Unit-4 (Memory Management)

Date

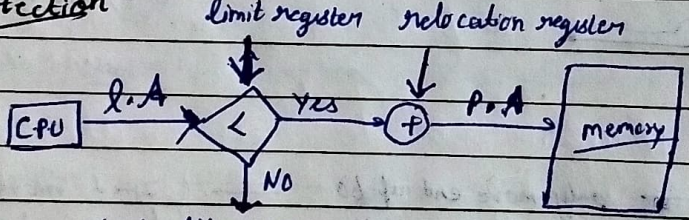
Logical Address or Virtual Address
{address generated by your program, act as placeholder for where data is supposed to be}
{range: 0 to max}

Address mapping
↓
Translate LA to PA
Done by Hardware called Memory management unit (MMU)

Physical Address
{actual location in your computer's memory (RAM) where data is stored}
↓
{range: R+0 to R+max
R → base register value}

Swapping → It increases degree of multi-programming by temporarily moving a process out of main-memory to backing store

- Context switch time in swapping = 100 mb process, disk transfer 50mb/s
  ∴ swap time = 100/50 = 2s or 2000 ms
  And for both In and Out = 4000 ms.
- In * Mobile device, swapping is avoided {∵ flash memory has limited R/W cycles}
  So, * Android terminate process
  - IOS → ask app to voluntarily free memory!
- **Memory protection**



So, the relocation register scheme allows O.S to change dynamically,
∵ O.S contain code and buffer space for device drivers, If device driver not in use, this code will go out from memory, ∴ called transient O.S code.

best fit – first – worst – fit.

- **fragmentation**
  - external fragmentation → unused free space scattered blw allocated blocks
  - Internal fragmentation → unused memory within allocation blocks due to fixed size partitioning.

Spread

50% rule → Half of many blocks are wasted as allocated

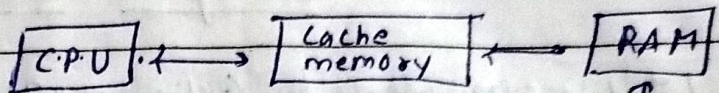
~~Fragmentation~~ of first-fit e.g → 10 blocks are allocated, approx 5 are lost in fragmentation.

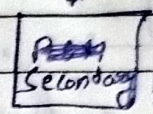
What to do? → Compaction → • merge scatter free blocks into a large single block.

• requires dynamic relocation during execution.

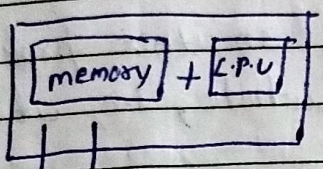
Date 31-08-24

Memory Management





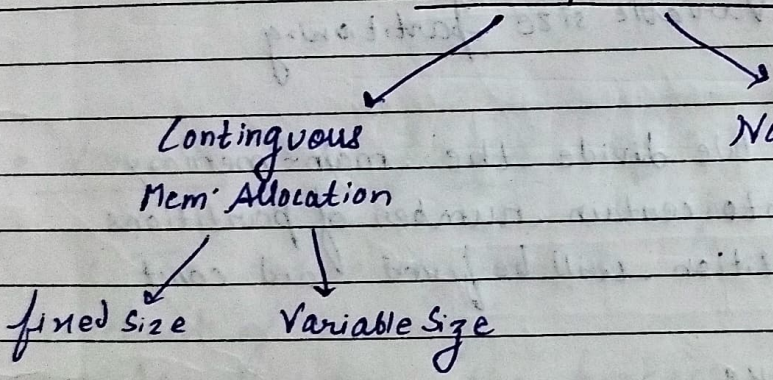
Computer



Idle characteristics of Memory → Size ↑
→ Access Time ↓
→ Per unit Cost ↓

We have Hierarchy of Memory.

. Memory allocation



• ~~Contiguous mem allocation~~ : C.P.U can access only main memory directory, and secondary memory is attached at the back-up side with main-memory.

So, whatever process we want to execute, we must break that process from S.M to M.M., and then C.P.U can access that process in the main memory.

. There are two types of memory allocation scheme → Contiguous and Non-Contiguous

Spiral

Date

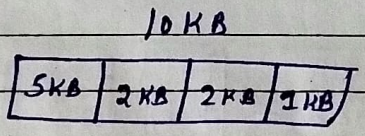
Continguous Mem Allocation

- What do you mean by Contingvous?

↳ • Suppose there is a process P1, so this process is stored in the main-memory as a whole, i.e, it can't be broken down into pieces.

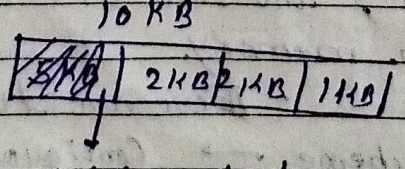
- There are two kinds of Contingvous memory partitioning scheme
  - → fixed size partitioning
  - → Variable size partitioning

- **Fixed size partitoning :** • We divide the main-memory into certain number of partitions
  - Size of each partition will be fixed and can't be changed.



The number of partitions, in the main-memory and the size of each partition may vary from system to system.

Let us assume, we have a process { P1 requiring 4 K.B }



So, P1 = 4KB

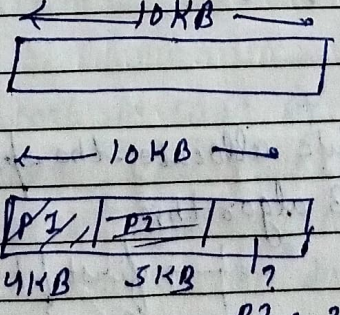
1 KB waste here !

Date

- Disadvantages of fixed size → • Can't have more than one process in one partition.
  • It leads to Internal fragmentation { when the memory space in a fixed partition is left, and we can't use it }

- advantage → easy to manage.

- Variable Size partitioning : • Here we don't have pre-defined partitions
  • The entire system is treated as a single chunk of memory
  • As and when the process comes, we allocate the memory to it, according to its memory requirement.



P3 : 2KB { Not able to fit in this main-memory }

Spiral

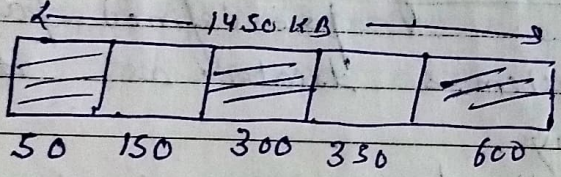
Date

- adv → There is no internal fragmentation.
- dis-adv → It can lead to external fragmentation, i.e, the total space which is required by a process is available in the main memory, but it is not available in the contiguous fashion,

∴ That space can't be allocated.

· Variable Size Partitioning Scheme

Consider, the present status of memory, is as given below.



P1 = 300 KB
P2 = 25 KB
P3 = 125 KB
P4 = 50 KB

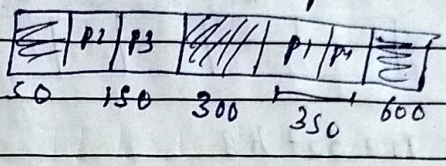
} first fit, Best fit, Worst fit → 3 algo

~~first fit algo~~ → We allocate the ~~process~~ memory to the processes on the basis of 3 algorithm

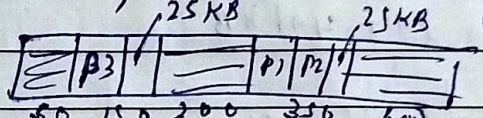
- first fit
- Best fit
- Worst fit

Date

- **First fit:** Start searching the memory from the beginning and allocate that space, which is capable of holding that process



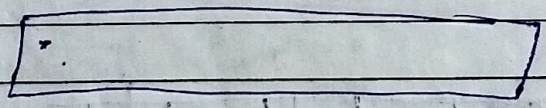
- **Best fit:** Search all the empty spaces available, and allocate the smallest block, which can satisfy the best of the process



- **Worst fit:** Opp' best fit
  - ↳ request of P4 can't be fulfilled (50KB) ↓ external fragmentation
  - bcoz, space available in memory is 50KB, but not in cts fashion.
  - thus leading to external frag.

- The key point to note in all the algo is, that we can include the left over memory space, inside the same block again.

Worst fit:



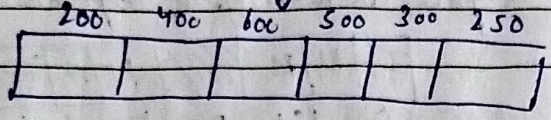
- Worst fit, works best in case of Variable size partitoning, bcoz, the remaining block will also be of large size, suitable to accomodate the need of another process.
  ∴ avoiding external fragmentation.

Spiral

Date

HW

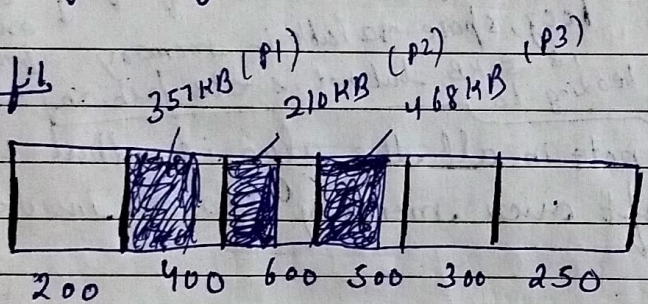
• fixed Size partitioning



P1 = 357 KB
P2 = 210 KB
P3 = 468 KB
P4 = 491 KB

↳ apply all algo

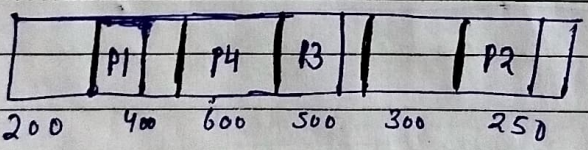
first fit



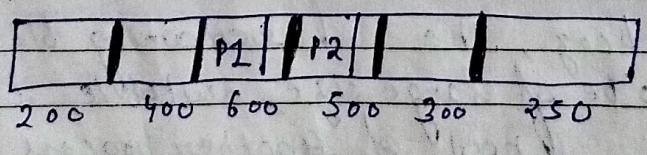
P4 can't be accomodated in the memory

(Internal fragmentation occur)

best-fit



Worst-fit



∴ P3, P4 can't be allocated

Spiral

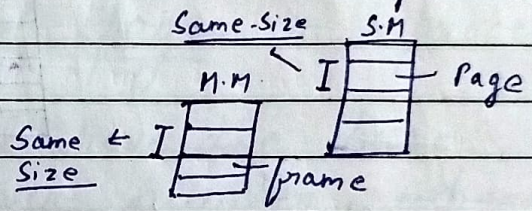
* In Non-contiguous → (P1, P2, P3) represent Pages for a particular process (X).

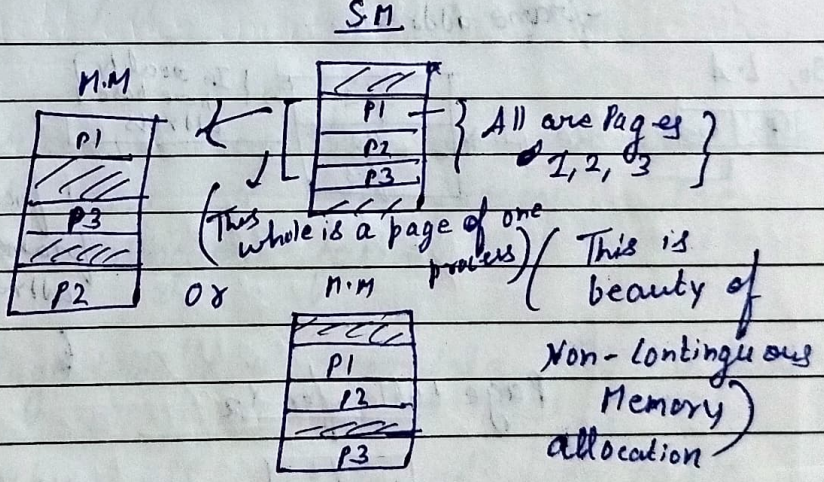
Date _____

Non-Contingous Memory Allocation

To support non-contiguous memory allocation secondary memory is divided into equal size partition, and each partition is called a page.

Main Memory is also divided into equal size partitions and size of each partition is same as the secondary memory partition. Here each partition is called a frame.

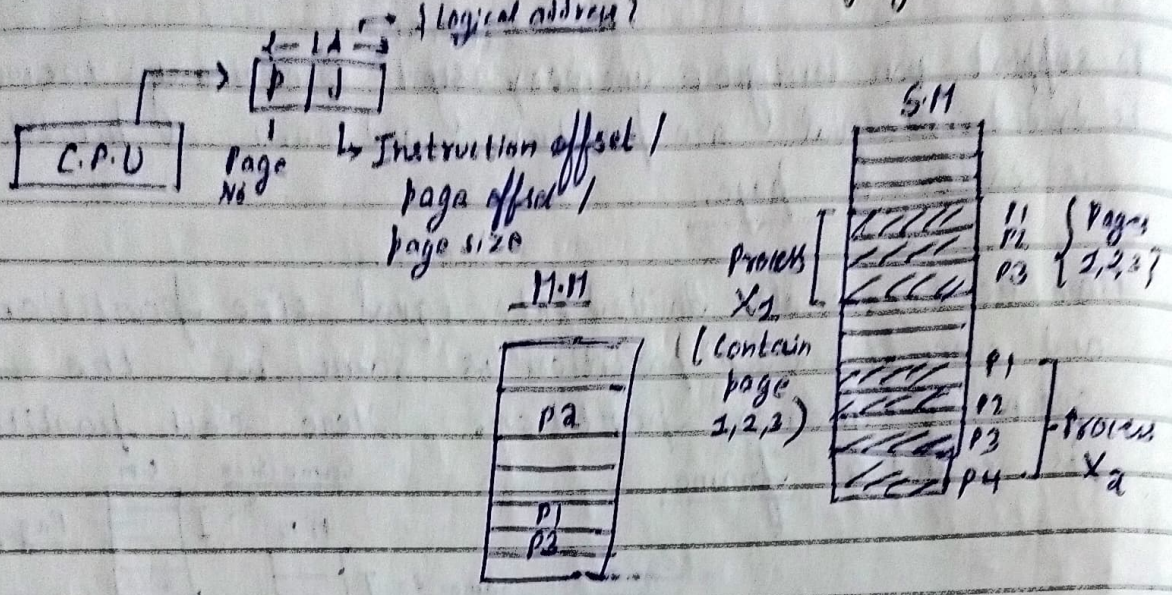




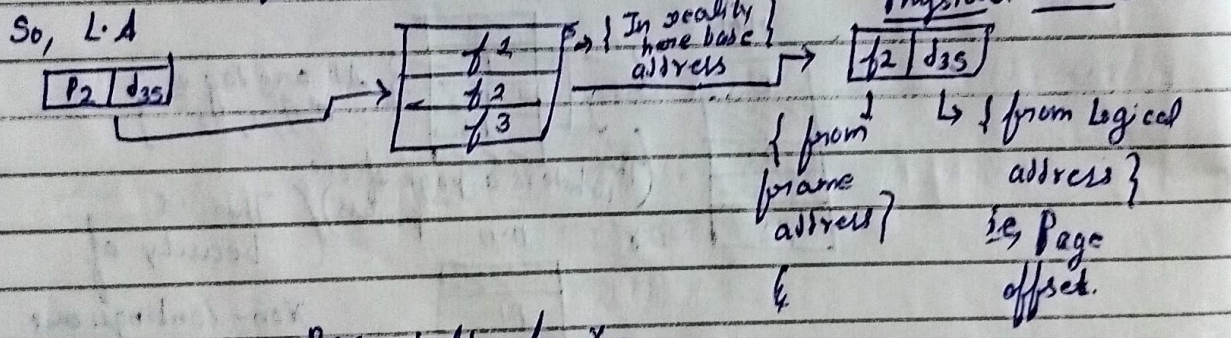
- C.P.U Generates logical address (L.A) and this L.A is for Secondary memory. But, Now we have taken a process from S.M and kept it in M.M. So, to access M.M we need physical address (PA) Hence, we need to convert L.A to P.A

Date

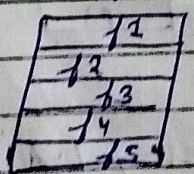
• Address Translation in Paging



Page table for process X_1 storing the frame address



Page table for X_2



Spiral

Date

· Translation of L.A to P.A

We will maintain a New Data structure, called a page-table. This page-table will be for each process, stored in the M.M.

Number of slots in the page-table, will be equal to the number of pages in a process.

Each slot of page table, will have the base address of the frame number, where that respective page is stored.

· adv of paging : · Acess is very fast and we are able to store the process in the M.M, in a non-continguous fashion.

· Disadv : · Page table is a meta-data (data about data), it's a penalty.

Every process will have an ind. page table.

· Paging is a fixed size partitioning scheme, ∴ it suffers from internal fragmentation.

· With, page table, instruction access time becomes double, as page table is also stored in Main-Memory,

∴ M.M is first accessed, to find the base address of the frame number, and, this (then) base address is combine, with instruction offset, to get actual physical address.

Date

This physical address is also stored in M.M, ∴ M.M is access twice.

$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1K$ (1024)
$2^{20} = 1M$
$2^{30} = 1G$
$2^{40} = 1T$

Numericals

Given: Logical address space = 4 G.B
physical address space = 64 M.B
page size = 4 K.B

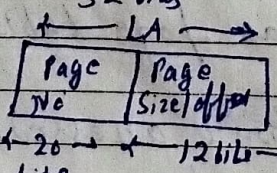
Q. find:

(a) No of pages, (b) No of frames? (c) No of entries in page table?
(d) Size of page table?

Soln :- We know that memory is byte addressable

(b) ~~No of frames~~ 1) L.A.S = 4G.B , ~~Page size~~
$= 2^2 \times 2^{30}$
$= 2^{32}$
∴ 32 bits are required to represent Logical address space

2) Page size = 4K.B
$= 2^2 \times 2^{10} = 2^{12}$
∴ 12 bits are required to represent the page size / page offset



Spiral

Date

∴ No of bits for page No (required) storing/represent = LAS − Page Size
= 32 − 12
= 20 bits

(a) ∴ No of pages = 2^{20}

- Physical address space = 64 MB
= $2^6 \times 2^{20}$
= 2^{26}
∴ 26 bits are required

frame size = page size
∴ No of bits required to represent frame offset / page offset = 12 bits

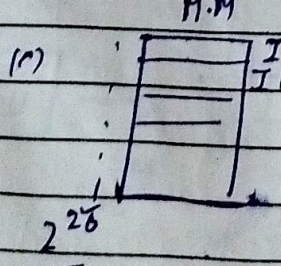
∴ ← PA → (26 bits)

base addr	page offset
← 14 bits →	← 12 →

∴ No of bits for base address/frame = 14 bits

(b) ∴ No of frame = 2^{14}

(c) M.M



No of entries = No of pages × No of bits required to represent a frame

⇒ $2^{20} \times 14$

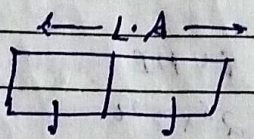
(d) Size of page table = No of entries in page table (Spiral)
⇒ $2^{20} \times 14$

Date

Q2) Assuming a 1KB page size, what are the page number and offset, for the following page address references? (Given in decimal)

(a) 3085

Soln



Page Size = 1KB

$= 2^{10}$ Bytes

$= 1024$ B

(a) 3085 → Page No $\Rightarrow \frac{3085}{1024} \Rightarrow 3$

(Mem add ref.) → Page offset = 3085 mod 1024 = 13

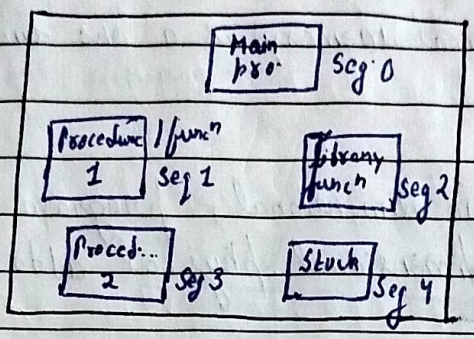
$\therefore \langle 3, 13 \rangle \rightarrow 3085$

Spiral

Date

Segmentation

In segmentation, we deal with different segments of a program, as shown below:



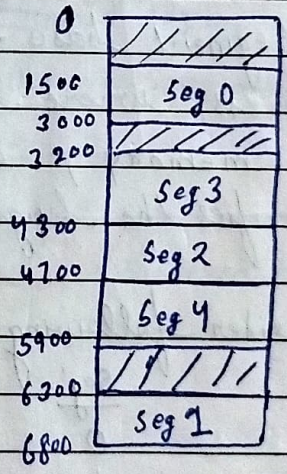
The Segment are name as Seg 0, 1, 2, 3 ... 'n', each has its own limit and base address.

{Size of Segment}

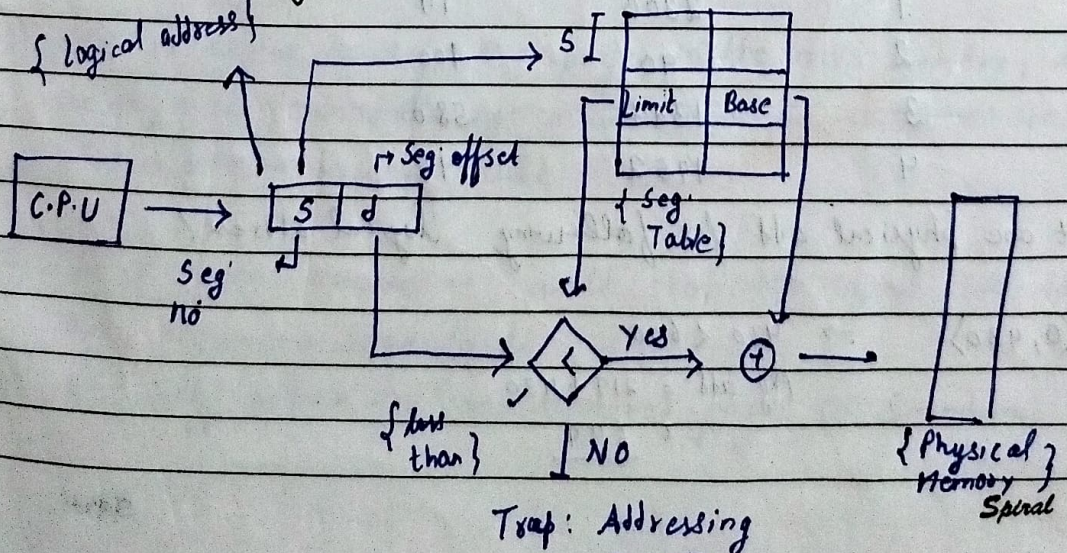
	Limit	Base
0	1300	1500
1	500	6300
2	400	4300
3	1100	3200
4	1200	4700

{base address}

Physical Memory



- Segmentation Hardware



{Phy add = base add + segment offset}

Date

Although, the programmer can refer to the objects in the main memory by a two dimensional address, but the actual physical memory is a one dimensional sequence of bytes.

∴ We must map two dimensional, program define address into one dimensional physical address

This mapping is done with the help of a Segment Table. Each entry in the segment table, has a segment base, and a segment limit.

The Segment base contains the starting physical address, where these segments resides in the memory, and the segment limit, specifies the length of the segment.

Q Consider following segment Table,

Seg	base	length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are physical add. for following logical address?

(a) ⟨0, 430⟩ → 430 < 600

Phy add = 219 + 430

= 649

Spiral

Date

(b) ⟨1,10⟩ → 10 < 14
Phy add = 10 + 2300 = 2310

(d) ⟨3,400⟩ → 400 < 580
Phy add = 400 + 1327 = 1727

(c) ⟨2,500⟩ → 500 ≮ 100
↳ Trap: addressing error

(e) ⟨4,112⟩ → 112 ≮ 96
↳ Trap: addressing error

Virtual Memory

The concept of virtual memory is used, when the size of a program is very Large.
It gives an illusion to the user, that the program whose size is larger, than the size of the main memory can be executed.
It's implemented using two concepts → Demand paging
→ Demand Segmentation

Demand Paging

Loading the page into memory on demand { whenever page fault occurs } is called demand paging.

- Steps for demand paging :

1) CPU is trying to refer the page, in the main memory, but the page is currently not available in the main memory. This is known as page fault

2) The program execution, will stop and signal will be sent to O.S, regarding page fault.
3) O.S will search for the required page in Secondary Memory.

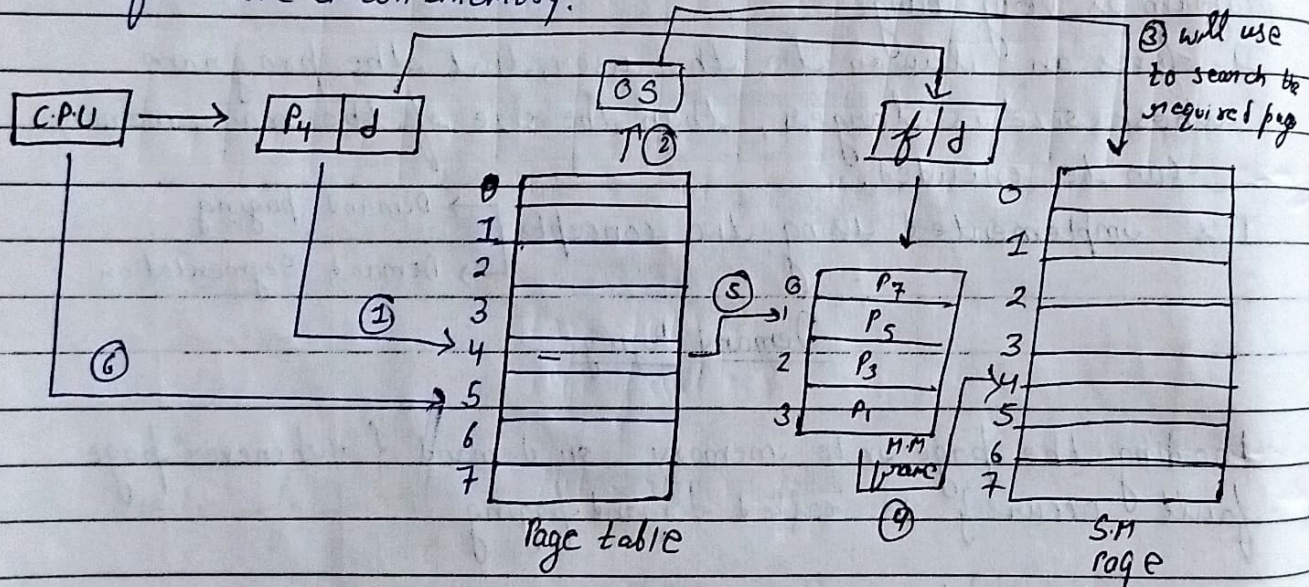
Spiral

Date

4.) The required page, will be brought from S.M to M.M. { There page replacement algo is used for the decision making of selecting the page for the replacement }

5.) The respective page table entry will be updated accordingly.

6.) Signal will be sent to C.P.U to continue the program execution and C.P.U will access the required page from the main memory.



Spiral

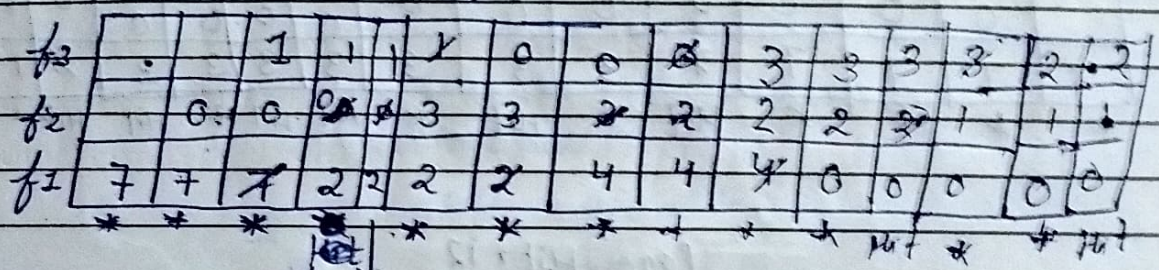
Date

Page Replacement Algorithm

1) FIFO
2) Optimal page replacement Algo
3) LRU {least recently Used}

FIFO {first in first out}

reference string : 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0



* - page fault

Page Hit = 3

Page fault / Page No = 12

Hit ration = No of hits / No of reference = 3/15 x 100 = 20 %

Hit (%) = 20 %

Page fault ratio = 12/15 x 100

Page fault (%) = 80 %

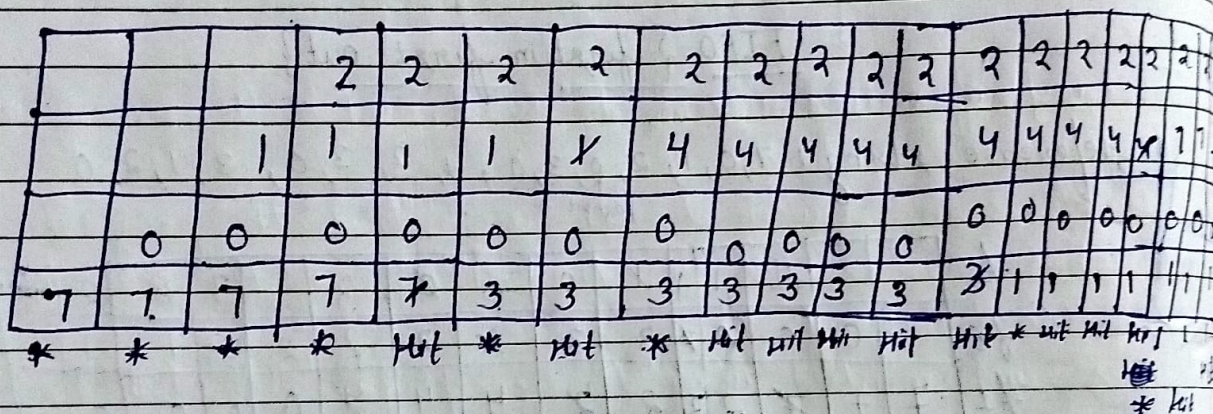
Spiral

Date

2) Optimal page replacement algo

Replace the page, which is not used in longest, dimension of time in future.

ref string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



Page Hit = 12
Page fault = 8

Hit ratio = $\frac{12}{20}$

Hit % = $\frac{12}{20} \times 100$ => 60 %.

Page fault ratio = $\frac{8}{20}$

Page fault % = $\frac{8}{20} \times 100$
=> 40 %.

Spiral

Date

3) LRU

Replace the least recently used page in the past

ref string : 7, 0, 1, 2, 6, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

f4				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
f3			1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1	1
f2		0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0
f1	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	7	7	7	7
	*	*	*	*	Hit	*	Hit	*	Hit	Hit	Hit	Hit	Hit	*	Hit	Hit	Hit	*	Hit	Hit

Page Hit = 12

Page fault = 8

Hit ratio = $\frac{12}{20}$

Hit % = 60%

Page first ratio = $\frac{8}{20}$

Page first % = 40%

Spiral