VIRTUAL MEMORY

Virtual Memory :-

Virtual mem gives an illusion to the programmer that the programs of size longer than the actual physical m/m can be executed

PAS = 100 KB

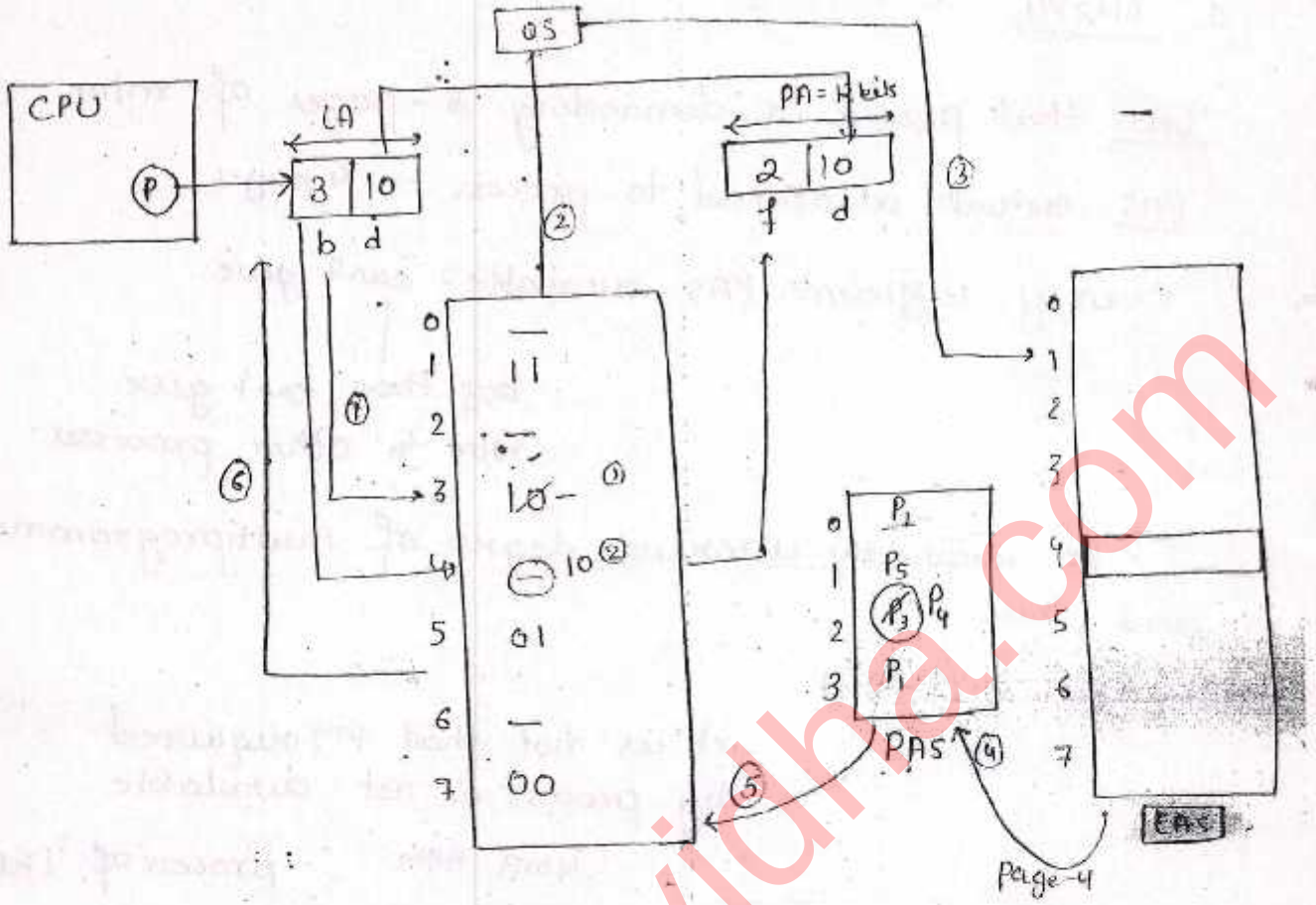
program size = 200 KB

- Still on run the program on PAS of 100KB using virtual memory

Virtual memory is implemented using :

- demand paging
  - if using paging
- demand Segmentation
  - when using segmentation

- OS generally follows demand paging



here, process is requiring 8 pages but is allocated m/m for 4 pages only.

1. LA > PA

LAS - that process is demanding 8 - pages of m/m.

PAS - actual allocated to process i.e. 4 pages

even if sufficient PAS available - can't give

bcz then can't give m/m to other processes.

i.e indirectly increase degree of multiprogramming.

and, thus

LA > PA

it is not that PM required by process is not available

4GB RAM ..... process of 1KB.

if LA = PA - no multiple - programming.

2. Why CPU always generates LA

- bcz process always require to access all the pages to complete execution.

∴ CPU generates address wrt size of the process.

prb:

if have to access P_4 - but not available in m/m.

Soln:

demand paging.

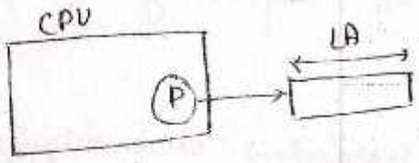
1) LA > PA:

Demand of the process will be always greater. It is not possible to satisfy the demand of the process. By allocating less m/m to the process, we are trying to execute more no. of processes.

∴ LA > PA.

ii) LA = PA
- no h/w support of paging req
Or
- simply no paging required.

2)



in order to complete the execution of the process, CPU definately requires to access all the pages of LAS. This is the reason, when CPU is generating the addr. it will generate w.r.t size of the process.

i) Page fault:

CPU is trying to access a page but the page is not available in m/m, then it is called as page fault.

ii) program execution will stop and signal will be sent the OS regarding page fault, then the OS validates the details of the process.

3) OS will search for the required page in LAS.

4) Required page will be brought from LAS to PAS.

problem:- no space (C means in space allocated to that particular process - no free space)

Soln:- page replacement algorithm

The page replacement algorithm will be used for the decision making of replacing the page in the PAS.

5) The page fault will be updated accordingly.
- if page table not updated then will again result in page fault

6) Signal will be sent to CPU to continue the program exec^n, then the CPU will access the required page in the m/m and will continue the program execution.

- bcz of this process, program execution will stop for some time - can't be observed by user.

Demand Paging

- loading the page into m/m form LAS to PAS on demand, whenever the page fault occurs.

The time taken to service a page fault is called as page fault service time.

page fault service time includes the time required to perform all the above 6 steps.

PFST = S

MMAT = M

page fault rate = P.

then, Effective memory access time (EMAT):

$$EMAT = P*S + (1-P)*M$$

not 2M bcz page table access time negligible

S >> M.

Que- Consider a system with page fault service time be 200 ns and the page hit ratio is 95% and MMAT is 20 nsec what is EMAT.

S = 200

1-p = 0.95

p = 0.05

EMAT = 0.05 x 200 + 0.95 x 20

= 10 + 19

= 29 nsec.

Que. Suppose if an inst^n takes 'i' usec and an additional 'j' usec. If page fault occurs what is the effective inst^n access time, if page fault occurs on an avg. for every k inst^n.

$$p = \frac{1}{k}$$

$$s = i + j$$

$$m = i$$

$$= \frac{1}{k}(i+j) + \left(1 - \frac{1}{k}\right) i$$

$$= \frac{i + j + ki - i}{k}$$

$$= \boxed{i + \frac{j}{k}}$$

Que. Let the page fault service time be 10ns with avg. m/m access time be 20 nsec If 1 page fault is generated for every 10^6 m/m access then what is EMAT.

$$s = 10^{-2} \text{ sec}$$

$$m = 20 \times 10^{-9} \text{ sec}$$

$$p = \frac{1}{10^6} = 10^{-6}$$

$$\text{EMAT} = 10^{-6} \times 10^{-2} + \left(1 - \frac{1}{10^6}\right) 20 \times 10^{-9}$$

$$= 10^{-8} + 20 \times 10^{-9}$$

$$= 30 \text{ nsec.}$$

Que. Consider a demand paging environment where if it takes 8msec. to service a page fault if either an empty frame is available or the replaced page is not to be modified and if it takes 20msec. to service a page fault if it is modified MMAT = 1msec & further assume page to be replaced is modified 70% of the time. Then what is max. acceptable page fault rate to get an EMAT not more than 2msec.

$S_1 = 8$ msec.

$S_2 = 20$ msec.

$m = 1$ msec

$$2 > p(0.3 \times 8 + 0.7 \times 20) + (1-p)(1)$$

$$p(2.4 + 14) + (1-p) < 2$$

$$16.4p - p < 1$$

$$p < \frac{1}{15.4}$$

$$= 0.064$$

Que: Consider a system using demand paging environmental table, the page fault service time = 200 msec. and MMAT = 10 msec. The TLB is added to improve the performance 80% references are found in TLB and that of remaining 10% cause page fault. TLB access time & page table access time both are negligible. What is EMAT.

$x(c+m) + (1-x)[c+ps+(1-p)m]$

$= 0.80(10) + 0.20(0.1 * 200 + 0.9 * 10)$

$= 8 + 0.2(20+9)$

$= 8 + 0.2(29)$

$= 8 + 5.8 \quad = 13.8$ msec

TLB hit means- page fault can't occur.

Page Replacement Algo

reference string:

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

– these are the page no. referred by the process.

1. First in First Out: (FIFO)

assuming that a process can have 4 frames in PAS.

instruction set archi. is used to decide the allocating of frames to the process.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
		1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	7	7	7
7	7	7	7	7	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2
*	*	*	*		*		*			*			*	*			*		

page faults = 10

= 30 msec.

taking 3 frames:

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

[15]

Belady's anamdy:

- by increasing # frames to the process no. of page faults must decrease.

but here are increasing

∴ this is Belady's anamdy

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

3-frames - 9

4 frames - 10

2. Optimal page replacement:

In the event of page fault, replace the page which is NOT used for the longest duration of time in the future

Using 4 frames-

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
		1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	7	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1	1
*	*	*	*		*		*					*				*			

follow FIFO

page faults = 6

for 3 frames:-

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7	
*	*	*	*		*		*			*			*				*		

page faults = 9

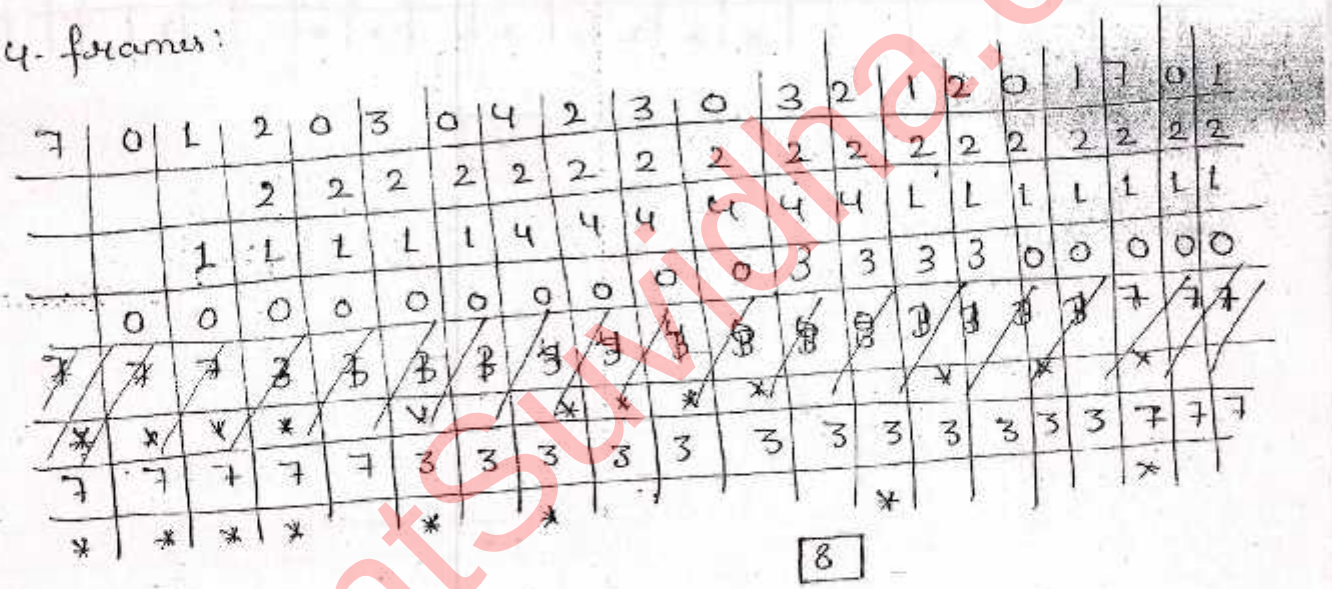
3. LRU page replacement:

(least recently used)

in the event of page fault, replace the page which is least recently used.

12

4 frames:



8

Q21

record #	page #
0-99	0
100-199	1
200-299	2
300-399	3
400-499	4
500-599	5

1 2 4 4 5 5 5 1 2 2 2

100 200 400 499 570 530 560 120 220 240 260

3 3

320 370

7

Most Recently Used

(MRU)

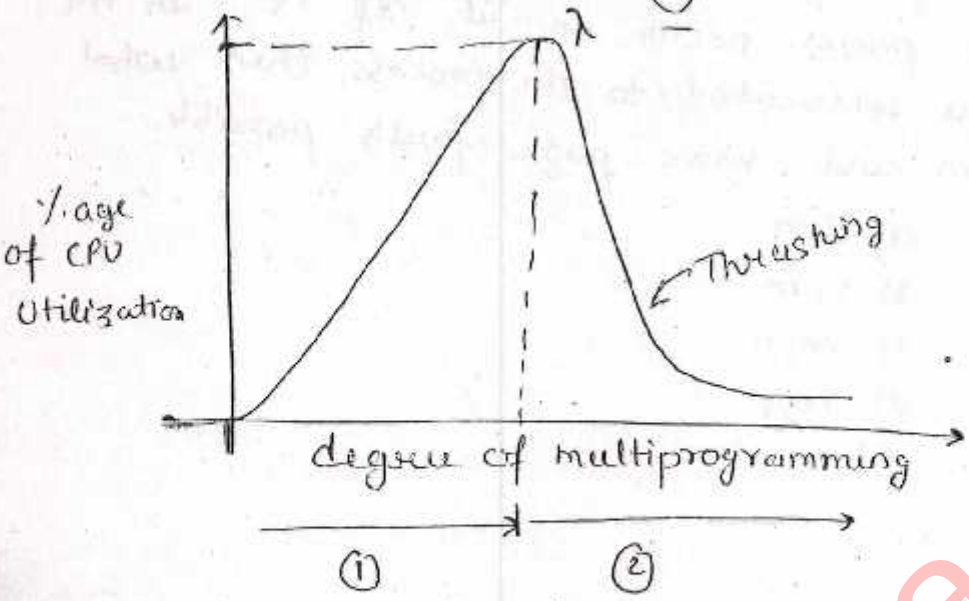
in event of page fault, replace the page which is most recently used.

7	0	1	2	3	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			2	2	2	2	2	2	2	3	0	3	2	2	2	0	0	0	0	0
		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	3	0	4	4	4	4	4	4	4	4	4	4	4	4	4
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
*	*	*	*			*	*	*		*	*	*	*			*				

Ques- Consider the system having a process with length of its LA. has 'n' pages and 'm' unique pages occur in it. If 's' in the # frames allocated to the process, then what is min and max page faults possible.

a) s, n
b) s, m
c) m, n
d) m, s

Thrashing



free frames = 300

① { # process in system = 100
each process, P_i = 3 frames

② { # process = 300
frames/process = 1 frame
i.e. nearly every page is a page-fault.

→ every page reference will be page fault and CPU remain busy in dealing with page faults → thrashing.

- In the initial, degree of multi-programming, the CPU utilization is very high and the system resources are utilized 100%.

more, if further increase the degree of multiprogramming, the CPU Utilization will drastically fall down & the time taken to complete the exec". of the process will increase & the system will spend more time only in page replacement. ——thrasing.

Causes of thrasing:

1) High degree of multi-programming
2) Lack of frames (m/m)

Recovery thrashing:

1) Don't allow the system to go into thrashing and instruct the long-term scheduler not to bring the processes into the m/m after the point of 'λ'

2) Allow the system to go into thrashing & then instruct the mid-term scheduler to suspend some of the processes so that the system will recover from thrashing.

Qu. Consider a system with below statistics:

a) the paging disk - 90%

b) the CPU utilization - 10%

time spent on page replacement.

which of the below factors will improve the CPU utilization.

1. Install the bigger disk
2. Install the faster CPU
3. Increase degree of multi-programming
4. Decrease degree of multi-programming
5. Install more m/m.
6. Decrease page size.
7. Increase page size.

a) ⇒ thrashing is there

1- No

2- No

3- No

4- Yes

5- Yes

6- No - bcz pages required by process will increase.

7 - pages required will be less.