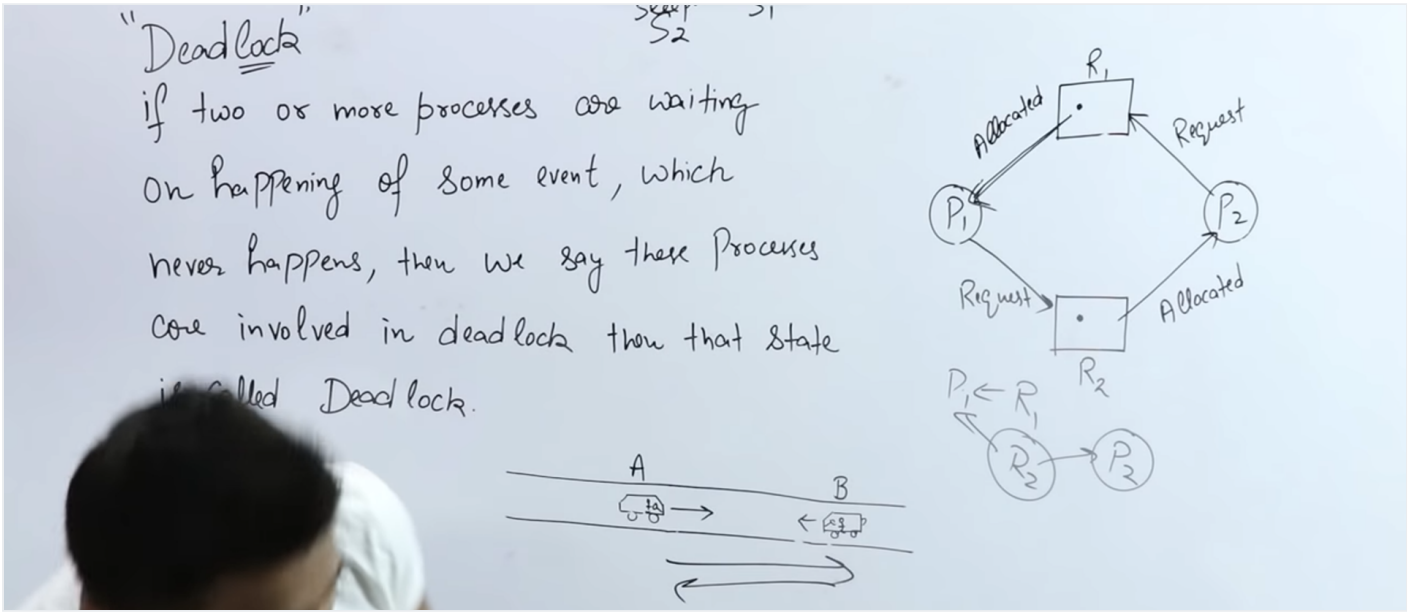
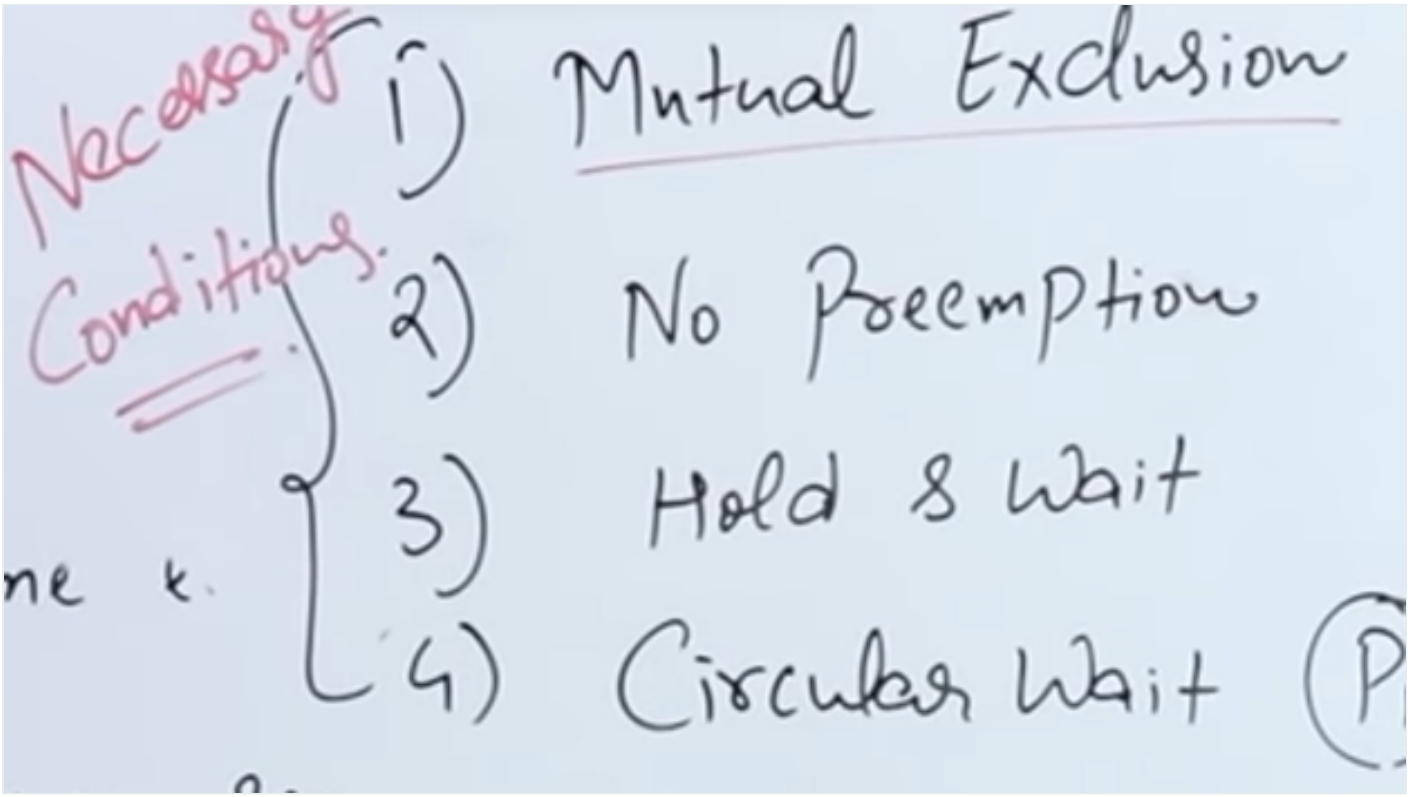
Deadlock

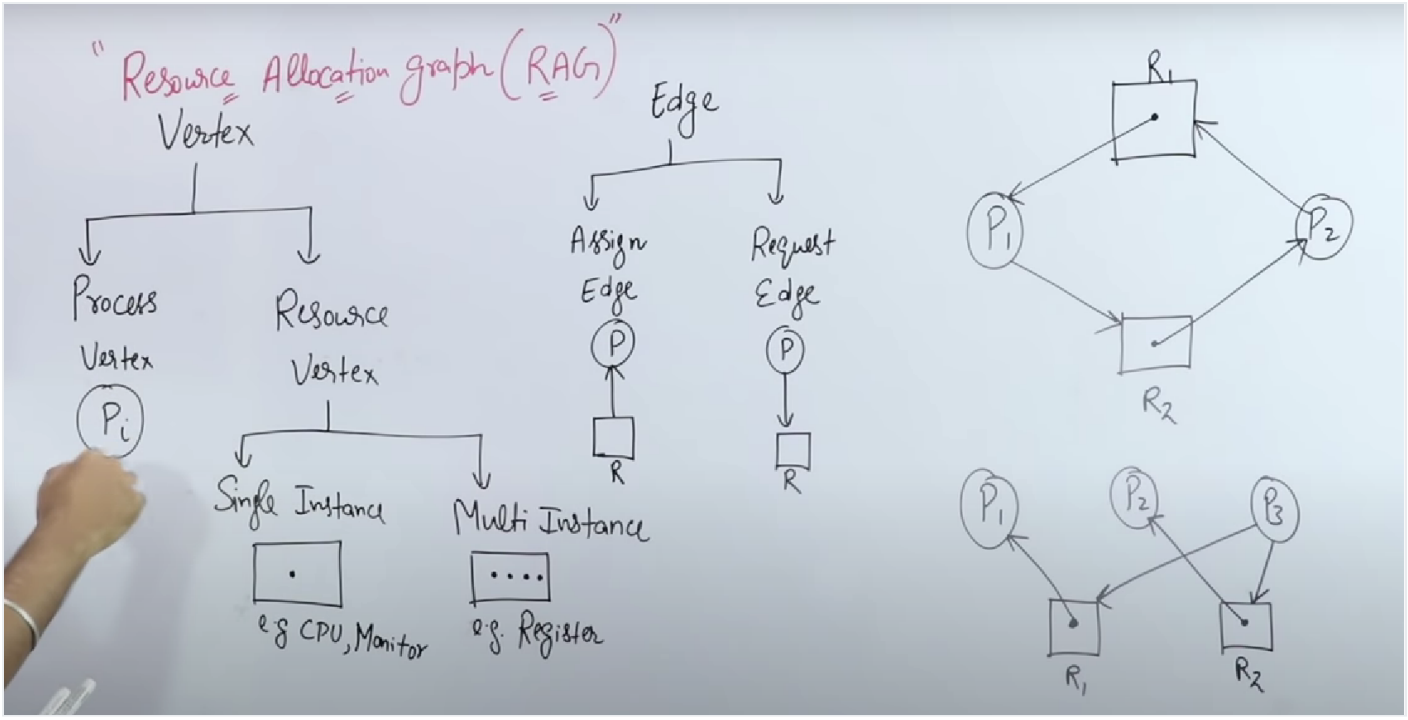


neccessary conditions for deadlock

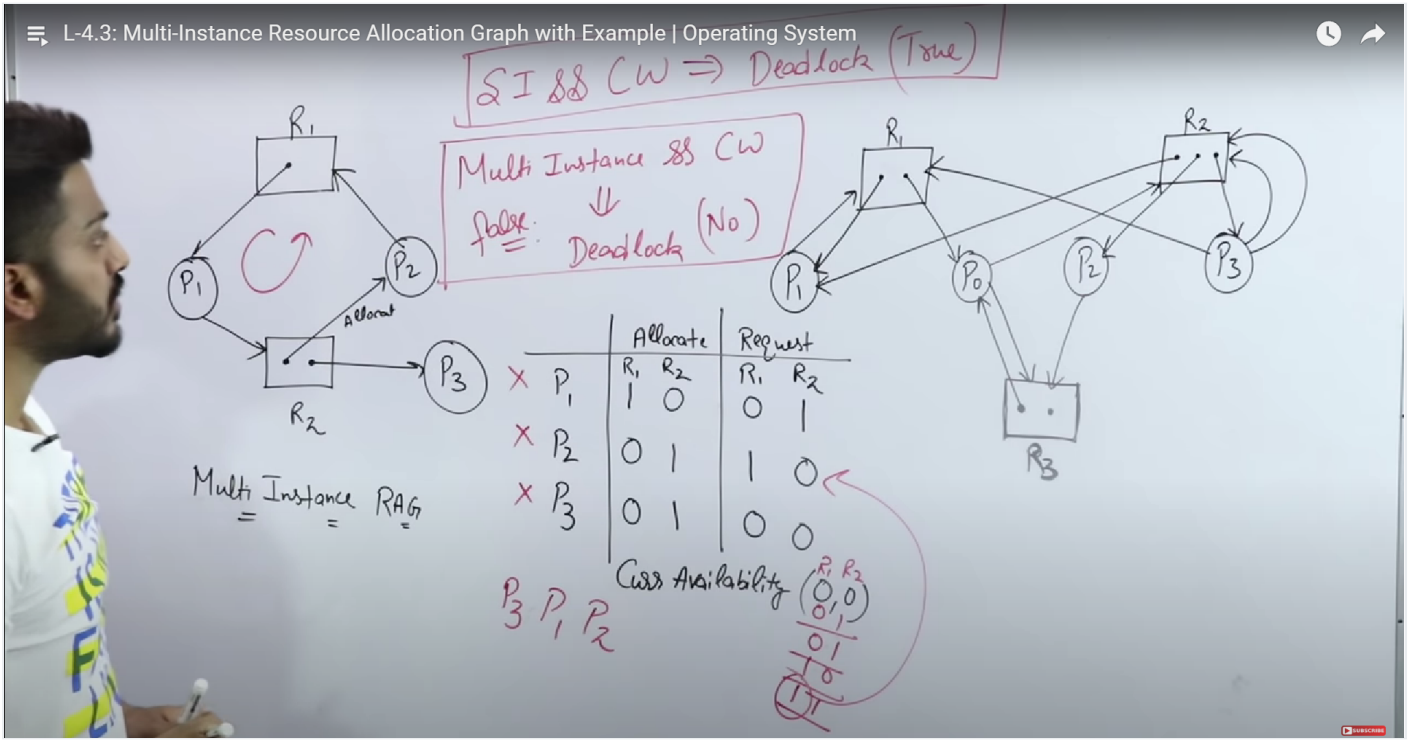


all the conditions should be true to cause a deadlock.

for single instance :



for multiple instances :



Deadlock Handling :

Deadlock ignorance: as the deadlock occurs very rarely so we ignore it. so that we don't have to increase the functionality of OS and trade it with the speed.

Deadlock prevention: either remove (or get it false) all or at least one of the following conditions ->

1. mutual exclusion - no mutual exclusion should be there
2. no pre-emptive - be pre-emptive
3. hold and wait - don't hold and wait

4. circular wait - give numbers to all resources and a process after requesting a resource can only request for resource of higher number than previous request.request shouuld be in increasing order.

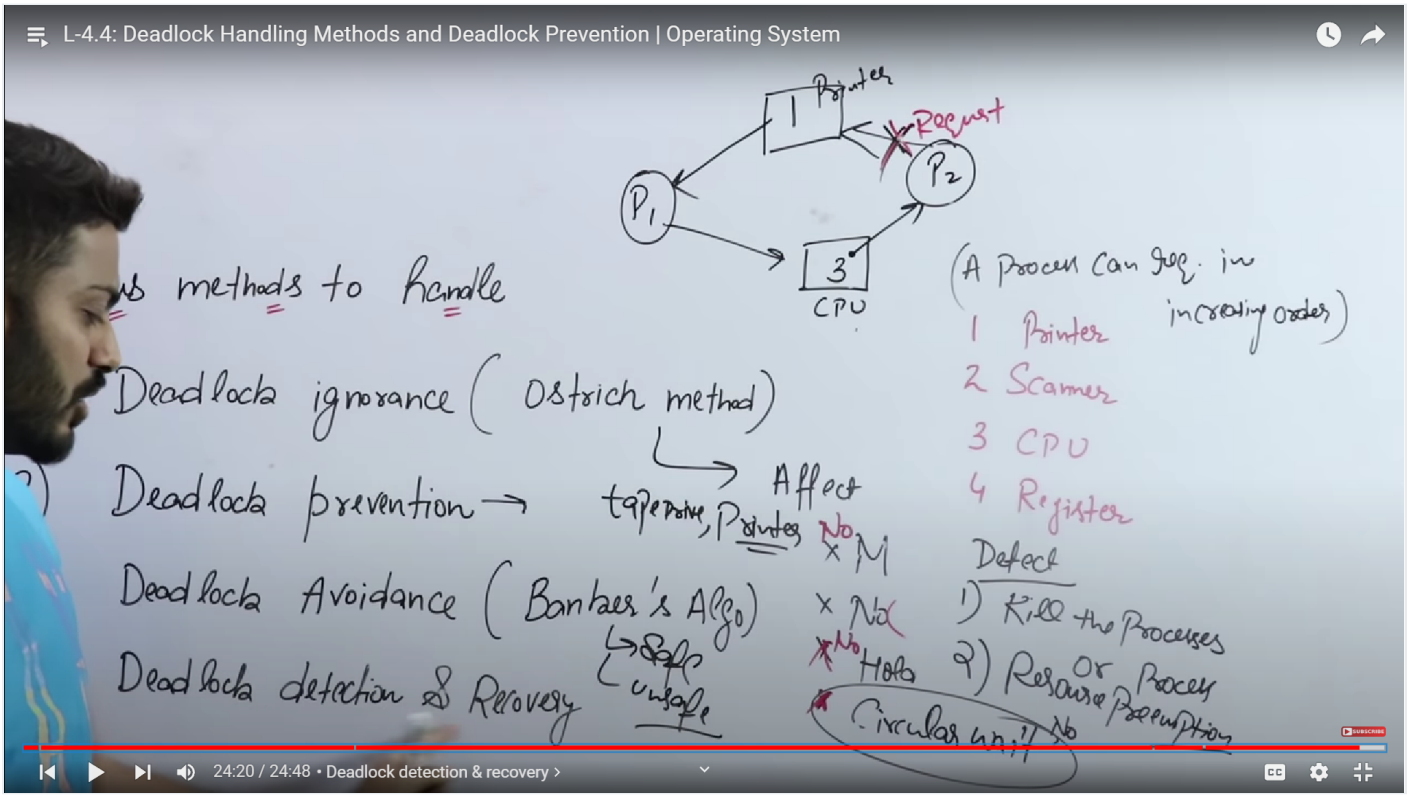
to prevent the deadlock.

Deadlock Avoidance (Banker's algorithm) : before giving the resource to the process it should check whether it is safe or unsafe to share the resource.

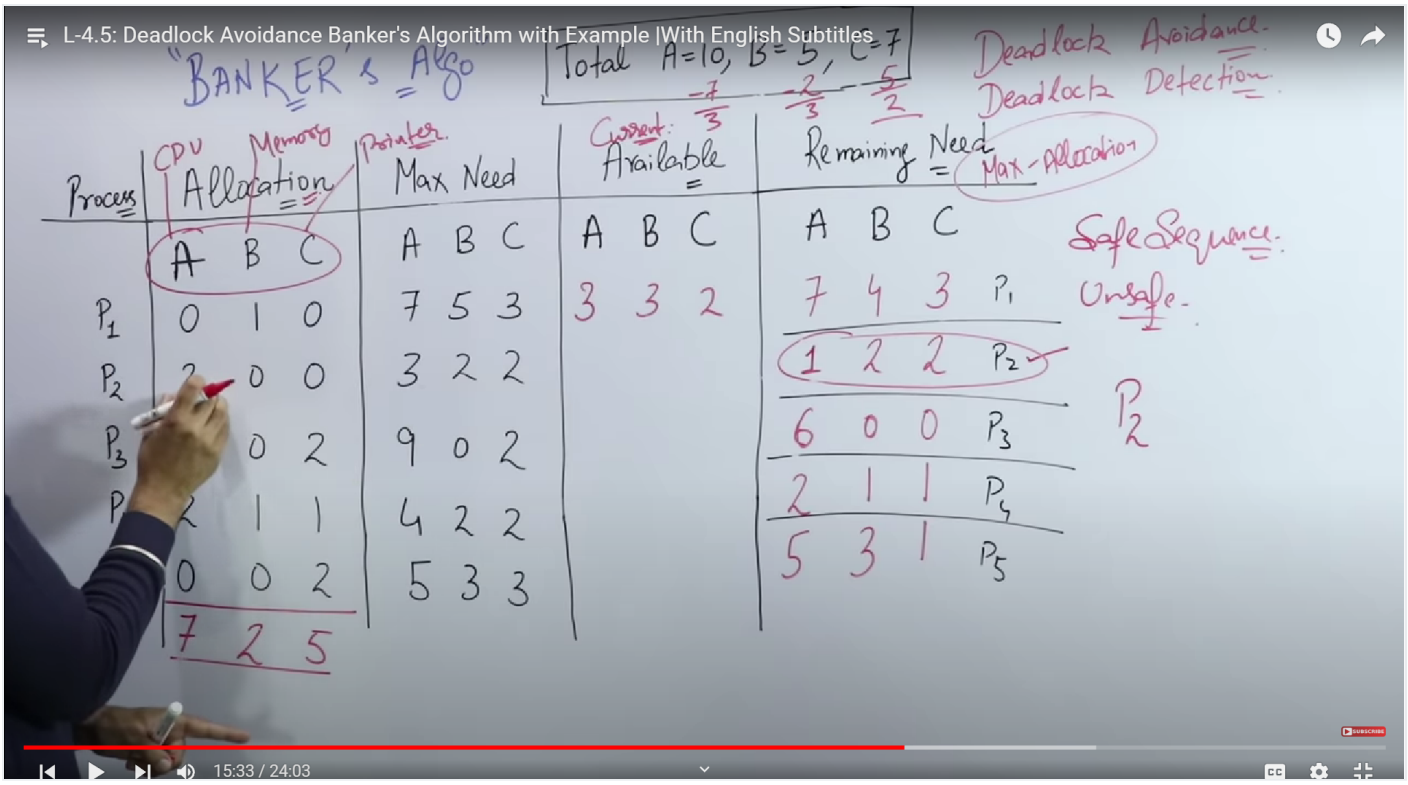
Deadlock detection and recovery : detect if there is deadlock or not.

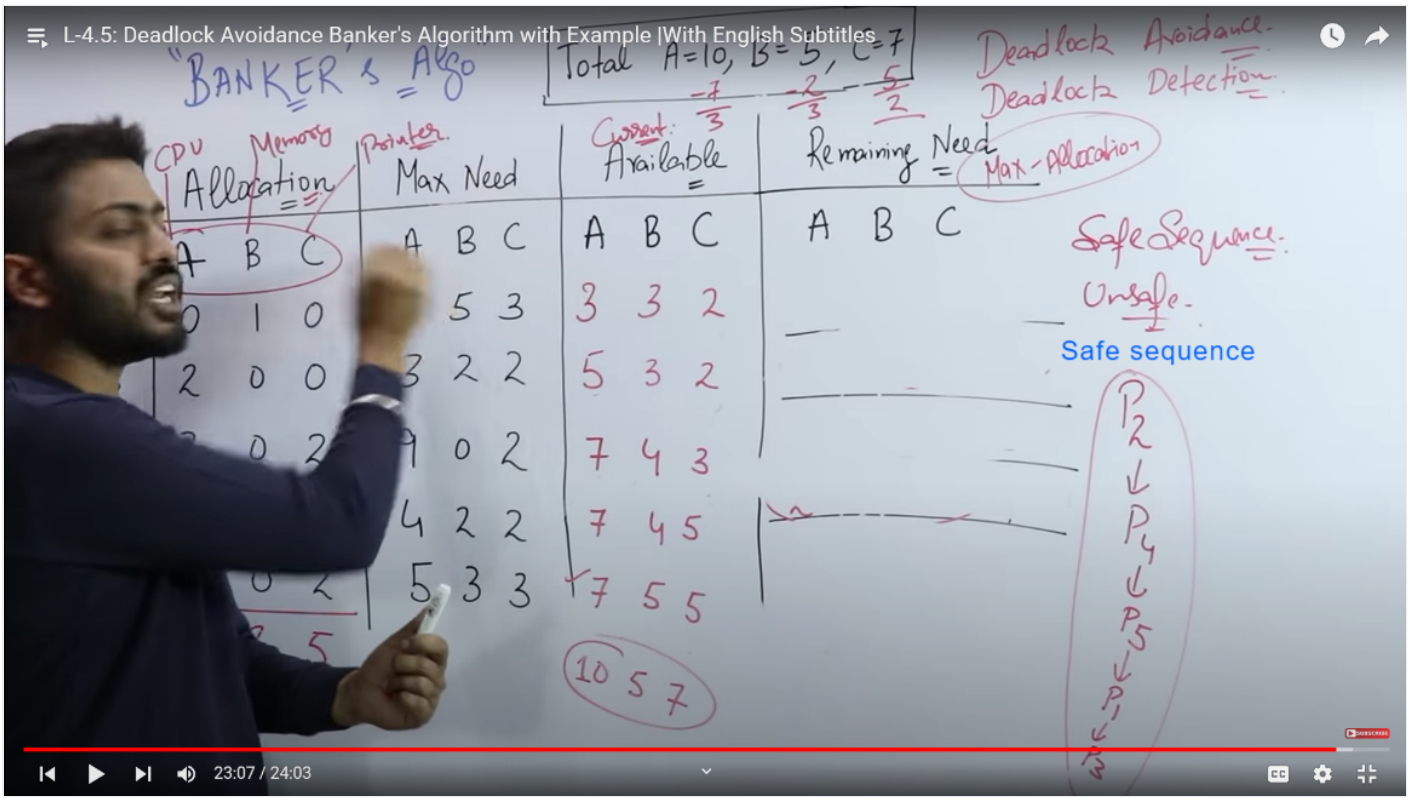
if there is a deadlock then you can use the following :

1. kill the processes or process.
2. resoucre pre-emption



Banker's Algorithm to avoid deadlock :

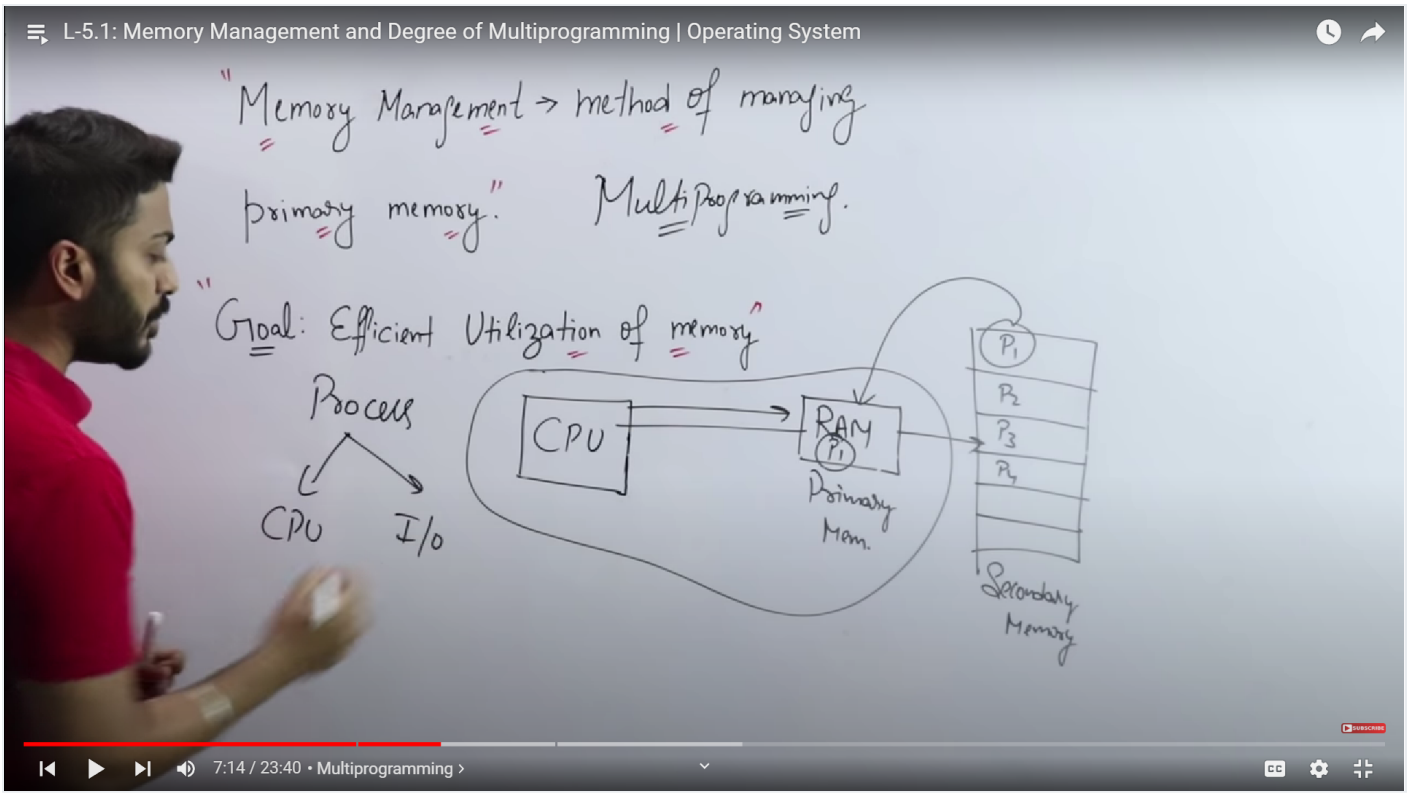


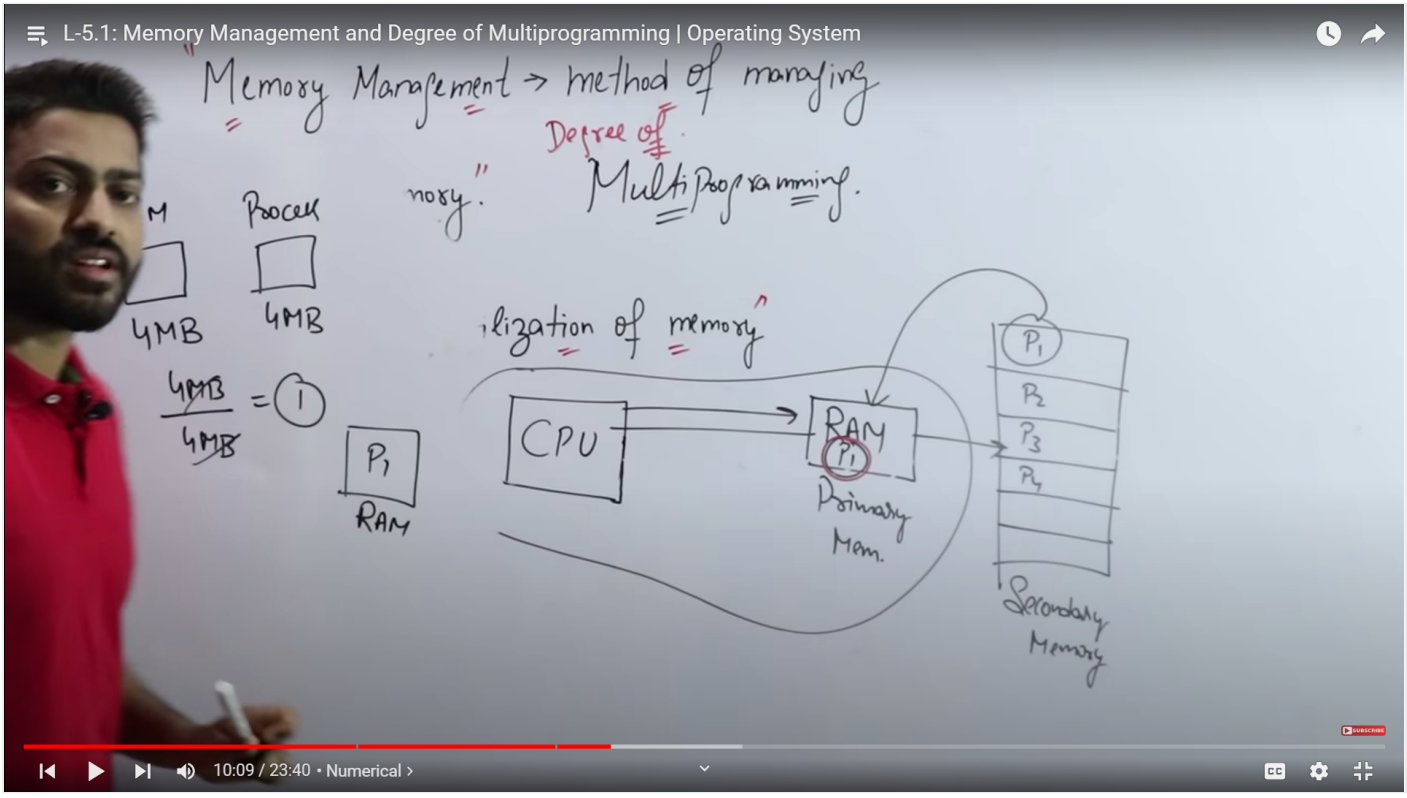


this sequence is not unique it can be change.

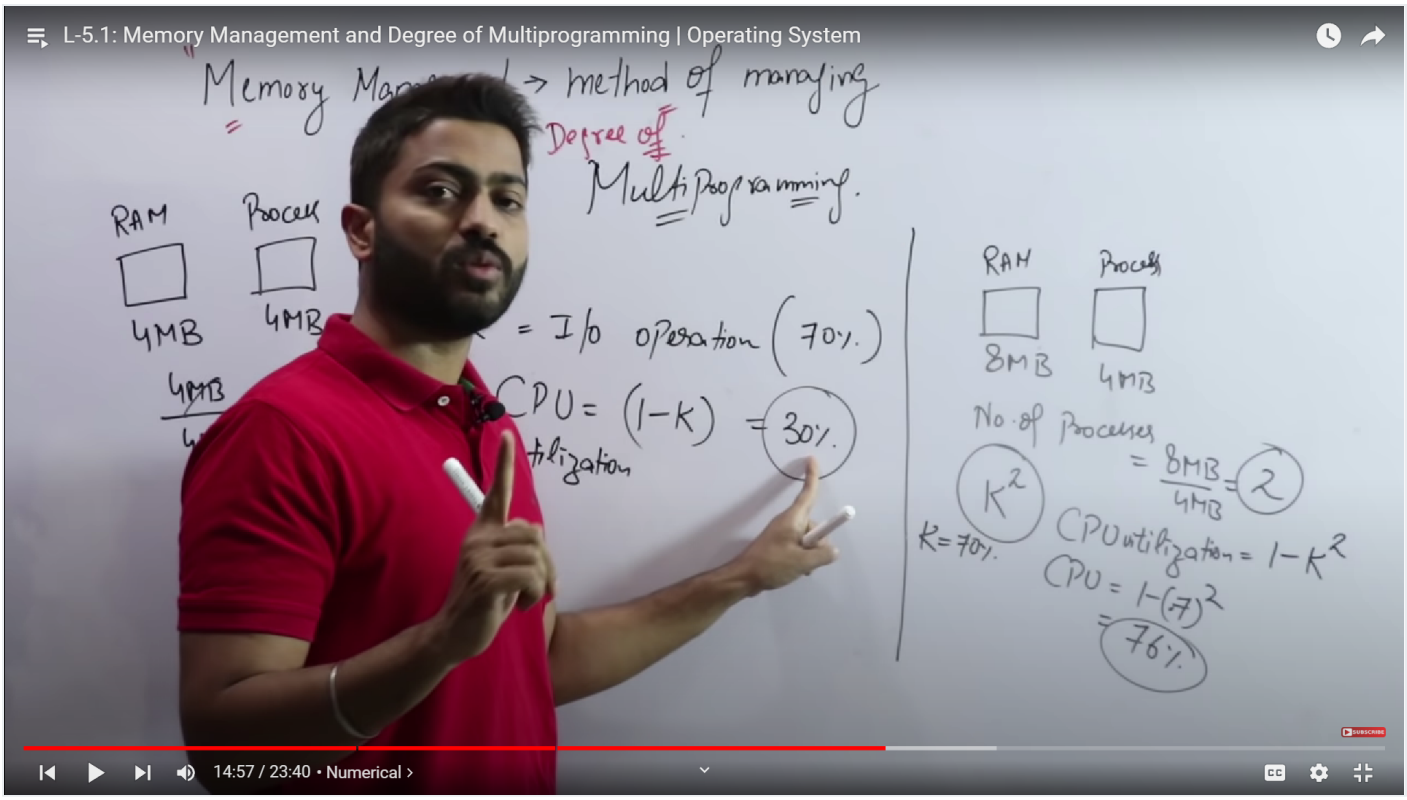
this is not possible in real life as the need of a process is dynamic and changes, but here we take its static need.

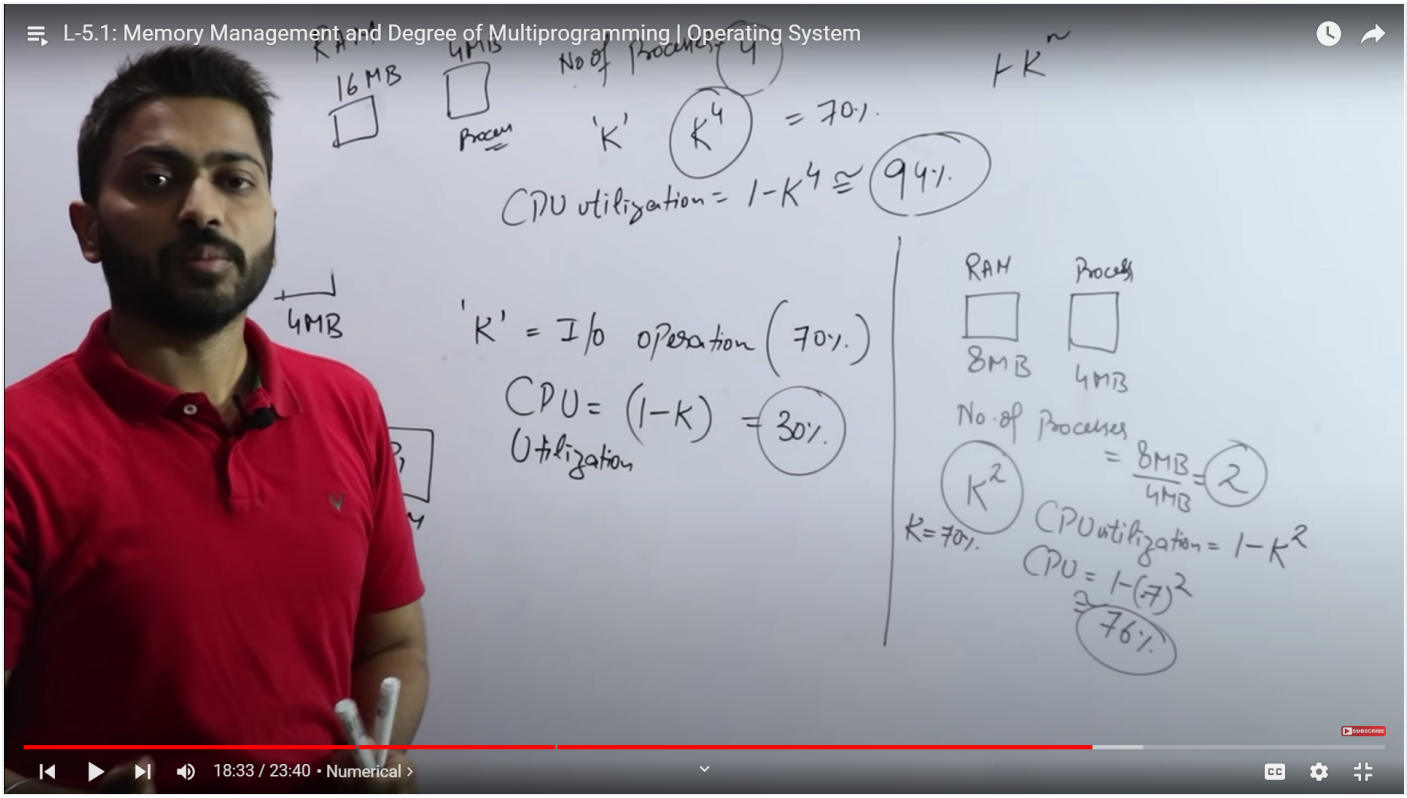
Memory management in operating system.

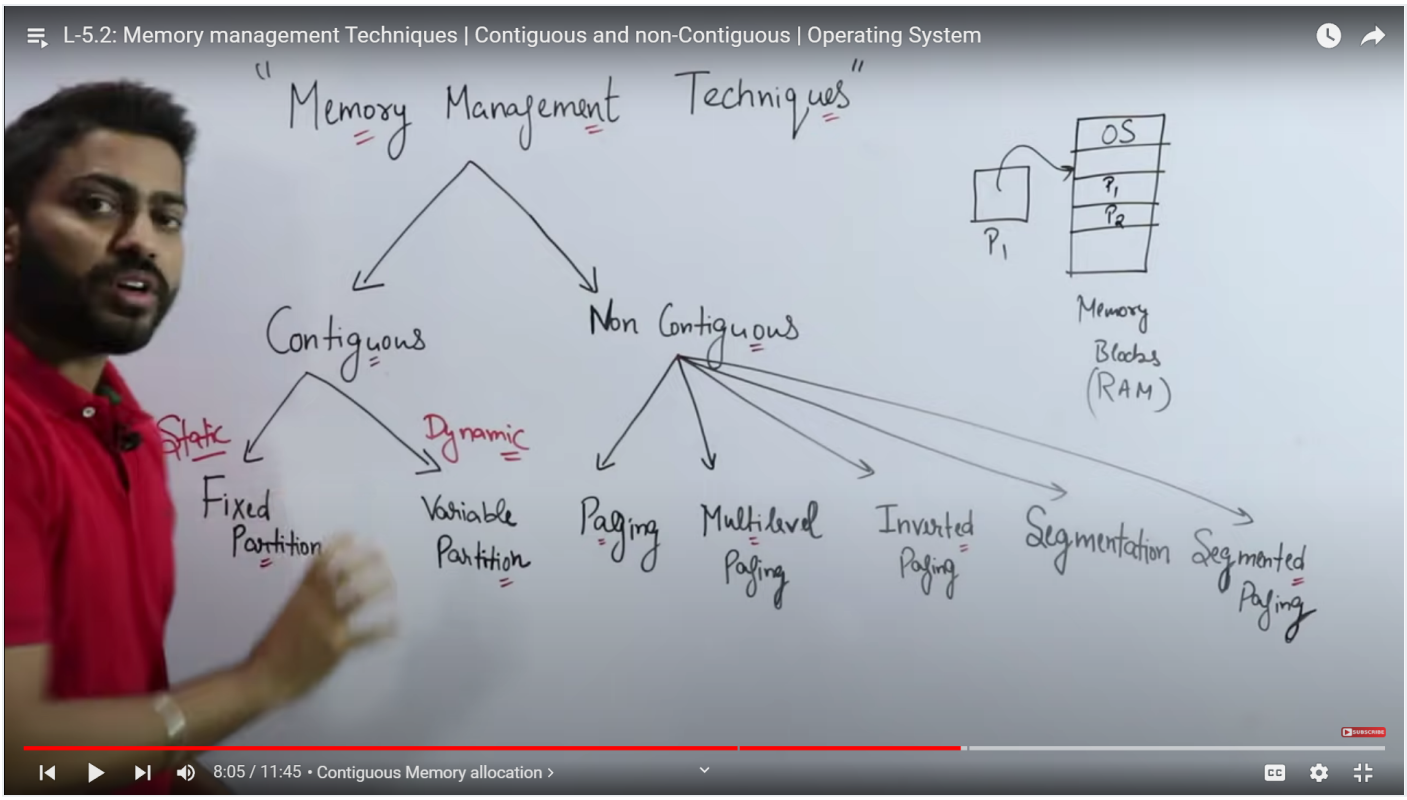




degree of multiprogramming : no of process taken to ram from secondary memory. we need to maximise it.

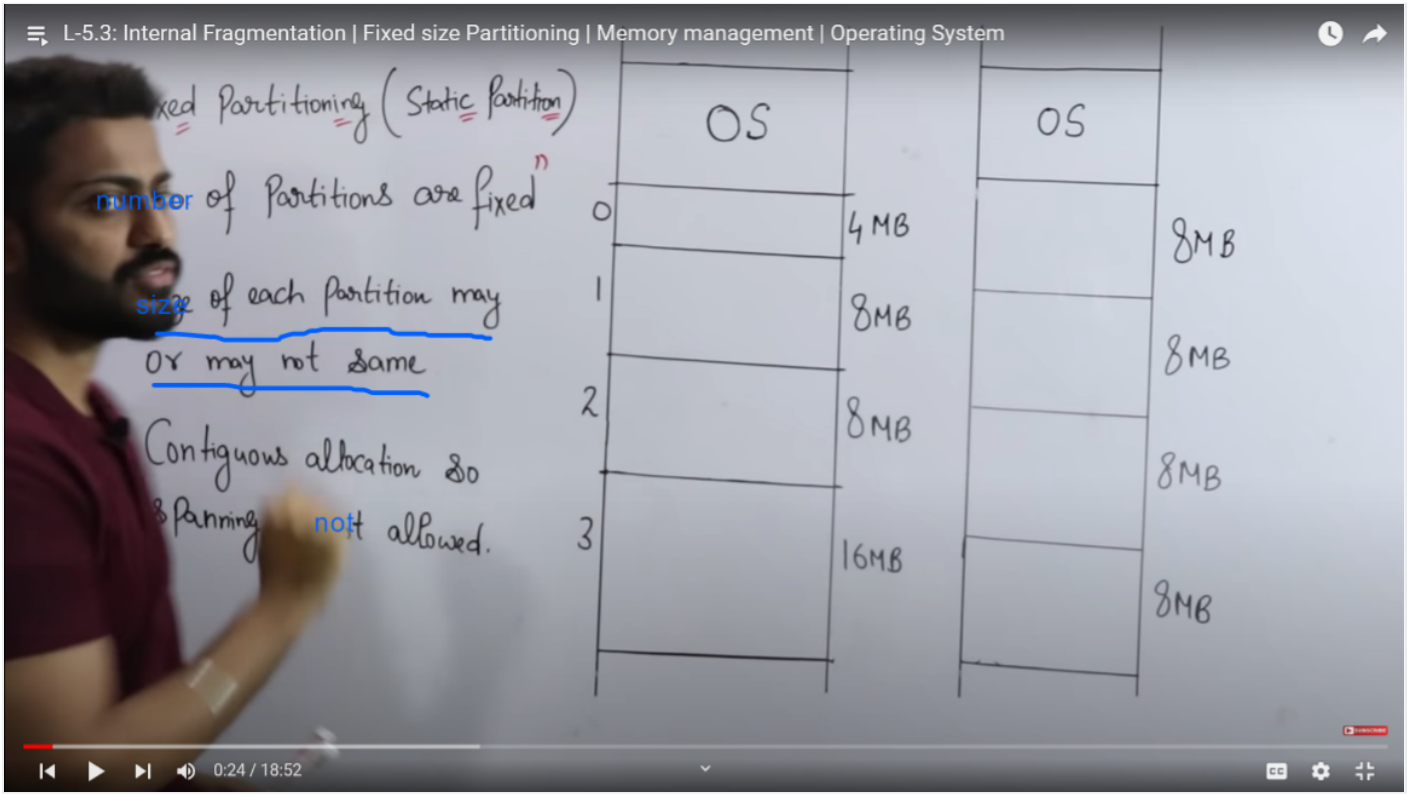

L-5.1: Memory Management and Degree of Multiprogramming | Operating System
Memory Mana → method of managing
Degree of
Multiprogramming.
RAM
Proces
4MB
4MB
= I/o operation (70%.)
CPU = (1-K) = 30%.
Utilization
RAM
Proces
8MB
4MB
No. of Processes = 8MB/4MB = 2
K=70%.
K²
CPU utilization = 1-K²
CPU = 1-(.7)²
= 76%.
14:57 / 23:40 • Numerical ›


L-5.1: Memory Management and Degree of Multiprogramming | Operating System
16MB
4MB
No of Proces
'K'
K⁴
= 70%.
CPU utilization = 1-K⁴ ≅ 94%.
4MB
'K' = I/o operation (70%.)
CPU = (1-K) = 30%.
Utilization
RAM
Proces
8MB
4MB
No. of Processes = 8MB/4MB = 2
K=70%.
K²
CPU utilization = 1-K²
CPU = 1-(.7)²
= 76%.
18:33 / 23:40 • Numerical ›



Continuous Memory Allocation

Static/Fixed partitioning : partitioning is done before arrival of process



problems in fixed partition →

Internal fragmentation : when a process of lesser space than the size of the partition in RAM occurs, then the remaining memory of the partition is wasted and can not be used. this is called internal fragmentation.

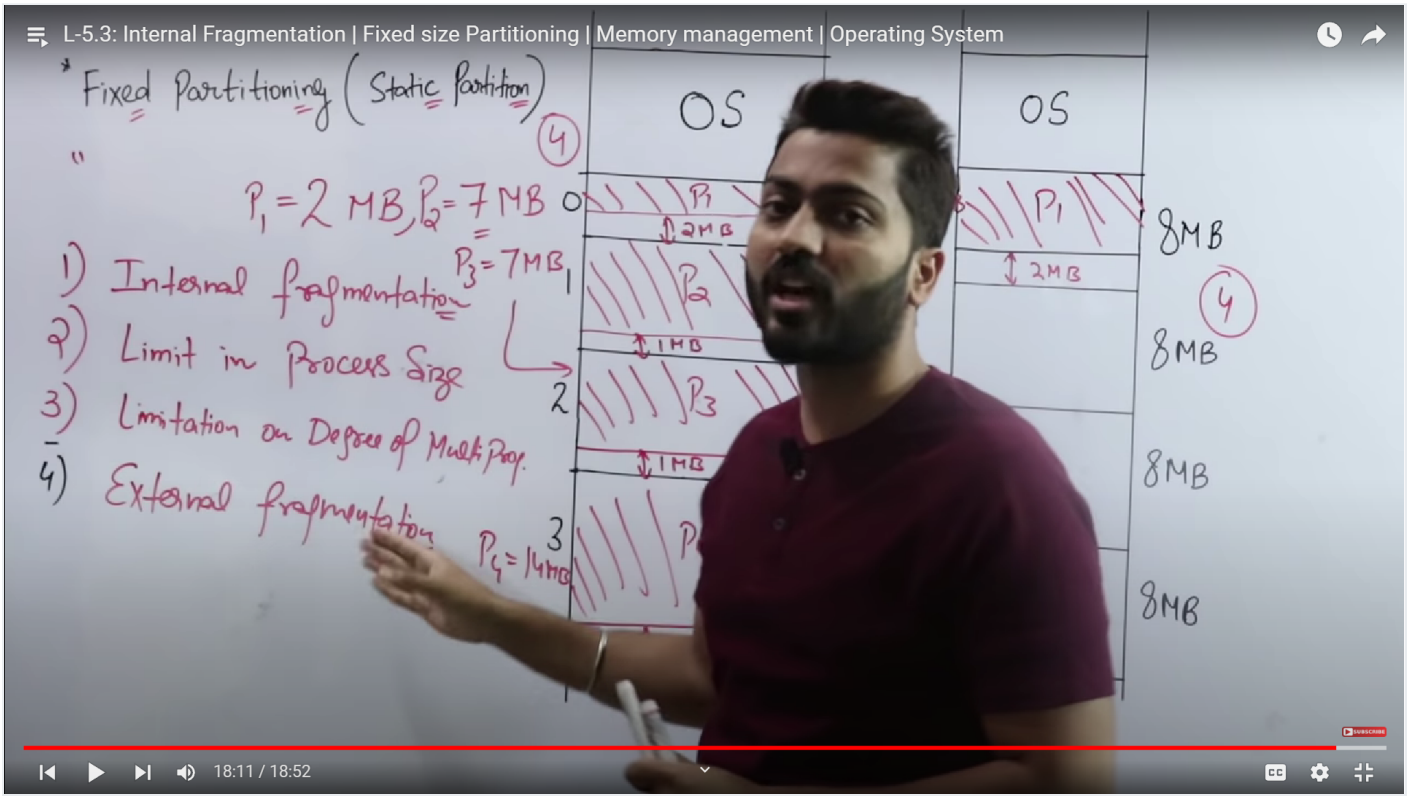
Limit in process size: if a process occurs of larger than the maximum size of partition it cannot be stored in memory.

Limitation on the degree of multiprogramming.

External Fragmentation: Although we are having the availability of memory in different-different sections combine which is more than the required memory, still we cannot accommodate the new process because of continous memory allocation.

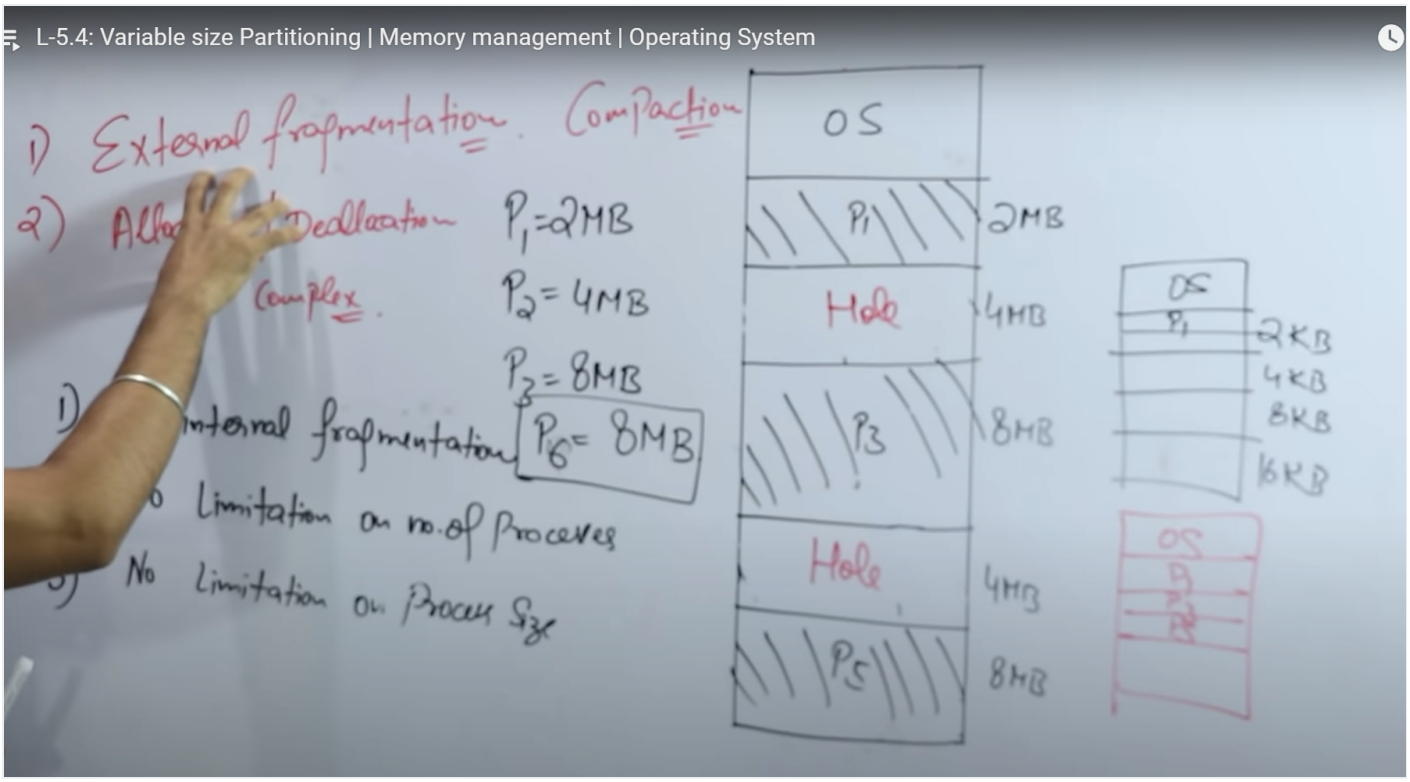
Advantage of fixed partition :

→easy to implement .

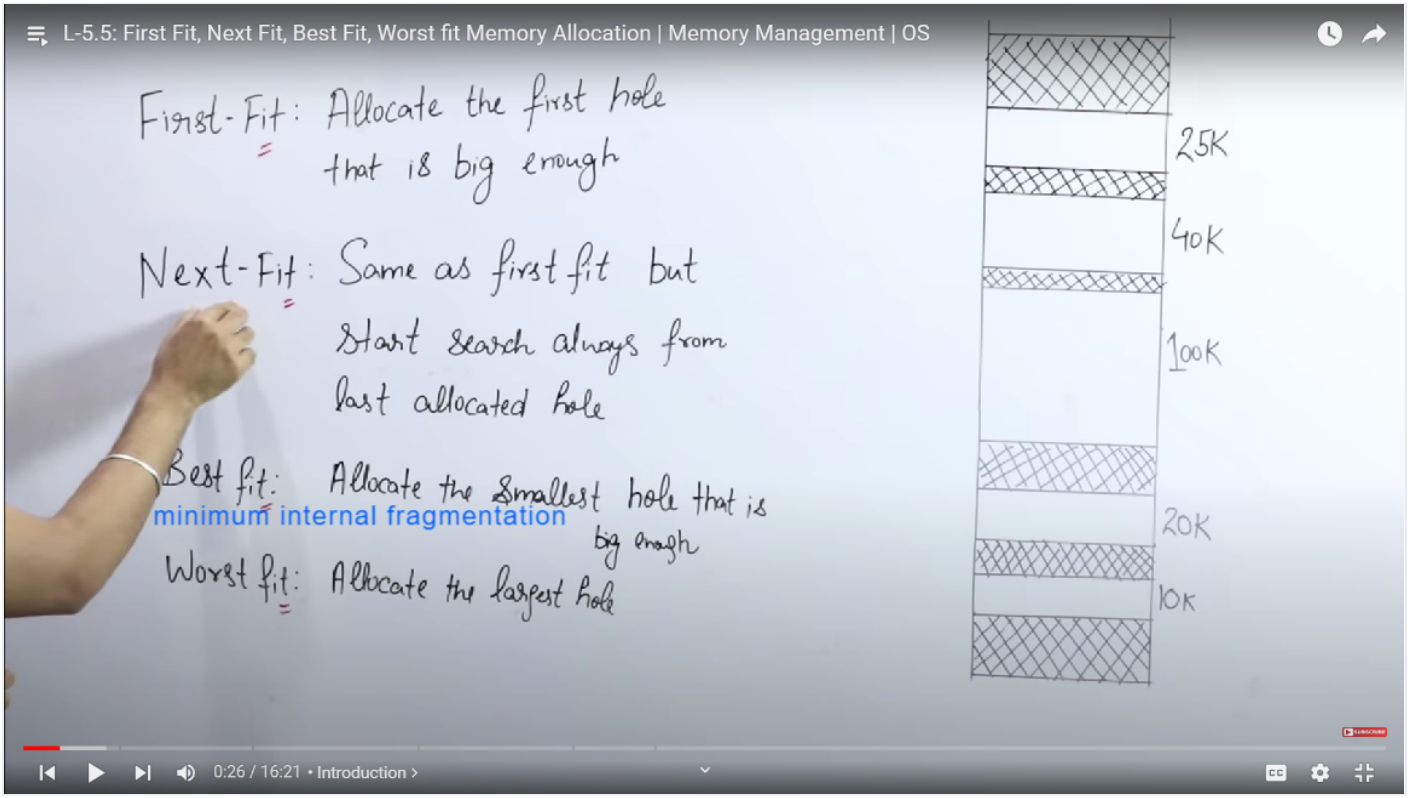


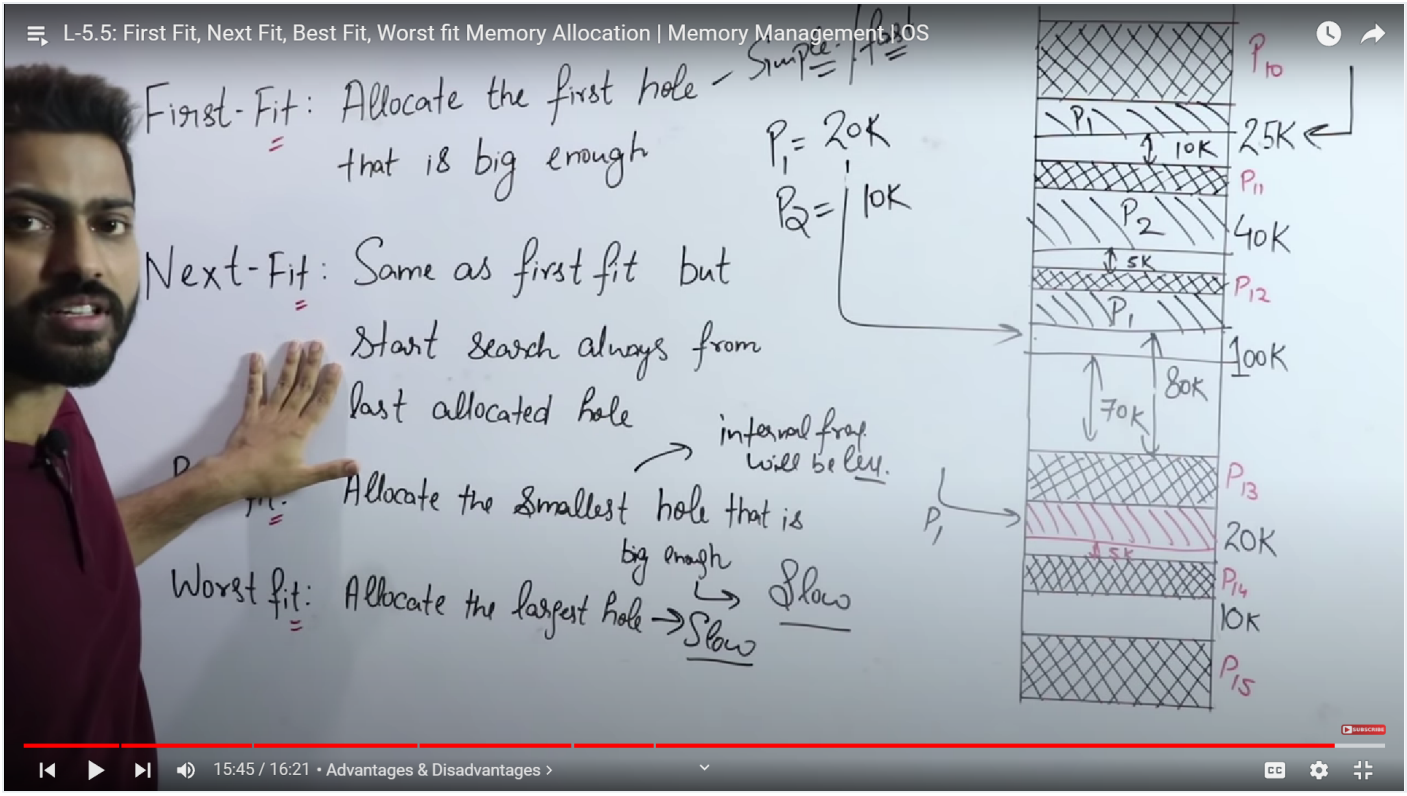
Dynamic/variable partitioning: the partitioning is done in real time as the process comes memory is allocated as per its requirement.

it also suffers with external fragmentation, which can be removed by compaction(to seperate the used memory adn free memory but in that we have to shift the running process to some another memory location which will take a lot of time).



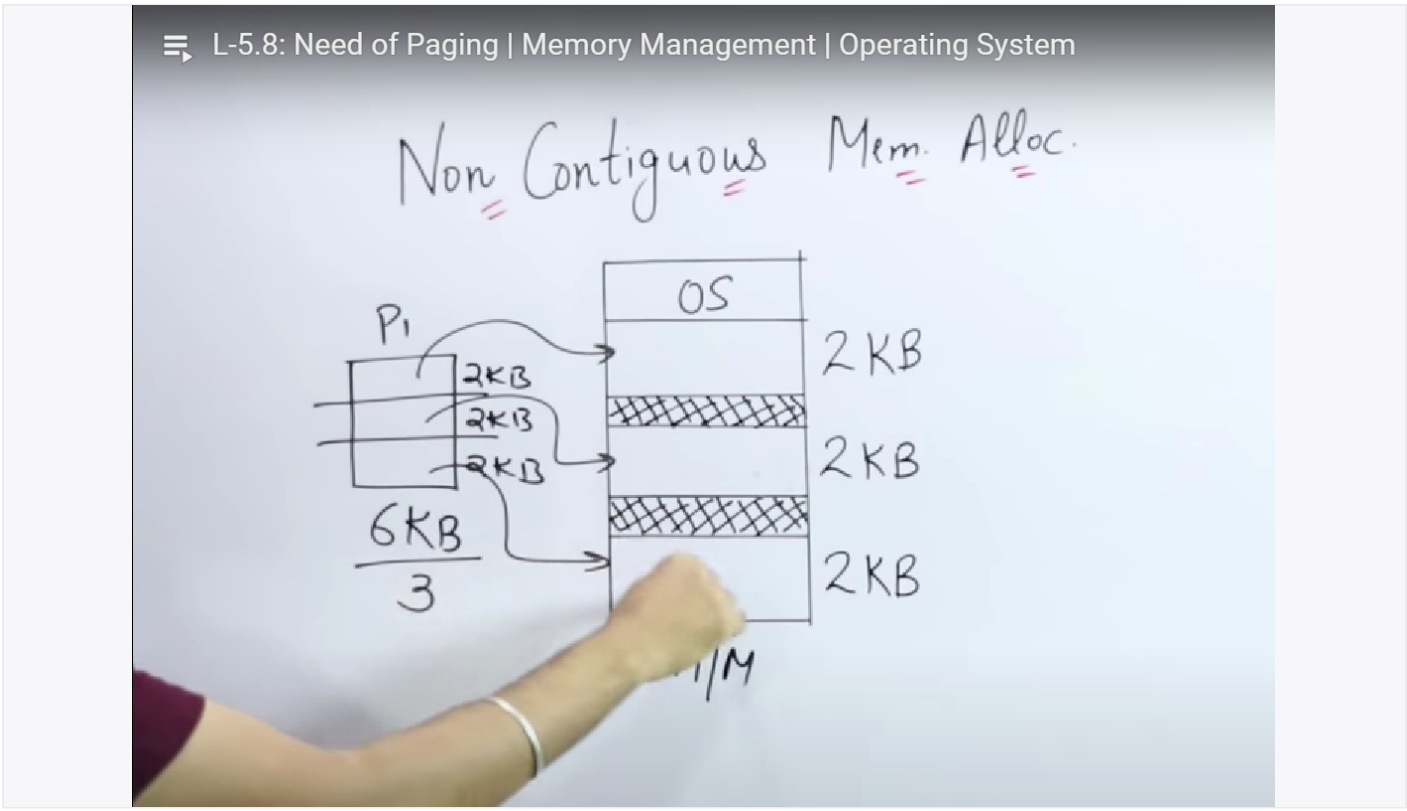
the hole is created in the memory as some process is terminated, so we can fill it with some other processes. to know which process will fit in the hole we can see :



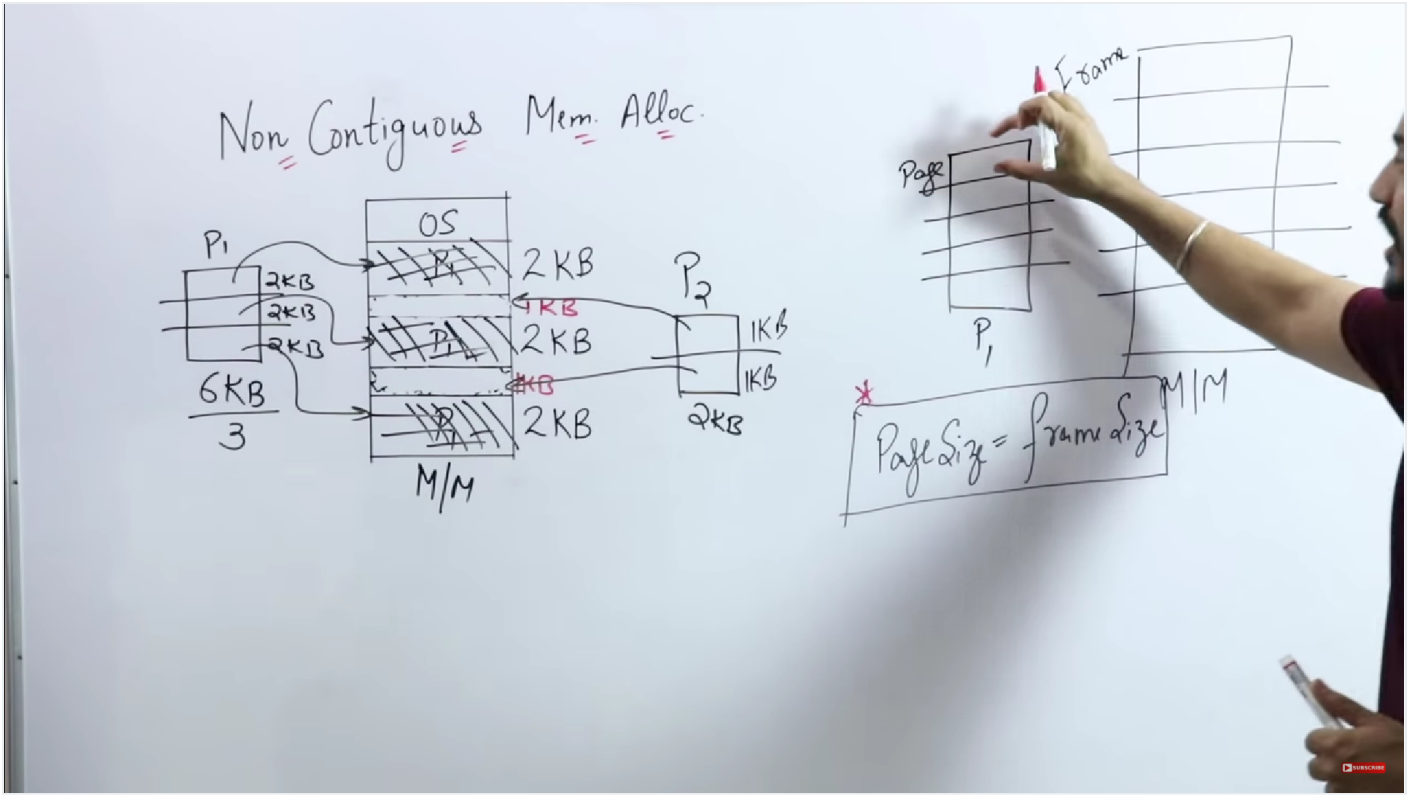


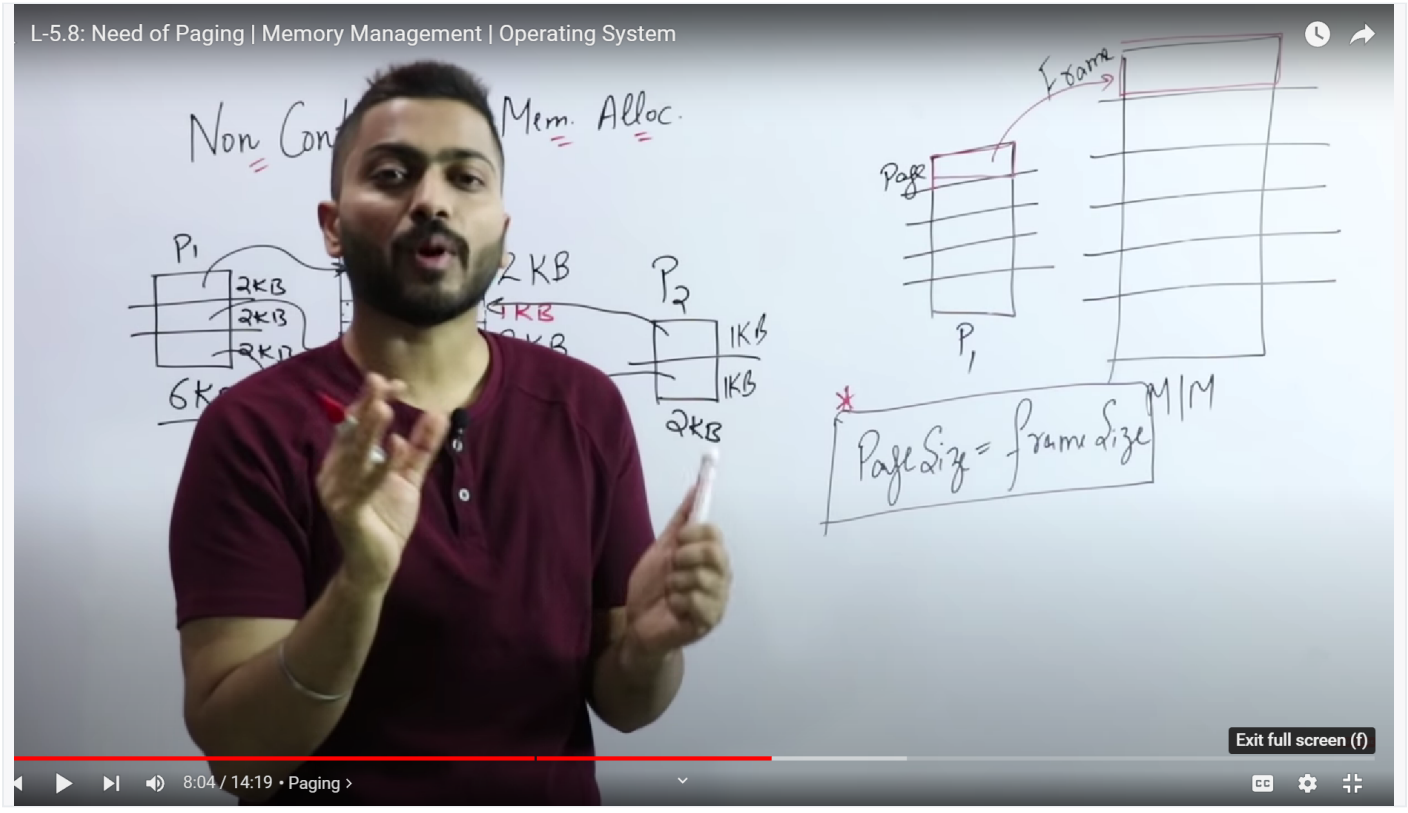
Non-Continous Memory Allocation

→external fragmentation can be presented using non-continuous memory allocation.



this process is very costly and time consuming as the RAM have to analyze again and again and look for the holes as holes changes very rapidly and are very dynamic in nature.



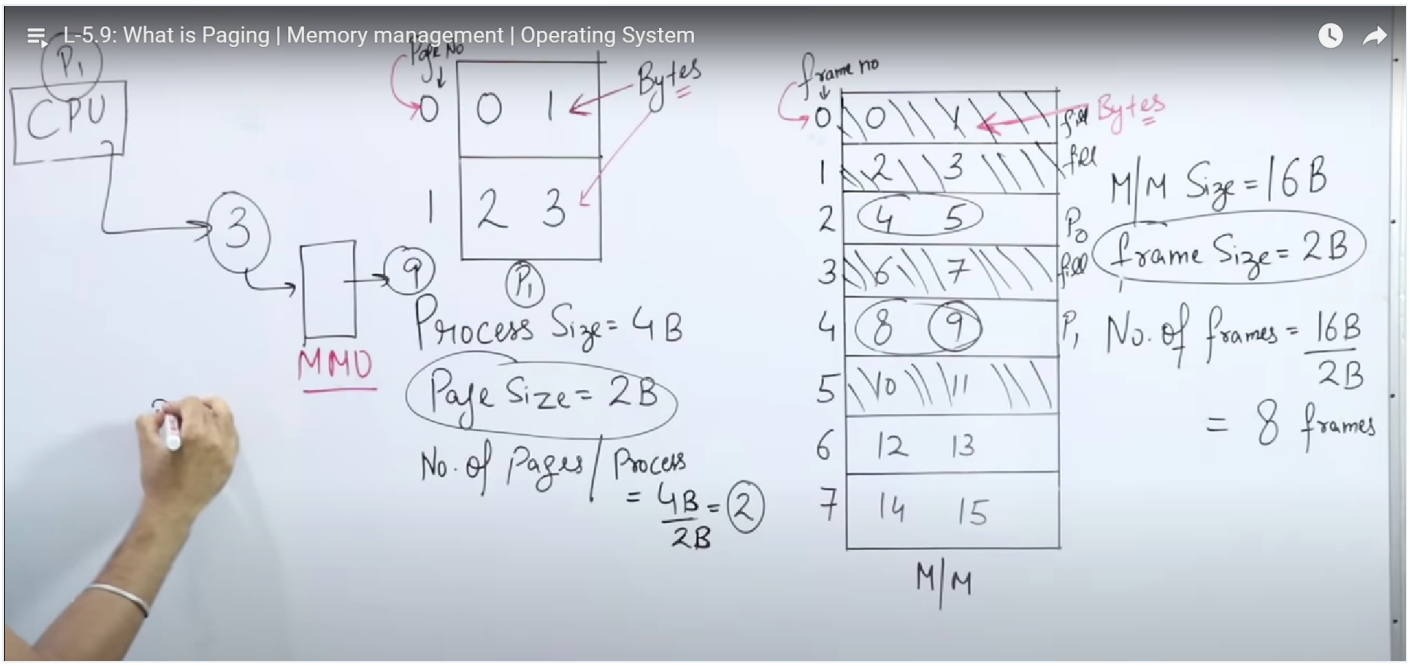


process is splitted into pages in secondary memory and main memory(RAM) is splitted into frames.

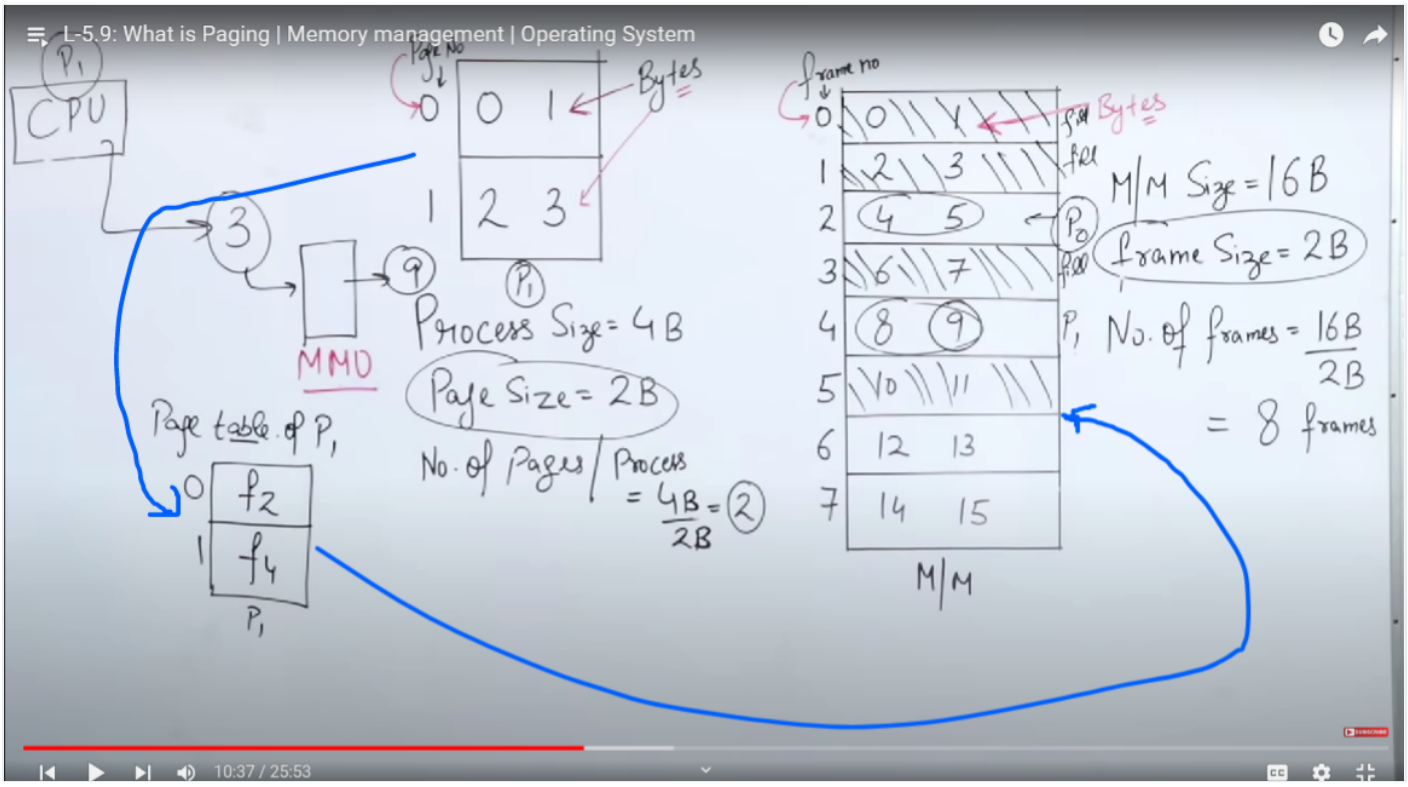
the advantage is paging is external fragmentation is completely removed.

mapping: when cpu request a byte it can be anywhere in memory so someone is requered to bring that byte from memoring then MMU- memory management unit does this work of mapping. it converts the CPU generated address to absolute address.

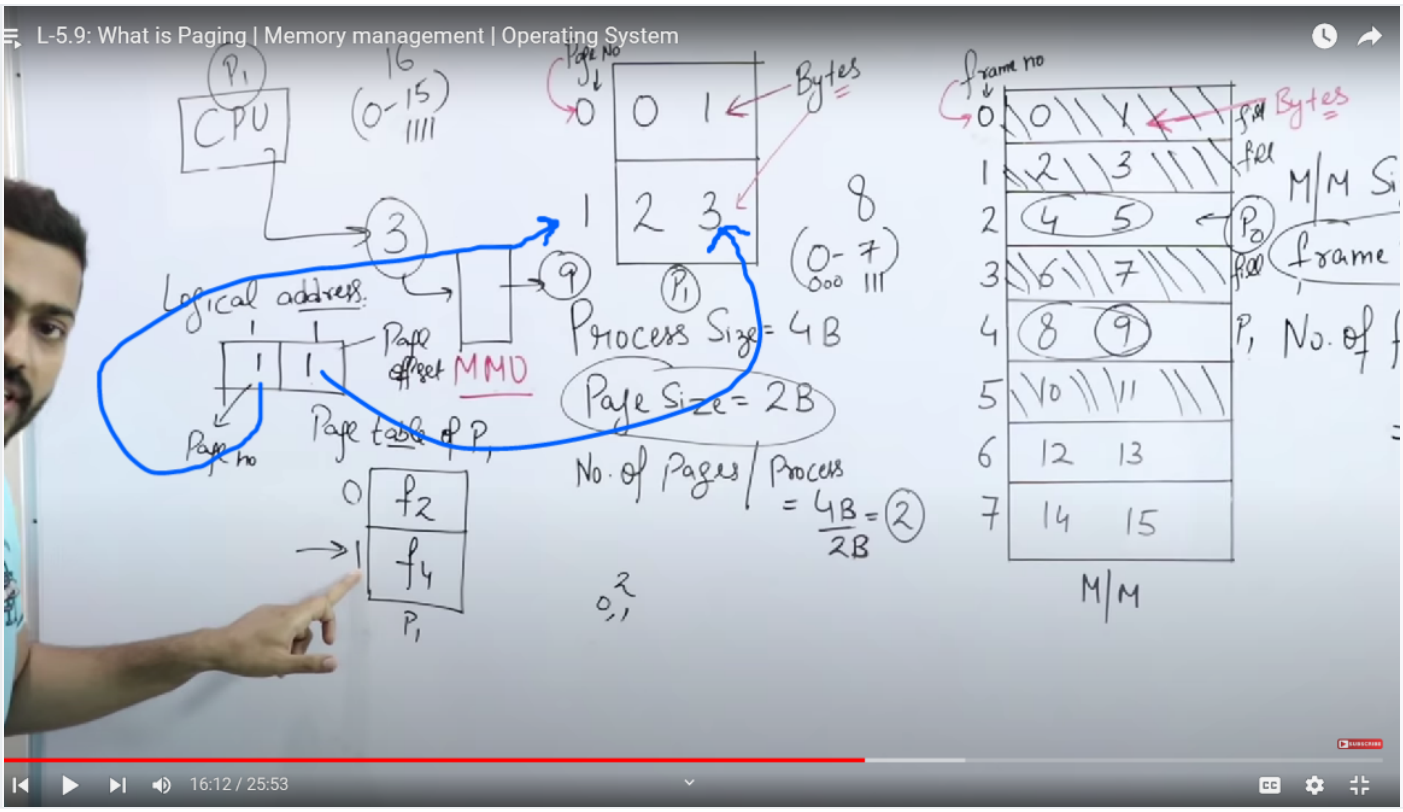
⇒for this MMU use page table.



page table contains the frame number. every process has its own page table.

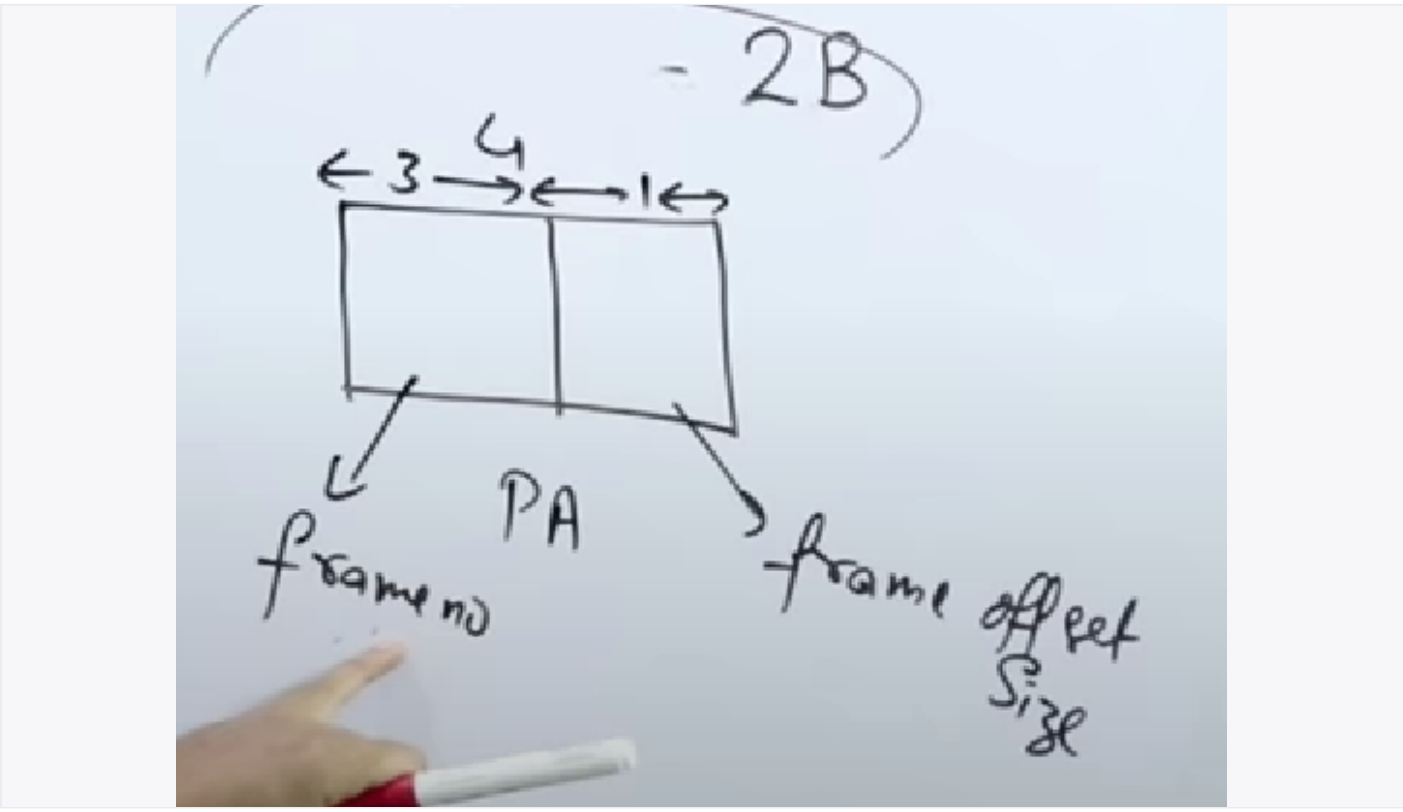


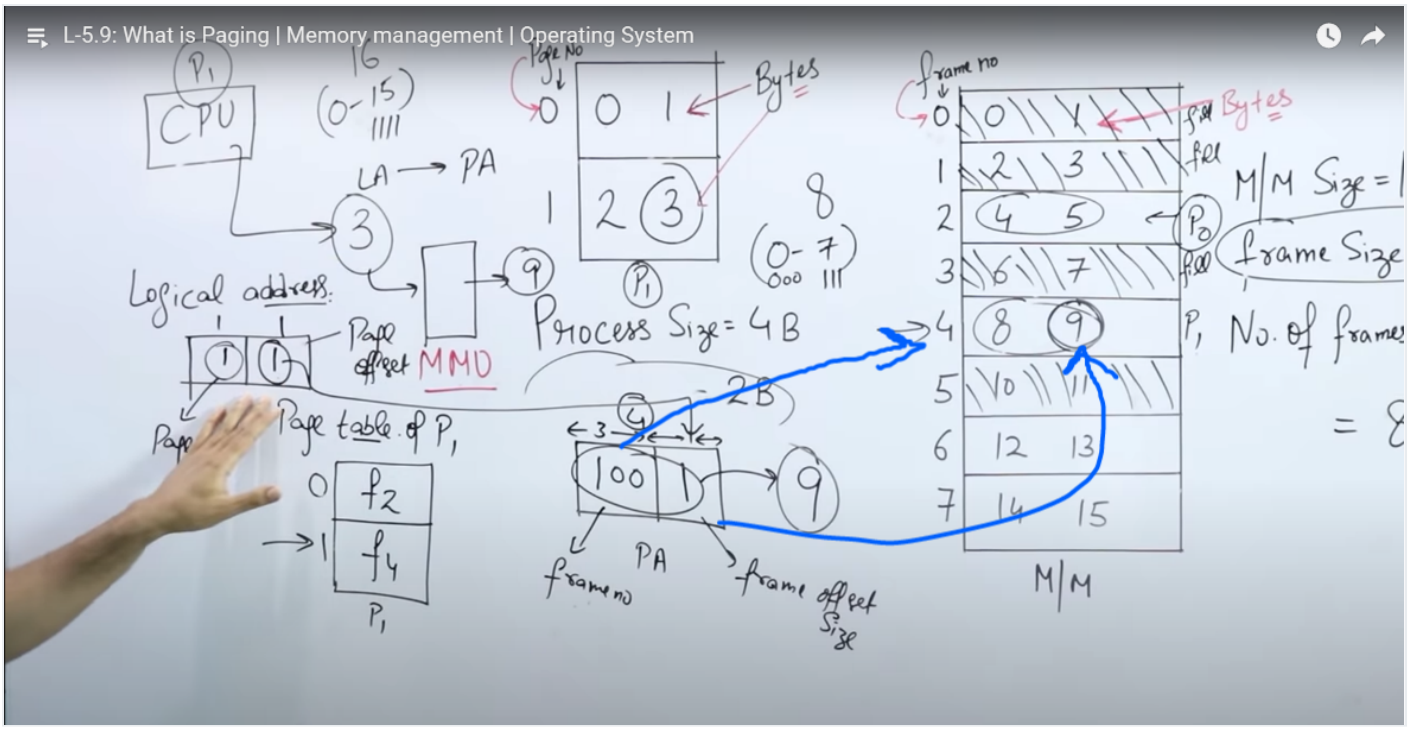
cpu works on logical address which contains 2 things: page number and page offset.



now logical address have to be converted to physical address which is related to main memory.

⇒ physical address:





Paging is **a storage mechanism used in OS to retrieve processes from secondary storage to the main memory as pages**. The primary concept behind paging is to break each process into individual pages. Thus the primary memory would also be separated into frames.

Thrashing

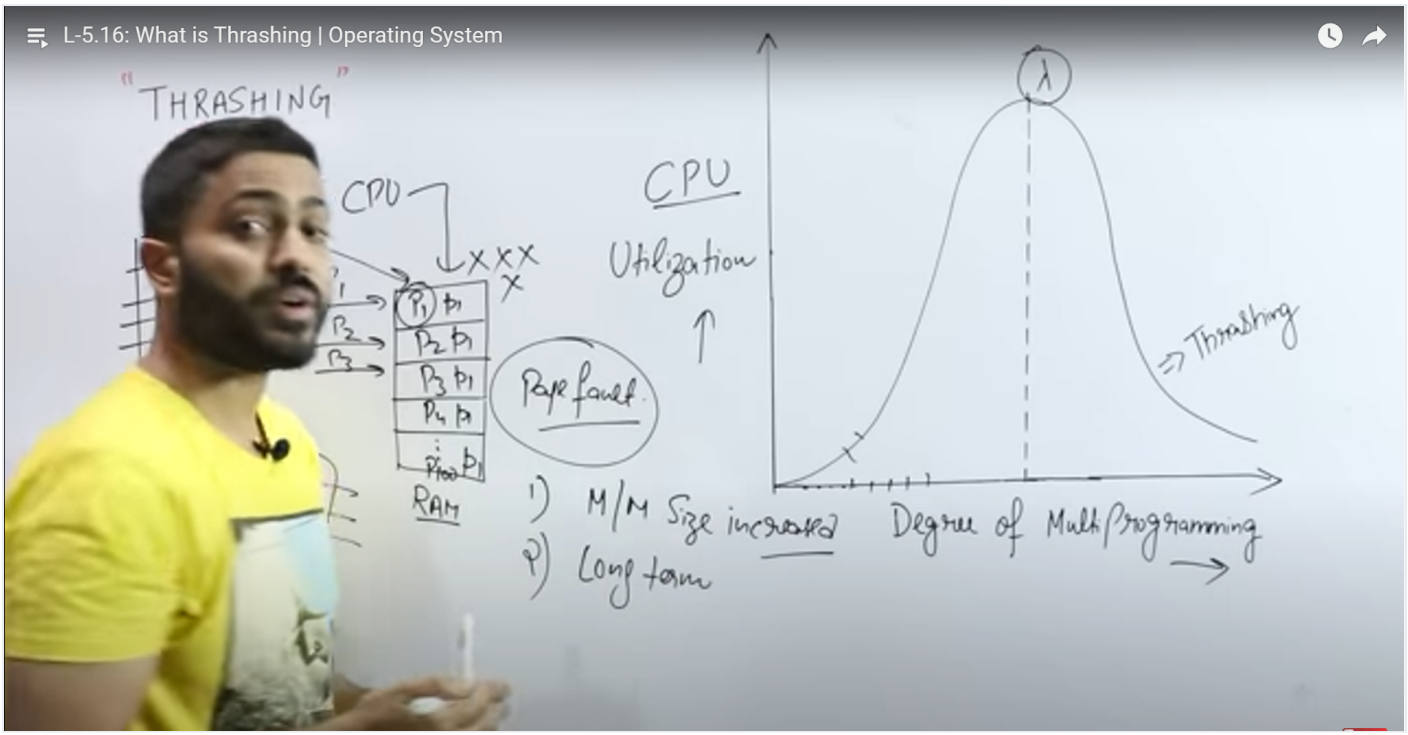
degree of multiprogramming can be maximum, if 1 page of process is brought to the RAM.

Thrashing, is the moment when the CPU utilization is maximum and then after there will be a lot of page faults and most time will be required to service the page, that is called thrashing.

Thrashing is **when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages**. This state in the operating system is known as thrashing. Because of thrashing, the CPU utilization is going to be reduced or negligible

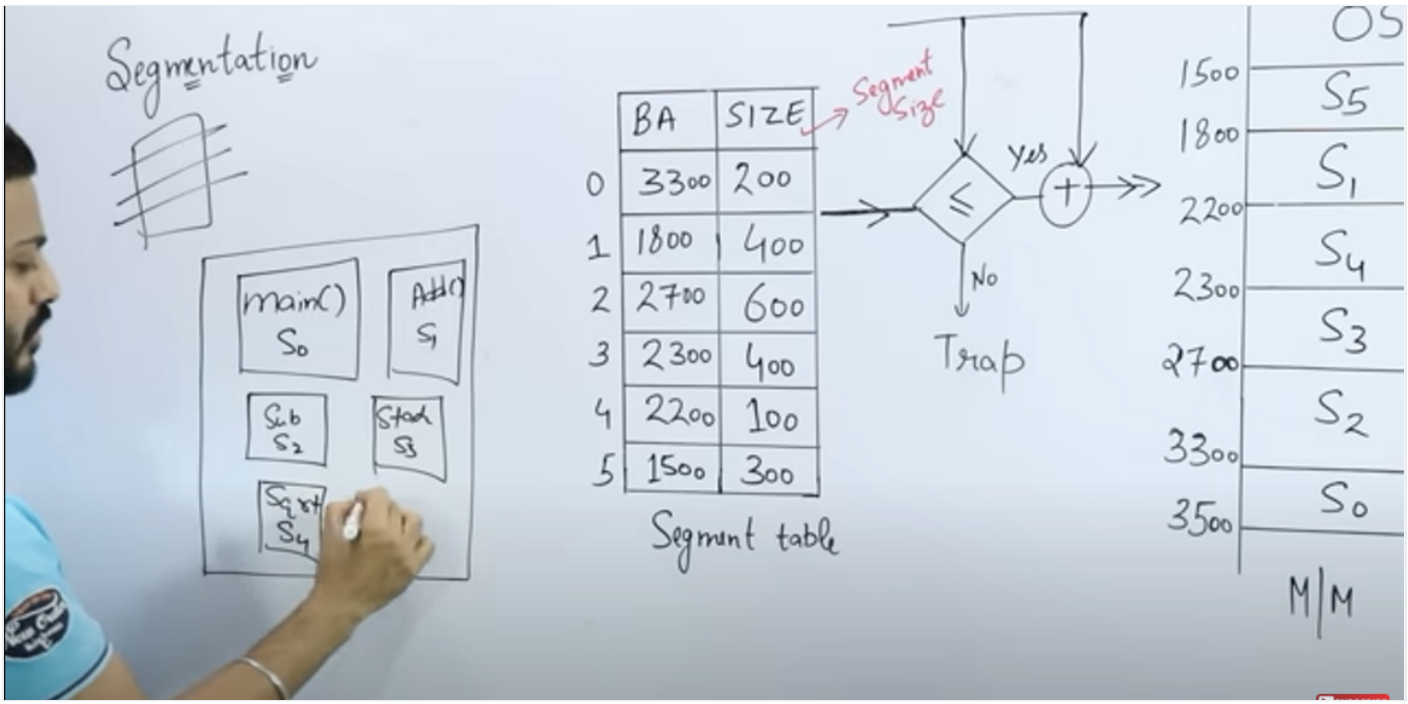
Thrashing can be removed by :

1. increase the size of main memory
2. reduce the speed of long term scheduler.



Segmentation - it is a method in which a process is divided into parts and put into the main memory.

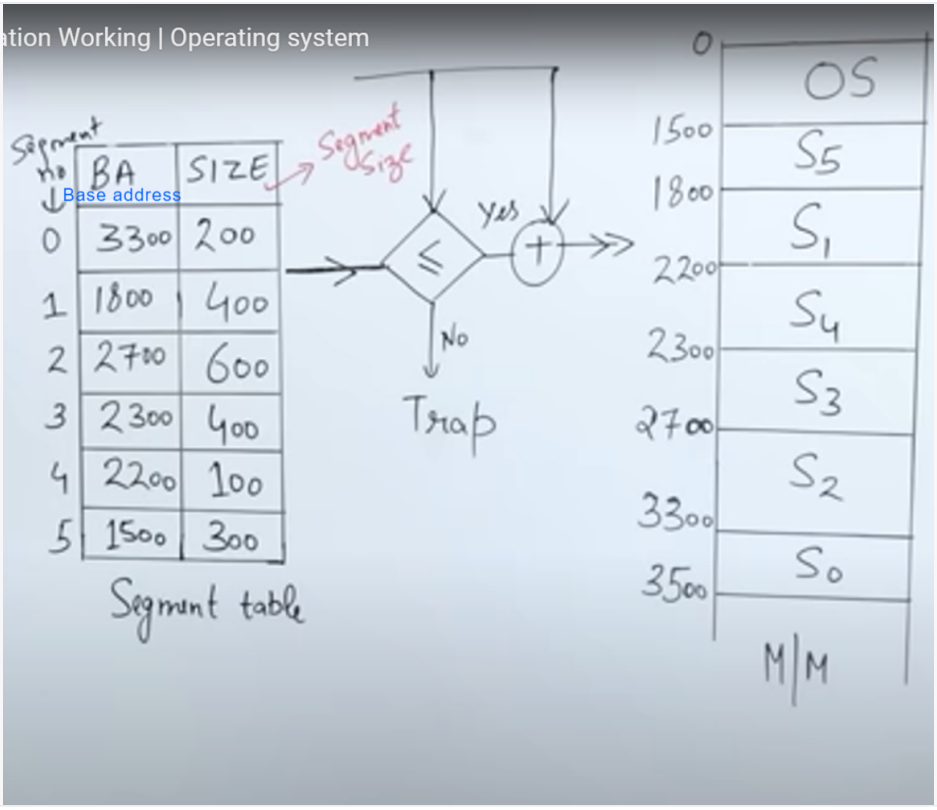
it works on user point of view.



size of page in paging is same, but in segmentation size of segment are diffrent.

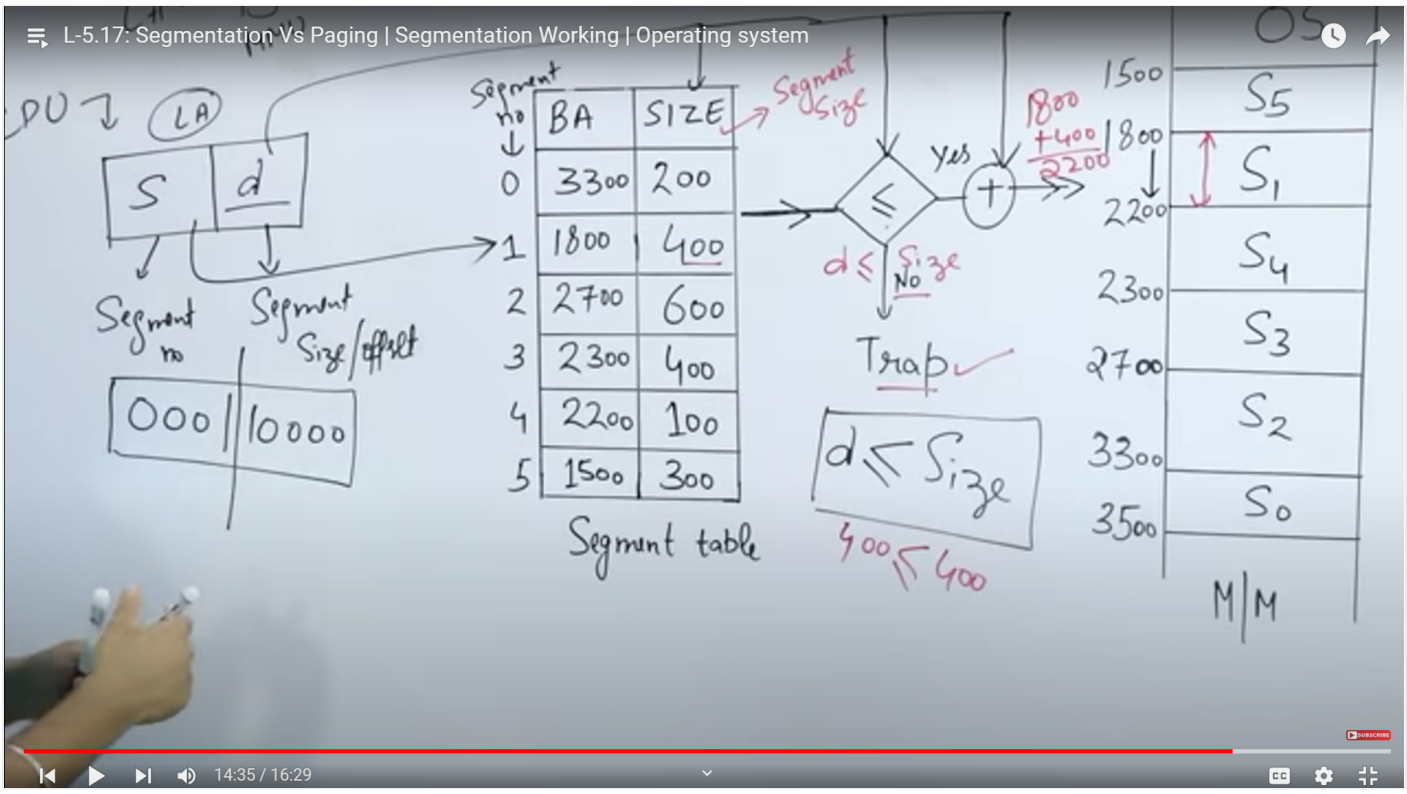
segement can be of various size.

segment table is used to know the physical address of the segment in main memory



pagging has fixed size of pages so it suffers with internal fragmentation.

segmentation has variable size of segments, so it suffers with external fragmentation



cpu will generate logical address which have two parts segment no. and other is distance to which size we have to read the data from main memory.

if the distance <= size then it is ok otherwise it will go in a trap.

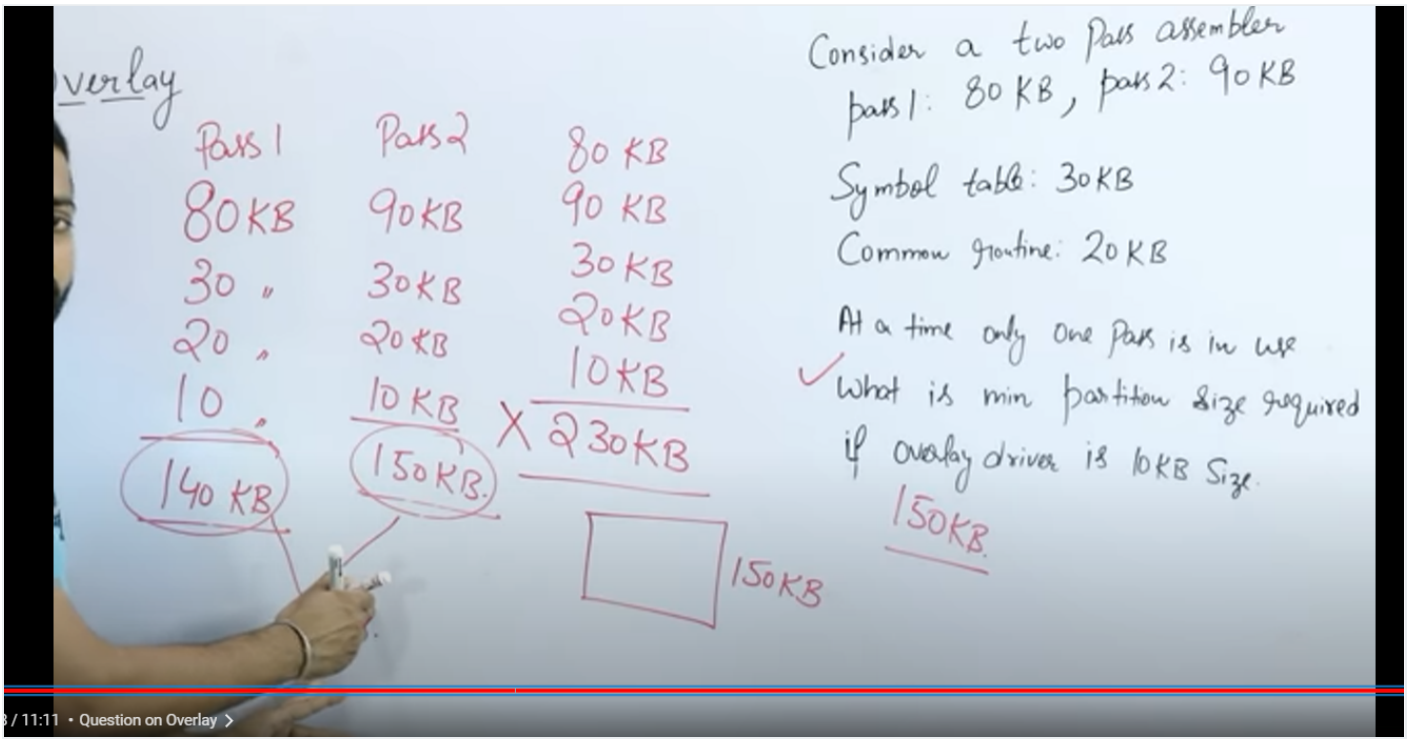
overlay: it is a method in which a process larger than RAM size is put into the main memory.

in this we can divide the process (object code) and bring the functionality we need first into the maain memory.

the concept of overlay is specially used in embeded systems.

the OS has no driver ti divide the process, the user himself have to divide the process.

the thing we have to look after is that the division should be independent, it should not be like the division we are using doesn't have sufficient information or the division has some code of some other division.



Virtual memory

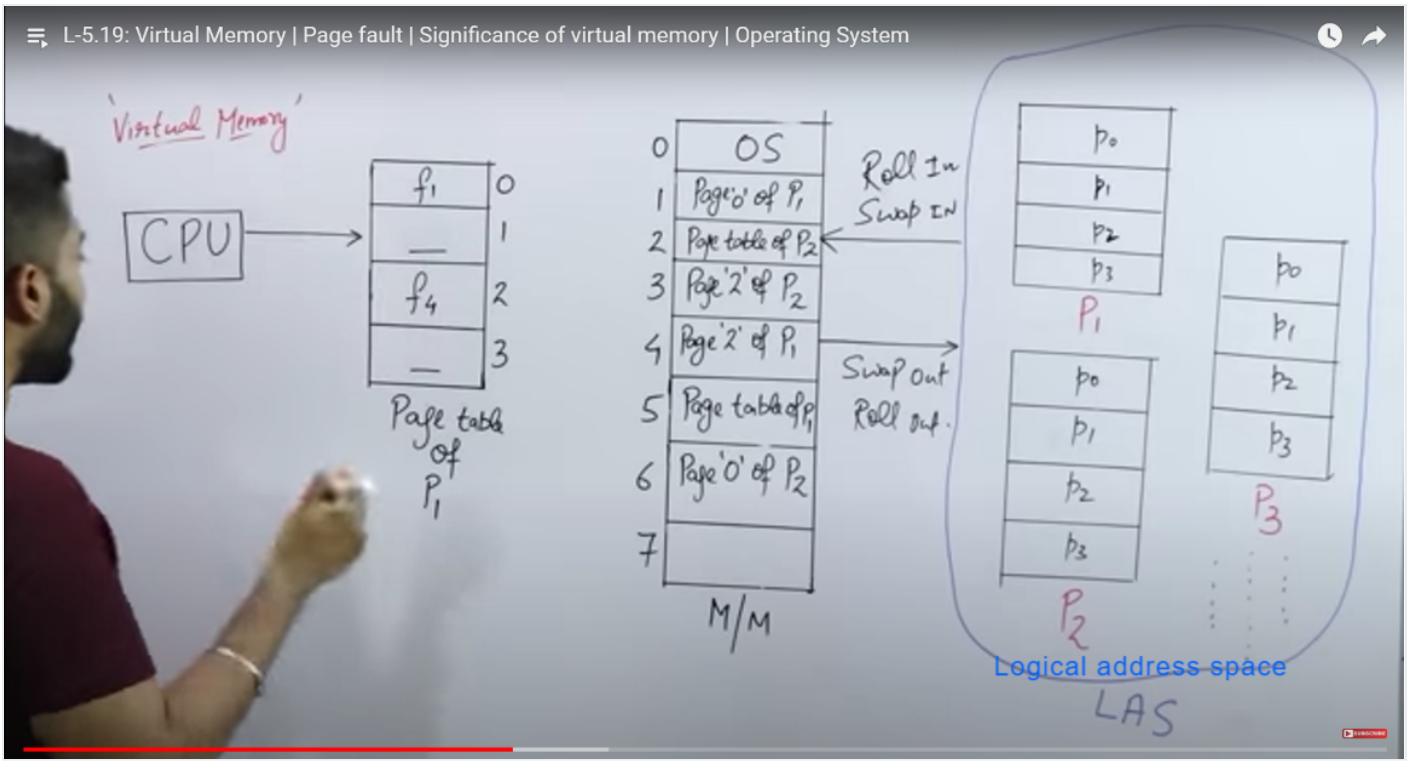
it is a concept.

its significance is it provides a illusion to the programmer that a process with size larger than the RAM can also be executed.

⇒advantage:

→no limitation on size of process

→no limitation on number of processes



it brings only some pages from harddrive to main memory which are required and also uses locality of refrence which means some pages which is related ton the page taken to RAM is also taken with it. and when the work regarding the page is done it roll out to the secondary memory and other pages are roll in the main memory.

page fault: when a cpu demands for a page and the page is not present int the main memory, its valid/invalid bit is 0 then there is a page fault.

how to service page fault?

when the page is not present in the M|M then the trap is generated and the control is taken from the user and given to the OS.

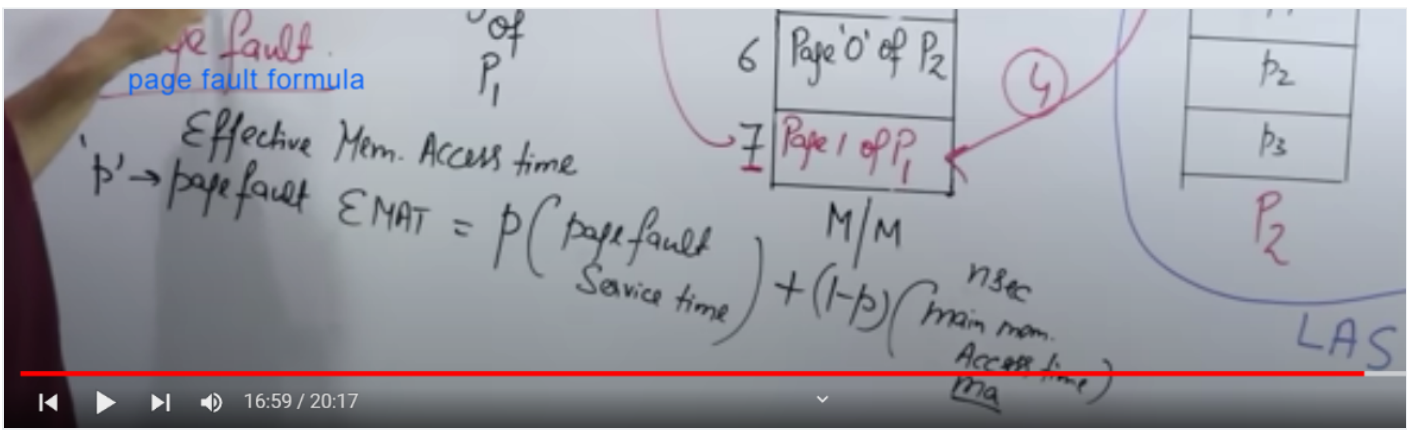
then the OS activates and authenticated the user for security purpose.

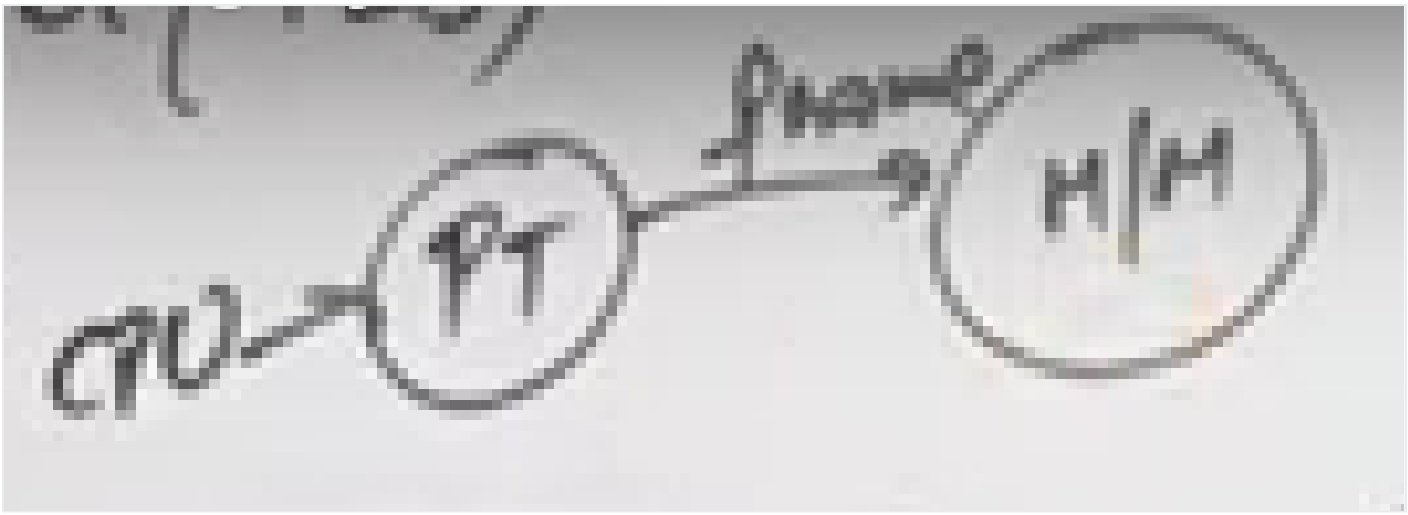
then OS will go to Logical address space bcoz all the data and pages are stored in the LAS(if not the data the address is).

the page will be searched in LAS and after finding it, it will be placed in some free slot of main memory and address of the page (frame number) is stored in the page table.

and controll is given back to the user.

this service is very time consuming.

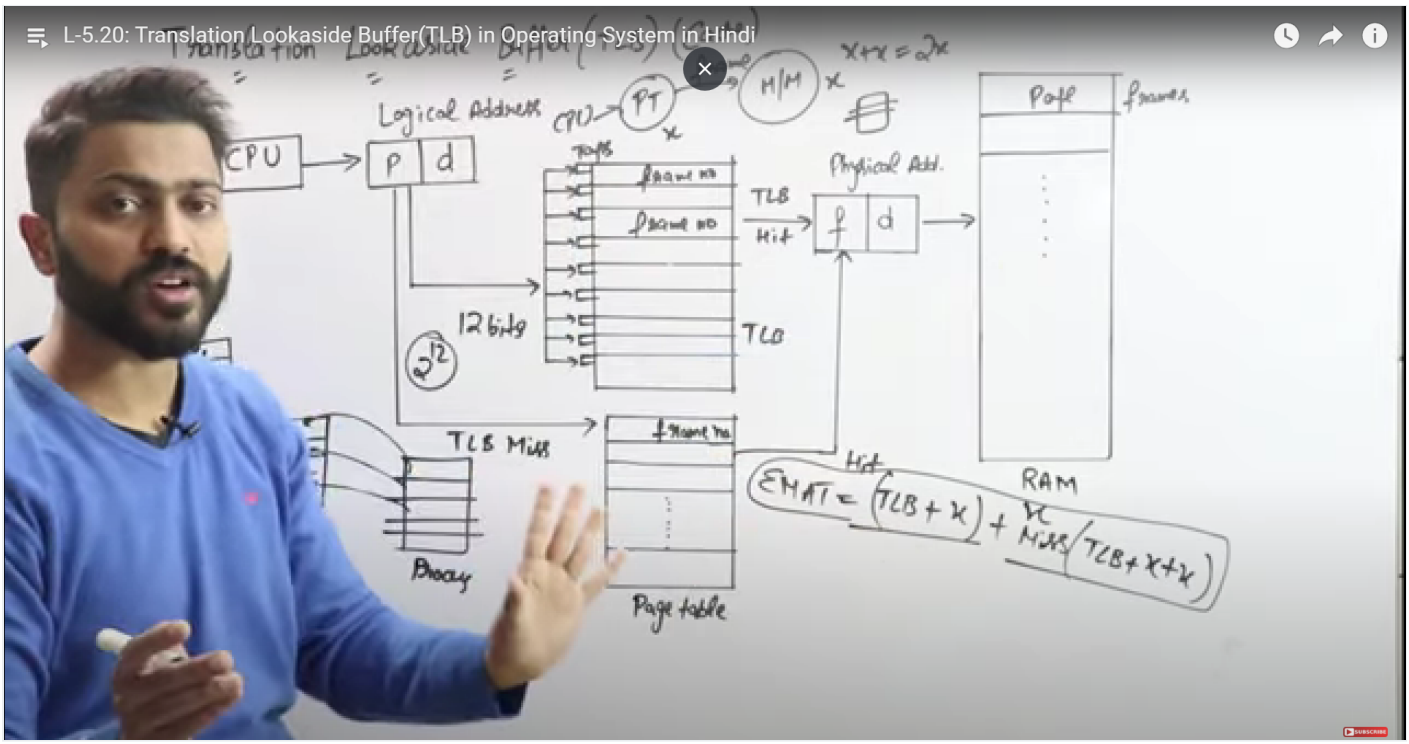




Translation look aside buffer:

when we usually get the logical address we first go to the page table which is in the main memory and from it we get the frame number with the help of it we get the page from the main memory and this all takes a lot of time.

To counter this we use a Translation lookaside buffer(TLB), it's a cache memory which is generally fast, having tags. tags store some frame numbers. if the page's frame number asked by the CPU is present at any of the tags then it's a TLB hit it will directly get the frame no. and get the page from the main memory and if it finds the frame no. then it's a TLB miss we have to look back to the page table and then again to amin memory. and there also can be a case of a page fault.



Page Replacement : the motive is to reduce page fault.

the page is taken to main memory from LAS and if there is no space left in main memory then a pre-existing page will be swaped out and new page is swaped in (replaced).

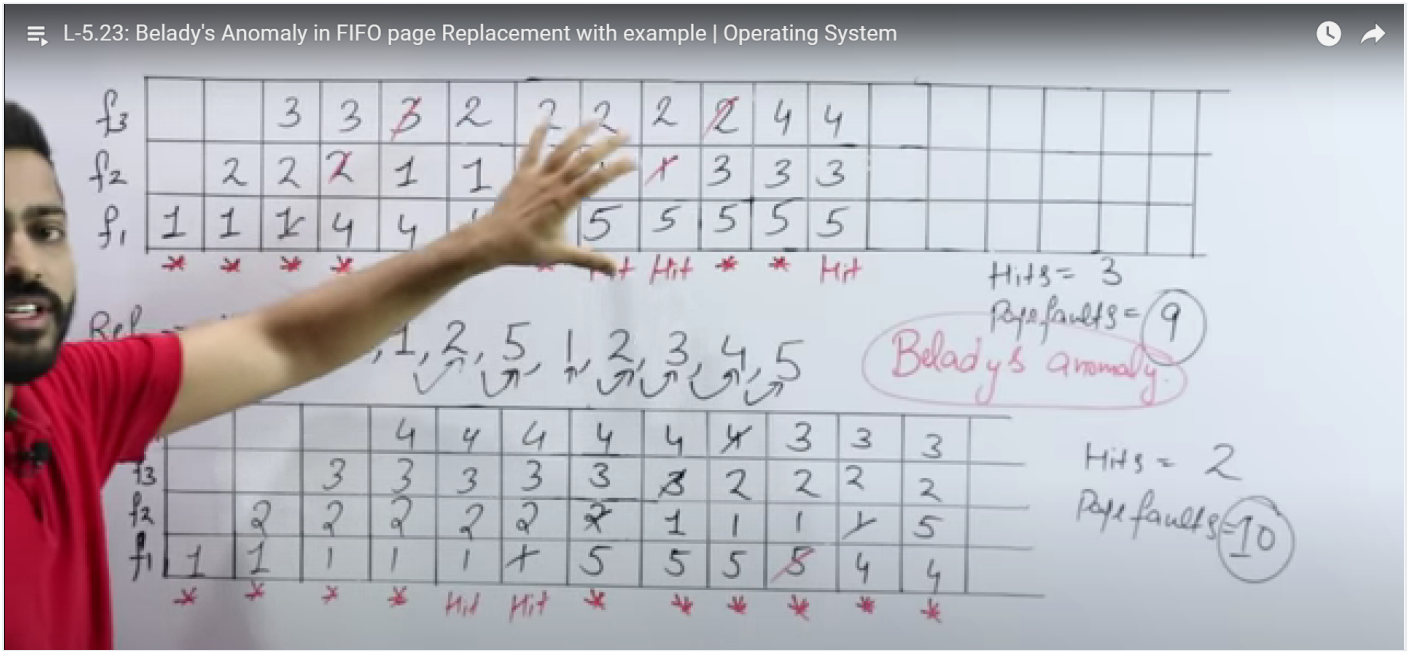
to do this there are three aalgorithms:

1. FIFO
2. optimal page replacement
3. least recently used (LRU)

|_ in FIFO page replacement algorithm it suffers with bellady's anomaly that is on increasing the frames the page fault should decrease as the page in memory increases but in reality page fault increases and page hit decreases. _|

(example done in white notebook.)

1.



2. optimal page replacement.

→ in this algo we replace the page which is not used for the longest duration of time.

→ jis page ko main memory mai lana hai vha se dekho right mai

⇒ konsa page jo frame mai hai aur uski demand sabse last mai hai.

⇒ agar do page ese hai jinki demand nhi hai to dono mai se kisi ko bhi replace krdo.

3. least recently used (LRU)- replace the least recently used page in the past.

⇒ jis page ko main memory mai lena hai usse page se left mai dekho.

⇒ konse page ki demand sabse pehle hui thi.

⇒usse replace krdo.

⇒slower than FIFO but better with page fault.

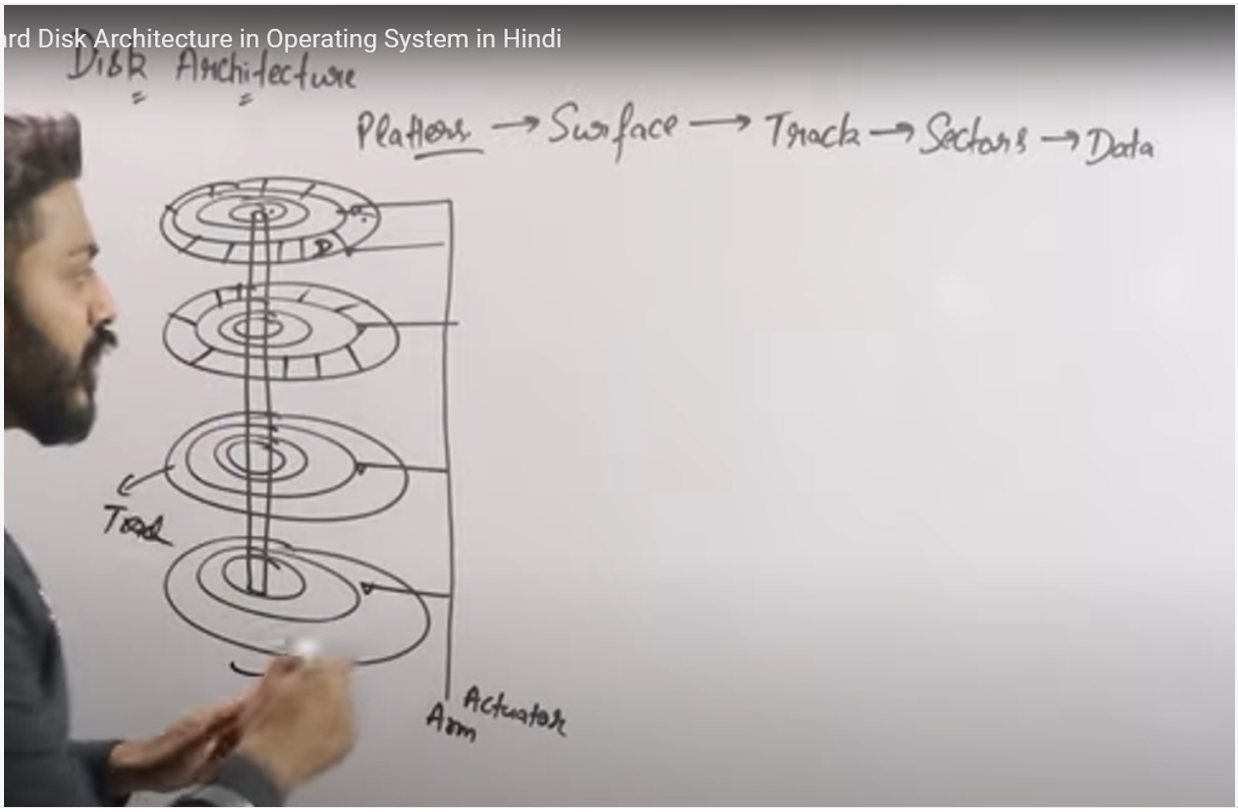
4. Most recenly used (MRU)- replace the most recent used page in the past.

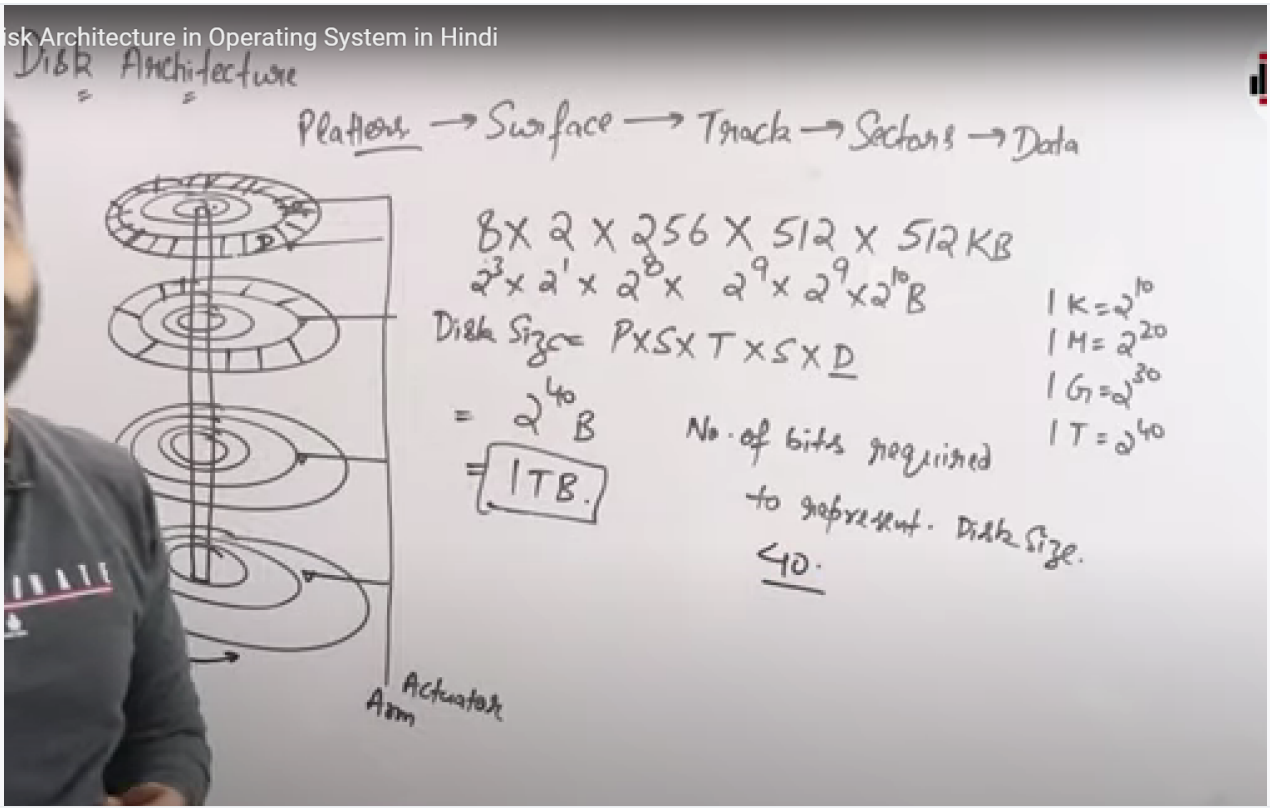
→jis page ko memory mai lana hai usse just left side wale page ko lelo.

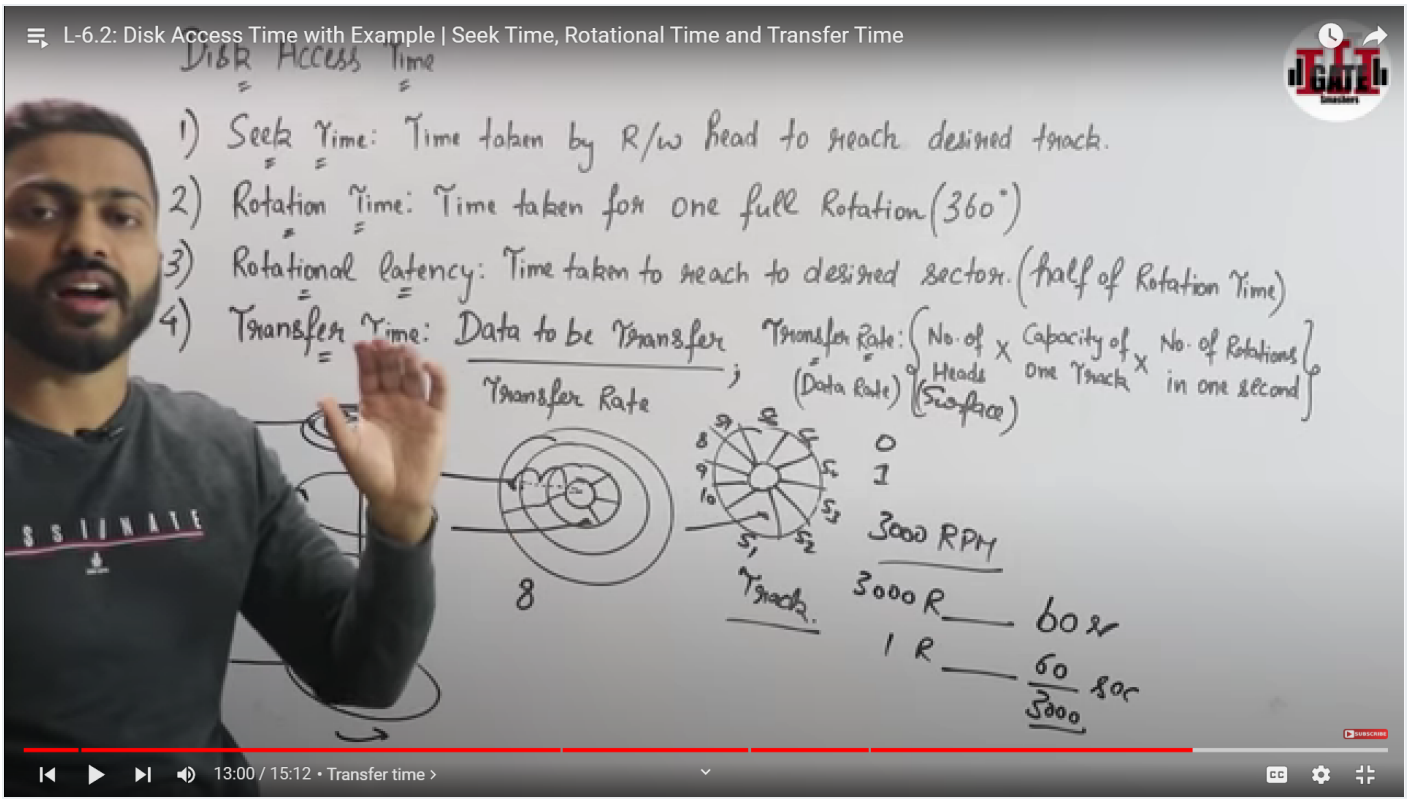
→ agar vo frames mai hai.

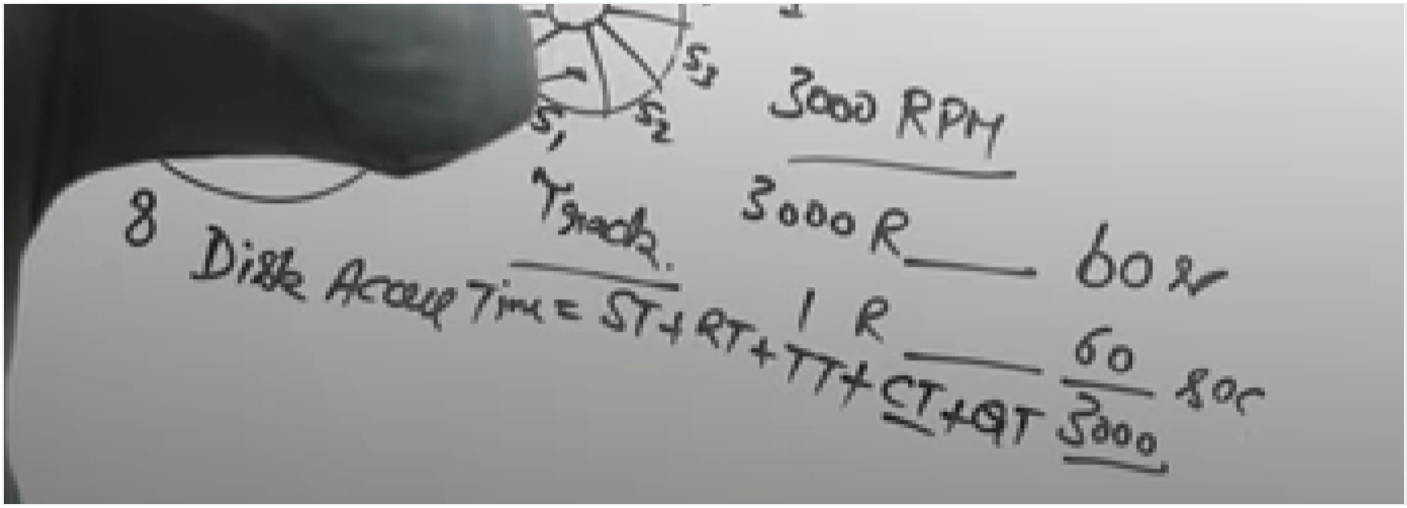
Disk management

Disk architecture :

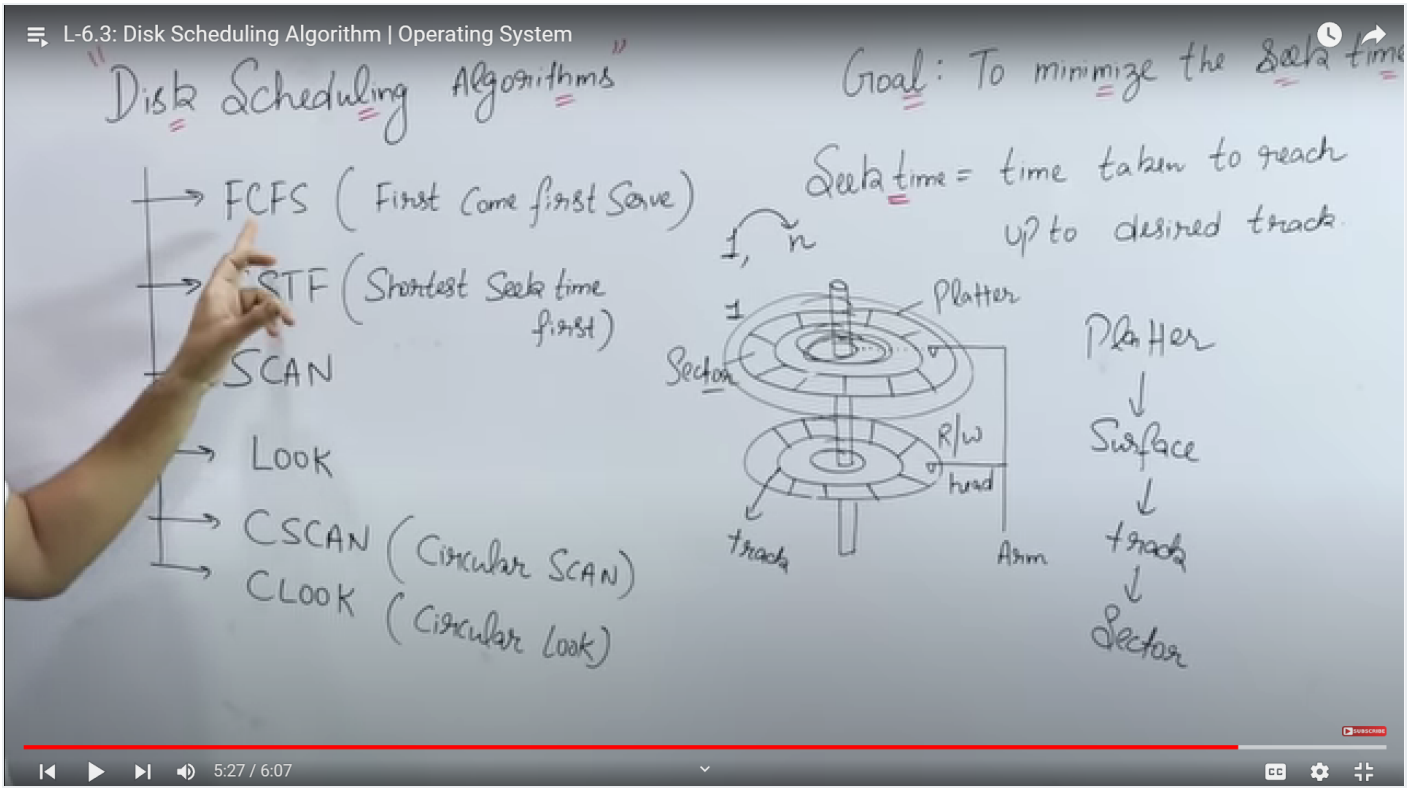






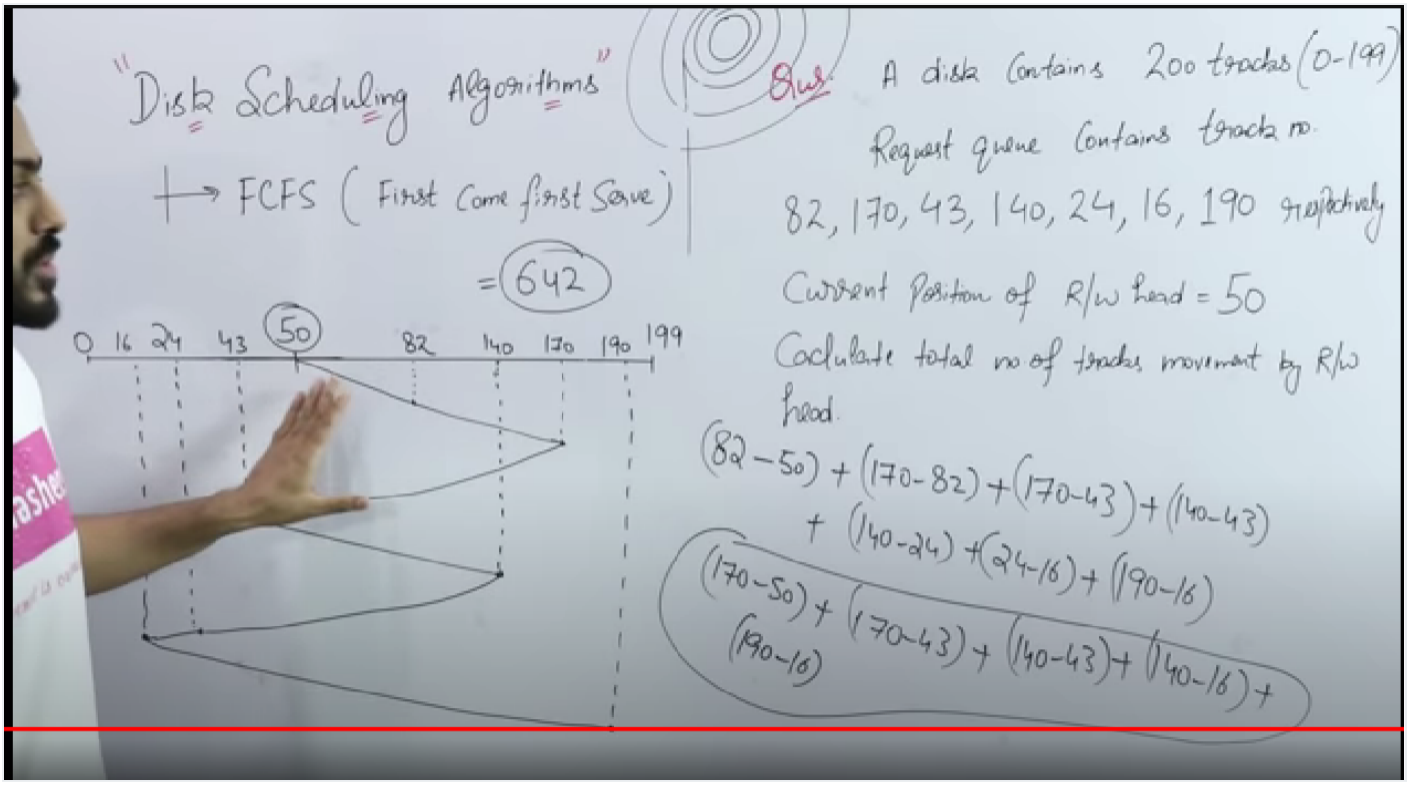


Disk Scheduling Algorithm:



FCFS

no starvation in this process



disadvantage:

seek time value is high.

performance is low.

SSTF

SHORTEST SEEK TIME FIRST IN

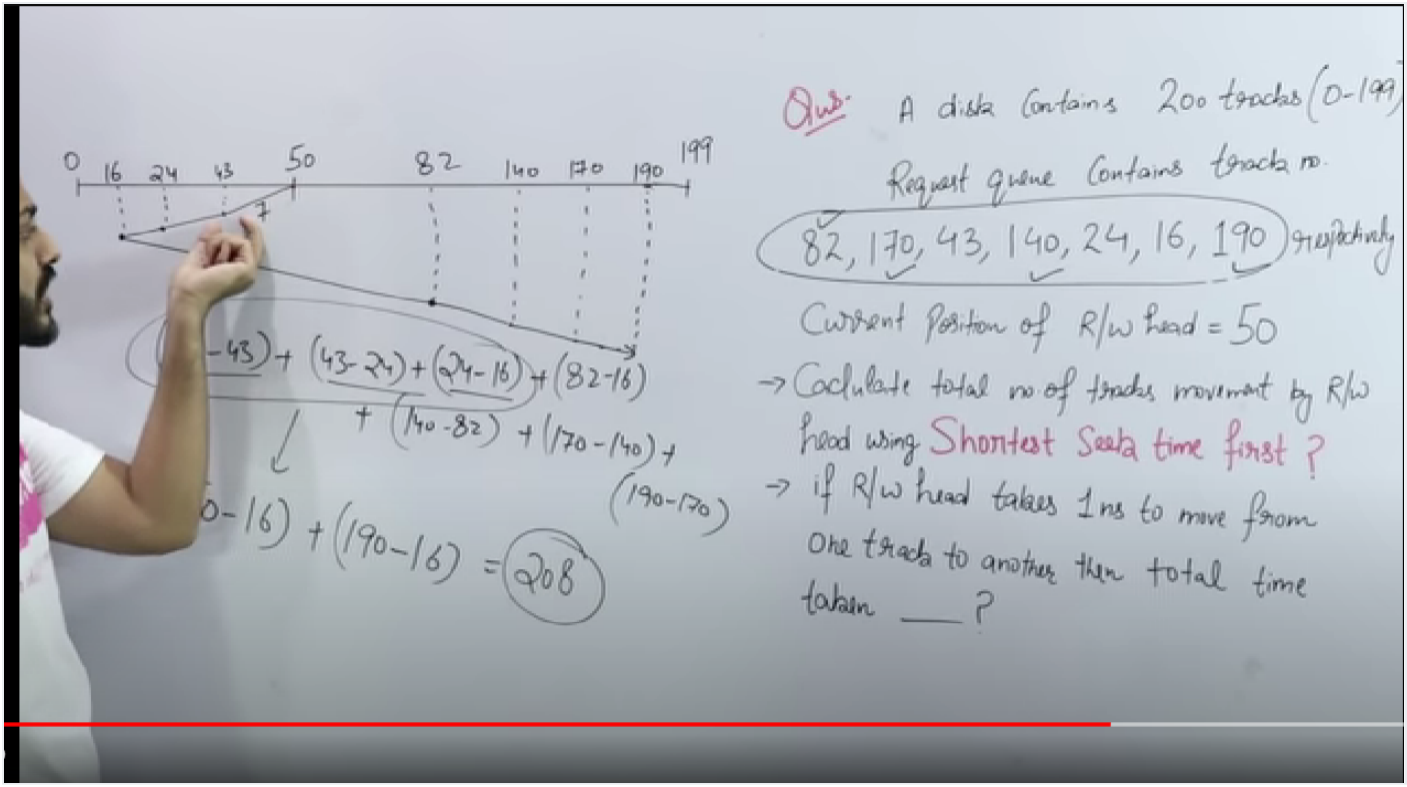
this algo will look the data and take the shortest seek time first not the smallest.

which means it will take the closest seek time from its actual location.

advantage:

tries to give a optimal result.

average response time is less.



Disadvantage:

Starvation.

overhead is generated.

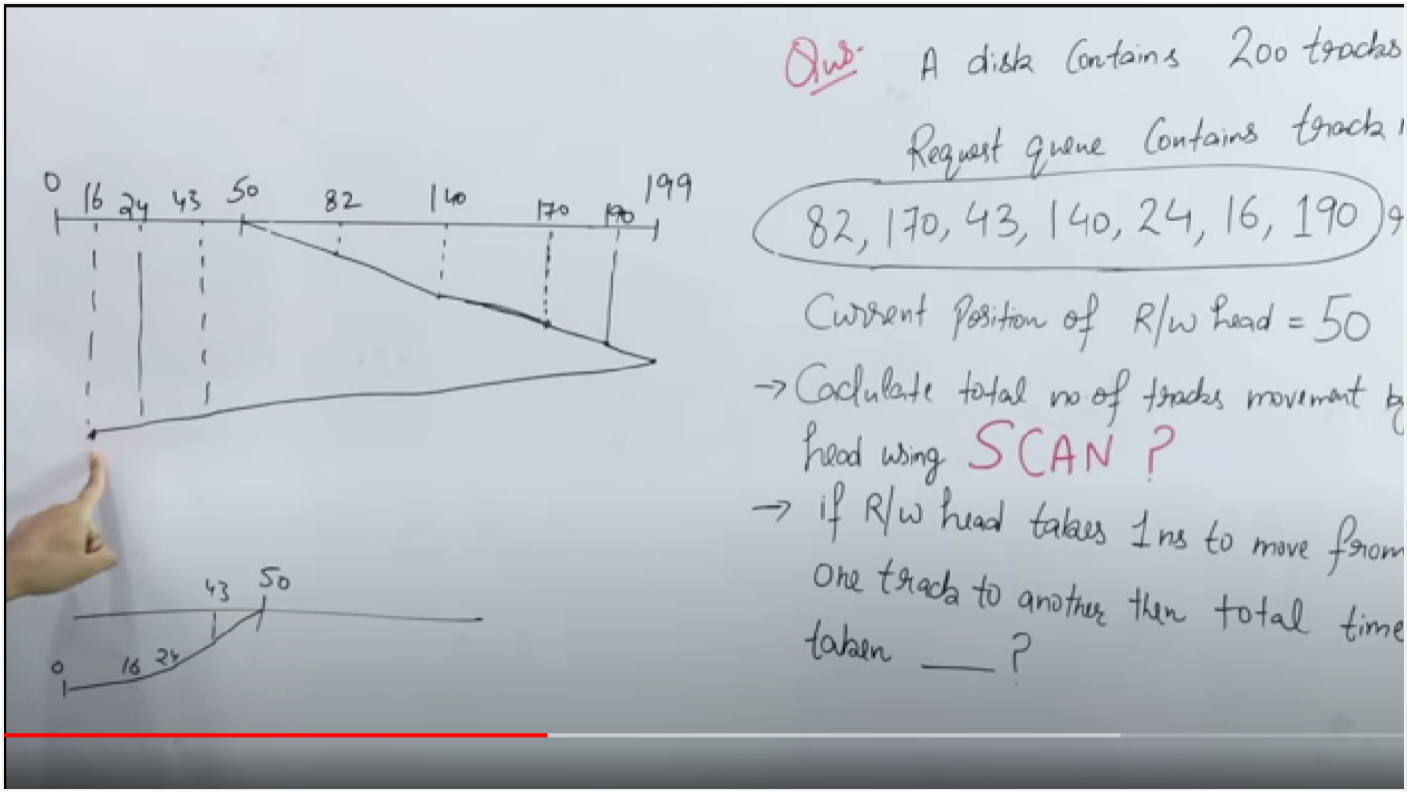
SCAN

elevator algorithm.

it will take a direction and move till the last track of the disk and service all the requests.

after covering a direction it will change the direction and service all the requests in that direction.

after going in another direction it will not go to the last track it will only go to the last requested track. (not like the previous direction)



advantage:

it is prepared for the first direction, if any further request comes dynamically it will serve it.

disadvantage:

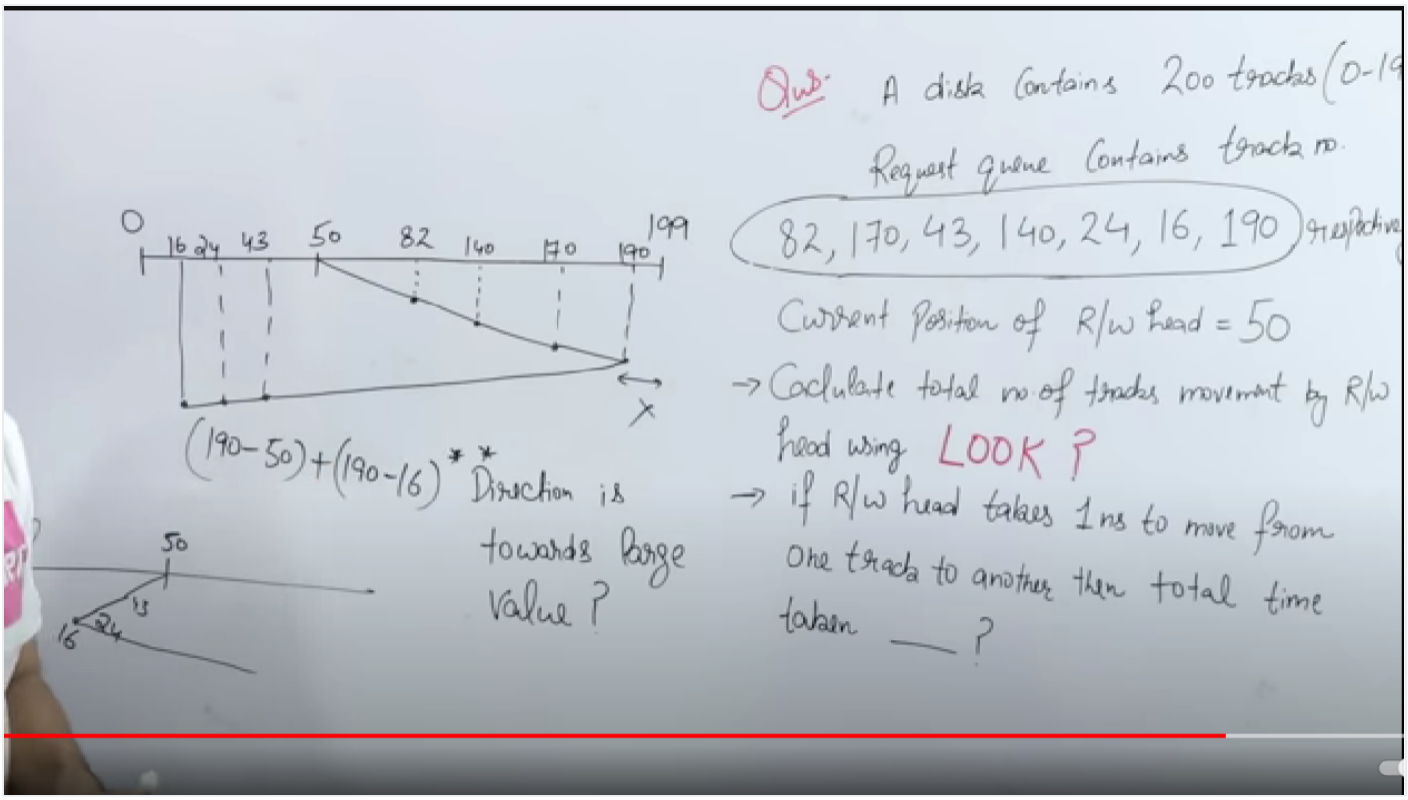
after changing the direction if the request comes for opposite direction, so it have to wait till the r/w head covers the entire requests of the direction.

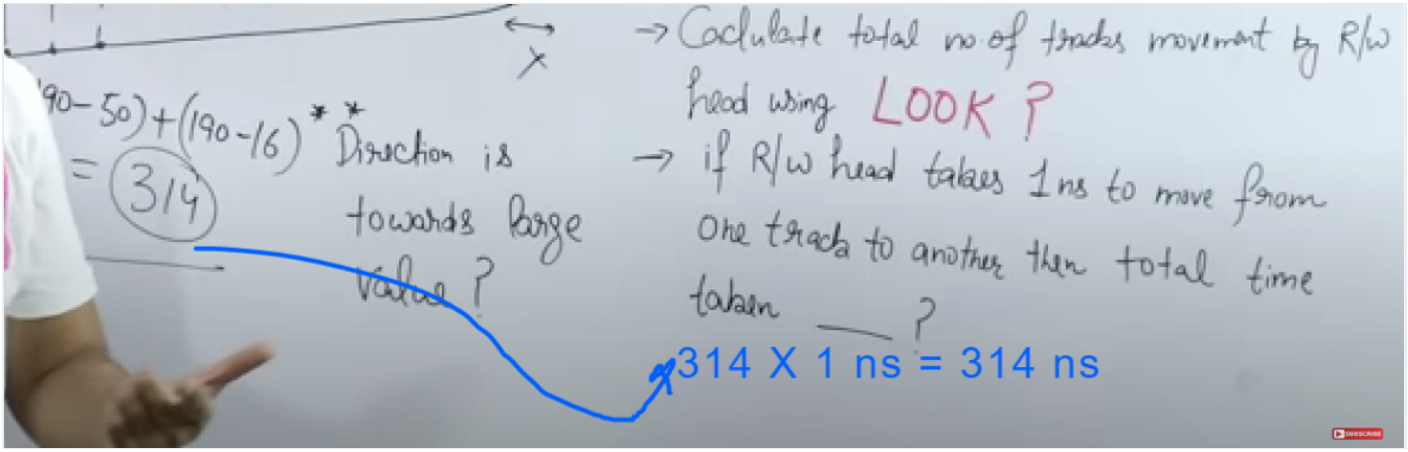
LOOK

direction has to be known.

after taking the direction service all the requests in those directions and then change the direction and service all the reuests in that direction.

we don't have to till last in first direction alike SCAN.





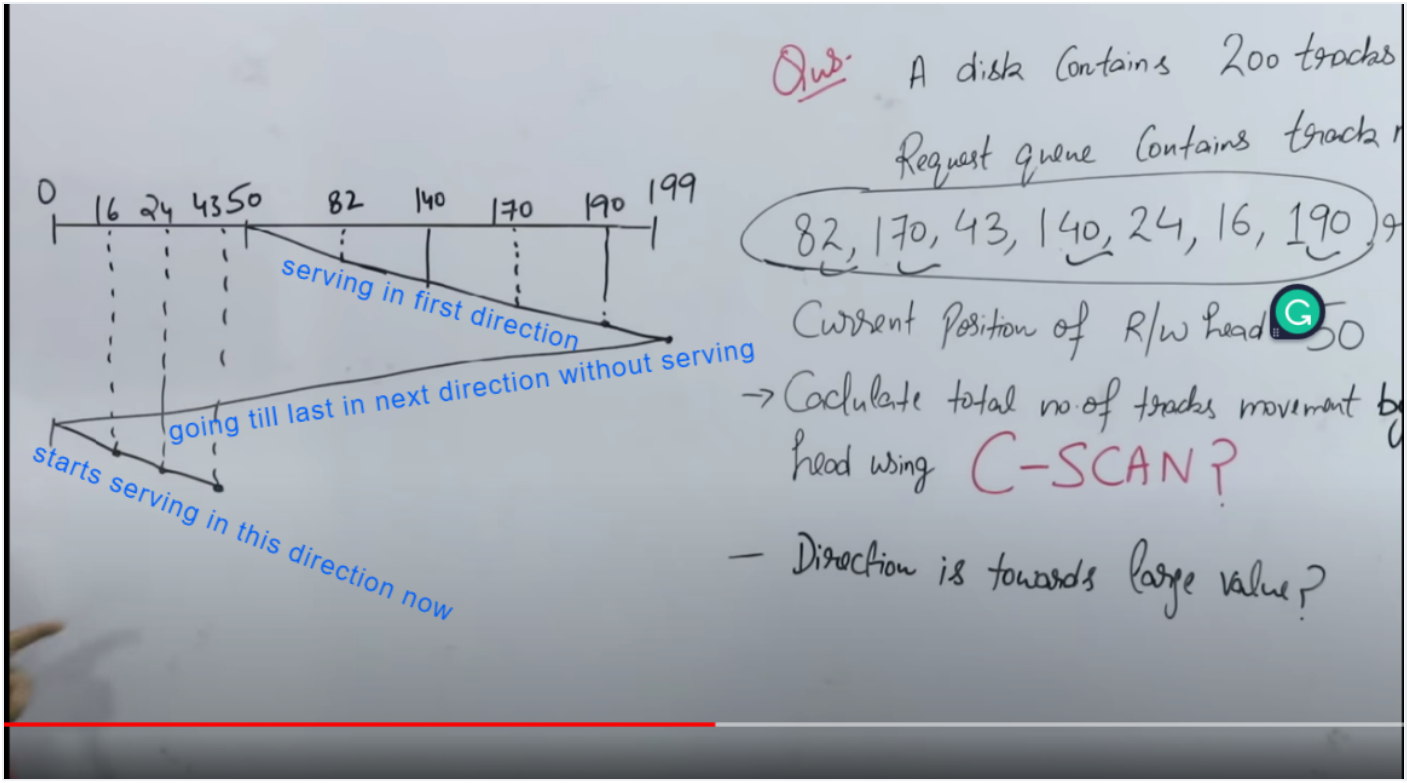
C-SCAN

circular scan

start direction is required.

it will take a direction and service all the requests in that direction and r/w head will go till the last of that direction and after reaching over there, it will change the direction and move till the end of that direction without serving any request.

after coming to the end it will start over and service all the remaining requests.



C-LOOK

circular look

direction is required.

it will take the direction and move in that direction and service all the requests in that direction and move till the last request (not till the last track alike C-look).

and then it will change the direction and move to the position of the last requested track without serving any requests.

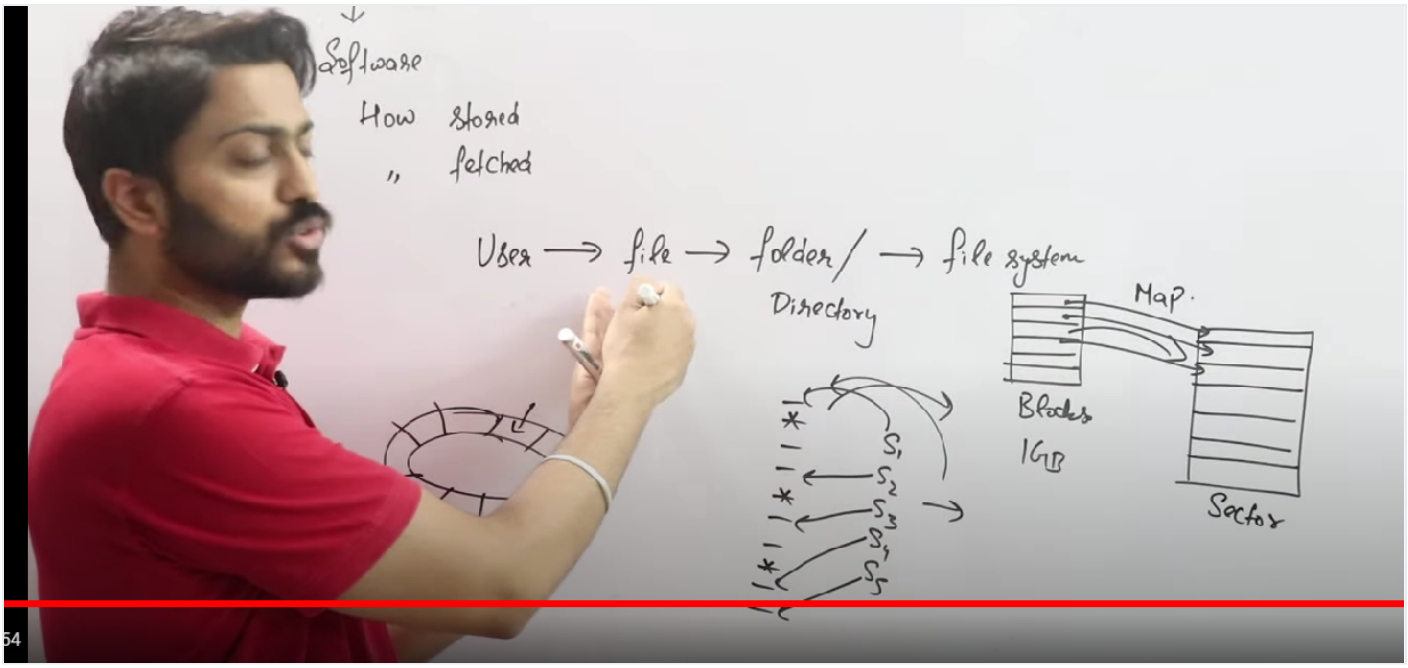
then it will again start serving the request in that direction

File System

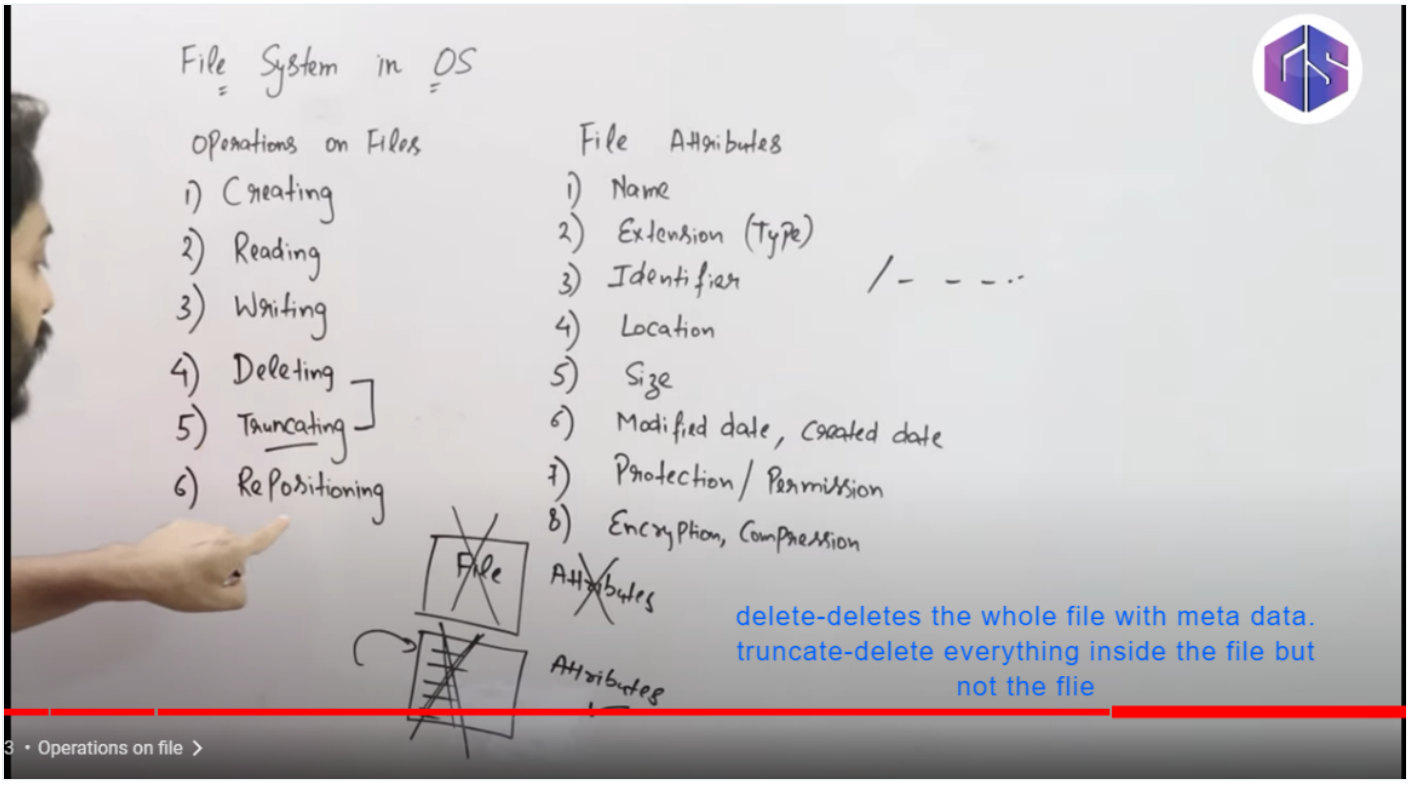
a file system is a software that manages the files and performs the operations to store and fetch the data.

in file system file is divided into blocks and then block are mapped into disc sectors.

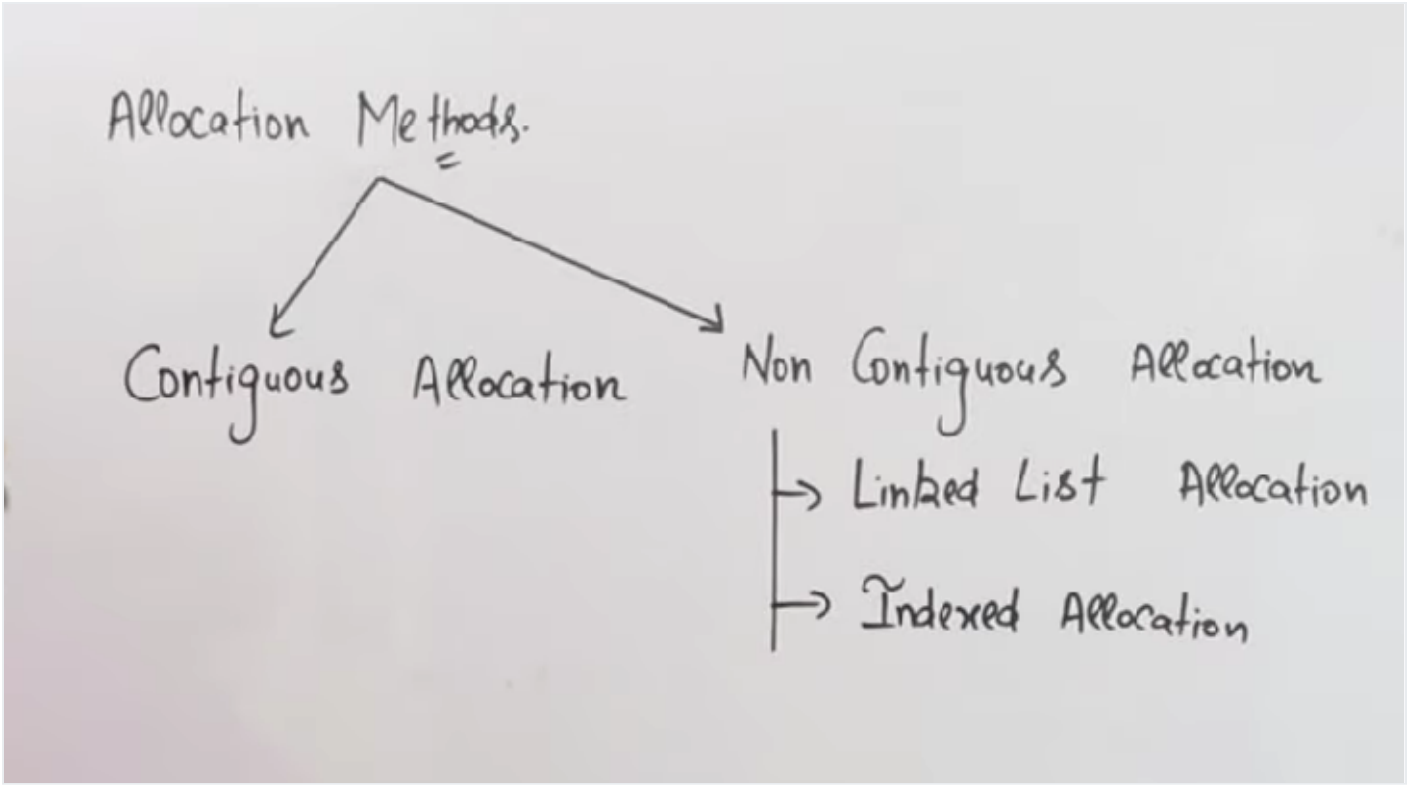
the file system is used to store this mapping.



file is a collection of data/information



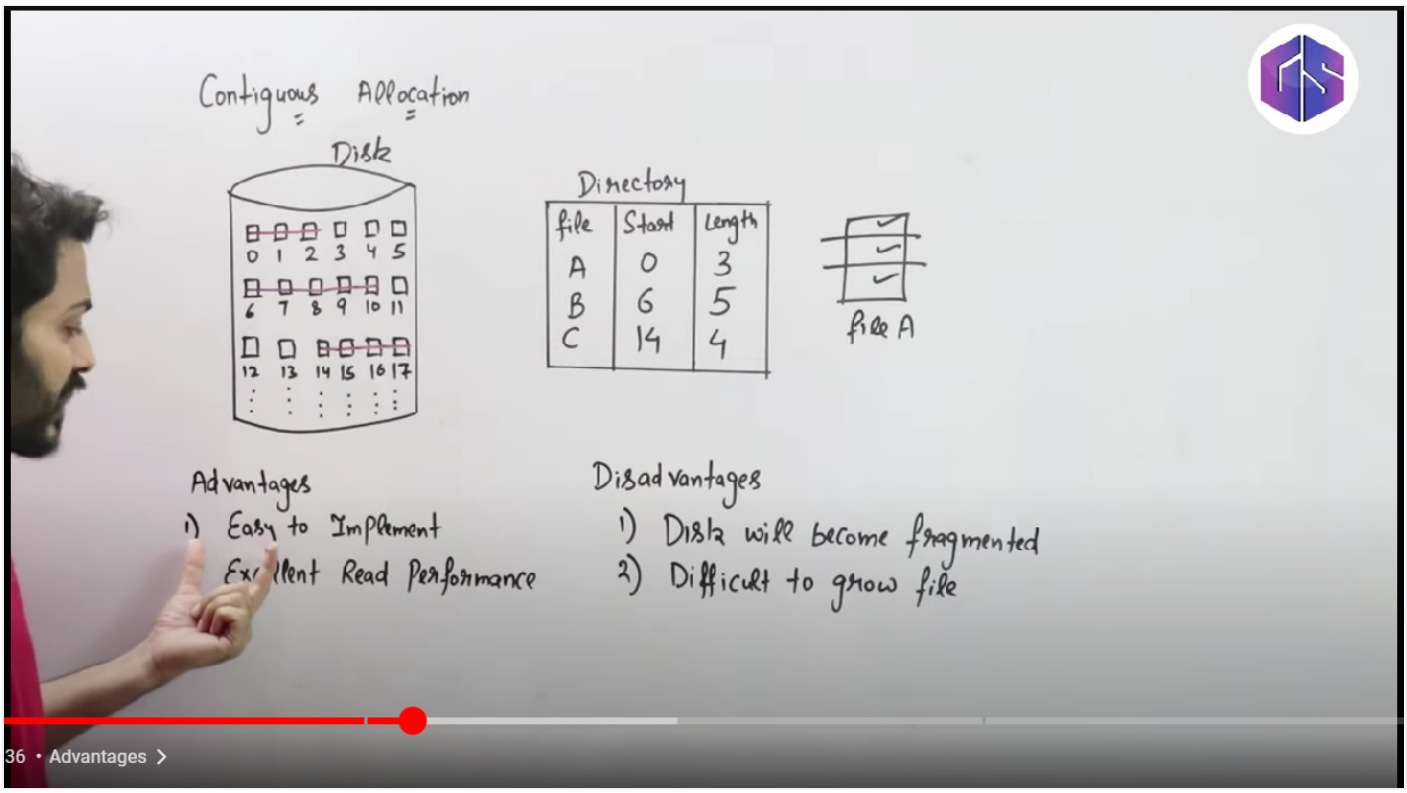
file allocation method:



purpose of file allocation:

1. efficient disc utilization.
2. access the data faster.

directory : contains all the information regarding the files.

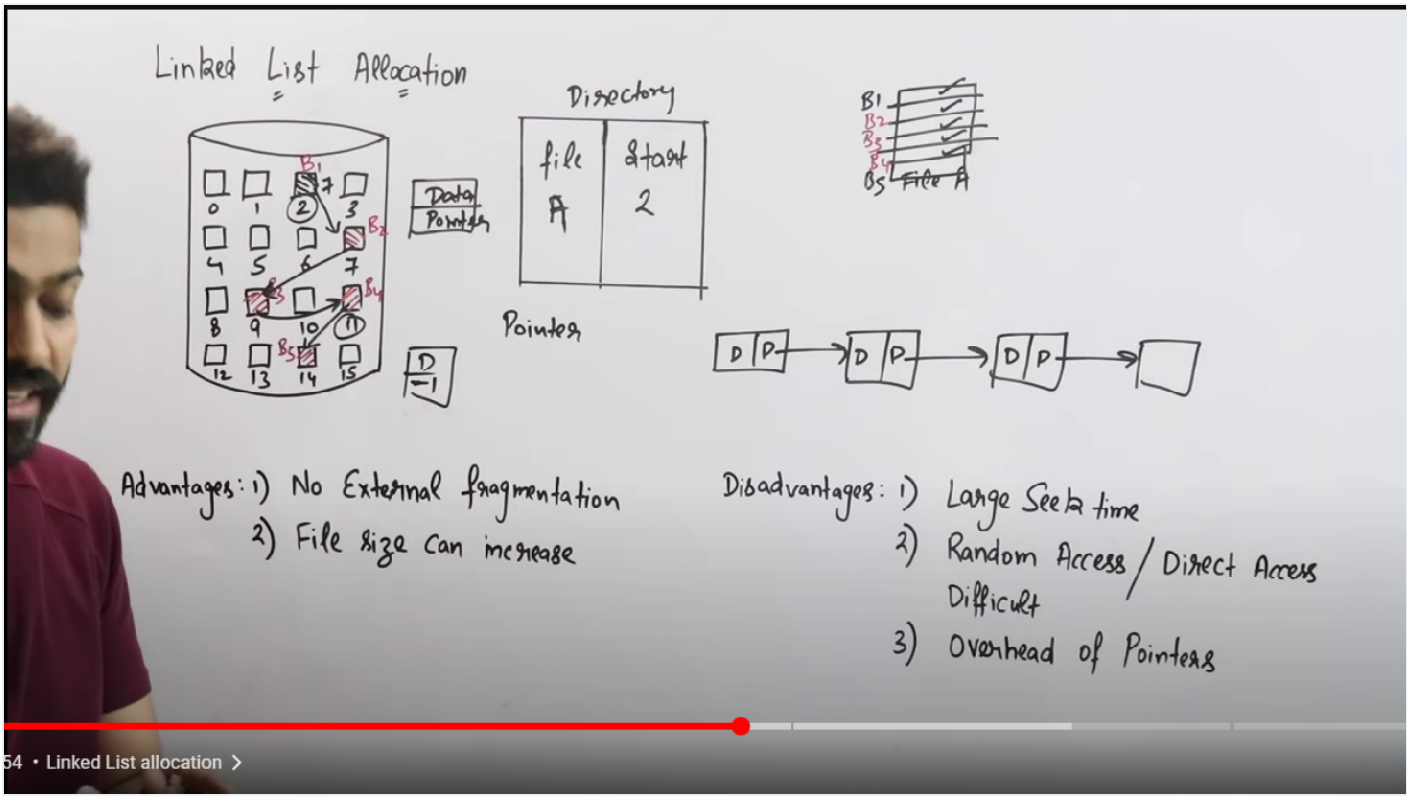


in continous file allocation the bock of file will be stored in disc in a continuous manner.

like given the directory it will start from address 0 and occupy till 3 sectors.

non-continous file allocation

linked list allocation:



in this allocation file blocks are stored in random manner.

the directory contains the starting address of the first the block.

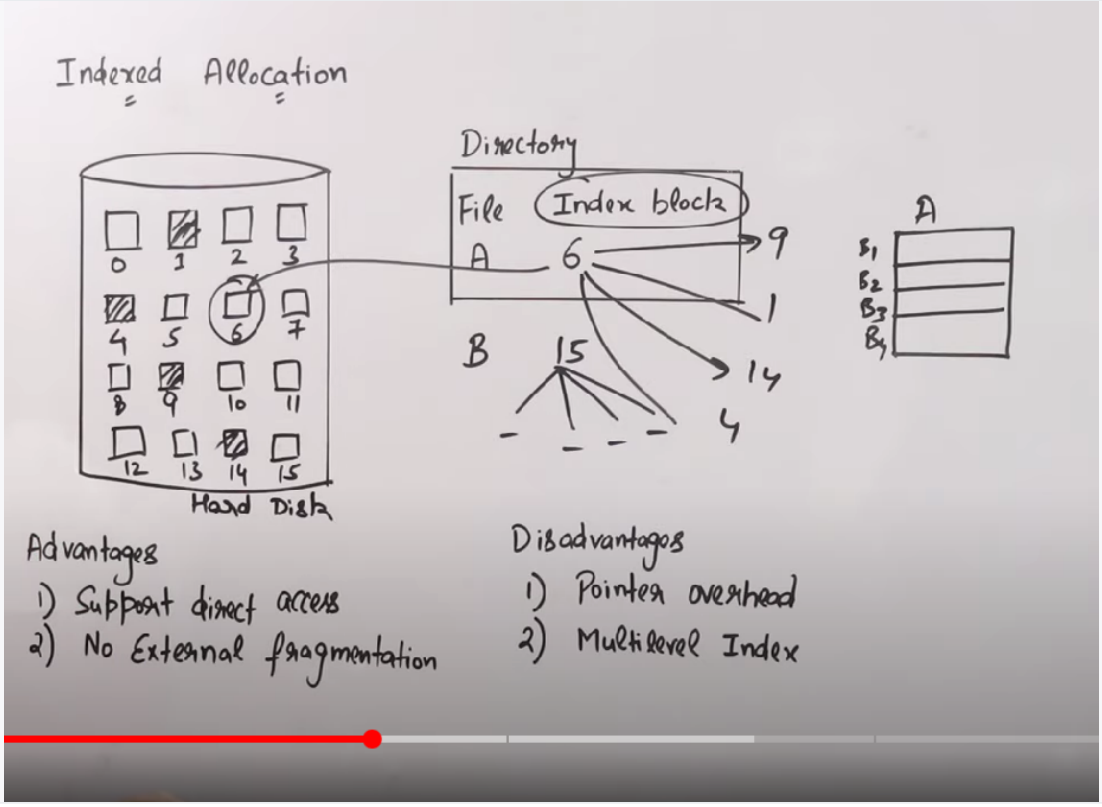
each block has two parts one is the data other is a pointer, pointing to the next block of the file.

indexed file allocation:

every file has its each index block.

directory contains the sector address in disc in which the index block is stored.

in the index block, there will be informed on all the blocks, and where they are stored.


Indexed Allocation
Directory
File
Index block
A
6
9
1
14
4
B
15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Hard Disk
A
Advantages
1) Support direct access
2) No External fragmentation
Disadvantages
1) Pointer overhead
2) Multilevel Index