9, 2023
Wednesday

ENHANCED SECOND CHANCE

→ when using reference

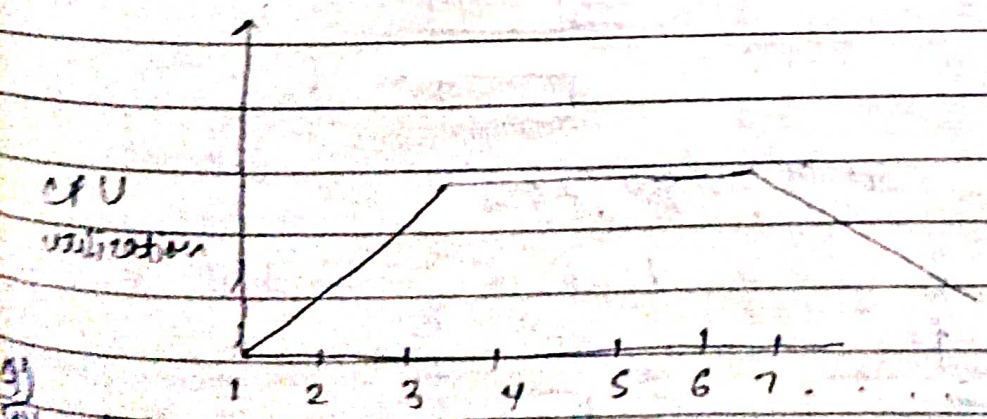
Dirty Bit	Ref Bit	Page	Priority	
0	0	P1	1	~~most~~ least Dirty Page to replace
0	1	P2	2	
1	0	P3	3	
1	1	P4	4	Most least Page to replace

- If contents of page is updated (dirty) ⇒ Then change the Dirty Bit = 1
  (Something is changed in that Page "Write" operation)

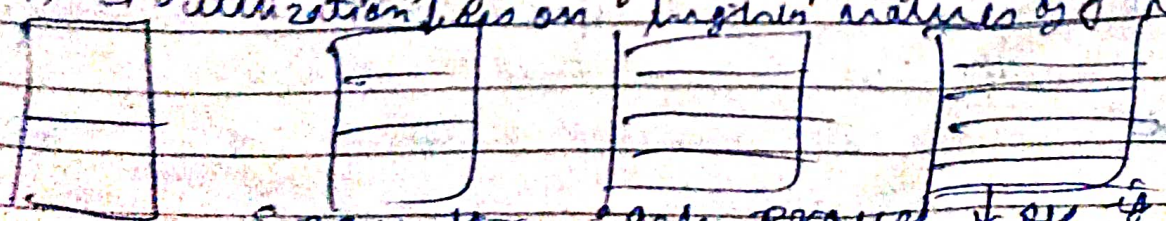
If Dirty Bit = 1 ⇒ Then you need to SAVE it to hard disk & then accumulate

Replace the ~~most~~ least cost Page.
Giving priority to last page, P4 to let it in memory

THRASHING :-



Why it CPU utilization depends on higher values of MP?



MP ↑ ...

majority of time CPU will Replace pages
(Load new page & removes previous page)
Page Replacement work ↑ ses.

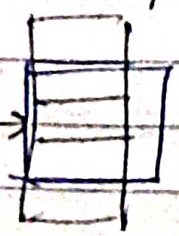
WORKING SET MODEL

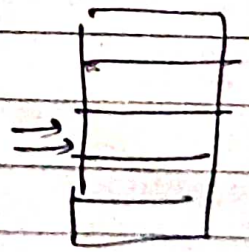
Working set:
set of unique page

Before replacing pages we should focus on any one principle.

Principle of Locality

- Spatial Locality
  - In coming future, we can refer to nearby address only
  - inst 9 is accessed
  - Then inst 8 & inst 10 can be accessed in future
- Temporal Locality
  - Process is referred multiple times then its may also be referred in future





Working Set of process → No of unique pages

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

t_1: 3 1 2 2 → {1, 2, 3}

t_2: 4 1 5 3 → {1, 3, 4, 5}

t_t: 5 4 4 5 → {4, 5}

← ΔW →

← Δt →

P. Mg.

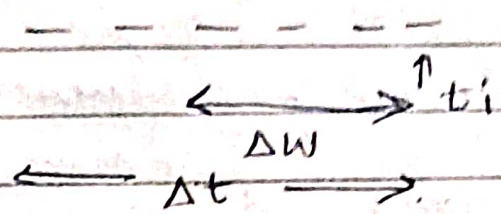
Only now I again & again so we can maximum 2, 3, 4 to something to condition that

After working set, we understand true requirements of process.

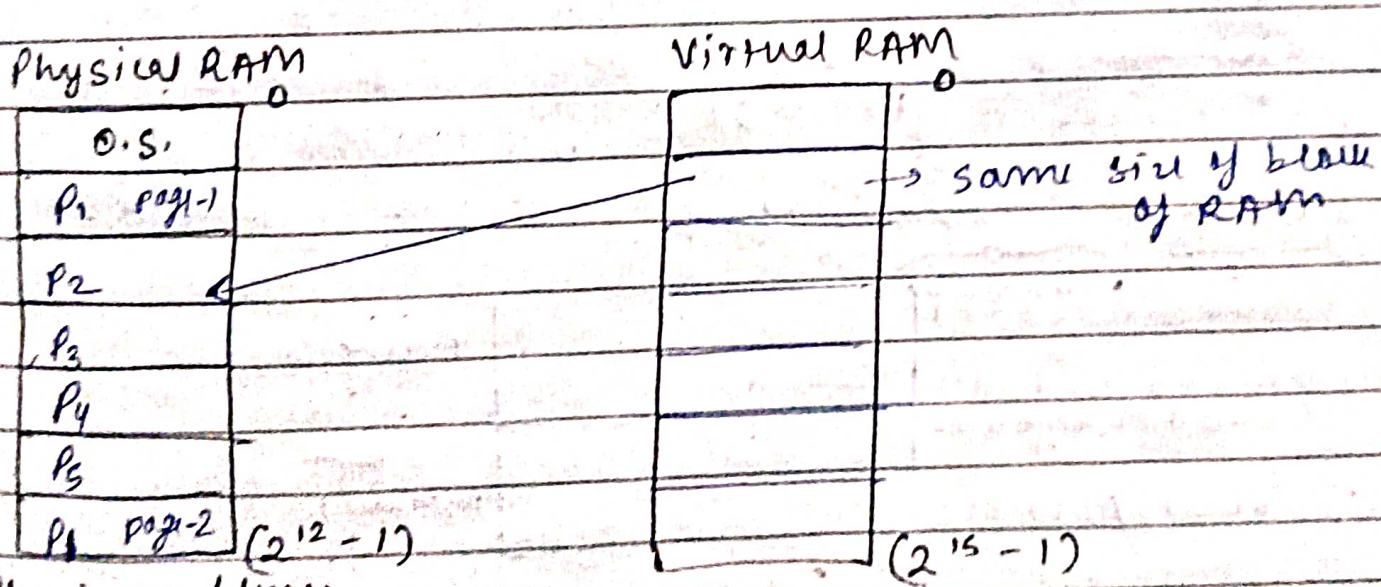
Degree of multiprogramming improved.

t = time interval = 6

ΔW = No. of past References = 4



NON CONTINUOUS MEMORY ALLOCATION (VIRTUAL MEMORY)



Physical Address
12 bit address

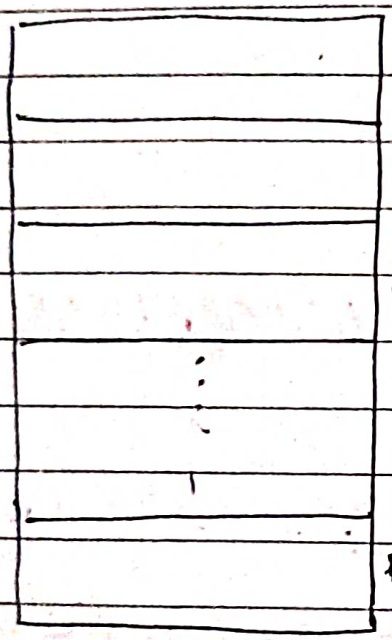
15 bit address
User address
Virtual address
logical address

User wants to run the process that is bigger than physical memory

Each Pi in physical memory will think that it is getting whole virtual memory

03.10.2023
Tuesday

frames inside Physical RAM
pages inside Virtual RAM
For implementing this,
two main concepts are there
Paging & Segmentation



No. of addresses.

Physical RAM $= 2^{12}$

Frame size = 128 words

Single word → 1 addr. is required

No. of frames = $\frac{\text{Physical RAM}}{\text{Frame size}}$

$= \frac{2^{12}}{2^7} = 2^5$

128 $= 32$

[0 to 31]

Frame No. | Frame offset

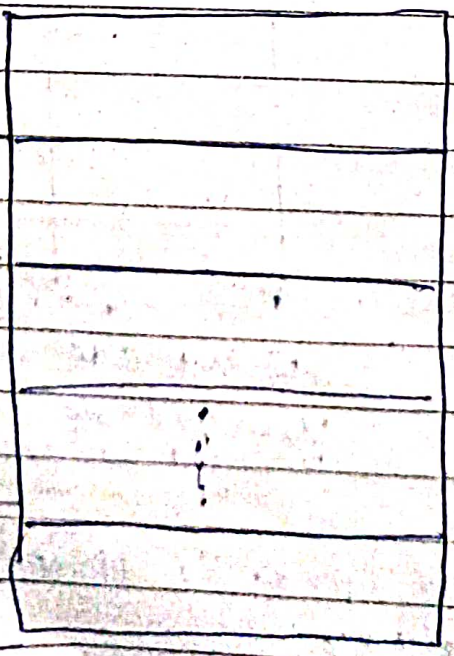
00000 | 0000000
00000 | 0000001

00000 | 1111111
00001 | 0000000

00001 | 1111111

11111 | 0000000

11111 | 1111111



Frame No:- It correspond which frame it is to

Frame offset:- To get particular word in frame

Frame No.	Frame offset
← 5 →	← 7 →

Q) Consider a 15 bit RAM where size of frame is 256 words. What will be No. of bits in frame No. & frame offset.

A: 2^{15} 2^{8} = 256 words

Frame No.	Frame offset
← 7 →	← 8 →

NOTE:

1) Size of frame = Size of Page
2) No. of frames & No. of pages may be different (Address space of Virtual & Physical RAM is different)

0	OS
1	~~User 3 - page 5~~
2	User 1 - page 4
3	User 5 - page 2
⋮	⋮
31	User 1 - page 2

	Resident bit	
0	0	Add^n of sector
1	1	11111
2	0	Add^n of sector
3	1	01001
4	1	
⋮	⋮	
(No. of pages - 1)	1	
	1	

3) For each user/process is maintaining its own "Page Table" to keep info of which page is loaded into which frame

(4) No. of entries in page table = No. of pages → Frame No. (RAM)

(5) Resident bit = 1 ⇒ Page is inside Primary memory
Resident bit = 0 ⇒ Page is in Secondary memory.
Address of sector

(6) Page Table is stored inside Page Table Buffer Register (PTBR)

PTBR points to address of first address of page table (starting add^n of page table)

(7) Page Table is part of PCB (Process Control Block)

Find next add^n = PTBR + offset
(array type)

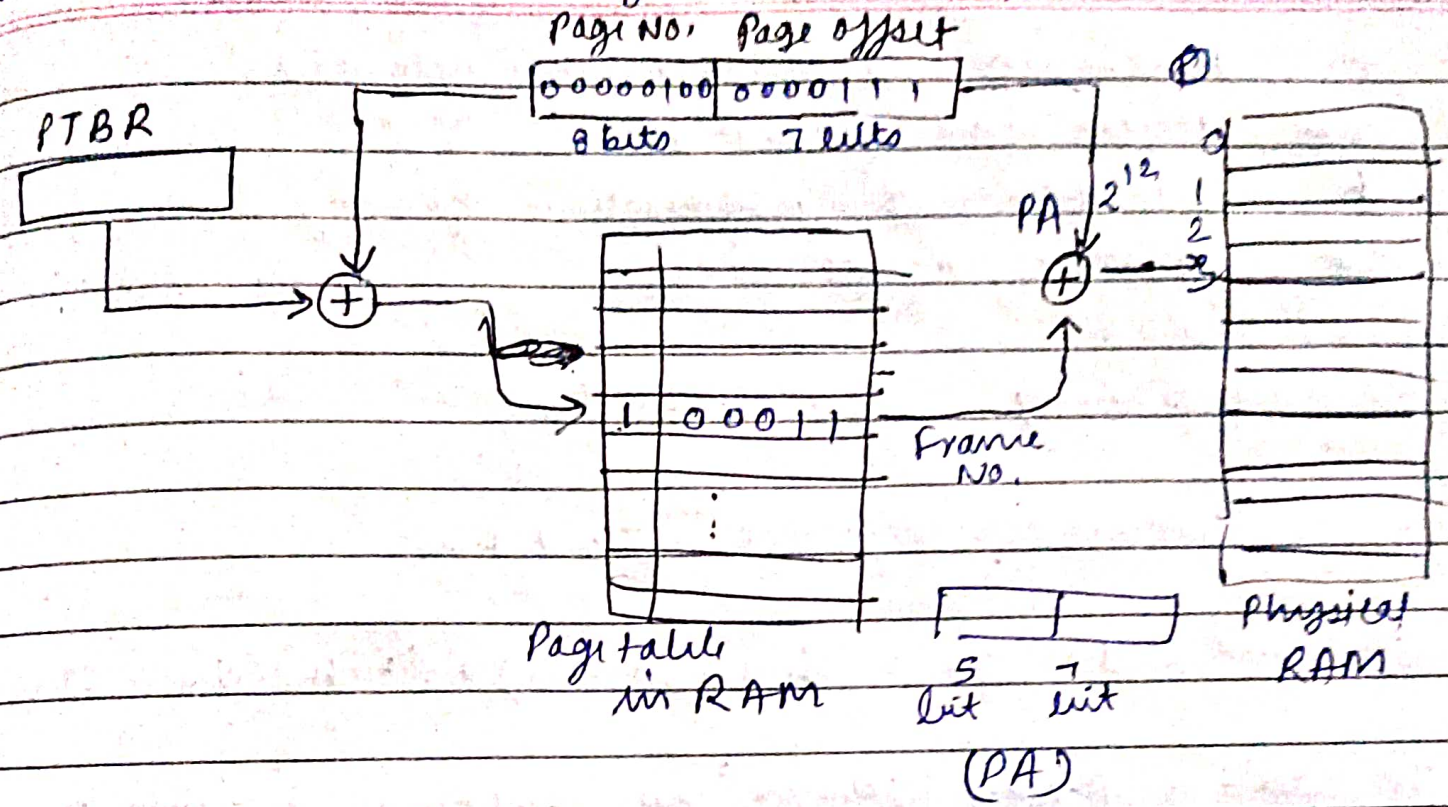
(8) Page offset == Frame offset
Page No. != Frame No.

(9) 2 memory access are required.
(1) page table (which is in RAM)
(2) Main Memory access of data.

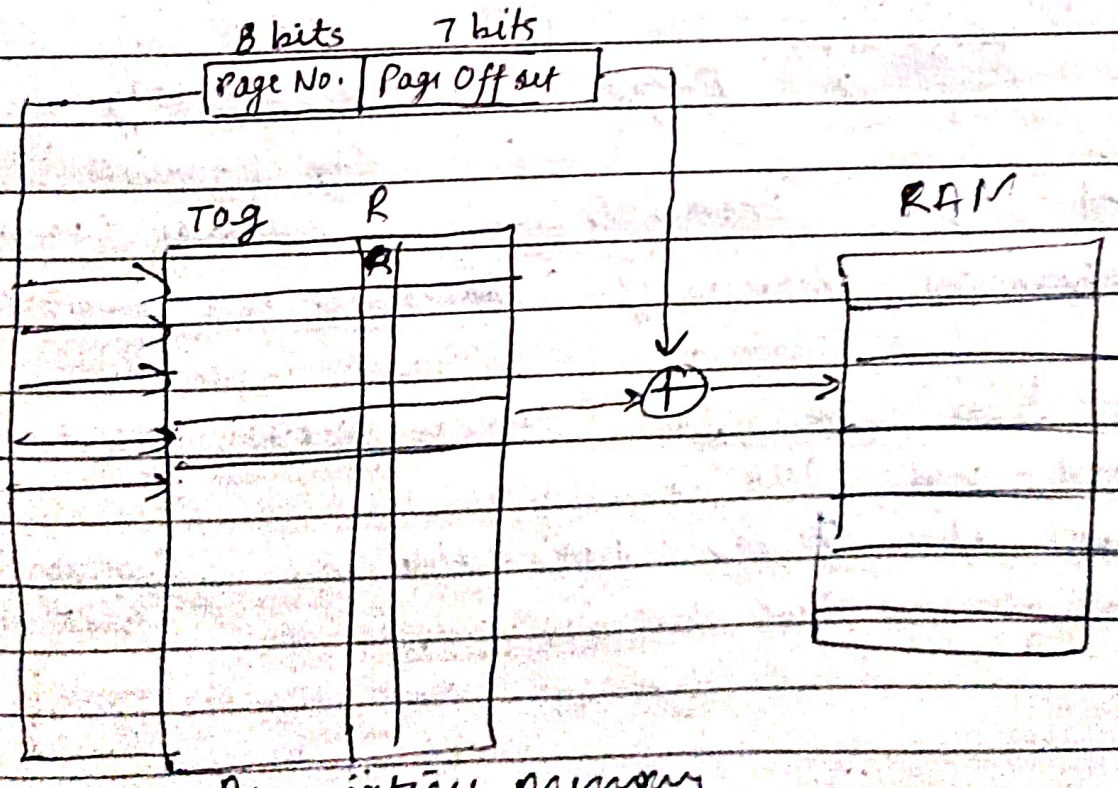
4.10.2023
Wednesday

2^{15}

Logical Address (LA)



Associative mapping stores ~~to~~ page Table



Associative memory

Entire page table in RAM
in Associative Memory

Part in Associative & complete in RAM

t_a Access time of Associative memory
t_m Access time of RAM
h Hit Ratio of Associative memory
t_e Effective Access time

① Direct Mapping $t_e = 2 \times t_m$

② Associative memory $t_e = t_a + t_m$

③ Combined $t_e = h(t_a + t_m) + (1-h)(t_a + 2t_m)$

In case of Paging scheme, no external fragmentation is there.

Process $\xrightarrow{\text{division}}$ Pages (Not perfectly divisible by page size)

Last page $\underbrace{\text{Process \% Page size}}$ → Not Complete

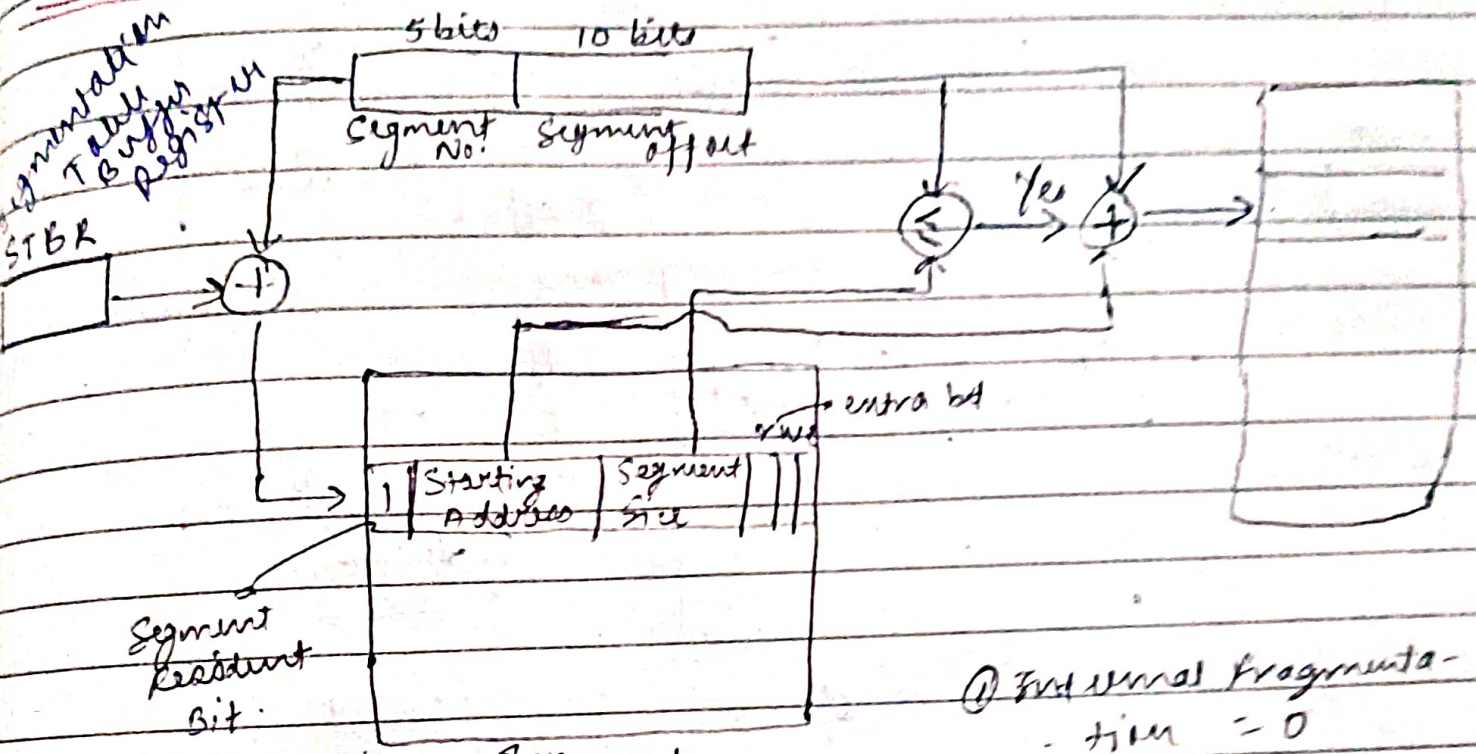
∴ There are chances of 'Internal fragmentation'

Page size ↓ se to ↓ se Internal fragmentation but not 100%

Because it ↑ ses Page table size. Burden for system.

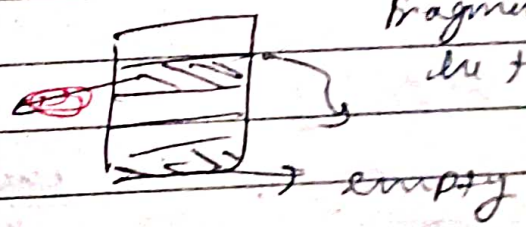
"Segmentation is the sol^n".

SEGMENTATION



① Internal fragmentation = 0

② External fragmentation can be there



PAGED SEGMENTATION

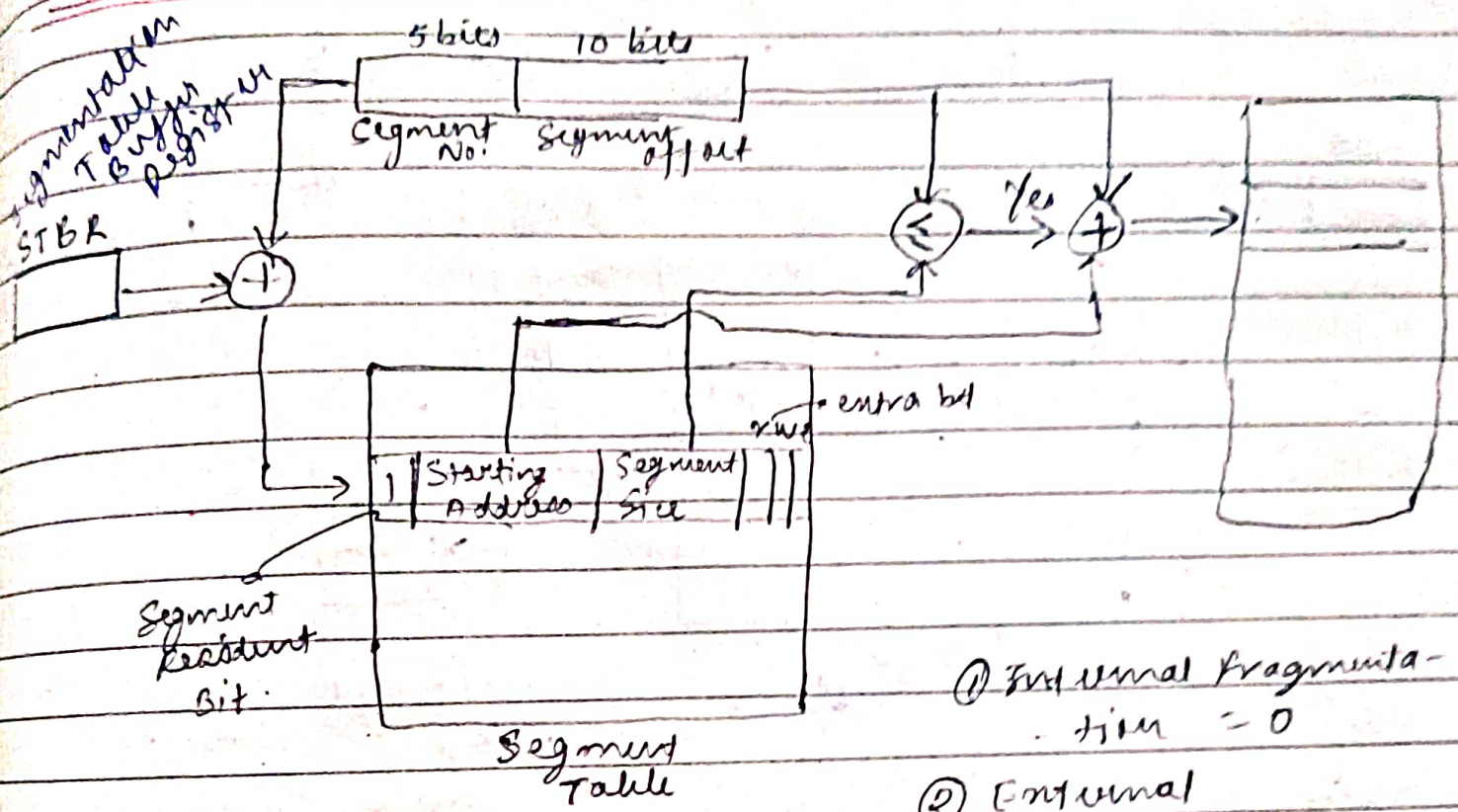
Inside a single segment we are implementing paging scheme.

5 bit | 10 bit → User memory (Two parts)

Segment No. | 3 bits (Page No.) | 7 bits (Page Offset) → For OS (Three parts)

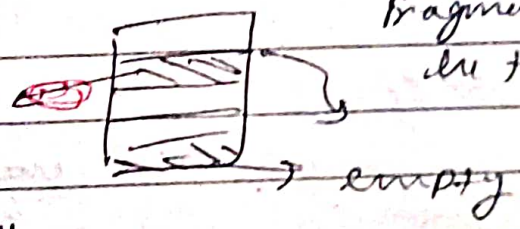
Each segment table has ~~entries for~~ pointer to Page table ~~out~~.

SEGMENTATION



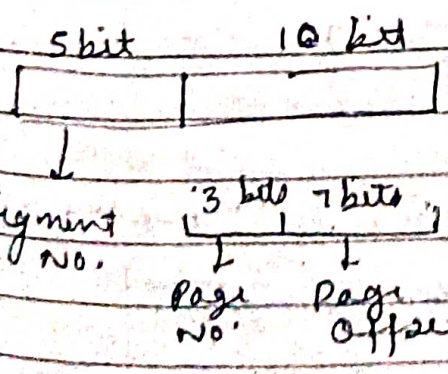
① Internal fragmentation = 0

② External fragmentation can be there



PAGED SEGMENTATION

Inside a single segment we are implementing paging scheme



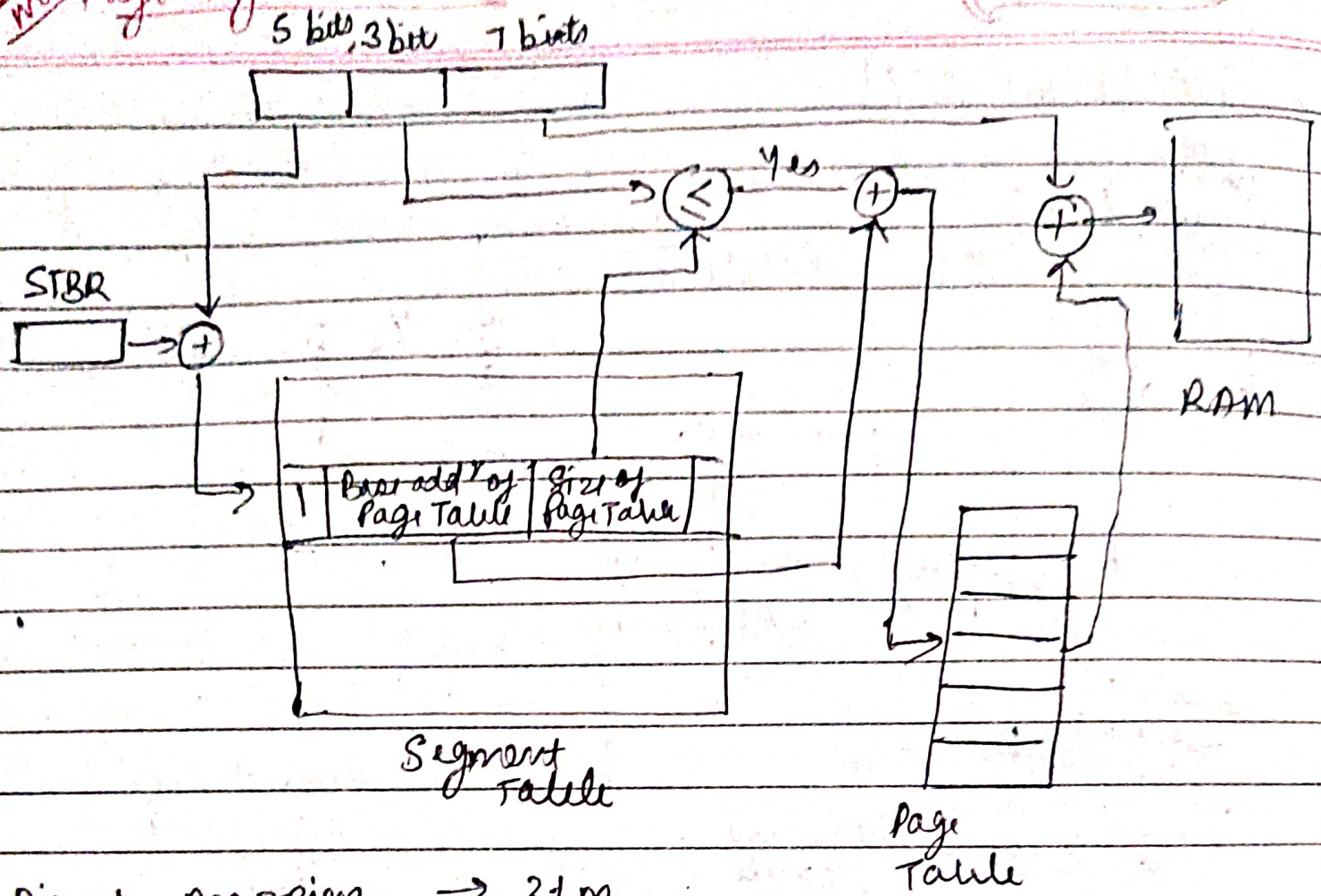
→ User memory (Two parts)

→ For OS (Three parts)

Each segment table has ~~entries for~~ pointers to Page table ~~part~~.

02.11.2023
Monday

Page segmentation



Direct mapping → 3tm

Association mapping ta + 2tm
(any one in cache)

Both in cache ta + tm
(or 2ta)

Hit ratio

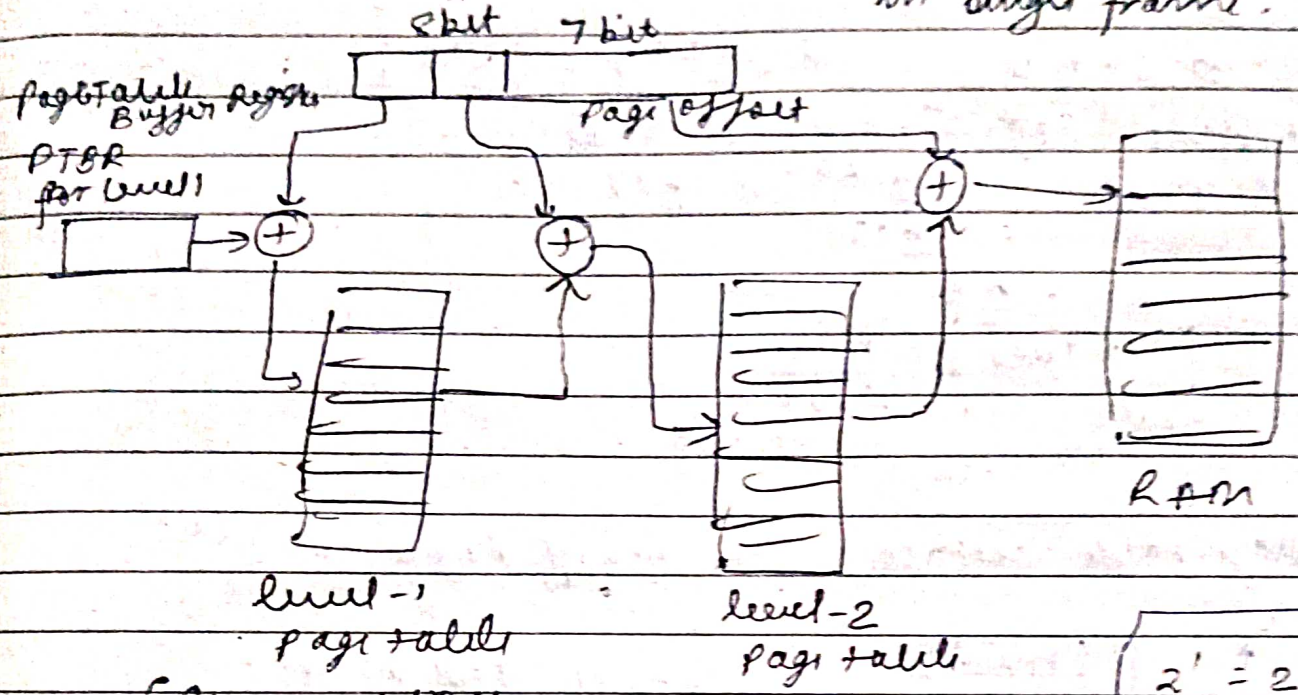
$$t_e = h(t_a + 2t_m) + (1-h)(t_a + 3t_m)$$

↳ एक को cache के अंदर

Segmented Paging

Multilevel paging

When single page table size is too high to be stored in single frame.



$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1K$
$2^{20} = 1M$
$2^{30} = 1GB$
$2^{40} = 1TB$

Q) The logical addr space is of 4 GB & physical addr space is of ~~GB~~ 4 MB. Find No. of pages, No. of frames, no. of page table entries, size of page table = ?
Page size = 4 KB

Sol: Page size = Frame size = 4 KB
$= 2^2 \times 2^{10}$
$= 2^{12}$

Paging scheme

LAS = 4 GB = $2^2 \times 2^{30} = 2^{32}$
Logical Addr space

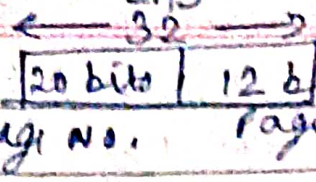
$\boxed{32 \text{ bits}}$

Page size = 4 KB = 2^{12}

No. of pages = $\frac{LAS}{\text{Page size}} = \frac{2^{32}}{2^{12}} = 2^{20}$

10.10.2023
Tuesday.

LAS

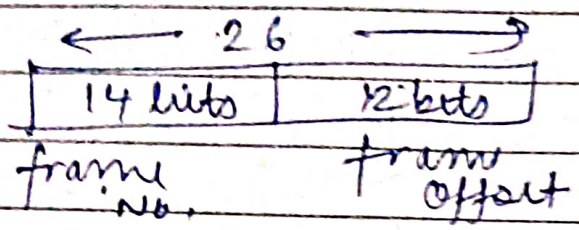


PAS = 64 MB = $2^6 \times 2^{20}$ B = 2^{26} B

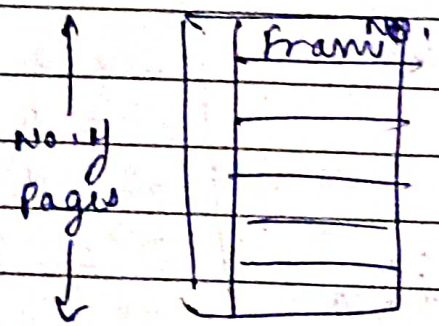
Frame size = Page size = 2^{12} B

∴ No. of frames = $\frac{2^{26}}{2^{12}} = 2^{14}$

PAS



No. of page table entries = No. of pages = 2^{20}



Size of page table =
No. of pages × Bits required to represent 1 ~~page~~ frame

$= 2^{20} \times 14$

$= 14$ MB

Q) consider a system having a logical address of 7 bits, physical address of 6 bits & page size is of 8 words. Find no. of pages & no. of frames.

Sol:

8 bytes = Page size

No. of pages = $\frac{2^6}{2^3} = 2^3$

Page size = 2^3 Bytes

7 bits

LA | 4 bits | 3 bits |
Page No. | Page offset

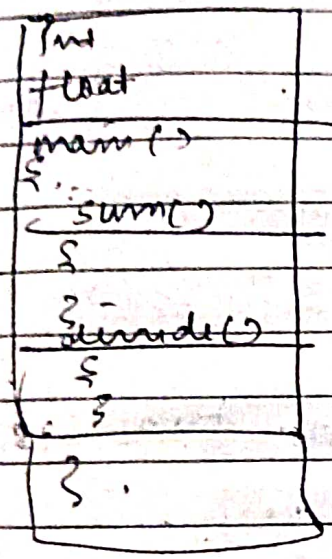
6 bits

PA | 3 bits | 3 bits |
Frame No. | Frame offset

Problems in Paging :-

① Internal Fragmentation (When process is divided it is possible that last part has very less data)

②



If sum is used we need to load the 2nd block too for function.

→ Segment can be this
∴ This can be advantage in segmentation.

③ Overhead of page table (Page table size > segment table)

Multilevel Paging :-

Q) Physical address is of 256 MB & your logical addⁿ is of 4GB, Frame size 4 KB. Page table entry is of 2B.

Sol. PA = 2^{28} B
LA = 2^{32} B
Frame size = 2^{12} B = Page size
Page table size = 2B = 16 bits

No. of Pages = $\frac{LA}{\text{Page size}} = \frac{2^{32}}{2^{12}} = 2^{20}$

Size of Page Table = $2^{20} \times 2$
= 2^{21} MB

PA | Frame No (16) | Frame offset (12) | ← 28 →

LA | 20 | Page offset (12) | ← 32 →

$\frac{2^{20}}{2^9} = 2^{11}$

$\frac{2^{21}}{2^{12}}$

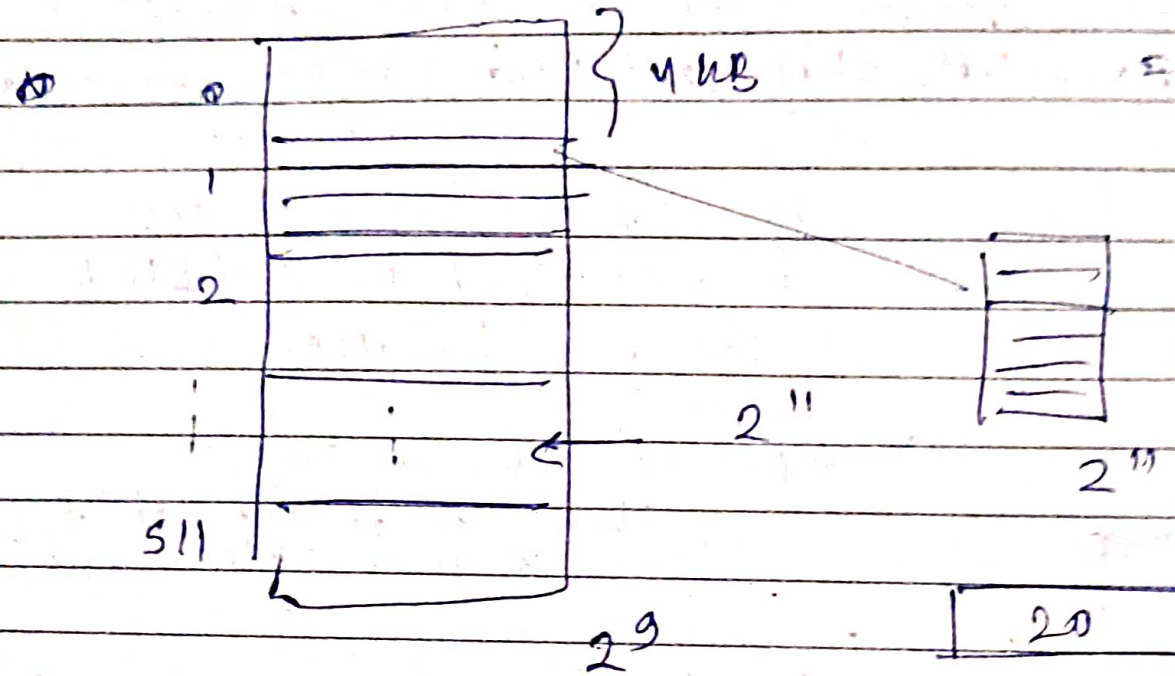
Page Table Size > Frame Size

- Approach: table entries

No. of pages ~~required to~~ store in one frame $= \frac{\text{Page table size}}{\text{Frame size}}$

$= \frac{2MB}{4KB} = \frac{2^{21}}{2^{12}}$

$= 2^9 = 512$ ~~pages~~



20	12

9	11
Outer	Inner

Approach 1

Outer → Inner

Approach 2

Outer ← Inner

Q) LA space = 4 GB PA space = 16 TB Page size = 4KB

How many levels of paging is required

Sol: Frame Size = Page Size = 4KB = 2^{12} B

PA

Frame No	
32	12

$16TB = 2^4 \times 2^{40}$

← 44 →

LA

20	12

$4GB = 2^{32}$

No. of pages $= 2^{20}$

Size of Page Table $= 2^{20} \cancel{\times 2^{12}}$ - × No. of bits to represent frame

11.10.2023
Wednesday

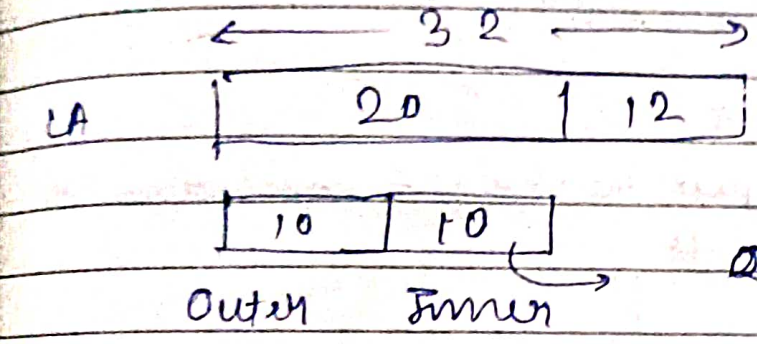
$32/8$

$= 2^{20} \times 2^{3} = \cancel{2^{25}}\ B$ 4 MB

Page Table size > Frame size

No. of page table entries story in 1 frame $= \dfrac{\text{Page table size}}{\text{Frame size}}$

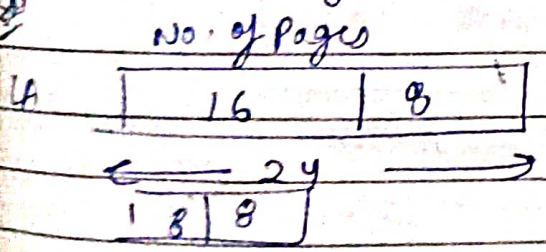
$= \dfrac{\cancel{2^{25}}\ 4MB = 2^{22}}{2^{12}} = \dfrac{\cancel{2^{13}}\ B}{2^{10}\ \cancel{B}}$

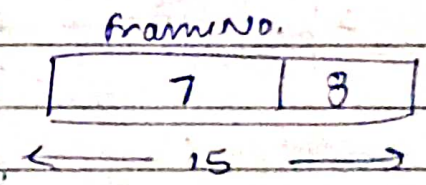


Q $\dfrac{4\ KB}{32} \rightarrow \dfrac{\text{Frame size}}{\text{Frame No.}}$

Q) Virtual memory is of 24 bits & Physical memory is of 15 bits & Page size is 256 words.
Single Page or multilevel paging?

A)





Page size = 256 words = 2^{8} B = Frame size

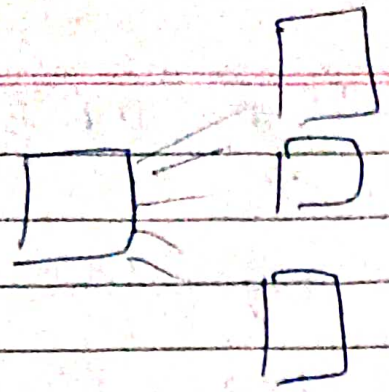
2^{16} Page table size $= 2^{16} \times \left\lceil \frac{7}{8} \right\rceil = 2^{16}$ B

Page table size > Frame size

No. of page table entries $= \dfrac{2^{16}}{2^{8}} = 2^{8} \rightarrow$ (I)

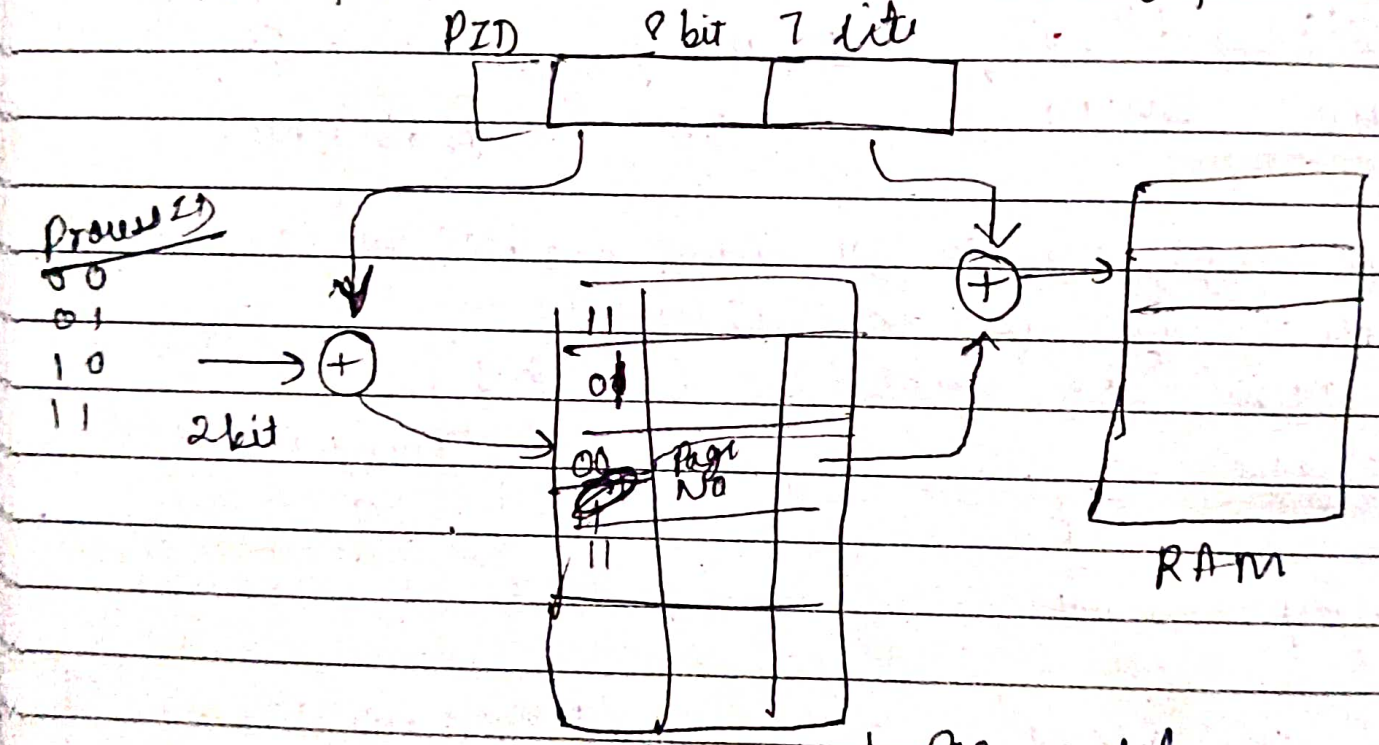
Size of page table $= 2^{8} \times \left\lceil \frac{7}{8} \right\rceil = 2^{8}$

No. of page table entries (II) $= 2^{8}/2^{8} = 1$



INVERTED PAGING :-

Size of the inverted page table is equal to no. of frames in the RAM



In this there is overhead of storing process Id & page No.

0 → No modification Page is modified or not.

1 → Some on Hard Disc

& then display it. Pg. refer

Page table only	Frame No.	valid	RWX	Reference bit	caching enabled	Dirty
	1/0	000		0/1	1/0	0/1

Is there requirement to load page of memory

S0
S1
S2
S3
S4

process (segment of process)

0	
500	
1000	S4
1200	
1500	S2
2700	S0
3000	
3800	S1
4000	
5000	S3

LA:

s	d

the RAM.

also checks whether

LA segment No. displacement

s	d

Yes → Base + d → (P.A.)

	Base	length
S0	1500	1200
S1	3000	800
S2	1200	300
S3	4000	1000
S4	500	500

<2, 25>

Segment No. → displacement

Segment No. 2

Base = 1200

max. Add = 1200 + 300 = 1500

Req^uest = 1200 + 25 = 1225 (2)

1225 < 1500 ✓ valid.

Request = <2, 441>

failed 1200 + 441 = 1641 > 1500 limit