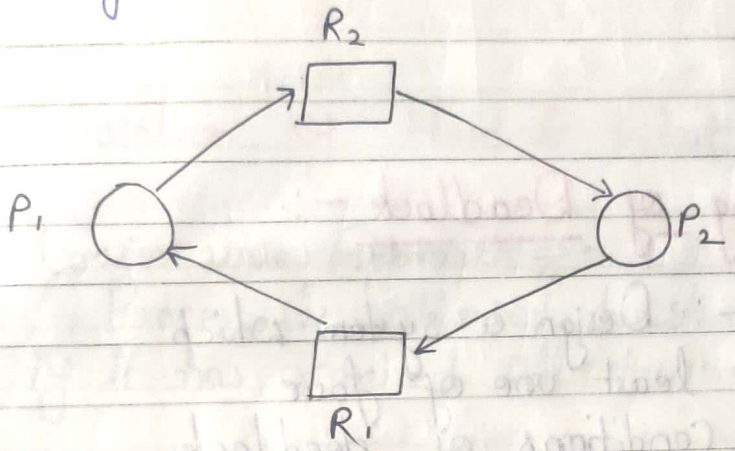
DEADLOCK.

In a multiprogramming system, a no. of process compete for a limited no. of resources, & if it needs isn't available at that instance then process enters ns. into waiting state.



If a process is unable to change its waiting state indefinitely because the requested resources are held by another waiting process, then the system is said to be in deadlock.

Necessary Conditions for Deadlock -:

1) **Mutual Exclusion** -: At least one ~~system~~ resource type in the system which can be used in non-sharable mode. (Eg -: Printer)

2) **Hold & Wait** -: A process is currently holding at least one resource & requesting additional resources which are being held by other processes.

3) Non-Preemption —:
A resource can not be preempted from a process by any other process resource will be removed only by the process holding it (voluntarily)

4) Circular Wait —:
A cycle.

* Handelling of Deadlock —:

(1) Prevention —: Design a system which violate at least one of four necessary conditions of deadlock & ensure independence from deadlock.

(2) Avoidance —: System maintain a set of data which it takes a decision weather to entertain a new request or not, to be in safe state.

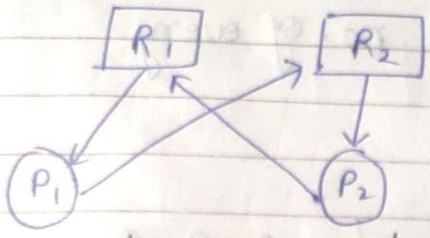
(3) Detection & Recovery —: Wait until a deadlock occur and once detected the recover from it.

(4) Ostrich Algo —: Ignorance

Deadlock Prevention —:

APPU DATE _____ PAGE _____

Violation of Mutual Exclusion —:



→ We cannot make a resource sharable.
→ Mutual exclusion can't be handeled & it depends upon the h/w conditions.
→ Only one process use the CPU at a time.

Violation of Hold & Wait —:

① Conservative Approach ⇒:
Process is allowed to start execution if & only if it has acquired all the resources.

② Do not hold ⇒:
Process will acquire only desired resource, but before making any fresh request it ~~may~~ must release all the resources that it currently holds.

③ Wait Timeouts ⇒:
Process release all the resources in a timeout.

Violation of No-Preemption —:

✓ forceful preemption.

→ This method may be used by high priority or system process.
→ Process which are in waiting state must be selected as a victim instead of process in running state.

Violation of Circular Wait :-

→ → By giving a natural no. of every resource.

→ $f: N \rightarrow R$

→

P_1	P_2
R_1	R_1
R_2	R_2

←

P_1	P_2
R_1	R_2
R_2	R_1

R_1, R_1, R_1

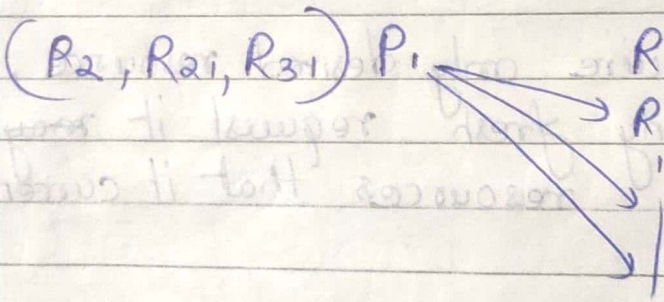
R_1, R_2, R_3

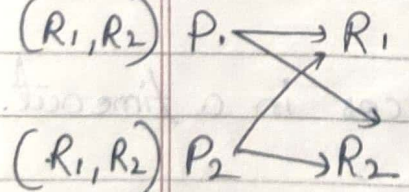
→

→ → Allow the process to i.e. only in the

→ increasing or decreasing order of the resource no.

DAC – Direct Acyclic Graph.



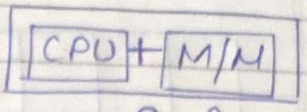


If a process require a lesser no. than it must be first release all the resource largest than required no.

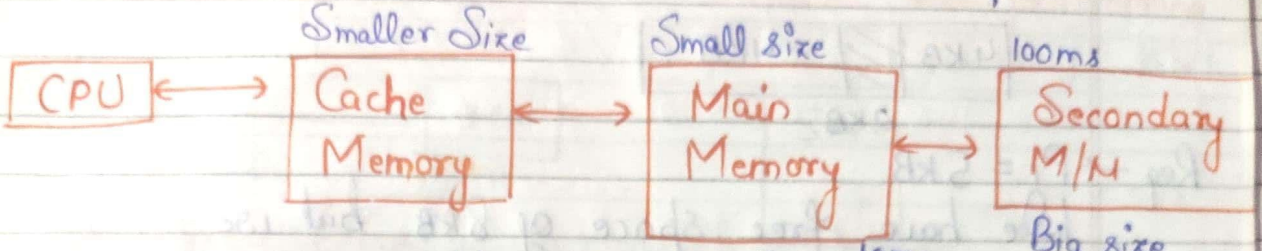
Basics of M/M management –:

1. Size ↑
2. Access Time ↓
3. Per unit cost ↓

} We can't have them all in one.



Computer



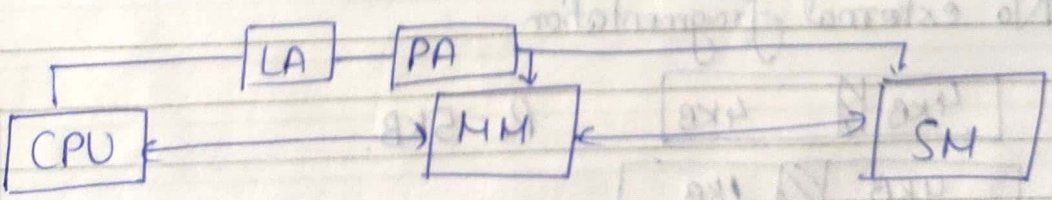
Heirarchy of M/m.

✓ Hit Ratio –:. Data we want we get how many times.
CPU interact with main memory.

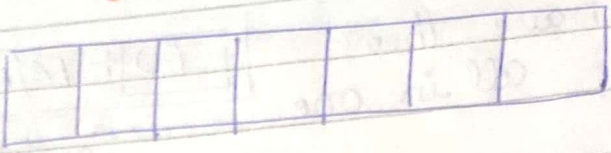
hit % = 90 %
= .9(10) + .1(10+100)
= 20 ms //

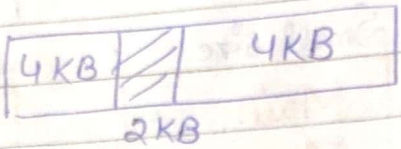
✓ Locality of Reference – High access time.

Logical address is used to access secondary memory.
CPU generates LA.
Physical address access Main m/m.
LA needs to be changed into PA by O.S.



Contagious Policy → Allocation in Sequence

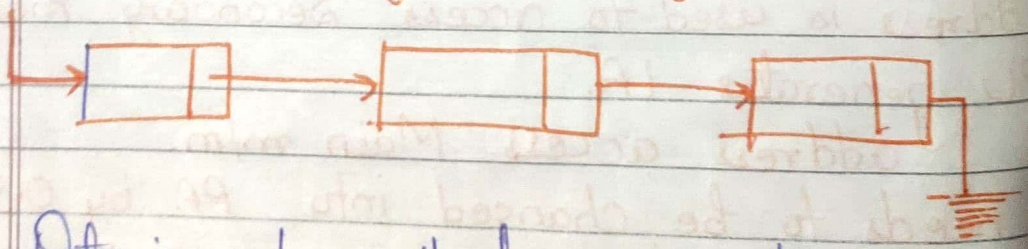




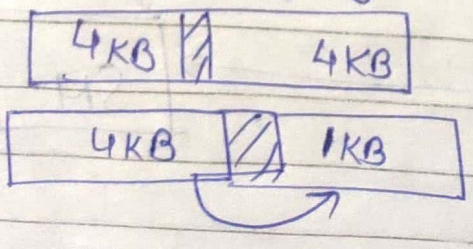
Req → P_1 = 5KB.
We have free space of 8KB but we cannot allocate P_1.
This is called external fragments.

Contigous m/m allocation locates in sequences, Address translation & access is easy. We need to know Base address only.
Always suffer from external fragmentat.

Non-Contageous Policy →



✓ Data in m/m can't be accessed in middle
✓ No external fragmentation



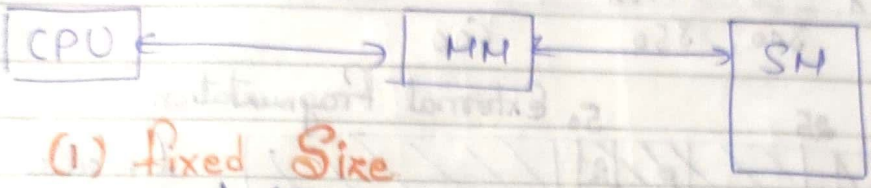
P_1 = 5KB.

APPU DATE _____
PAGE _____

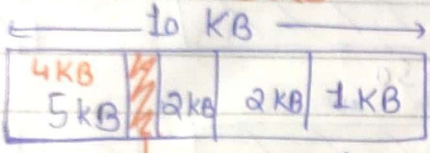
Contigious M/M

Fixed size partitioning

Variable size partitioning



(1) Fixed Size



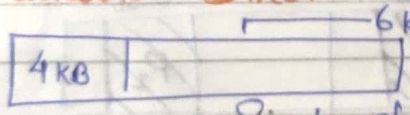
1KB is lost, that is internal fragmentation

$P_1 = 4KB$

- One partition have one process.
- We cannot reneed a partition.
- Internal partition.

Used nowadays as Paging. Still

(2) Variable Size.

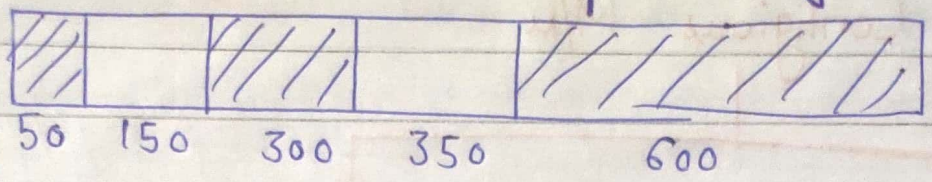


$P_1 = 4KB$ Single chunk of m/m.

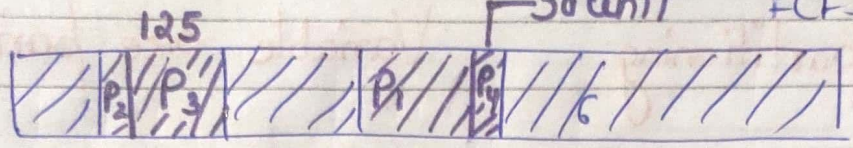
- No predefined partition
- No internal fragmentation.

Q) Variable sized partitioning

$P_1 = 300$
$P_2 = 25$
$P_3 = 125$
$P_4 = 50$



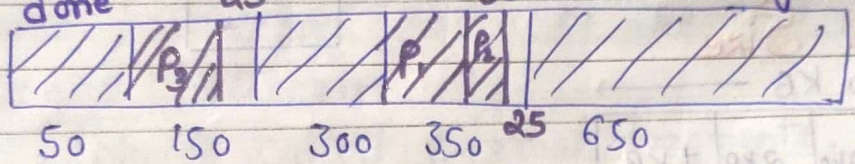
First fit — 50 unit FCFS



$P_4 = 50$ Can't be done

External fragmentation

Best fit — smallest block fills first

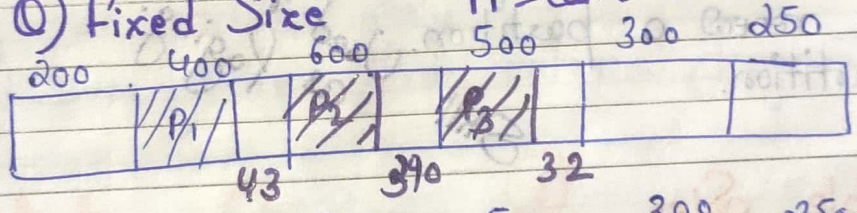


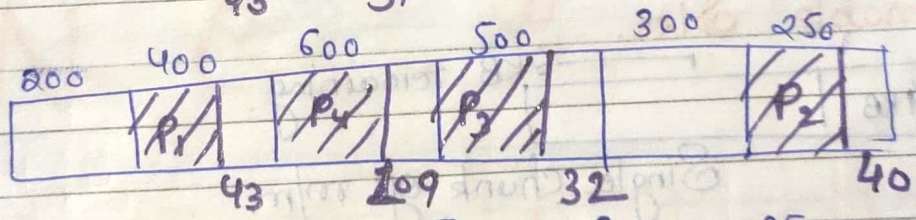
Worst fit — largest one gets filled

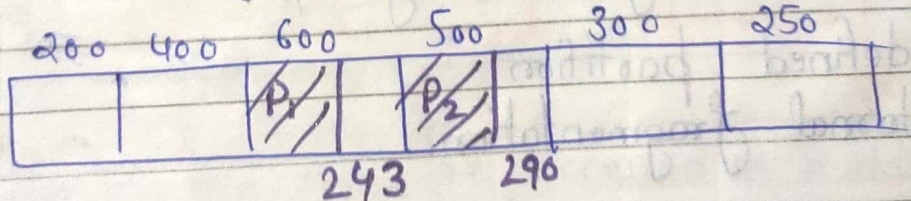
Q) Fixed Size

$P_1 = 357$
$P_2 = 210$
$P_3 = 468$
$P_4 = 491$

P_1 = Can't be done



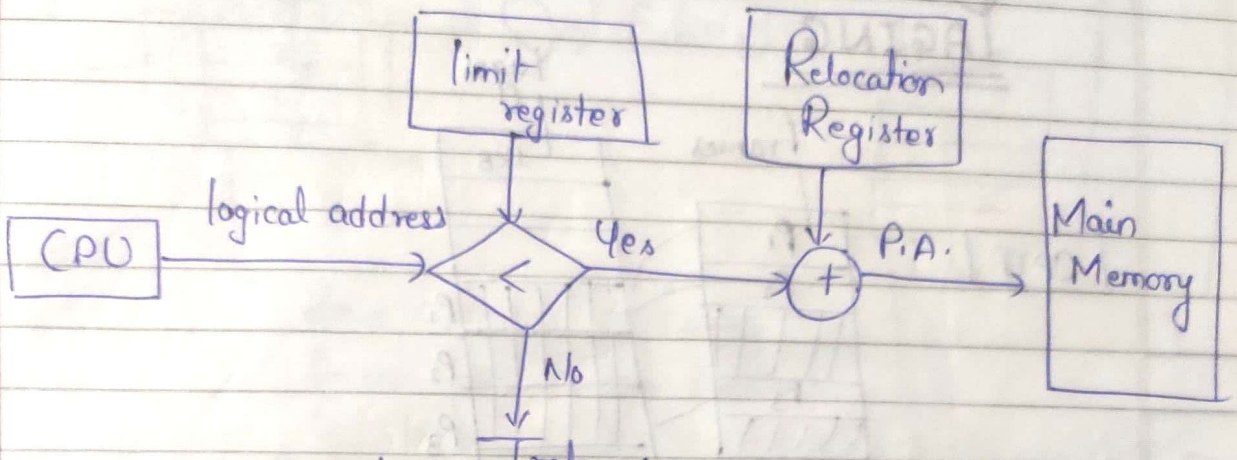




P_3 & P_4 can't be done.

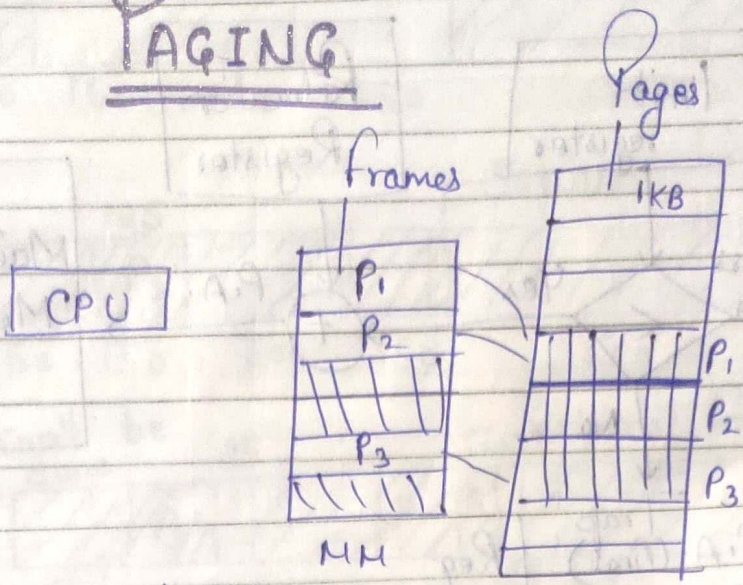
APPU DATE _____
PAGE _____

Address Translation -:



	LR	RR	f.A. (final)	Req	
P_0	500	1200	1650	450	
P_1	275	550		300	✓ can't be satisfied
P_2	212	880	1090	210	
P_3	420	1400		450	✓ can't be satisfied
P_4	118	200	280	80	

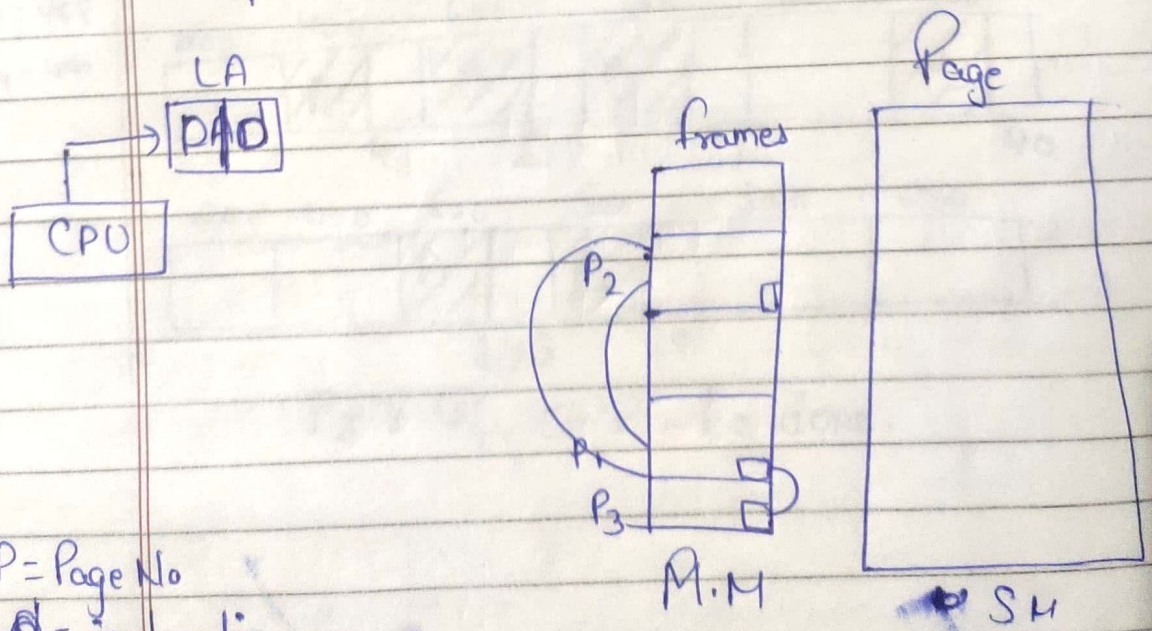
PAGING



Non-Contagious allocation

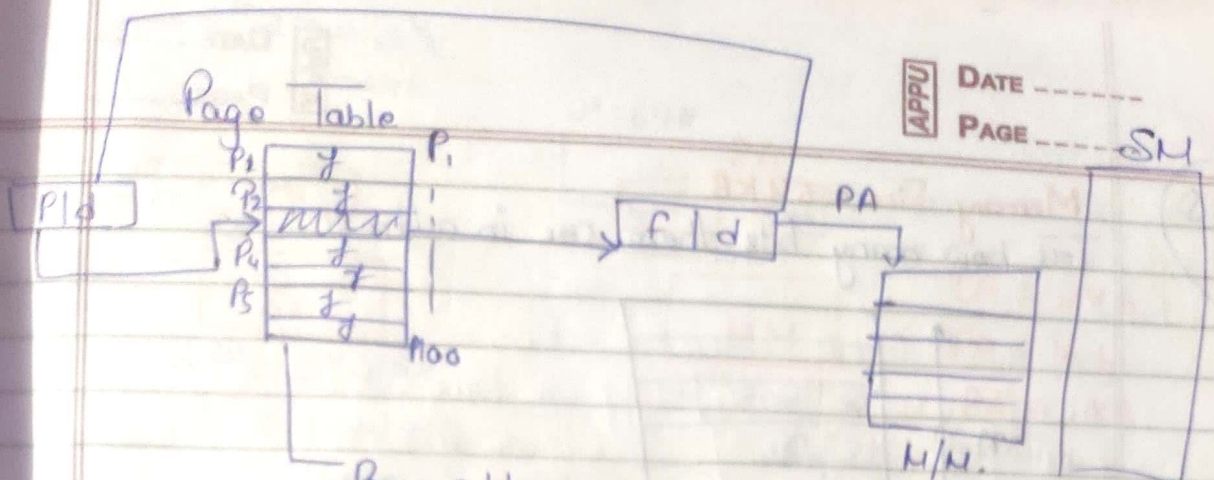
SM is divided into equal sized partitions that is called pages.

~~Hai~~ for 1KB page = 1KB frames
Main m/m divides into frames.
Independence from external fragmentation



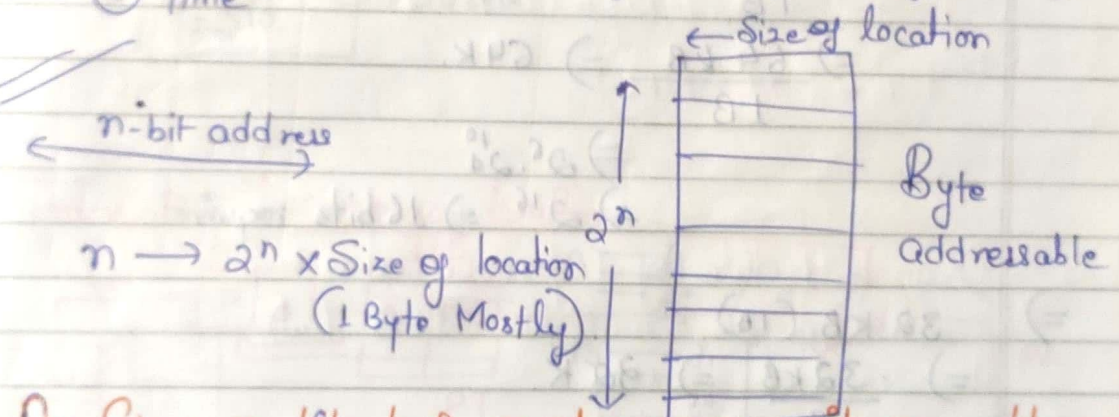
P = Page No
d = instruction offset

We can access P2 by going to P1



Base address of every frame your page table

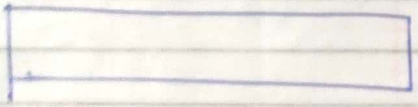
Disc —: ① Fixed Size partitioning (Internal Fragmentation)
② Time



$n \longrightarrow 2^n \times$ Size of location (1 Byte Mostly)

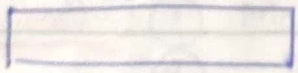
for Given n bit how much memory it can address

14 Bit



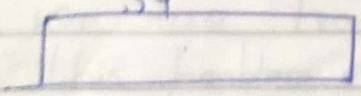
$= 2^{14} \times 1$
$= 2^4 2^{10}$ 1B
$= 16$ KB $[2^{10} = 1K]$

22



$= 2^{22} \times 2B$
$= 2^2 2^{10} 2^{10} \times 2B$
$= 8$ MB $[2^{20} = 1M.]$

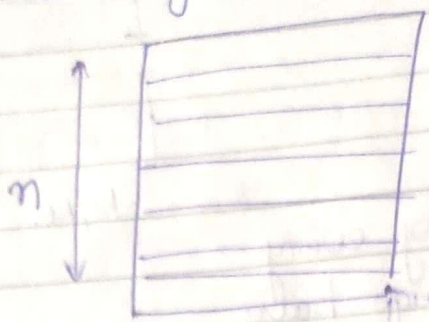
34



$= 2^{34}$
$2^{30} \times 2^4 \times 1$
$\Rightarrow$ 16GB. $[2^{30} = 1G]$

$2^{40} = 1T$ & $2^{50} = 1P.$

Q) Memory Size = 64 KB.
Tell how many bits there are in address



M.S. = n × 1B.

$n \Rightarrow \frac{MS}{1B.}$

$\Rightarrow \frac{64\ KB}{1B.} \Rightarrow 64K.$

$\Rightarrow 2^6 \cdot 2^{10}$

$\Rightarrow 2^{16} \Rightarrow$ 16 bits require

$\Rightarrow$ 32 KB (1B).

$\Rightarrow \frac{32\ KB}{1B} \Rightarrow 32K.$

$\Rightarrow 2^5 \cdot 2^{10}$

$\Rightarrow 2^{15} \Rightarrow$ 15 bits.

$\Rightarrow$ for 256 MB (1B).

$\frac{256\ MB}{1B} \Rightarrow 256M$

$\Rightarrow 2^8 \cdot 2^{20} \Rightarrow 2^{28} \Rightarrow$ 28 bits

$\Rightarrow$ for 16 GB (4B)

$\frac{16\ GB}{4B} \Rightarrow 4G.$

$\Rightarrow 4\ 2^2 \cdot 2^{30}$

$\Rightarrow 2^{32}$

$\Rightarrow$ 32 bits

$2^4 \cdot 2^{10} = 2^4 K$.
$= 64K$.

LA

14 P	10 d

24

SM

$2^{24} = 2^4 \cdot 2^{20}$
$= 16M \times 1B$
$= 16MB$.

$2^6 = 64$

PA

6 f	d 10b

16B

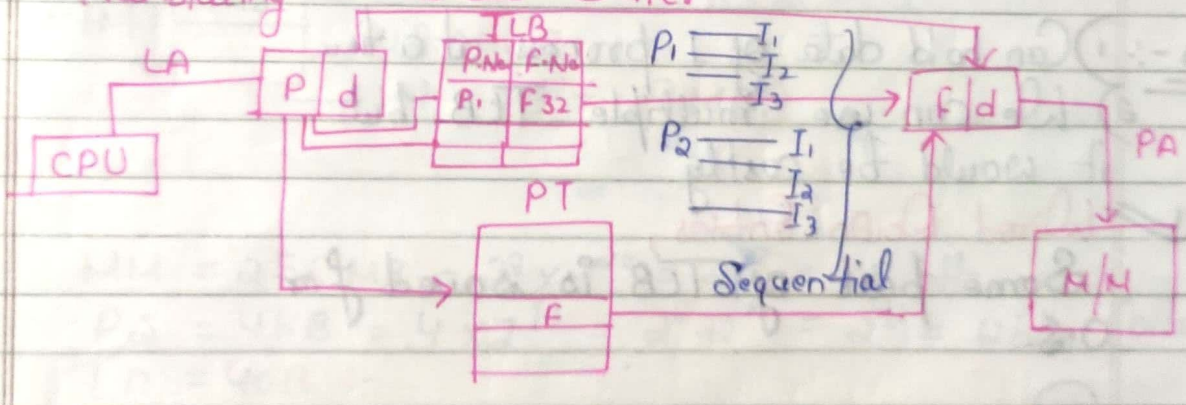
MM

$2^{16} = 2^{10} \cdot 2^6$
$= 32K \times 1B$
$= 32KB$

LA = 24 b
PA = 16 b
PS = 1 KB Page Size

$\frac{1KB}{1B} = 2^{10}$. $\Rightarrow$ 10 bits

* Translating Lookahead Buffer



Every process have specific page table.

Disad-: (1) Main Memory Access (for Page table)
(2) " " " (for Instruction)

Avg. time $\Rightarrow$ 2 x MM time

We need to access same page again & again we use Translation Lookahead Buffer. for this problem. Main m/m is accessed only once in a TLB.

Access times — :

MM = 400 ms

TLB = 50 ns

hit Ratio = 90%

Now, Without TLB = 400 × 2 = 800

⇒ 0.9 [50 ns + 400 ns] (If we have a TLB hit)
+ .1 [50 + 400 + 400] (If we miss TLB hit)

⇒ .9 [450] + [850] .1

⇒ 405 + 85 ⇒ 490 ns.

Avg. Access Time = (Hit Ratio) [TLB + MM] + (1 − Hit Ratio) (TLB + MM + MM).

Dis -:
1) Can hold data of 1 process at a time
2) We can use multiple TLB but it would be costly.

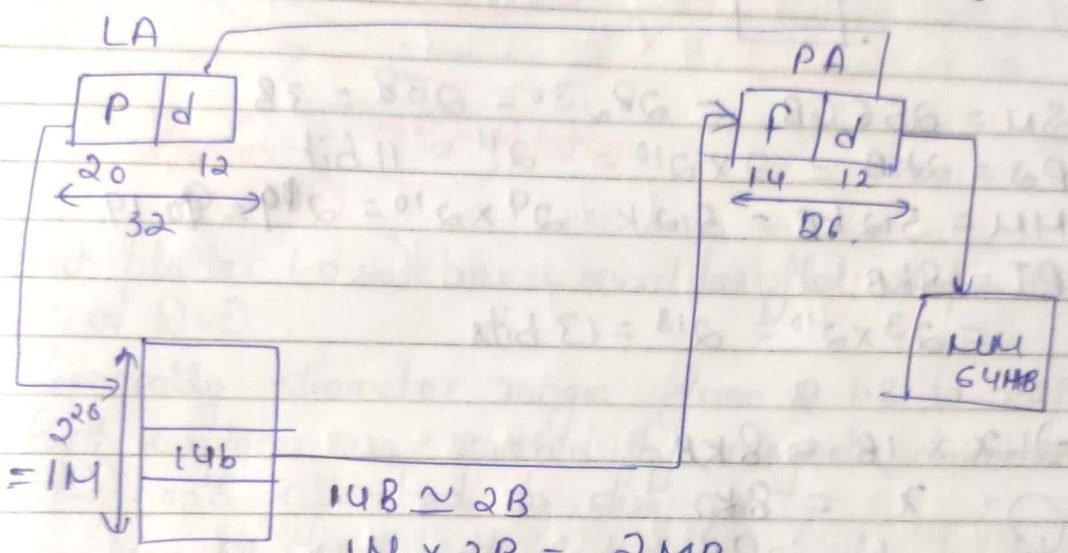
✓ Wired Down Entries,
Some part of TLB is saved for O.S.

Page table is a Metadata.

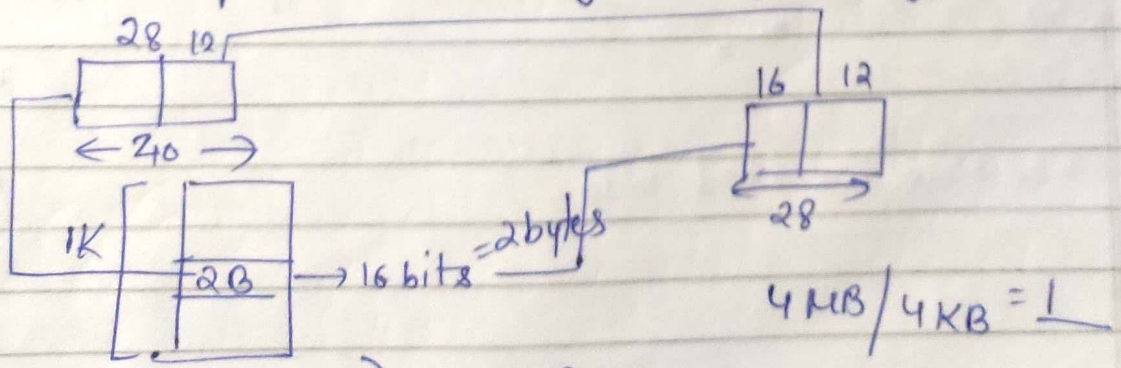
Size -:

Q) MM = 64 MB = ~~64 x~~ $2^6 \times 2^{20} = 2^{26}$ location = 26 bits
LA = 32 b
PS = 4 Kb = $2^2 \times 2^{10} = 2^{12}$ = 12 bits page table in
Total space wasted in continuing the main m/m?

✓ Size of Page Table = Entry Size x No. of entries

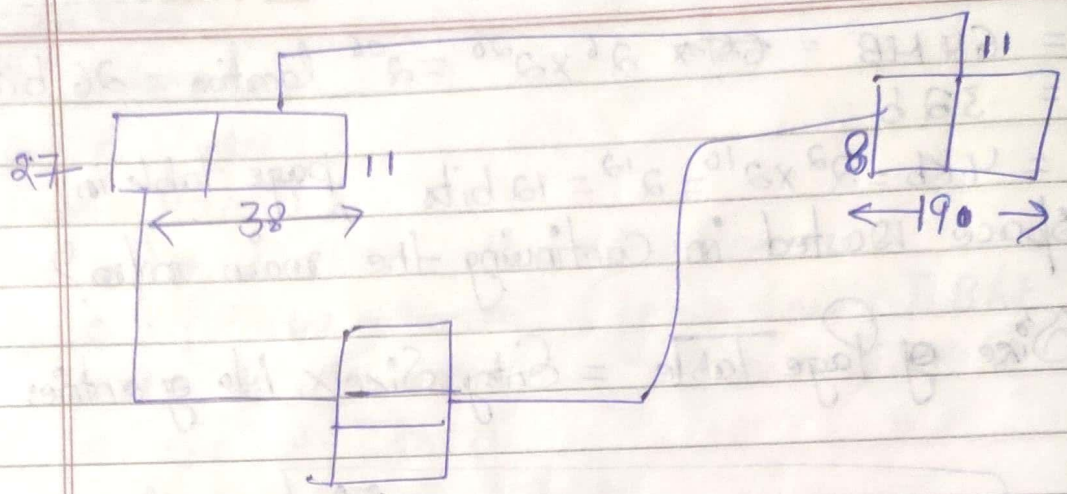


Q) MM = 256 MB = $2^8 \times 2^{20} = 2^{28}$ = 28 bits
PS = 4KB = $4 \times 2^{10} = 2^2 \cdot 2^{10} = 2^{12}$ = 12 bits (offset)
LA = 40B =
Process Size = 4 MB
Total spaces wasted for a process given?



4MB / 4KB = 1

⇒ 1K x 2KB
⇒ 2KB.



SM = 256 GB = $2^8 2^{30} = 2^{38} = 38$

PS = 2KB = $2 \times 2^{10} = 2^{11} = 11$ bit

MM = 512 KB = $512 K = 2^9 \times 2^{10} = 2^{19} = 19$

PT = 8KB.

$= 2^3 \times 2^{10} = 2^{13} = 13$ bits

$\Rightarrow x \times 1B = 8KB$

$x = 8K.$

Process have 8K page.

$= 8K \times 2KB$

$= 16 MB$