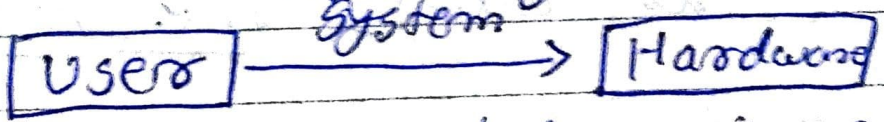
OS

→ Trending

1) What is an operating system? Explain it's structure with figure

- The Operating system is a program that acts as an intermediary between user of a computer and computer hardware
- For working the hardware part we need to install some software to run that hardware
- It acts as interface between the user of the computer and computer hardware



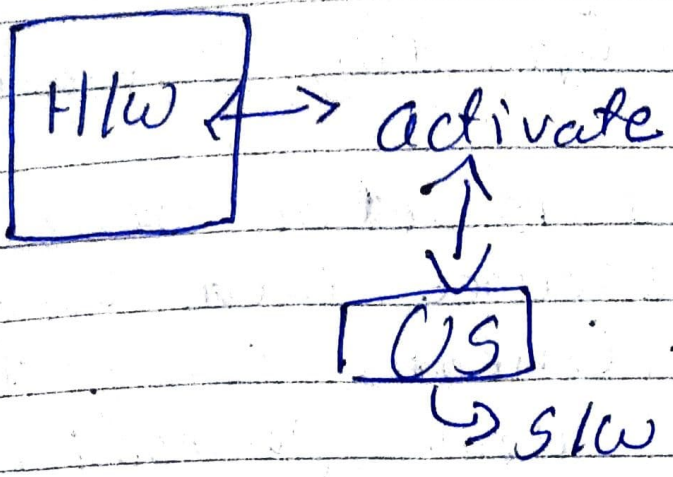
- It is nothing but a program.
- It is program that controls the execution of application program
- Acts as an interface between application and hardware

→ Main objective of an OS:
- Convenience
  - efficiency (system efficiency)
  - Ability to solve problem or Ability to evolve

→ OS services
- It can provide program development
- It can provide service to program execution
- It help to access the I/O devices
- It control's file system access
- It also provide error detection and response (quick response)

It is the software which unables the hardware.

Computer



- There is any ~~gujet~~ gadget here had taken so in that you are having the hardware
- To ~~hardw~~ activate these hardware components to activate means to work these hardware components we need operating system that is nothing but it is a Software.

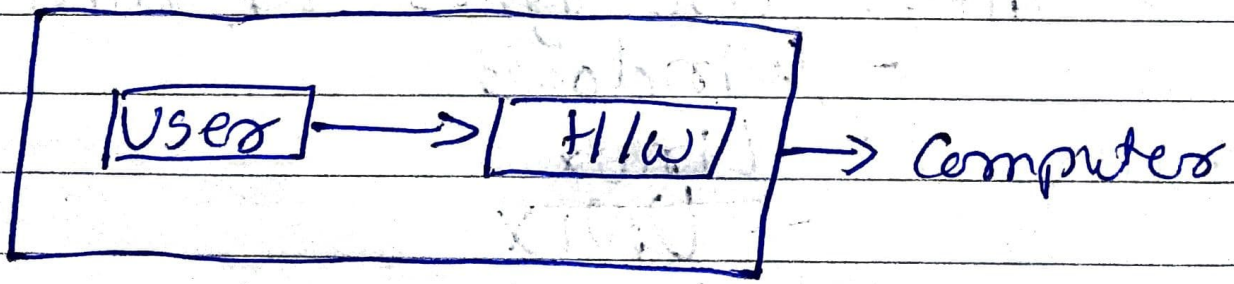
→ There are types of different OS
- Windows
- Linux
- UNIX
- Ubantu
- Android

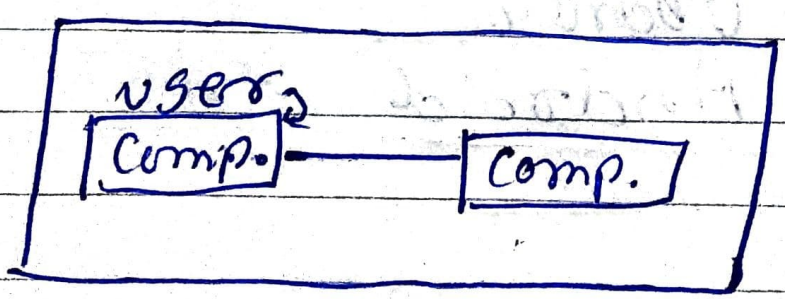
→ Role of Operating operating system

- A computer is set of resources. These resource were used for moment (means coping the data, storage and processing data.

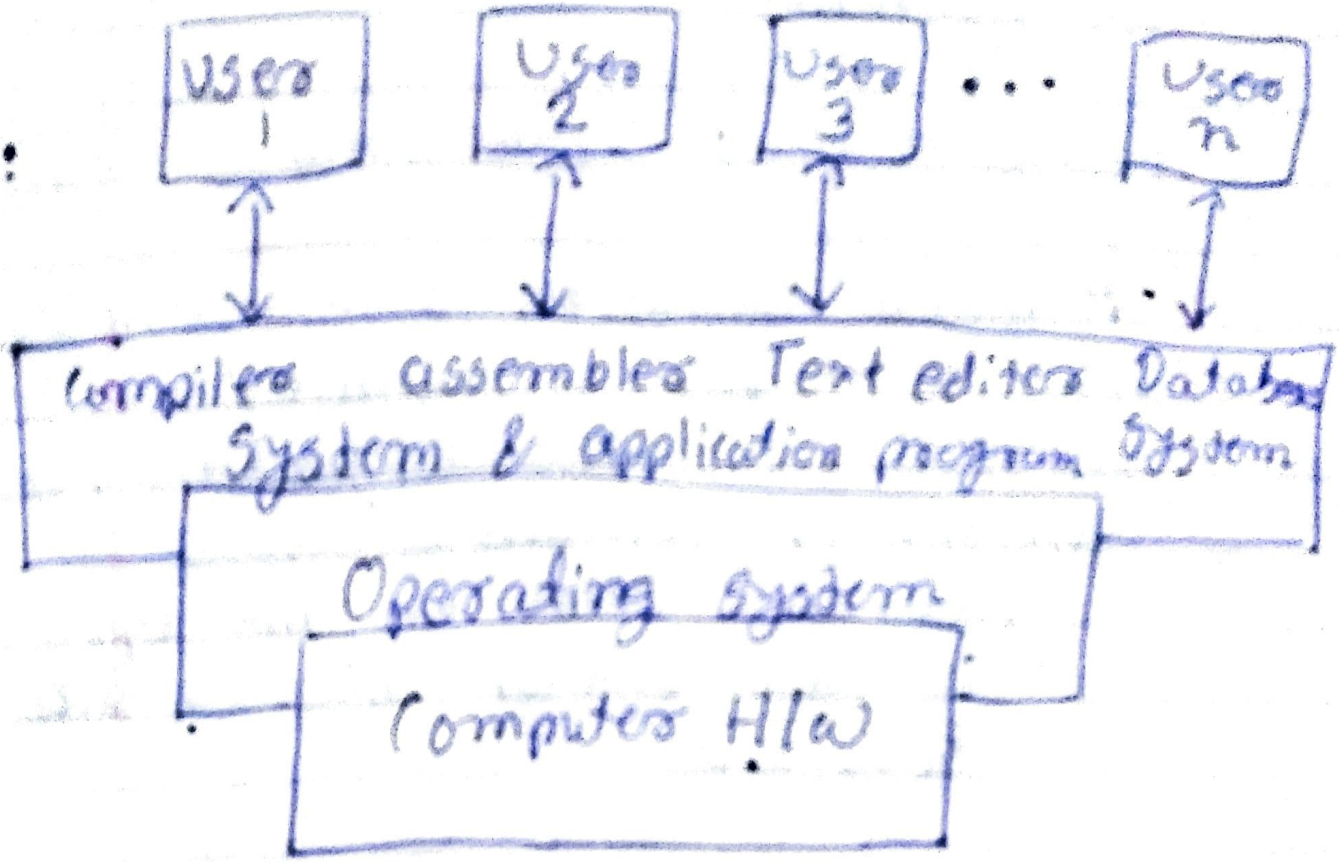
- The operating system is main responsible for managing these resources.

→ Computer system components:
  → Hardware (CPU, memory, I/O)
  → Operating system
  → Application programs.
  → Users (People, machine, other P





→ Abstract view of system.



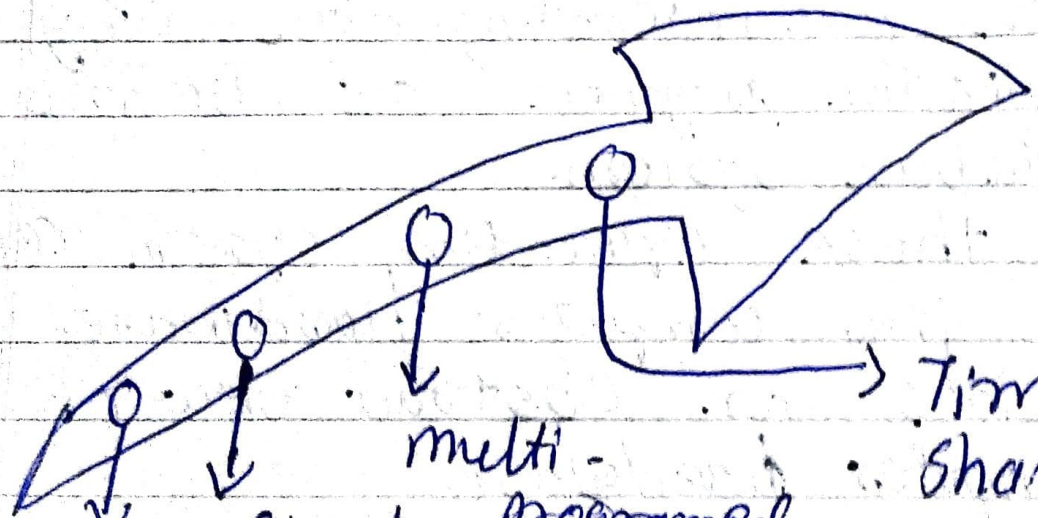
- Users are interacting with any of application program or else the database system.
- So these Operating system consists of the computer Hardware.
- So it's an abstract view of the computer hardware
- The operating system is acts as an intermediate b/w the H/w and application program (compiler, text editor, assembler)

- The user may be people, computer or machines.

→ Evolution of operating system:

- A major operating system will evolve over time for a number of reasons. So each time it's upgrating from number of reasons
  - Reasons may be h/w updates, or new type of H/w or new services

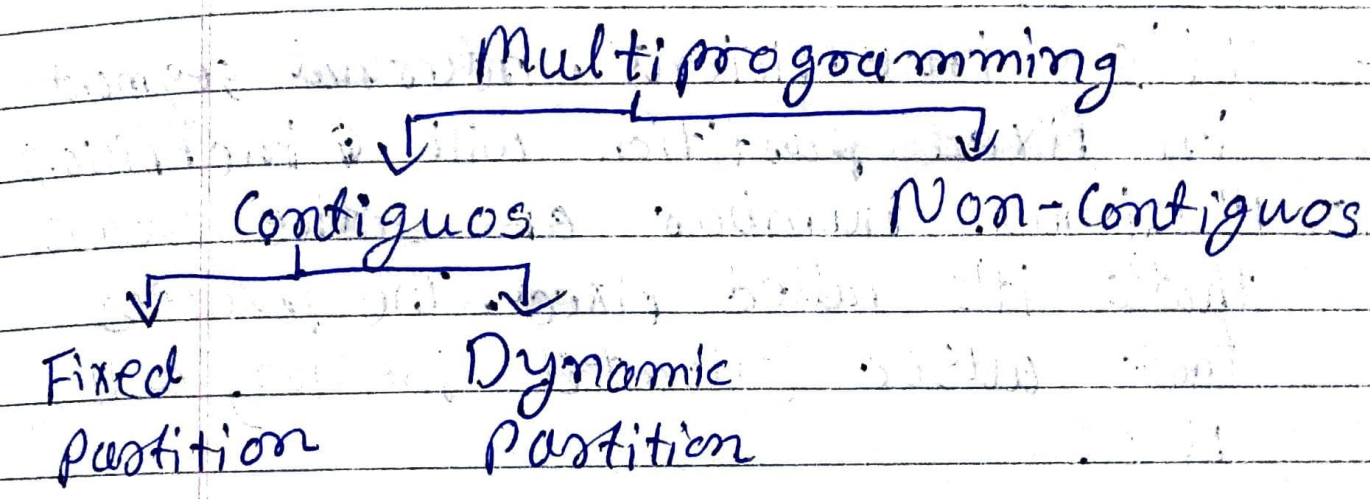
- Stages included:



2) Explain how multiprogramming can achieved by
1) Fixed partition
2) Variable partition

-> Fixed Partition

- Multiprogramming memory manage again divided into two parts



- Fixed partitioning
- It's also know as static partition
- The main memory divided into ~~math~~ number of static portion at system generation time.
- This allocated in equal size partition or unequal size partition

→ Pros of Fixed partition

- It is simple to implement.
- Little operation system overhead because it is static partition/fixed partition so little overhead will be there

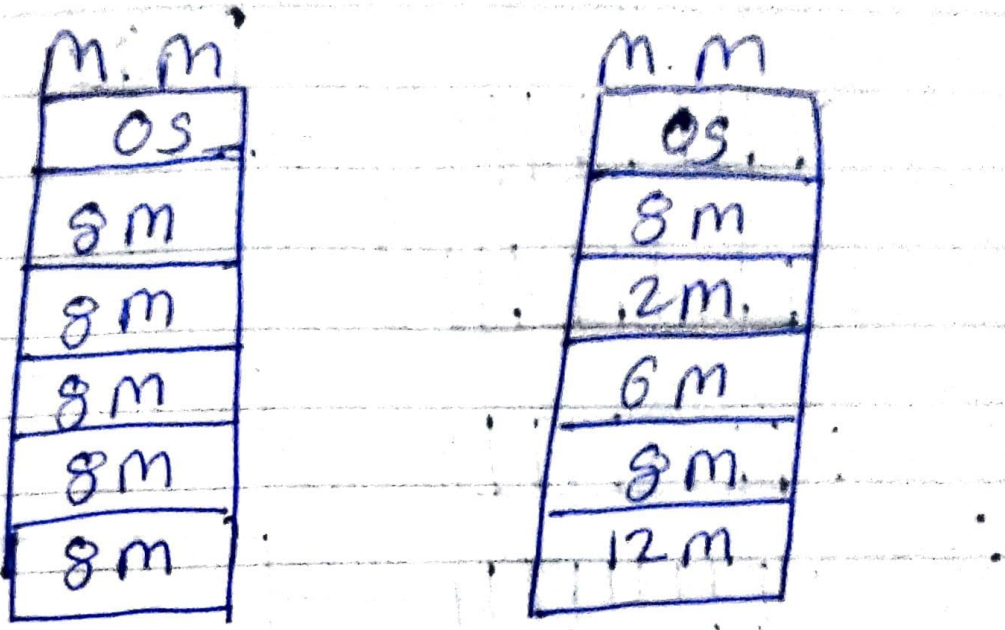
→ Cons of Fixed partition

- It's inefficient. Internal segment in fixed partition will be inefficient
- Maximum number of active process that's it's called fixed. The process that attsoo allocated memory will fixed

→ Placement algorithm

- One process queue per partitioning
- Single queue per partitioning

- Example



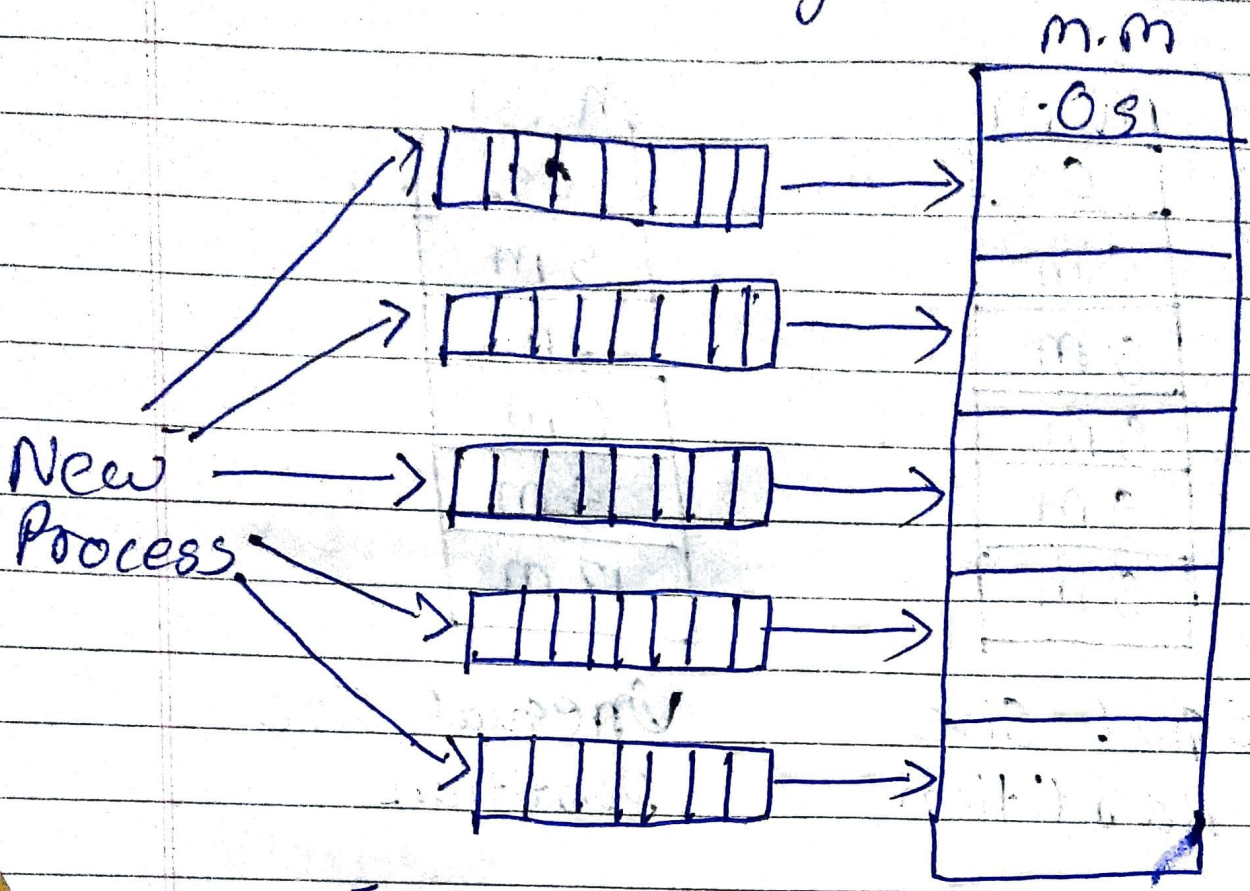
Equal Size Partition

Unequal Size Partition

- Equal size partition:
  - 64MB memory can be allocated like 8-8-8...so on.

- Unequal size partition:
  - It divided into unequal sizes but It's fixed only. means one process can enter into only one slot/segment.

→ Fixed partitioning placement algorithm



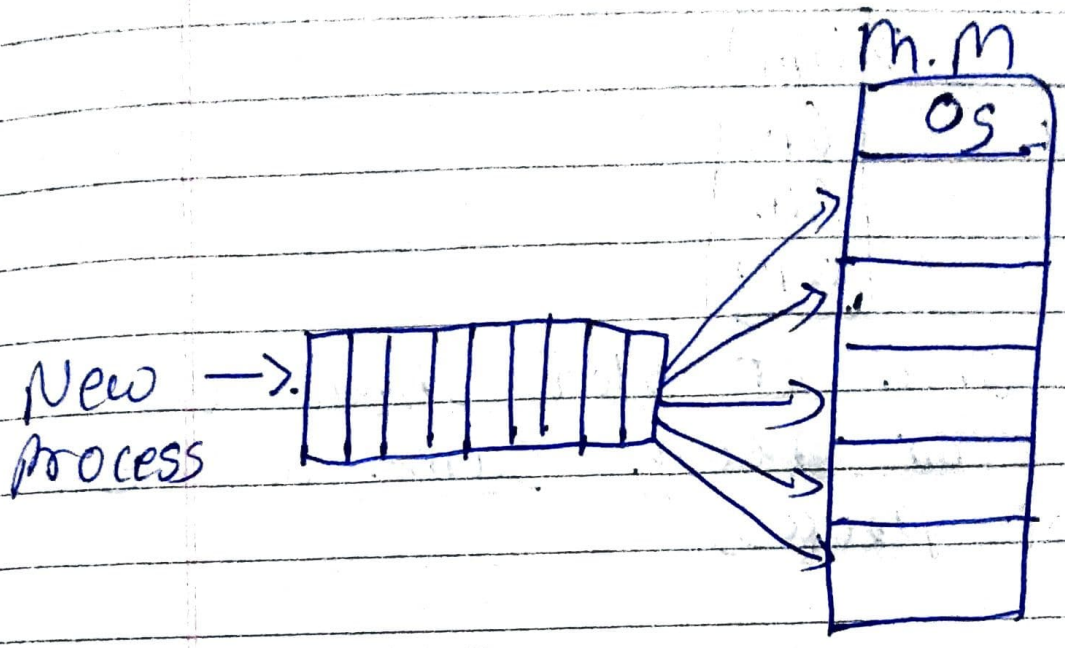
+ [One process queue per partition]

- A new process divided into multiple queues
- each queus will occupied one slot / one segment

→

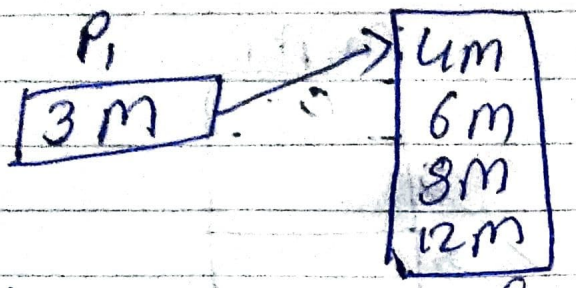
→ Single queue per partition

- A new process having a single queue will occupied all the the partition



2) Variable partition

- Variable partition also knowces dynamic partition
- The partition are created dynamica in memory is called variable partition.
- Each process is loaded into a partition of exactly the same size of that process.
- Suppose the process is 3M sized it's store only 3MB of memory
- In that you have to chosee the main memory which is very close to 3M.



- hear size of slot should be exatly or nearly same to the size of process

→ Placement algorithm

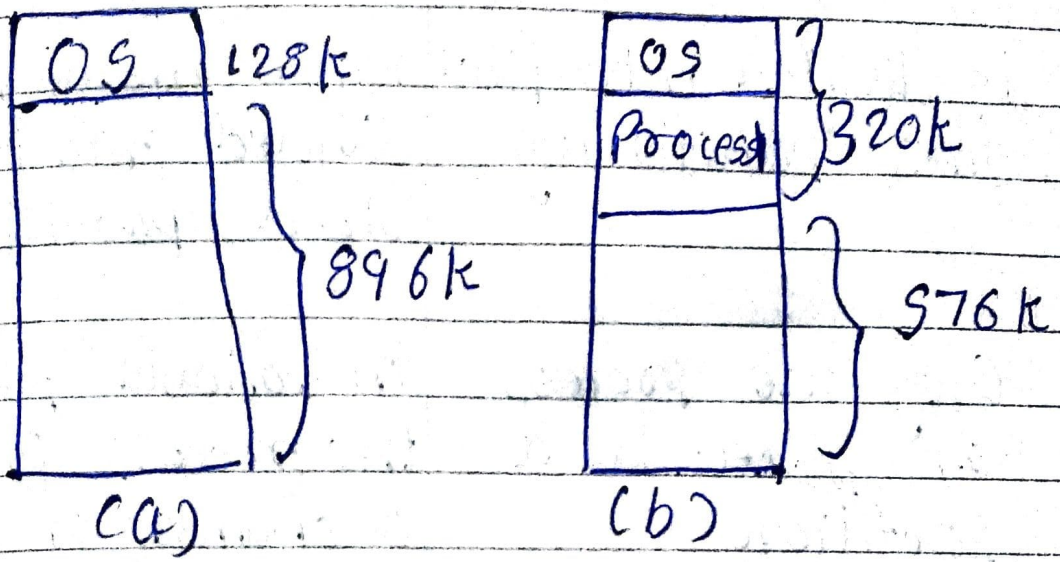
- First fit :
- Best fit
- Woost fit

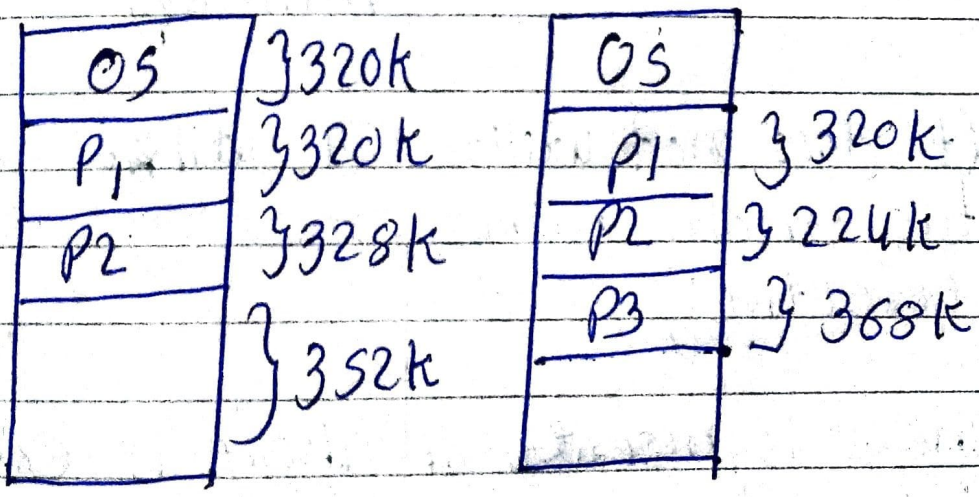
- First fit : The ifirst hole that is big enough is allocated to progoram

- Best fit : The smallest hole that is big enough is allocated to program

- Woost fit : The largest hole that is big enough is allocted to program

- Example





→ Difference between Fixed and Variable partition

Fixed Partitioning	Variable partitioning
- In multi-programming with fixed partitioning the main memory	In multiprogramming with variable partitioning the

Fixed partitioning	Variable partitioning
is divided into fixed sized partitioning	main memory is not divided into fixed sized partitions.
- Only one process can placed in a partition	In variable partitioning the process is allocated a chunk of free memory
- It does not utilize the main memory effectively	It utilizes main memory effictively
- There is presence of Internal fragmentation and external fragmentation	There is external fragmentation
- Degree of multi-programming is less	Degree of multi-programming is higher
- It is more easier to implement	It is less easier to implement

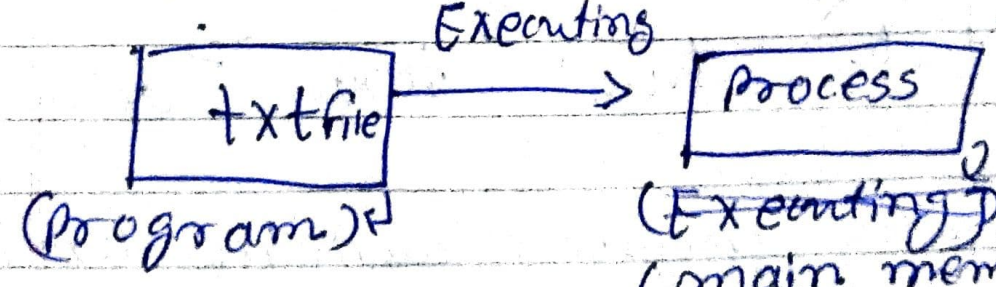
Fixed Partitioning	Variable partitioning
- There is limitation on size of process	There is no limition of size of process.

3) What is process ? Differentiate between process and program Explain process control block diagram

OR

Define process. Explain the process state diagram in details.

- Process is programing execution.
- Any program, function or module that is exequting that called as a process.

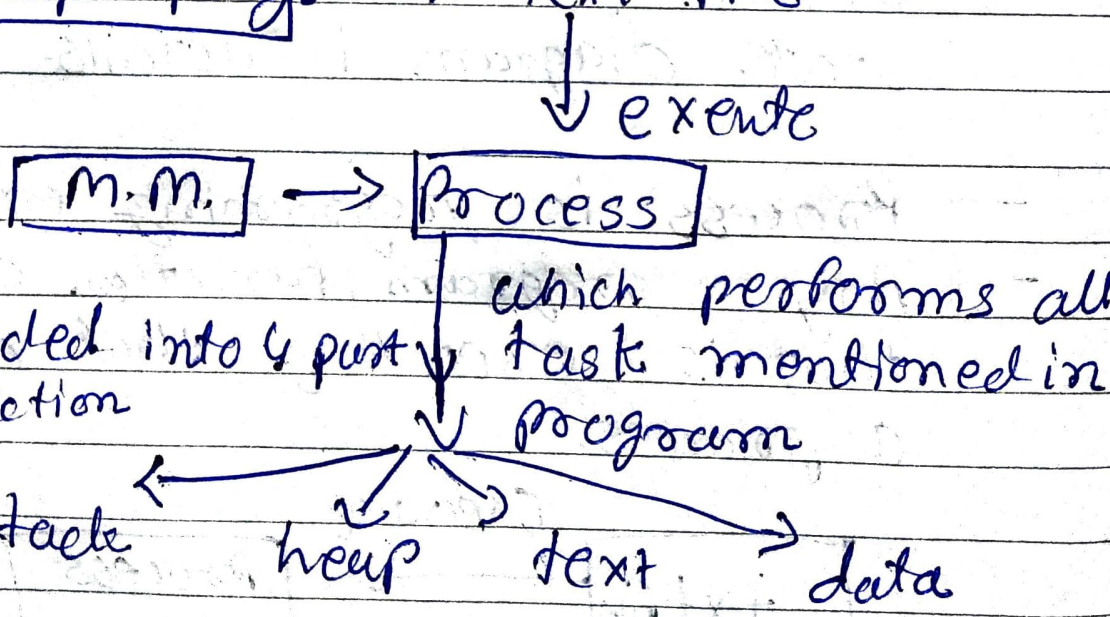


- User writing a program in txt file it wants to exented and

, enters into main memory then it called it as a process.

- A process can be define as:
  - A program in execution
  - It is an instance of running program
  - The entity that can be assign to, an execute on a processor

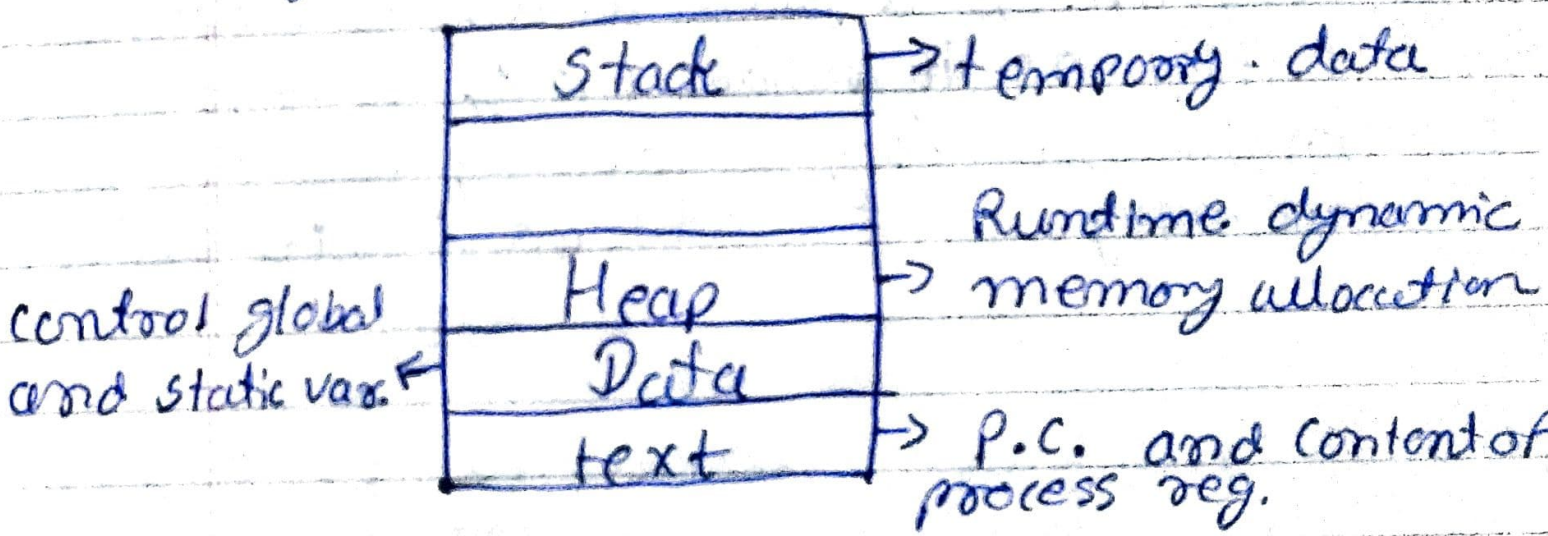
→ Comp. prog. in text file



- Computer program write in text file when you execute a program which becomes a process which in main memory.

- A process do all the tasks which are mentioned into a program
- It's further divided into 4part or section
  - Stack
  - heap
  - text
  - data

→ layout the process inside memory



- Stack contain temporary data such as function, parameters, local variables
- Heap dynamically allocated memory to process during runtime
- Data contains global variables and static variables

- Text contain program counter and content of process register.

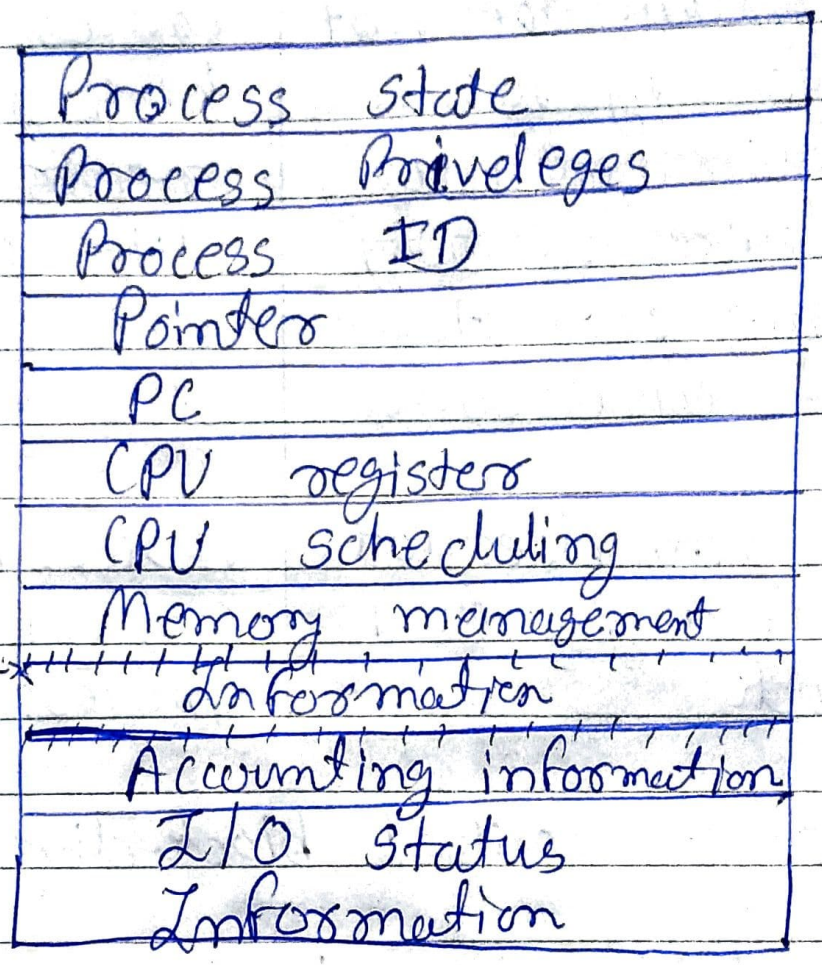
Key point	Program	Process
Definition	Program contains a set of instructions designed to complete a specific task	Process is an instance of an executing program
Memory Store	Program is a passive entity as it resides in the secondary memory	Process is a active entity as it is created during execution and loaded into main memory
life	Program exists at a single place and continues to exist until it is deleted	Process exists for a limited span of time as it gets terminated after the ~~task~~ completion of task
entity type	Program is a static entity	Process is a dynamic entity

Key Point	Program	Process
Resource	Program does not have any resource requirement, it only requires memory space for storing the instruction	Process requires high resources, it needs resources like CPU, memory address, I/O during its lifetime
Control block	Program doesnot have any control block	Process has its own control block called Process control block

→ Process control block diagram

- Process control block is data structure that main by the OS for every process
- There are server process which main Each of these main by the PCB (Process control block) with their Id
- The PCB identified by an integer that is process ID (PID)

- PCB keeps all the information needed to keep track of a process.



- Process state indicate current state wheather it is running state, ready state, waiting state.

- Process privileges allowing or disallow access to system resources. To used the system resources that process need to permission that also

Store in PBC.

- Process ID is unquick id form which each process will be stored in operating system. Based on this id we can identify the process.

- Pointer always point to parent process

- Program counter address the next instruction to be executed in process. The next process's related address will store in program counter

- CPU register used to store for execution. Storing data temporary

- CPU Scheduling means decides the process priority. Process priority will store.

- Memory management information
It includes a information of peges, page table or memory limit.

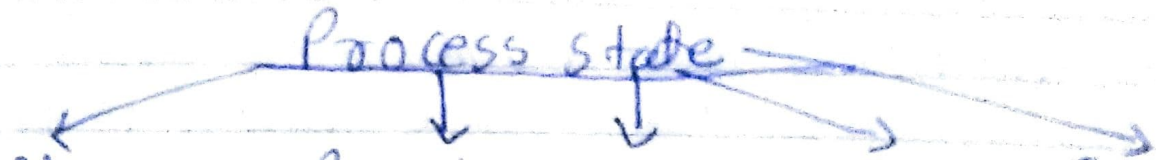
- Accounting information comes under a amount of CPU used for process execution, time limit, execution id, etc.

- I/O status information includes list of i/o devices allocated to process.

→ Explain process state diagram in details

- A process executes whenever it is chaning in states.

- The program we taken (can that program is written in text file if that program is executes it stores in main memory so there it is chaning it's States (like New running, waiting ready, terminated) while executing



- It's chaning it's behaviors like human being. Like I have diffreent roles to perform like means state is changing so when I am doing a job I have to do a job work that is process state. When I come to home I will be like a mother for the kids

- So single process whenever the program that you're exeuting so I'm single person chaning behavior according to the circumstance or the priorities I have to change that.

- As same program process also have some state on based of behavior so flow that shows in which it should be there.

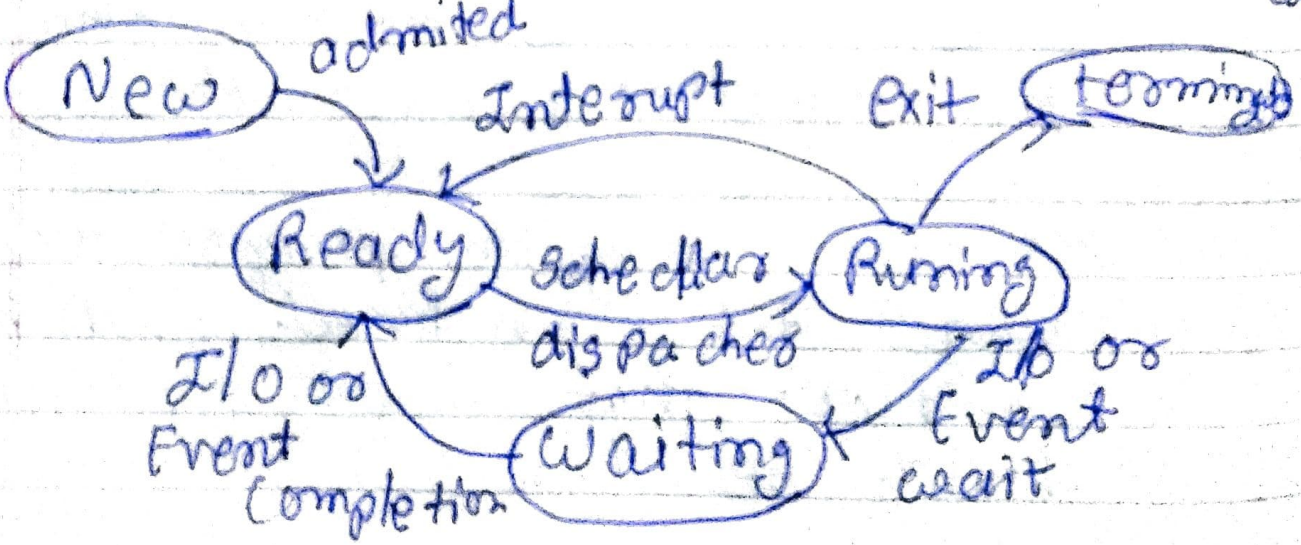
- As New state: whenever process being created that you called it as new state

- Running state: Instruction are being executed

- Waiting state: when process is waiting for some event to occour

- Ready state: whenever process event is occure the waiting state is over the process waiting is over it should be assign to processer.

- Terminated state: The process has finished execution then it is terminated



- The process is admited then it goes or chemeges state to Ready state

- when an interrupt is generated running to ready state occure

- Whenever schedule is dispatcher the process become to Ready to Running

- Suppose an event is occure (ie. I/O, external signal) it has to enter into a waiting state.

- Whenever operation is completed then it goes to ready state.

- Everything a program is over it comes to the exit state.

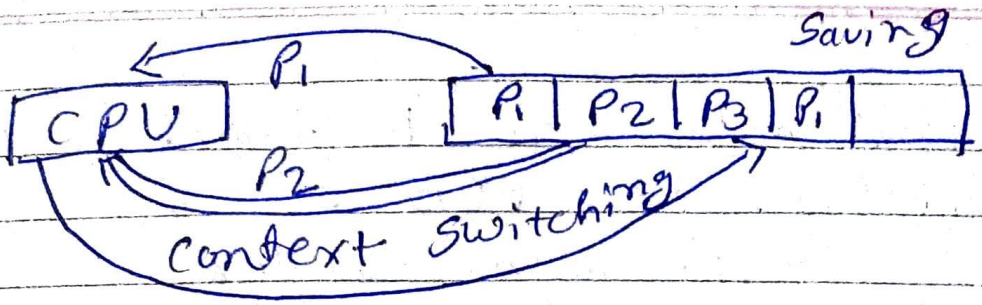
4) Explain context Switching. Discuss performance evaluation of FCFS (First come first serve) & RR (Round Robin) scheduling

OR

Explain any three Disk arm scheduling algorithms with suitable illustrations

→ Context Switching

- Context Switching also know as process switching.
- Context Switching means se suppose a CPU switches from one process to another another process is know as context switching
- For example CPU working on one application and any high interrupt is accure CPU wants to switch from that process to another high request process that Switching technology call it as "context switching"
- Switching of CPU to another process means saving the state of old process and loading saved state for new process



- In above example the P_1 process is first come & and it's running in the CPU
- but P_2 process is comes which having higher priority interrupt so now CPU will saving the state of P_1 and then P_2 process will be executed
- This state (the remaing statements) save in the PCB. It store address of the P1 process.

- In context switching the process is stored in PCB to serve the new process.
- So that old process can be resumes from the same part it was left.
- To make context switch time to be less, register which are the faster access memory are used.

- Information to be stored in context switching in PCB is:
  - Program counter
  - Scheduling information
  - changed state
  - Accounting information
  - Base and limit register

→ Scheduling Algorithms

- The scheduling algorithms are used based on the user's request.
- The different types of scheduling algorithms are as follows
  - First come first serve (FCFS) scheduling
  - Shortest Job first (SJF) scheduling
  - Priority scheduling
  - Round Robin scheduling.

- As the name suggest First come first serve means the process comes first that will executes first

- Shortest job first means the process which

- Burst time: the total time that CPU taking for process to run/control

is shortest that will executed first. The time of execution is shorted.

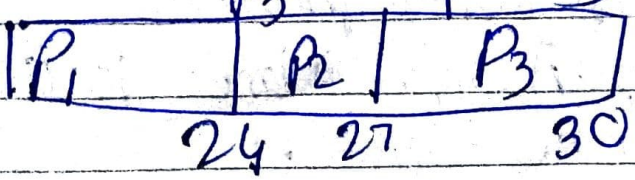
- The priority of process is highest that will be executing first.

- Round Robin will rotating the process in the circular fashion. here we assign a certain time in that process has to be over in that time suppose it's no the time is over it's goes to the next process. After completing a ~~full~~ process it will executes the again first process. The process disptach in FIFO manner

(1) First come first serve:

eg

Process	Burst time
P_1	24
P_2	3
P_3	3



avg waiting time $= \frac{0+24+27}{3}$

$= 17$

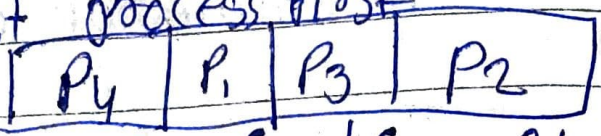
in order, if order change avg time chang

disadv. or limitation
↳ we know the waiting time first (execution time should be know before) to minimise

(2) Shortest job first: best for waiting time and select smallest process first

eg

P_1	6
P_2	8
P_3	7
P_4	3



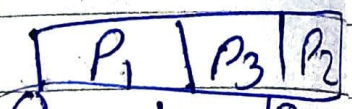
avg waiting time = $\frac{0+3+9+16}{4}$

= 7

(3) Priority scheduling

eg

process	Burst time	Priority
P_1	10	1
P_2	1	3
P_3	2	2



$= \frac{10+12+0}{3} = 7.$

- int number assign to the processes priority.
- SJF also priority scheduling but in reverse order. It's non preemitive
- Problem: starvation: low priority process may never execute in the CPU
  Soln: Aging: as time progress increase the priority process

(4) Round Robin: Assign time is know as quantum or time-slice.
- If process don't end in quantum time then it will execute next process and process will in next step

quatum time = 20

process	Bursttime	
P1	53 ⇒ 33	13 = 0
P2	17 ⇒ 0	
P3	68 ⇒ 48 ⇒	28 ⇒ 8
P4	24 ⇒ 4 ⇒	0

P1	P2	P3	P4	P1	P3	P4	P1	P3
0	20	37	57	77	97	117	121	134

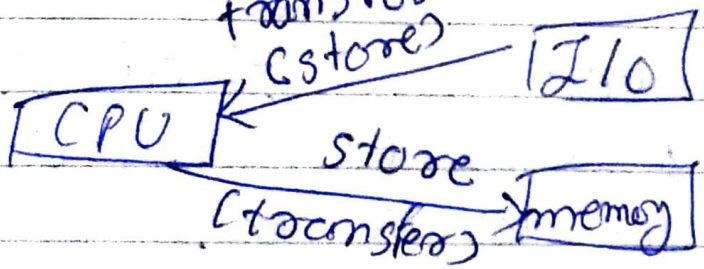
154 62

- DMA controller
  Direct memory access

CPU . . . I/O Peripherals } main components

memory

- when I/O device want to access memory without interactive with CPU is called direct memory access



It required 2 cycles. It's slow
- Make data transfer rate fast.
- So for that we want to transfer that directly
- here every thing done by CPU.
- CPU is master
- It has to maintain everything
- DMA is design by Intel to transfer data at fastest rate

- It's allows device to transfer data directly to/from without CPU involvement

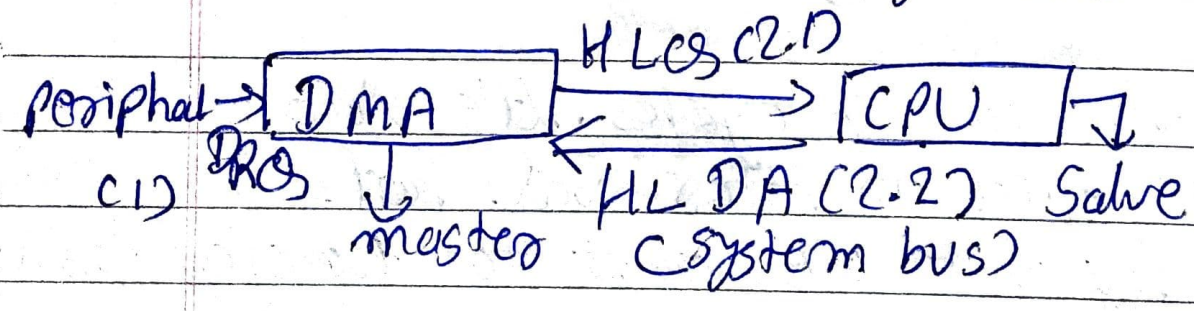
— How DMA operation are performed?

1 Device (peripheral): wants to send data to memory. first device has to send DMA request ~~to~~ (DRQs)

↓

DMA controller

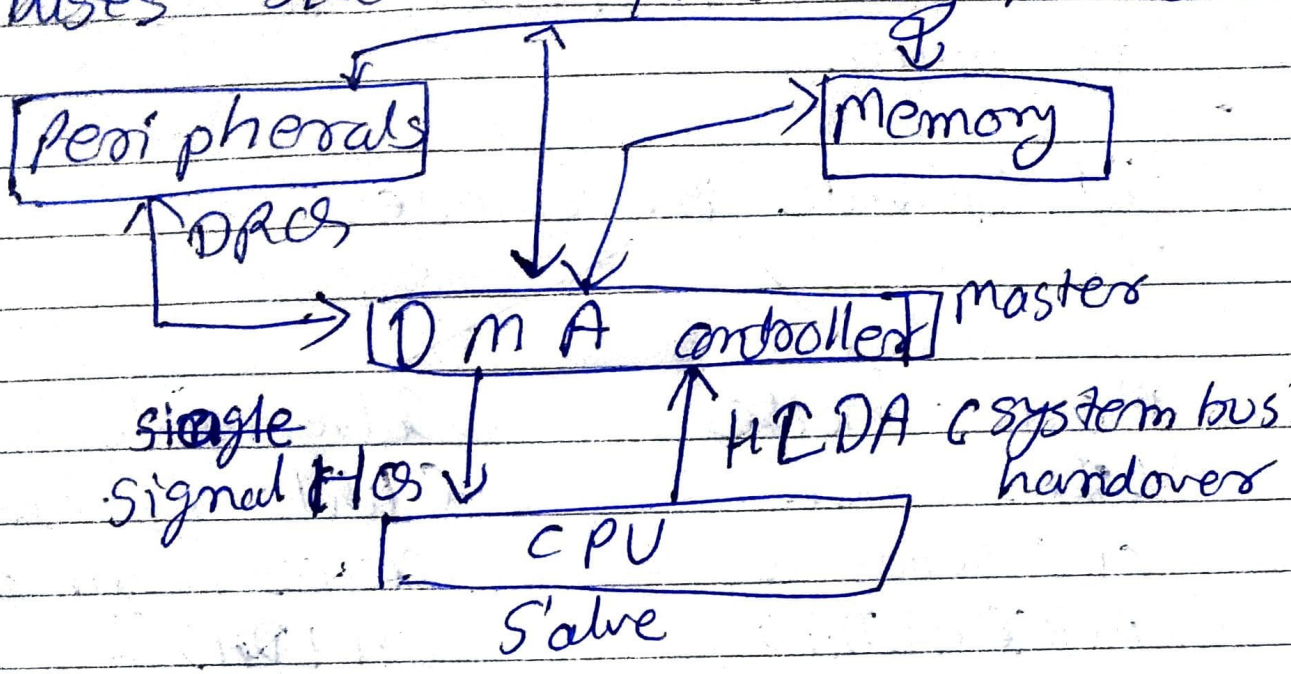
2 DMA controller send → HLDs (holdreq to CPU & waits for the CPU to send HLDA (hold acknolgement)



3 The CPU ~~gi~~ leaves the control over bus and ack. the ~~HDI~~ HLDA single to DMA controller

4 Now CPU is in Hold-state, DMA controller has to manage operation over

buses b/w CPU, memory, I/O device



- Feachure

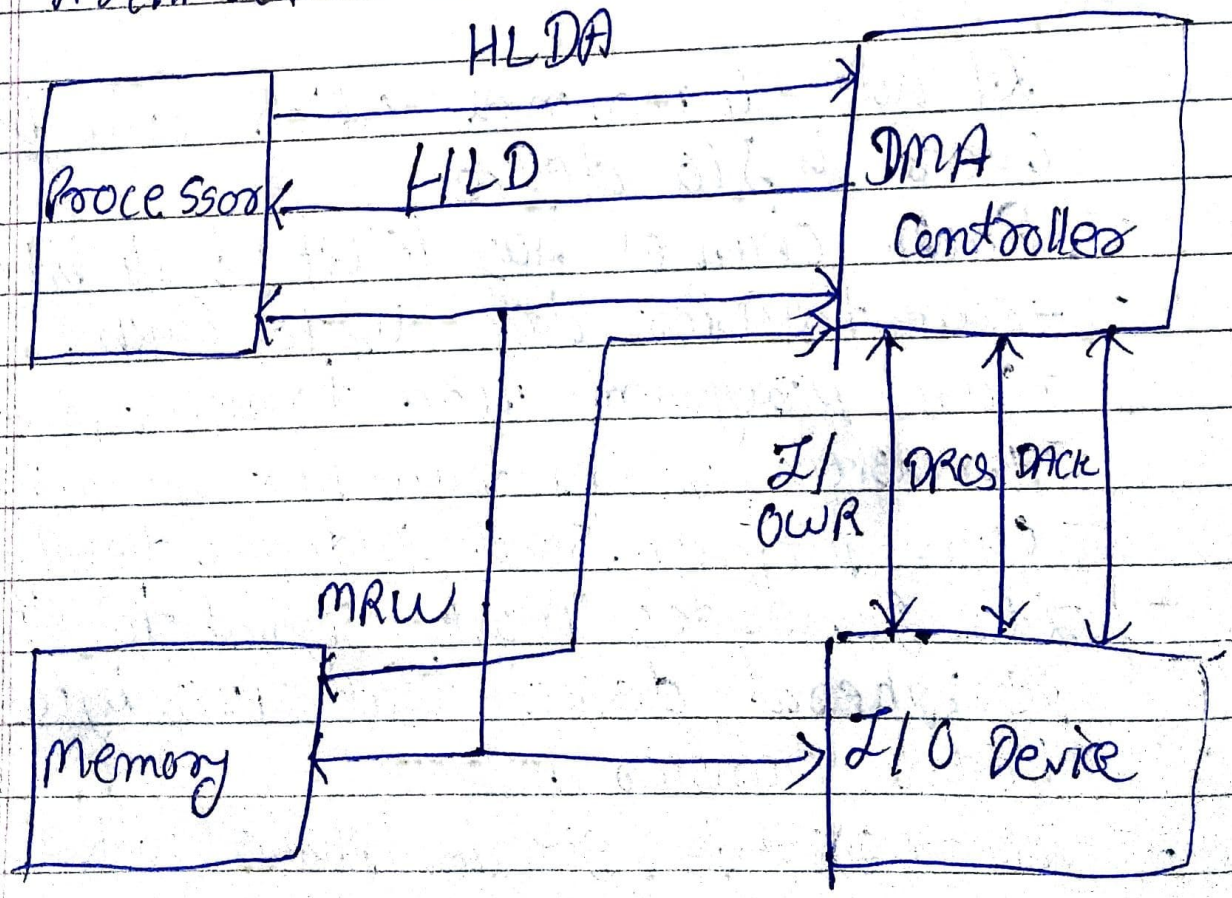
- It has 4 channel which can be used over 4 I/O device
- Each channel has 16 bit & 14 bit counter,
  - can transfer data up to 64kb,
  - can program read transfer, write transfer, verify transfer operation,
  - can program mod independently
- It generates Mark signal to the peripheral device that 128 bytes have been transfered.
- It required single phase lock.

- It requires frequency range from 250Hz to 3MHz.
- It operate in 2 modes

Master mode
(It's when CPU sends a ACK [HLDA])

Salve mode
(While not receiving access from CPU [HLDA])

- Architecture



→ File allocation method

- File allocation methods: Allocate space to the files so that disk space is utilized in efficient manners.

- Factor to consider while allocating file
  1- Processing speed of sequential access and random access of file.
     - If the space of file allocation is after that sequential and random access that too slow then that allocation consider to inefficient
  2- Ability to use use multi sector and multi track transfer.
     - After allocation it should be used multi sector and multitrack transfer
  3- Disk space utilization
     - It means file need to be allocated such that maximum number of file can be stored in a disk.

Disk → 2 block

500k | 400k

350k

have to allocate block of : 350k
So here 2 option are there 500k and 400k.

- if 500k choose 110k memory waste
  if 400k choose 10k memory waste
- So choosing better option in which disk space is better utilized

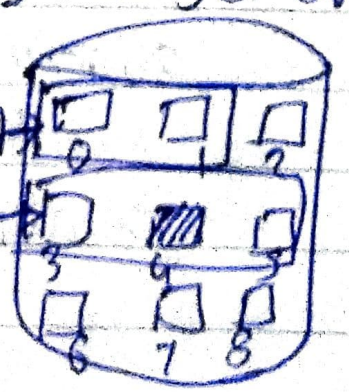
4) - Main memory requirement
It should be allocated such manner that memory requirement is less or lesser

→ Contiguous Allocation:

- Each file occupies a set of contigious address on disk

Directory

File	start	len
A	0	2
B	3	3



Directory

File name

- So for A it's start with memory add 0 and goes upto 2 block (length)
- And B it's start with 3 and go 3 block up

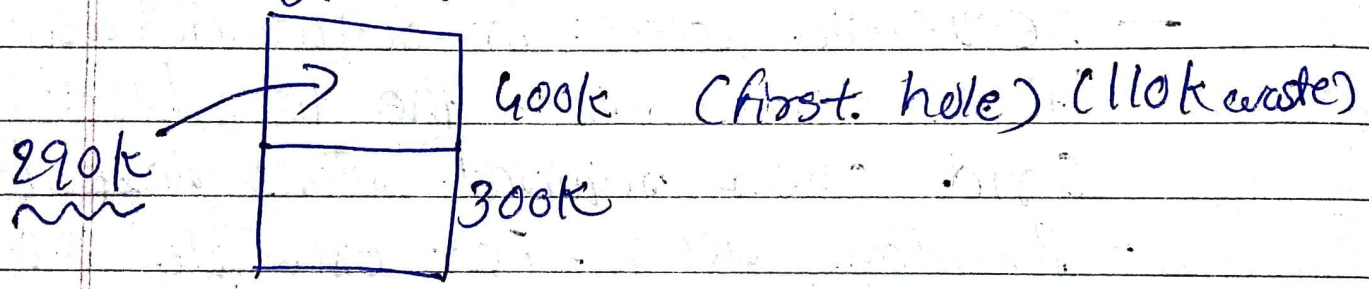
- It's in linear ordering
- Location of file is defined by the disk addr. of the first block and it's length.
- Both Sequential and direct/random access are supported
  - Sequential access: In sequencial access it ~~read~~ remember the last block and when required reads next block.
  - Random access: For example it's have 'B' ~~to~~ for start the reading and i^{th} (location for) record to find So address location will be "B+i"

- For example (in above example) here record of file B so it will be ~~B+3~~ 3 + 0+1 (first record second record = 4 so it will directly access 4 block

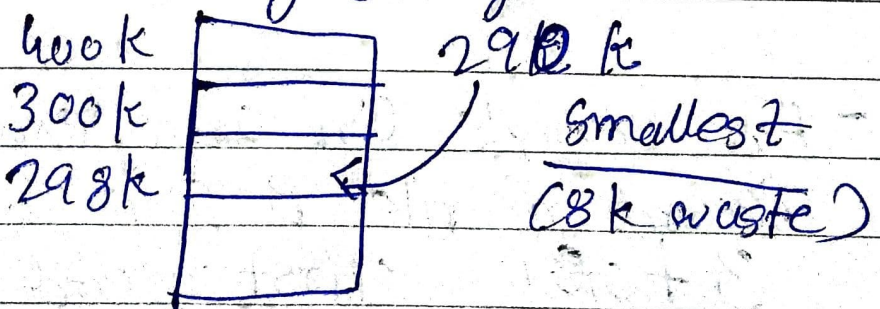
- dis advantage:
  - Finding space for new file is difficult bcz we have to keep track of free block
  - External fragmentation

- Contigeous memory allocation technique is dynamic storage allocation

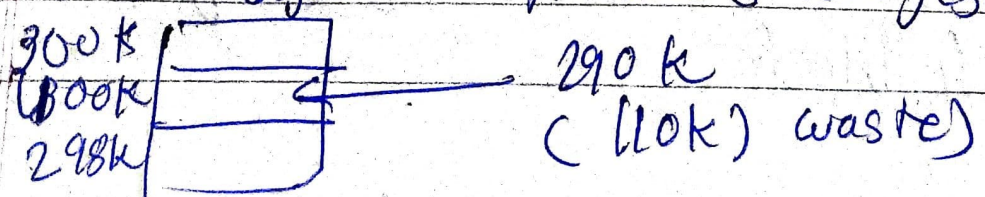
(i) First-Fit : allocate first hole that is big enough



(ii) Best fit : Allocate the smallest hole that is big enough



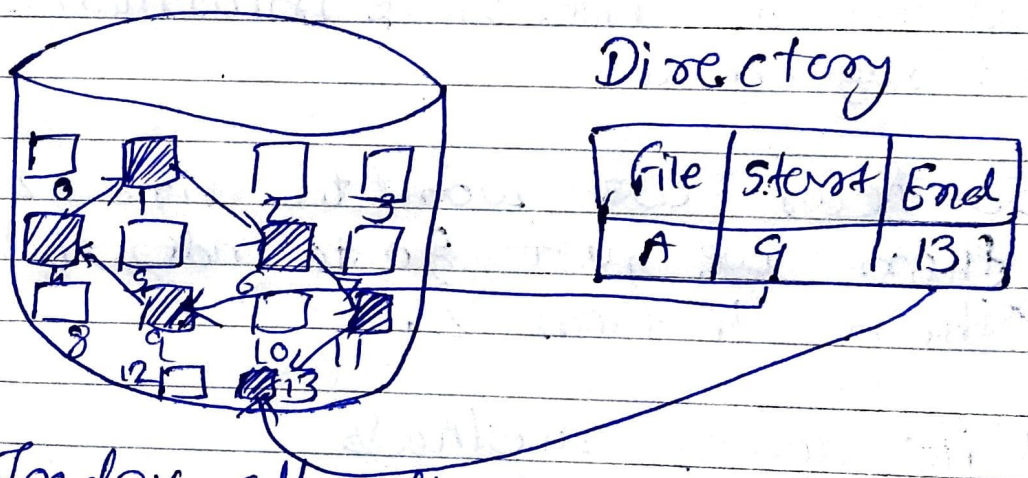
(iii) Worst fit: Allocate largest hole



- Suffer from external fragmentation

(iii) linked allocation:

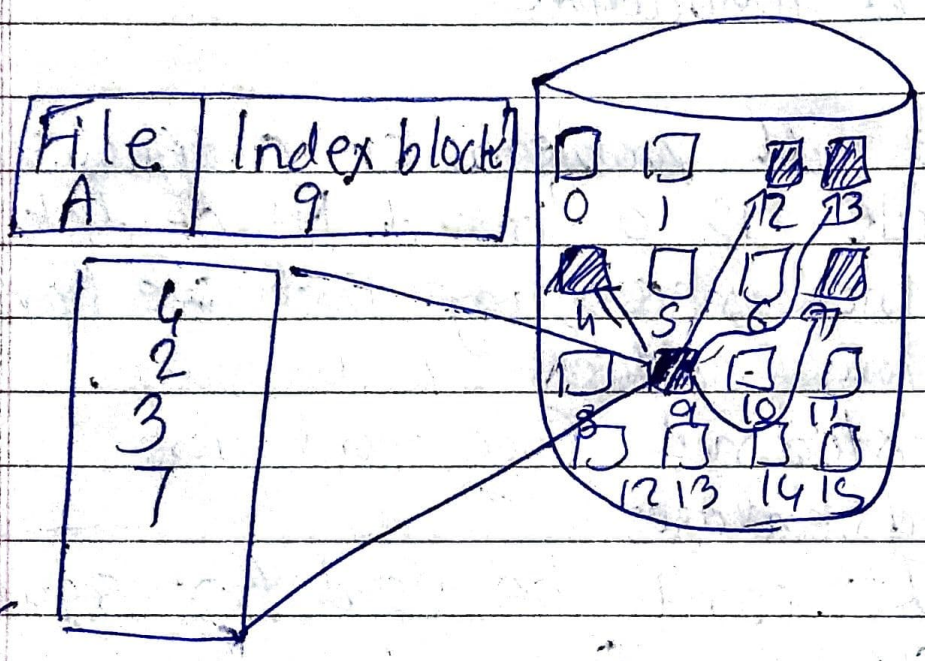
- Solves all problems of contiguous allocation. Each file is a linked list of disk block
- It's not necessary that are have contigious address to be save.
- No external fragmentation
- disadvantages:
  - It can't be used for sequential access file.



(iv) Index allocation:

- It solve problem of linked allocation (NO Random access). In this all the pointers are brough together into one location called

index block. Each file has its own index block



↳ It means file store information of this storage area.

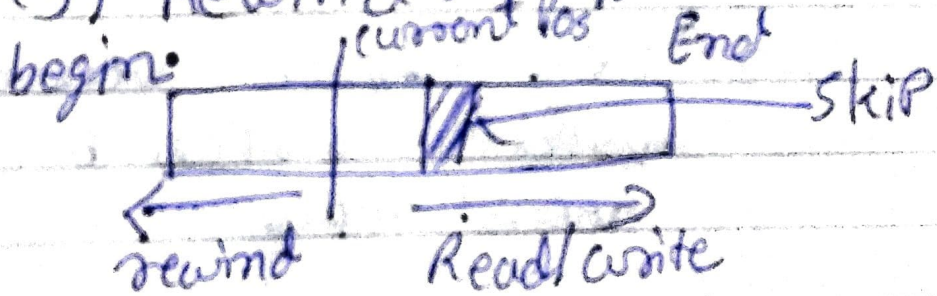
- So hear we want to find 2nd block then we just go to index block 9 then 4 then 2.

→ File access methods

- 3 types : 1) Sequential Access.
2) Direct Access
3) Indexed Access

(1) Sequential Access:

- Emulates magnetic tape operation
  - ~~One~~ operation
  - One record is process after other
  - Supports 5 records are there.
    - r_1 r_2 r_3 r_4 r_5
  - Operations 6 are as follow
    (1) read next: Read record and point next.
    (2) Write next:
    (3) - Write record ~~and~~ and more position
    (3) Rewind:



  - moving back to early location
    (4) Skip n records
    - You can skip n number of records

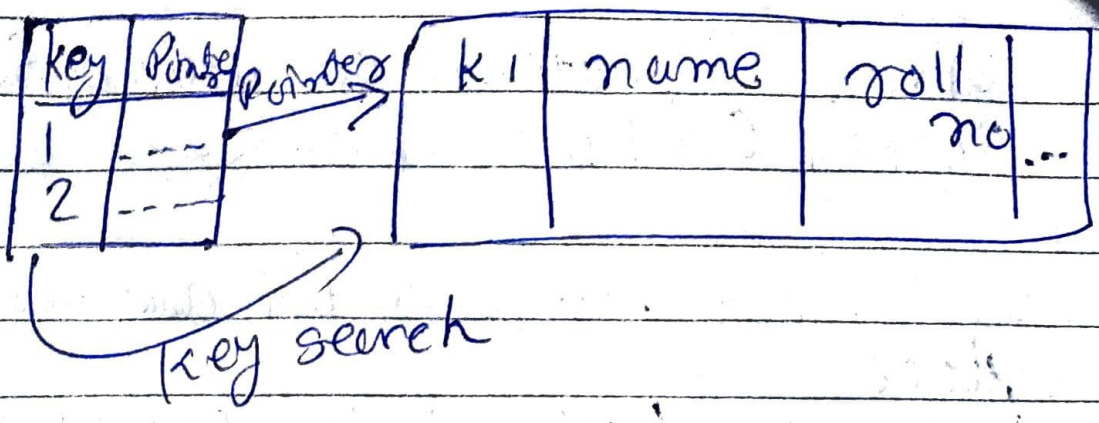
(ii) Direct Access: It is based on disk model. It allows random access. User can jump to any records and access that record.

- Following operation supported
  (i) Read n → Reading record no n
  (ii) Write n → Write Record no n
  (iii) Jump to record n → (Here we are at 5th record went to jump at 10), (0 or end of file)
  (iv) Query current record:
  used to written back to this record letter :

CCR

5 ——→ 10

(iii) Indexed Access: Index is created which contains a key field and pointers to the various blocks



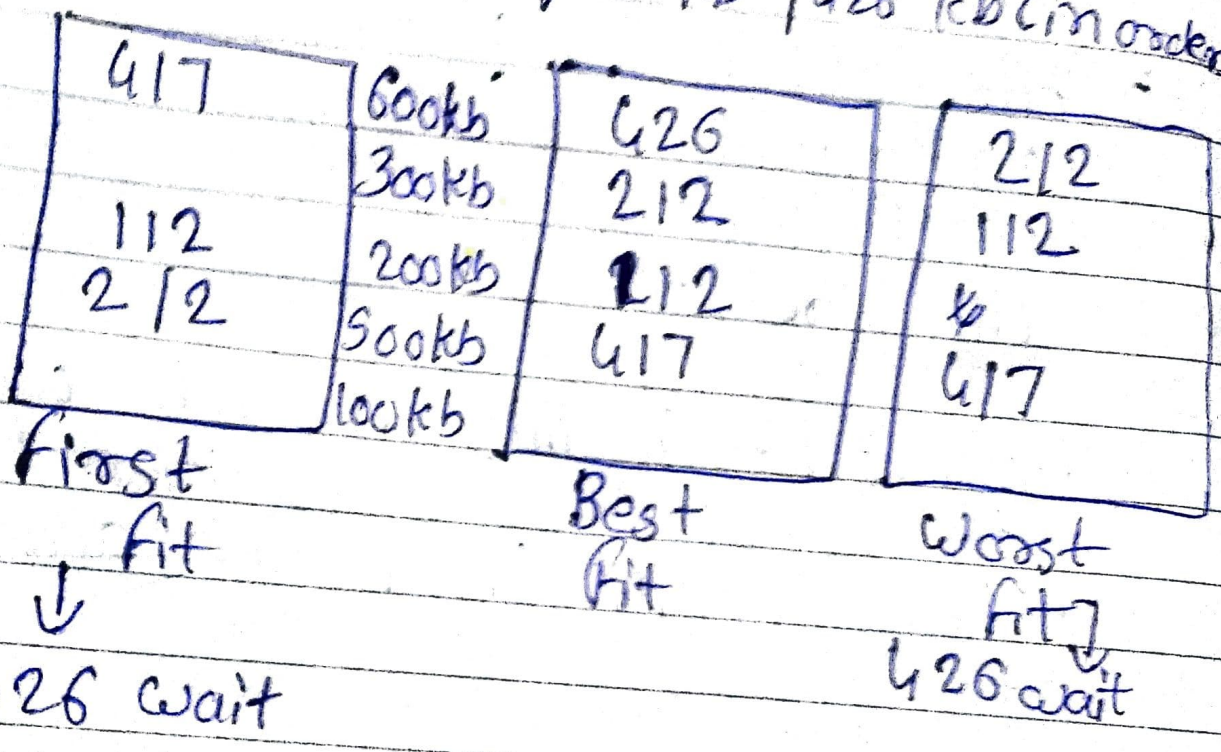
→ File allocation method
(i) First fit
(ii) Best fit
(iii) Worst fit

- First fit: In this approach we allocate the first ~~fit~~ free partition or hole large enough which can accomodate the process

- Best fit: It deals with allocating the smallest free partition which meets the ~~request~~ requirement of the requesting process.
  - the smallest hole which big enough to acquire process
- Worst fit: Here we allocate the largest available for partition. Reverse of best fit

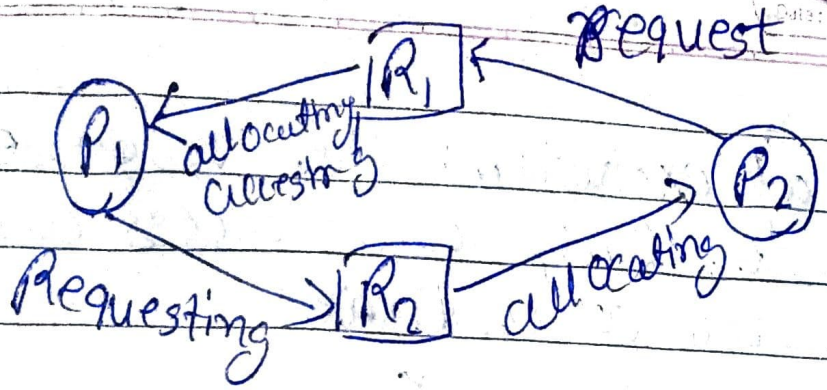
c) Given 5 partitions of 100Kb, 300Kb, 200Kb, 600Kb, 500Kb (in order)

Use First Fit, best fit and worst fit to place
212 kb, 417kb, 112kb, 426 kb (in order)



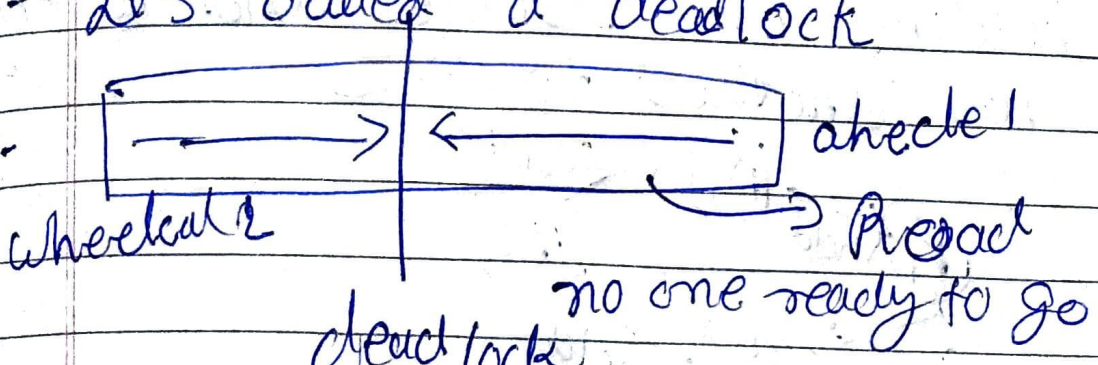
→ Deadlock

- Deadlock is problem
- It is situation or condition when two or more process sharing same resources and effectively preventing each other from accessing the resources



Until P_1 releasing R_1 P_2 can't access
Until P_2 releasing R_2 P_1 can't access

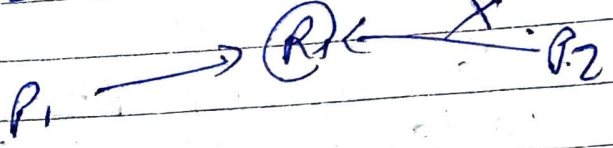
- It's called a deadlock



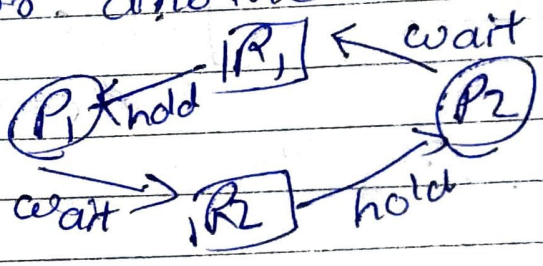
- when both process in waiting state. it's create deadth lock

— Condition for deadlock
If process
- Mutual exclusion → not shurable
- Hold & wait state →
- No preemption
- Circular wait

- Mutual exclusion: The resource is not sharable. Only one process can use resource at a time



- hold and wait:
  - holding one resources and waiting for another to come

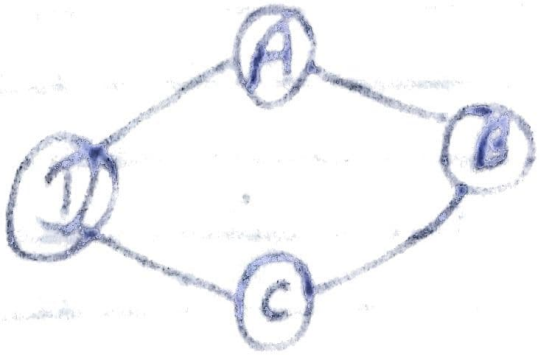


- No preemption:
  - ~~A resouce can be release resource~~
  - A resource can be release only voluntary by process it self
  - only process can release.
  - Not willing to given resource

- Circular wait:
-

- Circular wait

there are four process: A B C D



A acquire Resource needed by B (as same next)

- Various method to handle deadlock

(i). Deadlock Ignorance
(ii). Deadlock Prevention
(iii). Deadlock avoidance
(iv). Deadlock detection and Recovery

1) Deadlock Ignorance

- Most of os try to ignorance the dead lock
The deadlock occure rarely. whenever it occure the os will hang.

Printer

- So just restart your system that problem will rectified

why ignore?
- If we write a code for that the OS will decrease
- add deadlock code also monitor and every time it's reading even it's not required so system performance will decreases.
- system speed, capbility will increase if you avoid it.

2) Deadlock Prevention

- Try to find soln before deadlock accuses

Necessary condition for deadlock
- Mutual exclusion
- No preemption
- Hold & wait
- Circular wait

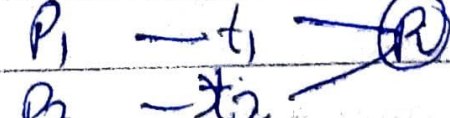
- try to remove any of condition or make four condition false.

- Dead lock prevention says:

→ To prevent deadlock try to make:

- (1) make mutual exclusion false,
  ⇒ Just shared the resouse
  - but it's not possible in case of pointer.
  - It's not work in some cases

(2) No preemption false
- Preemption is true
- to make preemption true using time quantium method



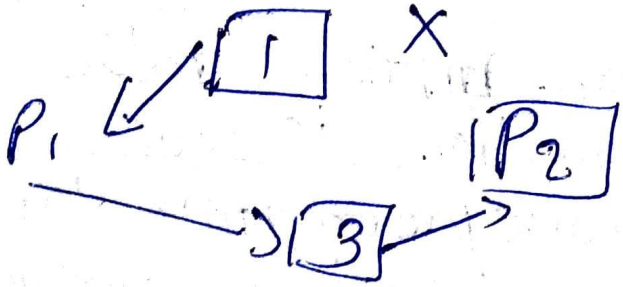
here P_1 have to release Resuce after t_1 time

(3) Hold Hold and wait false
- try to do no hold & wait then deadlock can be prevented.
- give all the Resource before it's start

④ Circular wait:
- try to make circular wait false
- just give the numbering to all resources so that the process can request resources in increasing order

eg

CPU - 1
Printer - 2
Scanner - 3
⋮



3- Deadlock avoidance

- Simplest & most useful model requires that each process declare max no of resources that it may need.
- Deadlock avoidance algo. dynamically examines the resource allocation such never be a circular wait condition.

- Basic facts
  (i) if a system is in safe state => No deadlock
  (ii) if a system is in unsafe state => posibility of deadlock

- Safe state: means if one process allocate to one resources in a sequence manner without any confusion or confict then it is in safe state. So your process has to execute successfully without any problem. So there is no deadlock

- Unsafe state: then problem may occure. there is possibity of deadlock.

- Avoidance: Ensure that a system will never enter a unsafe state

eg Deadlock avoidance using bankers algo.
- Conditions:
  - There is a chance for multiple instance
  - Each process priore taime maximum use
  - Process request for resouce it may have to wait

Total A=10 B=5 C=7

- When process gets all it's resources it must return them in finite time

Process	Allocation A B C	Maximum A B C	current work A B C	Remaining A B C
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2	5 3 2	1 2 2
P_2	3 0 2	9 0 2	7 4 3	6 0 0
P_3	2 1 1	4 2 2	7 4 5	2 1 1
P_4	0 0 2	5 3 3	7 5 5	5 3 1
P_4	7 2 5		10 5 7	update

Remaing need = maximum - allocation
Current work = total - Allocation(total)

- So now we have to calculate safe state process.

Need$_i$ $\leq$ work (current work)
$\Rightarrow$ work = work + allocation

P_0 7 4 3 $\leq$ 3 3 2 (each resources)
(not total value)

P_0 Not Safe state

Safe state $<P_1, P_3, P_4, P_0, P_2>$ = Process execute

P_1 122 ≤ 332 ✓ safe state

current work = 332 + 200
= 532

P_2 = 600 ≤ 532 ✗ Sta

P_3 = 211 ≤ 532 ✓
= 743

current work = 211 + 532
= 743

P_4 = 531 ≤ 743 ✓
current work = 743 + 002
= 745

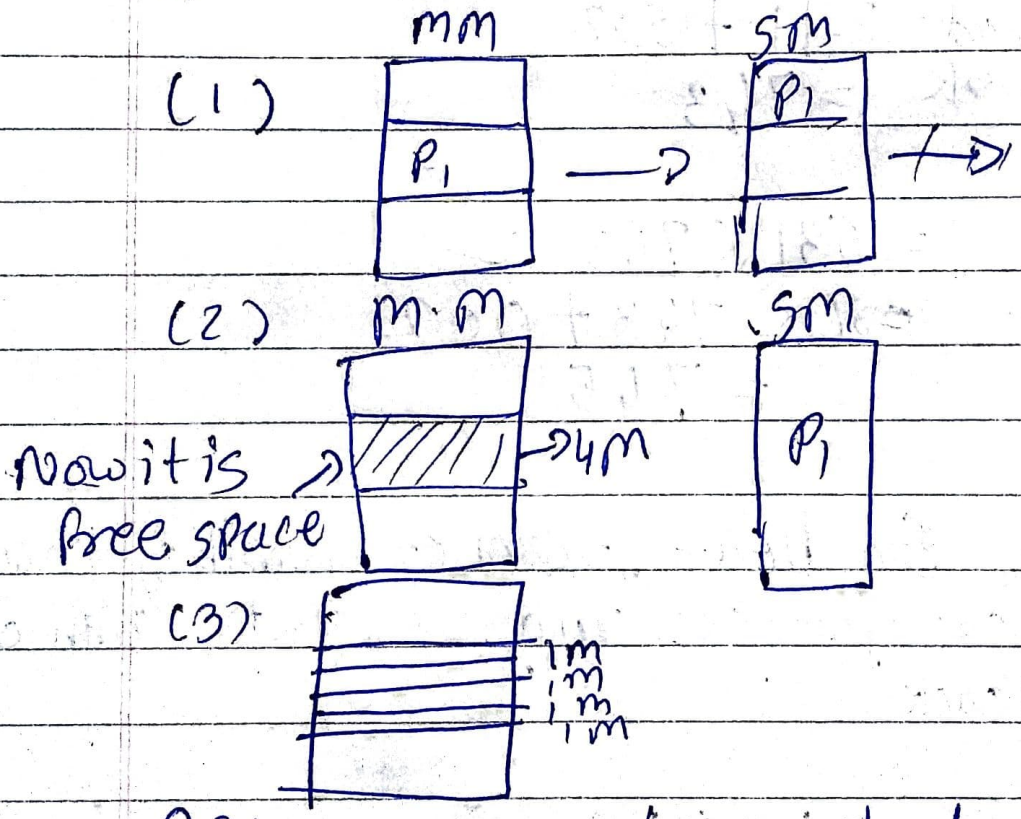
So hear resouce avalibity increasing so other process may execute with out dead loock

P_0 · 743 < 745 ✓
C. work = 745 + 010
= 755

P_2 = 600 ≤ 755 ✓
C. work = 755 + 302 = 1057

→ Fragmentation

- Important concept of Main memory
- How the data is store/remove from the main memory and swapping to secondary memory
- A process are baded/remove from the memory then the free memory space is between into little pieces.

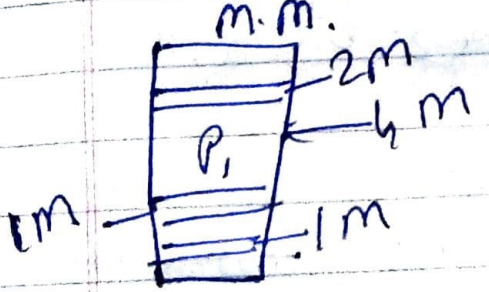


- After some times that processes can't be allocated to memory because of small size and memory block remains unused this problem is called fragmentation.

- It is devided into 2 parts
  (i) External fragmentation
  (ii) Internal fragmentation

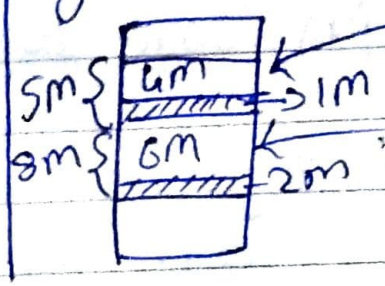
External Fragmentation

- Total memory space is enogh to satisfy a requirement or reside a process in it but it is not contigious so it can't be used



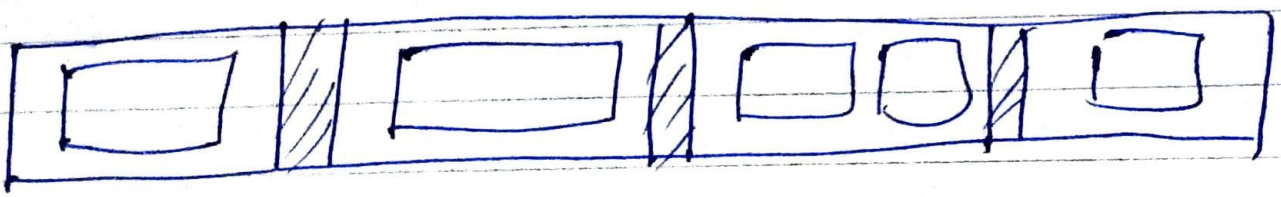
Internal Fragmentation

Memory block assigned to process is bigger some portion of memory is left unused as it can't be used by another process

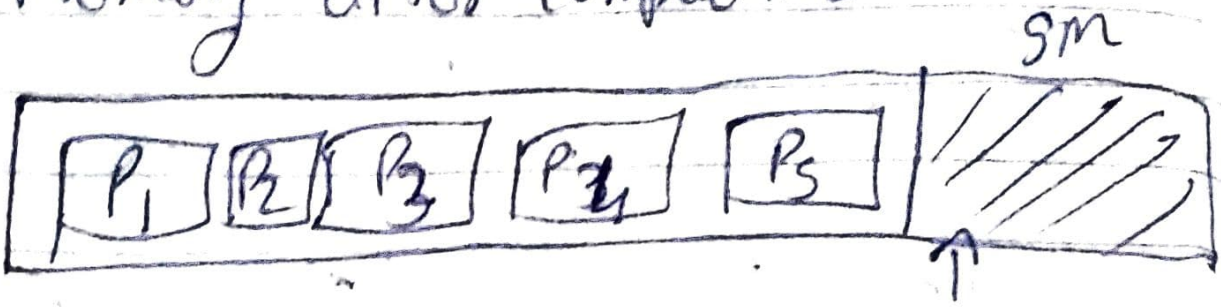


- To overcome this fragmentation memory can be compact

Before compaction



- Memory after compaction



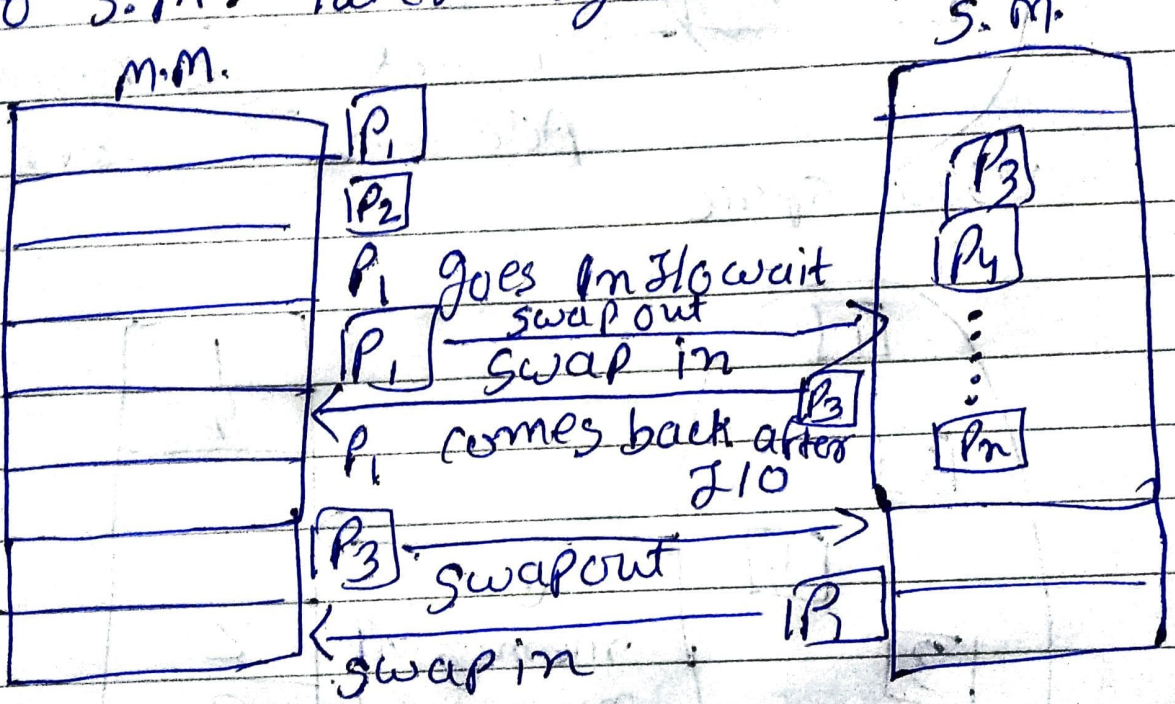
Ex

- External fragmentation:
  Can be reduced by compaction or shuffle free memory together in one toger longer block..

- Internal Fragmentation:
  Can be reduced by ~~compaction~~ or effectively assigning the smallest partition but large enough for the process.

→ Swapping

- We have to swap data to main memory to secondary vise versa
- It is mechanism in which process can be swapped temporarily out of M.M to S.M. later again swaps back to M.M.

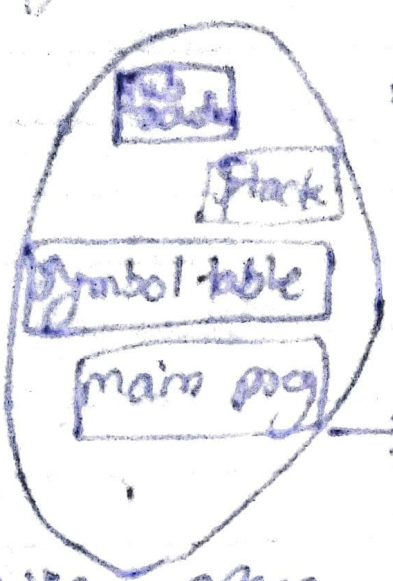


→ ~~Fragmentation~~ Segmentation

- A program is a collection of segments
- A segment is a logically unit such as

Segment { Main Program, Procedure, Function, Local var, global var, common block, stack, Symbol table, arrays

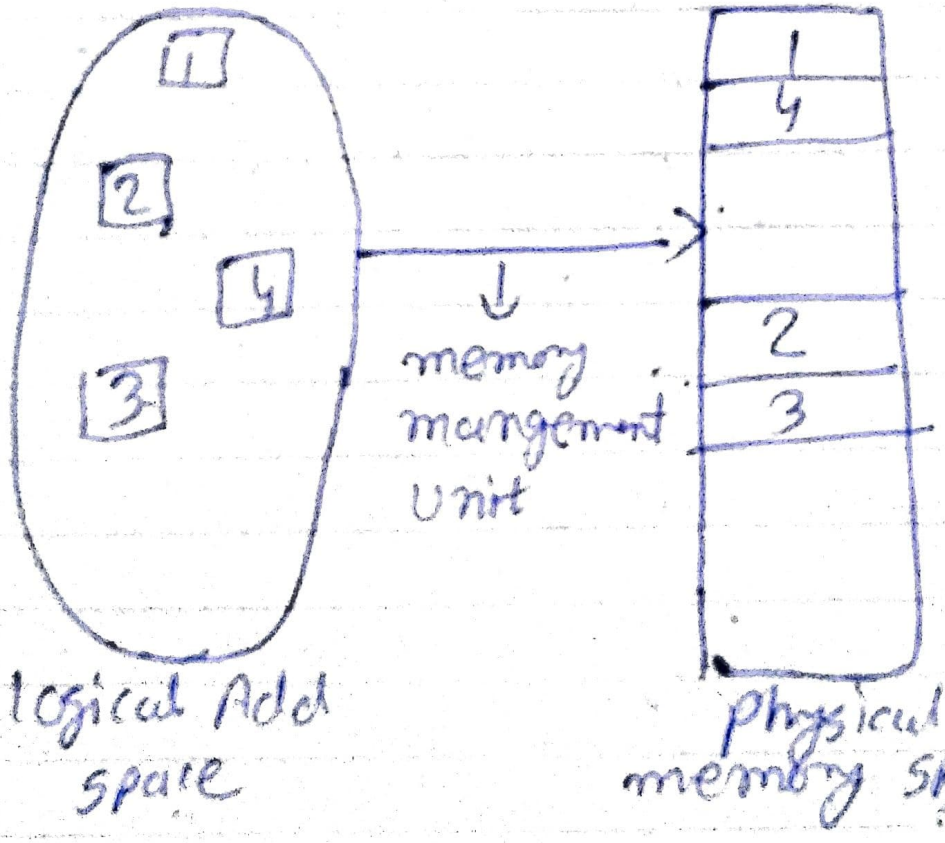
- logical view of segmentation



=> Specific each addr by 2 quantities
(a) segment no.
(b) segment offset

User space

logical Address => <segment #, offset>

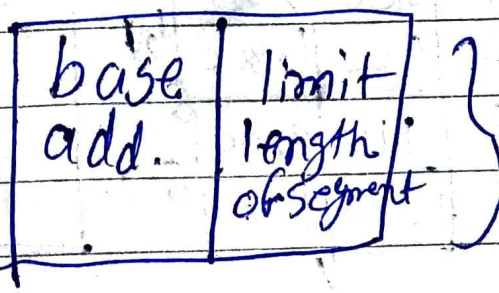


logical Add space

Physical memory space

- Segmentation Architecture

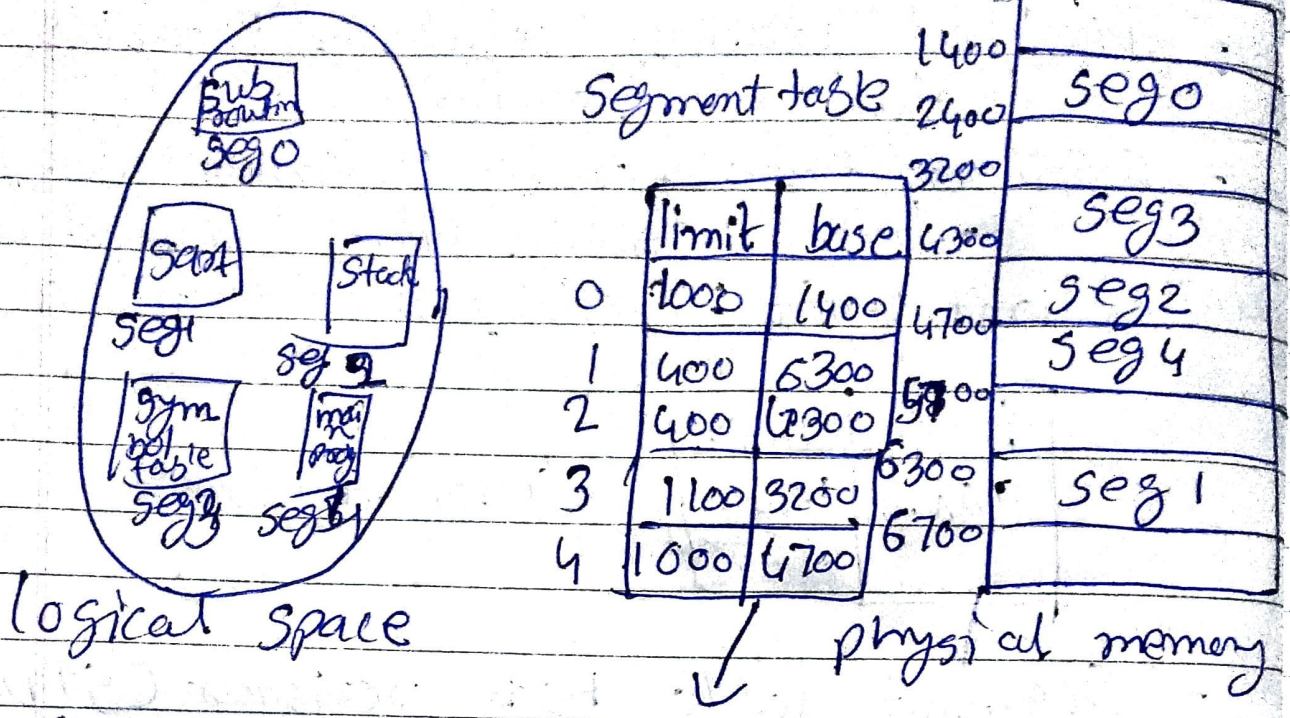
- Logical address contain 2 tables
  <seg no., offset>
- segment table : maps 2 dimensional phy. addr.
  - base : Contain starting physical add.
  - limit ~~limit~~ : Specifies the length of seg.



- Segment table base register (STBR)
  - Points to the segment table location in memory

- Segment table length ~~register~~ (STLR)
  - Indicate member of segment used by program
    segment no "S" is legal (when) if
    S < STLR

- Example of segmentation



→ Paging

- It's concept / example of non-contiguous allocation (Virtual memory)

- It provide easy to users
- Increase CPU utilization

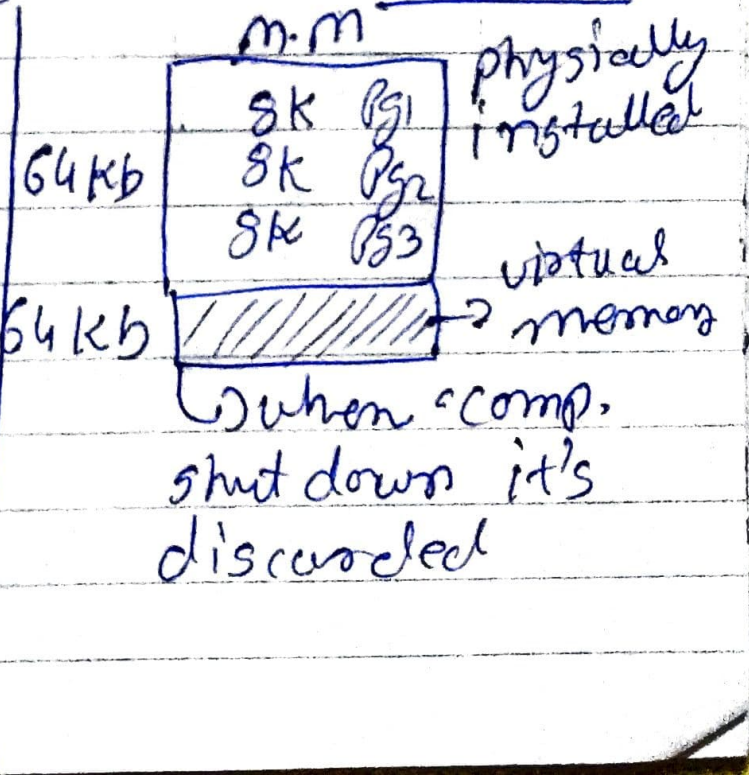
→ no real

- Os giving illusion to the user such that
  - user can write a very big program
  - & user thinks it's entire program present in RAM.
  - All the space allocated to the user is contigious

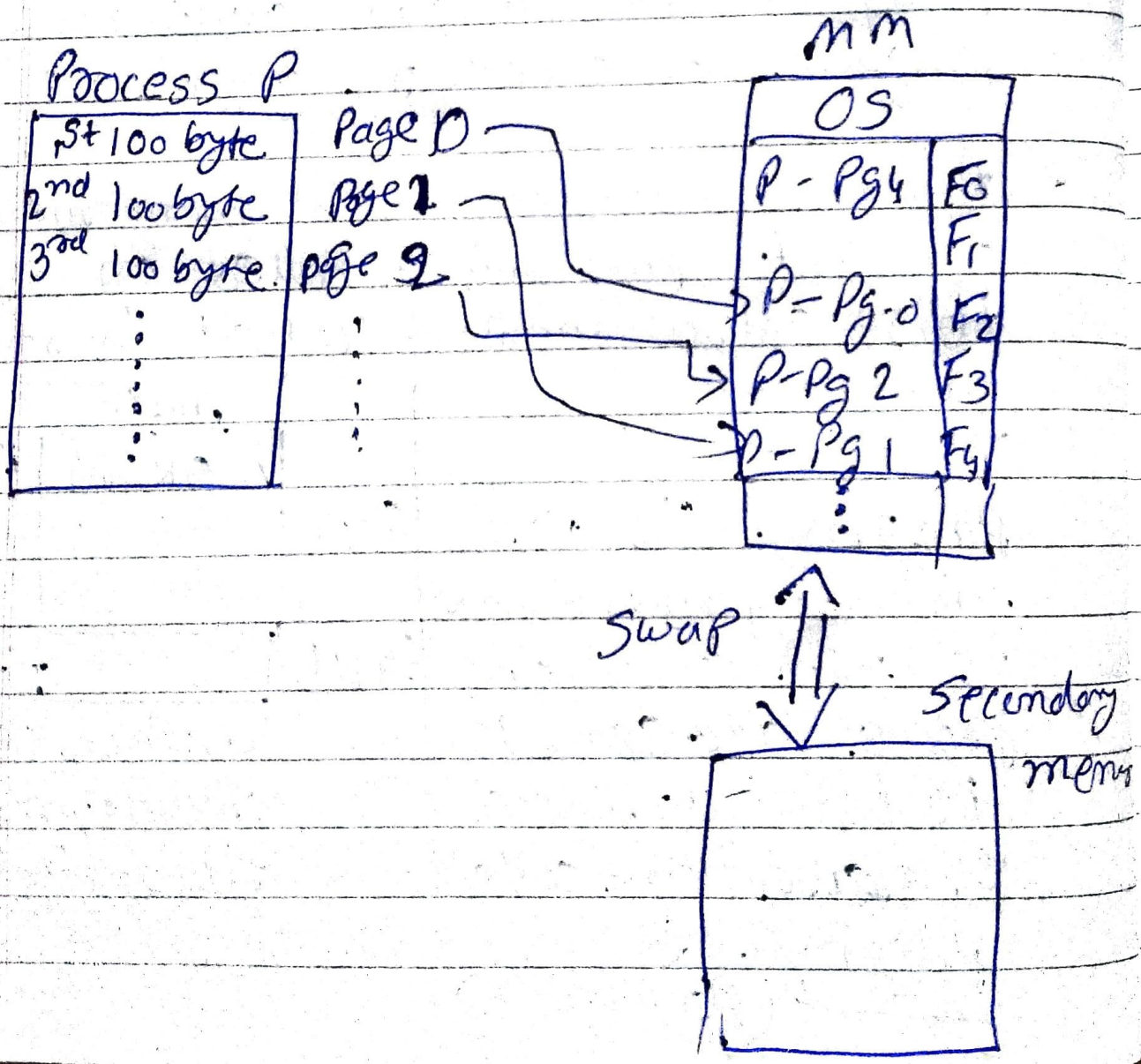
- In reality only a small potion of the user program is in RAM which may or may not be contigious while the remaining program in secondary memory.

- A computer can add more memory than amount physically installed on system. this extra memory called "virtual memory"

- Paggingis is A memory managment technich in which process add space is broken into blocks of same size called paging. ↓ (power of 2)



- The size of process can be measured in number of pages.
- M.M is divided into smalled fixed block (physically) called as "Frames"
- Size of the frame can be same as size of page frame size = size of page
- We can avoid external fragmentation

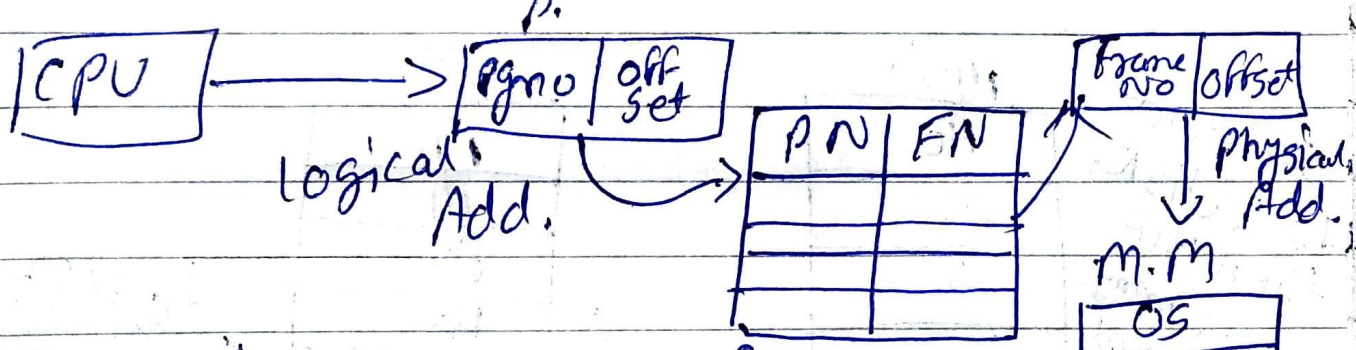


- Address translation

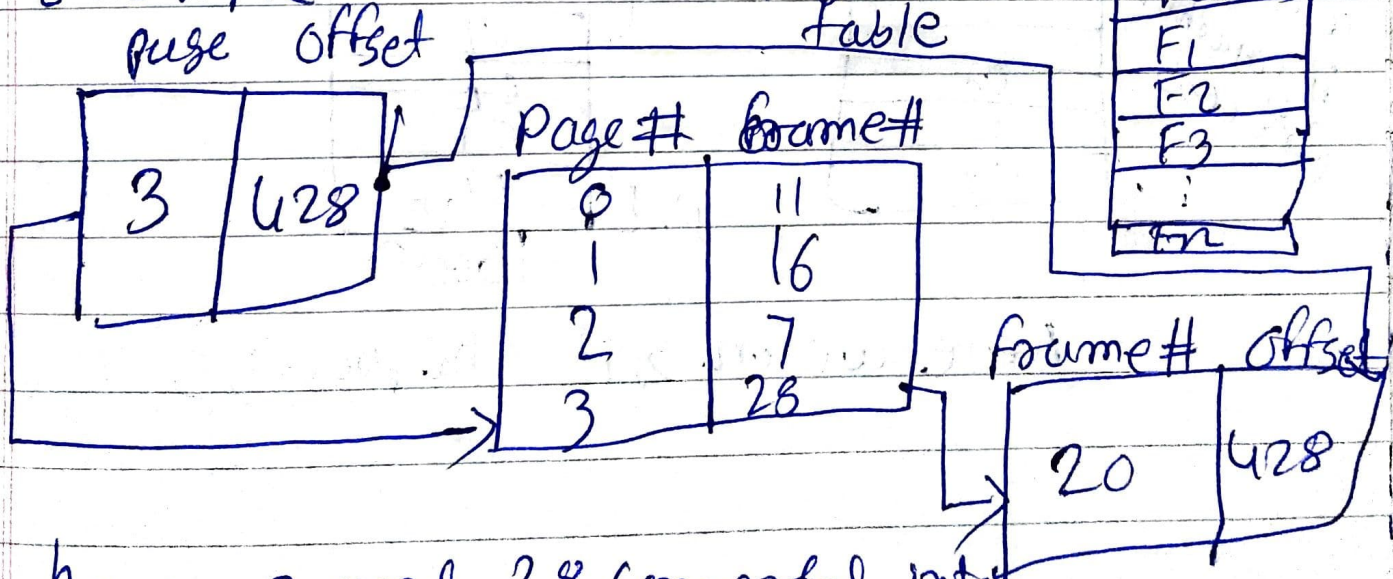
Page addr is logical addr. & represented by Logical Add. = pg. no + page offset

- Frame Add. is called physical add. & it is represented by

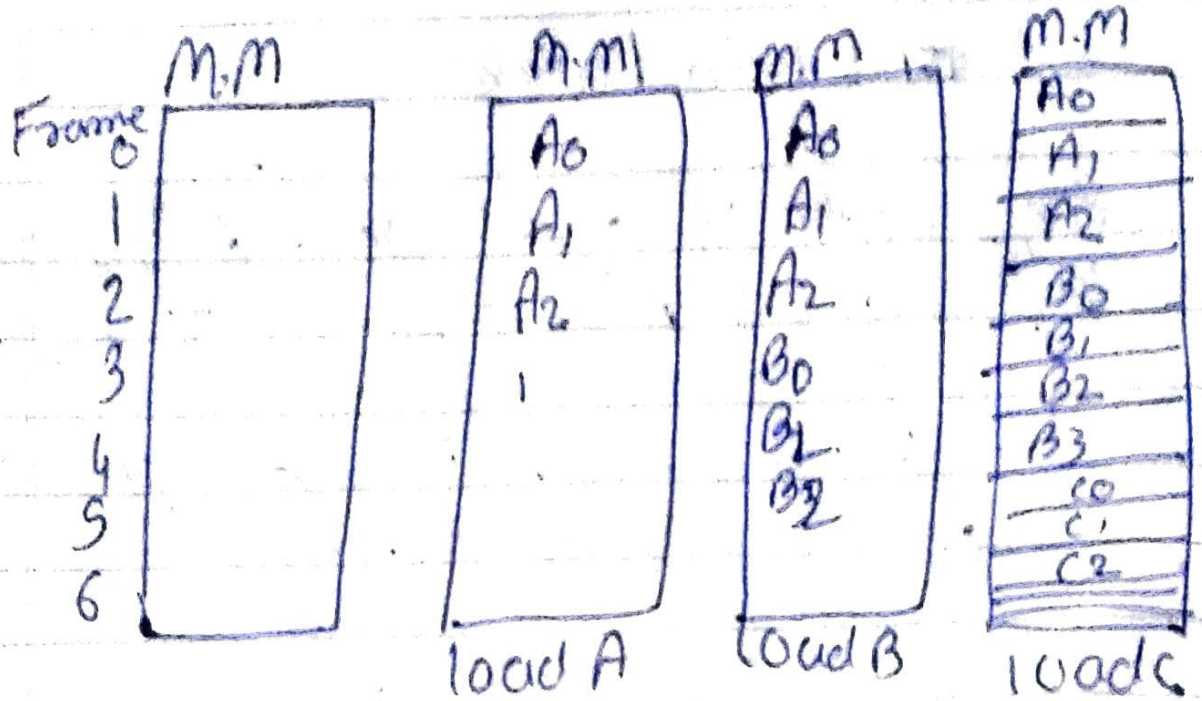
Physical add = frame no + page offset

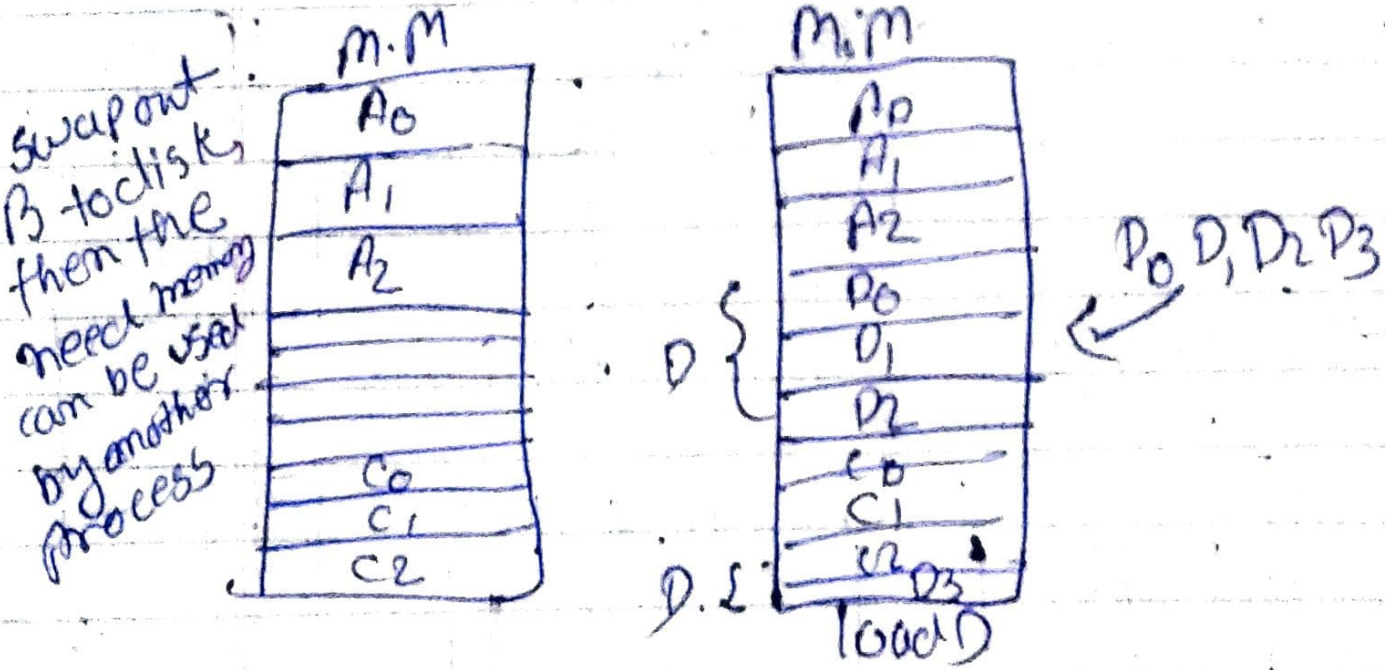


→ Example



- here 3 and 28 converted into 3 and 28





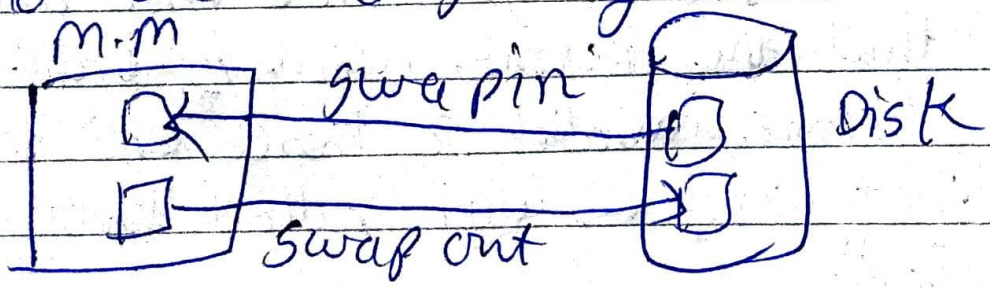
here we can split the process

- Advantages.
  - Reduce External Fragmentation bcz no of pages = no of frame
  - Simple to implement assume as an efficient memory management
  - equal size of frame & paging it is swapping easy to do.

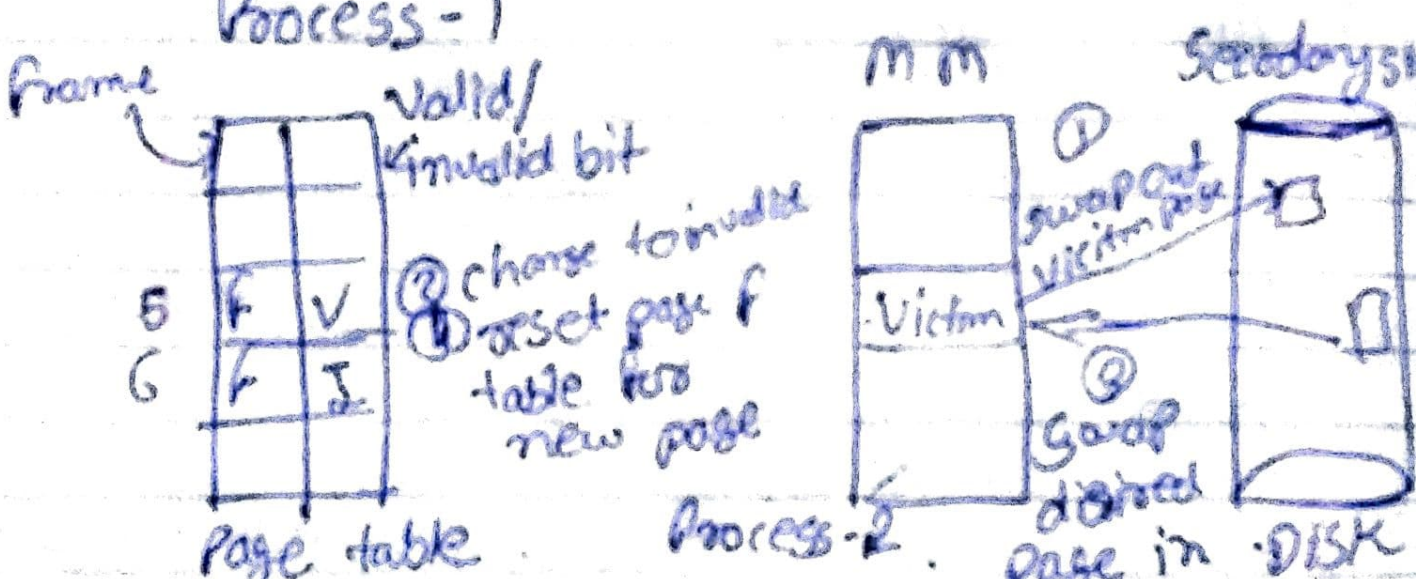
- Disadvantages
  - It's suffer from internal fragmentation
  - It's not suitable for small ram

→ Page replacement

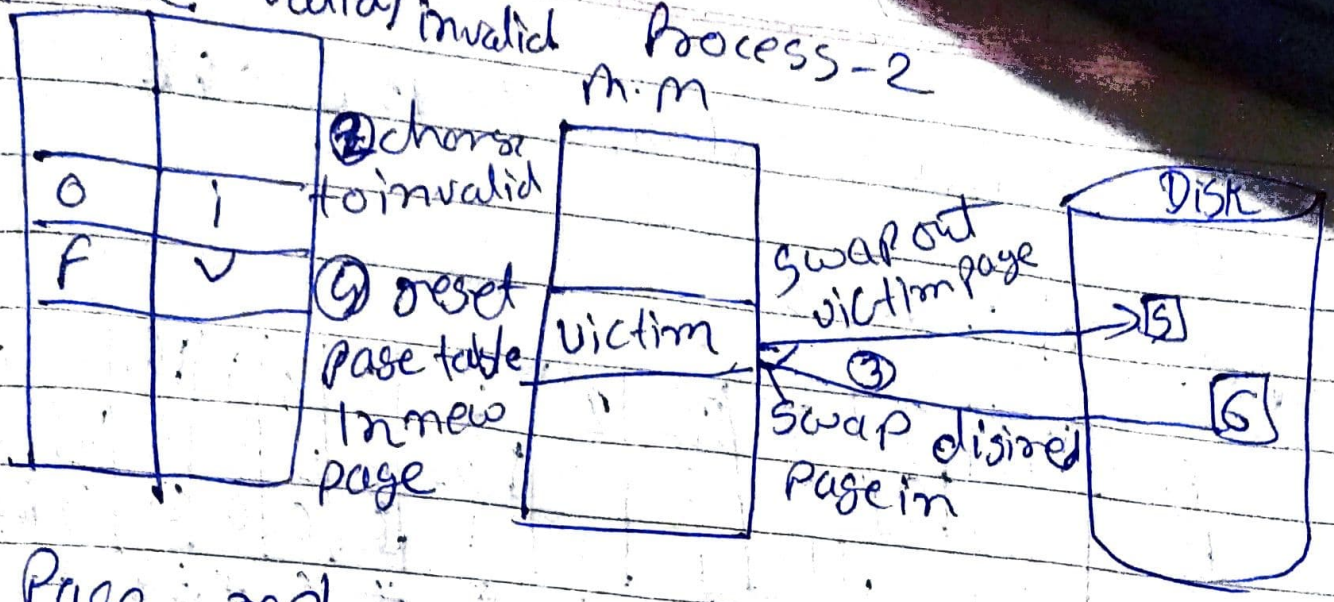
- Whenever page fault means the page is not presented in the main memory and it's demaning by CPU. and also main memory is full and we are swapping the main memory which are not used in the disk and we are getting back to disk



- Prevent over allocation of memory by modifing page-fault service routine to include page replacement
- Use modify (dirty) bit to reduce overhead of page transfers. It indicates that this page should be written to disk
- So Page Replacement completes the logical and physical memory so bcz the large virtual memory can be provided on smaller primary memory



- Here victim goes into the Disk after that the frame which is valid bit become invalid so here CPU req. for that and OS will called that page from S.M.



→ Page replacement algorithm

x- It will deside that which technic should be allocat.

- P.R. algo. deside which page should be replaced.
- 3 algorithm are there
  (1) FIFO
  (2) LRU
  (3) Optimal

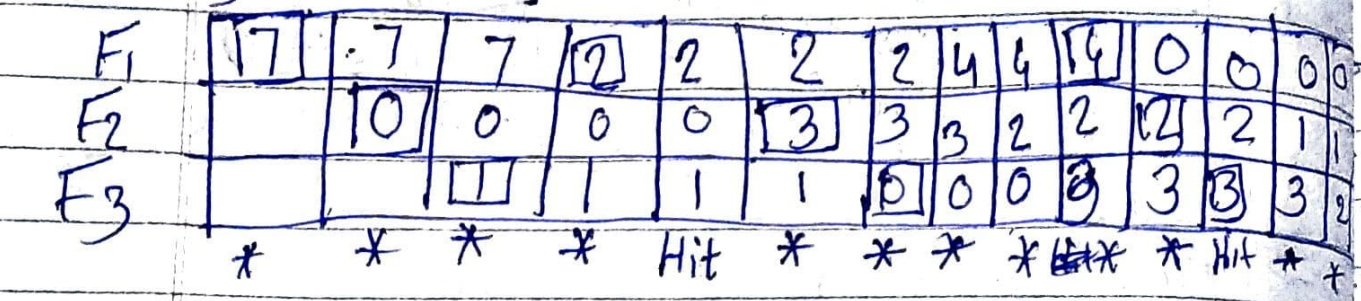
(1) FIFO page replacement algorithm
- Replace pages that has been in memory longest time.

not present in mm ✓

Page fault: The page demaning by CPU
Page hit: demaning by ~~which in~~ CPU which present in M.M.

eg Reference string: 7 0 1 2 0 3 0 4 2 3 0 3 1 2

First P_1 => Pages



at begin m.m (RAM) is fresh means free

Page Fault = * Page hit = Hit

7 ⟶ disk 2 ⟵ 2

swap

total page hit = 3

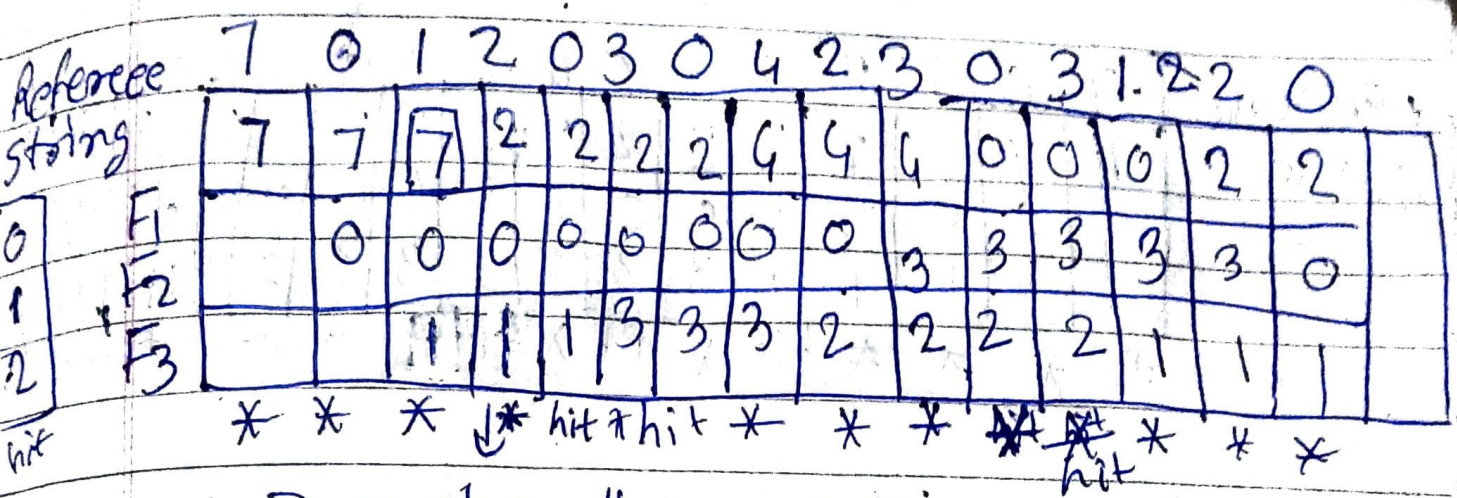
page false = 12

hit ratio = $\frac{\text{no of hits}}{\text{Total number of reference}}$

$= \frac{3}{15} = \frac{1}{5} \times 100 = 20\%$

Page false ratio = $\frac{12}{15} \times 100 = 80\%$

→ LRU (Least Recently Used)



1 → most recently use
7 → near use
901

0 → most recent
1 → least use
3030 (Page)
120

0 - most recent
2 - least use
2030

hit = 3
Fault = 12

$$\text{hit ratio} = \frac{3}{15} \times 100 = 20\%$$

$$\text{False ratio} = \frac{12}{15} \times 100 = 80\%$$

→ Here we need to find least value if the 3rd last value is present in the least then we have to go for next left value

Optimal Page replacement algo

Reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1

- In optimal which is used later case.
- In up to now we used left side but in optimal page replacement we have to check right side we need to find which is not used longer time in future

- At last replacement we need to replace that page that we used to long at time

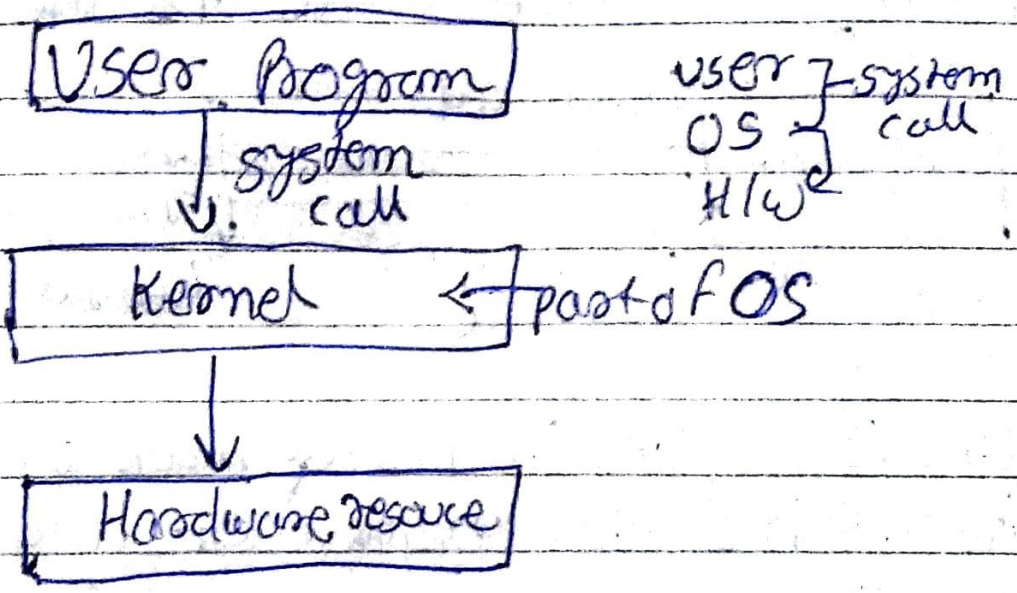
page hit = 9
page faults = 9

$$\frac{9}{18} \times 100 = 50\%$$

$$\frac{9}{18} \times 100 = 50\%$$

→ System call

- System call provides interface b/w user programs & operating system



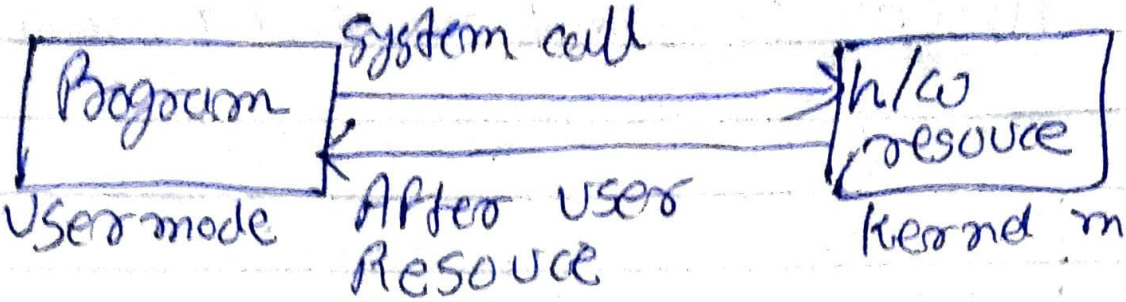
- It is programmatic way in which a computer program request a service from the kernel of the operating system

- Program 2 modes → User mode
  excute
  → Kernel mode

- In user mode perform all type user operation. It' can't access the hlw resource

- Kernel mode: When it requires the RAM
  It can access h/w like RAM, printer

- When program needs any h/w resouce
  it needs to make a call to the kernel



- The I/O's goes in kernel mode only when
  it requires the resource of H/W

- Due to security reasons; User applications are not given access to H/W resources.

- When they need to do any I/O or resouce some memory, if request Os to use all these.

- This request made through system call

- Types of system calls
  (1) Process control
  (2) File management
  (3) Device management
  (4) Information management
  (5) Communication

1) Process control: This system calls deals with processes
example.. (i) end, abort
(ii) load, execute
(iii) Create process, terminate process
(iv) get process attribute, set process attribute
(v) wait for time
(vi) wait event, signal event
(vii) allocate & free memory

2) File Management: These system calls are responsible for file manipulation such as
(i) create file, delete file
(ii) Open, close file
(iii) read, write, reposition
(iv) get & set file attributes

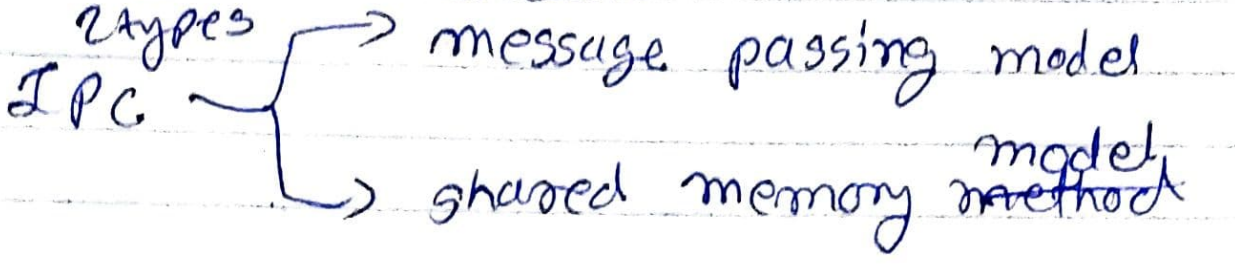
3) Device management

- A process need several resouce to execute main memory, disk drives, access file, I/O devices & so on..
- If these resources are free, they will be allocated to process otherwise the process have to wait until resouce are available.
- These system calls are responsible for device manipulation such as
  (i) Request device, resouce device
  (ii) Read, write
  (iii) get device attributes, set device attribute
  (iv) logically attached or detach devi

4) Information Management
- These system calls handle information & transfer b/w the sperate system & the user program
- ex - most system have a system call to return the amount & date & time
- The number of amount user, program

the vesion of OS the amount of free memory or disk space & so on.
(1) get time or date, set time or date
(2) get system date, set system date
(3) get & set process, file or device attributes

5) Communication
These system calls are useful for inter process communication



- In message passing model, the communication process exchange message with one another to transfer information
- In shared memory model, processes use shared memory to exchange information by reading & writing on it.
- Some system calls:
  (1) create, delete communication connection
  (2) send and receive msg
  (3) transfer status information
  (4) Attach & Detach remote device

- Disk scheduling

- Turn around time: Time diff. between completion time and arrival time.
  Turn around time = Completion time - Arrival time
- Waiting time: Time difference between turn around time and bust time
  Waiting time = Turn around time - Burst time

-> Longest job first: It is similar to SJF scheduling algorithm, but in this scheduling algorithm, we give priority to the process having the longest bust time
This is non-premitive in nature. ie. when any process starts executing, can't be interrupted before complete execution

-> Shorted remaing time first: It's premptive mode of SJF algorithm in which jobs are schedule according to Shortest remaining time.

-> Longest remaing time first: It's

preemitive mode of LJF algorithm in which we give priority to the process having largest burst time remaining

- Highest Response Ratio Next (HRRN): In this scheduling, processes with highest response ratio is scheduled. This scheduling avoids starvation.

$$\text{Response Ratio} = \frac{(\text{waiting time} + \text{Burst time})}{\text{Burst time}}$$

- Multilevel Queue scheduling: According to the priority of process, processes are placed in the different queues. Generally high priority process are placed in the top level queue. Only after completion of process from top level queue, lower level queued processes are scheduling it can suffer from starvation

- Multilevel Feedback Queue scheduling: It allows the process to move in between queue. The idea is separate processes according to the characteristics of their CPU bousts. If a process uses too much CPU time it is moved to a lower-priority queue