its lifetime.

. With this additional knowledge, the OS can decide for each request whether process should be wait or not.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

Soln ⇒

	Max Need		
	E	F	G
P_0	4	3	1
P_1	2	1	4
P_2	1	3	3
P_3	5	4	1

	Allocation		
	E	F	G
P_0	1	0	1
P_1	1	1	2
P_2	1	0	3
P_3	2	0	0

System Max		
E	F	G
8	4	6

	Current Need		
	E	F	G
P_0	3	3	0
P_1	1	0	2
P_2	0	3	0
P_3	3	4	1

Available		
E	F	G
3	3	0

- First, P_0 complete, 330 + 101 = 431 → New Available
- Second, P_2 complete, 431 + 103 = 534
- Third, P_1 complete, 534 + 112 = 646
- Fourth, P_3 complete 646 + 200 = 846

$P_0 \rightarrow P_2 \rightarrow P_1 \rightarrow P_3$

Safe Sequence

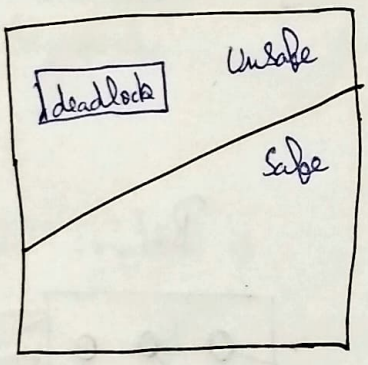
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Safe Sequence ⇒ Some Sequence in which we can satisfies demand of every process without going into deadlock, if yes, this sequence is called safe sequence.

☆ Safe state ⇒ If their exist at least one possible safe sequence

☆ Unsafe State ⇒ If their exist no possible safe sequence.



☆ Safety algo ⇒

Main Ideas, Questions & Summary:

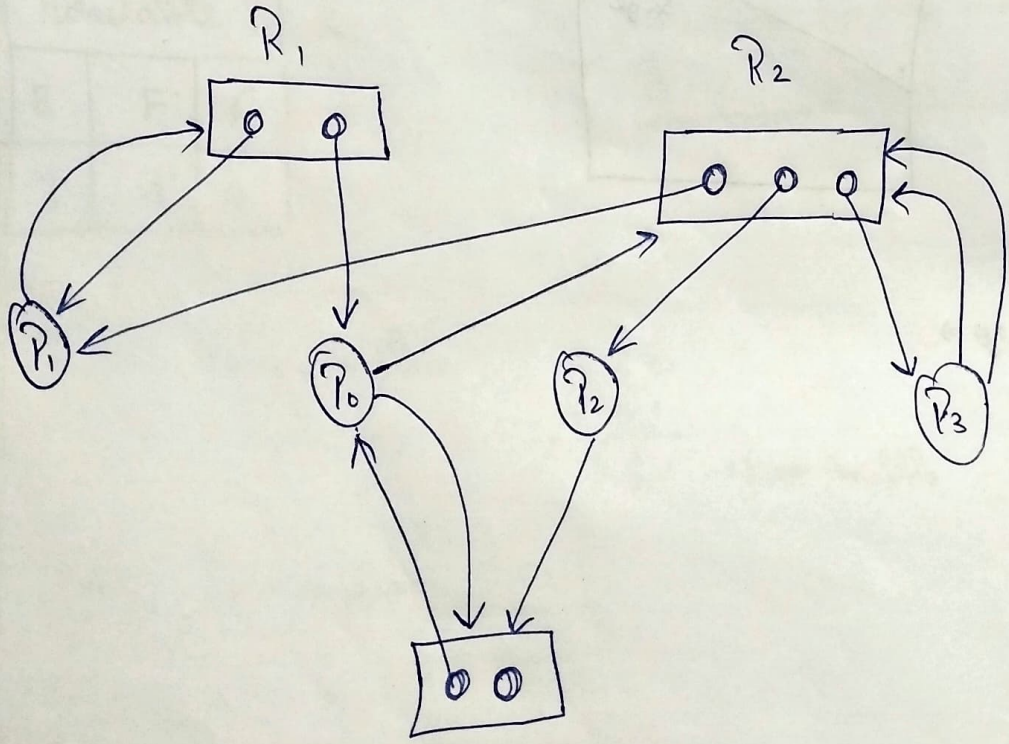
Library / Website Ref.:-

✡ Resource Allocation graph ⇒

- Deadlock can also be described in terms of a directed graph called System Resource-allocation graph.
- Set of vertices is partitioned into 2 different types of nodes

$P = \{P_1, P_2 \dots P_n\}$ Processes

$R = \{R_1, R_2 \dots R_m\}$ Resources



$P_2 \rightarrow P_0 \rightarrow P_1 \rightarrow P_3$

No deadlock

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(3) Deadlock detection & recovery ⇒

- Once deadlock is detected there are 2 options for recovery from deadlock

- Process Termination ⇒
  - → Abort all deadlocked processes
  - → Abort one process at a time until the deadlock is removed.

- ~~Resources~~

- Recourse pre-emption ⇒
  - → Selecting a victim
  - → Partial or complete rollback

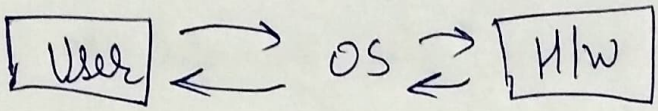
Main Ideas, Questions & Summary:

Library / Website Ref.:-

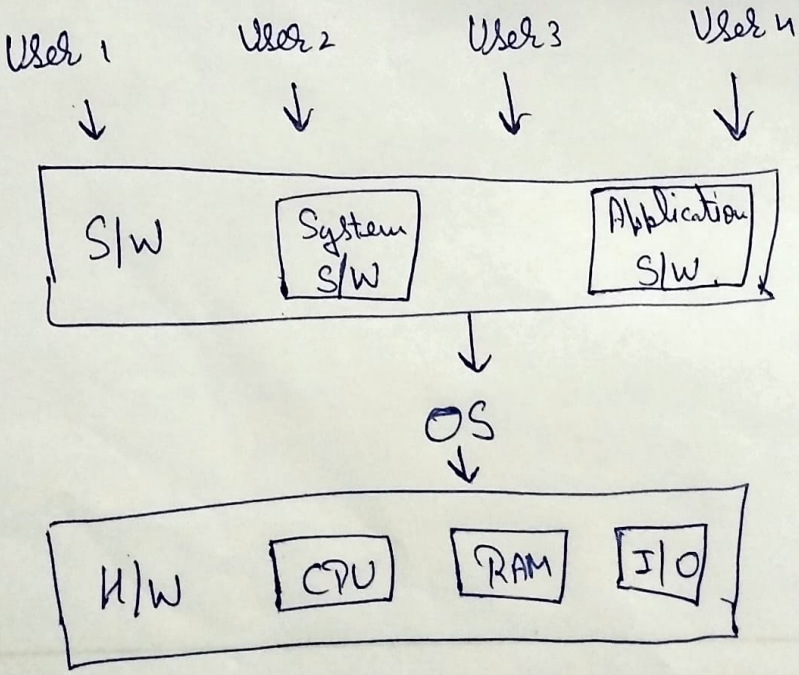
④ Ignorance (Ostrich algo) ⇒

- OS behaves like there is no concept of deadlock.
- Ignoring deadlock can lead to system performance issues as resources get locked by idle processes.
- Many OS opt for this approach to save on the cost of implementing deadlock detection.
- Deadlocks are often rare. Manual restart may be required when a deadlock occurs.

✡ Operating System ⇒ It act as a intermediate b/w user and hardware by using application.



- Controls the system resources among various applicati
- It act as platform on which applications can be installed.
- Ex ⇒ Windows, IOS, Linux, Android
- Conceptual view of OS ⇒



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Functions of OS ⇒

1. Process Management
2. Memory management
3. I/O device management
4. File management
5. Network management
6. Security & Protection

☆ Batch OS ⇒

1. Early computers were not interactive device
2. User use to prepare a job which consist of
   - (i) Program
   - (ii) Control information
   - (iii) Input data
3. Only one job is given input at a time as there is no memory.
4. Common I/O device were punch card or tape driver.
5. Devices are very slow & processor remain ideal most of time.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Spooling ⇒ It is a process in which data is temporarily held in memory or other volatile storage to be used by a device or a program.

☆ Multiprogramming OS ⇒

☆ Multitasking / time sharing OS ⇒ → Difference

☆ Multiprocessing OS ⇒

☆ Symmetric & Asymmetric processing ⇒

☆ Real time OS ⇒

☆ Distributed OS ⇒

☆ Structure of OS ⇒

☆ Micro Kernal ⇒

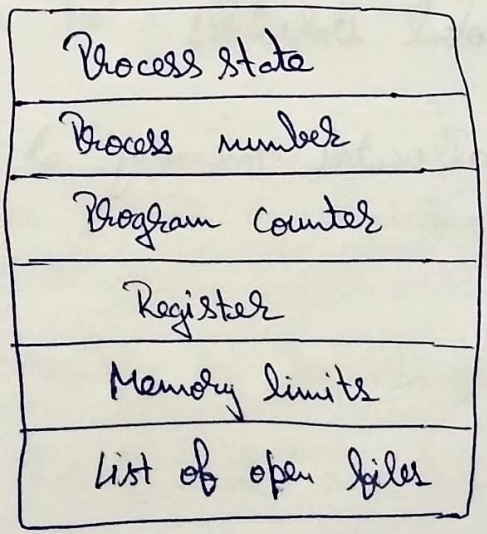
☆ System call ⇒

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Process ⇒ A process is a program in execution.

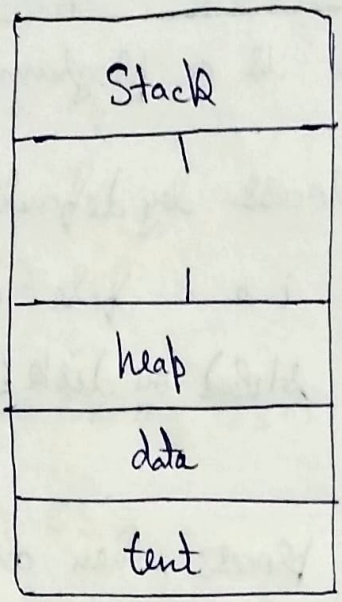
- A program is not a process by default. A program is a passive entity i.e a file containing a list of instructions stored on disk (often called executable file).
- A program becomes a process when an executable file is loaded into main memory & when it's PCB (Process Control Block) created.
- A process is an Active Entity, which requires resources like main memory, CPU time, register, etc.



Main Ideas, Questions & Summary:

Library / Website Ref.:-

- Process consist of following sections ⇒

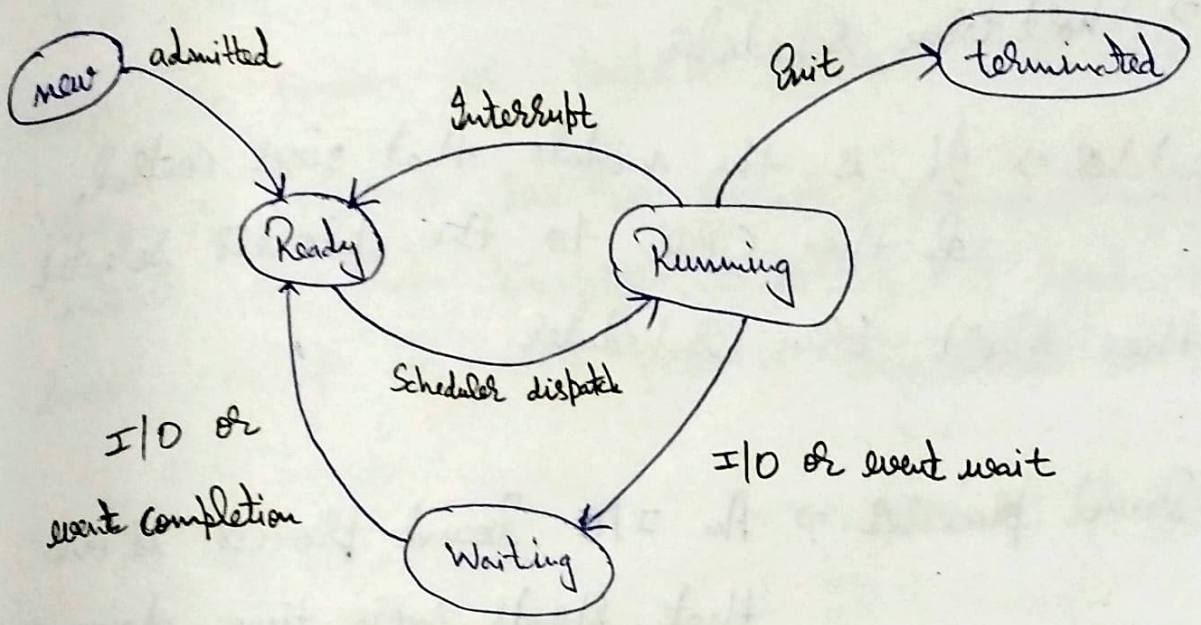


- Text ⇒ Also known as program code
- Stack ⇒ Contains temporary data (Function parameter, local variable)
- Data ⇒ Contain global variables
- heap ⇒ dynamically allocated memory at run time

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Process State ⇒ A process changes states as it executes.



- New ⇒ Process are being created
- Running ⇒ Instruction are being executed
- Waiting (Blocked) ⇒ The process is waiting for some event to occur
- Ready ⇒ Process is waiting to be assigned to processor
- Terminated ⇒ Process has finished execution.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Schedulers ⇒ The main task is to select the jobs to be submitted into the system and to decide which process run.

- Types ⇒ Long term scheduler | Spooler
  - ⇒ Medium term scheduler
  - ⇒ Short term scheduler

☆ Dispatcher ⇒ It is the module that gives control of the CPU to the process selected by the short term scheduler.

☆ I/O bound processes ⇒ An I/O bound process is one that spends more time doing I/O than it spends ~~doing~~ doing computations.

☆ CPU Bound process ⇒ Generates I/O request infrequently, using more of its time doing computations

☆ A system should have both I/O bound & CPU bound equally.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Content Switch ⇒ Switching the CPU to another process requires performing a state save of current process & a state restore of different process. This task is known as content switch.

- When a content switch occur, Kernal saves the content of old process in its PCB & load the saved content of new process scheduled to run.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

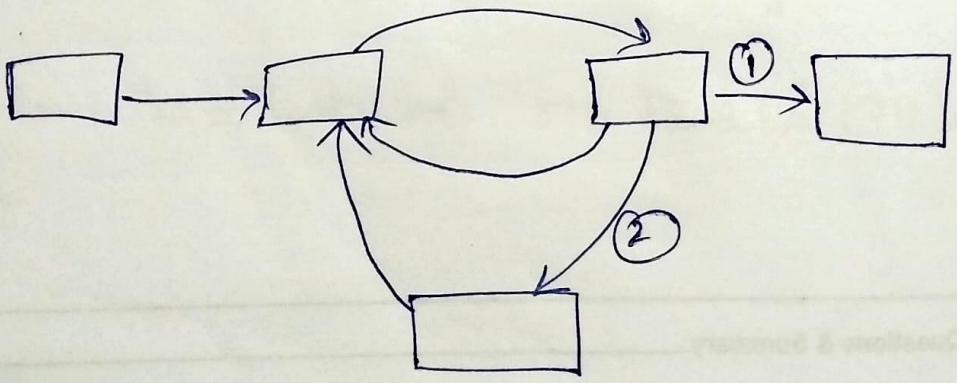
✱ CPU Scheduling ⇒ Process that handles the removal of running processes from CPU and selection of another process on basis of particular strategy.

- Various algo are used for decision making.
- Make system fast
- Min waiting time
- Maximize throughput

✱ Types of Scheduling ⇒

① Non pre-emptive ⇒ Scheduling which takes place when a process terminates willingly.

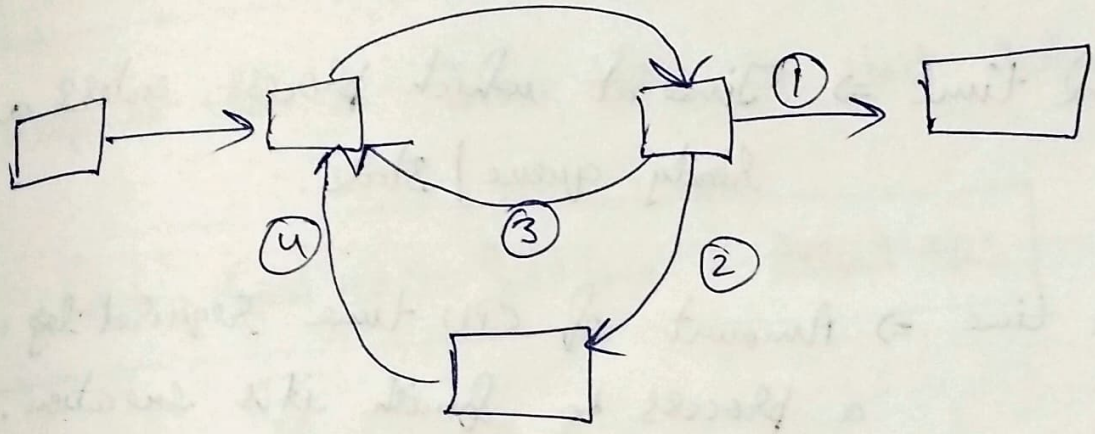
- Process will leave CPU only
  (i) When a process completes its execution.
  (ii) When a process wants to perform some i/o operation



POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(2) Pre-emptive ⇒ Scheduling which takes place when a process switches from running state to ready or from waiting state to ready state.



☆ CPU utilization ⇒ Keeping the CPU as busy as possible

☆ Throughput ⇒ Number of processes that are completed per time unit, called throughput.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ Waiting time ⇒ It is the sum of periods spent waiting in Ready queue.

✡ Response time ⇒ Time takes to start responding, not the time it takes to O/P the response

✡ Terminology of scheduling ⇒

① Arrival time ⇒ Time at which process enters a Ready queue | state.

② Burst time ⇒ Amount of CPU time required by a process to finish it's execution.

③ Completion time ⇒ Time at which process finishes its execution.

④ Turn Around time ⇒ Amount of time to execute a particular process.

Completion time - Arrival time

Waiting time + Burst time

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

(5) Waiting time ⇒ Amount of time a process has been waiting in ready queue.

Turn around time - Burst time

(6) Response time ⇒ Time takes to start responding.

Time at which a process get CPU first time - Arrival time

★ Scheduling Algorithms ⇒

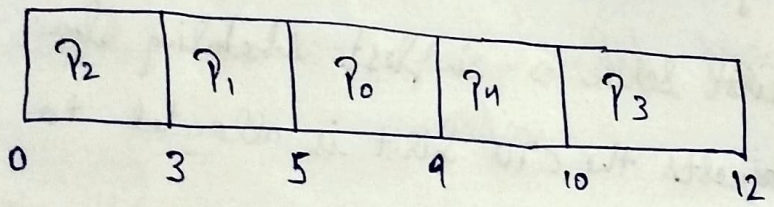
① First come First Serve ⇒ Simplest scheduling algo.

- Process that requests the CPU first is allocated to CPU first
- FIFO queue is used
- Always non pre-emptive in nature

Main Ideas, Questions & Summary:

Library / Website Ref.:-

P.No	Arrival time (AT)	Burst time (BT)	Completion time (CT)	Turn Around time TAT = CT - AT	Waiting time WT = TAT - BT
P_0	2	4	9	7	3
P_1	1	2	5	4	2
P_2	0	3	3	3	0
P_3	4	2	12	8	6
P_4	3	1	10	7	6
Avg				7+4--+7 / 5 ⇒	3+2+7 -- 6 / 5 ⇒



- Advantage ⇒ Easy to understand
  ⇒ Easily implemented using queue
  ⇒ Can be used for background processes where execution is not urgent.

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

. Dis Advantage ⇒ Suffer from convoy effect
⇒ Troublesome for time sharing system.
⇒ Higher avg waiting time

★ Convoy effect ⇒ If the small process have to wait more for CPU because of larger process then this effect called convoy.

(2) Shortest Job First (SJF) ⇒

. In this, the process having smallest burst time requirement will be assigned to CPU for execution of it.

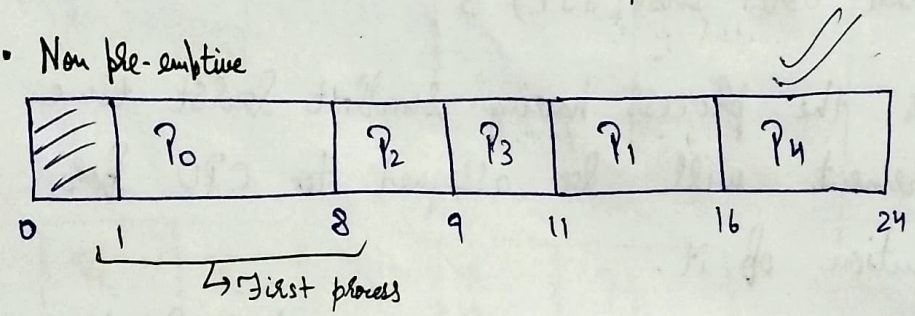
. If there is a ~~tie~~ tie, FCFS is used to break tie

Main Ideas, Questions & Summary:

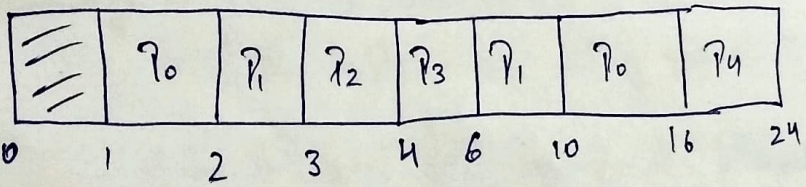
Library / Website Ref.:-

P.No	Arrival time (AT)	Burst time (BT)	Completion time (CT)
P_0	1	~~7~~ 6	
P_1	2	~~5~~ 4	
P_2	3	1 ✓	
P_3	4	2 ✓	
P_4	5	8	

- Non pre-emptive



- Pre-emptive ⇒



POORNIMA

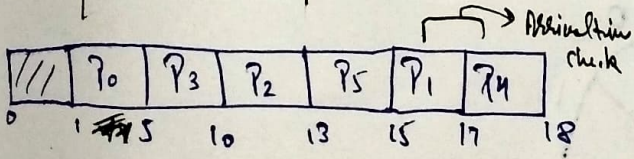
Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

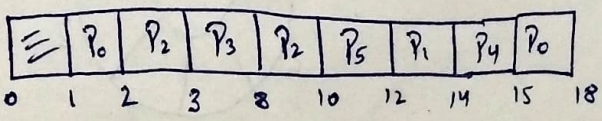
. Advantage ⇒ Guarantees minimal average waiting time.
⇒ Better average response time compare to FCFS.

. Disadvantage ⇒ Process with longest CPU burst time can goes to starvation.
⇒ This algo cannot be implemented as there is no way to known the length of next CPU burst.

(3) Priority Scheduling ⇒

P.No	Arrival (AT)	Burst (BT)	Priority	CT
P_0	1	4 3	4	
P_1	2	2	5	
P_2	2	3 2	7	
P_3	3	5	8 (High)	
P_4	3	1	5	
P_5	4	2	6	





Main Ideas, Questions & Summary:

Library / Website Ref.:-

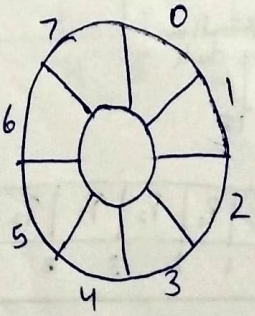
- Advantage ⇒ Allow us to run important process
  ⇒ Support by system

- Disadvantage ⇒ Starvation with smaller priority
  ⇒ No idea of response time or waiting time

✱ Ageing ⇒ A technique of gradually increasing the priority of process that wait in system for long time

(4) Round Robin ⇒

- This algo is delign for time sharing system.
- In this, we don't complete one process & then start another. Instead we divide time of CPU among the process in ready state (circular)
- Time quantum which decides how much time should be given to processes.



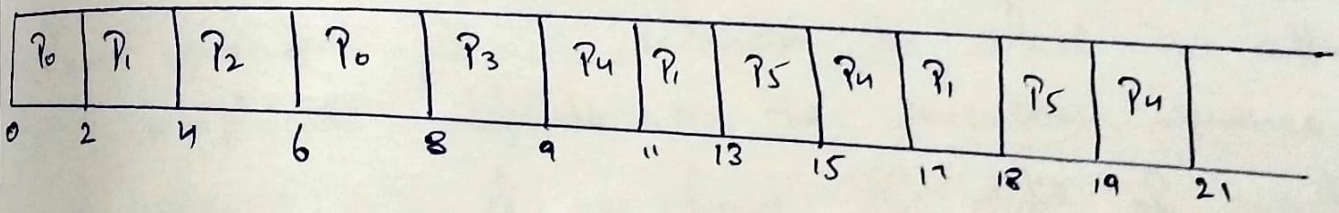
POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

P. No	Arrival (AT)	Burst (BT)	Completion (CT)
	0	4 2 ✓	
P_0		5 3 1	
P_1	1	2 ✓	
P_2	2	1 ✓	
P_3	3	6 4 2	
P_4	4	3 1	
P_5	6		

T.Q (Time quantum) = 2

P_0 P_1 P_2 P_0 P_3 P_4 P_1 P_5 P_4 P_1 P_5 P_4



Main Ideas, Questions & Summary:

Library / Website Ref.:-

- Advantage ⇒ Best in terms of average response time
  ⇒ Works well in time sharing system
  ⇒ Kind of SJF implementation

- DisAdvantage ⇒ Longer process may starve
  ⇒ No idea of priority
  ⇒ ↓Heavily depends upon on time quantum.
  Performance

✡ Multi Level queue scheduling ⇒

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Process Synchronization & Race Condition =>

- In a multiprogramming environment a good no of processes complete for limited no of resources. Concurrent access to shared data at ~~som~~ some time may result in data inconsistency.

P_1

```
i++
--
10
11
--
11
```

i = 10 "
↓
i should see 12 but "

P_2

```
i+1
--
10
11
--
11
```

- Race condition is a situation in which the output of a process depends on the execution sequence of process i.e. if we change the order of execution of diff process with respect to other process the o/p may change.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

✡ General structure of a process ⇒

- Initial section ⇒ Where process is accessing private resources
- Entry section ⇒ Entry section is that part of code where, each process request for permission to enter its critical section.
- Critical section ⇒ Where process is access shared resources.
- Exit section ⇒ It is the section where a process will exit from its critical section.
- Remainder section ⇒ Remaing code
-

```
P(){
    while(T){
        Initial section
        Entry section
        Critical section
        Exit section
        Remainder section
    }
}
```

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Criterion to Solve Critical Section problem ⇒

- Mutual Exclusion ⇒ No 2 processes should be present inside the critical section at the same time, i.e. only one process is allowed in the critical section at an instant of time.

- Progress ⇒ If no process is executing in its critical section and some processes wish to enter their critical section, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next. There should be no deadlock.

- Bounded Waiting ⇒ There exists a bound or a limit on the number of times a process is allowed to enter its critical section and no process should wait indefinitely to enter the CS.

Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Some points to remember ⇒

- Mutual exclusion & progress are mandatory requirements that needs to be followed in order to write a valid solution for CS problem.
- Bounded waiting is optional criteria, if not satisfied then it may lead to starvation.

☆ Solution to critical section problem ⇒

① Two process solution
   (i) Using boolean variable turn
   (ii) Using boolean array flag
   (iii) Peterson's solution

② Operating System solution
   (i) Counting semaphore
   (ii) Binary semaphore

③ H/W solution
   (i) Test & set lock
   (ii) Disable interrupt

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

* Two process Solution ⇒

- In general it will be difficult to write a valid solution in the first go to solve critical section problem among multiple processes, so it will be better to first attempt two process solution and then generalize it to N-process solution.

- Using boolean ~~to~~ variable turn ⇒
- 0/1
-

P_0	P_1

while (1) {
while (turn != 0);
Critical section
turn = 1
Remainder section
}
	while (1) {
	while (turn != 1);
	CS
	turn = 0;
	Remainder section
	}

turn = 0 (initially)

Follow Mutual exclusion

Not follow Progress as we not ask Process to enter into CS or not

Main Ideas, Questions & Summary:

Library / Website Ref.:-

- Boolean array flag ⇒ flag

0	1
F	F

initially

P_0	P_1

while (1){
flag[0] = T;
while (flag[1]);
CS
flag[0] = F;
Remainder section
}
	while(1){
	flag[1] = T;
	while(flag[0]);
	CS
	flag[1] = F;
	Remainder section
	}

Follow Mutual exclusion
Not follow progress

- Peterson Solution ⇒ Follow both ①, ② not ③

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Operating System Solution (Semaphores) ⇒

① They are Synchronization tools using which we will attempt n-process solution.

② A semaphore (S) is a simple integer variable that, but apart from initialization it can accessed only through 2 standard atomic operations:

wait (S)
Signal (S)

③ The wait (S) operation was orginigally termed as P(S) & signal (S) was originally called V(S).

```
wait (S){
    while (S<=0);
    S--;
}
```

```
Signal (S){
    S++;
}
```

Main Ideas, Questions & Summary:

Library / Website Ref.:-

④ While solving critical section problem only we initialize semaphore $\boxed{S = 1}$

⑤ Semaphores are going to ensure Mutual exclusion and progress but does not ensures bounded waiting

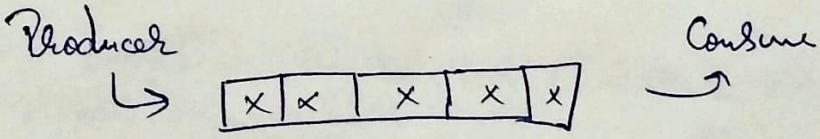
```
Pi() {
    while(T){
        Initial Section
        Wait(S)
        CS
        Signal(S)
        Remainder Section
    }
}
```

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

☆ Producer Consumer Problem | Bounder buffer problem ⇒

- There are 2 process : Producer & Consumer
- Producer produces information & put it into buffer which have n cell, that is consumed by a consumer. Both producer & concusemer can produce & consume only one article at a time.
- A producer needs to check whether the buffer is overflowed or not after producing an item, before accessing the buffer.
- Similarly, a consumer needs to check for an underflow before accessing the buffer & then consume an item.
- Produce & Consumer cannot access buffer together.



Main Ideas, Questions & Summary:

Library / Website Ref.:-

☆ Reader - Writer problem =>

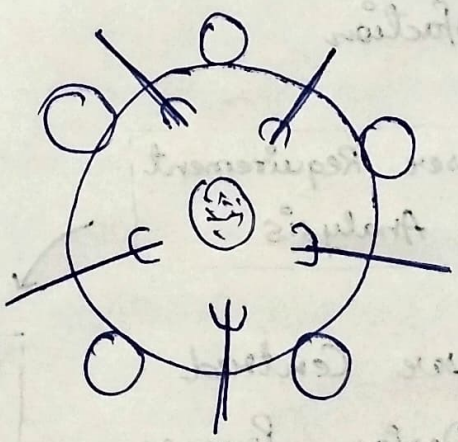
- Suppose that a database is to be shared among several concurrent processes. Some of these processes may want only to read database (readers), other may want to update (that is, read & write) database (writers).
- If 2 readers access data simultaneously, no adverse effects will result.
- If a writer & some other processes (reader or writer) access the database simultaneously, chaos may ensure
- Solution may allow more than one reader at a time but should not allow any writer.
- If writer is writting, no other reader / writer should not allow

POORNIMA

Date	Unit No.	Lecture No.	Faculty	Subject Name	Subject Code	Main Topics:-

★ Dinning Philosopher Problem ⇒

- Consider 5 philosopher who spend their lives thinking & eating.
- Philospher share a circular table surrounded by 5 chairs, each belonging to one philospher.
- In center of table is a bowl of rice, & the table is laid with 5 single chopsticks.
- When a philospher thinks, she does not interact with her colleagues.



- Solution ⇒ Allow philospher to pick her chopstick only if both chopsticks are available.

Main Ideas, Questions & Summary:

Library / Website Ref.:-