MEMORY MANAGEMENT

M.m/m also called as

- main memory
- primary memory
- physical memory
- RAM (volatile)

Functionality: allocating and de-allocating the memory to the processes.

Goal: The efficient utilization of memory by minimizing the internal and external fragmentation.

$2^8 = 256$

$2^9 = 512$

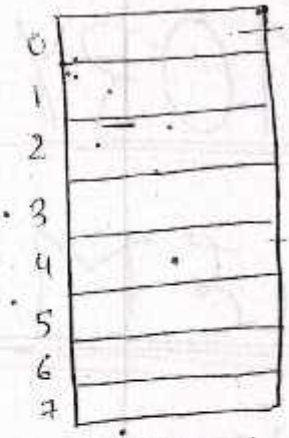
$2^{10} = 1024 \simeq 10^3 = 1K$

$2^{20} = 1M$

$2^{30} = 1G$

$2^{40} = 1T$

Memory
- Word Addresable
  - Eg: 2kw
- Byte Addresable
  - Eg: 2KB



each word of 16 bits size
(there are 8 words in m/m)

Memory

Capacity of memory = # of words in memory * word size

= 8 * 16 bits

= 8 * 2 Bytes

= 16 Byts

To represent all 8 words in the m/m, we have required (3 bit) of binary bit.

Q. Consider the system which has 256 KW in the m/m and each word has the size of 64 bits. Then what is the capacity of m/m in Bytes?

Sol: capacity of m/m = 256 KW × 64 bits

$= \frac{256 \times 64}{8}$

$= 256 \times 8K$ or $= 2^8 \times 2^3 \times 2^{10}$

$= 2048 KB$ $= 2^{21} = 2^1 2^{20}$

$= 2MB$ Ans.

Q. Consider a system which has 512 MW in the m/m & each word has the size of 16 Bytes Then. what is the capacity of m/m in Bytes?

Sol: capacity of memory = 512M × 16B

$= 2^9 \times 2^{20} \times 2^4 B$

$= 2^3 \cdot 2^{30} B$

$= 8 GB$ Ans.

Q Consider a system where the 32 Binary bits are used to represent all words of the memory. And each word is having the size of 32 bits. Then what is the capacity of m/m in bytes?

Sol:-

$32 = 2^5$

Capacity of memory = $2^{32} \times (32/8)$

$= 2^2 \cdot 2^{30} \times 4\ B$

$= 16\ GB$ Ans.

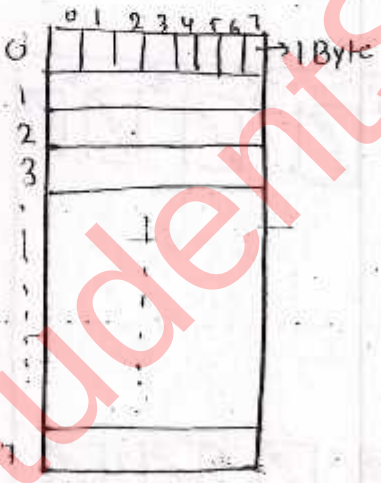
RAM CHIP IMPLEMENTATION

Ram chip size = 128B

Organize the main m/m capacity = 16KB

Q1) # of RAM chip required?

Q2) Draw the memory organization map?



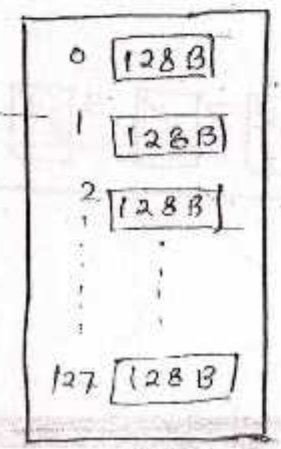
RAM chip = 128B

# of Ram chip = $\frac{\text{m/m capacity}}{\text{Ram chip size}}$

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

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

$= 2^7 = 128$



16KB main m/m. map.

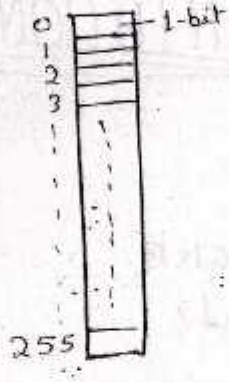
Q. RAM Chip size = 256 × 1-bit
Organize the main m/m capacity = 32 MB
Q1), Q2), same ?

Sol: # of RAM chip = $\frac{32\,MB}{256 \times 1\text{-bit}}$

$= \frac{2^5 \times 2^{20} \times 2^3 \text{ bit}}{2^8 \times 1\text{-bit}}$

$= 2^{20}$ chip — 1M ✓ ~~1MB~~ ×

RAM chip



# of rows & columns -

no. of rows = $\frac{32\,M \times B}{256 \times (1\text{-bit})}$

$\frac{2^5 \times 2^{20}}{2^8}$

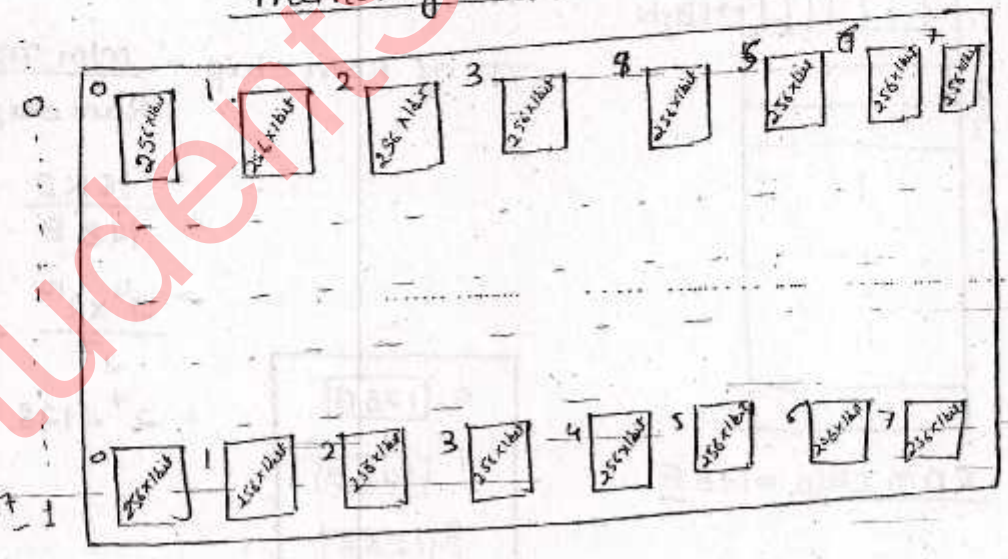
$= 2^{17}$

no. of column = $\frac{B}{1\text{-bit}}$

$= \frac{8\text{-bit}}{1\text{-bit}}$

$= 8$

memory map



Q. RAM chip size = 512 x 2-bits
organize the main m/m capacity = 64 GB
same?

Sol.

# of ram chip required = $\frac{64\ GB}{512 \times 2\ bits}$

$= \frac{64 \times 2^{30} \times 2^3}{2^9 \times 2^1}$

$= \frac{2^6 \times 2^{30} \times 2^3}{2^{10}}$

$= 2^{29} = 2^9 \cdot 2^{20} = 512M$

no. of rows & column.

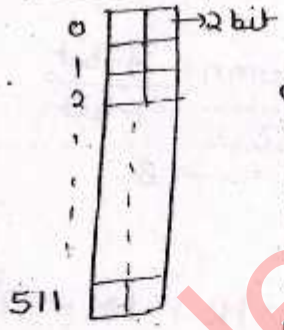
no. of rows = $\frac{2^6 \times 2^{30}}{2^9}$

$= 2^{27}$

$\frac{64\ G \times B}{512 \times 2\ bit}$ no. of column = $\frac{8\text{-bit}}{2\text{-bit}}$

$= 4$

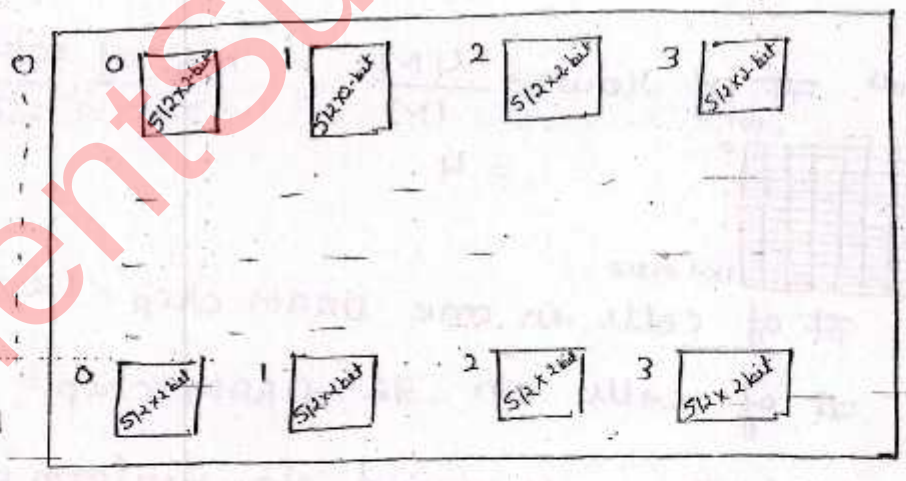
Ram chip



512 words
of each
2 bit
size

$2^{27} - 1$

memory map organization.



Gate 2012

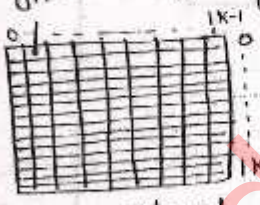
Ques: A main memory unit with capacity of 4MB is built using 1M × 1-bit DRAM chips. Each DRAM chip has 1K rows of cells with 1K cells in each row. The time taken for a single refresh operation is 100 ns. The time required to perform one refresh operation on all the cells in the memory unit is?

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

Sol:- # of DRAM chip $= \frac{4MB}{1M \times 1\text{-bit}}$

$= \frac{2^2 \cdot 2^{20} \times 2^3 \text{ bit}}{2^{20} \times 1\text{-bit}}$

$= 2^5 = 32$



# of rows $= \frac{4M}{1M} = 4$

no. of column $= \frac{8\text{-bit}}{1\text{-bit}} = 8$

# of cells in one DRAM chip $= 1K \times 1K = 1M = 2^{20}$

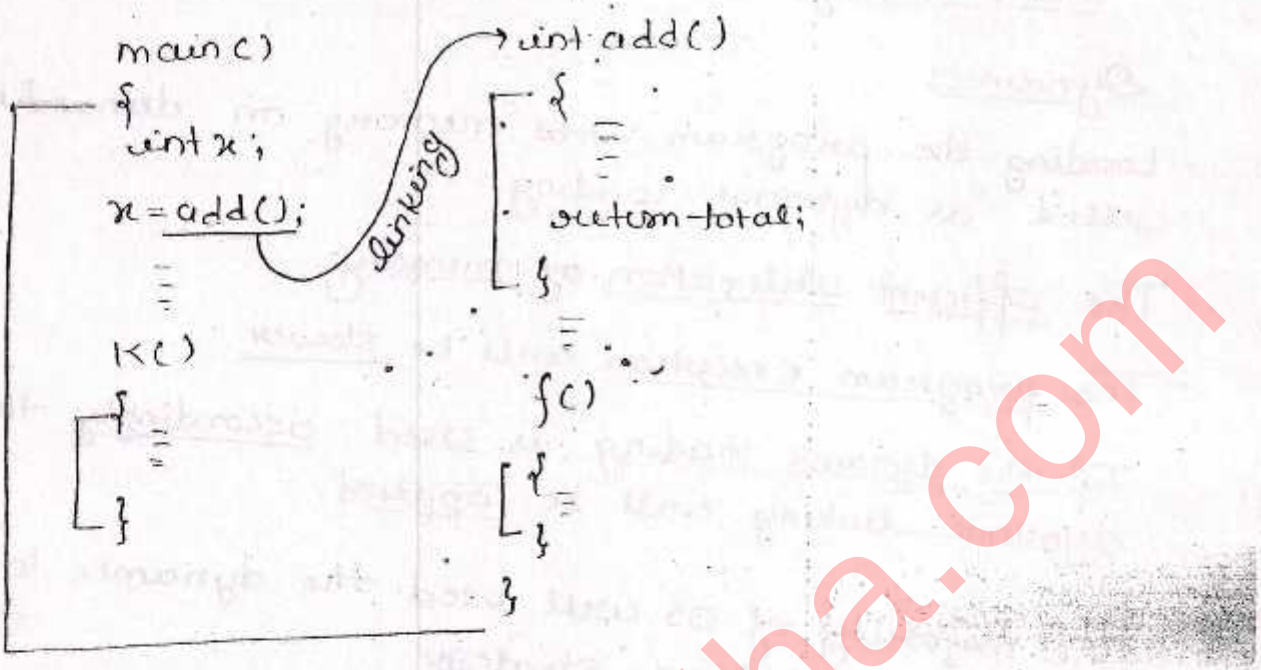
# of cells in 32 DRAM chip $= 32 \times 2^{20}$

Time required to perform of oper^n on all cells $= 32 \times 2^{20} \times 100\,ns$

$= 3200 * 2^{20}$ ns A.

Option (d) is correct

LOADING, LINKING & ADDRESS BINDING



Loading: Bringing the program from the secondary memory to the main memory.

Linking: linking all the modules, functions or methods of the program in order to execute is called as linking.

Loading and linking are categorised in two types

1. Static
2. Dynamic

Static - Loading the entire program in the memory before start of program execution is called as static loading.

- Inefficient utilization of memory bcz whether it is required or not required, the entire program is brought into main m/m.
- The program execution will be faster

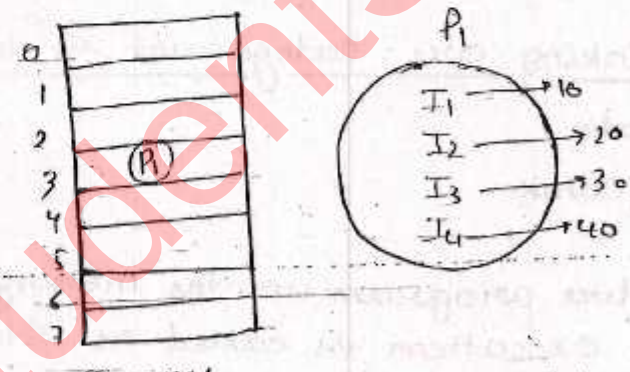
- If the static loading is used accordingly, the static linking will be applied.

Dynamic:

- Loading the program into memory on demand is called as dynamic loading.
- The efficient utilization of memory.
- The program execution will be slower.
- If the dynamic loading is used, accordingly the dynamic linking will be applied.
- The majority of OS will used the dynamic loading & dynamic linking strategy.

Address Binding:

- Association of program instructions and data to the actual physical memory locations is called as address binding.



Memory

PC
10 20 30 40 ...

After dumping or loading the process into memory. Address Binding will be done. means associate cirⁿ to actual physical address

Address Binding

- Compile time AB → compiler } Theoritically used
- Load time AB → loader.
- * Execution time AB - practicaly used (or)

Dynamic AB → processor

1) Compile time Address Binding:

- If the compiler takes the responsibility of the association of the program instructions and the data to the actual physical m/m locations it is called as Compile time address binding.

- The compile time address binding will be done before loading the program into main memory.
- The compiler needs to interact with the OS m/m manager to perform Compile time AB.

2) Load time Address Binding:

- This type of address binding will be done after loading the program into main m/m.
- This type of A.B. will be done by OS m/m manager (ie loader)

3) Execution Time Address Binding.

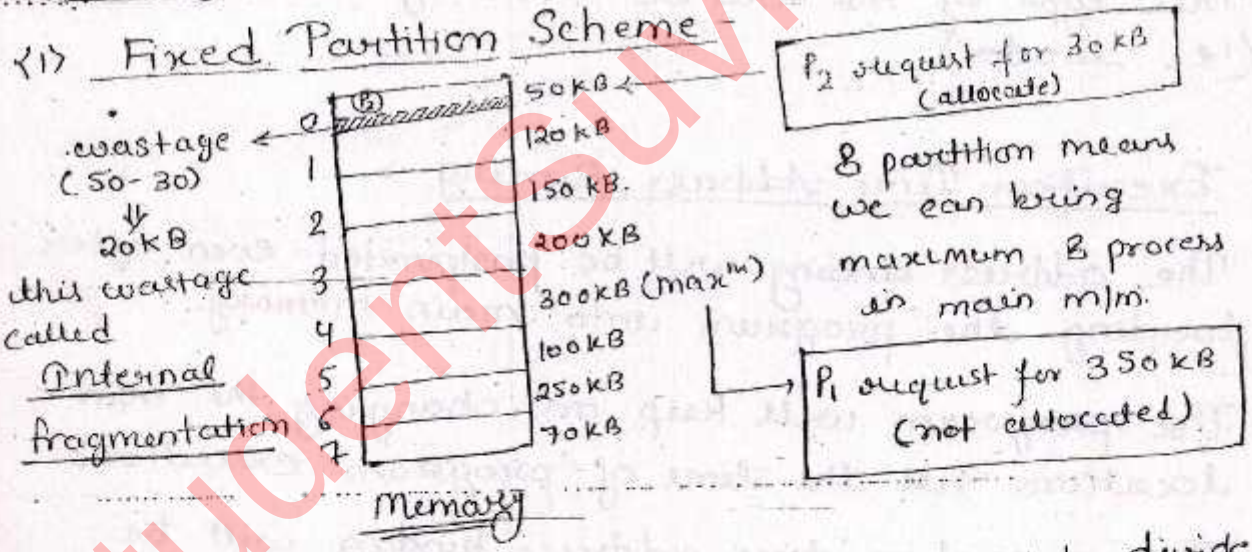
- The address binding will be postponded even after loading the program into main memory.
- The program will keep on changing the m/m locations till the time of program executions.
- The execution time address binding will be done at the time of program execution.
- This type of address binding will be done by the processor.
- The majority of the OS will practically implement the execution time address binding.

Memory Management Techniques

- Contiguous
  - Fixed partition scheme
  - Variable partition scheme
- Non-contiguous (Imp)
  - paging
  - multilevel paging
  - Inverted paging
  - segmentation
  - segmented paging

Contiguous

(1) Fixed Partition Scheme



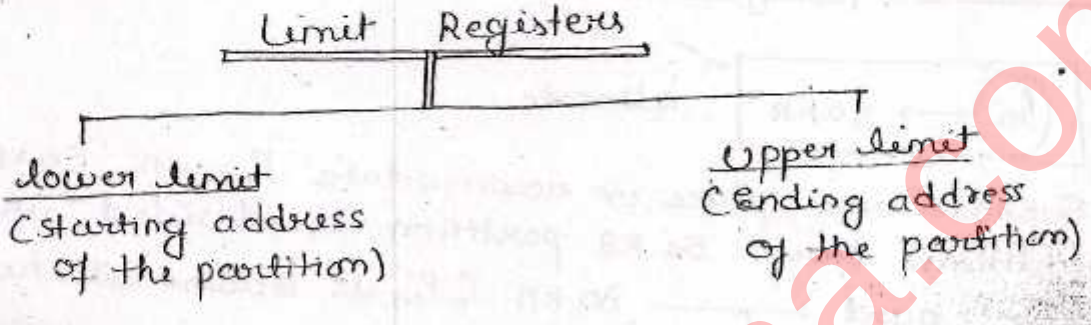
- In the fixed partition scheme, the m/m will be divide into fixed no. of partitions.
- fixed means the no. of partitions are fixed but not the size of the partition.
- In one partition, only one process will be accomodated.

- Degree of multi-programming is restricted by the no. of partitions in the memory.

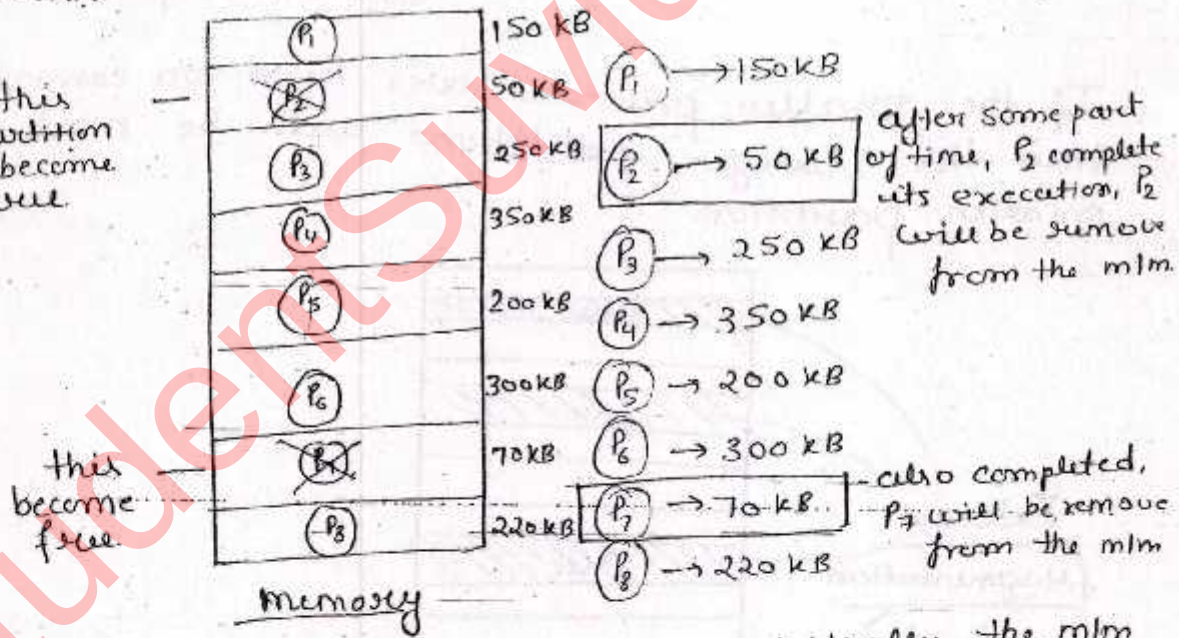
To avoid these two pbs we will move on Variable Partition Scheme

- The maxm of size of process is restricted by the maxm size of partition.

- The every partition is associated with the limit registers.



(2) Variable partition Scheme -

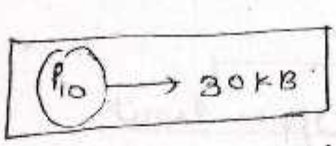


- In the variable partition scheme, initially the m/m will be single continuous free block.

- whenever the request by the process comes in the accordingly partition will be made.

- total we have (50+70) = 120 kB of memory free.

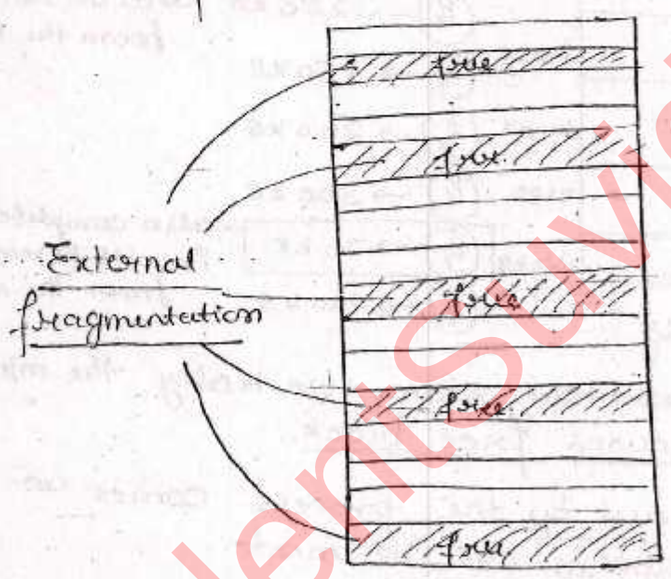
- when Process P_9 is requesting for allocate 100KB of memory — X not allocated.
- Even though memory is freely available, but we can't allocate them to process P_9 bcz m/m is not contiguous. This is known as External fragmentation



allocate

Suppose we place or accomodate P_{10} in 50 KB partition. Now 50 KB partition is divided into two part
- 30 KB (P_{10} is accomodate here)
- 20 KB (freely available) don't say it is internal fragmentation.

- If the smaller processes are keep on coming then the larger partitions will be made into smaller partition.



To avoid the external fragmentation the below techniques are followed -

1) Compaction -

⇒ Moving all the process towards the top or towards the bottom to make the free available memory in a safe continuous place.

⇒ The compaction is undesirable to implement bcz it interrupts all the running processes in the m/m.

2) Implementing non-contiguous m/m manag^{n}. techniques

⇒ to avoid problem of external fragmentation.

PARTITION ALLOCATION METHOD

- When more than one partition is freely available to accomodate the request of the process, then decision making of selecting a partition will be done by partition allocation method.

(i) First fit -

- Allocate the process in a partition, which is the first sufficient partition from the top of the m/m.

(ii) Best fit -

- Allocate the process in a partition, which is the smallest sufficient partition among the free available partitions.

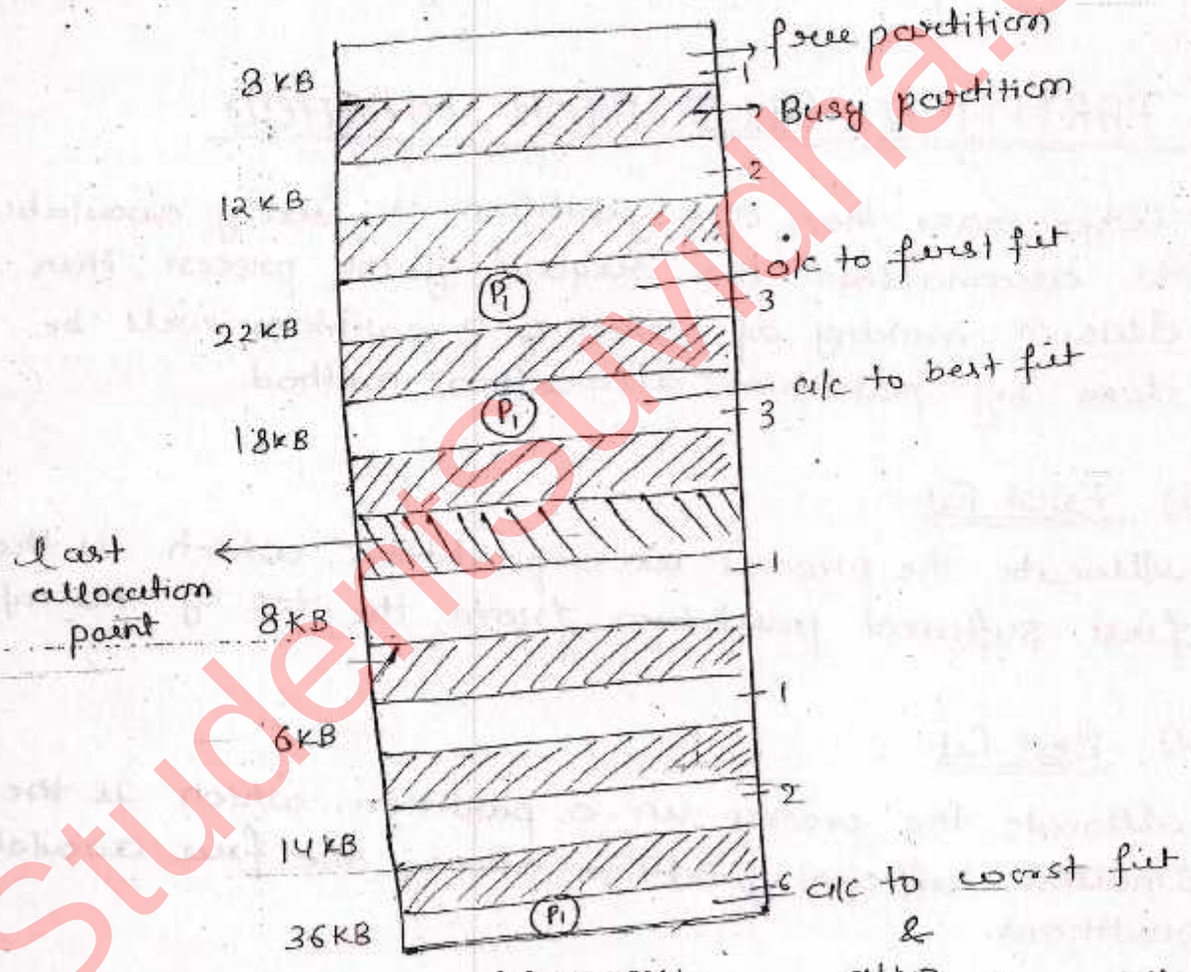
- To find out the smallest sufficient partition, it requires to search all the free available partition in the m/m.

3) Worst fit

- Allocate the process in a partition, which is the largest sufficient among the free available partitions.
- To find out the largest sufficient partition it also requires to search all the free available partitions in the m/m.

4) Next fit -

- The next fit also works like first fit but it will search for the first sufficient partition from the last allocation point.



First fit - P_1 → 16KB → 22KB

Best fit - P_1 → 16KB → 18 KB

Worst fit → 36 KB

next fit → 36 KB

Q) How many successive requests of 6KB will be satisfied by using the first fit and assuming variable partition scheme?

a) 16

b) 17

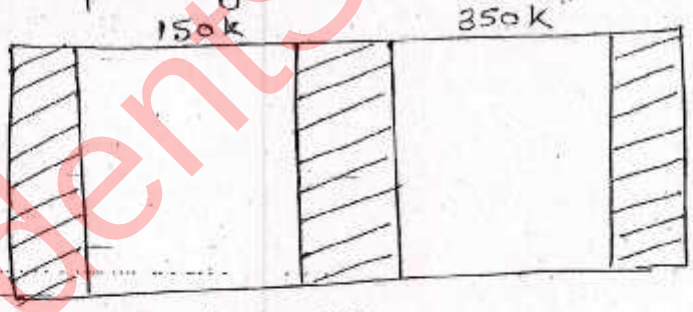
c) 18

d) 19 (1+2+3+3+1+1+2+6)

e) 20

Gate

Q. The request from the processes are 300K, 25K, 125K, 50K respectively



memory

The above requests can be satisfied by using? (assume variable partition scheme)

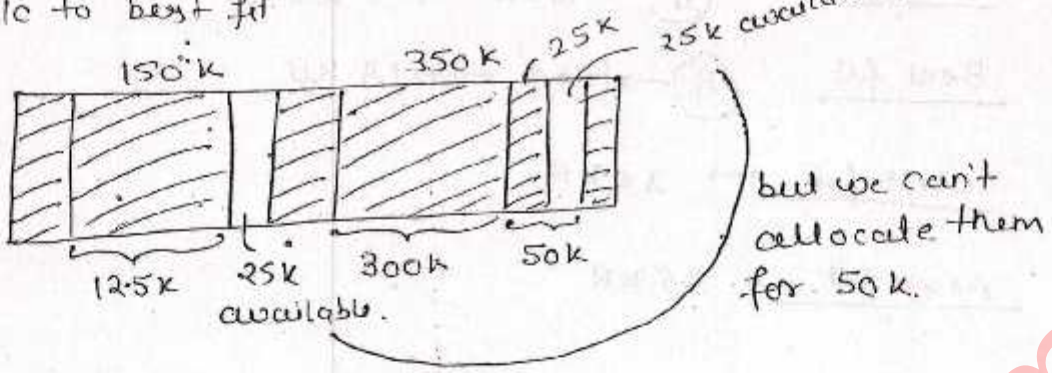
a) Best fit but not first fit

b) first fit but not best fit

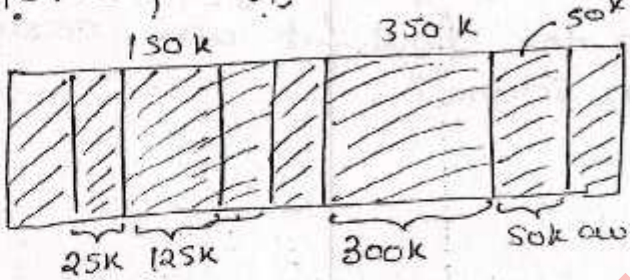
c) Both first fit & best fit

d) neither first fit nor best fit

Sol: a/c to best fit



a/c. to first fit



Note - If the context switching is disabled then the process is not allowed for pre emption.

Non-Contiguous Mm Techniques

PAGING

1) Logical Address Space (L.A.S) (or) Virtual Address Space (V.A.S) - represented in the form of < words / Bytes

2) Logical Address (L.A) (or) Virtual Address (V.A) - (bits)

3) physical Address space (P.A.S) < words / Bytes

4) physical Address (P.A) - (bits)

→ LA = 39 bits

LAS = 2^{39} W = $2^9 \cdot 2^{30}$ = 512 GW

→ LAS = 64MW

LA = 64M = $2^6 \cdot 2^{20}$ = 2^{26} ⇒ 26 bits

→ LAS = 128 KB

LA = $2^7 \cdot 2^{10}$ B = 2^{17} ⇒ 17 bits

→ PA = 28 bits

PAS = 2^{28} = 256MW

→ PAS = 32GW

PA = $2^5 \cdot 2^{30}$ = 35 bits

CO - CPU generates physical address

OS - CPU generates logical address ✓

→ actual process is reside in physical address. physical address is greater

→ logical address is always greater than physical address.

→ CPU can't directly access the "ins" residing in the main m/m. / physical address with the generated logical address bcz address mis-match

$$LA > PA$$

→ we need a mechanism to map processor generated logical address to physical address.

→ The technique of mapping the processor generated logical address to physical address is called as paging.

Paging

LA = 13 bits → LAS = 2^{13} = 8KW

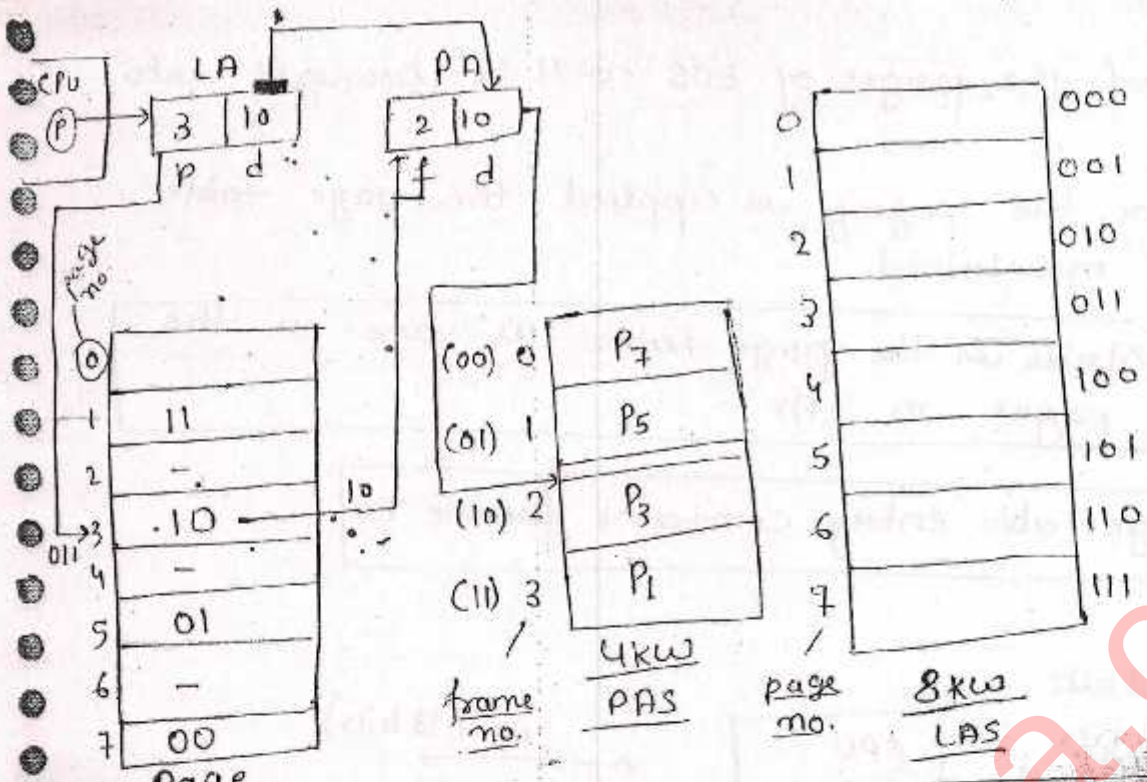
PA = 12 bits → PAS = 2^{12} = 4KW

logical Address Space will be divided into equal size pages.

Assume page size = 1KW

$$\#\text{ of page} = \frac{LAS}{\text{page size}} = \frac{8KW}{1KW} = 8$$

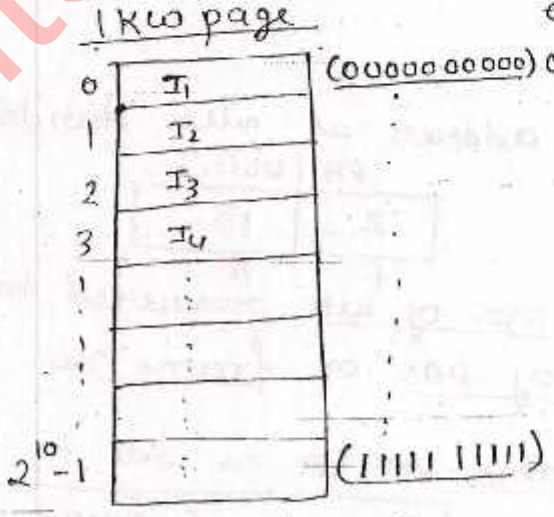
The physical address space will be divided into equal size frames.



The page size will be always same as frame size

$$\#\text{ of frame} = \frac{PAS}{\text{frame size}} = \frac{4KW}{1KW} = 4$$

1KW page means, page is having 2^{10} words. (to identify every words of 1 page we require 10 binary bits



PAS is actual main m/m allocated to the process.

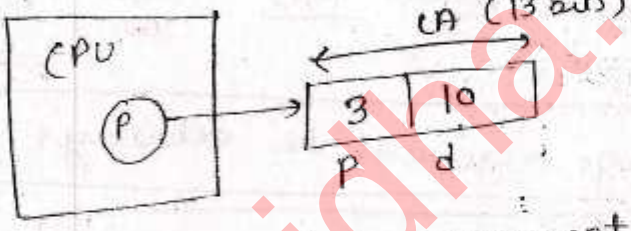
⇒ Some of the pages of LAS will be brought into PAS.

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

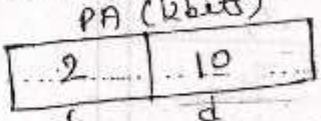
⇒ No. of entries in the page table is same as the no. of pages in LAS.

⇒ The page table entry contains frame no

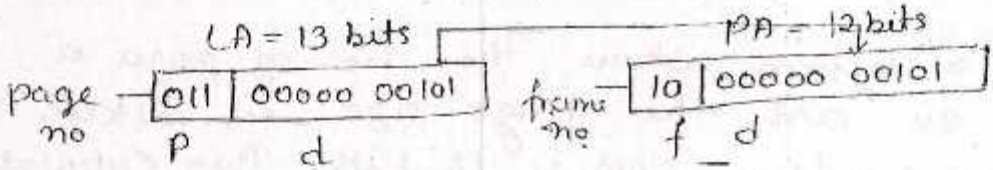
⇒ LA = 13 bits
PA = 12 bits
page size = 1KW



- 'p' is the no of bits required to represent pages of LAS or page no.
- 'd' is the no. of bits required to represent the page size. or word no of the page or page offset.
- physical address is also devided into two parts.



- f is the no. of bits required to represent the frames of PAS or frame no.
- d is same in LA as well as PA bcz Page size = frame size.



Important points:

- The paging is with respective to every process & every process will have its own page table
- The page tables of the processes will be stored in the main m/m. (PA's).
- There is no external fragmentation in paging.
- The internal fragmentation exists in the last page. The internal fragmentation in the paging is considered as $\boxed{P/2}$ where P is page size.

LAS = 8KW
page size = 3KW

$$\#\text{ of pages} = \frac{8KW}{3KW} = 2.6$$

= last page may not be full always. it may be more than or less than page size.

⊙ Consider a system with the logical address = 27 bits and the physical address = 21 bits. The page size is 4KW. Then calculate the no. of pages and the no of frames.

Sol: LA = 27 bits → LAS = $2^{27} \Rightarrow 2^7 \cdot 2^{20} = 128MW$

page size = 4KW

$$\#\text{ of pages} = \frac{128MW}{4KW} = \frac{2^7 \cdot 2^{20}}{2^2 \cdot 2^{10}}$$

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

$$= 32K$$

$$\#\text{ of frames} = \frac{2^{21}W}{4KW} = \frac{2^1 \cdot 2^{20}}{2^2 \cdot 2^{10}} = 2^9 = 512$$

Q. Consider a system where the no. of pages is equal to 2K and the page size is 4Kw. The physical address (PA) is 18 bits. Then calculate the logical address & the no. of frames in the PAS.

Sol:- # of pages = $\frac{LAS}{page\ size}$.

$$2K = \frac{LAS}{4Kw}$$

$$\Rightarrow LAS = 8K \times Kw = 8Mw$$

PA = 18 bits

$$PAS = 2^{18} = 2^8 \cdot 2^{10} = 256Kw$$

$$\# \text{ of frame} = \frac{PAS}{page\ size}$$

$$= \frac{256Kw}{4Kw} = \frac{2^8}{2^2} = 2^6 = 64$$

$$LAS = 8Mw$$

$$LA = 2^3 \cdot 2^{20} \Rightarrow 23 \text{ bits}$$

Q. Consider a system with LAS of 128Mw and PA = 24 bits. The PAS is divided into 8K frames. Then what is the page size & how many pages in the LAS?

Sol:- PA = 24 bits $\rightarrow$ PAS = 2^{24}

$$\# \text{ of frame} = \frac{PAS}{page\ size}$$

$$8K = \frac{2^{24}}{PS} \Rightarrow PS = \frac{2^{24}}{2^{13}} = 2^{11} = 2Kw$$

# of pages = $\frac{128MW}{2KW}$

$= \frac{2^7 \cdot 2^{20}}{2^1 \cdot 2^{10}} = 2^6 \cdot 2^{10} = 64K$

Gate 2002

Q. Consider a system with the logical address = 32 bits and PAS = 64MB and the page size is 4KB The memory is byte addressable. The page table entry size is 2 Bytes. Then what is the approximate size of page table in Bytes.

Sol:- PAS = 64MB

PA = $64M \rightarrow 2^6 \cdot 2^{20} = 26$ bits

LA = 32 bits

LAS = $2^{32} = 4GB$

# of pages = $\frac{4GB}{4KB} = \frac{G}{K} = \frac{2^{30}}{2^{10}} = 2^{20} = 1M$

# of frames = $\frac{64MB}{4KB} = \frac{16M}{K} = \frac{2^4 \cdot 2^{20}}{2^{10}} = 16K$

# of entries in page table = # of pages = 1M

Size of page table = $1M \times 2B = \underline{2MB}$.

page table size =	# of entries in page table	*	size of page table entry

Q Consider a system having a page table with 4K entries. LA = 29 bits. What is the PA? If the system has 512 frames.

Sol:- # of pages = # of entries = 4K

LA = 29 bits

LAS = $2^{29} \rightarrow 2^9 \cdot 2^{20}$ = 512 MW

$$\#\text{ of pages} = \frac{\text{LAS}}{\text{page size}}$$

$$\Rightarrow \text{page size} = \frac{\text{LAS}}{\#\text{ of pages}}$$

$$= \frac{512\text{ MW}}{4\text{K}}$$

$$= \frac{2^9 \cdot 2^{20}\text{ W}}{2^2 \cdot 2^{10}} = 2^7 \cdot 2^{10}\text{ W}$$

$$= 128\text{ KW}$$

$$\#\text{ of frames} = \frac{\text{PAS}}{\text{PS}}$$

PAS = # of frame x PS

= 512 x 128 KW

= $2^9 \cdot 2^7$ KW

= 2^{16} KW

PA = $2^{16} \cdot K = 2^{16} \cdot 2^{10}$ = 26 bits

Another method

LA = 29 bits

12	17
p	d

PA = 26 bits

9	17
f	d

Q Consider a system with LAS = 256 MB
PA = 24 Bits. The PAS is divided into 8 KB frames. The m/m is byte addressable. Then how many pages are there in LAS.

Sol: PA = 24 bits

$PAS = 2^{24} = 2^4 \cdot 2^{20} = 16MB$

~~# of frames~~ $= \frac{PAS}{PS}$

$PS = \frac{PAS}{\#\text{ of frames}}$

$\#\text{ of page} = \frac{256MB}{8KB}$

$= \frac{2^8 \cdot 2^{20}}{2^3 \cdot 2^{10}} = 2^5 \cdot 2^{10} = 32K$

Q. Consider a system with

$LAS = PAS = 2^{16} B = 2^6 \cdot 2^{10} B = 64KB$

$PS = 512 B$

The m/m is byte addressable. The page table entry size is 2B. The page table entry contains besides the other information like
1 bit → valid / Invalid
1 bit → reference
3 bits → page protection
1 bit → dirty bit

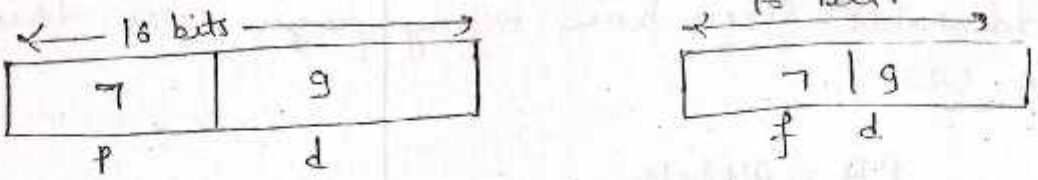
Then how many bits are still available in the page table entry to store ageing information?

Sol: $\#\text{ of pages} = \frac{LAS}{PS} = \frac{64KB}{512B} = \frac{2^6 \cdot 2^{10}}{2^9} = 2^7 = 128$

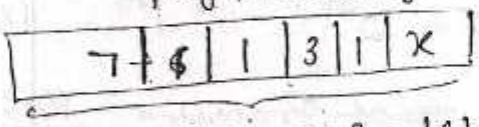
$\#\text{ of frames} = \frac{64KB}{512B} = 128$

LAS = 2^{16}

LA = 16 bits PA = 16 bits



page table entry



2B = 2x8 = 16 bit

16 = 7+6+x

x => 16 - 13 = 3 bit Ans.

Q. Consider a system with

LAS = PAS = 'S' Bytes

Page size = 'p' Bytes

page table entry size = 'e' Bytes

The m/m is byte addressable. Then what is the optimal value of page size by minimizing the m/m overhead of maintaing the page table size & internal fragmentation in the paging.

a) $p = \sqrt{2Se^2}$ c) $p = \sqrt{2Se}$

b) $p = \sqrt{2S^2e}$ d) $p = \sqrt{2(Se)^2}$

Sol: # pages = # frame = $\frac{S}{p}$

entries = # pages

size of page table = # entries * page table entry size

= $\frac{S}{p} \times e$ (in bytes)

Memory overhead = P.T. size + I.F in page size

$$= \frac{S}{P} \times e + \frac{P}{2}$$

$$\frac{d}{dP}\left(\frac{S}{P} \times e\right) + \frac{d}{dP}\left(\frac{P}{2}\right) = 0$$

$$\Rightarrow \frac{-Se}{P^2} + \frac{1}{2}\frac{d(P)}{dP} = 0$$

$$\Rightarrow \frac{1}{2} = \frac{Se}{P^2} \Rightarrow P^2 = 2Se \Rightarrow \boxed{P = \sqrt{2Se}}$$

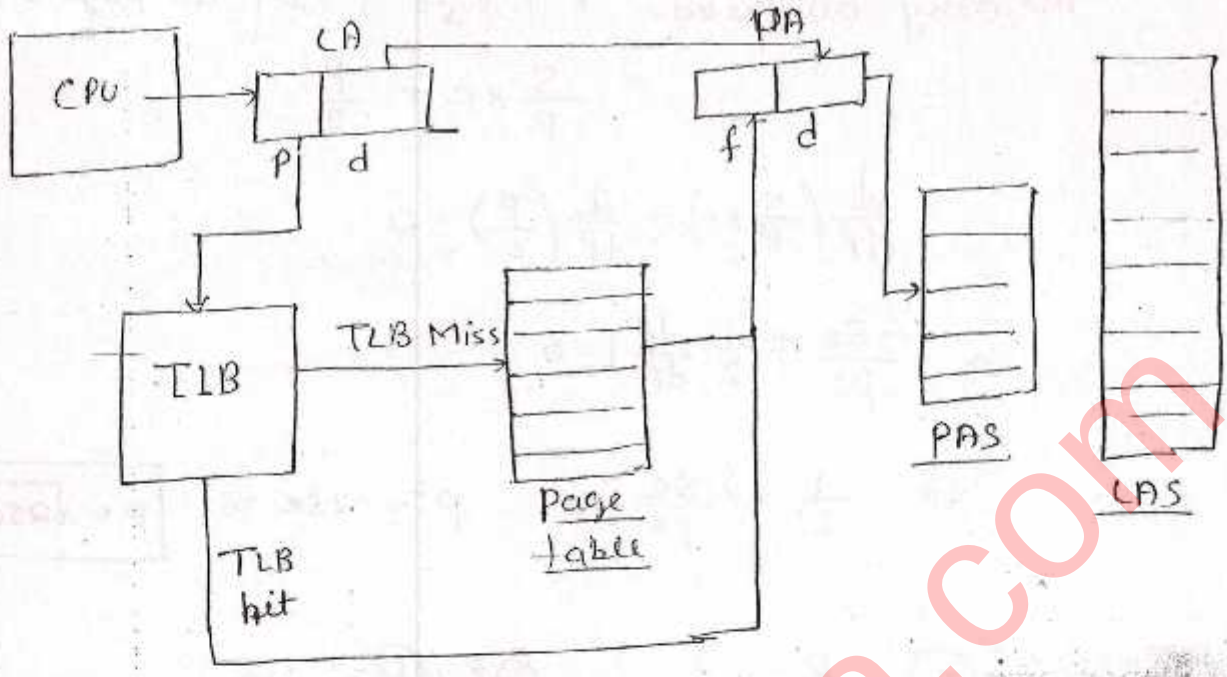
Performance Of Paging

- The main memory access time (MMAT) = 'm'
  and if the page tables are store in the main m/m.
- Effective Memory Access time (EMAT)

$$\boxed{EMAT = 2M}$$

  - 1M for accessing page table
  - 1M for accessing physical m/m

- If the translation look aside buffer (TLB) is added to improve the performance
- TLB contains frequently referred page no's & corresponding frame no's
- TLB is a h/w device, which is implemented by using associative registers.
- TLB will be placed before the page table.



→ The TLB access time = 'C'
and TLB hit ratio = 'x'
Effective Memory Access time (EMAT)

$$EMAT = x(C+M) + (1-x)(C+2M)$$

Q. Consider a system which has the main m/m access time = 100 ns and TLB access time is 20 ns. TLB hit ratio is 95%. then what is EMAT with TLB & w/o TLB?

Sol: EMAT w/o TLB = 2M = 200 ns Ans.

EMAT w. TLB = $0.95(20+100) + (1-0.95)(20+200)$

$= 0.95(120) + 0.05(220)$

$= 114 + 11$

$= 125$ ns. Ans.

Q. What hit ratio is required to reduce the effective MAT from 300ns w/o TLB to 250ns with TLB. The TLB access time is 60 ns

Sol:- EMAT w/o TLB = 300ns

$2M = 300ns$

$M = 150ns.$

EMAT w TLB = $x(C+M) + (1-x)(C+2M)$

$250 = x(60+150) + (1-x)(60+300)$

$250 = 210x + (1-x)360$

$250 = 210x + 360 - 360x$

$360x - 210x = 360 - 250$

$150x = 110$

$x = \frac{110}{150} = \frac{11}{15} = 0.73$ or 73%.

Q. Consider a system with

LA = 32 bits → LAS = $2^{32} = 4 \cdot 2^{30} = 4GW$

Page size = 4KW

page table entry size = 4B

Then what is the approximate size of page table in bytes?

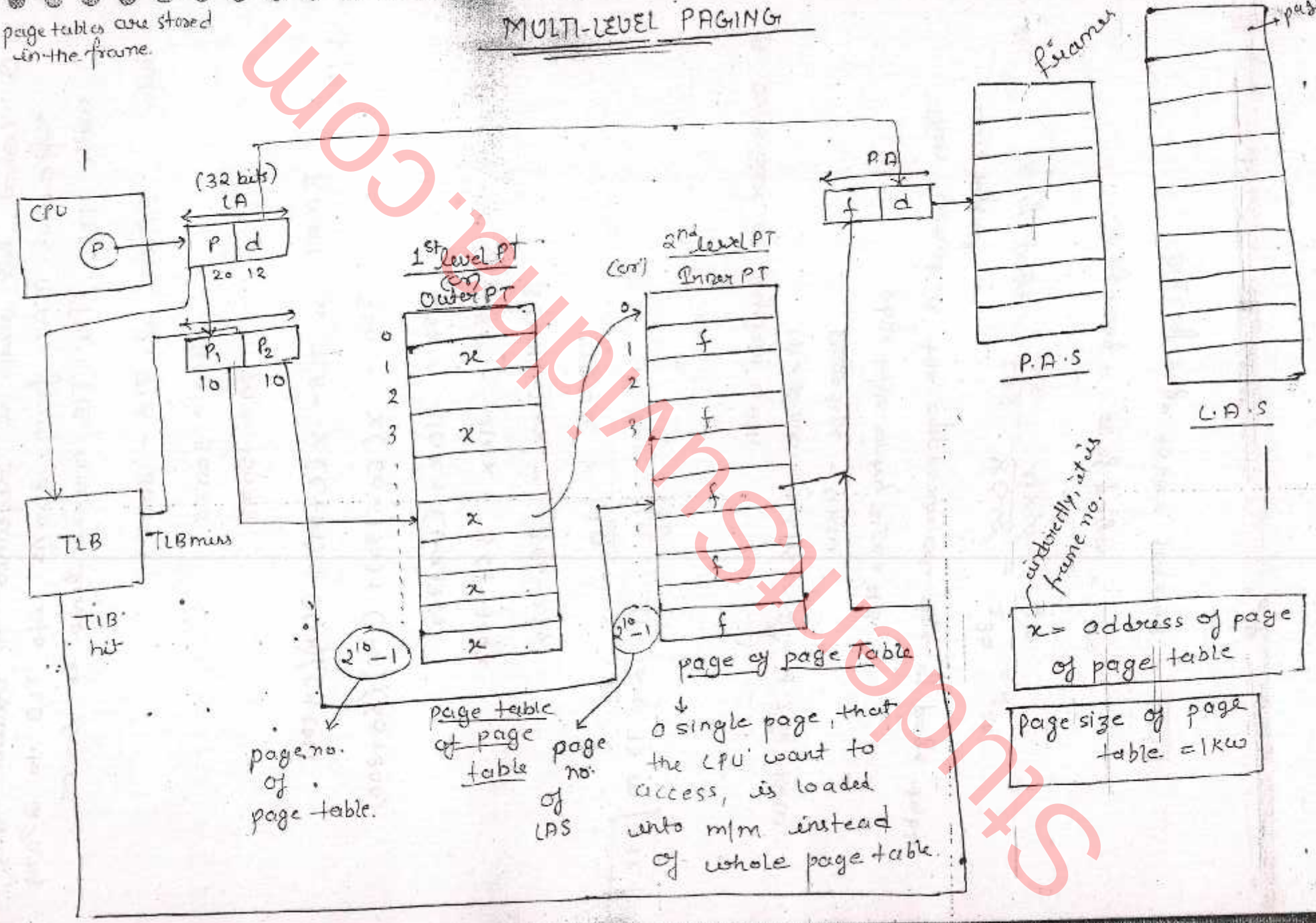
Sol: # of pages = $\frac{LAS}{PS} = \frac{4GW}{4KW} = \frac{2^{30}}{2^{10}} = 2^{20}$

# of entry = # of pages = 1M

Size of page table = 1M x 4B = 4MB

page tables are stored in the frame.

MULTI-LEVEL PAGING



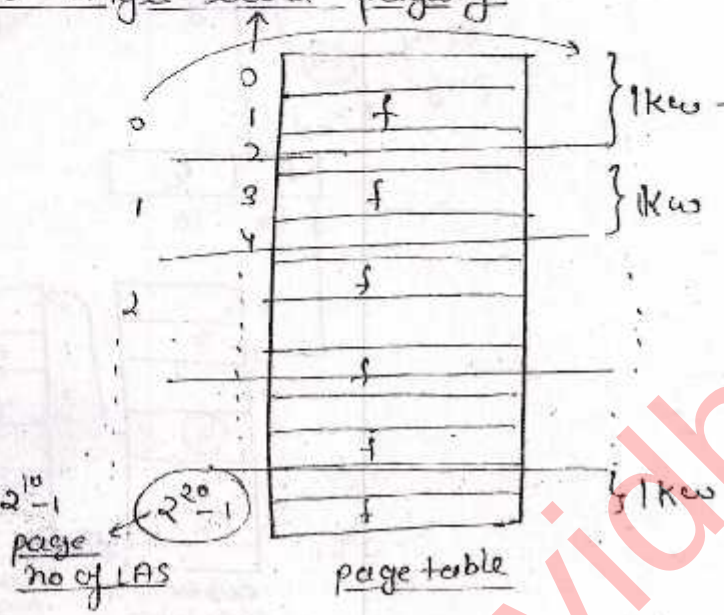
LA = 32 bits

LAS = 2^{32} = 4GW

PS = 4KW

# of pages = # of entries in page table = $\frac{LAS}{PS} = \frac{4GW}{4KW} = 2^{20}$

a/c to single-level paging:



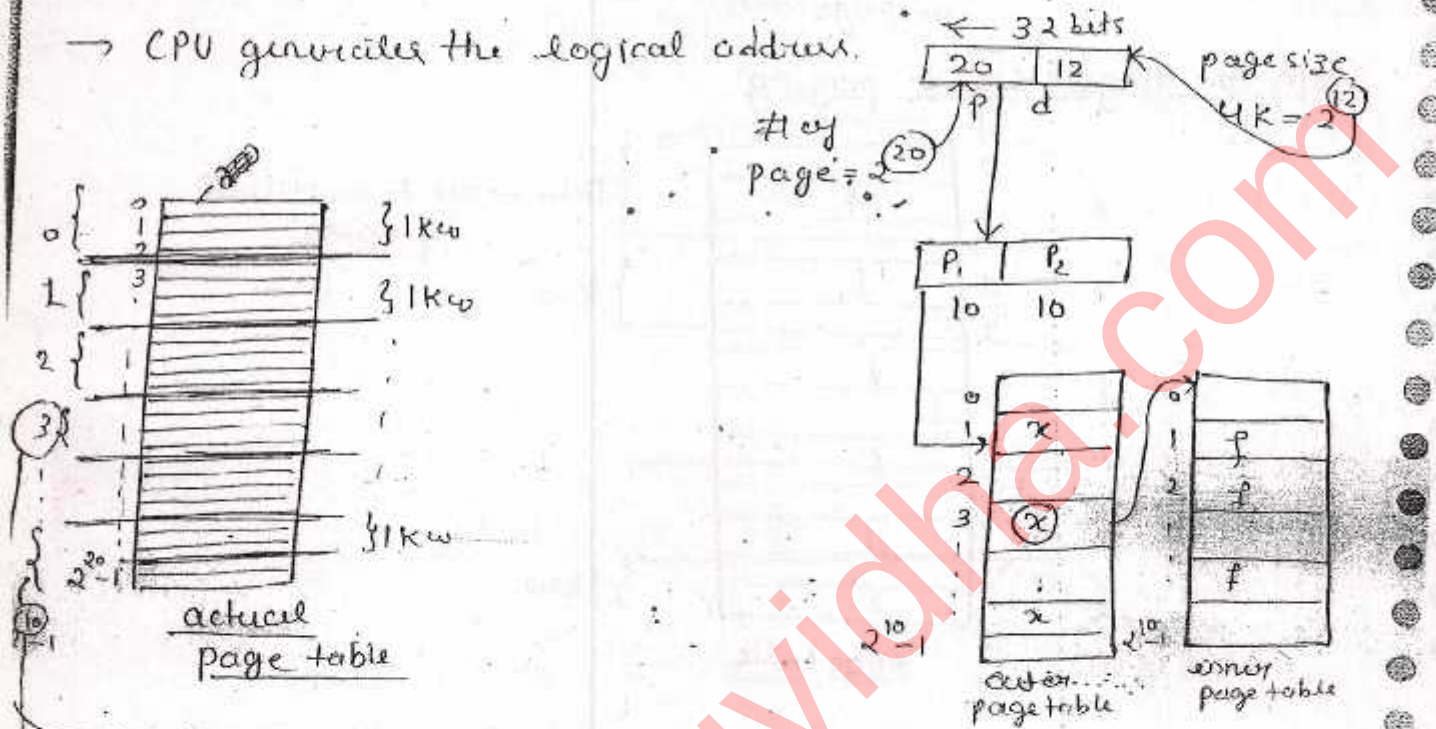
- To avoid the overhead of maintaining, the large size page tables, the multi-level paging will be implemented.
- In the multi-level paging, the paging will be applied on the page table.
- Instead of bringing the entire page table into main m/m, the pages of page table will be brought into main m/m.
- We have divided the pages of page table of page size of 1KW.

# of pages on PT = $\frac{2^{20}}{2^{10}} = 2^{10}$

- P1 → no. of bits required to represent pages of page table or page no. of page table

- P_2 - no. of bits is required to represent the page size of page table or word no. of page of page table.

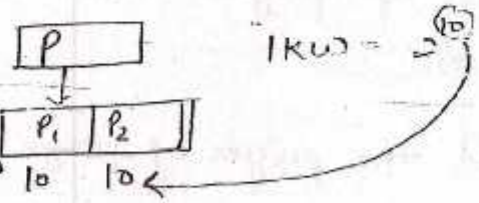
→ CPU generates the logical address.



→ CPU wants to access page no. [x] of page table.

$P_1 = 3$ $\quad$ $P = x$ $\quad$ $x = 0, 1, 2, \ldots, 2^{20}-1$

→ P_1 gives address ie. 'x' of page no. 3, with the help of x, we can load entire page no. 3 in main m/m instead of whole page table.



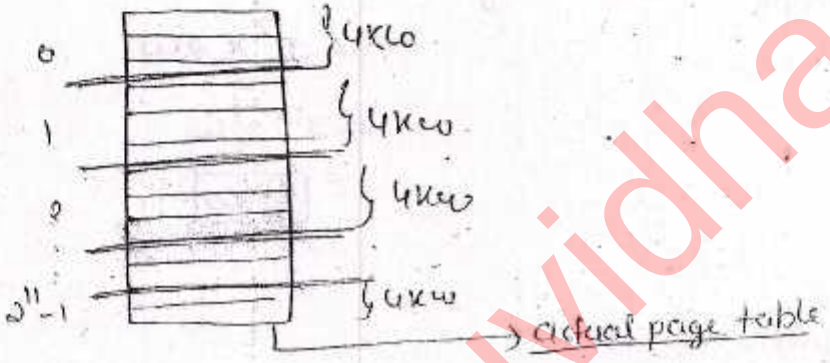
→ inner page table provide f ie. provide frame no. where actual page of LAS is placed in actual physical address.

P_2 → provide page no. of LAS in actual page table.

Q. consider a system with 2-level paging applicable. The page table has divided into 2K pages, each of size 4Kw. The page table entry size is 2w. The m/m is word addressable, the PAS is 64MW which is divided into 16K frames then calculate -

(i) length of LA (35 bit)
(ii) length of PA (26 bit)
(iii) outer page table size (4Kw)
(iv) inner page table size. (8Kw)

Sol:-



P_1 = 11 bits

$P_2 \Rightarrow 4K \Rightarrow 2^2 \cdot 2^{10} = 12$ bits

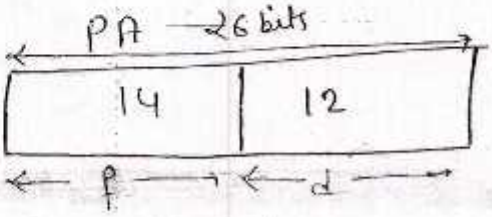
$P = P_1 + P_2 = 11 + 12 = 23$ bits

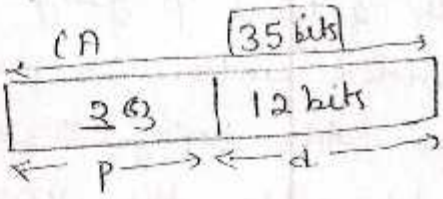
PAS = 64MW

$PA \Rightarrow 2^6 \cdot 2^{20} =$ **26 bits**

# of frames = 16K

$f \Rightarrow 2^4 \cdot 2^{10} = 14$ bits





Outer page table entry size = $2^{P_1} \times 2\omega$.

$= 2^{11} \times 2\omega$.

$= 2^{11+1} \cdot \omega = 2^2 \cdot 2^{10} \cdot \omega = \boxed{4K\omega}$

Inner page table entry size = $2^{P_2} \times 2\omega$

$= 2^{12} \times 2\omega$.

$= 2^{10} \cdot 2^3 \omega$

$= \boxed{8K\omega}$

Important points

In Multilevel paging

1) Addressing the P.T is same as addressing frame.
2) no. of bits in 'x' = # of bits in 'f'

Q. Consider a system with 2-level paging applicable. The page table has divided into 8K pages, each of size 16KB. The m/m is byte addressable. The PAS is 128 MB. which is divided into 4KB frames. The page table entry size is 32 bits then calculate-

(i) length of logical address (39 bits)
(ii) length of physical address (27 bits)
(iii) Outer page table size (32KB)
(iv) inner page table size. (64KB)

Sol: $P_1 \Rightarrow 8K \Rightarrow 2^3 \cdot 2^{10} = 13$ bits

$P_2 \Rightarrow 16K \Rightarrow 2^4 \cdot 2^{10} = 14$ bits

PAS = 128 MB

PS = 4KB $\Rightarrow$ d $\Rightarrow$ 4K $\Rightarrow 2^2 \cdot 2^{10} = 12$ bits

PA $\Rightarrow$ 128M $\Rightarrow 2^7 \cdot 2^{20} = 27$ bits

P = $P_1 + P_2$ = 13 + 14 = 27

(i) LA = | P | d | 27 + 12 = 39 bits

(ii) PA = 27 bits

(iii) Outer page table size = $2^{P1} \times (32/8)B = 2^{13} \cdot 2^2 = 2^{15} = 32KB$

(iv) inner page table size = $2^{P2} \times (4)B = 2^{14} \cdot 2^2 = 2^{16} = 64KB$

PERFORMANCE OF 2-level Paging

- The main m/m access time = 'M' (MMAT)
- If the PT's are stored in main m/m Then Effective memory Access Time i.e. EMAT

$$EMAT = 3M\ (M + M + M)$$

(M: outer P table; M: inner P-table; M: PAS)

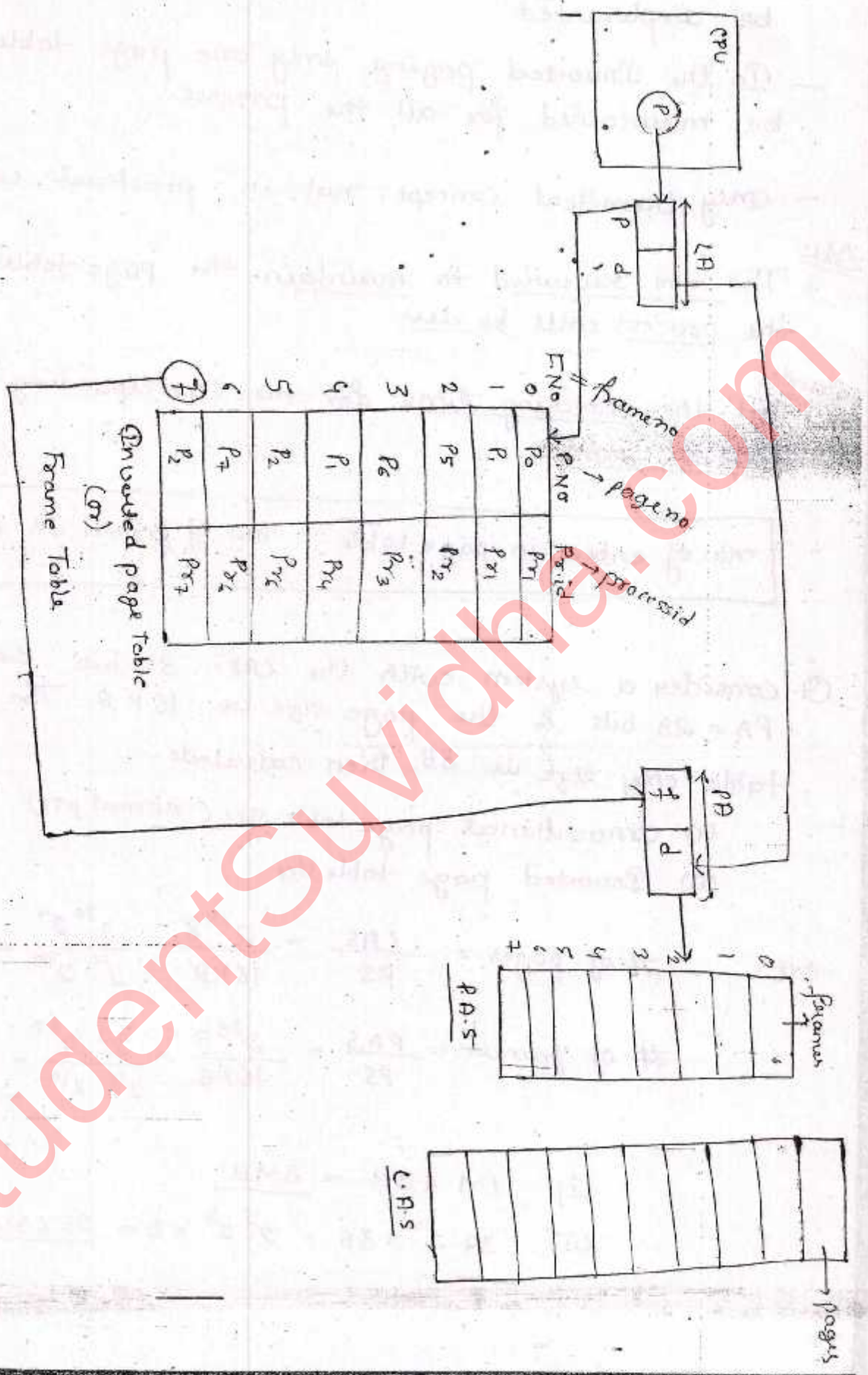
- If translation lookaside buffer is added to improve the performance
- The TLB contain frequently referred the page nos & corresponding frame nos.
- TLB access time = 'C'
  TLB hit ratio = 'x'

$$EMAT = \underbrace{x\,(C+M)}_{\text{hit}} + \underbrace{(1-x)\,(C+M+M+M)}_{\text{miss}}$$

(C: access TLB; M: access PAS; C: TLB; M: OPT; M: IPT; M: PAS)

$$EMAT = x\,(C+M) + (1-x)(C+3M)$$

INVERTED PAGING



— To avoid the overhead of maintaining the page table for every proces, the Inverted Paging will be implemented.

— In the Inverted paging, only one page table will be maintained for all the process.

— Only theoritical concept, not in practical use.

Adv
- The m/m required to maintain the page tables of the process will be less.

Desadv
- But the searching time for the corresponding process will be more.

- | no. of entries in page table = no. of frames in PAS |

Q. Consider a system with the LAS = 34 bits and the PA = 29 bits & the page size is 16 KB. The page table entry size is 8B. Then calculate -

(i) conventional page table size (normal PTS)

(ii) Inverted page table size

Sol:- # of pages $= \frac{LAS}{PS} = \frac{2^{34}B}{16KB} = \frac{2^{30}.2^4}{2^4.2^{10}} = 2^{20} = 1M$

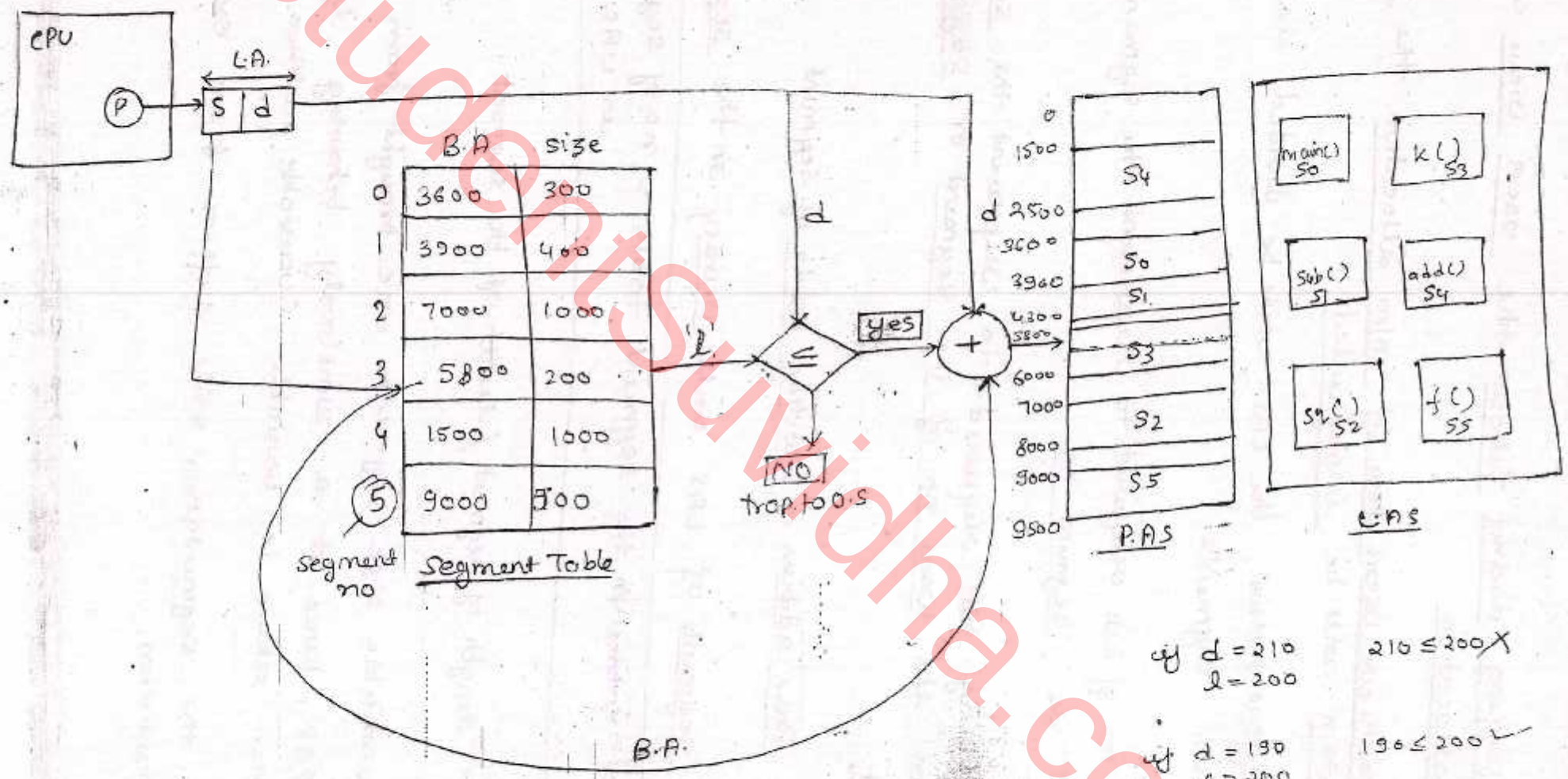
# of frames $= \frac{PAS}{PS} = \frac{2^{29}B}{16KB} = \frac{2^9.2^{20}}{2^4.2^{10}} = 2^5.2^{10}$
$= 32K$

(i) $1M \times 8B = \underline{8MB}$

(ii) $32.2^{10} \times 8B = 2^5.2^3.KB = \underline{256\,KB}$

SEGMENTATION

$d \leq l$



if $d = 210$, $l = 200$ — $210 \leq 200$ ✗

if $d = 190$, $l = 200$ — $190 \leq 200$ ✓

$5800 + 190 = 5990$

- The paging doesnot follow the users view of m/m allocation.

- To achieve users view of m/m allocation, the segmentation will be implemented.

- In segmentation, the LAS will be divided into various segments.

- S = no. of bits required to represent the segments of LAS. or segment no.

- d = no. of bits required to represent the segment size or the word no. of the segment or segment offset.

- B.A = Base address or starting addr of segment

- The segments of LAS will varay in the size.

- No. of entries in the segment table = no of segments in LAS.

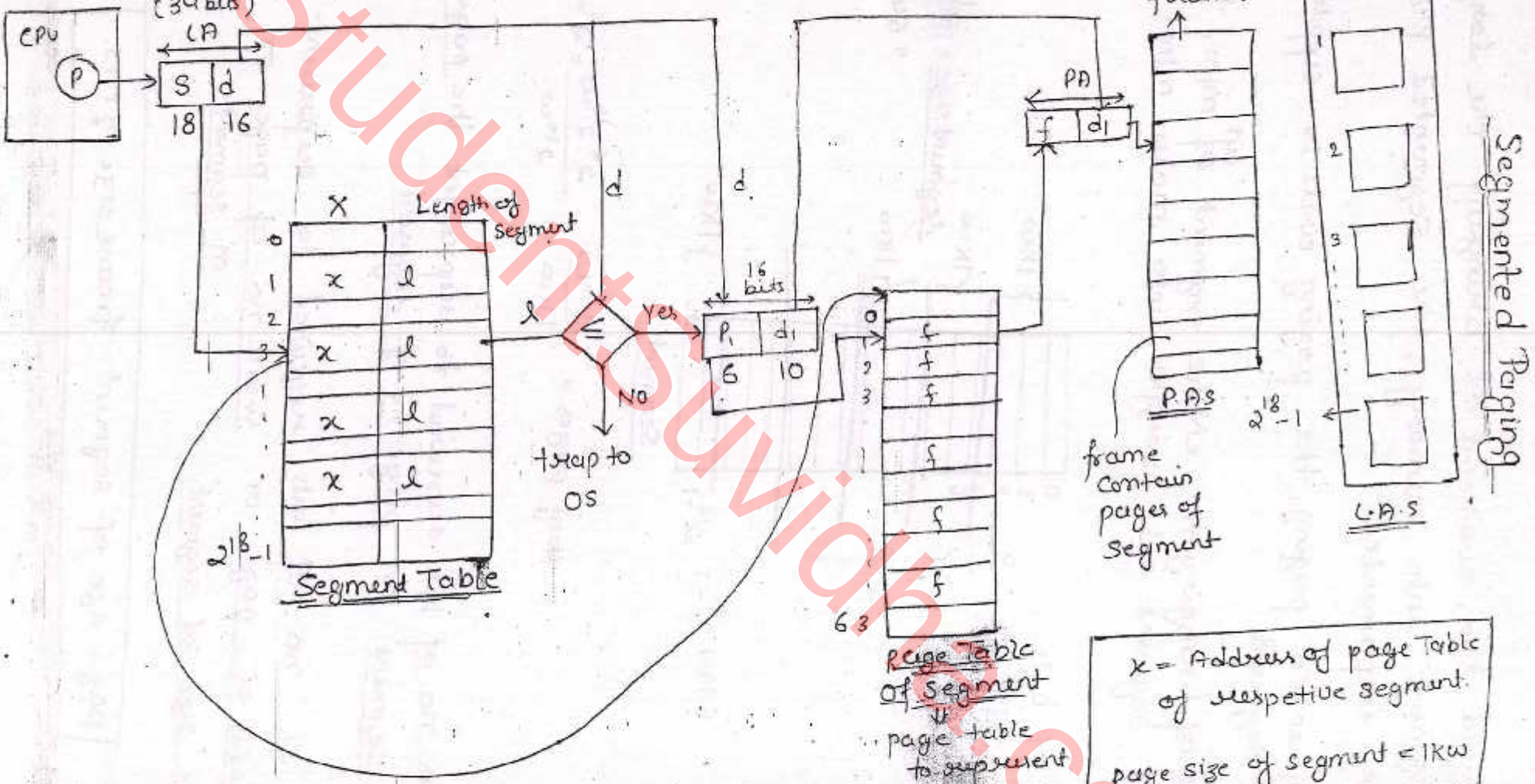
- 'l' = length of segment / limit of the segment

- The variable size segments are brought from LAS to PAS, hence it is similarly behaving like partition scheme ie. variable (variable partition scheme)

- Then, the segmentation still suffers from external fragmentation.

Segmented Paging

LA = 34 bits
<s,d> = (18,16)



x = Address of page Table of respetive segment.

page size of segment = 1Kw

Segment size = 2^{16} w

paging on segment, when segment size increases

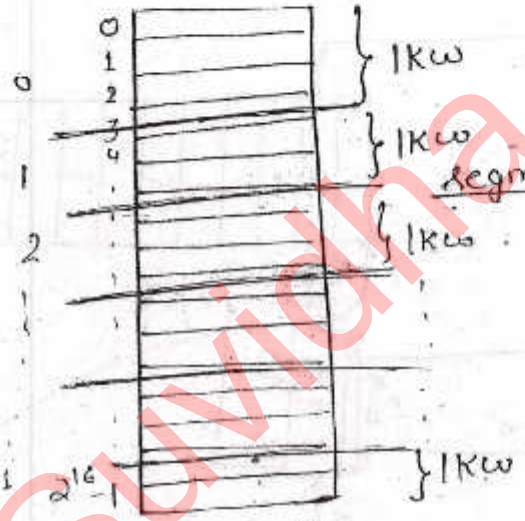
- To avoid the overhead of bringing the large size segment into memory, the segmented paging will be implemented.

- In segmented paging, the paging will be applied on the segment.

- Instead of bringing the entire segment of m/m, the pages of segment are brought into main m/m.

- LAS = 34 bits

$\langle s, d \rangle = (18, 16)$



Segment size = 2^{16} w = 64Kw

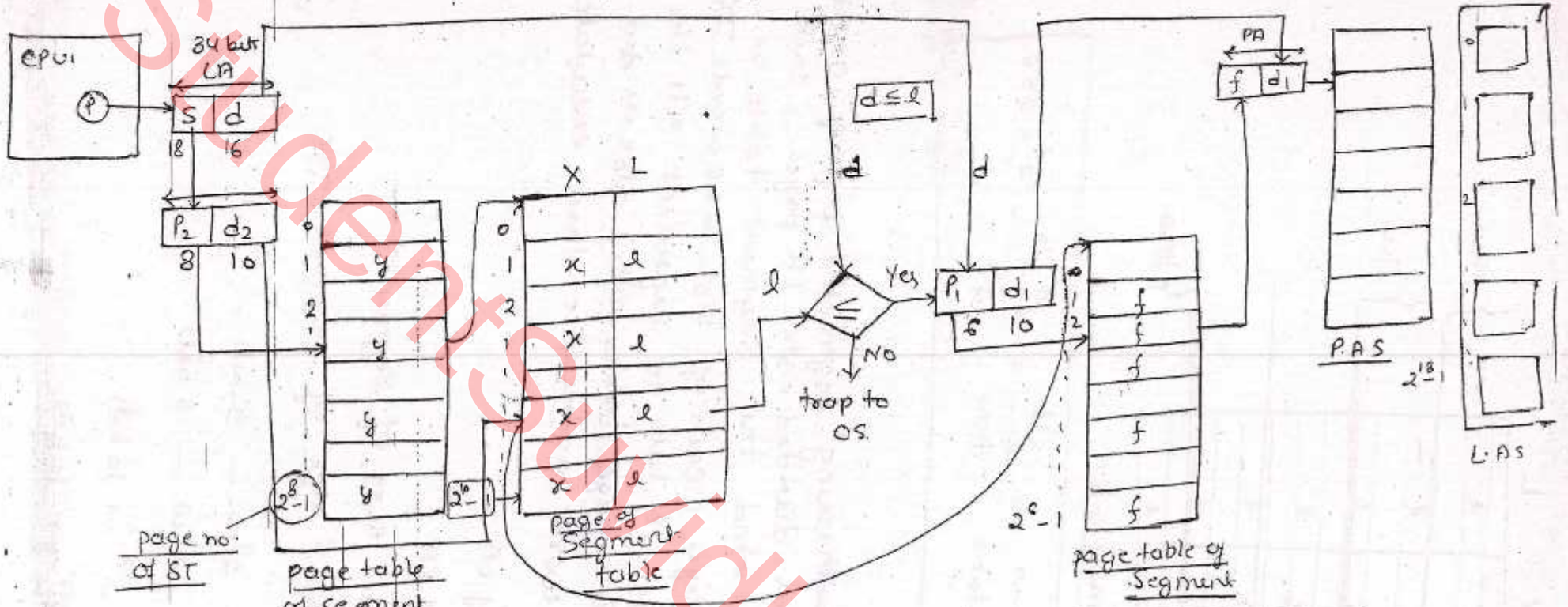
$$\#\text{of pages} = \frac{2^{16}w}{1Kw} = \frac{2^6 \cdot 2^{10}w}{2^{10}w} = 2^6 = 64$$

- p_1 is no of bits required to represent the pages of segment or page no of segment

- d_1 is no. of bits required to represent words of page or word no of page or page size of segment. or of segment

page size of segment = frame size of P.A.S.

paging on segment, when segment size increases & also paging on STable, when S.T. size increases

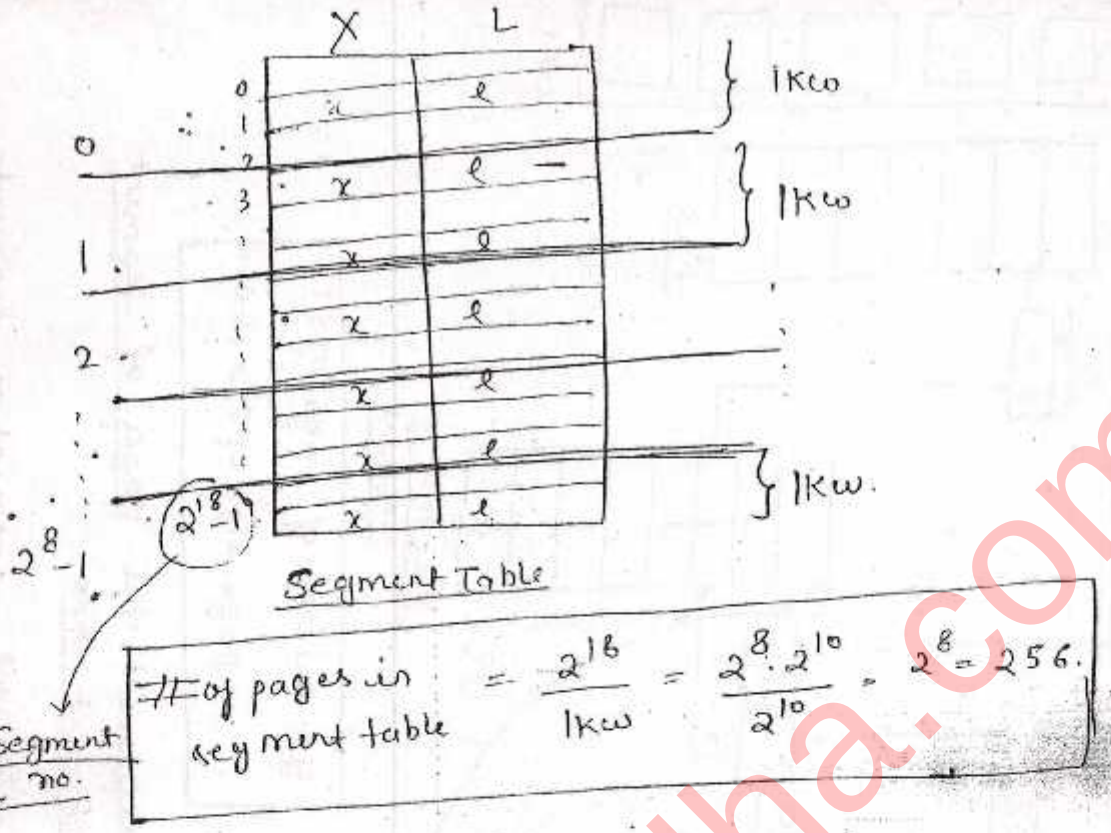


y = address of page of ST.

page size of ST = 1kw

P_2 - no. of bits required to represent the pages of segment table or page no. of segment table.

d_2 - no. of bits required to represent segment of segment-table or segment no. of S.T. or page size of ST or word no. of page of ST or LAS



Q Consider a system, with using segmented paging archi. where the segment is divided into 1K pages, each of size 512 words. And the segment table is divided into 1K pages, each of size 256 words. The frame no. requires 18 bits to represent all the frames of PAS. The page table entry size is 2w and the m/m is word addressable then calculate-

(i) length of LA

(ii) length of PA

(iii) page table size of segment

(iv) page table size of segment table.

Sol:- $d_1 = 512w = 2^9 w$ — 9 bits

$d_2 = 256w = 2^8 w$ — 8 bits

$P_1 = 1K = 2^{10} \rightarrow$ 10 bits

$P_2 = 1K > 2^{10} \rightarrow$ 10 bits

f = 18 bits

S = $p_2 + d_2$ = 10 + 8 = 18 bits

d = $p_1 + d_1$ = 10 + 9 = 19 bits

(ii) LA = | 18 (S) | 19 (d) | = 18 + 19 = 37 bits

(iii) PA = | 18 (f) | 9 (d_1) | = 18 + 9 = 27 bits

(iii) PT size of segment = $2^{10} \times 2w$ = 2Kw

(iv) PT size of segment table = $2^{10} \times 2w$ = 2Kw

Q consider a system using segmented paging architecture. The segment is divided into 8K pages, each of size 2KB. and the segment table is divided into 4K pages each of size 1KB. The m/m is Byte addressable. And the PAS is 64MB and the page table entry size is 32 bits. Then calculate –

(i) length of LA
(ii) length of PA
(iii) PTS of S
(iv) PTS of ST
(v) no. of frames in PAS

Sol: d_1 = 2KB = 11 bits
p_1 = 8K = $2^3 \cdot 2^{10}$ = 13 bits

d_2 = 1KB = 2^{10} = 10 bits
p_2 = 4K = $2^2 \cdot 2^{10}$ = 12 bits

$d_1 + p_1 = 24$
$d_2 + p_2 = 22$

frames = $\frac{PAS}{PS} = \frac{PAS}{2KB} = \frac{64MB}{2KB} = \frac{2^6 \cdot 2^{20}}{2^1 \cdot 2^{10}} = 2^5 \cdot 2^{10} = 32K$

(i) | 22 (S) | 24 (d) | = 46 bits

(ii) | f | d_1 | = 15 + 11 = 26 bits

(iii) $2^{13} \times 2^2 = 2^5 \cdot 2^{10}$ = 32KB

(iv) $2^{12} \times 2^2 \cdot 2^4 \cdot 2^{10}$ = 16KB

(v) 32K

Q. Consider a system using segmented paging architecture. Where the LAS = PAS = 2^{16} B = $2^6 \cdot 2^{10}$ B = 64KB. The LAS is divided into 8 equal size segments. each The m/m is byte addressable. The segment is divided into equal size pages which are of powers of '2'. The page table entry size is 2B. The page tables are stored in the main m/m. Then —

what must be the size of the page of the segment in Bytes. So that the page table of segment exactly fits in 1 page frame?

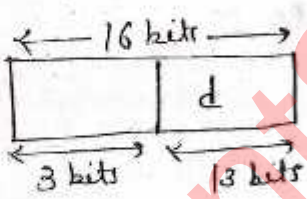
Sol:- LAS = PAS = 64KB

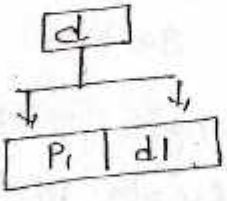
LA = 16 bits

PA = 16 bits

# of segment = 8 :

$$\text{Segment size} = \frac{\text{LAS}}{\#\text{ of seg}} = \frac{64KB}{8} = \frac{2^6 \cdot 2^{10}}{2^3} = 2^3 \cdot 2^{10} = 2^{13} = 8KB$$





# of pages in segment = 2^m

$$2^m = \frac{2^{13}}{\text{page size}} \Rightarrow \text{page size} = \frac{2^{13}}{2^m} = 2^{13-m}$$

page size = frame size

size of page table entry = $2^m \times 2B = 2^{m+1}$ B. = frame size

page size = frame size

$$\frac{2^{13}}{2^m} = 2^{m+1} \Rightarrow 2^{13} = 2^m \cdot 2^{m+1}$$

$$2^{13} = 2^{2m+1} \Rightarrow m = 6$$

page size = $\frac{2^{13}}{2^6} = 2^7 = \underline{128\ B}$.

---

page size of segment = 2^x

$$\#\text{ of page} = \frac{2^{13}}{2^x} \Rightarrow 2^{13-x}$$

Size of page table entry = $2^{13-x} \times 2 \Rightarrow 2^{14-x}$

$$2^x = 2^{14-x}$$

$$x = 14 - x \Rightarrow 2x = 14 \Rightarrow x = 7$$

page size = $2^7 = \underline{128\ B}$