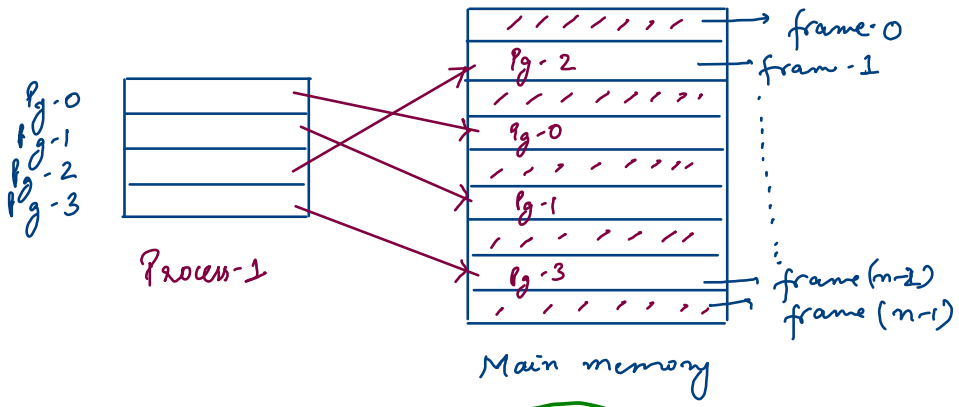
Paging

→ non-contiguous memory allocation technique.

Paging:- mapping CPU generated LA to PA.

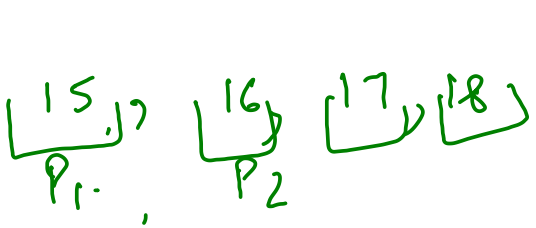
→ Frames - unit of phy. memory, divided into blocks of same size called frames.

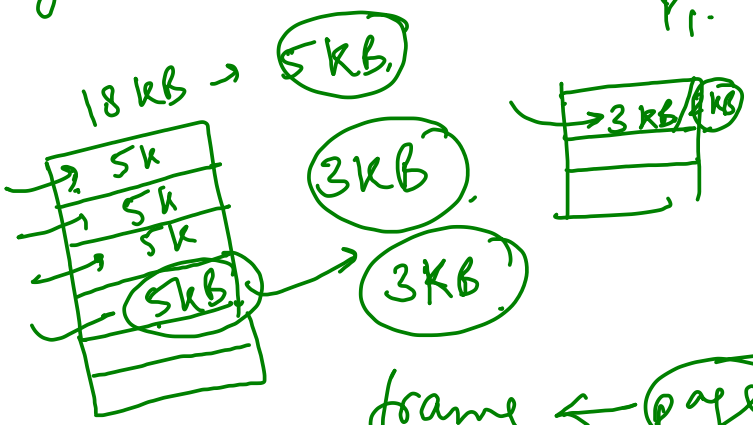
Pages - unit of logical memory, divided into blocks of same size called pages.



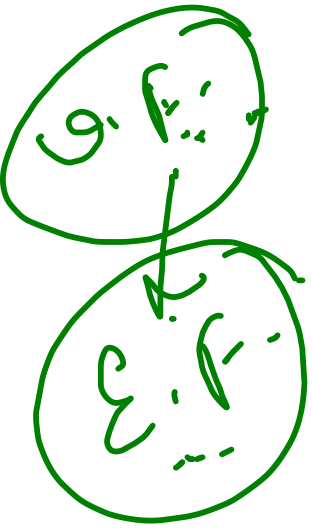
Paging

Page - 2KB ← 1KB

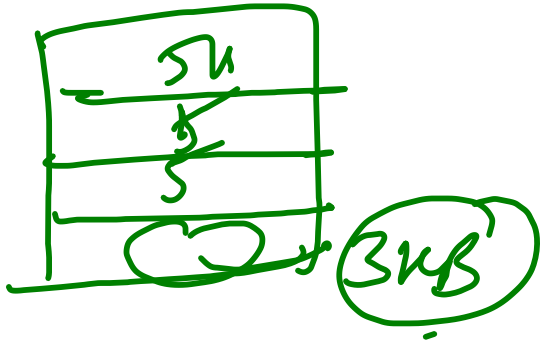


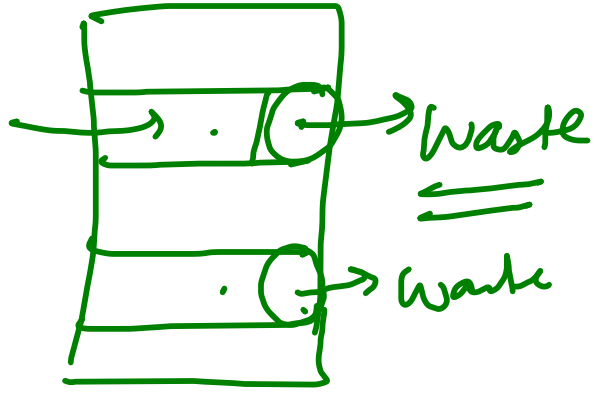


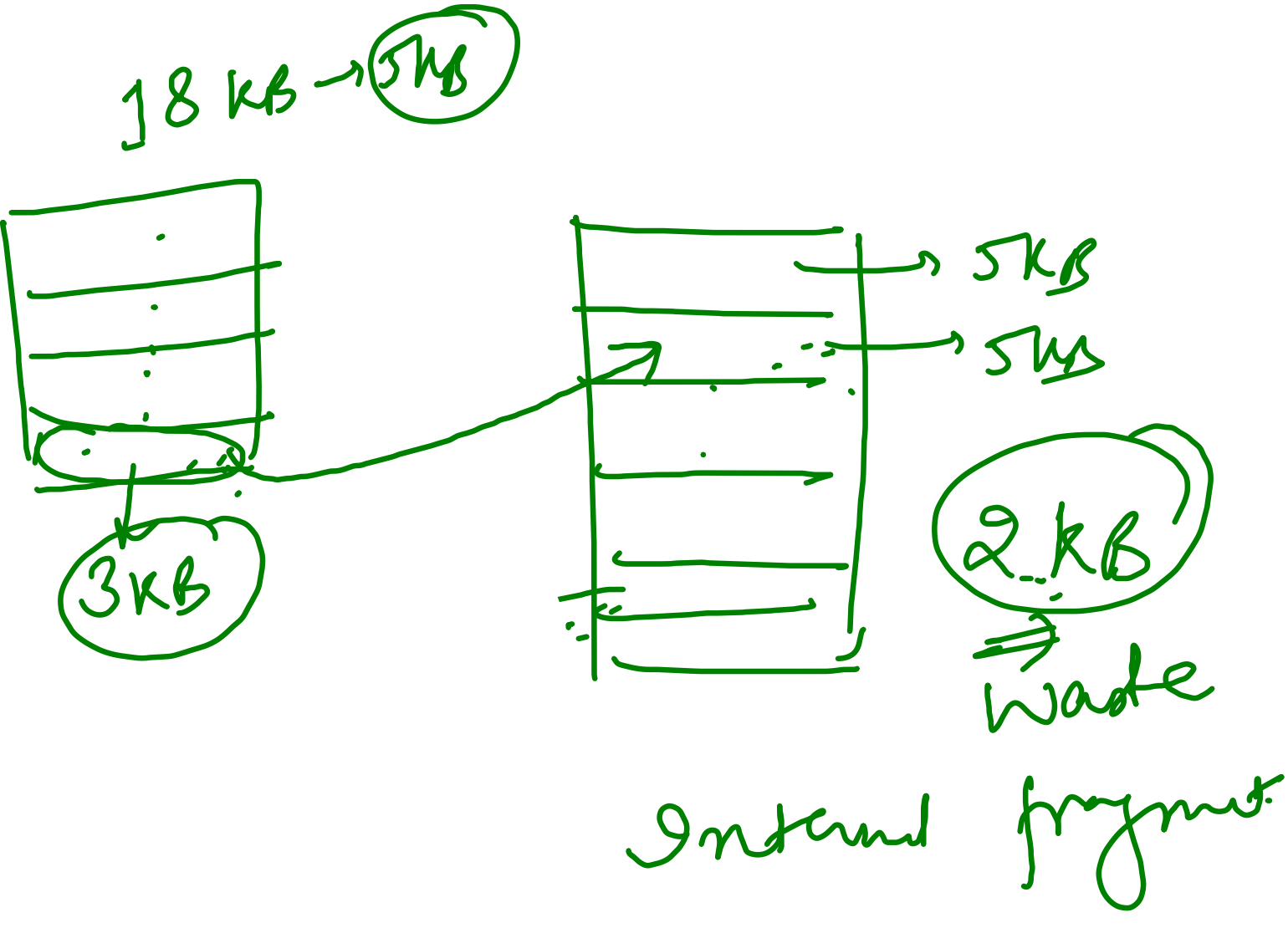
frame ← page



18 KB → 5KB

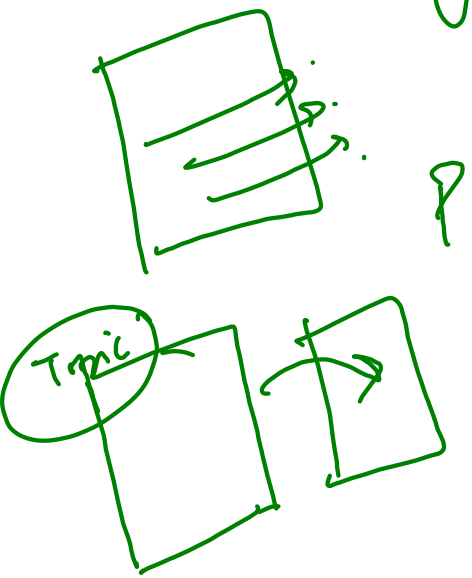






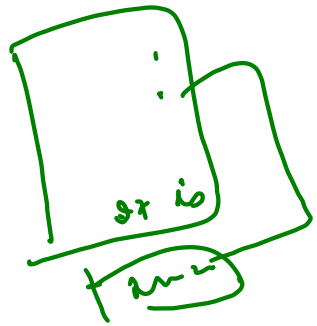
Internal fragmentation

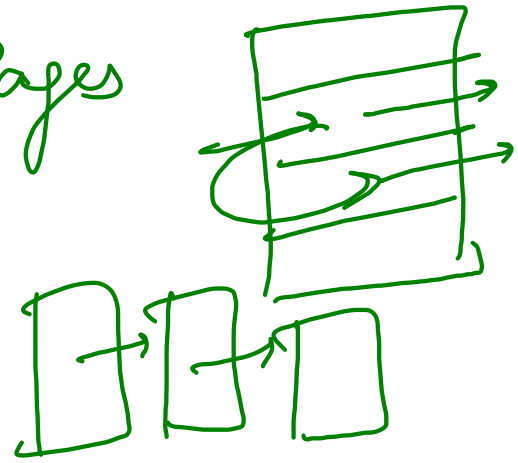
Segmentation → Process → logical segments

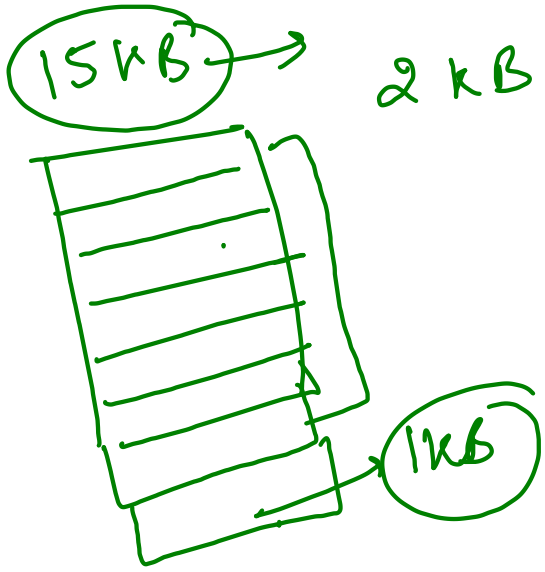


Paging → Process → Pages

h/w → 2KB

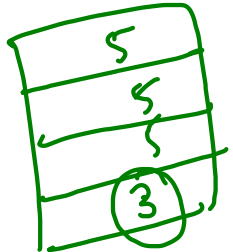






2 x 7 + (1)

18 KB → 5KB

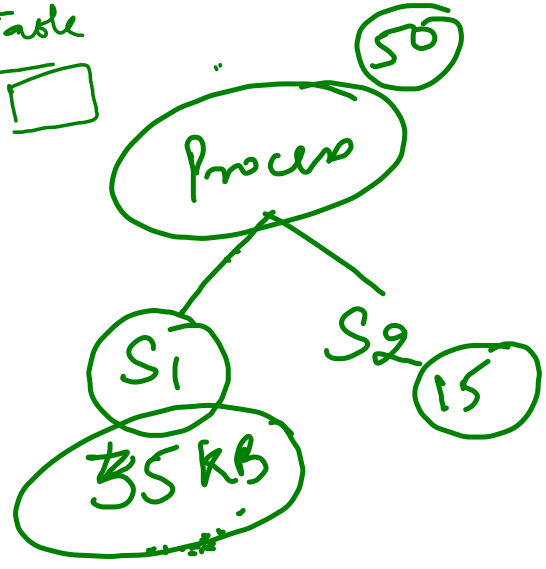


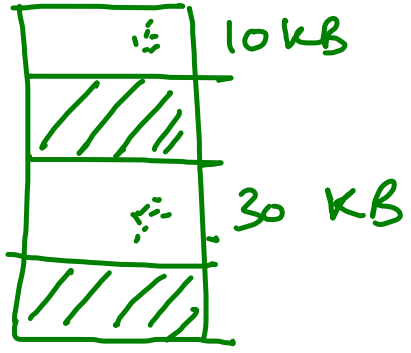
Segmentation → Process → logical segments

External fragmentation

CPU → L.A
↓ MMU
M.M – P.A

Segment Table

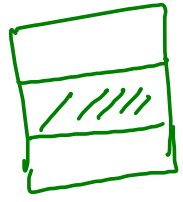




40

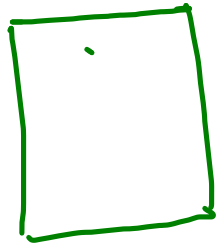
Fixed → Process ✗

Dynamic Process ✗



Paging. Process ✓

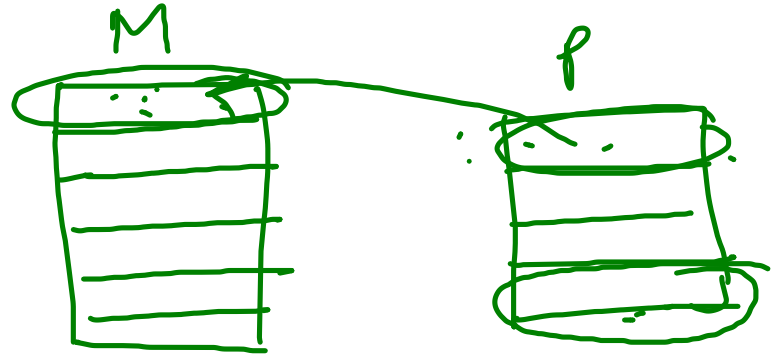
Segmentation Process ✓

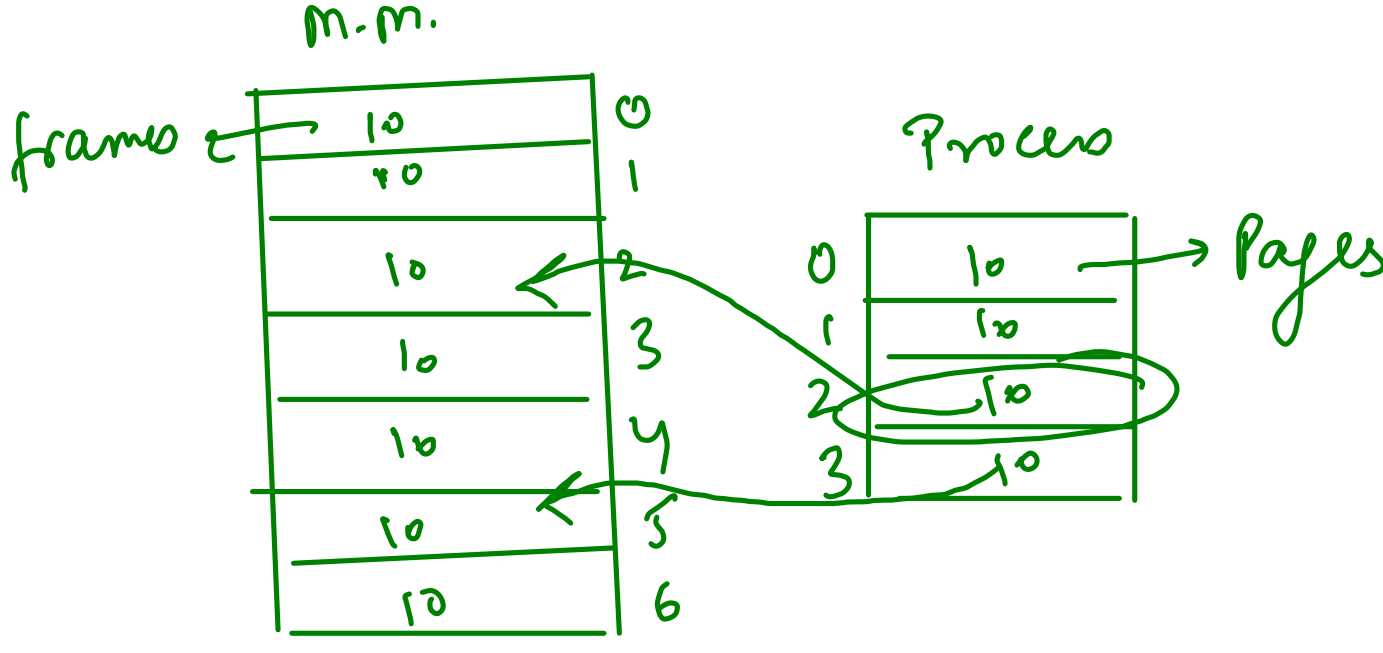


Paging.

Process → Pages

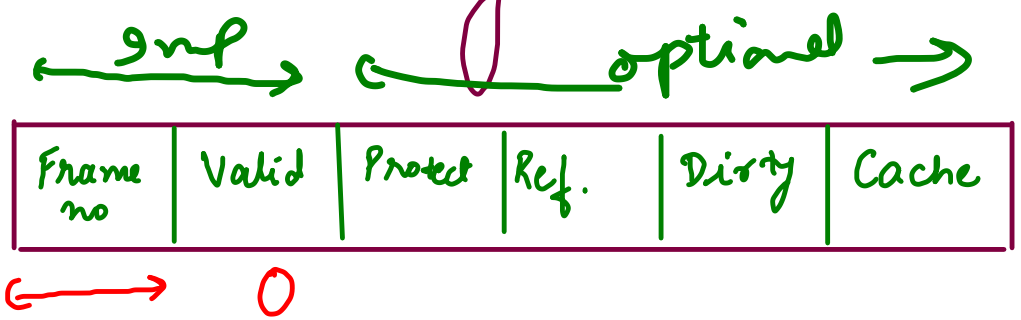
Memory → Frames





Page Table

Page Table Entry



1) Frame no

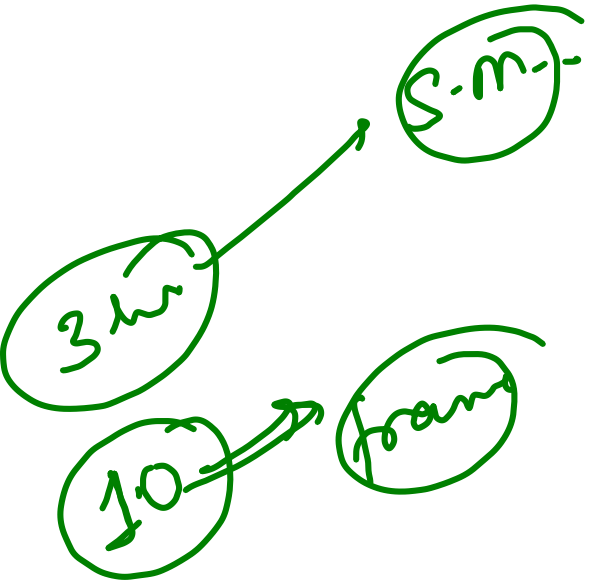
2) Valid - To check whether the page number is valid/invalid
   - 0 → not in memory
   - 1 → in memory

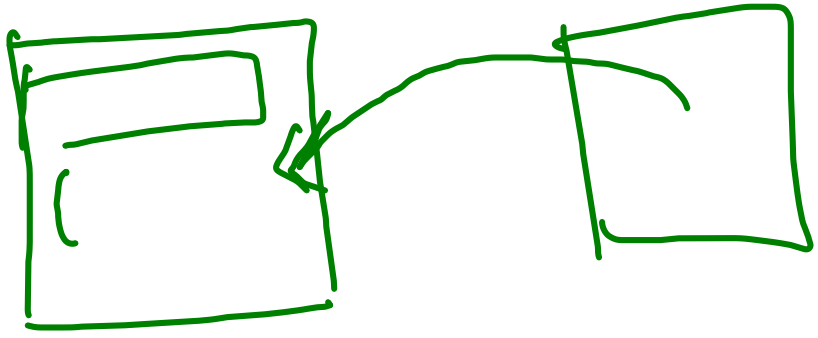
3) Protect - R/W/X → to protect & provide security

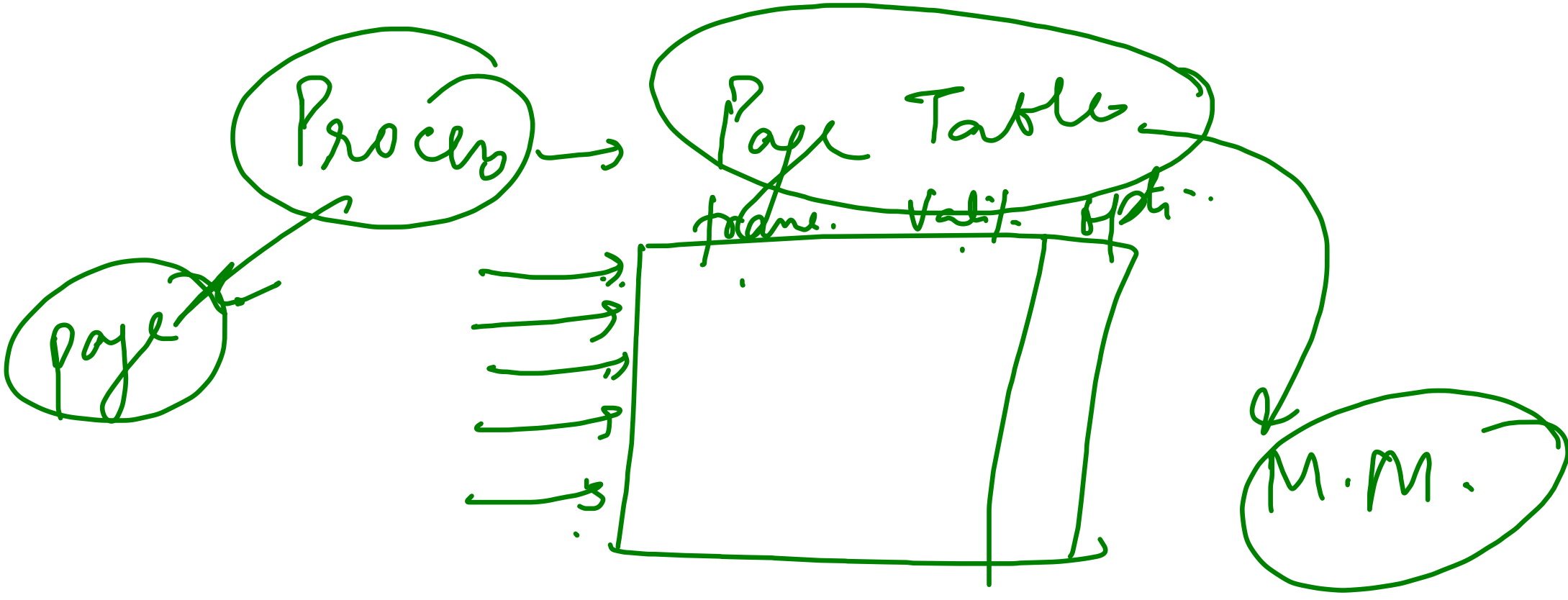
4) Cache - Enable / disable cache

5) Dirty - whether the page has been modified or not.

6) Reference - identify how many times the page is referred.


S.M
3
10



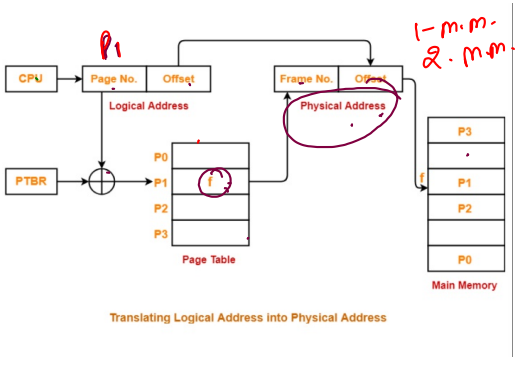

Process
Page Table
Page
M.M.

PS = F.S.

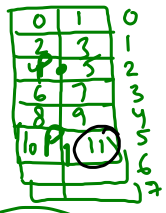
Address Translation

logical address → page no & offset

Physical address → frame no & offset



Translating Logical Address into Physical Address



Process → 4B

Page size ⇒ 2B ⇒ 4/2 ⇒ 2

1B

P0
P1

P0 | 0 | 2
P1 | 3 | 3

P.T.

0	f2
1	f5

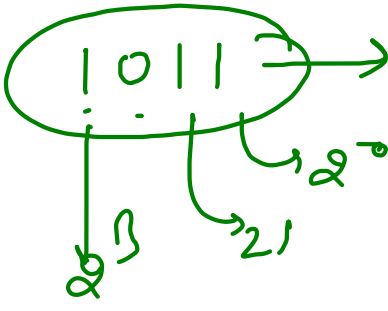
L.A. | 1 | 1 |

4 + of 1 ⇒ 5

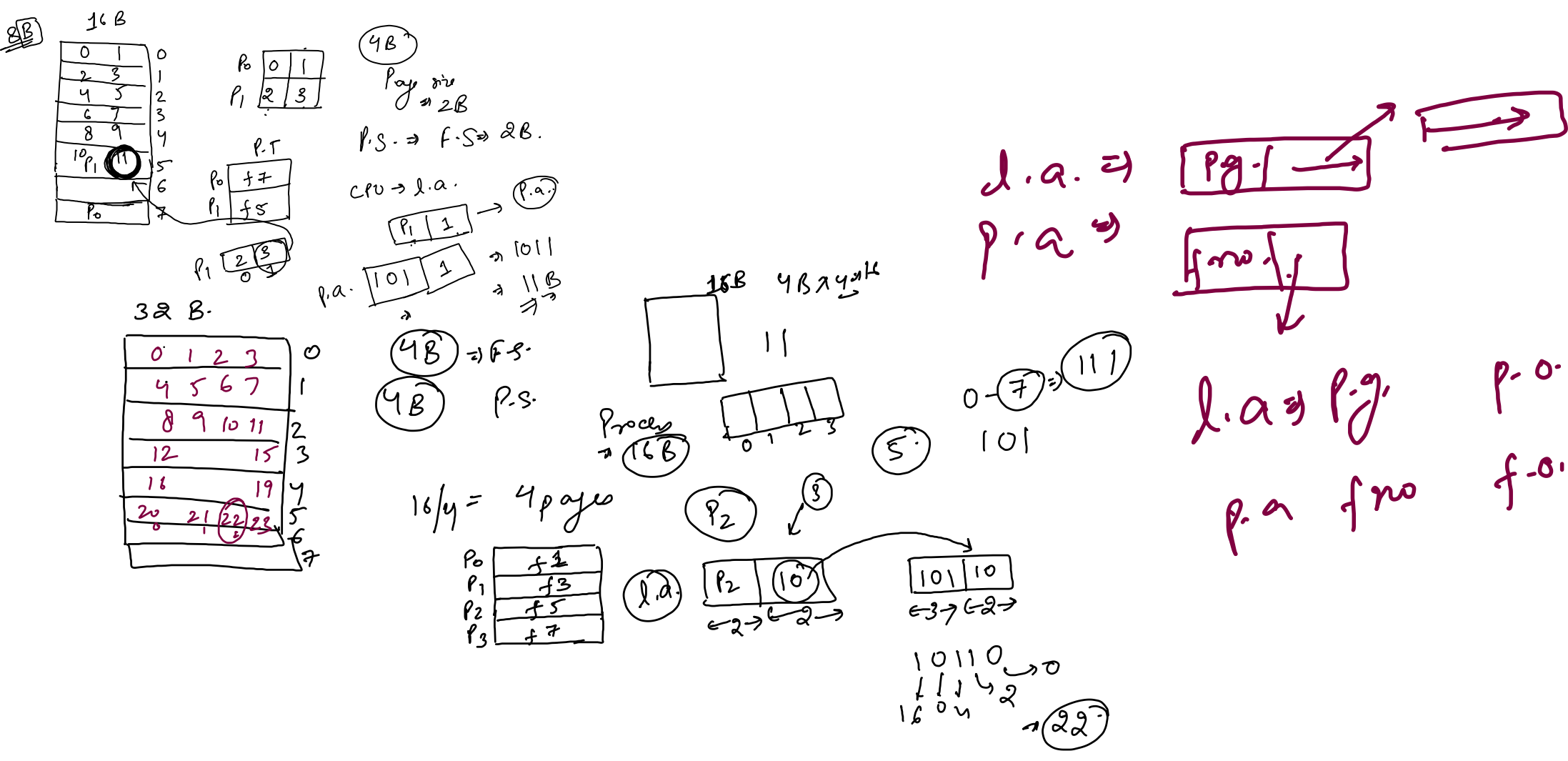
| 1 | 0 | 1 |

2^2 2^1 2^0

P.A. | 101 | 1 |



8 + 2 + 1 ⇒ 11


8B
16 B
4B
Page size ⇒ 2B
P.S. ⇒ F.S ⇒ 2B.
P.T
CPU → l.a.
P.a.
32 B.
4B ⇒ F.S.
4B P.S.
16B 4B × 4 = 16
Process ⇒ 16B
16/4 = 4 pages
l.a.
5.
l.a. ⇒
p.a ⇒
Pg.
fno.
l.a ⇒ Pg. p.o.
p.a fno f.o.
10110
22

l.a. ⇒ page no , page offset

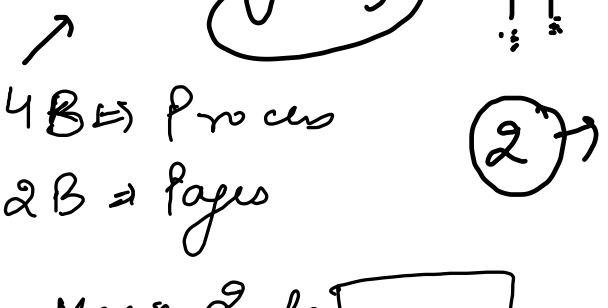
p.a ⇒ frame no frame offset

P.S. = F.S.

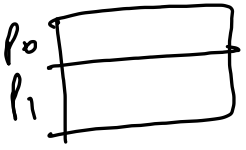
8B ⇒ 0 – 7

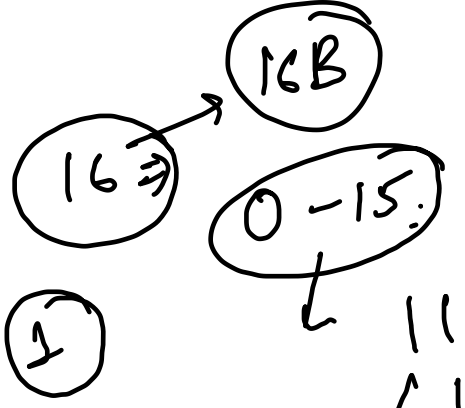
4B ⇒ Process

2B ⇒ Pages



Max ⇒ 2





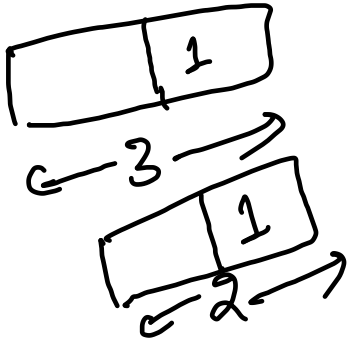
1111 → 2^4

2^5 → 2^4 2^1

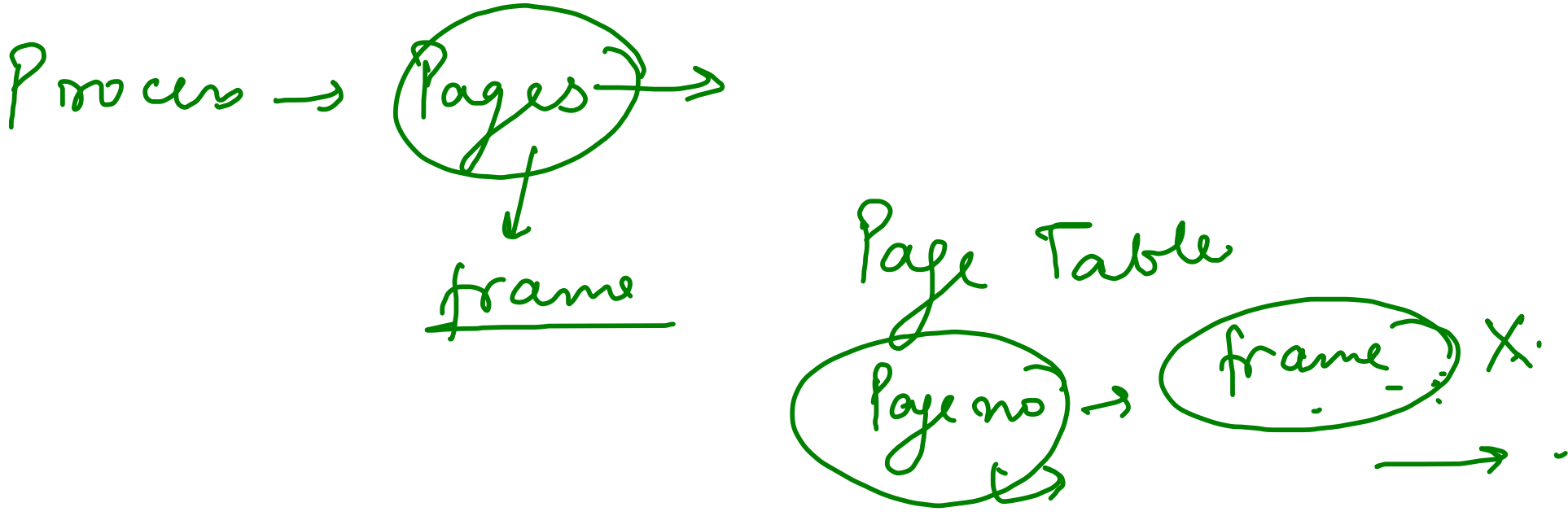
8 + 4 + 2 + 1

⇒ 15

p.a.



l.a


Process
Pages
frame
Page Table
Page no
frame
X

1) All pages are of same size =>
2) All frames are of same size
3) Size of frame = Size of page
4) Page size is defined by h/w.

$2^2 \times 2^{30}B = 2^{32}B$ GB $\Rightarrow 2^{30}B$

LAS = 4GB = 2^{32}B Try this numerical!

PAS = 64MB = 2^{26}B

Page Size = 4KB = 2^{12}B

No. of pages = ? 2^{20}

No. of frames = ? 2^{14}

LAS = 8GB $\Rightarrow 2^3 \times 2^{30} \Rightarrow 2^{33}$B

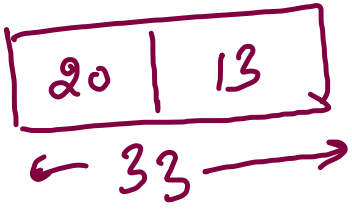
PAS = 16GB $\Rightarrow 2^4 \times 2^{30} = 2^{34}$B

Page size = 8KB $= 2^3 \times 2^{10} \Rightarrow 2^{13}$B

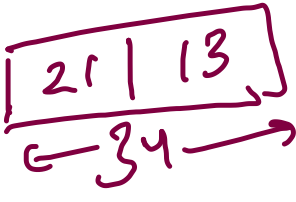
No. of pages = ? 2^{20}

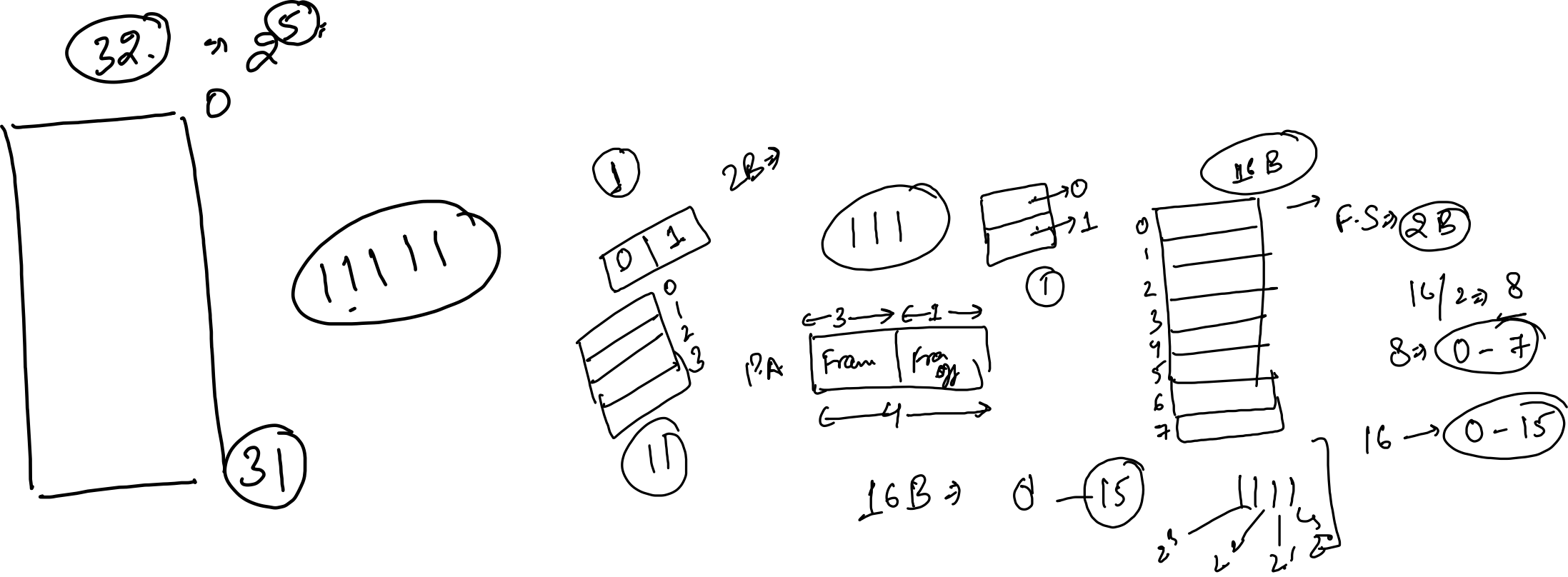
No. of frames = ? 2^{21}

LAS $\Rightarrow$



PAS $\Rightarrow$



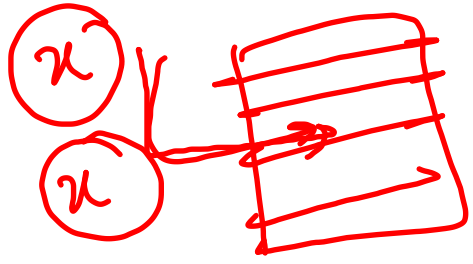

32
2^5
0
31
11111
1
2B
0 1
0
1
2
3
11
111
0
1
1
PA
3
1
Frame
Frame offset
4
16 B
F.S
2B
0
1
2
3
4
5
6
7
16/2 ⇒ 8
8 ⇒ 0-7
16 → 0-15
16B ⇒ 0 — 15
1111
2^3
2^2
2^1
2^0

Paging

Performance of paging

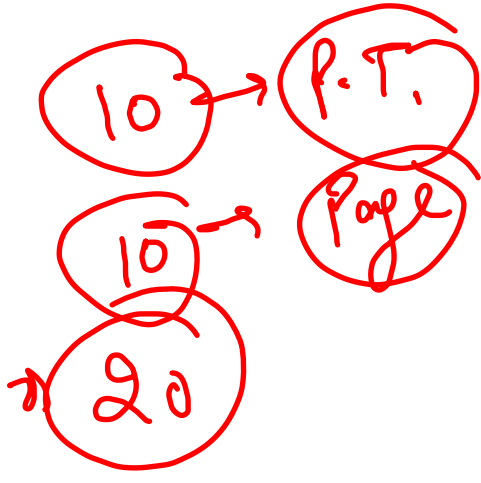
TLB

1) Page Tables are also stored in m.m.
2) Pages are also stored in m.m.



Effective mem. access time = one for accessing page table + one for accessing page

r + r



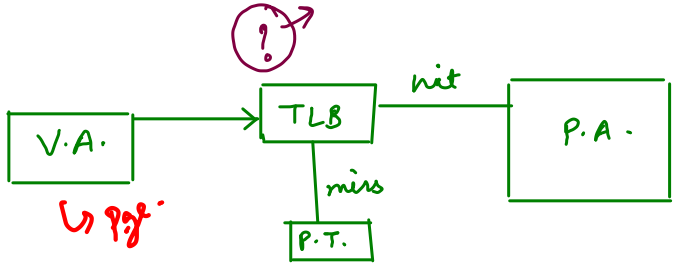
Translational lookaside buffer (TLB)

→ cache that holds recently accessed page table entries

→ hardware device implemented using registers.

→ TLB access time is very less compared to m.m. access time.

→

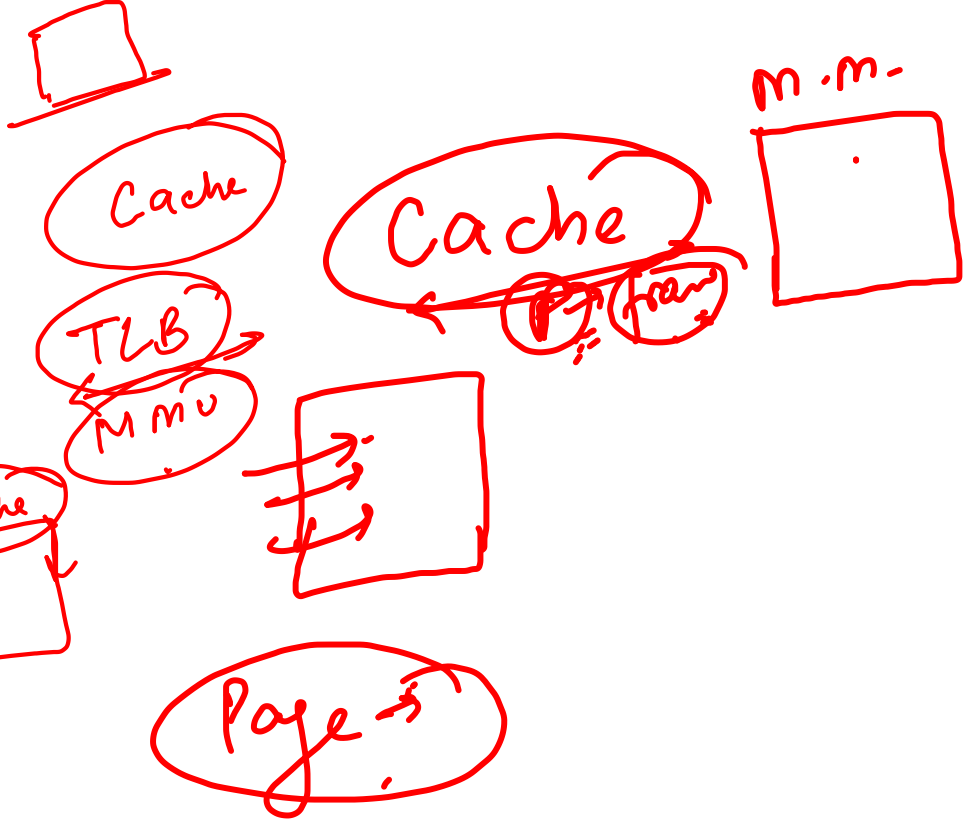


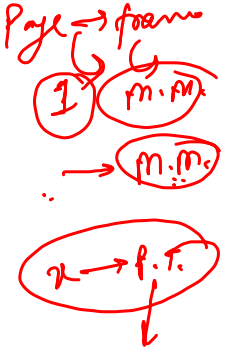
If TLB hits → page no. is found in TLB.

→ frame no. is accessed from TLB directly.

If TLB miss → page no. is not found in TLB.

→ extra memory reference is needed to access the frame no. from P.T.





m.m. → P.T.

m.m → Page

hit ⇒ 90%

miss = 10%

EMAT = hit(TLB fn) + miss(TLB fn + fn)

$$0.9(y+x) + 0.1(y+x+x)$$

TLB hit ⇒ Page → Page Table → cache → frame

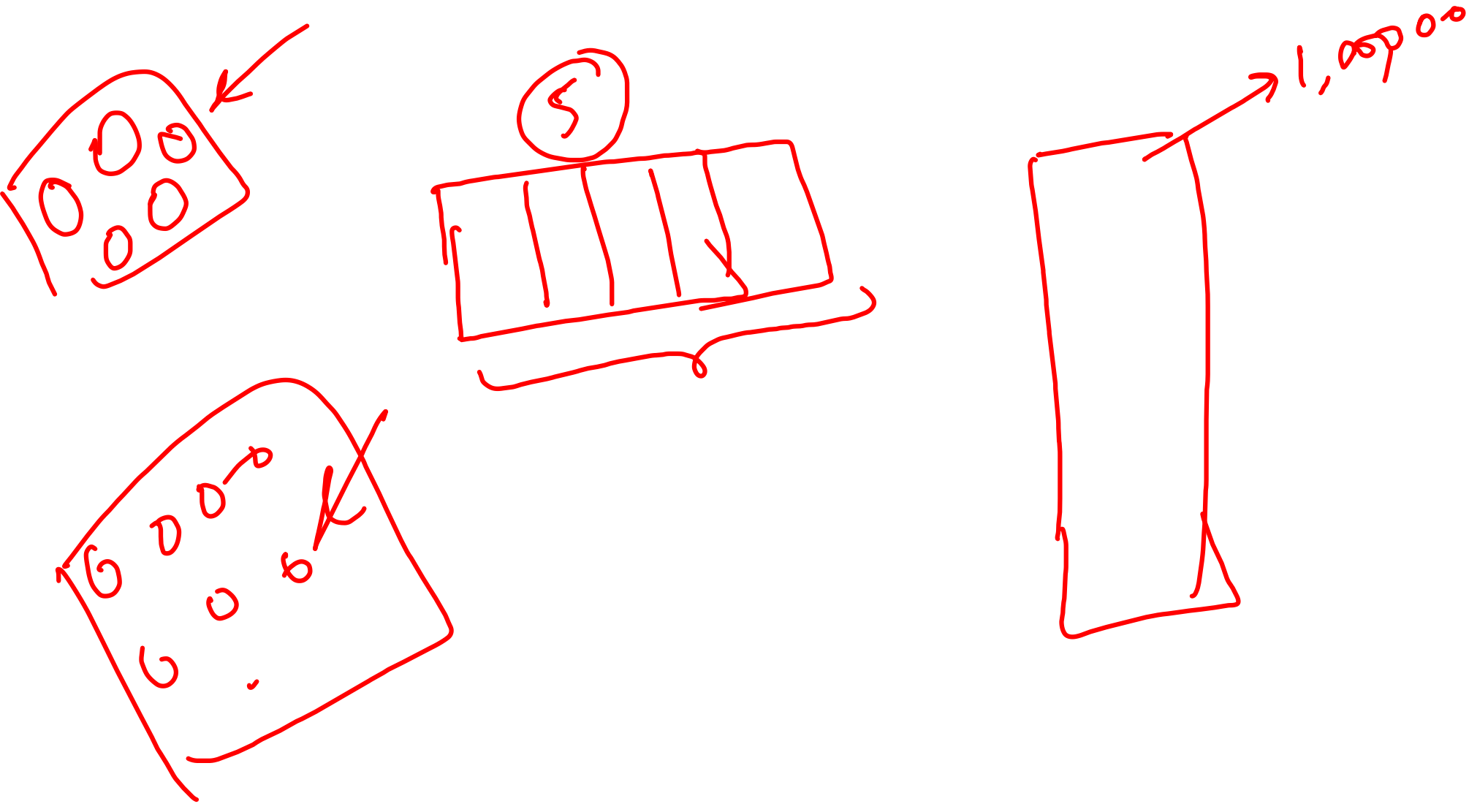
1 → M.M.

TLB miss → Page → Page Table ✗ → 1

M.M → Page Table

P.T → frame

M.M. — frame → 2


5
1,000

1) Memory access time = 100ns.

TLB access time = 20ns

TLB hit ratio = 95%

Calculate EMAT.

EMAT = hit(TLB + m.a.) + miss(TLB + m.a. + m.a.)

$0.95(20 + 100) + (0.05)(20 + 100 + 100)$

$\Rightarrow 0.95 \times 120 + 0.05(220)$

$114 + 11$

$\Rightarrow$ 125 ns

Memory access time = 100ns.

TLB access time = 20ns

TLB hit ratio = 95%

100 - 95%

⇒ 5%

Calculate EMAT

Effective memory access time.

EMAT ⇒ hit (TLB + m) + miss (TLB + m + m)

0.95 (20 + 100) + 0.05 (20 + 100 + 100)

⇒ 120 × 0.95 + 0.05 (220)

114 + 11

125 ns

TLB = 125 ns

200 ns

P.M. → Pg

M.M ⇒ Page

Paging → Frames, pages

Process

m.m

logical division →