MEMORY MANAGEMENT

/* INTRODUCTION */ (1)

- Functionality :- Allocating and de'allocating memory to the process.
- GOAL :- Efficient utiligation of memory by using minimiging the Internal and external fragmentation.

$$\begin{cases} 2^{10} \Rightarrow 1024 \approx 10^3 \Rightarrow (1K) \\ 2^{20} \Rightarrow (1M) \\ 2^{30} \Rightarrow (1G) \\ 2^{40} \Rightarrow (1T) \end{cases}$$

- Memory Organization Map -

0	1010 1101	000
1	---- ----	001
2	---- ----	010
3	---- ----	011
4	---- ----	100
5	---- ----	101
6	---- ----	110
7	---- ----	111

(Memory)

8-word $\rightarrow 2^3$

1 word → 8 bits

Capacity of the memory in Bytes,

$\Rightarrow$ no of words * word size

$\Rightarrow$ 8 * 8

$\Rightarrow$ 64 bit

$\Rightarrow$ 8 byte.

- **8-1** Consider a system which has 256 KW and each has size of 64 bits. Then what is the capacity of memory in bytes.

→ # no of words $\Rightarrow$ 256 K —— (1)

size of word = 64 bit.

= 8 byte. —— (2)

Capacity = (1) * (2)

= (256 X 8) byte. $\Rightarrow 2^8 * 2^{10} * 2^3 \Rightarrow 10^{21}$ byte

$= 2 \times 2^{20} =$ 2 MB.

Q-2 Consider a system which has 512 M words and each word has the size of 16 bytes then capacity of memory is -

→ # no of words = 512 M — (1)

Size of each word = 16 bytes — (2)

Capacity of memory = (1) * (2)

$= 512M * 16$

$= 512 * 2^{20} * 2^4$

$= 2^9 * 2^{20} * 2^4$

$= 2^{33} \Rightarrow 2^{30} * 2^3$

$= \boxed{8\,GB}$

Q-3 Consider a System where 32 binary bits are used to represent all the words of memory, and each word has the size of 32 bits. Then what is the capacity of memory in Bytes -

→ # no of words $= 2^{32}$

word size = 32 bit

$\Rightarrow$ 4 bytes.

Capacity $= 2^{32} * 4$

$= 2^{32} * 2^2$

$= 2^{34}$

$= 2^{30} * 2^4$

$= 16\,GB$.

2010

Ram Chip Introduction :-

RAM chip Implementation -

Q-1

Ram chip Sige = 128 Bytes.

Organige the memory capacity of 16 KB by using above Ram chip -

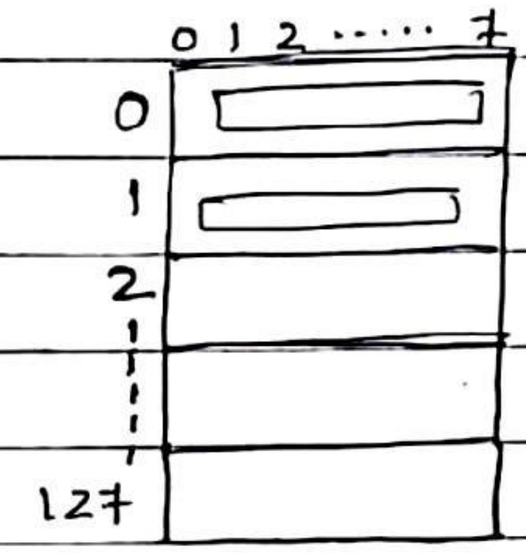
a) NO of RAM chips required.

b) Draw the memory Organigation Map.

→ 128)(B

no of word

word size.



✓

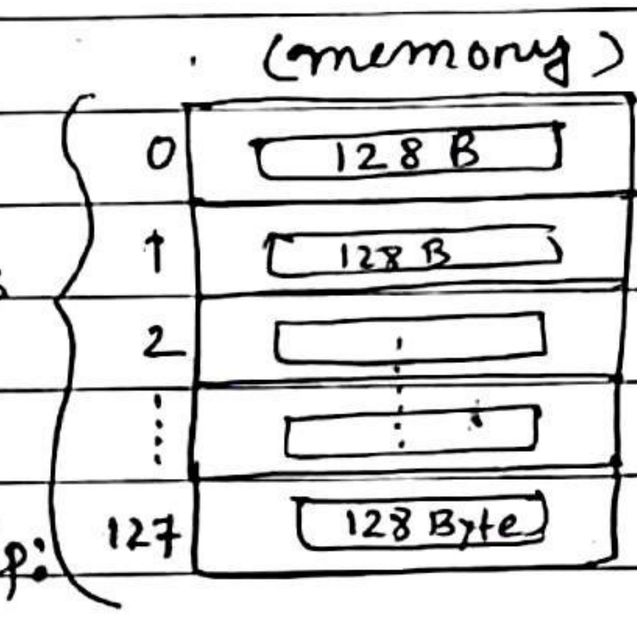
(a) No of RAM chip required,

$$= \frac{16\ KB}{128\ B}$$

$$= \frac{2^4 \times 2^{10}}{2^7}$$

$$= 2^7$$

$$= 128$$



(b) Memory organigation Map:

- Q-2 (Ram chip Implementation)

RAM chip :- 256 * 1 bit.

Memory capacity = 16 MB.

i) No of RAM chip require $= \frac{16\ MB}{256 * 1\ bit} \Rightarrow \frac{2^4 * 2^{20} * 2^3}{2^8 * 2^1}$

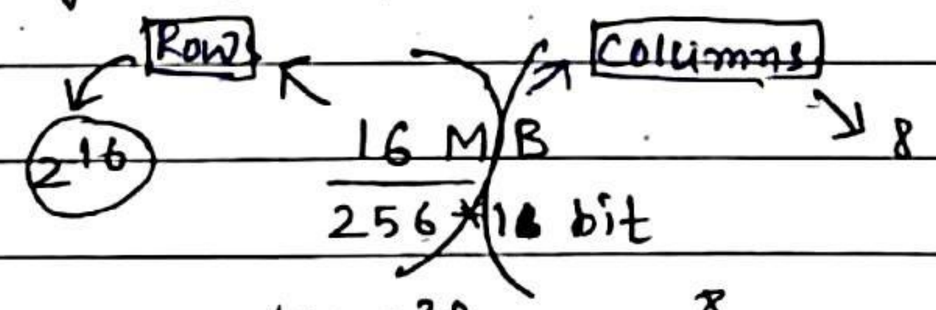
$\Rightarrow 2^{19}$

$\Rightarrow 2^9 * 2^{10}$

$\Rightarrow$ 256 K

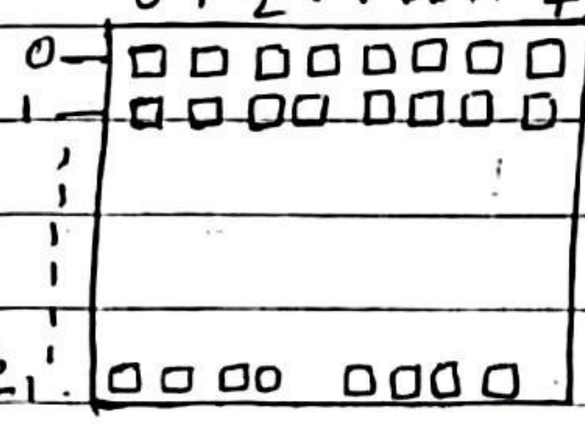
OR

ii) Memory organigation Map:



$\frac{16 \times 2^{20}}{2^8}$ $\quad \frac{8}{1}$

$= 2^{16}$.



memory (16MB)

RAM chip (no) * RAM chip (size) = 16MB

$\Rightarrow$ RAM chip no $= \frac{16\ MB}{256 * 1\ bit}$

$= 256K$

• **8-3** Ram chip Implementation -

Ram chip size = 512 * 2 bit.

Memory capacity = ~~25~~ 32 MB.

i) No of Ram chip = $\frac{32\,MB}{512 * 2\,bit} \Rightarrow \frac{2^5 * 2^{20} * 2^3}{2^9 * 21}$

$= 2^{18}$.

(2^{16}) Row

32 MB → Column (4)

512 * 2 bit

~~$\frac{2^5 * 2^{20}}{2^9}$~~ $\frac{8}{2}$ ::

$= 2^{16}$ $= 4$

ii) Map -

	0	1	2	3
0				
:	:	:	:	:
$2^{16}-1$				

main memory (32 MB) :

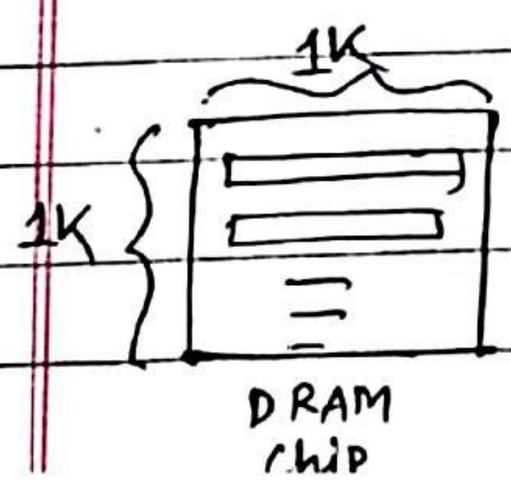
✓ GATE (2010)-8 :

A main memory unit with capacity of 4MB is built using 1M * 1 bit DRAM chips. Each DRAM chip has 1K rows of cells with 1K cells in each row. The time taken for single refresh operation is 100 ns. The time required to perform one refresh operation on all the cells in the memory unit is -

a) 100 ns b) $100 * 2^{10}$ ns c) $100 * 2^{20}$ ns d) $3200 * 2^{20}$ ns

→ no of RAM chip = $\frac{4\,MB}{1M * 1\,bit} \Rightarrow \frac{2^2 * 2^{20} * 2^3}{2^{20} * 2^0} \Rightarrow 2^5 \Rightarrow 32$

→ $32 * 2^{20} * 100$ ns
= $3200 * 2^{20}$ ns



cells in one DRAM chip, = 1K * 1K

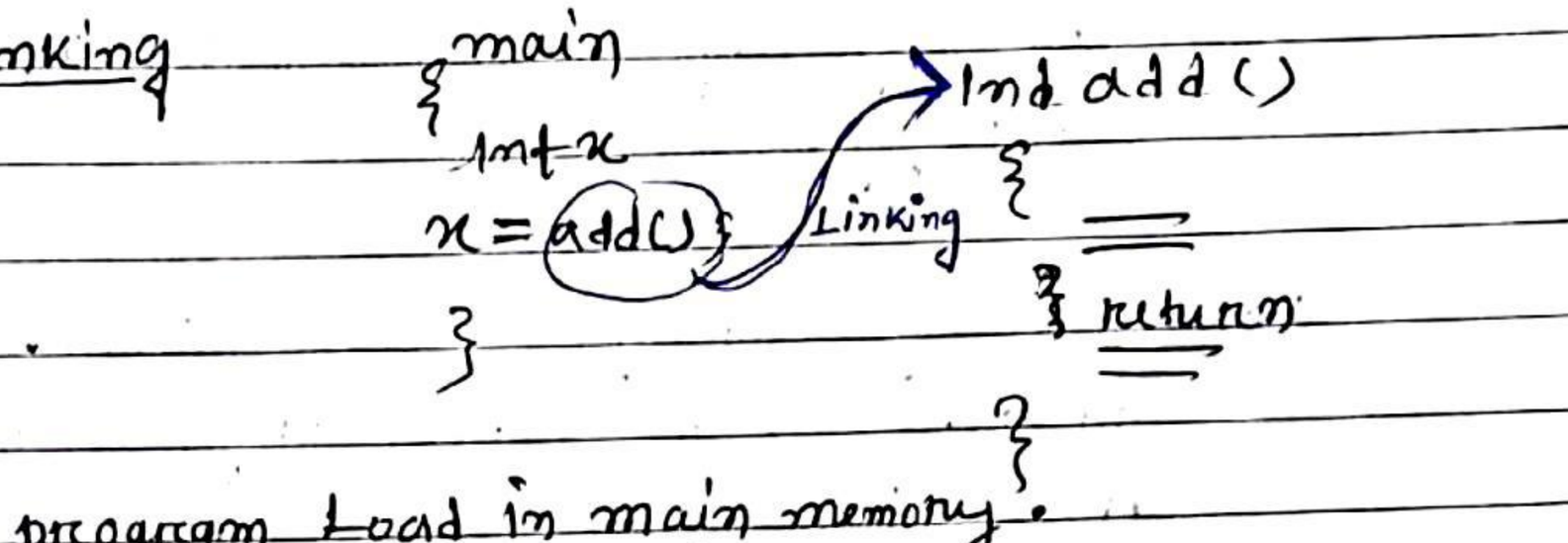
$= 2^{10} * 2^{10} \Rightarrow 2^{20}$ cells.

Total cells to build (4MB) main memory = $32 * 2^{20}$

Loading, Linking and Address Binding:

→ Loading & Linking

- → Static.
- → Dynamic.



Loading:- When program load in main memory.

- Static :-

→ Loading the entire program into memory before the start of the program execution is called static loading.

→ Inefficient utiligation of memory because whether it is required or not required, the entire program is brought into main memory.

→ The Program execution will be faster.

Linking → Linking all the modules / functions of the program.

→ If the static loading is used, then static linking will be applied.

- Dynamic :-

→ Loading the program into memory on demand is called as dynamic loading.

→ The efficient utiligation of memory.

→ The program execution will be slower.

→ If the dynamic loading is used accordingly the dynamic

- linking will be applied.

→ The megority of O.S will use dynamic loading/linking strategy.

- Addressing Binding :- Association of program Instruction and data to the actual physical memory locations is called Address Binding.

Program P1

I_1 → 1000
I_2 → 2000
I_2 → 3000

- Type of Address Binding :-

→ Compile time Address Binding (by- compiler)
→ Load time A.B (Loader)
→ Dynamic or Execution time A.B (processor).

*. compile time A.B :-

→ If the compiler is ~~re~~responsible of association of program instructions and data to the actual physical memory locations is called compile time A.B.

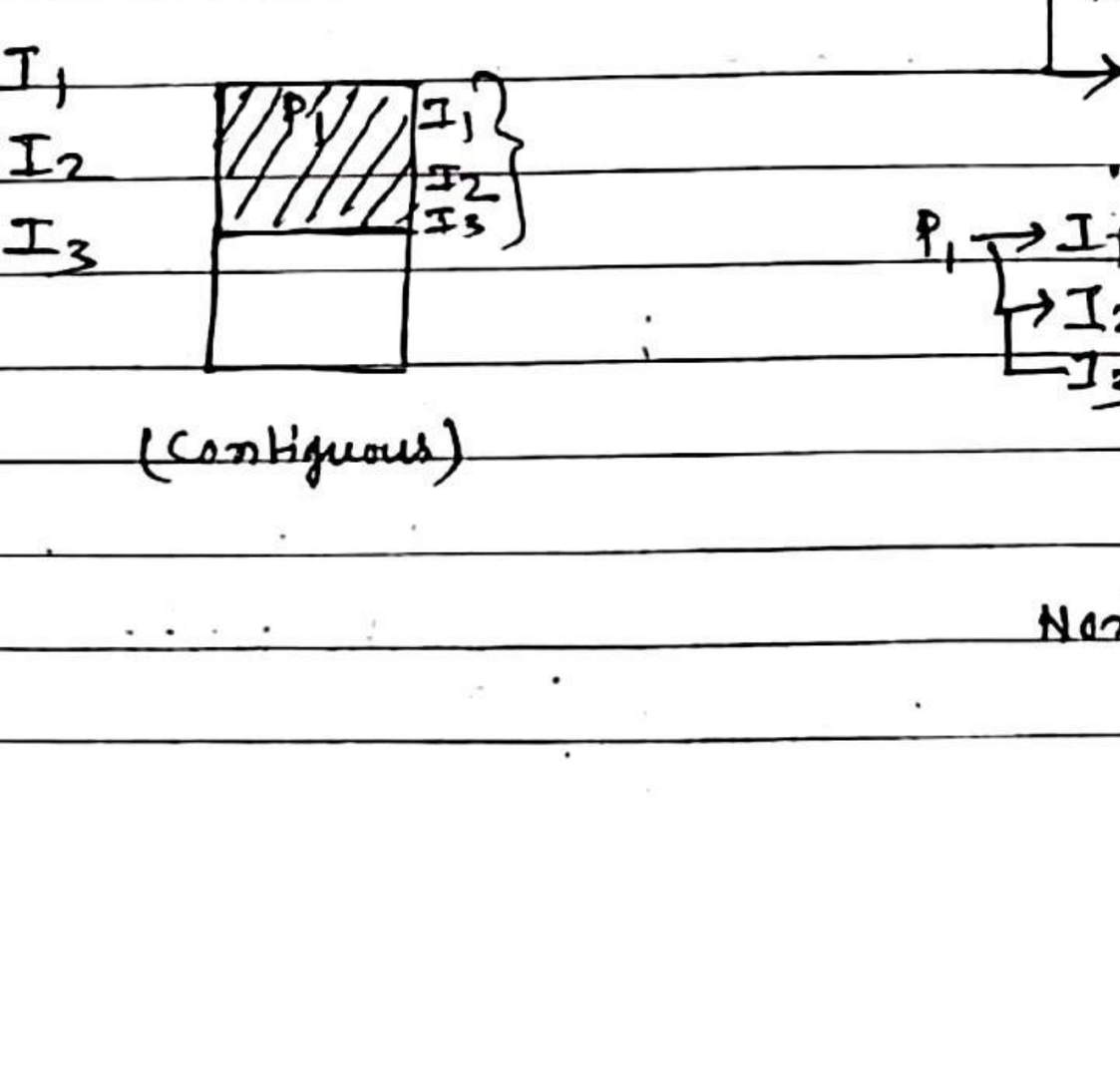
→ The compile time Address Binding will be done before loading the program into ~~me~~ main memory.

→ The compiler leads to interrupt with O.S memory manegers to perform compile time address binding.

*. Load time A.B :- The Load time A.B will be done after the program is loaded into main memory.

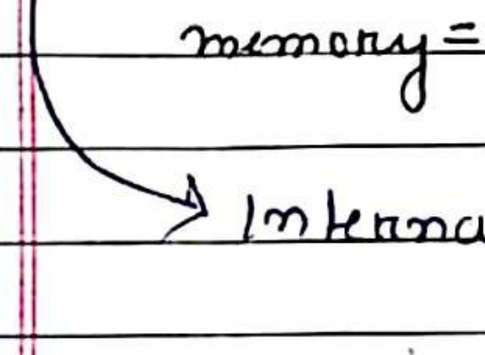
*. Execution time or dynamic A.B :- The Address binding will be postponded even after the loading the program into main memory.
→ This type of A.B will be done at the time of program execution.
→ processors does this type of A.M.
→ The Majority of the O.S uses the Dynamic A.B.

Memory Management Techniques Introduction :-



- Fixed partition Schemes :-

[Contiguous Allocation Schemes]



→ In fixed partition schemes memory will be devided into fixed no of partitions. $P_1 = 60KB$

→ fixed means no of partitions are fixed not the size of partition.

→ In one partition only 1 process will be accomodated. = 8 process.

→ the degree of multiprogramming is restricted by the no of partitions in the main memory.

→ Internal fragmentation.

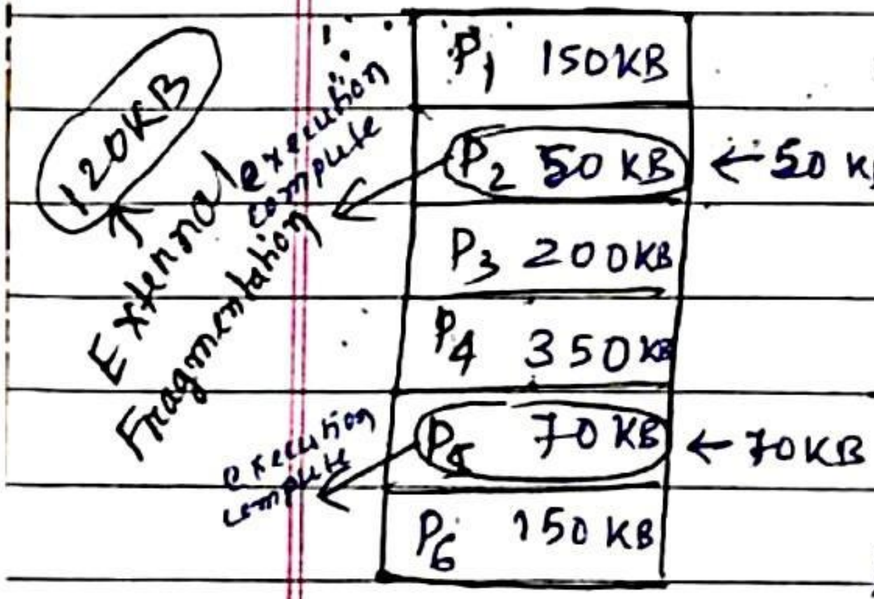
Every partition is associated with the limit Registers-

1) Lower limit :- starting address of the partition.

2) Upper limit :- Ending address of the partition.

- Variable partition Schemes :-

[Contiguous Allocation Schemes]



→ In this scheme, initially the memory will be single contiguous free block.

→ Whenever the request by the process in accordingly the partition will be made.

$P_1 \rightarrow 150KB$ | $P_4 \rightarrow 350$ KB

$P_2 \rightarrow 50$ KB | $P_5 \rightarrow 70$ KB

$P_3 \rightarrow 200$ KB | $P_6 \rightarrow 150$ KB

$P_7 \rightarrow 100$ KB)

To Avoid the problem of External fragmentation:

*. Compaction :- is undesirable. Moving all the process towards the top or towards the bottom to make the free available memory in a single contiguous place is called as compaction.

→ Compaction is undesirable to implement because it disturbs all the running process in the memory.

- Partition Allocation Schemes:

When more than one partition is freely available to satisfy the request of the process then for the decision making of selecting a partition will be done by partition allocation methods.

1) First fit :- Allocate the process in the perition which is first sufficient partition from the top of the memory.

2) Best fit :- Allocate the process in the perition which has the smallest sufficient partition among the free available partition.

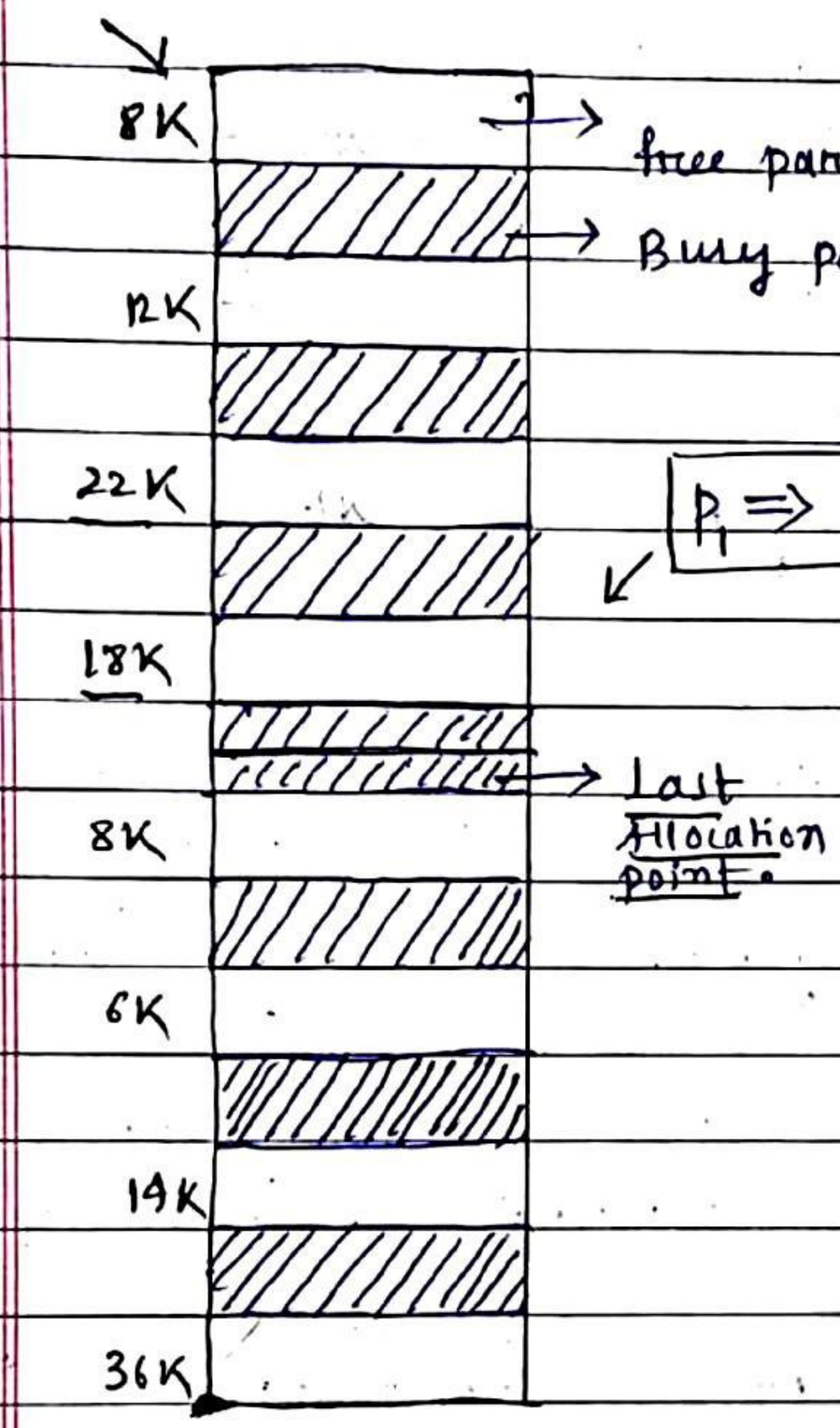
To do this, we need to do Searching. To find out the smallest partition, it requires searching all the partions in the memory.

3) Worst fit :- Allocate the process in a partition which is largest sufficient among all the free available partition.

To find out the largest sufficient, it requires to search all the partitions in the memory.

4) Next fit :- Next fit also works like the first fit but it will search for first sufficient partition from the last allocation point.

- Best fit, first fit, worst fit, & Next fit (Question-1) :-



a) first fit ⇒ 22K (P_1)
b) Best fit ⇒ 18K (P_1)
c) worst fit ⇒ 36K (P_1)
d) Next fit ⇒ 36K (P_1)

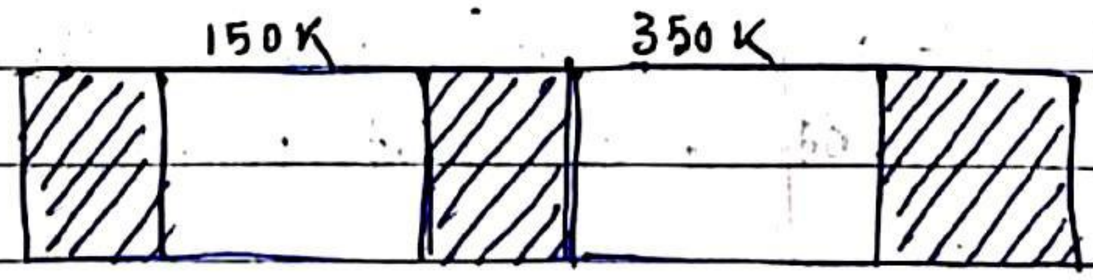
(Question-2) :- How many successive request of 6K will be satisfied by using the first fit and assuming variable partition scheme? (from above diagram)

a) 16 b) 17 c) 18 d) 19.

→ 1 + 2 + 3 + 3 + 1 + 1 + 2 + 6 ⇒ 19

Date (Question-3) :-

Request from the process are 300K, 25, 125K, and 50K respectively.



The above request could be satisfied with (Assume variable partition scheme)

a) Both best fit and first fit.

b) First fit but not Best fit.

c) Best fit but not first fit.

d) neither ~~best~~ first fit nor best fit.

✓ (first fit)

150K	350K
125K	50K
0K	0K

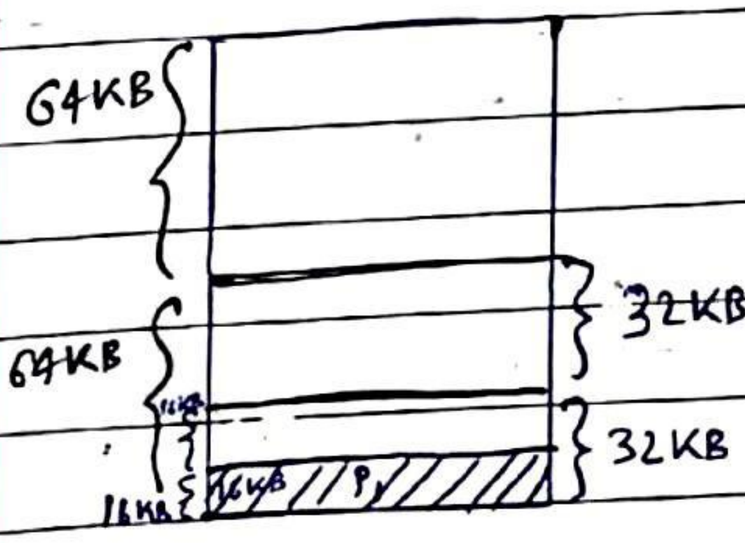
✗ (best fit)

150K	350K
25K	50K
	25K

Buddy System Theory: In the Buddy System initially the the memory will be single continous free block.

→ whenever the request by process comes in the memory will be devided into 2 half blocks.

→ If the request still small, the lower block of the memory is further divided into 2 half block again. In the buddy system, the memory will be allocated from the lower level blocks to the higher blocks.



memory
(128KB)

• Question-1 (Buddy System) -
In a Buddy system of memory management, successive request of 50K, 25K, and 35K are satisfied with 256K of Available memory. How many blocks (with size) are left.

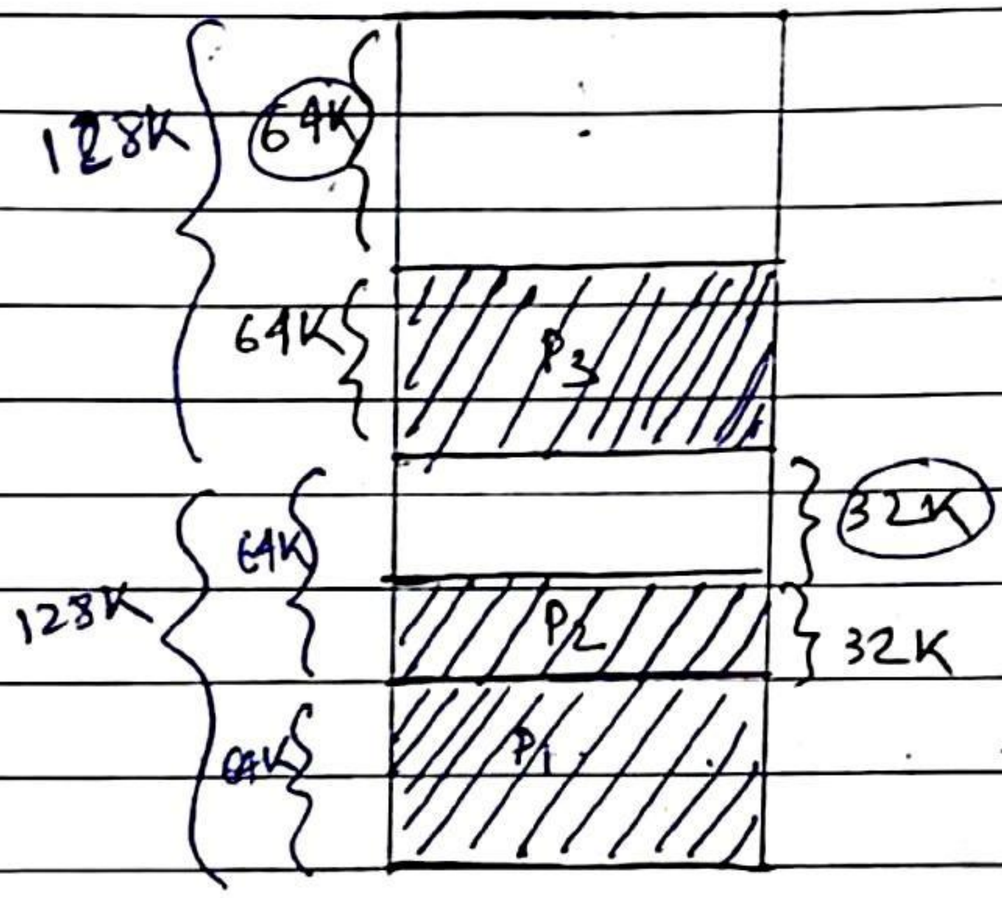
✓ Two Block (32K, 64K).

b) One block (94K)

c) One block (145K)

d) Memory blocks with varying sizes.

→



P_1 – 50 K

P_2 – 25K

P_3 – 35K

256K (memory)

GATE **

PAGING (Non Contiguous) (11)

- Logical Address Space (LAS) or virtual Address space (VAS) → Words → Bytes.
- Logical Address or virtual Address (VA) → Bits.
- Physical Address space (PAS) → Words → Bytes.
- Physical Address (PA) → Bits.

(1) L.A.S = 128 MW

$\Rightarrow 2^7 * 2^{20}$

$\Rightarrow 2^{27}$

L.A = 27 bits

(2) L.A = 38 bits

LAS = 2^{38}

$= 2^8 * 2^{30}$

= 256 GW.

(3) P.A = 16 bits

PAS = 2^{16}

$= 2^6 * 2^{10} \Rightarrow$ 64 KW.

(4) PAS = 32 MW

$= 2^5 * 2^{20}$

$= 2^{25}$

P.A = 25 bits

• LAS and PAS :

*. CPU always generate Logical Address.

L A >> PA

→ The Technique of Mapping processor generated logical Address to physical Address is called as paging.

• LAS division into PAGES –

L.A ⇒ 13 bits → L.A.S ⇒ 2^{13} ⇒ $2^3 * 2^{10}$ ⇒ 8KW ⇒ 8 pages.
P.A ⇒ 12 bits → PAS ⇒ 2^{12} ⇒ 4 KW.

① L.A.S will be devided into equal sige pages.

Page sige = 1KW

$$\# \text{ pages} = \frac{\text{L.A.S}}{\text{page sige}} \Rightarrow \frac{8\text{KW}}{1\text{KW}} \Rightarrow 8$$

L.A.S

0		← 000
1		← 001
2		← 010
3		← 011
4		← 100
5		← 101
6		← 110
7		← 111

page nos ↑ (111 ↓ 3 bits.)

• PAS division into Frames –

PAS is divided into equal sige frames.

*

Page SIZE is Always same as frame size

Frame size ⇒ 1KW

$$\# \text{ Frames} \Rightarrow \frac{\text{P.A.S}}{\text{frame sige}} \Rightarrow \frac{4\text{KW}}{1\text{KW}} \Rightarrow 4$$

0	
1	11
2	
3	10
4	
5	01
6	
7	00

page table

0	P_7	00
1	P_5	01
2	P_3	10
③	P_1	11

frame No. ↑ (11 ↓ 2 bits.)

(*) Some Important Conclusions —

→ $\boxed{\#\text{pages} = \frac{\text{L.A.S}}{\text{Page size}}}$ L.A.S is divided into equal size pages.

$\boxed{\#\text{frames} = \frac{\text{P.A.S}}{\text{frame size}}}$ P.A.S is divided into equ size frames.

PAGE size is Always same as FRAME size

Whenever the paging is applied the page the page table will be maintained.

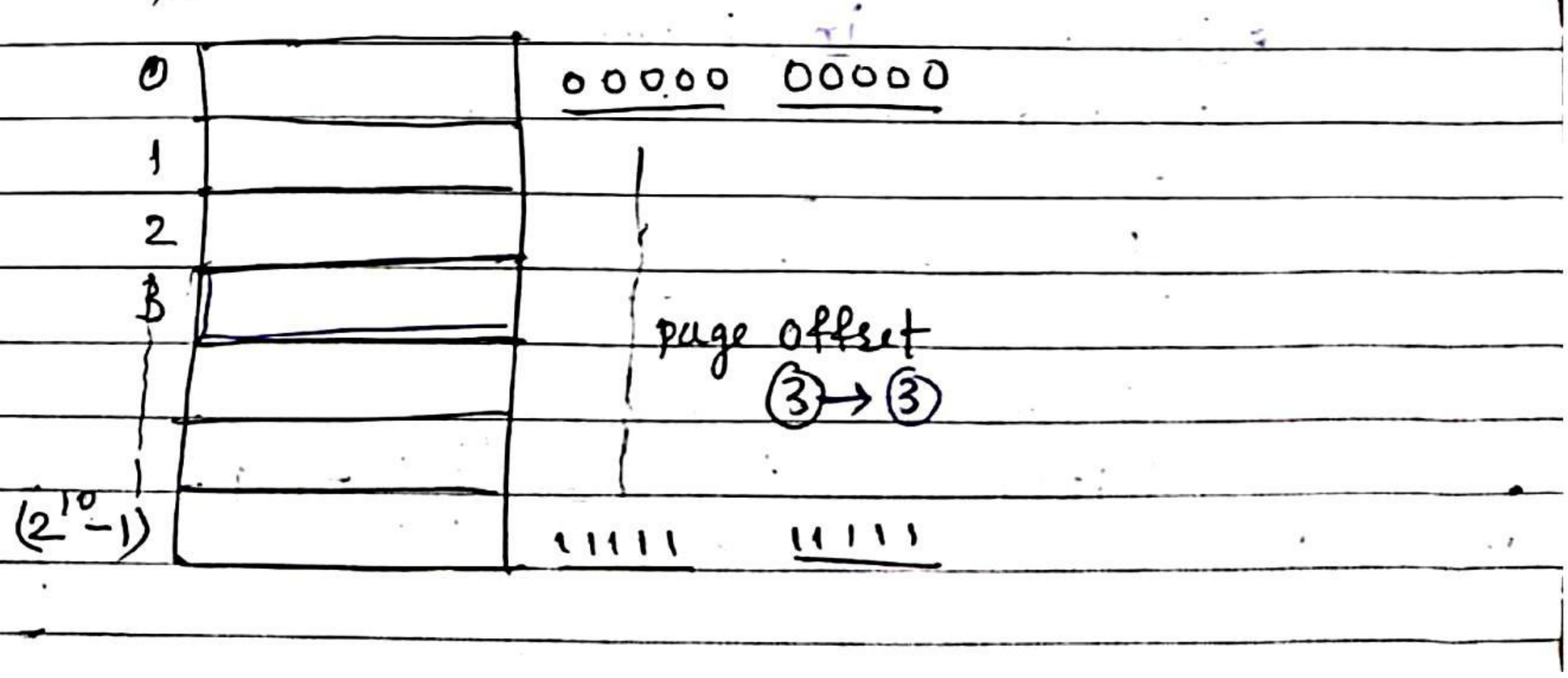
→ No of entries in the page table is same as no of page in L.A.S.

→ page table entry contains the frame no.

- **1KW page Details:**

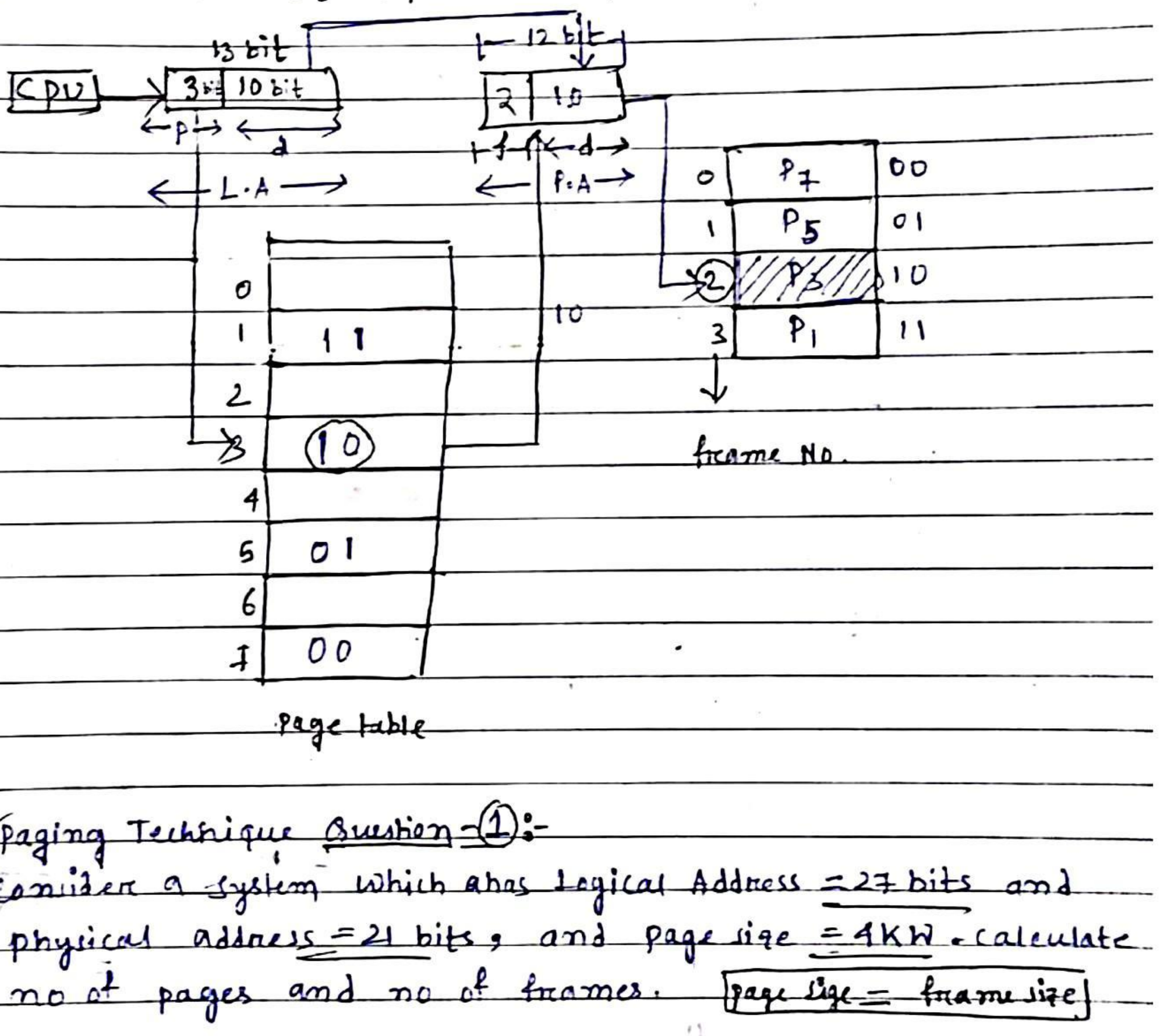
1KW page will look like this.

$1K_{2^{10}}$ → (10 bits)



- Paging Diagram :-

When: L.A => 13 bits.
P.A => 12 bits.
p → no of bits required to represent pages of L.A.S or page no.
d → no of bits required to represent the pages offset.
f → The no of bits required to represent the frame no.



- Paging Technique Question-(1):-
Consider a system which has Logical Address = 27 bits and physical address = 21 bits, and page size = 4KW. calculate no of pages and no of frames. page size = frame size

→ L.A = 27 bits
L.A.S ⇒ 2^{27}

no of pages = $\frac{L.A.S}{4\,KW}$
$= \frac{2^{27}}{2^2 * 2^{10}} \Rightarrow 2^{15}$
$\Rightarrow 2^5 * 2^{10} \Rightarrow 32K$

P.A = 21 bits
P.A.S = 2^{21} bits

No of frames = $\frac{P.A.S}{4\,KW} \Rightarrow \frac{2^{21}}{2^2 * 2^{10}}$
$\Rightarrow 2^9$

• Question-(2).

Consider a system which has no of pages = 2K and page size is 4KW. The physical Address 12 bits then calculate L.A and no of frames.

P.A = 18.

→ # no of pages = $\frac{L.A.S}{page\ size}$

⇒ L.A.S = 2K * 4KW

= $2 * 2^{10} * 2^{2} * 2^{10}$

= 2^{23}

L.A = 23 bits

no of frames = $\frac{P.A.S}{no.of\ space}$

= $\frac{2^{18}}{4KW} \Rightarrow \frac{2^{18}}{2^2 * 2^{10}}$

= 2^6

= 64

• Question-(3)

Consider a system with L.A.S of 128MW and physical Address = 24 bits. The P.A.S divided into 8K frames. Then what is the page size and how many pages in L.A.S.

→ L.A.S = 128MW

page size = 2KW

no of pages = $\frac{L.A.S}{size\ of\ page}$

= $\frac{128 * 2^{20}}{2 * 2^{10}}$

= $\frac{2^7 * 2^{20}}{2^{11}}$

= 2^{16}

= $2^6 * 2^{10}$

= 64K

P.A = 24 bits

P.A.S = 2^{24}

no of frames = 8K

no of frams = $\frac{P.A.S}{frame\ size}$

⇒ frame size = $\frac{2^{24}}{8K}$

= $\frac{2^{24}}{2^3 * 2^{10}} \Rightarrow 2^{11}$.

= $2 * 2^{10}$

= 2K

(page size = frame size)

Gate-2002

• Question = (1)

Consider a system with L.A equal to 32 bits. The P.A.S is 64 MB. The page sige is 4KB. The memory is byte addressable. The page table entry sige of 2 bytes. what is the approximate sige of page table in bytes.

(a) 1MB (b) 2MB (c) 4MB (d) 8MB.

*

→ (No of entres in page table = No of pages in L.A.S)

No. of pages = $\frac{L.A.S}{\text{page sige}}$

$= \frac{2^{32}}{4KB}$

$= \frac{2^{32}}{2^2 * 2^{10}}$

$= 2^{20}$

L.A = 32 bits

L.A.S = 2^{32} bi

So approximate sige of page table = No of entres * entry sige.

$= 2^{20} * 2$

$= 2 * 2^{20} = 2M.$

• Question = (5)

conider a system having a page table with 4K entres. The L.A is 29 bits. what is the physical Address if the System has 512 frames.

→ L.A = 29 bits.

L.A.S = 2^{29}.

no of page = $\frac{L.A.S}{\text{page sige}}$

page sige = $\frac{2^{29}}{2^2 * 2^{10}}$ (4K)

$= 2^{17}$

# of entres in P.T = # of page is L.A.S

no of frames = 512

no of frames = $\frac{P.A.S}{\text{frame sige}}$

frame sige = $\frac{P.A.S}{512}$

⇒ P.A.S = $512 * 2^{17}$

$= 2^9 * 2^{17}$

$= 2^6 * 2^{20}$

$= 2^{26}$

frame sige = page sige

P.A = 26.

Question - (6)

Consider a Question system Where the L.A.S = P.A.S = 2^{16} bytes and page size => 512 bytes. The memory is byte addressable. The page table entry size is 2 bytes. The page table entry also contains other information like 1 bit for valid and invalid, 1 bit for reference, 3 bit for page information/protection and 1 bit for dirty bit. How many bits are still available in page table entry to store Ageing information.

→ 1 bit, 1 bit, 3 bit, 1 bit } 16 bits

6 bit

no of frame (pg entry) = $\frac{P.A.S}{\text{frame size}}$

$= \frac{2^{16}}{512} \Rightarrow \frac{2^{16}}{2^9} \Rightarrow 2^7$

7 bit Require to identify a particular frame no.

16 − (6+7)
= 3 bits (freely available) to Store Ageing information.

Question - (7)

Consider a system which has L.A.S of 256 MB. The physical Address = 24 bits. The memory is byte addressable. The P.A.S is divided into 8 KB frames. Then how many pages are there in the L.A.S.

→ L.A.S = 256 MB

P.A = 24 bits.

~~no of frames = 8 KB~~

no of page = $\frac{L.A.S}{\text{page size}} \Rightarrow \frac{2^8 * 2^{20}}{8KB}$

$\Rightarrow \frac{2^8 * 2^{20}}{2^{13}}$

$\Rightarrow 2^{15}$

$\Rightarrow$ 32K

• Question - (8)

*[The internal fragmentation in the paging is considered as. (P/2) where p is the page size]

Consider a system L.A.S = P.A.S = 's' Bytes. and
page size = 'p' Bytes.
page table entry sig = 'e' Bytes.
The memory is Byte addressable. what is the optimal value of page size by minimizing the memory overhead of maintaing page table and internal fragmentation in the paging.

(a) $P=\sqrt{2se^2}$ (b) $P=\sqrt{2s^2e}$ (c) $P=\sqrt{2se}$ (d) $P=\sqrt{2(se)^2}$.

→ No of pages $= \frac{L.A.S}{page\ size}$

$= \frac{S}{P}$

P.T = $\frac{S}{P}$ * sig of entry

$= \frac{Se}{P}$

P.T + I.F

↓ ↓

$\frac{se}{P} + P/2 \Rightarrow \frac{d}{dP}$

$\Rightarrow \frac{d}{dp}\left(\frac{se}{P}+\frac{P}{2}\right)=0$

$\Rightarrow -P^{-2}se+\frac{1}{2}=0$

$\Rightarrow P^{-2}se=\frac{1}{2}$

$\Rightarrow P^{-2}=\frac{1}{2se}$

$= P^2=2se$

$\boxed{P=\sqrt{2se}}$

• Performance of paging:

→ The page table of the processes will be stored in the main memory.

→ The main memory Access time = M;

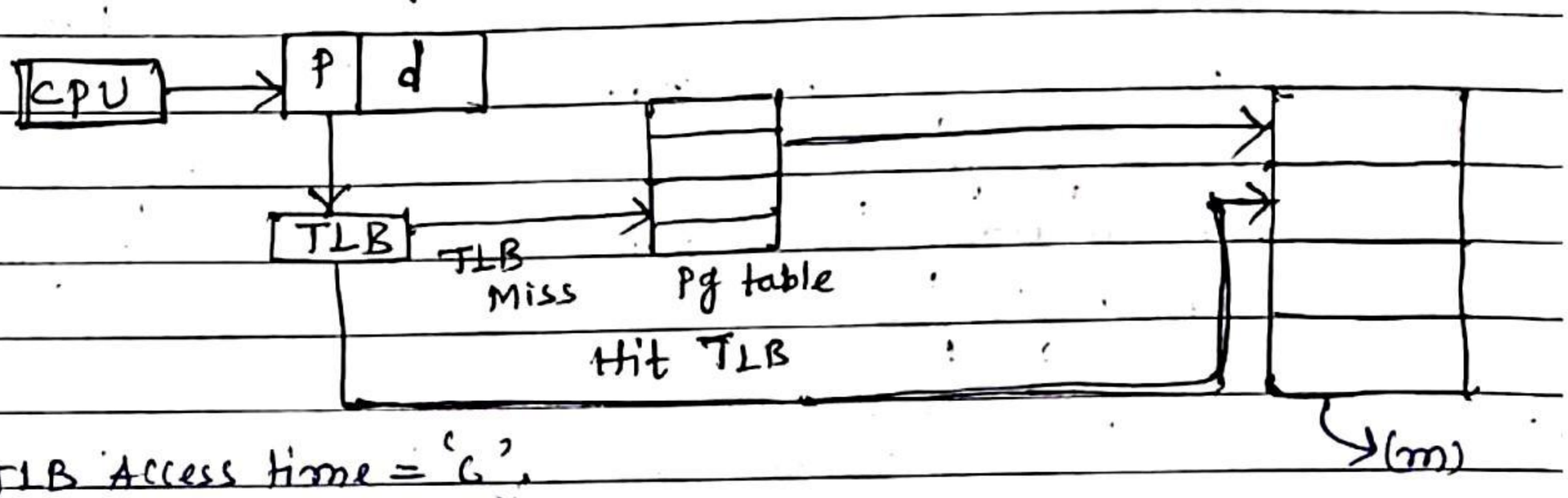
→ If the page table stored in the main memory, the formula for effective memory Access time (EMAT).

$\boxed{EMAT=2m}$

- TLB Introduction:
(Translation LookAside Buffer).

→ This is added to improve the performance of paging.

→ The TLB contains frequently referred page numbers and corresponding frame numbers.



TLB Access time = 'c'.

TLB hit ratio = α

$$EMAT \Rightarrow \alpha(c+m) + (1-\alpha)(c+m+m)$$

$$\boxed{EMAT \Rightarrow \alpha(c+m) + (1-\alpha)(c+2m)}$$

↓ TLB hit ↓ TLB miss

- Q-(1)

Consider a system which has main memory Access time = 100 ns and TLB access time = 20 ns and TLB hit ratio 95% then what is the effective memory access time with TLB and without TLB.

→ m = 100 ns.

TLB = 20 ns

α = 95% => .95

(i) With TLB:

$$EMAT = \alpha(m+c) + (1-\alpha)(c+2m)$$
$$= 0.95(120) + (1-0.95)(20+200)$$
$$= 0.95 * (120) + 0.05 * 220$$
$$= 114 + 11$$
$$= 125 ns.$$

(ii) without TLB:

EMAT = 2m

= 2 * 100

= 200 ns

Q-2

What hit ratio is required to reduce effective memory access time from 300ns without TLB to 250 with TLB. TLB Access time is 60ns.

→

without TLB = 300ns

EMAT ⇒ 2m ⇒ 300ns

m ⇒ 150 ns.

with TLB = 250

EMAT = x(c+m) + (1-x)(c+2m)

⇒ 250 = x(60+150) + (1-x)(60+300)

⇒ 250 = 210x + 360 - 360x

⇒ 150x = 110

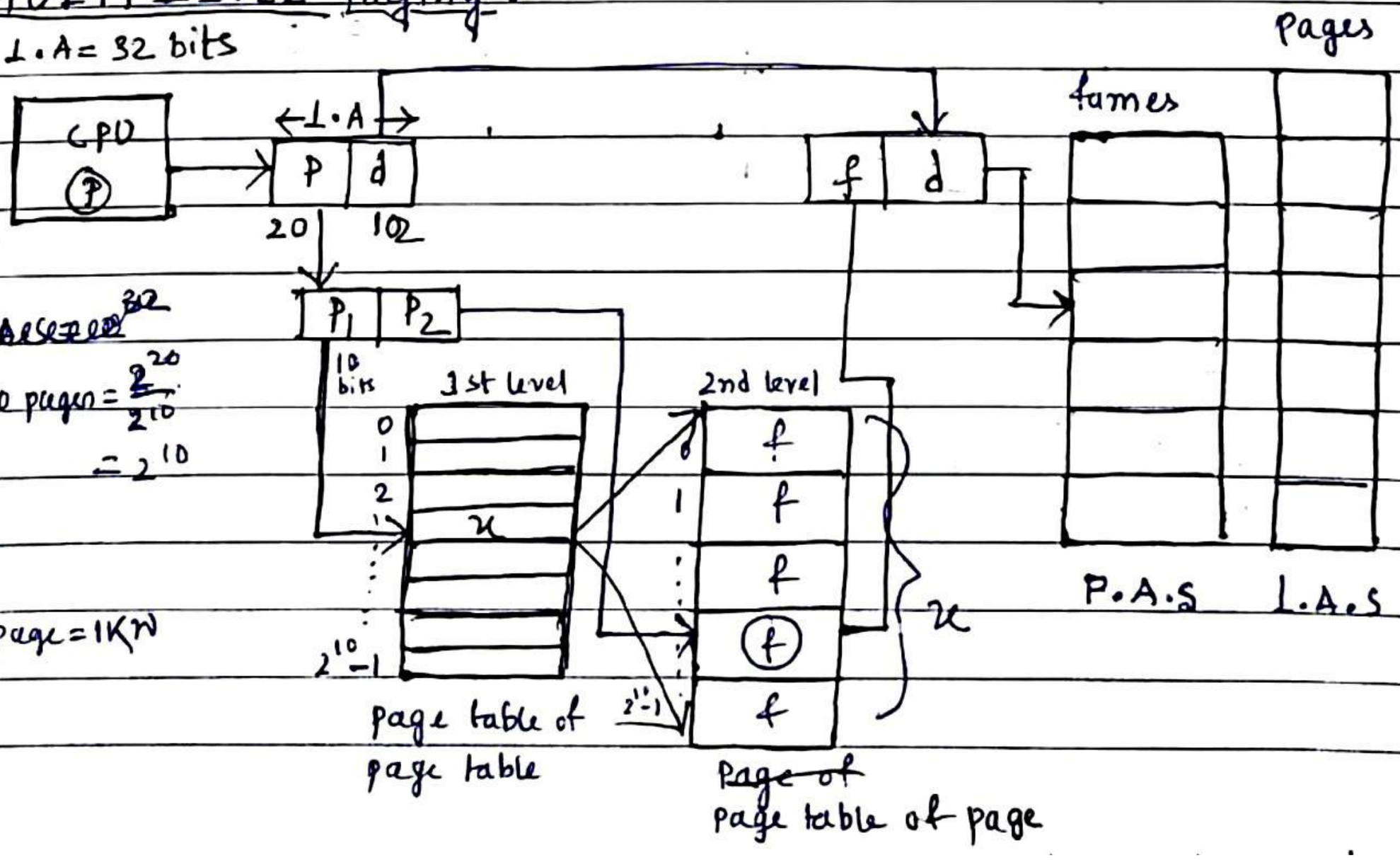
⇒ x = 11/15

⇒ x = .733

⇒ x = 73.3 %.

MULTI LEVEL Paging:

L.A = 32 bits



No. of pages = $\frac{2^{32}}{2^{10}}$

$= 2^{10}$

1 page = 1KW

→ TO Avoid overhead of maintaining large page tables, the multilevel paging will be Implemented.

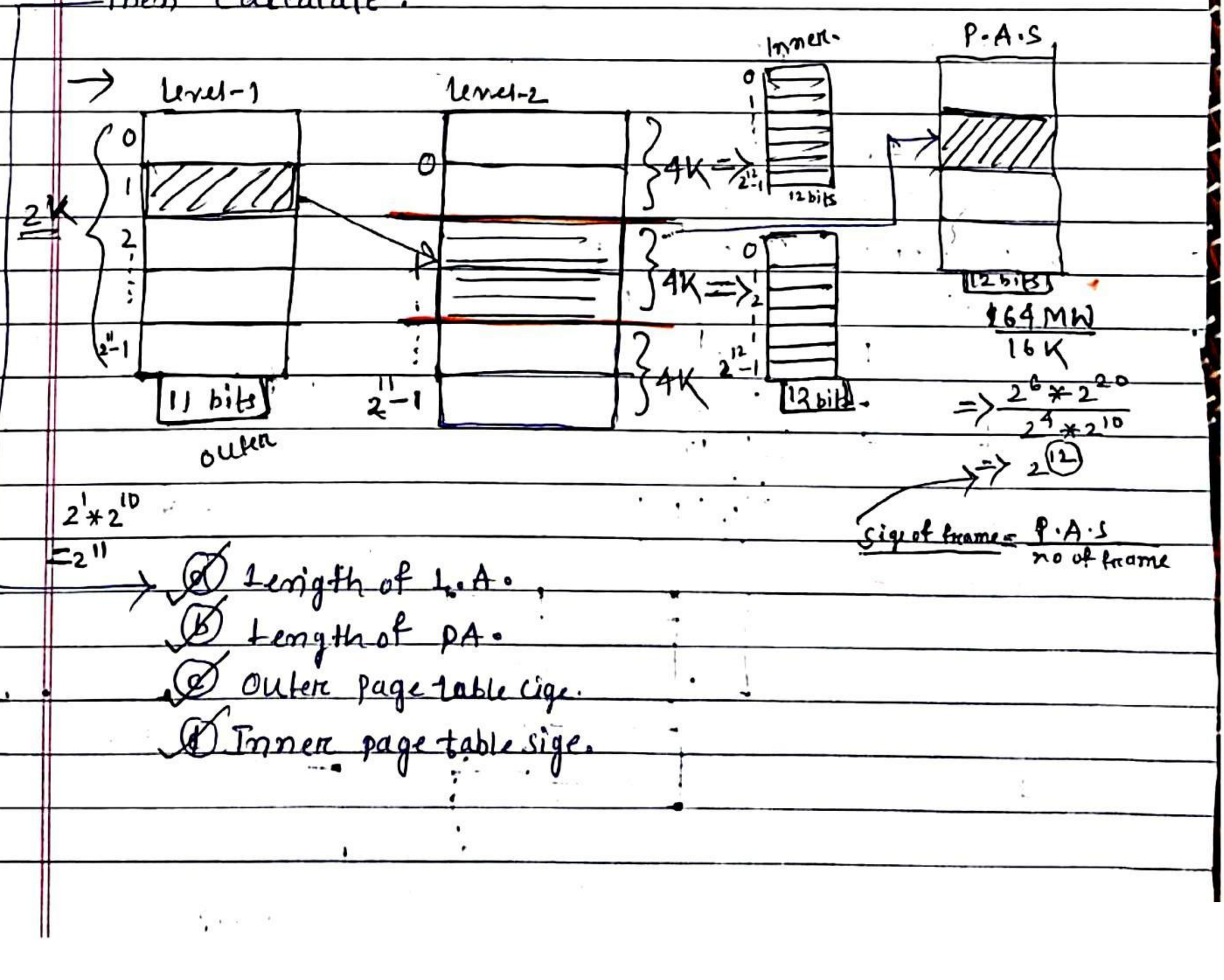
→ In the multilevel paging, the paging will be applied on page table.

P_1 → NO of bits required to represent the page of page table.

P_2 → The no of bits required to represent the page size of page table.

x → Address of page of page table.

Q-1 Consider a system with 2 level paging applicable. The page table has been divided into 2K pages. Each of size 4K words. If P.A.S is having 64 MW, which is divided into 16K frames, and the page table entry size is 4 Bytes, then calculate.



→ (A) Length of L.A.

(B) Length of PA.

(C) Outer page table size.

(D) Inner page table size.

(a) (11+12+12) => 35 bits.

(b) 61 * MB => $2^6 * 2^{20}$ => 2^{26}

= 26 bits

(c) 2K * 1 Byte

=> $2^{11} * 1$

=> 2^{13} Byte.

(d) 1K * 4 Bytes.

= $2^{12} * 2^2$

= 2^{14} Bytes.

Q-2

Consider a system with 2 level paging appliable. The page table is devided into 8K pages, each of 16KB. The memory is ~~devit~~ byte addressable. If the P.A.S is 128 megabytes which is divided into 4KB frames. The page table entry sige of outer page table is 32 bits and page table entry size of inner page table is 64 bits. The calculate -

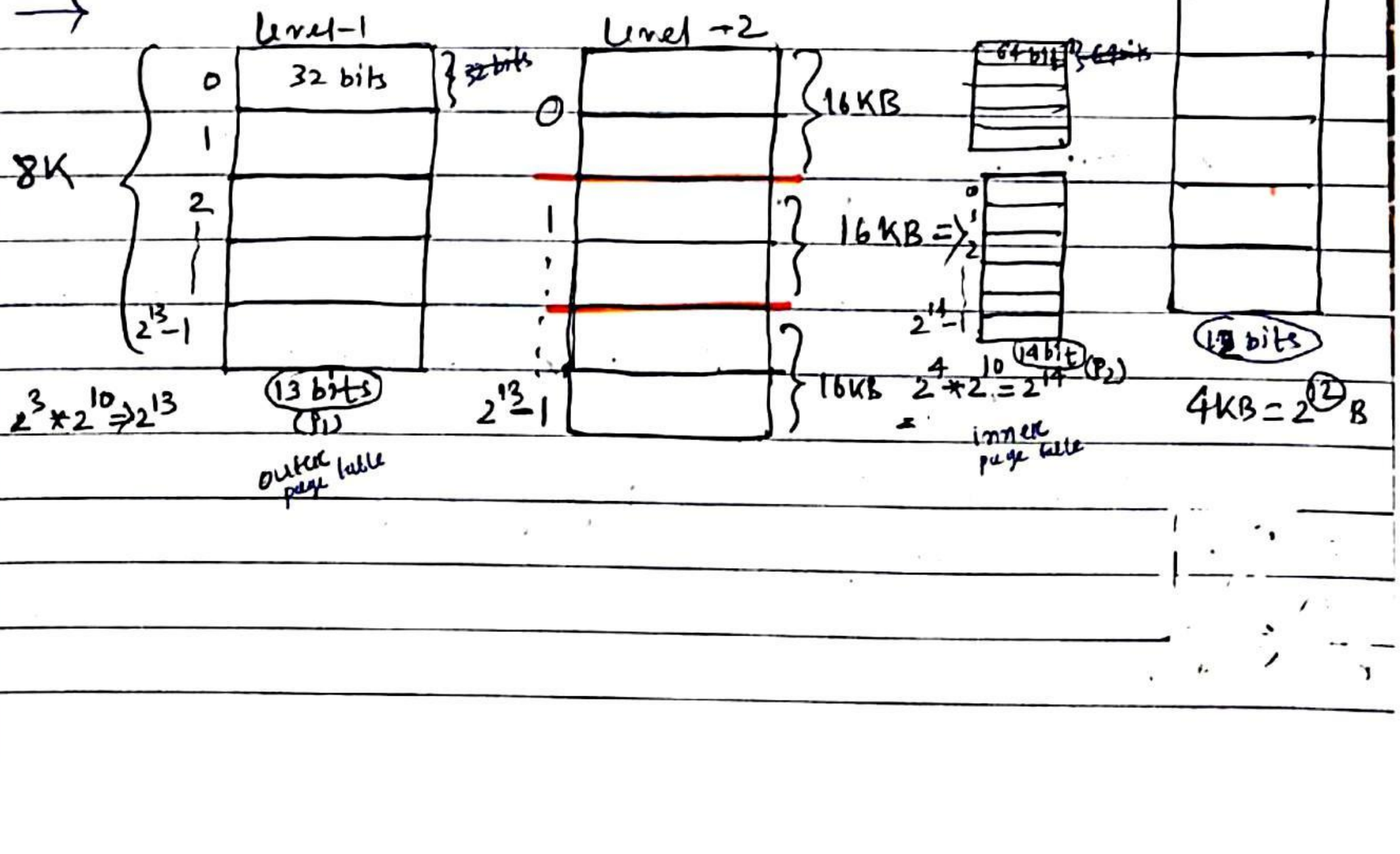
a) Length of LA. (13+14+18) bits => 39 bits.

b) Length of PA. 2^{27} => 27 bits.

c) Outer page table sige. 32 * 8K => $2^5 * 2^3 * 2^{10}$ => 2^{18} bits => 2^{15} Byte

d) Inner page table sige. $\frac{64}{bit} * \frac{16KB}{}$ => $(2^{6-3} * 2^{4+10})$ B => 2^{17} Byte.

→

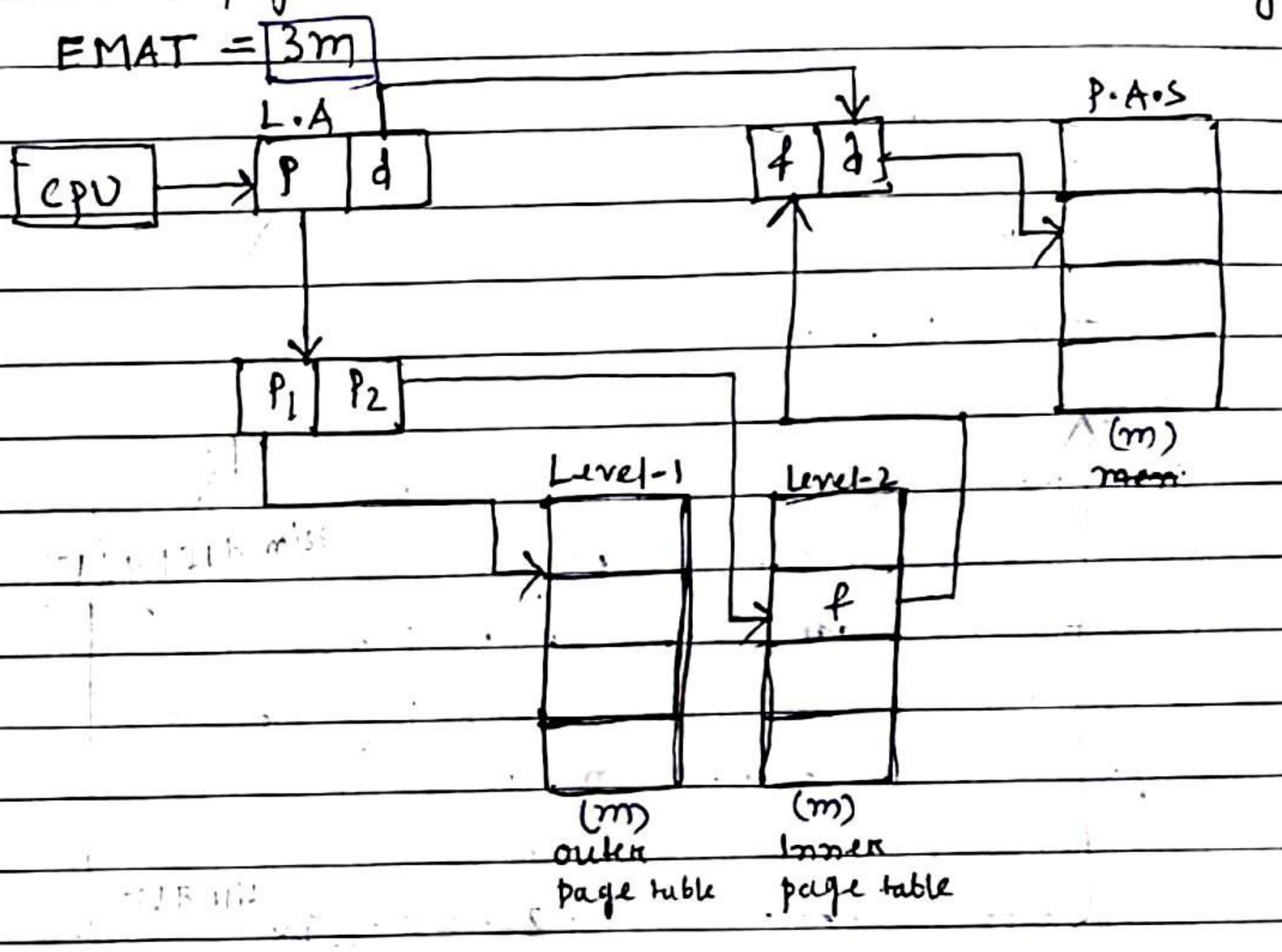


- Performance of 2 level paging with out TLB :-

Let the main memory access time = 'm'.

→ If all the page tables are stored in the main memory.

EMAT = 3m



→ If the TLB is added to improve the performance. The TLB contains frequently refferred page NO's and corresponding frame no.

EMAT = $x(c+m) + (1-x)(c+3m)$.

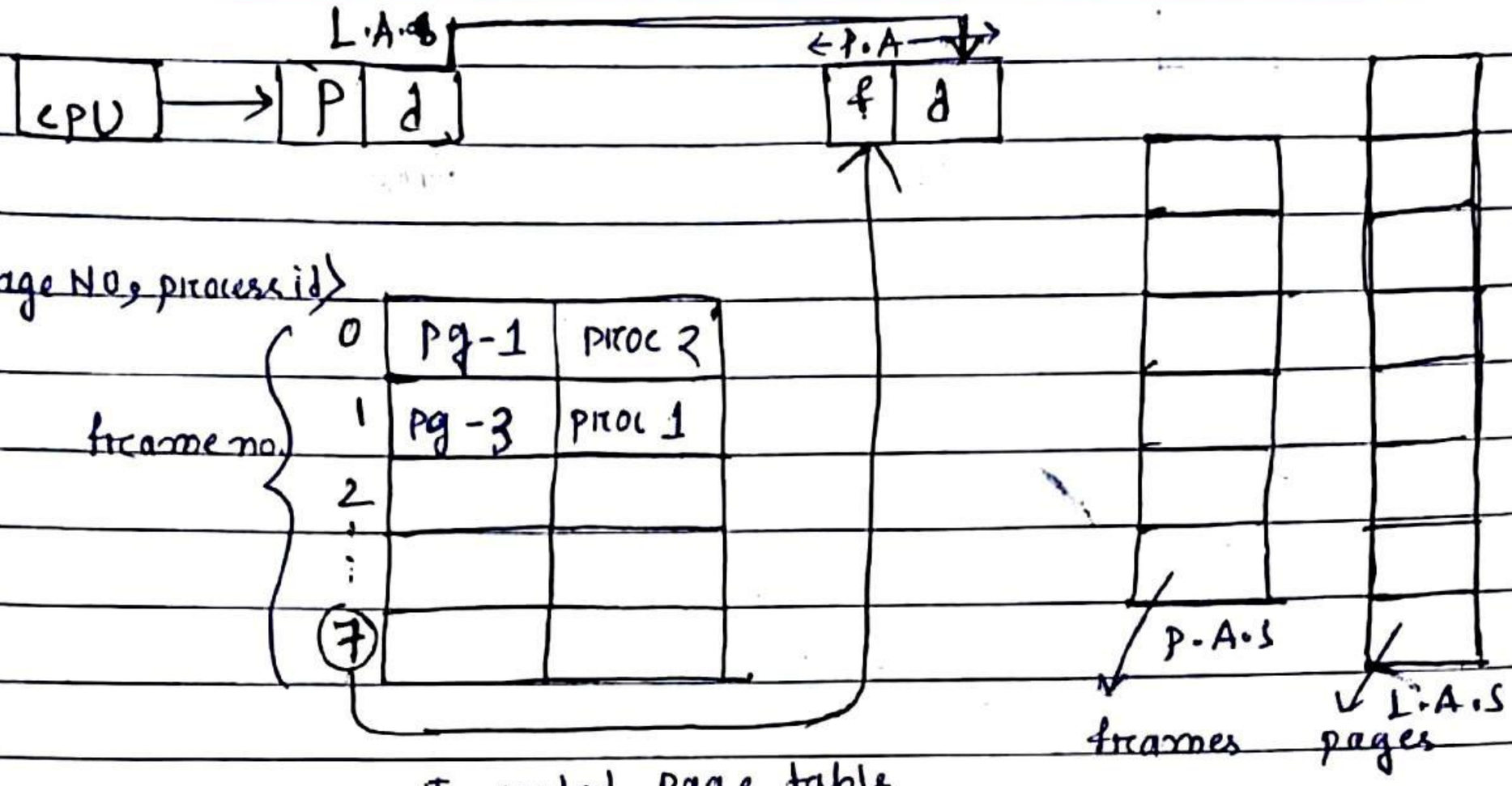
'c' = TLB access time

TLB Hit, TLB miss.

- INVERTED Paging :-

→ To Avoid the overhead of maintaining the page table of every process, the ~~inert~~ Inverted paging will be implemented.

→ In ~~Inert~~ Inverted paging only 1 page table is maintained for all the process.



Inverted page table
or
FRAME table

Q-1

Consider a system with Logical Address (L.A) = 34 bits and physical Address = 29 bits. The page size is 16 KB. The memory is byte Addressable. The page table entry size is 8 bytes. The calculate

1) conventional page table size.
2) Inverted page table size.

> PgT size = No of entry in page table * size of entry

→ 1) Conventional P.T size,

> No of entry = no of page

$$\text{No of entry} = \frac{\text{P.A.S}}{\text{page size}} \Rightarrow \frac{2^{34}\text{ bits}}{16\text{ KB}} \Rightarrow \frac{2^{34}}{2^{14}} \Rightarrow 2^{20}$$

$$\text{P.T size} = 2^{20} * 8 \text{ byte}$$

$$= \boxed{8\text{MB.}}$$

2) Inverted page table size,

> No of entry = no of frames

$$\text{No of entry} = \frac{\text{L.A.S}}{\text{frame size}} \Rightarrow \frac{2^{29}}{16\text{KB}} \Rightarrow \frac{2^{29}}{2^{14}} \Rightarrow 2^{15}\text{ Bytes}$$

$$\text{p.T size} = 2^{15} * 8\text{b} \Rightarrow 2^{18}\text{ B}$$

$$\Rightarrow 2^{8} * 2^{10}\text{ B} \Rightarrow \boxed{256\text{ KB.}}$$

Q-2

Consider virtual Address space of 32 bits and page size of 4KB. Consider RAM of 128KB. The what will be the ratio of page table and interverted page table size of each entry in Both is of 4 Bytes ?

(a) $2^{15}:1$ ✓ (b) $2^{20}:1$ (c) $2^{10}:1$ (d) None of the above.

→ L.A (v.A) = 32 bits
page size = 4KB

P.A.S = 128KB
frame size = 4KB

Page table size = no of entrys * size of entrys

$P.T_N = \left(\frac{2^{32}}{4KB}\right) * 4B$

$= \left(\frac{2^{32}}{2^{12}}\right) * 4B$

$= 2^{20} * 4B.$

$P.T_{INV} = \left(\frac{2^{7} * 2^{10}}{4K}\right) * 4B$

$= \frac{2^{17}}{2^{12}} * 4B$

$= 2^{5} * 4B.$

ratio (N : Inv) $\Rightarrow \frac{P.T_N}{P.T_{Inv}} \Rightarrow \frac{2^{20} * 4}{2^5 * 4} \Rightarrow \frac{2^{15}}{1} = 2^{15}:1$

Q-3

Suppose that a certain computer with paged virtual memory has 4KB pages, A 32-bit byte Addressable virtual Address and 30 bit Byte Addressable physical address. The system manages an inverted page table. Where each entry includes The page number plus 12 overhead bits. How Big is the basic Inverted page table including number and overhead bits.

(a) 2^{10} Bytes (b) ✓ 2^{20} Bytes (c) 2^{30} Bytes (d) 2^{32} Bytes.

→ Page size = 4KB
L.A = 32 bits

P.A = 30 bit

P.T size = no of entrys * size of entry

$= \left(\frac{P.A.S}{\text{Size of frame}}\right) * \left(\overset{20\text{bit}}{\text{Pg.NO}} + 12 \text{ bits}\right)$

$\frac{2^{32}}{2^{12}} \Rightarrow 2^{20}$

$= \left(\frac{2^{30}}{2^{12}}\right) * (20\text{bit} + 12\text{ bit})$

$= 2^{18} * 32 \text{ bit}$

$= 2^{18} * 4 \text{ Bytes}.$

P.T size $= 2^{20}$ Bytes

Q-4 Assume a machine with 64MB physical memory and 32 bit virtual Address. If the page size is 4KB, then what is the approximate size of the page table?

(a) 16 MB (b) 1 MB (c) 24 MB (d) 2 MB

→ P.T size = no of entrys * P.T entry size

(I) P.A.S = 64 MB | $2^6 * 2^{20} = 2^{26}$

(II) P.A = 26 bit |

(III) virtual address (V.A) = 32 bits

(IV) Page Size = 4 KB $= 2^{12}$ Byte.

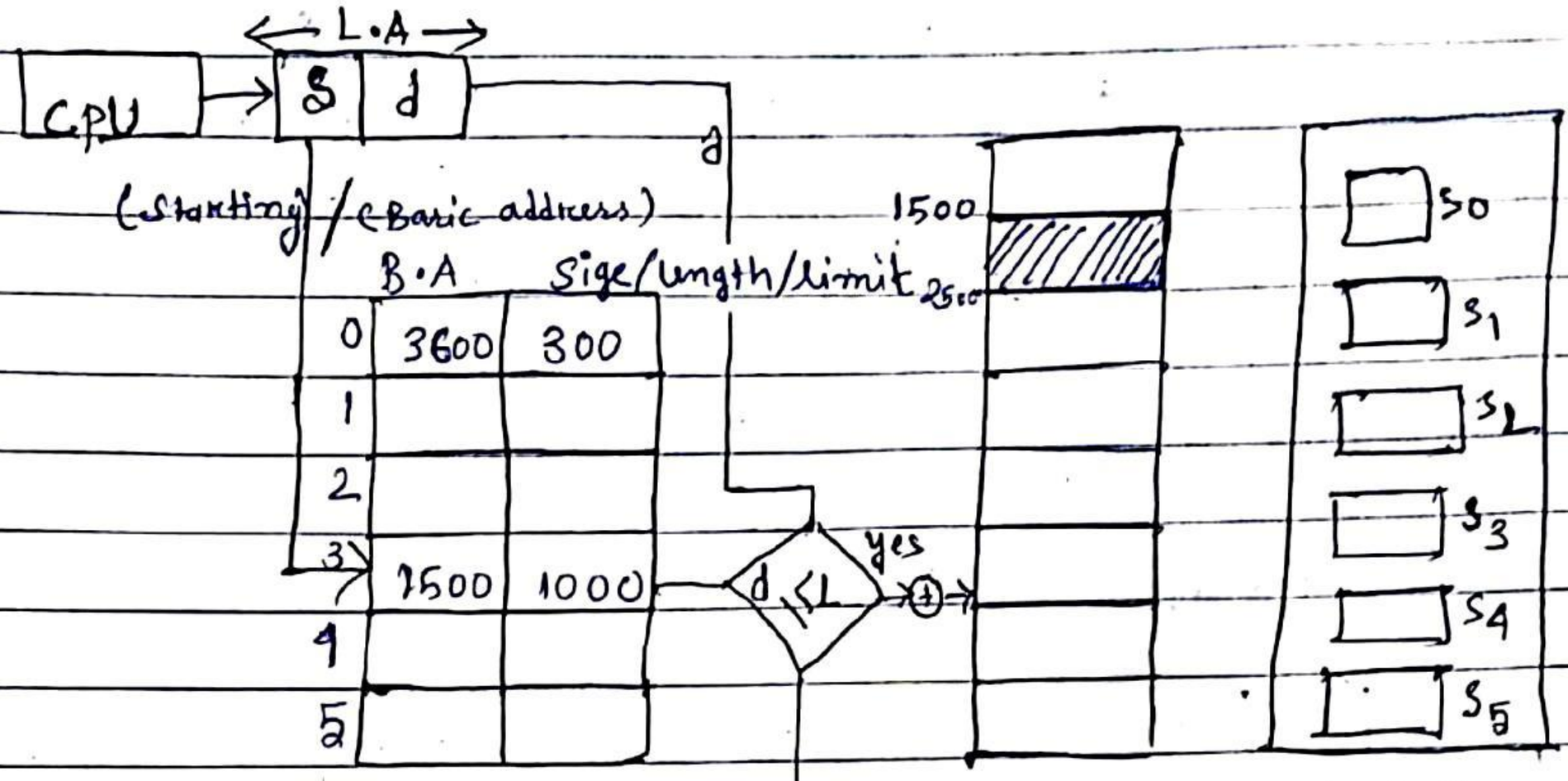
no of entry required $= \frac{2^{32}}{2^{12}} \Rightarrow 2^{20}$

P.T entry size = 19 bit (= 16 bit)
(frame size)

P.T size $= 2^{20} * 16$ bit
$= 2^{20} * 2$ byte.
$= 2$ MB.

no of frame $= \frac{P.A.S}{\text{Frame size}}$
$= \frac{2^{26}}{2^{12}}$
$= 2^{14}$

SEGMENTATION & SEGMENTED PAGING:



NO - Trap to O.S

- Some points :-

→ No of entries in segment table = No of segments in L.A.S.

→ The paging does not follow the users view of memory allocation.

→ To achive user's view of memory allocation, the ~~of~~ segmentation will be implimented.

→ In segmentation the L.A.S is devided into segments.

→ The segments will vary in Sige.

(S) → No of bits required to represent the segments of L.A.S or segment No.

(L) → segment offset No of bits required to represent segment Sige or word no of the segment.

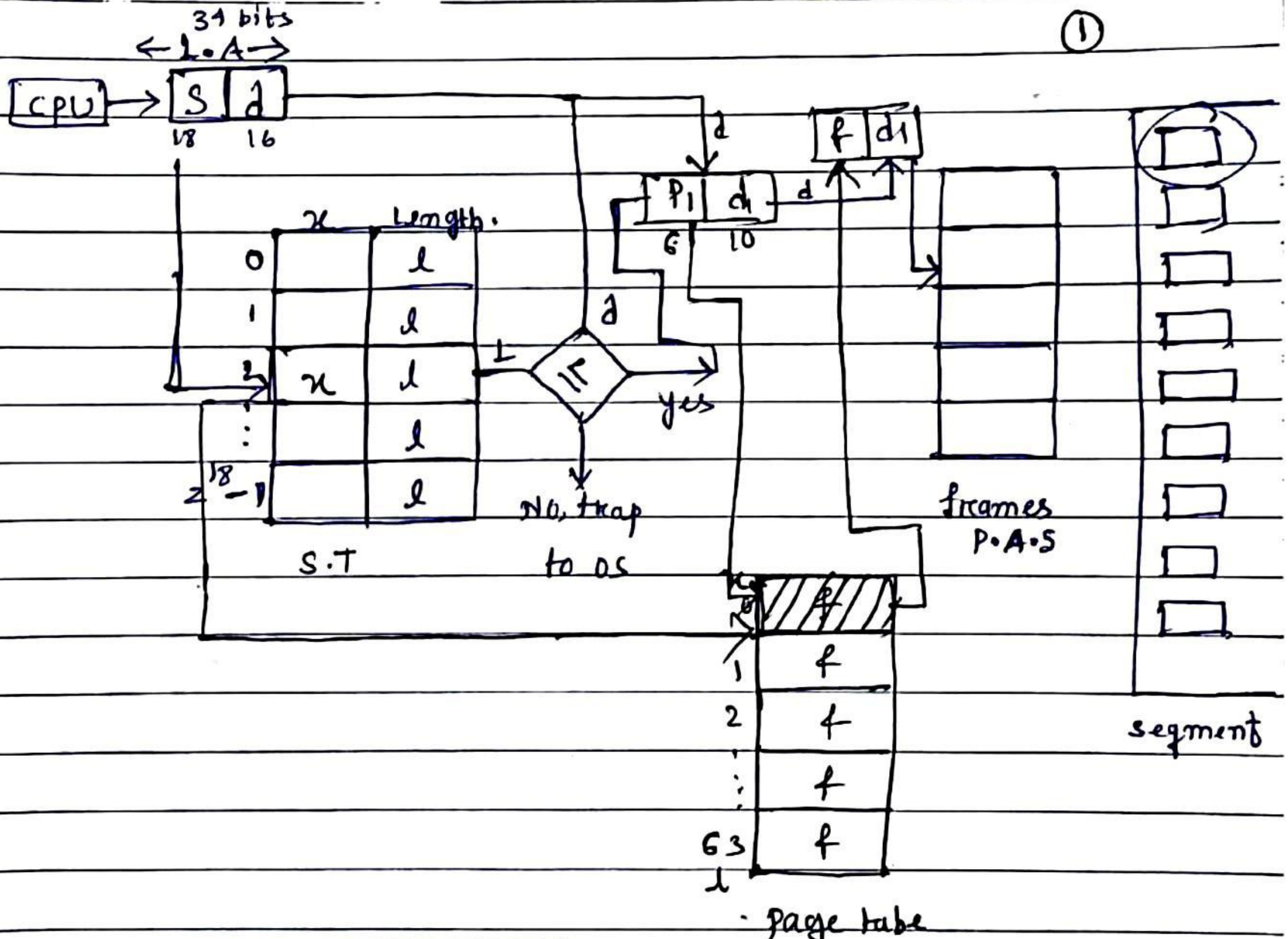
→ The variable size segments are brought into P.A.S so that the segmentation is just behaving like variable partition scheme.

⇓

Hence it suffers from External fragmentation

- paging on segmentation when segment size increase:

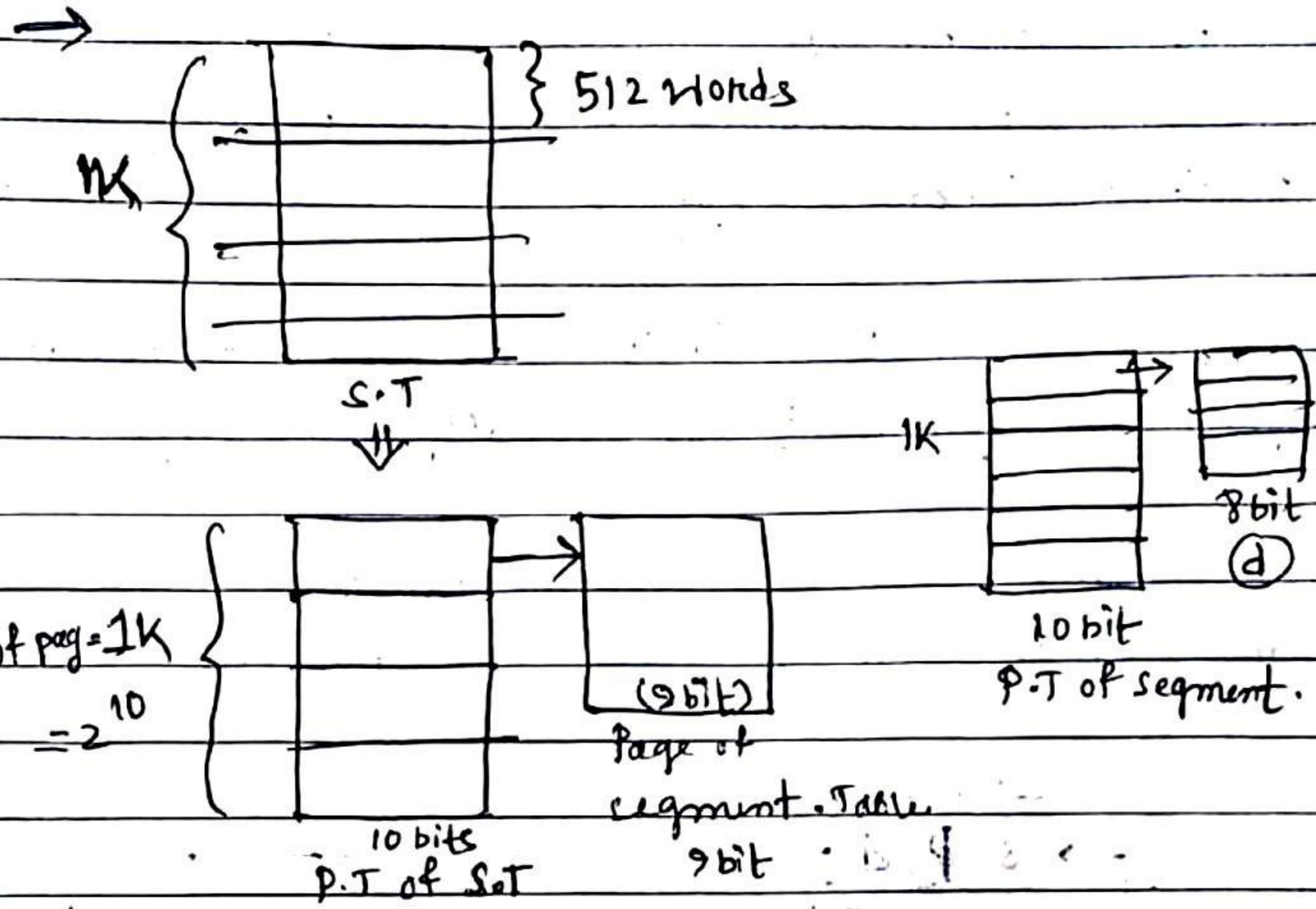
→ problems of external fragmentation and lengthy search times can be solve by paging the segments.



① paging on segments

Calculate →

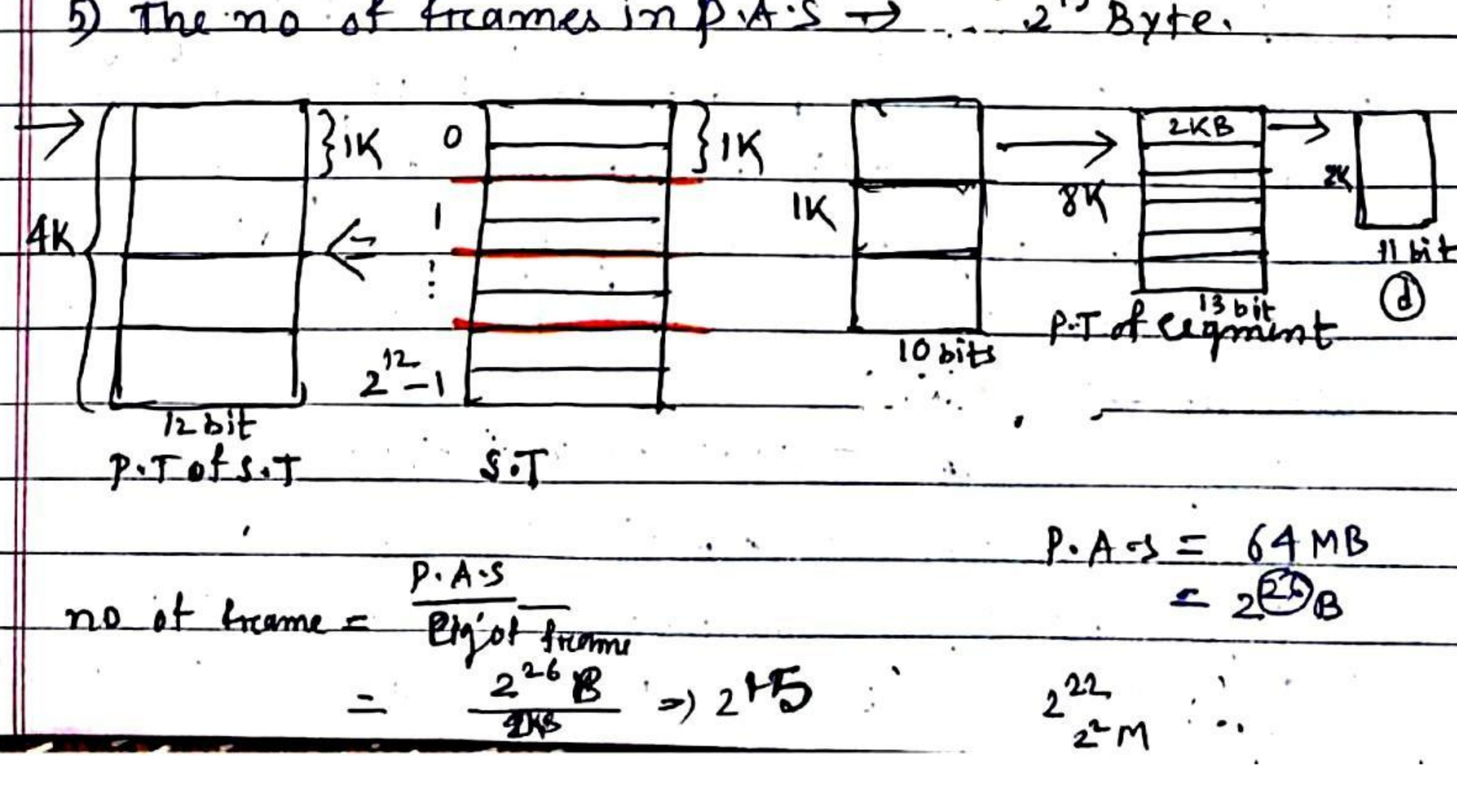
1) length of L.A. → (10+9+10+8) ⇒ 37 bit
2) length of p.A. → 18 + 8 → 26 bit
3) Page table size of segment. → 1K * 4byte → 4KB.
4) Page table size of segment table → 1K * 4B → 4KB



8-2

Consider a system using segment paging Architecture. The segment is divided into 8K pages each of sige 2KB and segment table is divided into 4K pages each of sige 1KB. The memory is Byte Addressable. The P.A.S is 64MB. The page table entry sige is 32 bits. Then calculate -

1) Length of L.A. → (12+10+13+11) → 46 bit.
2) Length of P.A. → 26 bits.
3) page table sige of segment → 2^{13} * 4byte → 2^{15} Byte → 32KB
4) page table sige of S.T. → 4K * 4B → 16KB
5) The no of frames in P.A.S → 2^{15} Byte.



no of frame = P.A.S / Pg of frame
= 2^{26} B / 2KB → 2^{15}

P.A.S = 64MB
= 2^{26} B

2^{22}
2^{2}M

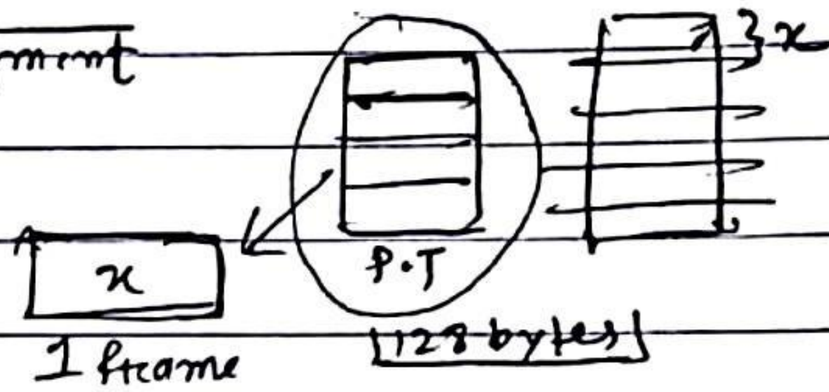
Q-3

Consider a system using segmented paging architecture where L.A.S = P.A.S = 2^{16} Bytes.

L.A.S is divided into 8 equal size segments. The segment is divided into equal size pages which are in the powers of 2 in term of page size. The memory is byte addressable. The page table entry size is 2 bytes. The page tables are stored in main memory. What must be the page size of segment in Bytes so that the page table of the segment exactly fits in 1 page frame?

→ size of segment $= \frac{2^{16}}{8} = \frac{L.A.S}{\text{no of segment}}$

$= 2^{13}$ byte.



P.T size $= \left(\frac{2^{13}}{x}\right) * 2$ byte

$\Rightarrow \frac{2^{14}}{x} = x$

$\Rightarrow \frac{2^{14}}{2^{\gamma}} = 2^{\gamma}$

$\Rightarrow 2^{2\gamma} = 2^{14}$

$\gamma = 7$

[P.T size = no of entry (no of page) * size of entry]

$= \left(\frac{2^{13}}{x}\right) * 2$

$2^7 = 128$

VIRTUAL MEMORY:

Virtual Memory gives an illusion to the programmer that programs of largere cige than actual physical memory can be executed.

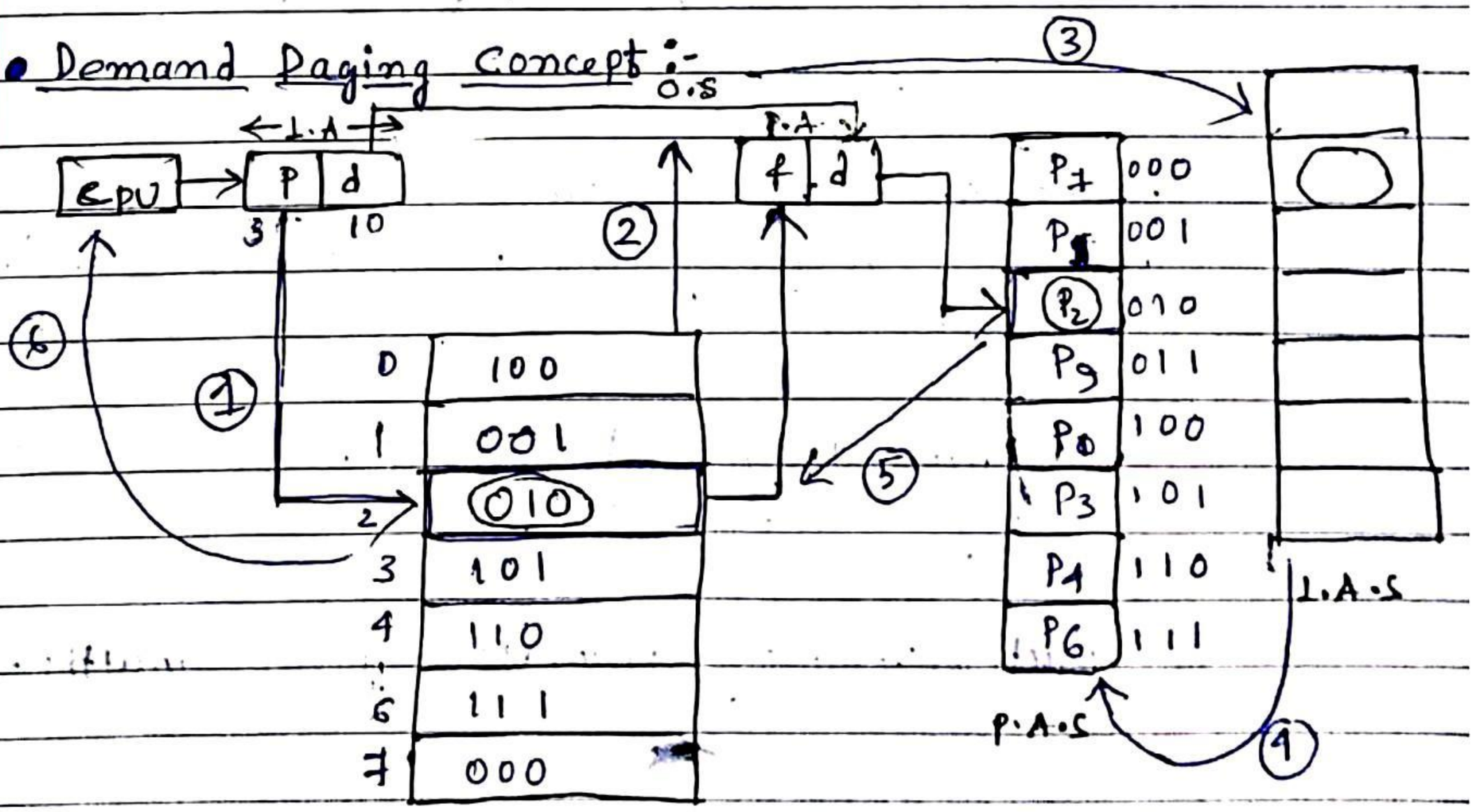
Example :- If program cige = 200KB
And Arailable P.A.S = 100KB.

- virtual Memory will be implemented Uring -
  1. Demand paging.
  2. Demand segmentation.

→ If we are Ring paging Technique as memory manegement technique ~~we~~ we use demand paging.

→ If we are ring segmentation as 'Memory Nanegement technique, we Ree demand segmentation.

- Demand Paging Concept :-



- Demand paging Conclusion :

Step 1):- The CPU is trying to access a page which is not available in the main memory, hence page fault occurs.

Step 2):- The program execution will be stopped and the signal will be send to the O.S regarding the page fault. Then the O.S authenticates the validity of the process.

Step 3):- Then O.S will be search for required page in the L.A.S.

Step 4):- The required page will be brought from L.A.S to P.A.S, The page replacement Algorithms will be used, for decision making of replacing of page in the P.A.S.

Step 5):- The page table will be uploaded accordingly.

Step 6):- Signal will be sent to CPU to continue the program execution.

Then the CPU will access the required page from the main memory and it will continue the program execution.

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

→ virtual memory increases degree of multiprogramming.

• Demand paging performance:

→ loading the page into memory on Demand.
(whenever the page fault occurs)

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

→ If page fault service time = S
Main memory access time = M.
page fault rate = P.
Then $\boxed{EMAT = P*S + (1-P)M}$

As S >> M.
page table access time is negligible in comparision to S, so we ignore it for page fault.

Virtual memory — 8-1

Consider a system which has main memory access time = 10ns (m)
page fault service time = 200 ns. page hit ratio is 95 %
What is EMAT. (S) (P)

→ $\boxed{EMAT = P*S + (1-P)*m}$

(Page fault)
P = 1 − 0.95
= 0.05

$\Rightarrow 0.05 * 200 + (1-0.05) * 10$

$= 2 * 95 + 0.5 * 10 \Rightarrow 10 + 9.5$

$= 190 + 5$ EMAT $\Rightarrow$ 19.5 ns.

$= 195$ ns.

* $1\,ms = 10^6\,ns$

Q-2

Suppose if an instruction takes i μsecond and additional j μsec if the page fault occures. What is the effective Instruction Access time. If the page fault occures on an average for every K instruction.

(a) $i + \frac{j}{K}$ (b) $j + \frac{i}{K}$ (c) $K + \frac{i}{j}$ ✓ (d) $i + \frac{j}{K}$.

→ $m \Rightarrow i$ — (1)

$s \Rightarrow i + j$ μsec — (2)

$p \Rightarrow \frac{1}{K}$ — (3)

$EMAT = p * s + (1-p) * M$

$= \frac{1}{K}(i+j) + (1-\frac{1}{K}) * i$

$= \frac{(i+j)}{K} + \frac{(K-1)}{K} * i$

$= \frac{1}{K}(i+j+Ki-i)$

$= \frac{j}{K} + i$

$EMAT = i + \frac{j}{K}$

GATE-2011 Q-3 The page fault service time is 10 ms and the main memory access time is 20 ns. If one page fault is generated for every 10^6 memory access. Then what is the effective Access time for memory.

(a) 21 ns (b) ✓ 30 ns (c) 23 ns (d) 35 ns.

→ $s \Rightarrow 10ms \Rightarrow 10^7$ ns.

$m \Rightarrow 20ns$.

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

$EMAT = p * s + (1-p) * m$

$= \frac{1}{10^6} * 10^7 + (1-\frac{1}{10^6}) * 20$

$= 10 + 20 - \frac{20}{10^6}$

$= 30 - \frac{20}{10^6} \Rightarrow 30 - 0$

$\approx 0 \Rightarrow 30$.

Q-4

Consider a demand paging environment and it takes 8ms to service a page fault, If either an empty frame is available or replaced page is not modified and it takes 20ms, If the replaced page is modified. Assuming main memory access time is 1ms. Further Assume that page to be replaced is modified 70% of time. Then what is the maximum Acceptable page fault rate to get an EMAT not more than 2ms.

(a) P = 0.32 (c) P = 0.064

(b) P = 0.16 (d) P = 0.72.

→

s → 8 ms (pg modified) ✗ } 70%
  → 20 ms (pg modified) ✓ }

P = ?

m → 1 ms

E.MAT = P * s + (1-P) * m

≤ P * [(.70 * 8ms) + (.30 * 20ms)] + P(1-P) 1. ≤ 2

≤ 15.4P ≤ 1.

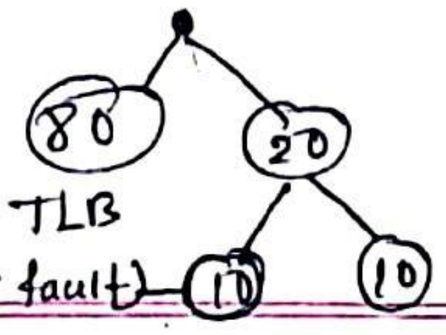
P ≤ 0.069.

Q-5 Consider a system where the page fault service time is 200 ms and main memory access time = 10 ms. The TLB is added to improve the performance. The 80% references are found in the TLB and that of remaining 10% cause page fault. The TLB access time and page table access time are negligible, then what is EMAT.

(a) 12.8 ms (b) 13.8 ms (c) 14.8 ms (d) 15.8 ms

→ s → 200 ms

m → 10 ms.



$$EMAT = .80 * (10ms) + .20 [p*s + (1-p)*m]$$

$$= 8 + .20 [.1*200 + .9*10]$$

$$= 13.8 ms$$

Page Replacement : (Algorithms) -

1) FIFO → First in First out.
2) optimal page Replacement.
3) Least Recently Used (LRU).
4) Most Recently Used (MRU)

Page replacement FIFO -

Question-1

Reference string :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Assume 4 frames are allocated 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 fault)

10 page faults

Q-2

Reference string :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1.

Assume 3 frames are allocated to the process.

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
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

(Page fault)

total page fault – 15 ✓

Q-3

Reference String: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

(1) Using 3 frames.

1	2	3	4	1	2	5	1	2	3	4	5
		3	3	3	2	2	2	2	2	4	4
	2	2	2	1	1	1	1	1	3	3	3
1	1	1	4	4	4	5	5	5	5	5	5
*	*	*	*	*	*	*			*	*	

total page fault – 9.

(2) using 4 frames.

1	2	3	4	1	2	5	1	2	3	4	5
			4	4	4	4	4	4	3	3	3
		3	3	3	3	3	3	2	2	2	2
	2	2	2	2	2	2	1	1	1	1	5
1	1	1	1	1	1	5	5	5	6	4	4
*	*	*	*			*	*	*	*	*	*

total page fault – 10

• CONCLUTION –

In Reference string 1, 2, 3, 4, 2, 5, 1, 2, 3, 4, 5.

3 frames → 9 page fault.

4 frames → 10 page fault.

BELADY'S ANAMOLY:

By increasing the no of frames to the process, we should get less no of page fault, but instead they are increasing. This problem is called as Belady's Anamoly.

Page Replacement (LRU):
(Least Recently Used)

Q-1

[In the event of page fault replace the page which is least recently used]

Reference String :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

(1) Using 3 frames.

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

page fault → 12.

Page Replacement (Optimal PR):

Q-1 [In the event of page fault replace the page which is not used for longest duration of time in future].

Reference String :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1.

(1) Using 4 frames.

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

FCFS

Total page fault → 8

- Page Replacement →
- Most Recently Used (MRU):

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

Q-1

Reference string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1.

(i) 4 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	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
	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
*	*	*	*		*	*	*		*	*	*	*			*				

Total page fault → 12

- Q- practice (Record)

The Address Sequence generated by tracing a particular program executing in a pure Demand paging system with 100 records per page with 1 free memory frame is re-worded as follows.

0100, 0200, 0400, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0370.

What is the number of page fault?

(a) 7 (b) 8 (c) 9 (d) 10.

Records →	page no
0 - 99	0
100 - 199	1
200 - 299	2
300 - 399	3
400 - 499	4
500 - 599	5

here frame → 1

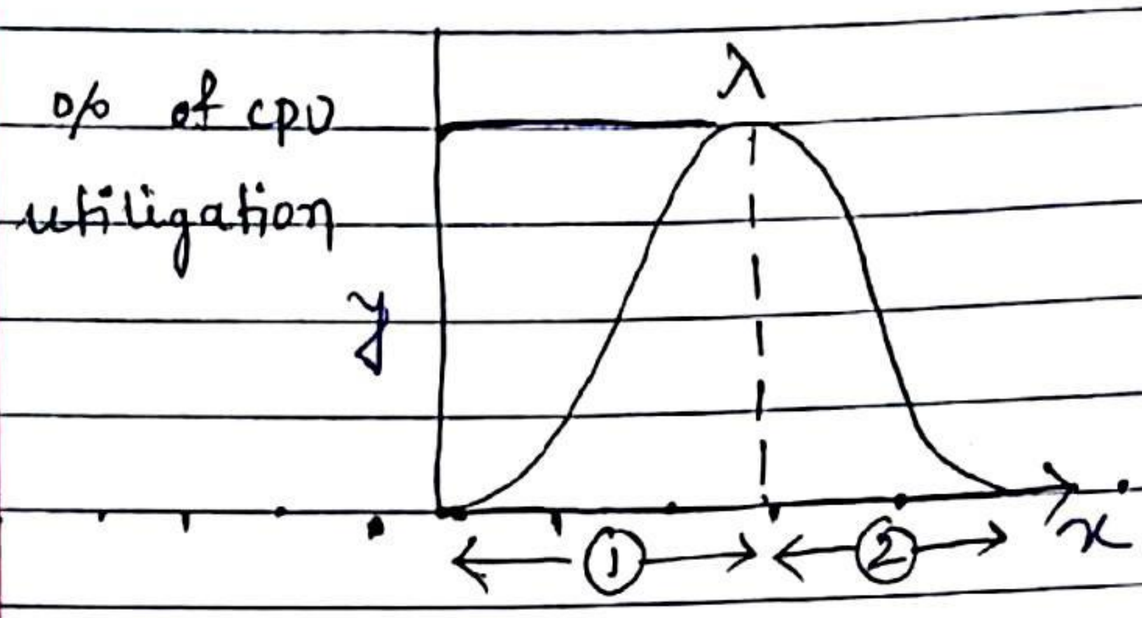
Reference Strings -

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

1	2	4	4	5	5	5	1	2	2	2	3	3
1	2	4	4	5	5	5	1	2	2	2	3	3
*	*	*		*			*	*			*	

Total page fault → (7)

• Thrashing Introduction :-



→ In the initial degree of multiprogramming the cpu utiligation is very high and the system resources are utiliged. 100%.

→ If you further increase the degree of multiprogramming after some extent of point i.e λ the cpu utiligation will drastically fall down and time taken to complete every request will increase and system will spend more time only in page replacement. This is called Thrashing.

Example - let free frames = 300.

Phase (1) - # of process = 100 (3)

Phase (2) - # of process = 300 (1)

• Threshing Causes And Recovery -

(*) Causes :-

(1) High degree of multiprogramming.

(2) Lack of frames.

(*) Recovery from Thrashing :-

(1) Don't allow the system to go into thrashing. Instruct the long term shedular, not to bring proces beyond the point of λ.

(2) Allow the system to go into thrashing, then insert the mid term shedular to suspend the processes so that system may recover from thrashing.

- Thrashing practice **Question – 1**

Consider a system with below statistics observation.

1) CPU utiligation → 10 %.
2) paging on disk → 90 %. → (page replacement).

Then which one of following will increase the cpu utiligation=

1) Install the faster CPU → NO
2) Install the faster disk → NO
3) Increase degree of multiprogramming → NO
4) Decrease the degree of multiprogramming → yes.
5) Install more main memory → yes
6) Decreas the page sige → NO
7) Increase the page sige → yes.