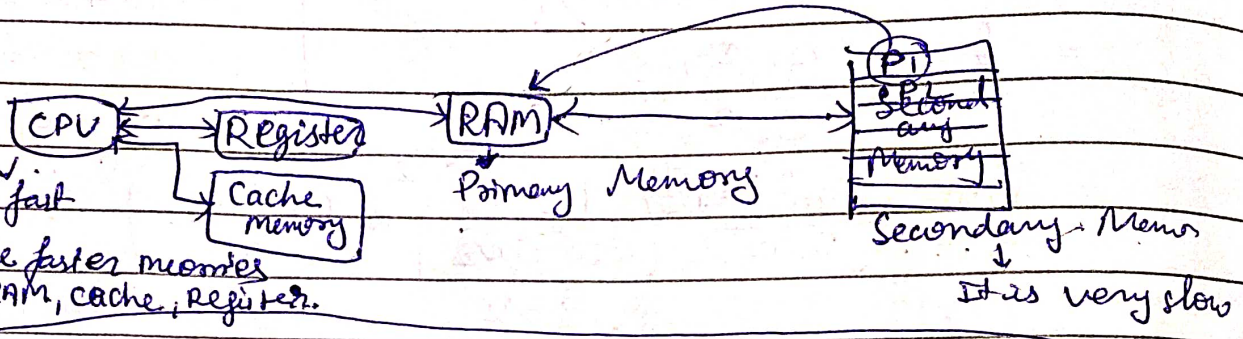
Date: 07/01/23 Page :

Unit-3

Memory Management → method or functionality to manage various kind of memories.

We have RAM, hard Hard disk and registers, method of managing memory resource in a more and more efficient manner.



Multiprogramming → Bring maximum no. of programs or processes from secondary memory in the RAM, in the main memory So that CPU can execute them.

* Higher the Degree of Multiprogramming higher the utilization of CPU.

Each process have two things/ or do only two jobs
- CPU → execute by CPU and at I/O operation CPU become idle or free at that time.
- I/O → some time process request I/O at this time CPU blocked or free the process and process start using I/O eg. like Scanning or printing files, read file from hard disk.

Degree of Multiprogramming: is keep more and more no of processes in the RAM and efficiency of the system is high, means when utilization of CPU is very high the performance of System will automatically increases.

* Degree of Multiprogramming को increase करने के लिए RAM का Size increase करना पड़ेगा।

RAM	Process
4MB	4MB

no. of process = $\frac{4}{4} = 1$ (lol)
K = I/O operation = 70%
CPU utilization = $1 - K$
$= 1 - 70\%$
$= 30\%$

RAM	Process
8MB	4MB

no. of process = $\frac{8}{4} = 2$
K = I/O operation = 70%
CPU Utilization = $1 - K^2$
$= 1 - (.7)^2$
$\approx$ approx 76%

RAM	Process
16 MB	4MB

no. of process = $\frac{16}{4} = 4$
K = I/O operation = 70%
CPU Utilization = $1 - K^4$
$\approx$ approximately 93%

allocation, deallocation, security that is done by the OS in the memory management.

Date: / / Page :

"Memory Management Techniques"

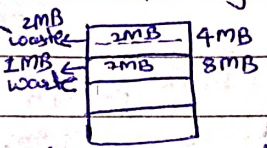
- Contiguous
  - static
    - Fixed Partition (in 1960's, ↓ use main frame ↑ server)
  - Dynamic
    - Variable Partition
- Non-contiguous
  - Paging
  - Multilevel Paging
  - Inverted Paging
  - Segmentation
  - Segmented Paging

(1MB / 2)

Fixed Partitioning → • No. of Partitions are fixed.
• Size of each partition may or may not same
• Contiguous allocation spanning is not allowed.

occurs when memory is distributed into fixed-size blocks

problems
① Internal fragmentation
② Limit in process size
③ Degree of multiprogramming is limited.



④ External fragmentation → Availability of memory in different slots and combination means addition of all the space. Is equivalent to the available size means the size of the new process is less than the available space.

Variable Partitioning or **Dynamic Partitioning** →
- run time memory allocation
- there is no internal fragmentation
- No limit on no. of processes
- No limitation on process size

→ advantages

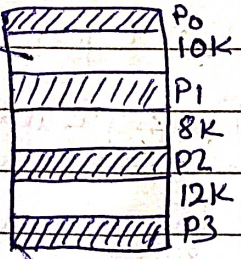
dis adv.
- External fragmentation → disadvantages
  - this can be corrected by compaction.
  - Allocation / Deallocation is complex as compared to fixed partition (used to remove the external frag.)

First fit → Allocate the first hole that is big enough. Simple, Fast

Next-fit → Same as first fit but start search always from last allocated hole.

Best fit → Allocate the smallest hole that is best big enough.
Less internal fragmentation, Search the entire list.
dis adv. → Slow.

Worst fit → Allocate the largest hole. Maximum internal fragmentation.
dis adv. → Slow.



Paging vs Swapping

Date: / / Page :

Paging	Swapping
① It is a technique of memory allocation	① It is the procedure of copying out the entire process.
② It occurs when some part of the process is transferred.	② It occurs when whole process is transfered to the disk.
③ It is a concept used in non-contiguous memory management.	③ It can be performed without any memory management.
④ The only active process can perform paging.	④ Swapping is done by inactive processes.
⑤ Paging is faster than swapping	⑤ Swapping is slower than paging due to copies the whole information.
⑥ In this different non-contiguous blocks of memory are assigned a fixed size called frame or pages.	⑥ In this process is swapped temporarily from main memory to secondary memory.
⑦ There are some processes in the main memory during paging.	⑦ There are many processes in the main memory during swapping.

Paging vs Segmentation

we break a process address space into blocks known as pages. Pages are blocks of fixed size. Paging	we break a process address space into blocks known as sections. Sections are blocks of varying size. Segmentation
① In paging the program is divided into fixed or mounted size pages.	① In Segmentation, the process is divided into variable size sections.
② Page size is determined by hardware.	② the section size is given by the user.
③ It is faster than segmentation	③ It is slow than paging
④ Paging could result in internal fragmentation	④ Segmentation could result in external fragmentation.
⑤ Paging results in a less efficient system	⑤ Segmentation results in a more efficient system.
⑥ The size of the pages need always be equal to the size of frames	⑥ There is no limitations on the size of segments.
⑦ Paging comprises a page table that encloses the base address of every page.	⑦ Segmentation also comprises the segment table which encloses segment number and segment offset.
⑧ In paging, the logical address is split into a page no. and page offset.	⑧ the logical address split into section no. and section offset.
⑨ Page table stores the page data	⑨ Segmentation table stores the segmentation data.

Physical address → directly related to main memory.
Logical address are generated by CPU
CPU works on logical address.
Memory is byte addressable.

16B (0-15) → 4 bits
8B (0-7) → 3 bits

Date: / / Page:

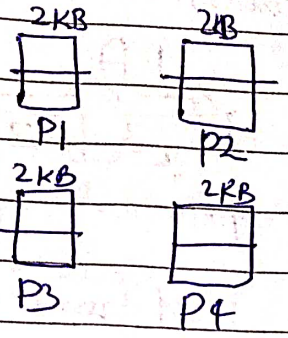
Size of Page table = No. of Page × No. of frame

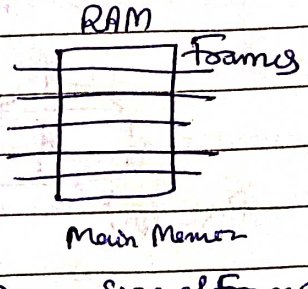
No. of entries in Page table = No. of Pages in a Process

Non-Contigous Mem. Alloc.
↓
used to remove external fragmentation.

P1	0
P1	1
P2	2
P2	3
P3	4
P3	5
P4	6
P4	7

8KB

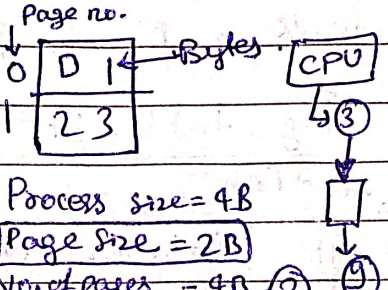




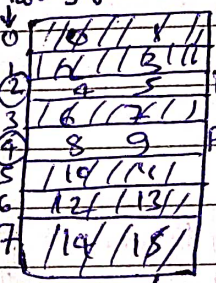
Size of Page = Size of frame
Page size = Frame Size

P1
P1
P5
P5
P3
P3
P5
P5

Memory are byte addressable
CPU always demands memory in byte for execution



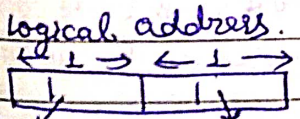
Process size = 4B
Page Size = 2B
No. of pages = 4B/2B = 2

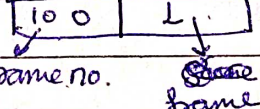


Mapping is getting done by the Memory management Unit (MMU).

MMU converts the address generated by the CPU into absolute address.
This conversion MMU uses Page Table.
Every process has its own page Table.
CPU always works on logical address.

Logical address.

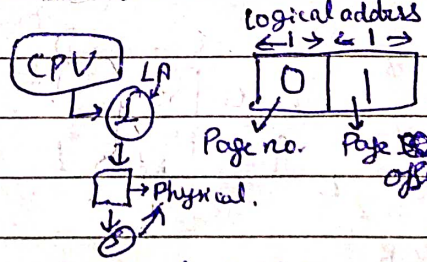




Page table of P1

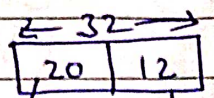
0	f2
1	f4

Page offset = frame offset

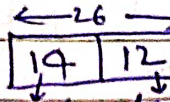


Q. Logical Address Space (LAS) = 4GB , Physical Address Space (PAS) = 64MB
Page Size = 4 KB.
Find out → No. of Pages = ? , No. of frames = ? , No. of entries in Page table, Size of Page table.

→ Logical Address = $2^2 \times 2^{30} = 2^{32}$



Page Size = $2^2 \times 2^{10} = 2^{12}$
Physical Address = $2^6 \times 2^{20} = 2^{26}$



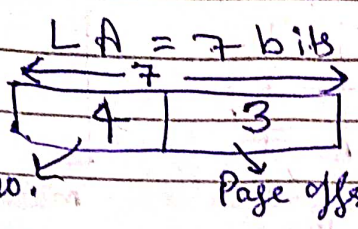
No. of frames → 2^{14}
No. of entries in Page table = 2^{20}
Size of page table → $2^{20} \times 14$ bits

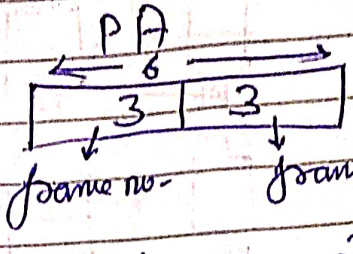
$2^1 = 2$	$2^{20} = 1M$
$2^2 = 4$	
$2^3 = 8$	
$2^4 = 16$	$2^{30} = 1G$
$2^5 = 32$	$2^{40} = 1T$
$2^6 = 64$	$2^{10} = 1K$
$2^7 = 128$	
$2^8 = 256$	

Date: / / Page:

Q LA = 7 bits, PA = 6 bits, Page size = 8 words/bytes then calculate no. of pages & no. of frames.

→





no. of pages = $2^4 = 16$

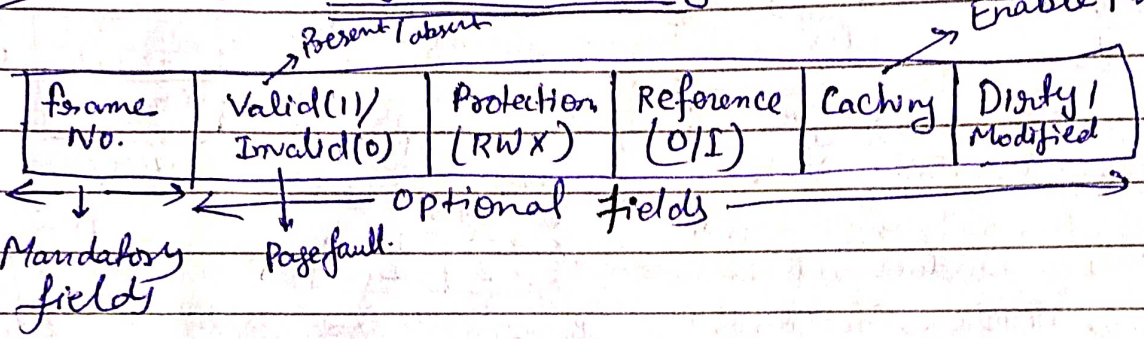
no. of frame = $2^3 = 8$

Size of Page table = $2^4 \times 2^3 = 128$ bits.

No. of entries in Page table = No. of pages in the Process = 16 Ans

RWX
Read Write Execute

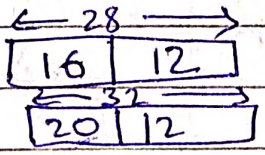
Page table Entry.



Multilevel Paging → when my original page table size is so big that it did not fit in my frame, then we divided those page table in further pages. and we stored those page table in the outer page table.

PA = $2^8 \times 2^{20} = 2^{28}$

LA = $2^2 \times 2^{30} = 2^{32}$

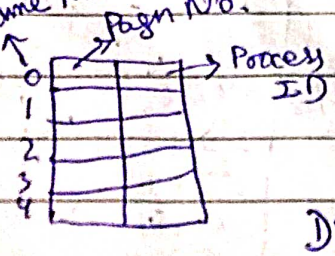


Frame Size = $2^2 \times 2^{10} = 2^{12}$

Page table Entry = 2B

Page table Size = $2^{20} \times 2 = 2^{21} = 2$ MB.

Inverted Paging → Each process has its own page table and Page table will be in main memory.



Q To avoid the wastage of memory to keeping the global page table for each process in main memory.

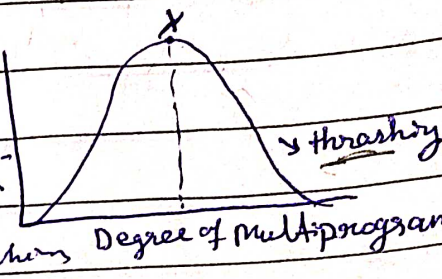
Disadvantage → Searching time is very high.

Thrashing

Date: / / Page :

Thrashing when the page fault and swapping happens very frequently at a higher rate and then the OS has to spend more time swapping the pages. This reduced the CPU utilization.

eg. let imagine we have 10 process in secondary memory and we perform the degree of multiprogramming to transfer the process (~~each process of 1st page of each process~~) is bringing in the main memory and CPU start the process CPU demand P1 1st page and complete it then after CPU demand P1 2nd page but this page is not in the main memory then page fault occurs, OS searching the table and thrashing occurs.



Page fault → Page fault occurs when a program attempts to access data or code that is in its address space, but is not currently located in the System RAM.

OS finds that a page fault has occured and tries to find out which virtual page is needed.

Segmentation

Segment No

	Base Addr	Size
0	3300	200
1	1800	400
2	2700	600
3	2300	400
4	2200	100

	Base Addr	Size
0	3300	200
1	1800	400
2	2700	600
3	2300	400
4		
5		