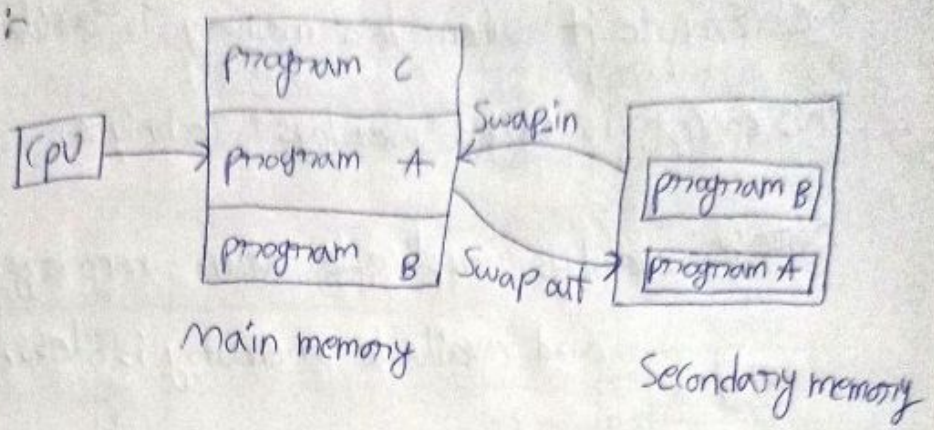
# UNIT-III

## Memory management

**Swapping:-**




when a program want to be Executed, it must loaded to main memory, because CPU can access only main memory. If no place in main memory to Store a program, then unwanted program swapped out to 2ndary (secondary) memory & program to be Executed is swapped in from secondary memory. This concept is called Swapping.

Swap in:- program from secondary memory to main memory.

Swapout:- program from main memory to secondary memory.

**Overlays:-**

To Execute a Job, it must load to main memory. If Sufficient memory is there, then without any difficulty, it can be loaded & Executed.

If Job requires more memory then, If available memory is less, Job can't done and we Execute it when less memory is available. This concept is called "overlays" Technique.

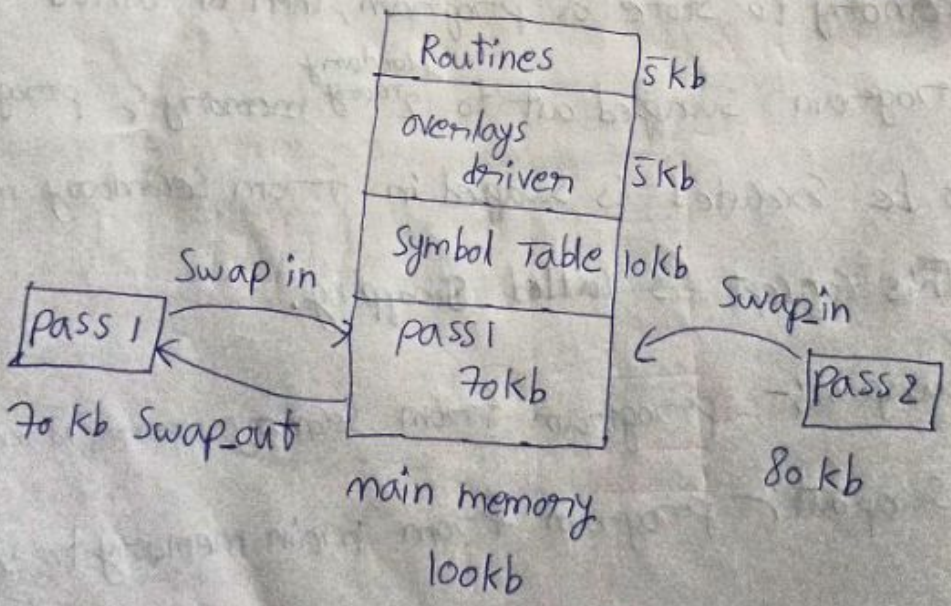
Note:- overlays Applicable when memory = more and available memory is less.

Ex:-

2 pass Assembler.

pass 1 : Symbol table construction

pass 2 : object code conversion using symbol table



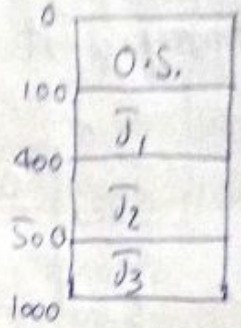

Here, pass1 and pass2 need not to be at a time in memory. So, pass1 is Swapped in & get Executed. Then pass1 is swapped out & pass 2 is Swapped in & Executed.

## Contiguous memory allocation management:-

### a) Basic concept:-

Different Jobs are allocated in the memory continuously. Each position may be of different size.

| Job | Size(in kb) |
|---|---|
| $J_1$ | 300 |
| $J_2$ | 100 |
| $J_3$ | 500 |

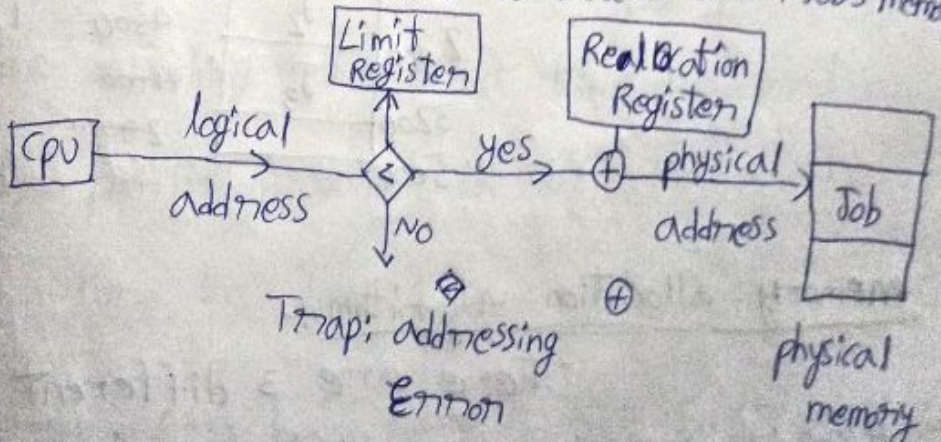



### b) memory protection/Hardware Support:-

By memory protection concept, in 2 ways the memory can be protected.

a) No Job is allocated to access OS memory.

b) No Job is allowed to access other Job's memory.



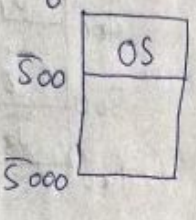

**Limit Register:-** It contains the size of the Job which is being Executed by CPU.

## Relocation Register (or) Base Register:-

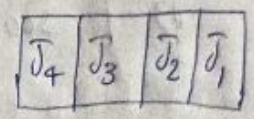
It contains the starting address of the Job being executed by CPU.

## c) memory Allocation:-

Initially, OS is available in the memory & it always occupies lowest part of the memory.

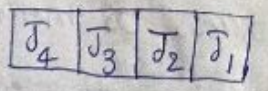


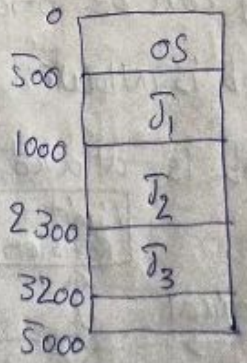

Job queue




After memory allocation, the memory is divided into multiple partitions as

Job queue




Left over memory

4500 1800

~~4000~~

~~2700~~

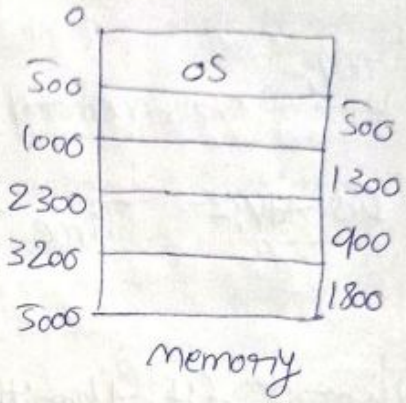
## memory allocation Algorithms:-

There are 3 different Algorithms, using them different jobs can be allocated to different partitions.

They are;-

a) First-fit Algorithm
b) Best-fit Algorithm
c) worst-fit Algorithm

| Job | Size |
|---|---|
| $J_1$ | 500 |
| $J_2$ | 1300 |
| $J_3$ | 900 |
| $J_4$ | 2100 |
| $J_5$ | 1000 |
| $J_6$ | 100 |
| $J_7$ | 900 |
| $J_8$ | 1800 |




a) First-fit Algorithm;-

In the Empty partition list, while searching the first time, we find an Empty partition which is more than Sufficient to store the Job.

Adv;-
— Takes Less time to find one Empty partition.

Dis Adv;-
memory wastage

b) Best-fit Algorithm:-

The Entire Empty partition list Should be searched. So that the Exact/Equal partition is identified & then the Job is allocated.

Adv:-
No memory wastage

Dis Adv:- Time taking process.

c) worst-fit Algorithm:-

The Entire Empty partition list should be searched & a biggest partition should be identified.

Dis Adv:-

Lot of memory is wasted.

Time taking process.

Note:- partition is also called "hole".

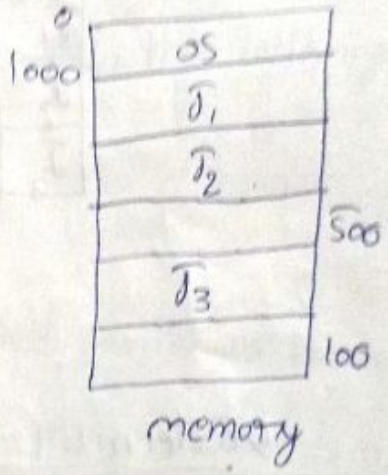
The

## Fragmentation:-

The memory is divided into more numbers of partitions. This concept is called "fragmentation". The problem with fragmentation is Even though the required memory is available, it cannot be allocated to a job, because the memory is scattered in different places.

Ex:-

| Job | Size |
|---|---|
| $J_4$ | 600 |




memory = 600kb

$J_4$ requires 600kb. But, Even though 600Kb is available

Job $J_4$ cannot be accomodated in the memory, because the memory is available in different places (not continuously).

## Types of Fragmentation:-

There are 2 types of fragmentations. They are:-

a) Internal fragmentation

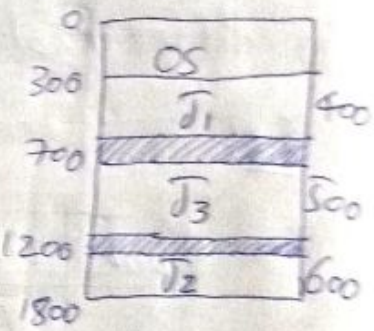
b) External fragmentation

a) Internal fragmentation:-

It is a fragmenta... where a little part of memory is wasted within a partition.

Ex:-

| Job | Size |
|---|---|
| $J_1$ | 100 |
| $J_2$ | 600 |
| $J_3$ | 400 |



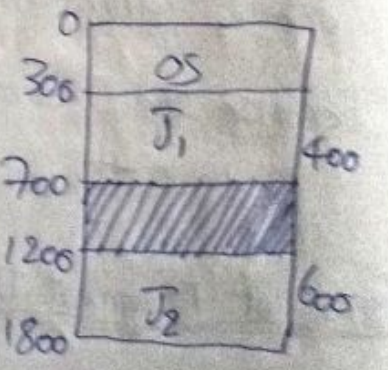

memory

b) External fragmentation:-

It is a fragmenta... where there is overall wastage of memory.

Ex:-

| Job | Size |
|---|---|
| $J_1$ | 400 |
| $J_2$ | 600 |
| $J_3$ | 700 |




memory

## Solutions for fragmentation:-

There are 2 solutions for fragmentation. They are:-

a) Compaction Technique
b) Paging Technique

### a) Compaction Technique:-

Compaction is a solution for fragmentation problem. According to compaction Technique to use unused memory, the following procedure is followed.

**Step-1:-**

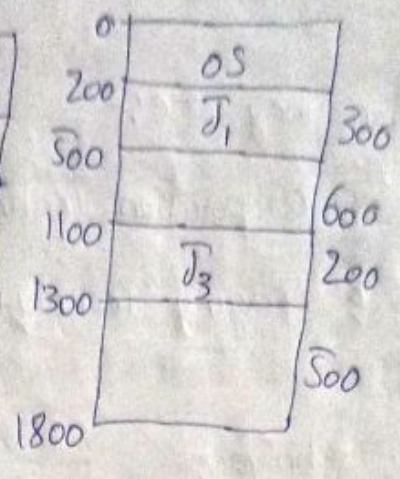
Move all the allocated partitions towards the lower part of the memory (towards os).

**Step-2:-**

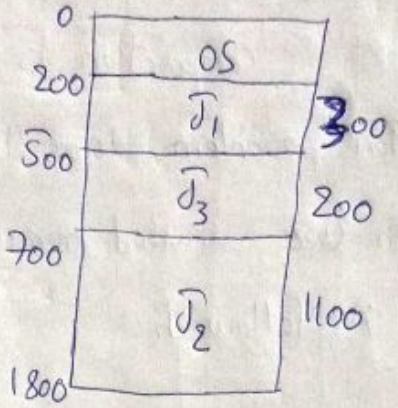
Move all the unallocated partitions in the opposite direction, so, that all the unused memory will be combined together, so that even a big Job can be accomodated.

Ex:-

| Job | Size |
|---|---|
| $J_1$ | 300 |
| $J_2$ | 1100 |
| $J_3$ | 200 |




After Compaction ⇒




⇒ Paging memory management Technique:-

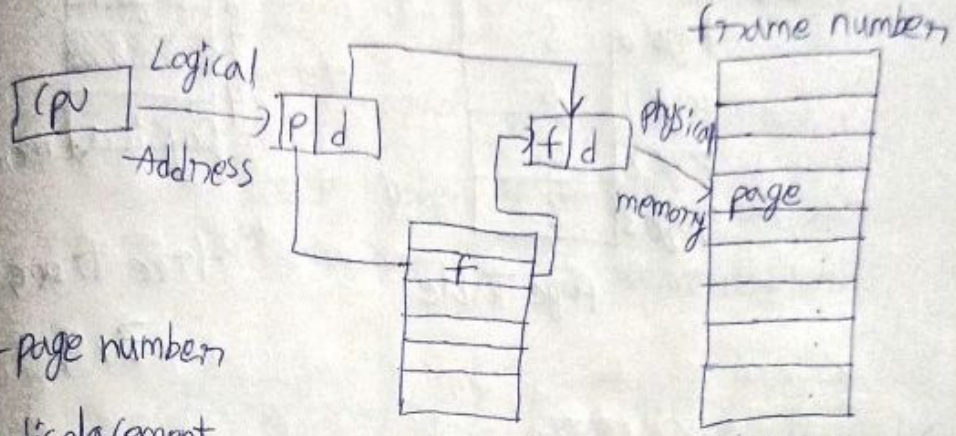
a) Basic concept:- (paging is a memory management technique used to solve fragmentation problem) Job is to be Executed is called Logical address which is divided into number of partitions, in Equal size. Each partition is called "page".

memory, also called physical address, which is divided into Equal partitions. Each partition is called "frame".

Note:- The size of page = size of frame.

b) Hardware Support:- whenever a job is created,

A page table is created. The page table acts as a bridge between Job & physical memory. The page table contains the frame numbers of physical memory & pages of Job used as index of page table.

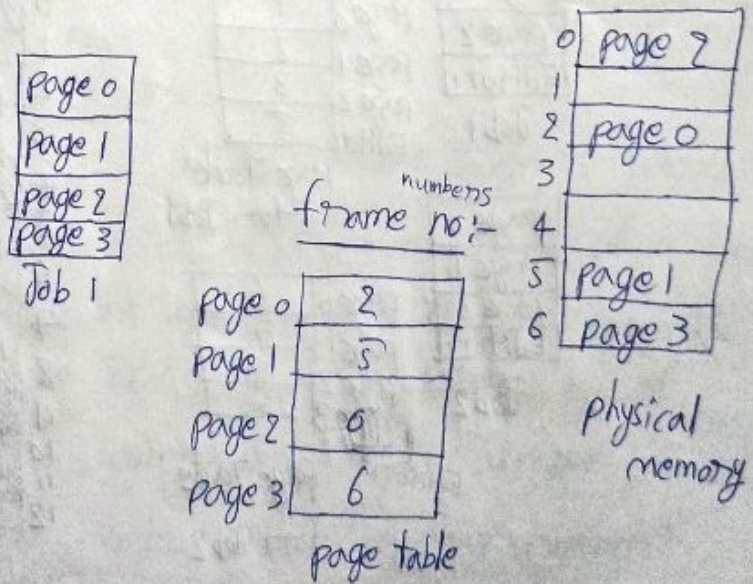



p-page number

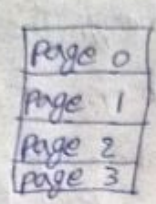
d-displacement

f-frame

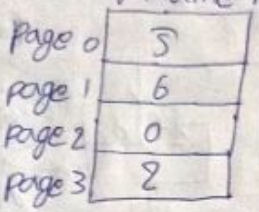
1) paging model:-



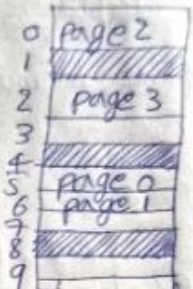
d) Implementation:-




Job



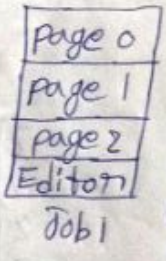

Page Table



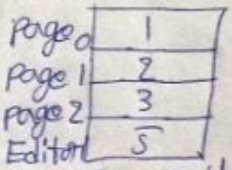

free frame list

3, 9

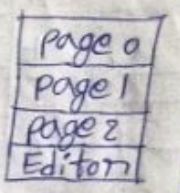
e) Sharing of pages:-



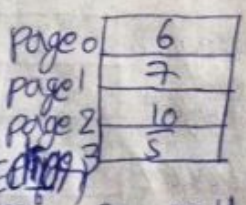

Job1




Page Table for Job1

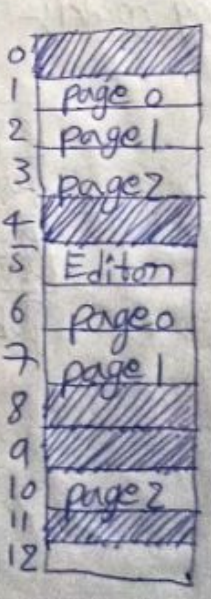



Job2




Editor

Page Table for Job2




free frame list

12

physical memory

when 2 (or) 3 jobs are having a common page, used by all the jobs can be stored in one (partition) particular frame & the same frame number is written in all page tables.

Advantages of paging:-

* Sharing of pages is possible.
* No external fragmentation.
* No internal fragmentation.

Note:-

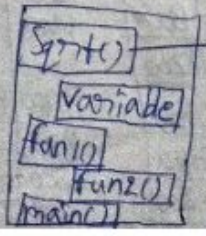
There may be a little bit of internal fragmentation but, it's negligible.

⇒ Segmentation:-

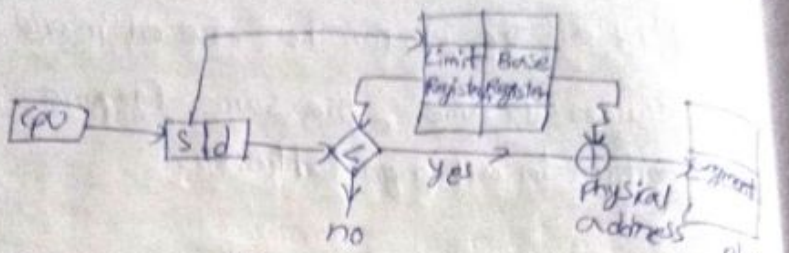
a) Basic concept:-

This technique supports user's view of memory i.e., the different parts of a Job are stored in the memory in different segments. The Job to be Executed is, divided into no. of segments. Each segment is assigned a unique Identification number called "segment number."

Ex:-



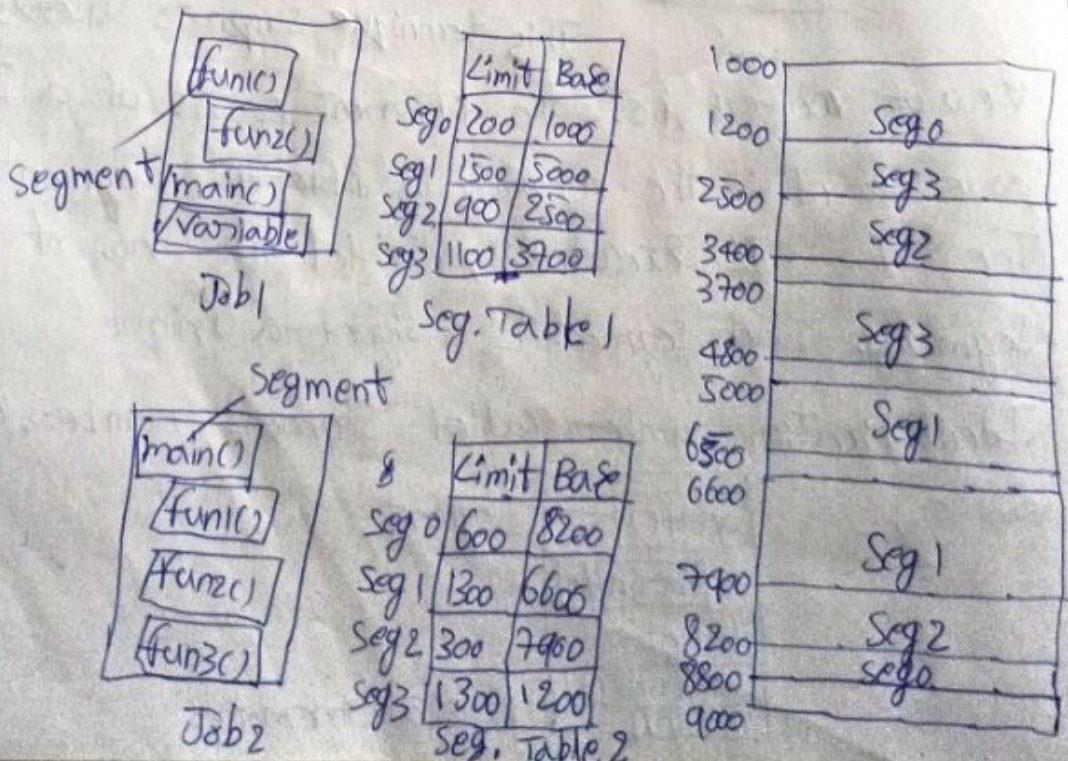
b) Hardware Support:-




Segmentation requires a Segment table which acts as a communication bridge between (Limit) Segment & physical memory.

(By using) The segmentation table contains two registers,
a) Limit Register
b) Base Register

c) Implementation:-



## 2) Sharing of segments / pages:-


**Advantages:-**

a) Sharing of segments is possible.

b) It supports user's view of memory.

**Disadvantages:-**

a) There is a possibility of internal fragmentation.

b) There is a possibility of external fragmentation.

30/12/21

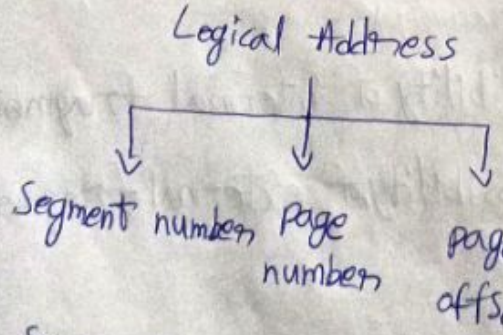
## ⇒ Segmentation with paging:-

### a) Basic concept:-

Segmentation paging is a scheme that implements the combination of Segmentation & paging. It is combined with paging to get best features.

main memory is divided into variable size segments which are further divided into fixed size pages.

1. Pages are Smaller than Segments.
2. Each Segment has a page table which means every program has multiple page tables.
3. The logical address is represented as, Segment number, page number, page offset.

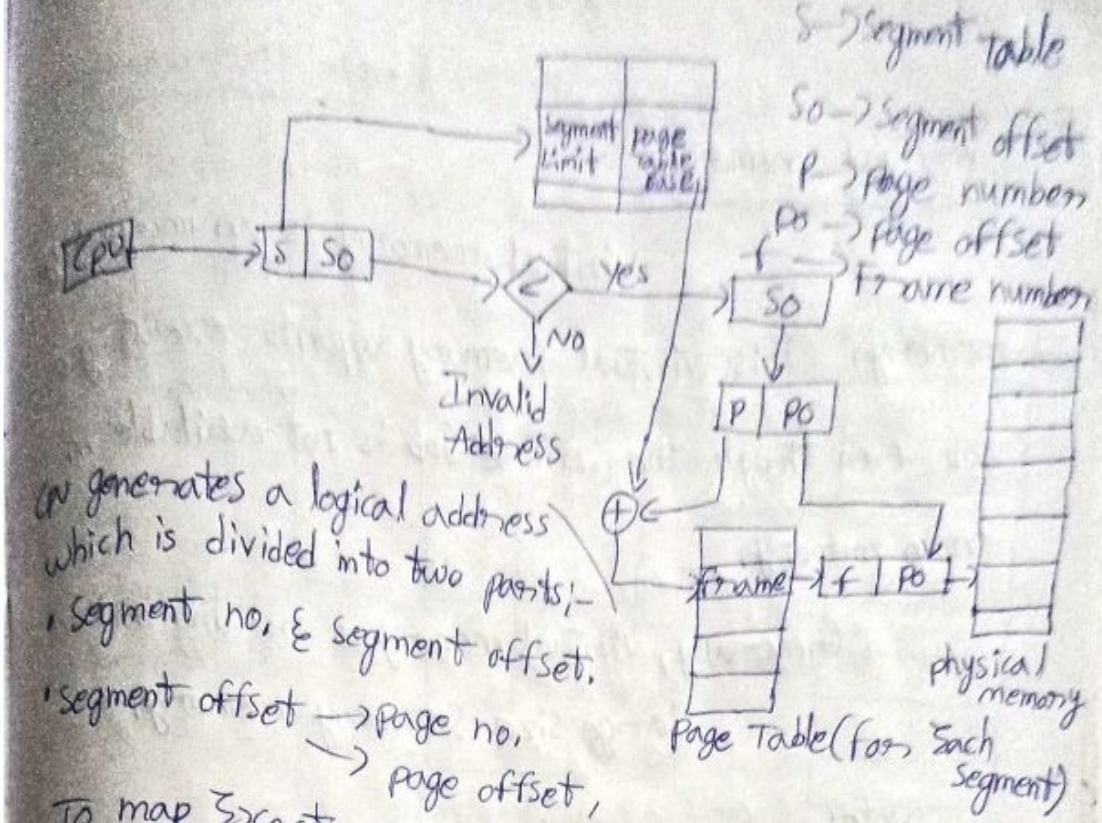



Segment number:- It points to the appropriate Segment number.

page number:- It points to the exact page within the Segment.

page offset :- It is used an offset within the page frame

## 4) Hardware Support:-




CPU generates a logical address which is divided into two parts:-

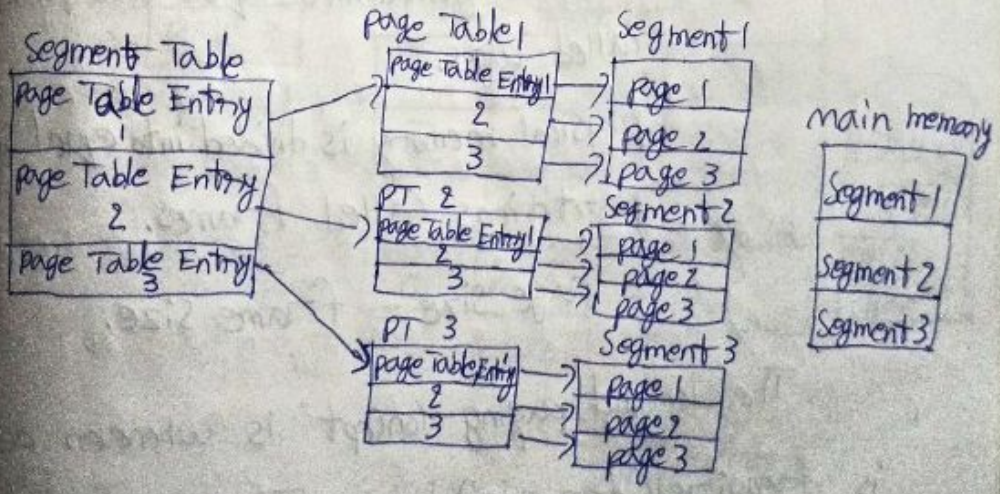
- Segment no. & Segment offset.
- Segment offset → Page no.
  → Page offset,

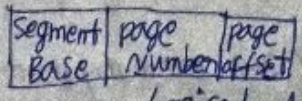
To map exact page no. in the page Table, the page no. is added into the page table base.

- The Actual frame number with the page offset is mapped to the main memory to get the desired word in the page of the certain segment of the process.

## 5) Implementation:-

PT - Page Table




Logical Address

=> Virtual memory:-

Virtual memory is an imaginary memory. This virtual memory supports executing a job even though the entire job is not available in main memory.

Generally, virtual memory is nothing but a large size Secondary memory.

Note:-

Demand paging is the memory management technique supported by virtual memory concept.

2) Demand paging:-

a) Basic concept:-

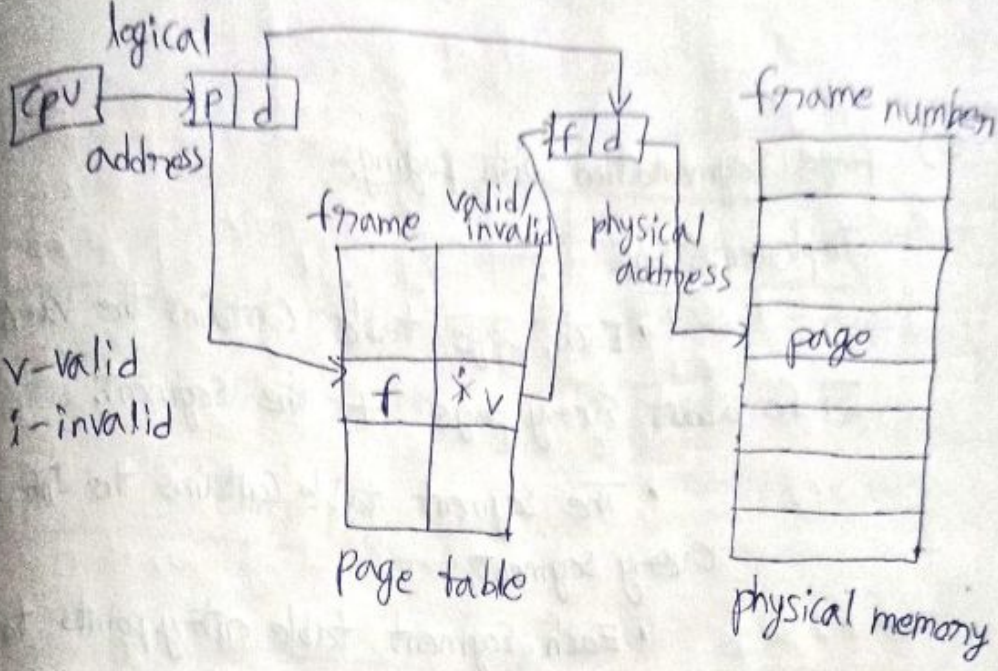
Job is divided into equal size partitions called pages.

physical memory is divided into equal size partitions called frames.

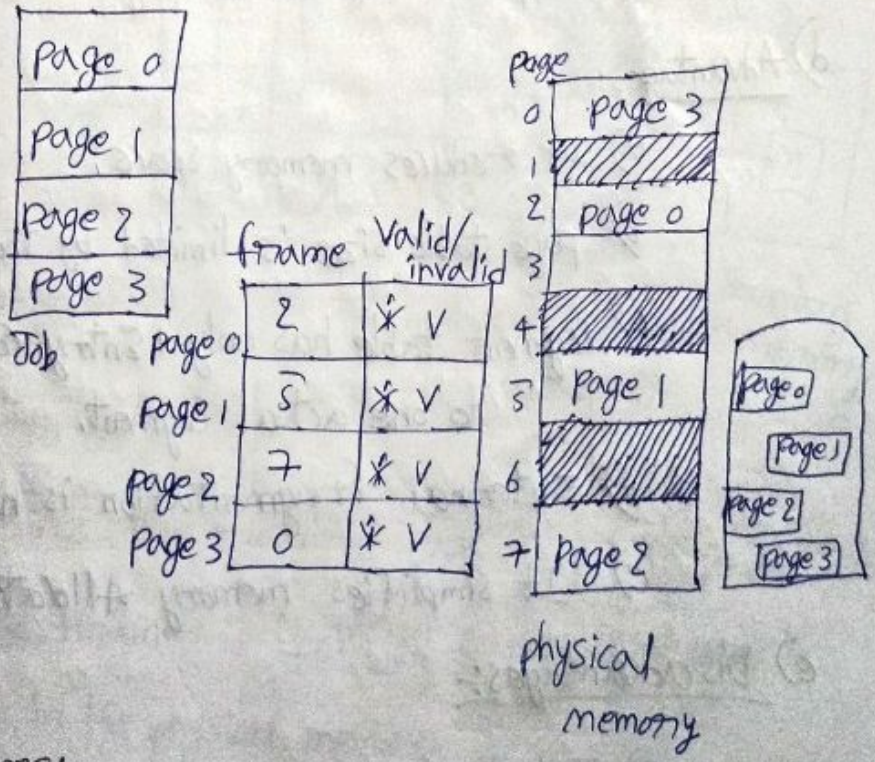
page size = frame size.

The demand paging concept is whenever a page is required (demanded), by CPU. Then only the page is loaded to main memory, until then all are in Secondary memory.

b) Hardware Support:-




c) Demand paging Implementation:-




d) Advantages:-

* Sharing of pages is possible.

* no internal fragmentation
* No External fragmentation
* Demand paging is powerful than paging.

=) Segmentation with paging:-

Implementation:-

- Each page table contains the various Info about Every page of the Segment.
- The Segment table contains the Info, about every segment.
- Each Segment table entry points to a page table entry & Every page table Entry is mapped to one of the page with a Segment.

d) Advantages:-

a) It reduces memory space.
b) page table size is limited by the Segment size.
c) Segment table has only 1 Entry corresponding to one actual Segment.
d) External fragmentation is not there.
e) It simplifies memory Allocation.

e) Disadvantages:-

a) Internal Fragmentation will be there.
b) The Complexity level will be much (bigger) higher

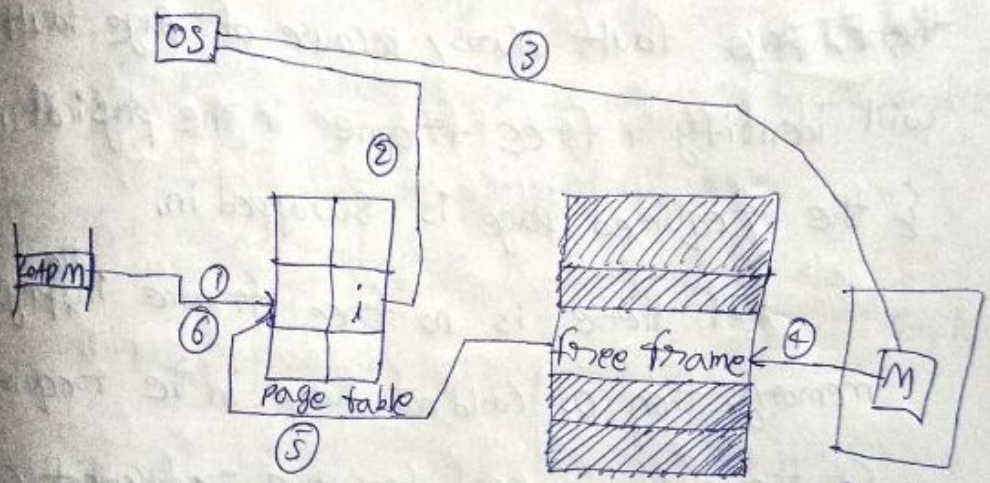
as compare to paging.

2) page tables need to be contiguously stored in the memory.

## Page fault:-

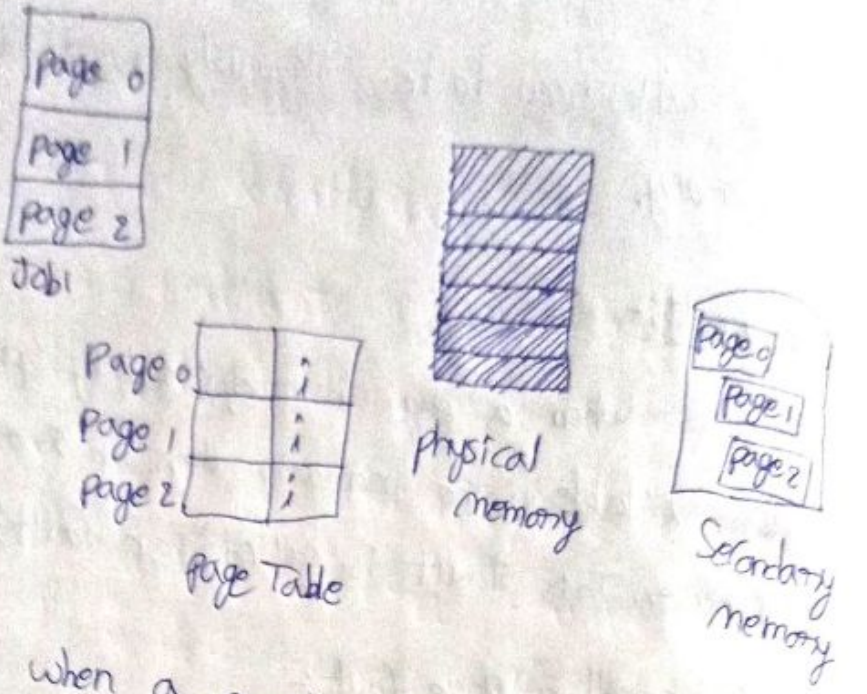
When a page is required by CPU, if it is not available in the memory, then an ~~Error~~ fault will occur. This fault is called "page fault".

## Steps in handling page fault:-




1) Reference
2) Trap to OS
3) OS finds the required page in Secondary memory.
4) A free frame is identified & required page is Loaded in the physical memory.
5) update the page table.
6) Restart the instructions.

=) need for page replacement method:-

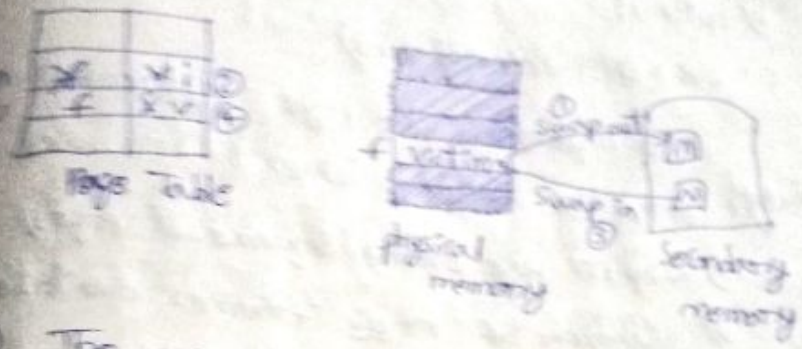



when a page is required by CPU, if it is not there, page fault occurs, Because of page fault, OS will identify 1 free frame in the physical memory & the required page is swapped in.

If there is no free frame in the physical memory, then OS could not swap in the required page. So, there is a need for page replacement concept.

According to page replacement concept, one unwanted page called "victim" is identified & swapped out & that frame is made Empty. In that Empty frame, the required page is swapped in.

Steps in page replacement process:




a) The unwanted page called "victim" is swapped out to secondary memory.

b) The page table should be adjusted to accomodate the Information of swapped out page.

c) The required page is swapped in into the Empty frame in the physical memory.

d) The page table should be adjusted to accomodate the Information of swapped in Page.

3) Page - Replacement Algorithms:-

There are several algorithms available to find the victim page/unwanted page), few of them are:-

a) FIFO (First in first out)

b) optimal

c) LRU (Least Recently used)

To implement any page replacement algorithm, the following issues to be noted,

a) Input page string (Sequence of pages)

b) number of free frames available.

c) number of page faults to be counted.

a) **FIFO:-**

According to FIFO, the page entered into memory first will be replaced first.

**Advantages:-**

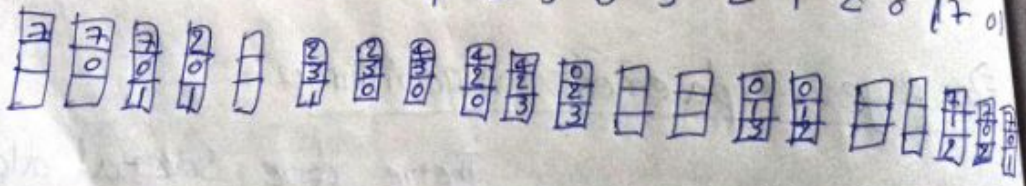
⇒ It is very easy to implement.

**Dis advantages:-**

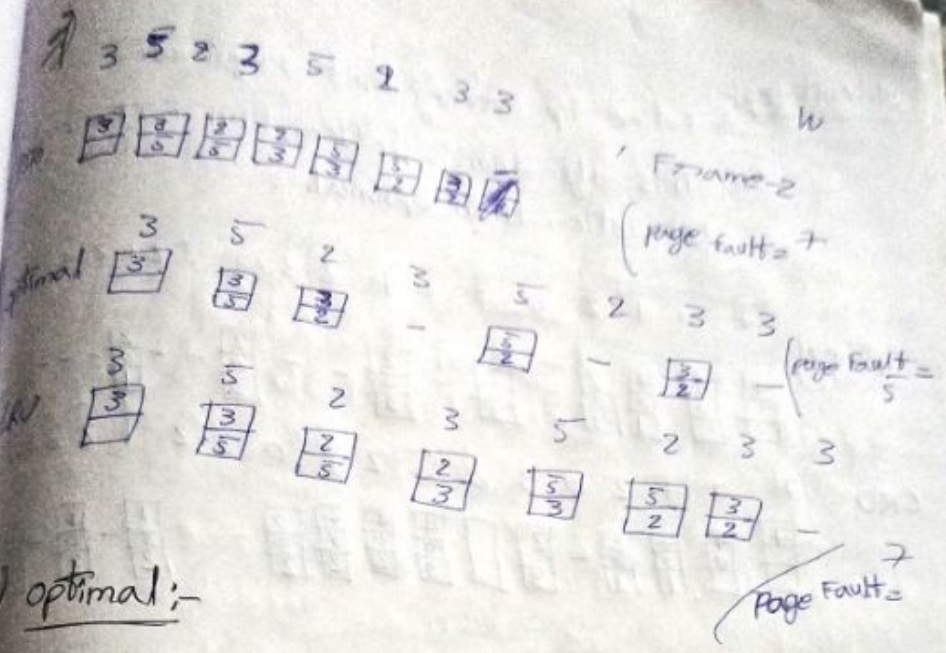
⇒ more number of page faults, will occur.

⇒ Execution takes more time.

**Ex:-** 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

FIFO 

Page fault = 15




1) optimal :-

According to optimal, logic followed is "The page will not be used for a longest time is replaced first".

Advantages :-

→ It gives less no. of page faults.

→ This is the best page replacement algorithm.

Disadvantage :-

→ Enter i/p string should be known in advance.

2) LRU :-

Logic of LRU is,

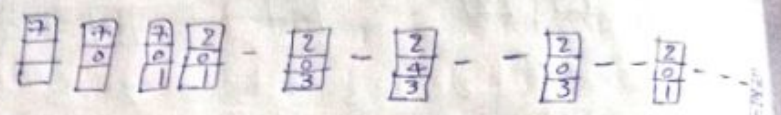
The page which hasn't been used for a longest time is replaced.

Advantages:-

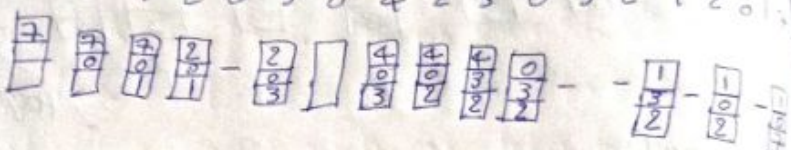
→ Entire i/p String isn't required.

→ This algorithm is better than FIFO, not better than optimal.

optimal 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0



LRU 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0



⇒ Allocation of frames:-

Depends on Job size, i/p String; the allocation of frames will be decided. But there is a logic that a min. no. of free frames should be allocated so that more page faults can be avoided as well as frames also not wasted.

Ex:-

i/p String: 20 pages

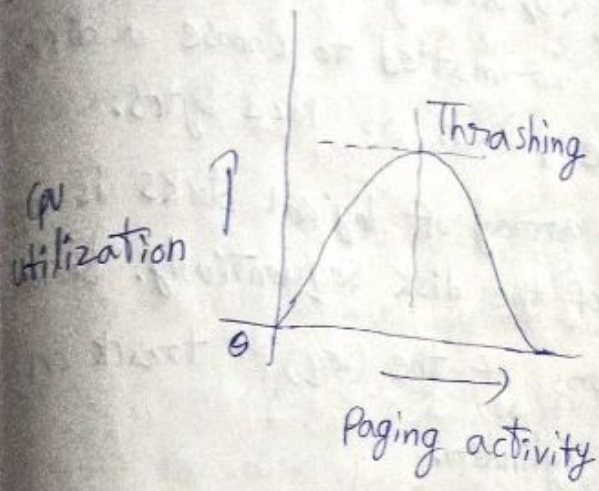
If we allocate only 1 frame, then 20 page faults will occur.

If we allocate more no. of frames for a small job, then more no. of frames are wasted.

## Thrashing:-

The high frequency of paging activity is called Thrashing.

The degree of multi-programming should be less so, that the CPU utilization is more & paging activity is less. The thrashing can be represented as follows.




When the paging activity is less, the CPU utilization is more, i.e., CPU Efficiency is increased.

When paging activity is more, the Efficiency of CPU drastically reduced.

## ⇒ Secondary Storage Structure;-

### mass storage structure;-

#### Disk structure;-

Modern disk drives are addressed as large on, dimensional arrays of Logical blocks, where logical block is the smallest unit of Transfers. The size of the logical block is $\frac{1}{2}$ kb, although some disks are/ can be low-level formatted to choose a different logical block size, such as 1024 bytes.

The 1D array of logical blocks is mapped into sectors of the disk sequentially. Sector 0 is the 1st sectors of the (seq) 1st track on the outermost cylinder.

#### Disk Scheduling;-

The major components in disk scheduling are;-

a) Seek time;- The time for disk arm to search the heads to the cylinder containing desired sector.

b) Rotation Latency;- The time waiting for disk to rotate the desired Sector to the disk head.

There are 3 different popular disk scheduling algorithms available.

a) FCFS scheduling
b) SSTF
c) SCAN

## a) FCFS scheduling Algorithm:-

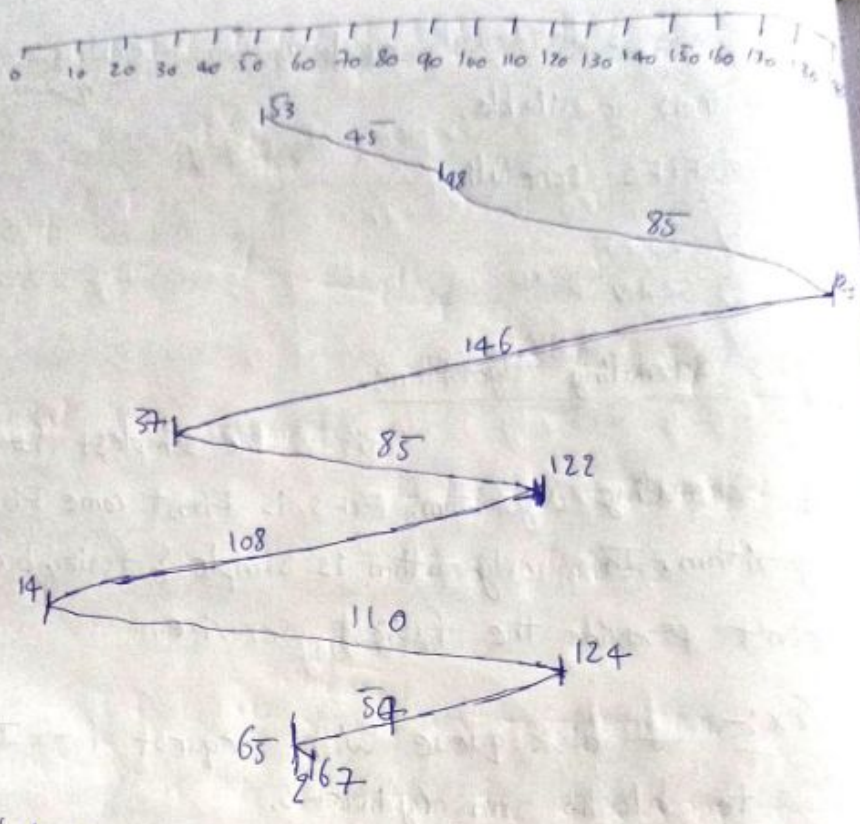
This is the simplest form of disk scheduling algorithm. FCFS is First come First Served algorithm. This algorithm is simple & fair, but doesn't provide the fastest service.

Ex:- A disk queue with request for I/o to blocks on cylinders.

98, 183, 37, 122, 14, 124, 65, 67 in the order.

If the disk hand is initially a cylinder 53. It will 1st move to 98, then to 183, 37, 122, 14, 124, 65, 67, for a total movement of 640 cylinders.

Request is:- [98, 183, 37, 122, 14, 124, 65, 67]




Total arm movements ⇒ $(98-53)45 + (183-98)85$

$+ (183-37)146 + (122-14)108$

$+ (122-37)85 + (124-10)114$

$+ (124-65)59 + (67-65)2$

$= 640$ cylinders.

## SSTF Scheduling Algorithm:

SSTF - Shortest Seek Time First

Logic :- All the requests close to the current head position, before moving to the head far away request.

So, this SSTF Algorithm selects the request with minimum seek time from the current head position.

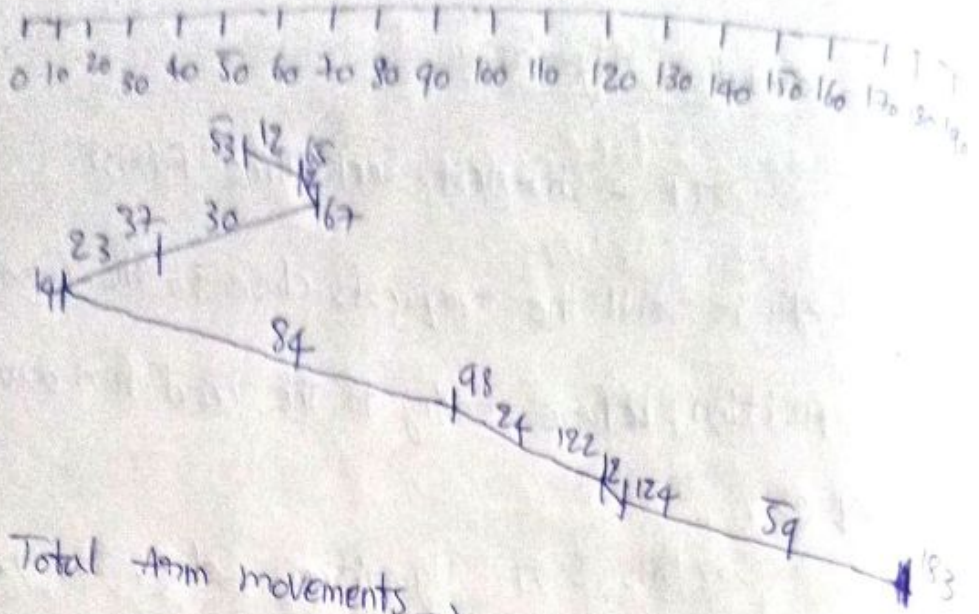
Ex:-

Request cylinders:- 98, 183, 37, 122, 14, 124, 65, 67.

Current position is 53rd cylinder, 65th cylinder is the closest to 53rd cylinder. So, arm moves from 53 → 65, Then 67, is closest to 65. So, arm moves from 65 → 67 and so on.

Request is 98, 183, 37, 122, 14, 124, 65, 67.

Current head position 53.




Total Arm movements =)

$12 + 2 + 30 + 23 + 84 + 24 + 2 + 59$

=) 236 Cylinders

c) SCAN Scheduling Algorithm:-

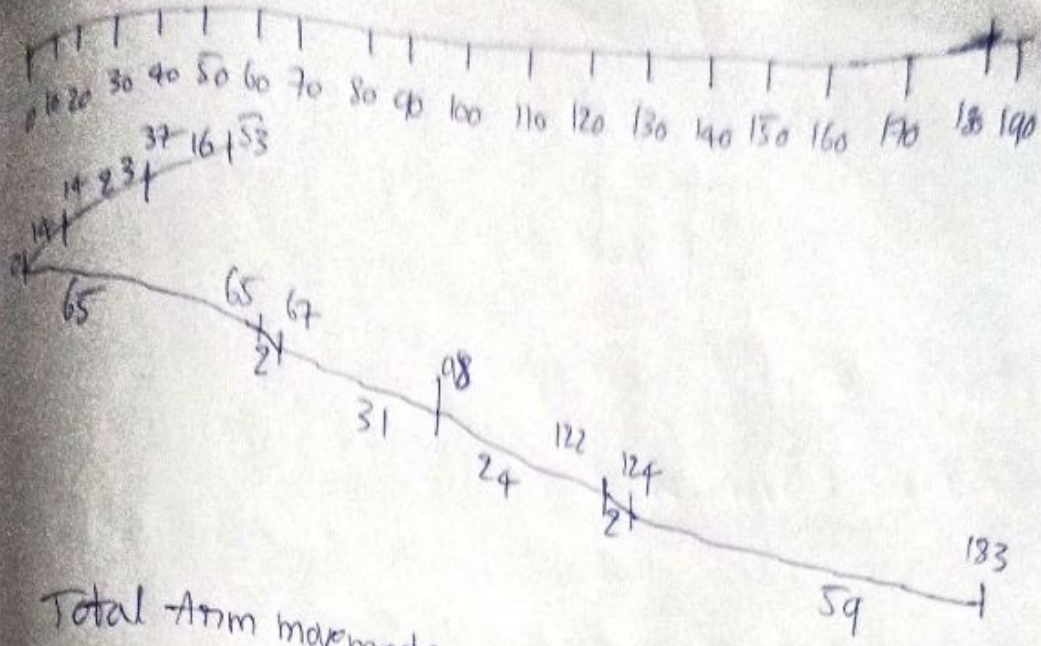
It is also (called as) Elevator Algorithm.

Logic:- In this Algorithm, the disk arm starts at one end of the disk & moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the end, the direction of head movement is reversed & servicing continues.

Ex:- Cylinder request:- 98, 183, 37, 122, 14, 124, 65, 67

Initially, we need to know the direction of head movement, in addition to the current position at 53.

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




Total Arm movements

=) 16 + 23 + 14 + 65 + 2 + 31 + 24 + 2 + 59

=) 236 cylinders