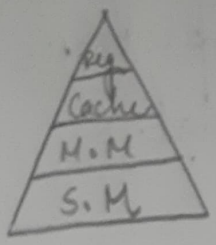
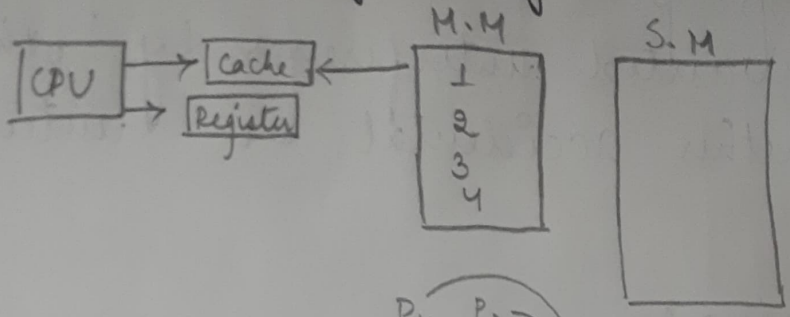
Date - 19.12.23

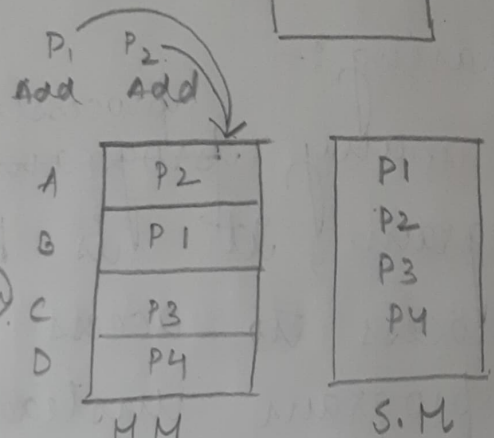
Ch: Memory Management

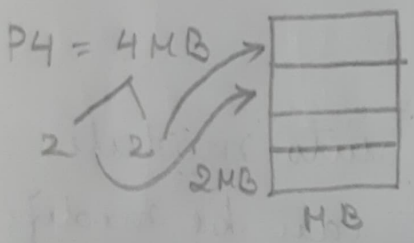


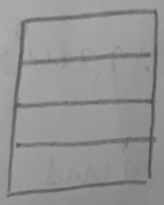


Requirements :-

1) Relocation
2) Protection
3) Sharing (same data, same operation)
4) Logical Organization
5) Physical Organization







→ Memory Management Requirements :-

1) Relocation :-
Once a program is loaded in main memory and swapped out it cannot be specified at what location it will swapped in Therefore, the OS needs to relocate the process to a different area of memory. The processor hardware and OS must be available to translate the memory references found in the code of program.

2) Protection :-
Every process should be protected against unwanted interference by other processes whether accidental or intentional.

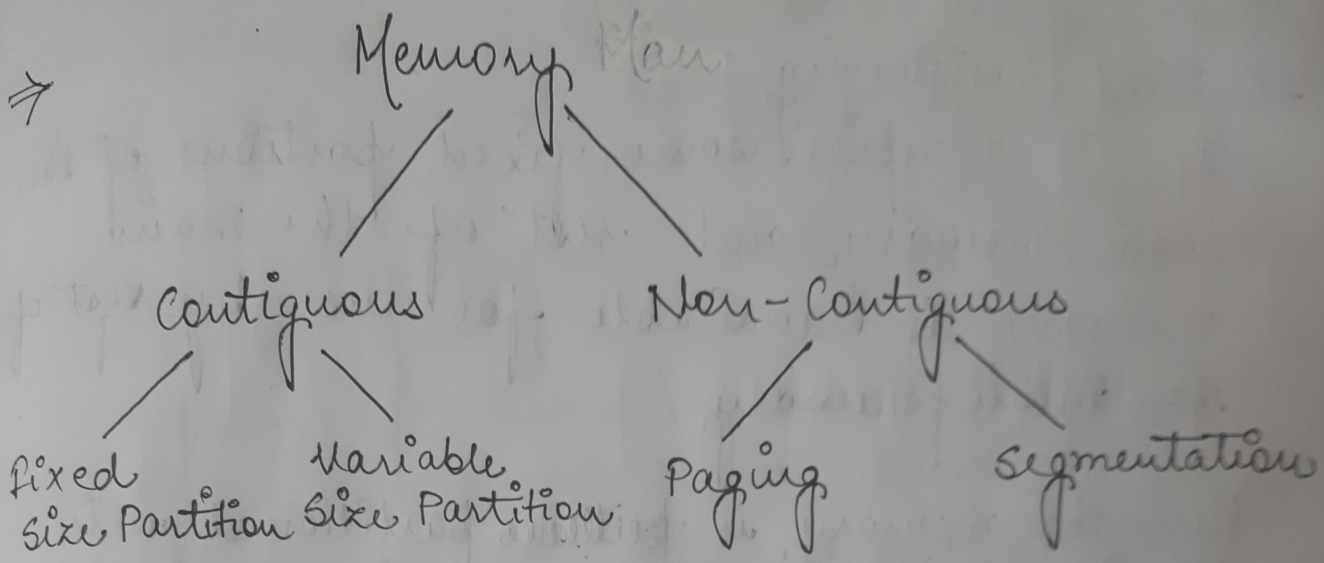
3) Sharing :-
If many processes are executing the same program, it is beneficial to allow each process to access the same copy of the program instead of having a separate copy.

4) Logical Organization :-
Most programs are organized into modules which contain data that can be modified. These modules can be written and compiled independently with all references from one module to another. The tool that is used is segmentation.

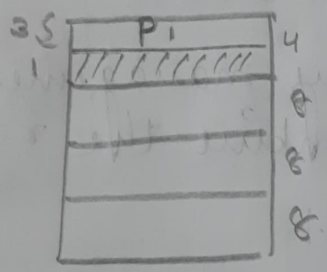
5) Physical Organization :-
The main memory may be insufficient for a program. In this case, overlaying can be used where program and data are organized in such a way that

various modules can be assigned the same region of memory. There is a main program for switching the module in and out as needed.

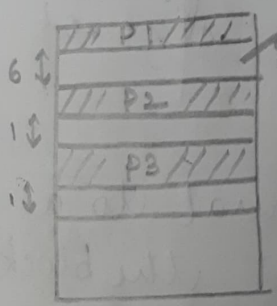
⇒



P1 = 3
P2 = 7
P3 = 7
P4 = 1
P5 = 9



Fixed Size Partition :-



Equal Size MM

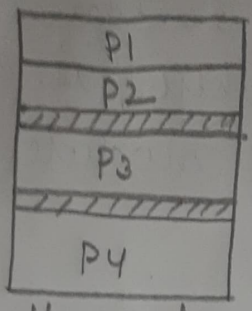
P1 = 2
P2 = 7
P3 = 7
P4 = 14

1) Internal fragmentation
2) limit on size of process
3) limit on no. of processes
4) External fragmentation

} disadvantage -3

⊙ No. of processes in main memory at a time is called degree of multiprogramming.

P1 = 2
P2 = 7
P3 = 15
P4 = 32



Unequal size MM

Disadvantages:-
1) Internal frag.
2) } same as
3) } equal division
4) }

⇒ Fixed Partioning :-

The OS occupies some fixed portions of the main memory and rest of the main memory is available for use ~~by~~ of multiple processes.

Here the memory is partitioned into region with fixed boundaries.
It is a contiguous memory management scheme where we cannot combine the regions.

* Disadvantages -

1) A program may be too big to fit into a partition. This scheme limits size of processes.

2) There is a wasted space internal to a partition due to the fact that the block of data is smaller than the partition which is known as Internal fragmentation

3) Even after having enough space, we cannot accomodate a new process which is known as external fragmentation.

4) The number of processes that can be loaded in main memory are limited, i.e the degree of multiprogramming is limited.

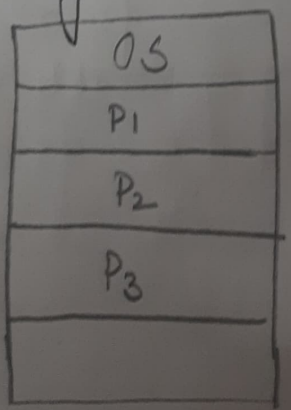
→ Fixed Size Unequal Partitions :-
This solves the problem upto some extent but still consists of above disadvantages.

⊙ In unequal sized partitions, a scheduling queue is needed for each partition to hold swapped out processes meant for that partition. But, it is not benefical as even after having space and smaller processes could have been assigned to it. So, we maintain a single queue for all the partitions and when it is the time to load a process into main memory, the smallest available partition that will hold the process is selected.

Date - 20.12.23

⊙ Variable Partioning.

20, 14, 18, 8



→ Here the space is allocated to the process dynamically on their request. There are no static partitions made before hand. It solves the problem of fixed partioning but still has few disadvantages :-

(i) External fragmentation
(ii) Allocation & De-allocation is complex.

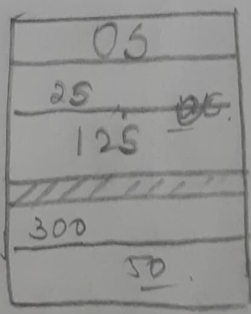
→ Compaction solves the problem of external fragmentation by combining the free space but it is not feasible as we need to copy millions of bits in order to achieve this.

Q. How processes are allocated in holes?

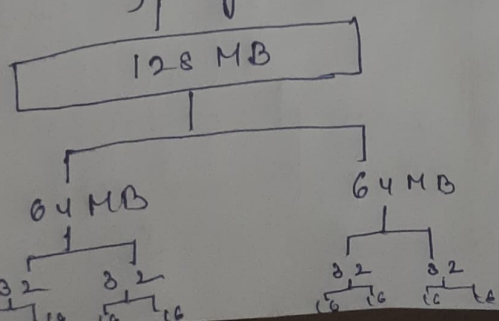
1) First fit → Best Performance (which can accomodate first)
2) Next fit (check from the next portion)
3) Best fit → Worst Performance (less wastage)
4) Worst fit (more wastage)

300, 25, 125, 50

✓ FIRST FIT
✗ BEST FIT



Buddy System

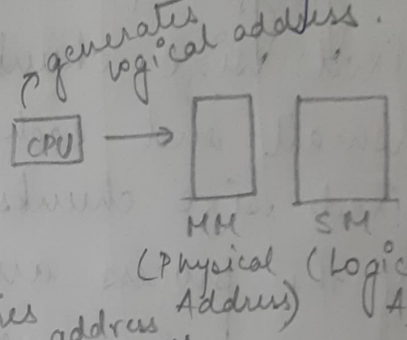


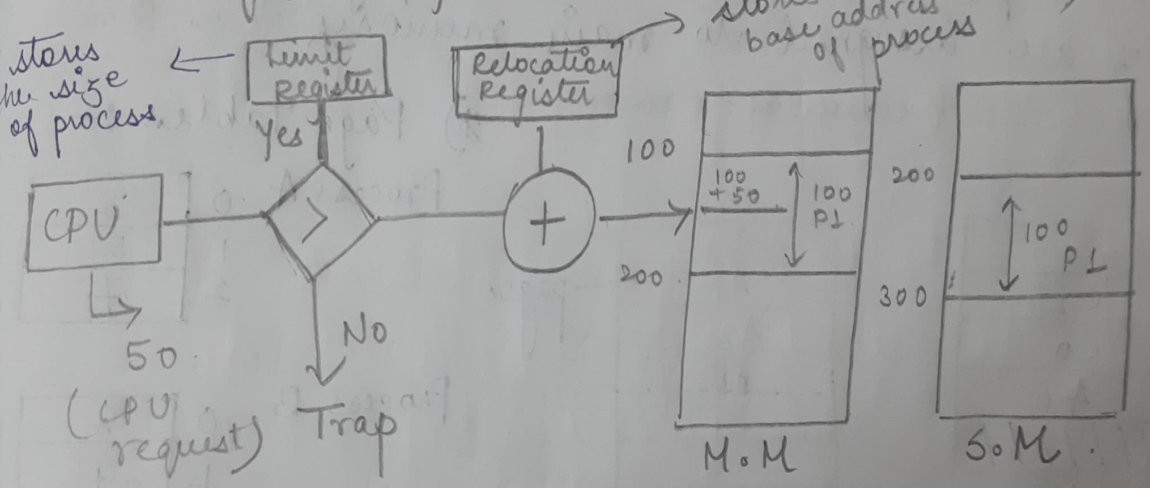
* Buddy System

In this system, memory is allocated based on the powers of 2. The memory is further split in powers of 2 if any process has size less than the current memory available

Date 22.12.23

* Address Translation :-
(Mapping b/w logical Address and Physical Address)





→ Logical Address is generated by CPU for secondary memory but we have data in main memory so we need to relocate the values using Address Translation. Logical Address is translated into Physical Address.

→ In contiguous memory, we use Relocation register and limit register for this purpose. Relocation register is a special purpose register which holds the base address of the process in main memory. Limit register stores the size of the process

* Non-Contiguous Memory Allocation :-
Advantage - No fragmentation

→ Partioning of processes in equal size chunks in secondary memory is known as Paging.

* ~~Paging~~ Main memory is partitioned into equal fixed size chunks that are relatively small and each process is divided into small fixed size chunks of the same size. The chunks of the process are known as pages and when they are assigned to main memory then these chunks are known as frames.

*) Page tables :-

Process A

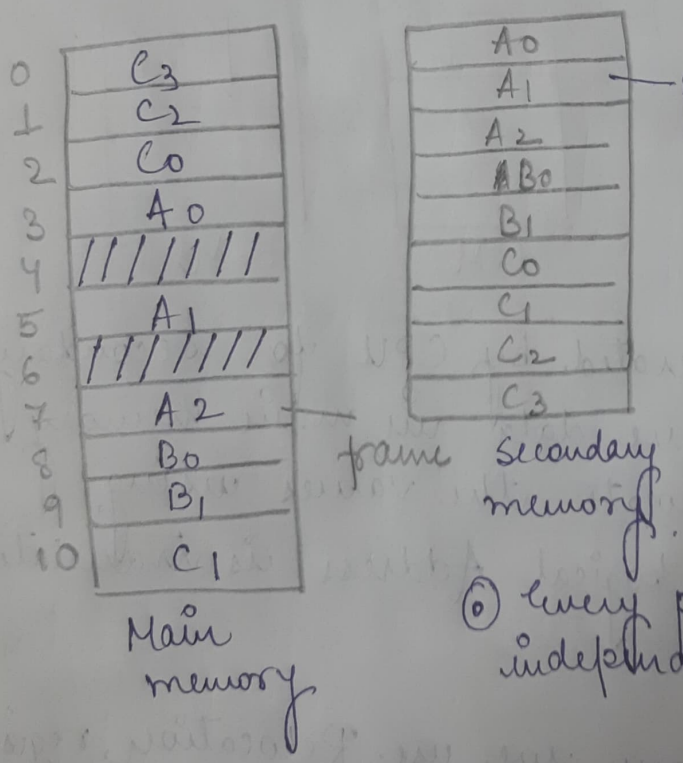
0	3
1	5
2	7

Process B

0	8
1	9

Process C

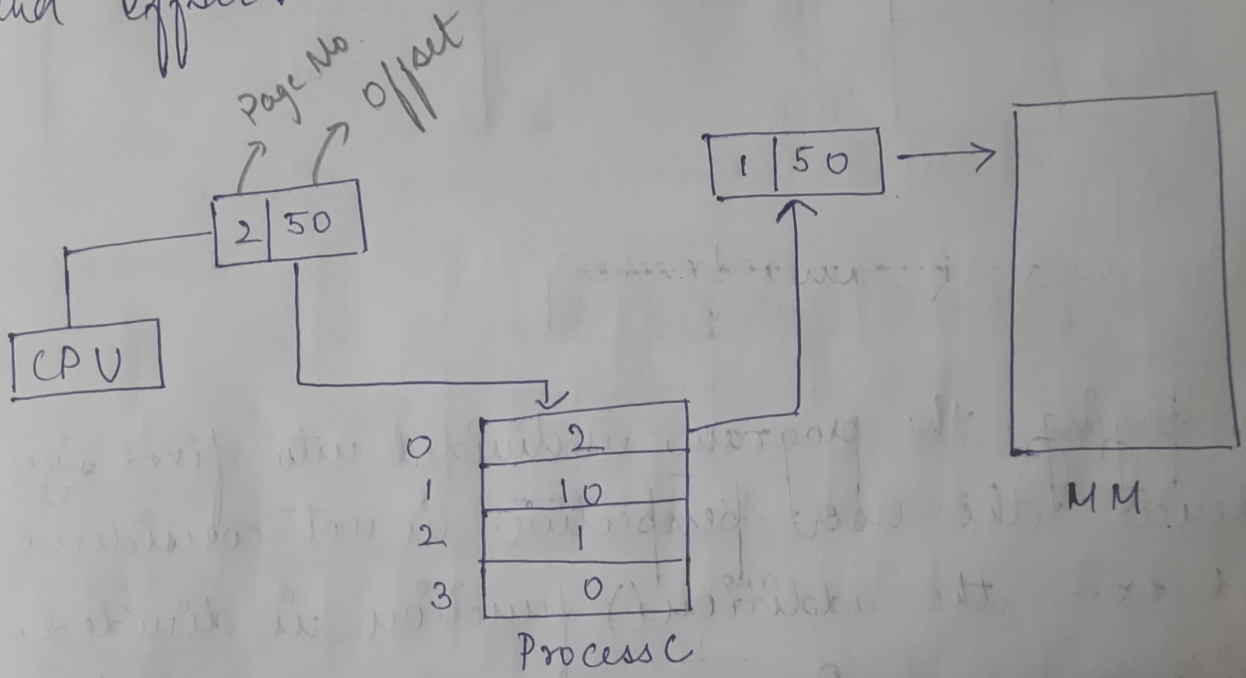
0	2
1	10
2	1
3	0



⊙ Every process has individual independent page table.

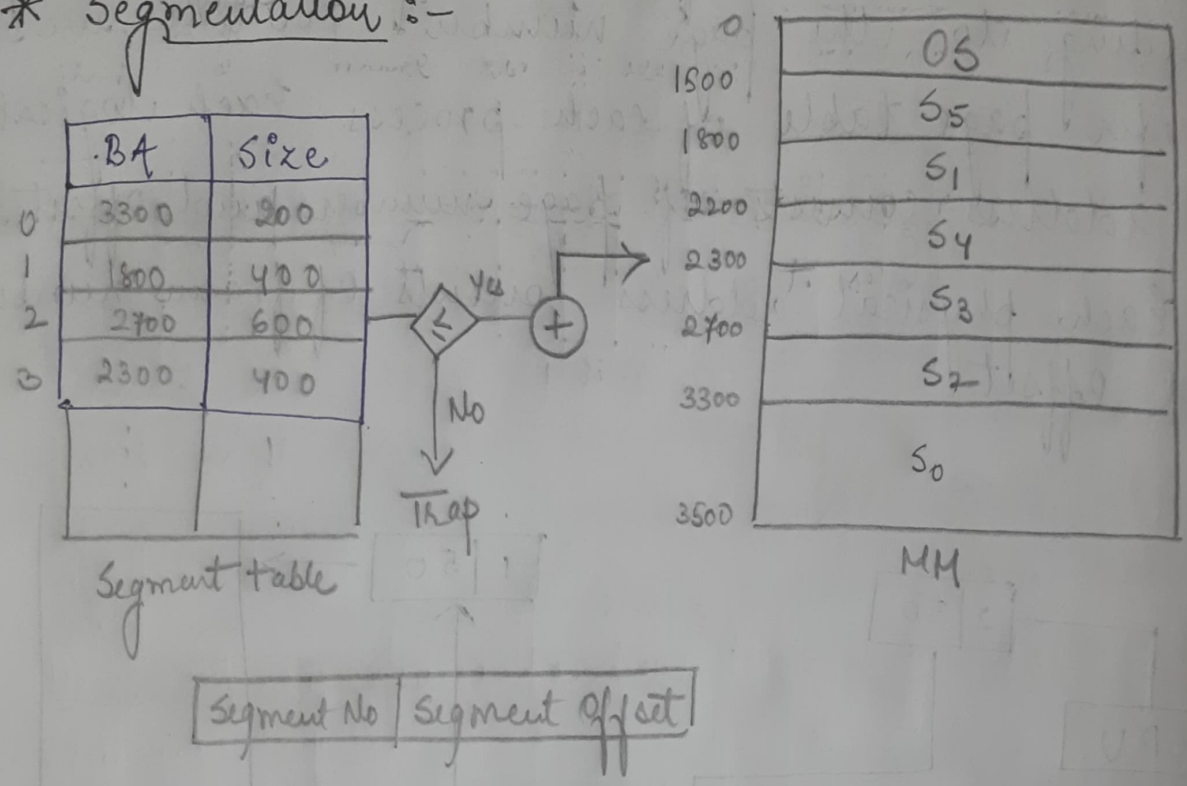
→ The OS maintains a page table for each process. The page table shows the frame location for each page of the process. Every page has independent page table.

⇒ The logical addressing scheme in case of paging is we need to find the frame no. corresponding to the page number. For this, we refer page table of each process. Each logical address consists of page number and offset. Each physical address consists of frame number and offset.



Date - 02.01.24 → division from user perspective.

* Segmentation :-



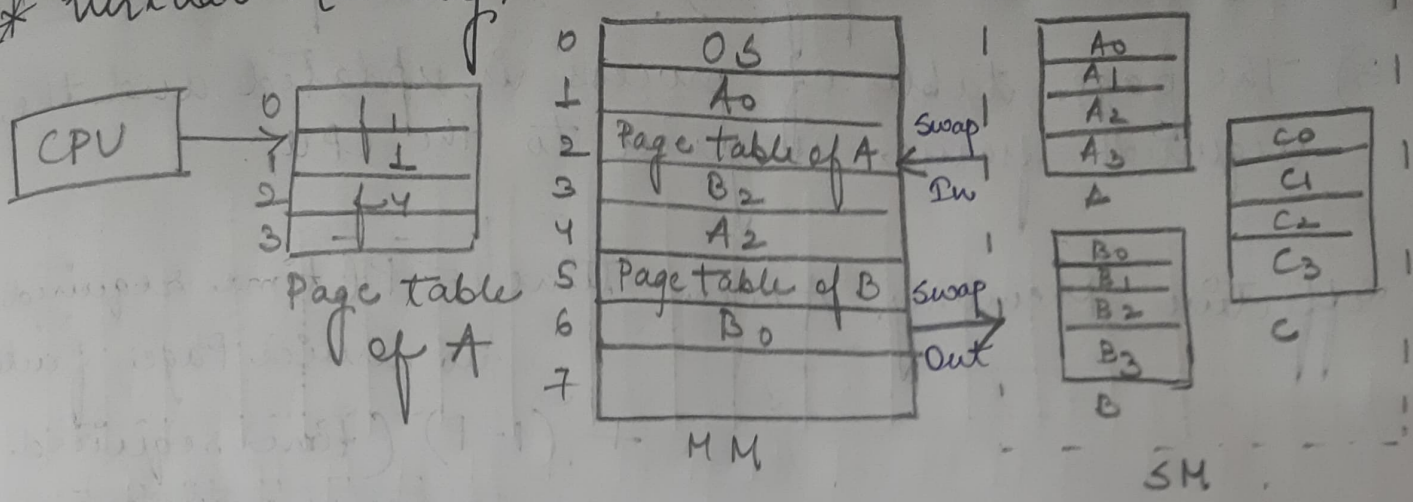
Segment No	Segment Offset

In paging, the program is divided into fixed size, hence the user perspective is not considered. for ex- the addition() function is divided into 2 frames. For execution, CPU needs to fetch both the frames but, it will degrade the performance of the system. Segmentation solves this problem as ~~by divid~~ the program is divided into segments from user's prospective.

Segments are not divided into fixed sizes but in variable sizes

A segment table is maintained using a segment table and offset.

* Virtual Memory :-



Virtual memory provides an illusion to the programmer, that a process whose size is larger than the size of main memory can also be executed. It also benefits the degree of multiprogramming as we can bring more and more processes in the main memory. Using virtual memory, we only bring required pages of the process in main memory.

Page fault → Occurs when asked page by the CPU has to be fetched from SM (i.e the required page is not present in MM).

* Steps for page placement if page fault occurs:

1) The mode is switched from user to OS.
2) The OS will then check the main memory to find if the page is present or not. If the page is not present, page will be fetched from secondary memory to the empty slot of main memory (if present)

otherwise some page needs to be replaced.

3) The page table of process is updated and the required frame is given to the CPU.

effective Memory Access time = P(time required to service Page fault) + (1-P)(time required to access main memory)

* Replacement Policy -

We need to consider few things for the replacement of pages :-

1) First one, how many page frames are to allocated to each active processks.
2) Second, whether the set of pages to be considered for replacement should be limited to those of the process that cause the page fault and encompass all the page frames in MM
3) Third, among the set of pages considered, which particular page should be selected for replacement.

There are 3 policies used for replacement:-

1) FIFO (First In First Out)
2) LRU (Least Recently Used)
3) ORU (Optimal Recently Used)

Date - 03.01.23

* Replacement Policies :-

FIFO . LRU . Optimal

Q1. 3 frame memory

1, 3, 0, 3, 5, 6, 3 (i) FIFO

3			0	0	0	0	3
2		3	3	3	3	6	6
1	1	1	1	1	5	5	5
	M	M	M	H	M	M	M

6 Misses $\frac{6}{7} \times 100\%$

1 Hit $\frac{1}{7} \times 100\%$.

Q2 FIFO

3 frame memory - 4, 7, 6, 1, 7, 6, 1, 2, 7, 2

3			6	6	6	6	6	6	7	7
2		7	7	7	7	7	7	2	2	2
1	4	4	4	1	1	1	1	1	1	1
	M	M	M	M	H	H	H	M	M	H

6 Misses

4 Hits

Q3 FIFO

3 frame memory - 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0

3			1	1	1	1	0	0	0	3	3	3	3	2	2
2		0	0	0	0	3	3	3	2	2	2	2	1	1	1
1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0
	M	M	M	M	H	M	M	M	M	M	M	H	M	M	H

3 Hits 12 Misses

Q4 4 frame memory (ii) ORU.

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3			1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1
2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7	7
	M	M	M	M	H	M	H	M	H	H	H	H	H	M	H	H	H	M	H

11 Hits.

Q5. 3 frame - 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 (ORU).

3			6	6	6	6	6	6	6	6
2		7	7	7	7	7	7	7	7	7
1	4	4	4	1	1	1	1	2	2	2
	M	M	M	M	H	H	H	M	H	H

5 Hit, 5 Miss

Q6. 3 frame - 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0

3				2	2	3	4	4	4	4	1	1	1	1	1	1	1	1	1	1
2		1	1	1	0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	0
1	6	6	6	6	6	6	6	6	6	2	2	2	2	2	2	2	2	4	4	4
	M	M	H	M	M	M	M	H	H	M	M	H	H	H	H	M	H	H	M	M

9 hits
11 Miss

Q7 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
4 frame (iii) LRU

4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3			1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1	1
2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7	7	7
	M	M	M	M	M	M	M	H	H	H	H	M	M	H	H	M	M	H	H

9 Hits, 11 Miss

Q8. 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 (LRU)

3			6	6	6	6	6	6	7	7	
2		7	7	7	7	7	7	2	2	2	
1	4	4	4	1	1	1	1	1	1	1	
	M	M	M	M	H	H	H	M	M	H	

4 hits
6 miss

Q9. 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0

3				2	2	2	4	4	4	2	2	2	2	2	2	2	2	2	2	0
2		1	1	1	1	3	3	3	0	0	0	0	0	0	0	0	0	1	1	1
1	6	6	6	6	0	0	0	6	6	6	1	1	1	1	1	1	3	3	4	4
	M	M	H	M	M	M	M	M	M	M	M	M	H	H	H	H	H	H	H	M	M

7 Hits

Date - 05.01.24

* Frame Allocation :-

Min^m requirement of pages in the MM for a process = no. of pages to execute a single instruction

→ Some specific frames are allocated for the processes, frame allocation techniques tells us how many frames are allocated to the process.

Minimum requirement is the pages required to execute one instruction & the max^m requirement is loading the whole process in M.M.

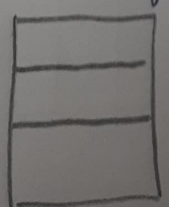
The frame allocation should lie b/w min^m and max^m requirement

Frame Allocation Techniques

↙ Equal

$P_1 = 20$
$P_2 = 100$
$P_3 = 500$

$\frac{60}{3} = 20$



↓ Weighted

$P_1 = 30, P_2 = 30, P_3 = 40$

$T = 100$, Frames = 10

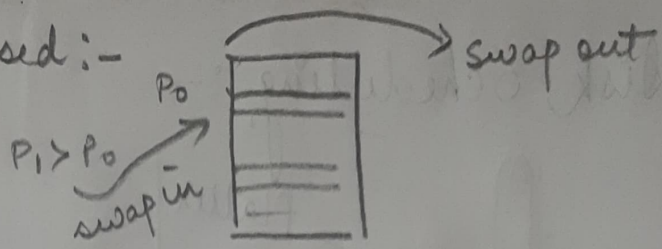
$P_1 = \frac{30}{100} \times 10 = 3$

$P_2 = \frac{30}{100} \times 10 = 3$

$P_3 = \frac{40}{100} \times 10 = 4$

→ Dynamic / Priority Based

Dynamic / Priority Based :-



→ There are 3 ways for the allocation of frames :-

1) Equal

Here, we just divide the frames with the total no. of processes without considering the weights of each process. This leads to inequality as some processes may allocate all of their pages and second is it doesn't consider priority of processes.

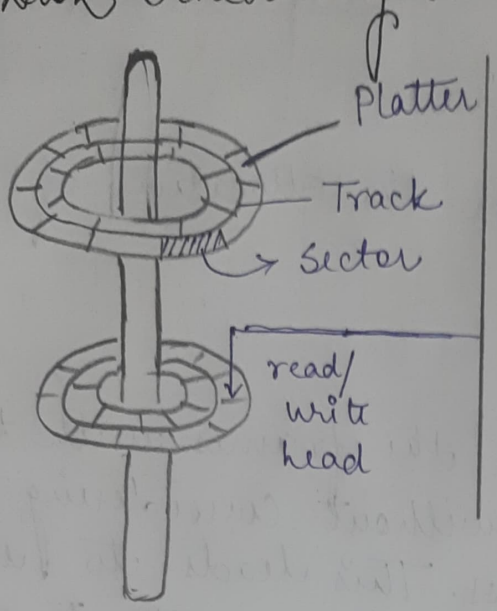
2) Weighted

Here, the weights of each process (no. of pages) are (taken into consideration) and then respective frames are allocated.

3) ~~P~~ Dynamic / Priority

The frames are allocated according to the demand at the runtime. If a higher priority process wants to allocate its frame, then the frames of lower priority process are swapped out.

* Disk Scheduling :-



Techniques :-

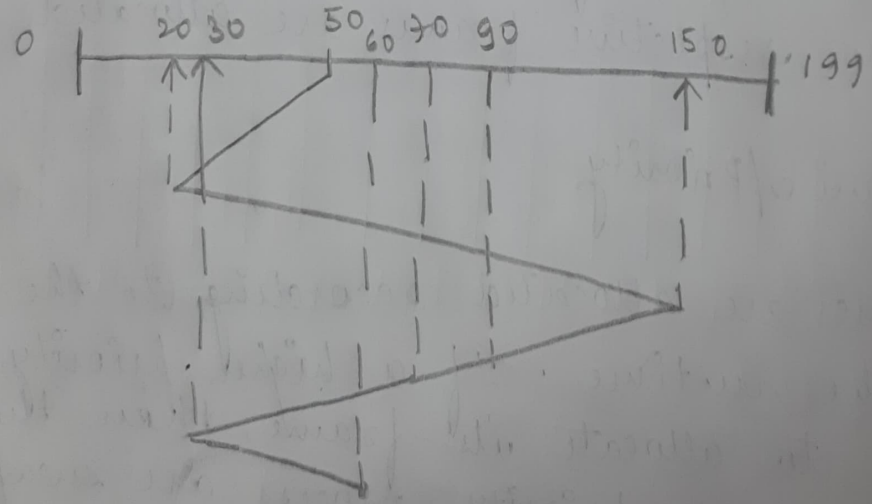
1) FCFS
2) SSTF

① FCFS

No. of tracks (0-199) = 200

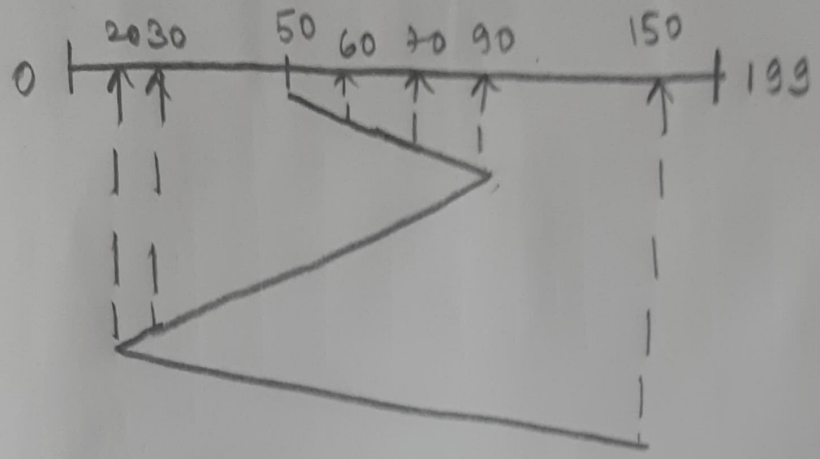
Current R/W head = 50

{ 20, 150, 90, 70, 30, 60



Seek time = (50-20) + (150-30) + (60-30)

(2) SSTF :-



Seek time = (90 - 50) + (90 - 20) + (150 - 20)