- Caching disabled bit → Helps to protect from using stale information of cache.
- referenced bit :→ whether the page is referenced or not.
- modified bit :→ Whether the page has been modified or not. So, that modified result can be kept at secondary memory. Also called as Dirty bit.
- Protection bits :→ It is a field containing several bits. It contains access information i.e. read or read/write or read/write/execute etc.
- Present/Absent bit :→ It says whether the page is present in the MM or not.

[Demand Paging → All the pages are not loaded at once, depending upon the requirement or demand, pages are loaded into memory.]

[Page Fault → CPU wants a page that is not present in the main memory.]

- Page Frame Number :→ It gives the frame number in M which the current page CPU looking for, is present.

29 - GATE 2004 question on Page Table Entry

In a virtual memory system, size of virtual address is 32 bit, size of physical address is 30 bits, page size is 4KB, and size of each page table entry is 32-bit. The main is byte addressable. Which of the following is maximum number of bits that can be used for storing protection and other information, in each page table entry?

a) 2, b) 10, c) 12, d) 14

VA = 32 bits → VAS = 2^{32} B PTE = 32 bits

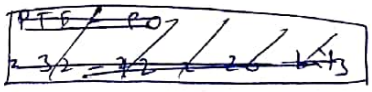
PA = 30 bits → PAS = 2^{30} B FS = PS = 4KB = 2^{12} B

PO = 12 bits

FNO = PA − PO

= 30 − 12

= 18 bits



∴ PTE − FNO

= 32 − 18 = 14 bits → Ans → C

PTS = No of pages * PTE

= $2^{(VA-PO)} * PTE = 2^{(32-12)} * 32$

= $2^{20} * 32$ = 32 MB

30 - Need For Multilevel Paging

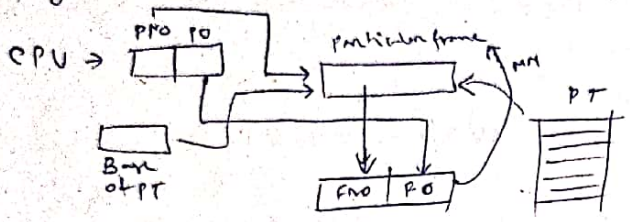
- Working set :→ The no. of pages the process can refer at any point of time.
- Locality of reference generally works with working set.

LA = 22 bits → LAS = 2^{22} B Page size = 4KB → PO = 12 bits

Page Number/PNO = LA − PO = 22 − 12 = 10 bits

No. of pages = 2^{10} B, PTE = 4B → PTS = 1K * 4B = 4KB

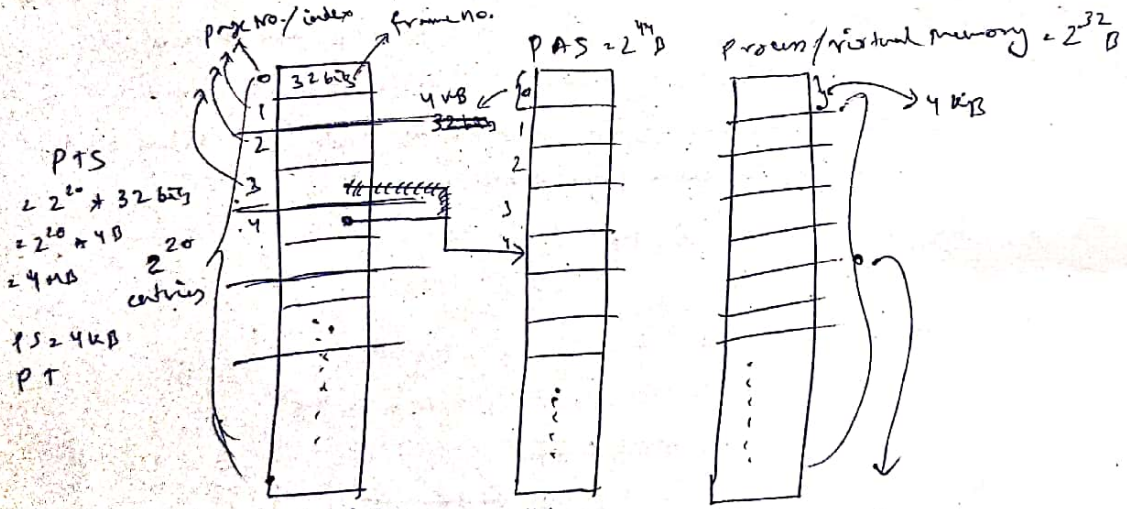
- To run this process, the Page Table should always be in the main memory. & MM is divided into frames, so PT should always divided into pages incase if it is too big to fit in one frame. So the PT size must be equal to the process size (no. of pages) & then PT must be kept in MM.
- Page Table = Page size = 4KB., so the entire PT can be kept in one frame in the main memory, otherwise the PT must be divided into no. of pages & put in main memory (PT).



31- Two Level Paging Example

LAS = VAS = 2^{32} B → LA = VA = 32 bits PAS = 2^{44} B

PS = 4KB → PO = 12 bits



So, the problem is size of PTS is so big that it can't fit within one frame of MM. So, PT must be divided again into many pages such that each page can fit in one frame.

$\frac{2^{44}}{2^{12}} = 2^{32}$ frames — 32 bits required to identify each frame

$\frac{2^{32}\text{ B}}{2^{12}\text{ B}} = 2^{20}$ pgs — 20 bits required to identify each page

No of entries in PT = No. of pages in Process

No of pages = $\frac{\text{PT size}}{\text{Page size}} = \frac{4\text{MB}}{4\text{KB}}$ = 1K pages are present in PT

So now one more PT is needed to keep track of pages of the 1st PT that are placed in the frames of MM.

No of entries in 2nd PT = No of pages in 1st PT

Size of one entry in 2nd PT is also 32 bits as it contains the frame number in which the page is going to be placed.

2nd PT = $2^{10} \times 4B$

= 4KB

PS = 4KB ∴ So, Now problem is solved. Page can be placed in MM.

∴ 12 bits are needed to identify one addressable unit / Byte out of 2^{12} Bytes.

(1st PT) In PT, How many entries are present in each page?

∴ Each entry size = 4B

Page size = 4KB ∴ No of entries = $\frac{4KB}{4B}$ = 1K entries = 2^{10}

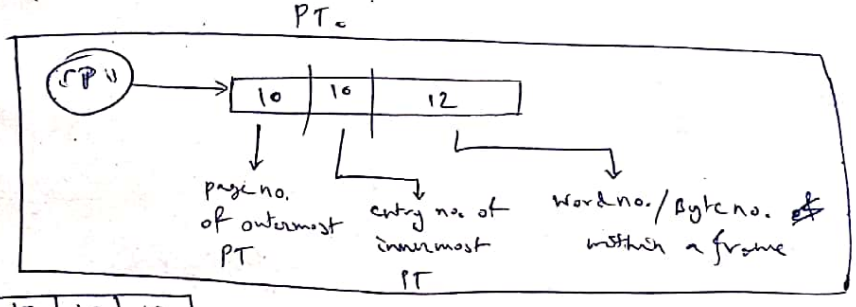
So, 10 bits are required to identify each entry in the PT.

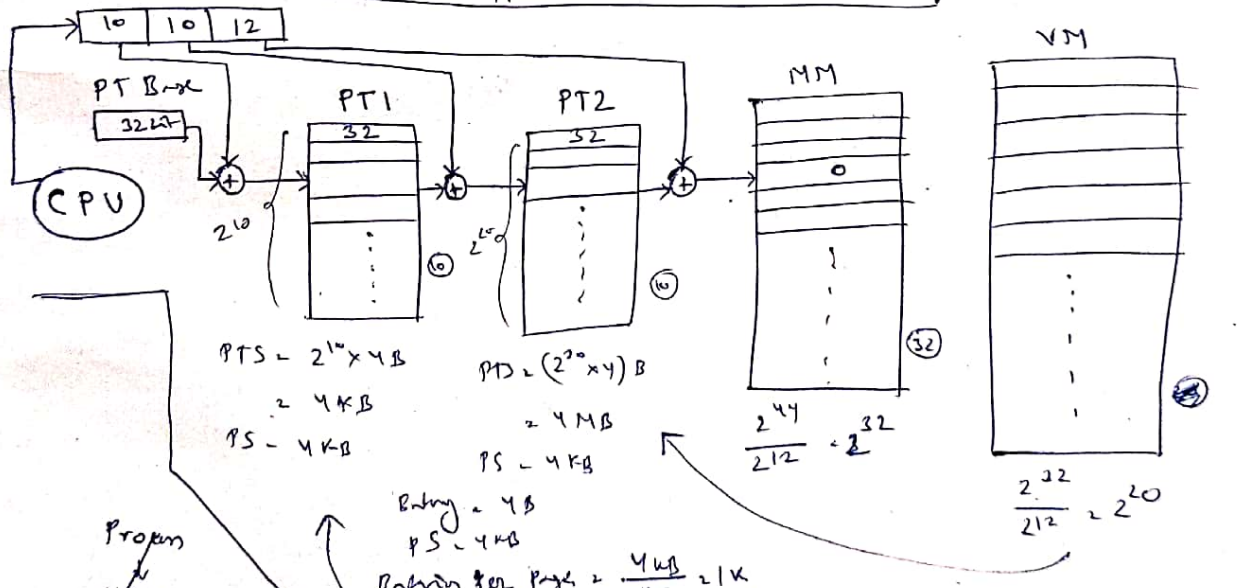
(2nd PT)

No. of pages in 2nd PT = 2^{10}

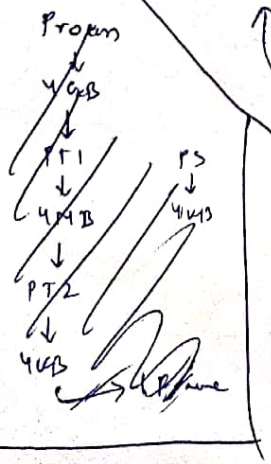
∴ 10 bits for page identification

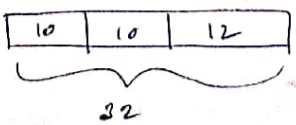
PT Base Address → containing address of the first entry of 2nd PT (outermost) because this PT is kept in MM. This points to base address of the frame that contains the outermost PT.

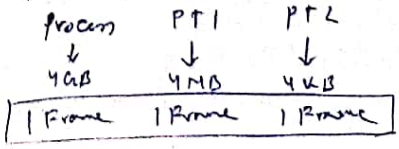




Entry = 4B
PS = 4KB
Entries per Page = $\frac{4KB}{4B}$ = 1K = 2^{10}







* $3 \times 4KB = 12KB$ Memory is enough to run this process

- Thrashing :- For every instruction executed, it every time a page is swapped out & a page is get into MM, then it is called thrashing.
  ∴ Access time = Page Fault time

So, the balance b/w no. of frames allocated to a process & the page sizes and other things helps to avoid thrashing & reduce memory access time.

32 - Example on Multilevel Paging

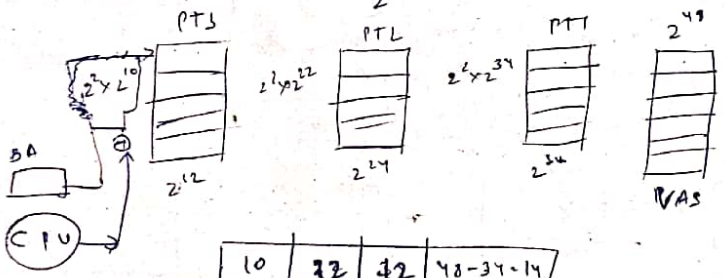
(✓ → given, ? → not given)

VA	Page size	PTE	PT1	PT2	PT3	Address Split
48 b	16 KB	4B	$2^{34} \times 2^2 = 2^{36}$ B	$2^{22} \times 2^2 = 2^{24}$ B	$2^{10} \times 2^2 = 2^{12}$ B	10 | 12 | 12 | 14 (PT3 PT2 PT1 PO)
64 b	1 MB	4B	$2^{44} \times 2^2 = 2^{46}$ B	$2^{24} \times 2^2 = 2^{26}$ B	$2^8 \times 2^2 = 2^{10}$ B	8 | 18 | 18 | 20 (PT1 PT2 PT3 PO)
72 b	1 GB	4B				
72 b	256 MB	4B				
72 b	16 MB	4B				

i) VA = 48 bits → VAS = 2^{48} B LAS = VAS & LA = VA

PS = 16 KB = 2^{14} B → No of Pages (PT1) = $\frac{VAS}{PS} = \frac{2^{48}}{2^{14}} = 2^{34}$

∴ PT2 (No of Pages) = $\frac{2^{36}}{2^{14}} = 2^{22}$ ∴ PT3 (No of Pages) = $\frac{2^{24}}{2^{14}} = 2^{10}$ KB



No. Entries present in each page = $\frac{16 KB}{4B} = 4K = 2^{12}$

10	12	12	48-34-14
PT3	PT2	PT3	PO

PNO

ii) VA = 64 bits → VAS = 2^{64} B

PS = 1 MB → No of Pages (PT1) = $\frac{2^{64}}{2^{20}} = 2^{44}$ B ∴ PT1 = $2^{44} \times 2^2 = 2^{46}$ B

∴ PT2 = $\frac{2^{46}}{2^{20}} = 2^{26} \rightarrow 2^{26} \times 2^2 = 2^{28}$ B

PT3 = $\frac{2^{28}}{2^{20}} = 2^8 \rightarrow 2^8 \times 2^2 = 2^{10}$ B ∴ $\frac{PS}{PTE} = \frac{1MB}{4B} = \frac{2^{20}}{2^2} = 2^{18}$

8	18	18	64-44=20
PT1	PT2	PT3	PO

[PTE is same in all PT]

If PTS > PS
  Create Another PT ↻
Else
  Stop because PT can fit into 1 page

iii) PT1 = $2^{42} \times 2^2 = 2^{44}$ B

PT2 = $2^{14} \times 2^2 = 2^{16}$ B

14	28	30
PT2	PT1	PO

↓ 42-14

iv) PT1 = $\frac{2^{72}}{2^{28}} \times 2^2 = 2^{72-28} \times 2^2 = 2^{44} \times 2^2 = 2^{46}$ B

PT2 = $\frac{2^{46}}{2^{28}} \times 2^2 = 2^{18} \times 2^2 = 2^{20}$ B

18	26	28
PT2	PT1	PO

46-18

v) PT1 = $\frac{2^{72}}{2^{24}} \times 2^2 = 2^{48} \times 2^2 = 2^{50}$ B

PT2 = $\frac{2^{50}}{2^{24}} \times 2^2 = 2^{26} \times 2^2 = 2^{28}$ B

PT3 = $\frac{2^{28}}{2^{24}} \times 2^2 = 2^4 \times 2^2 = 2^6$ B

4	22	22	24
PT3	PT2	PT1	PO

33 – More Examples on multilevel Pages

Page Size	PTE	Outer Page Table Size	Levels of Paging	Virtual Address Space
4 KB	4B	4 KB	1	
4 KB	4B	4 KB	2	
4 KB	4B	4 KB	3	
4 KB	4B	256 B	1	
4 KB	4B	256 B	2	
4 KB	4B	256 B	3	

Assume that except the outermost page tables, all the pages of inner level PT are ~~complete~~ completely full,

i) OPTS = 4 KB, PTE = 4B ∴ No of pages = $\frac{4\text{KB}}{4\text{B}} = 1\text{K} = 2^{10}$ pages are present in process

No of pages = $\frac{\text{VAS}}{\text{PS}}$ ⇒ VAS = PS * No of pages

⇒ VAS = 4KB * 2^{10}

⇒ VAS = $2^{12} * 2^{10}$ ⇒ VAS = 2^{22} B = 4 MB

ii) No of entries in outer PT = $\frac{4\text{KB}}{4\text{B}} = 1\text{K}$ Here, within 1K entries or specifically pointers will point to a page of the inner PT.

So, number of pages in inner PT = K & within this PT each page containing = $\frac{4\text{KB}}{4\text{B}} = 1\text{K}$ entries. So totally, within inner PT there are K pages & each ~~count~~ containing K entries so total no. entries coming out from inner PT is (K*K) entries. Then finally within these pointing each pointer points to a page belonging to the process.

Total no. of ~~process~~ pages in the process are (K*K) pages.

Size of each page = 4 KB, total size = ~~(K*K*4)B~~ (K*K) pages * (4KB)

= 4 GB

iii) 1st Outermost PT → K entries
2nd " " → (K*K) entries
3rd " " → (K*K*K) entries
PS = 4KB
∴ (K*K*K) * (4KB)
∴ VAS = 4TB.

iv) PTS = 256B = 2^8
∴ No of entries = $\frac{PTS}{PTE} = \frac{2^8}{2^2} = 2^6$ = 64 entries
∴ No of pages is also 64 pages
∴ VAS = 64 * 4KB = 2^{18} B = 256 KB

v) 1st outermost PT → $\frac{256}{4}$ = 64 entries
2nd " " → (64 * 1K) = 64K entries = 2^{16} ($\frac{4KB}{4B}$ = 1K)
∴ No of pages = 2^{16}
∴ VAS = 2^{16} * 4KB
= 2^{16} * 2^{12} B
= 2^{28} B = 256MB

vi) 1st outermost PT → $\frac{256}{4}$ = 64 entries
2nd " " → $\left(64 * \frac{4KB}{4B}\right)$ = 64K
3rd " " → $\left(64K * \frac{4KB}{4B}\right)$ = 64K * 1K
= $2^6 * 2^{10} * 2^{10}$
= 2^{26} entries
∴ No of pages = 2^{26}
∴ VAS = 2^{26} * 4KB = $2^{26} * 2^{12} = 2^{38}$ = 256 GB.

34 – Examples on How to make Page Table fit in One Page

Q:

VAS	PS	PTE
4MB	4KB	
4GB	128KB	
128TB	32MB	
256MB		4B
512KB		2B
16GB		4B

• Fill in the blanks in such a way that PT will fit in one page in single level paging.

i) VAS = 4MB, PS = 4KB
No of pages = $\frac{4MB}{4KB} = 2^{10}$ pages
PTS = No of pages * Size of each entry
∴ No of Pages * Entry size ≤ PS
∴ Entry size ≤ $\frac{PS}{Pages} = \frac{4KB}{1K} = 4B$
∴ PTE = 4B

ii) VAS = 4GB, PS = 128KB
No of Pages = $\frac{4GB}{128KB} = \frac{2^2 * 2^{30}}{2^7 * 2^{10}} = 2^{15}$
∴ Entry size ≤ $\frac{128KB}{2^{15}} = \frac{2^{17}}{2^{15}} = 2^2 = 4B$

iii) VAS = 128 TB, PS = 32 MB

Pages = $\frac{128\,TB}{32\,MB} = \frac{2^7 \times 2^{40}}{2^5 \times 2^{20}} = 2^{22}$

∴ Entry size $\leq \frac{2^{25}}{2^{22}} = 2^3 = 8\,B$

iv) VAS = 256 MB, PTE = 4 B

$= 2^{28}$ B Let PS = 2^K B / P Bytes

Pages = $\frac{2^{28}}{P}$

PTS = $\frac{2^{28}}{P} * 4 = \frac{2^{30}}{P}$

∴ Page size $\geq$ PTS

$P \geq \frac{2^{30}}{P}$

$P^2 \geq 2^{30}$

$P \geq 2^{15}$

∴ PS = 32 KB

v) VAS = 512 MB, PTE = 2 B, PS = P B

Page = $\frac{2^{19}}{P}$ ∴ PTS = $\frac{2^{19}}{P} * 2^1 = \frac{2^{20}}{P}$

Page size $\geq$ PTS

$P \geq \frac{2^{20}}{P}$

$P^2 \geq 2^{20}$

$P \geq 2^{10}$

∴ PS = 1 KB

vi) VAS = 16 GB, PTE = 4 B, PS = P B

Pages = $\frac{2^{34}}{P}$ ∴ PTS = $\frac{2^{34}}{P} * 2^2 = \frac{2^{36}}{P}$

Page size $\geq$ PTS

$P \geq \frac{2^{36}}{P}$

$P^2 \geq 2^{36}$

$P \geq 2^{18}$

∴ PS = 256 kB

35 - GATE 01 and 99 Question On Paging

01 Consider a machine with 64 MB physical memory and 32 bit virtual address space. If the page size is 4 KB, what is the approximate size of PT? a) 16 MB, b) 8 MB, c) 2 MB, d) 24 MB

PAS = 64 MB = 2^{26} B, VAS = 2^{32} B

PS = 4 KB = 2^{12} B, PTS = ?

No. of Pages = $\frac{VAS}{PS} = \frac{2^{32}}{2^{12}} = 2^{20}$

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

PT Entry size = Frame No.

∴ ES = 14 bit

No of Pages * Entry size $\leq$ PS

Entry size $\leq \frac{PS}{Pages} = \frac{2^{12}}{2^5}$

∴ PTS = $2^{20} * 14$ B bits

$= \frac{2^{20} \times 14}{2^3} = 2^{17} \times 14 \simeq 2$ MB

99 Which of the following is/are advantages of virtual memory?

a) Faster access to memory on an average
b) Processes can be given protected address spaces
c) Linker can assign addresses independent of where the program will be loaded in the physical memory.
d) Programs larger than the physical memory size can be run.

Ans → D

36 - GATE Theory Question

95 In a virtual memory system, the address space specified by the address lines of the CPU must be _____ than the physical memory size and _____ than the secondary memory size.

i) larger, ii) smaller

97

Dirty bit for a page in PT -

a) Helps avoid unnecessary writes on paging device
b) Helps maintain LRU information.
c) Allow only read on a page
d) None.

Ans → A

06 A computer system supports 32 bit virtual addressing as well as 32 bit physical addressing. Since the VAS is of same size as PAS, the OS designers decided to get rid of VM entirely. Which of the following is true?

a) Efficient implementation of multiuser support is no longer possible.
b) The processor cache organization can be made more efficient now.
c) H/W support for memory management is no longer needed.
d) CPU scheduling can be made more efficient now.

Ans → A

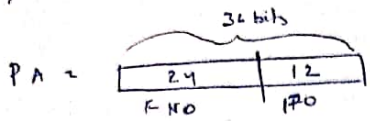
37 - GATE 2008 Question on Multilevel Paging

PA = 36 bits, VA = 32 bits, PS = 4 KB, PTE = 4B.
A three level PT is used. VA is divided as follows:
→ Bits 30-31 are used to index I level PT.
→ Bits 21-29 are used to index II level PT.
→ Bits 12-20 are used to index III level PT.
→ Bits 0-11 are used as offset within page.

No. of bits in PTE for frame no.

Then ~~PTE size~~ in I, II, and III level PT are?

A) 20, 20, 20, B) 24, 24, 24, C) 24, 20, 20, D) 25, 25, 24



PS = 4KB = 2^{12} B, PO = 12 bits Ans → B) 24, 24, 24

(The whole method is ~~done~~ checked ~~again~~ again in the video).

∴ In multilevel paging, the entry size of all PT is always same.

38 - GATE 2009 Question on Multilevel Paging

A multilevel PT is preferred in comparison to a single level PT for translating VA to PA because –

a) It reduces the memory access time to read or write a memory location.

b) It helps to reduce the size of PT needed to implement the virtual address space of a process.

c) It is required by the translation lookaside buffer.

d) It helps to reduce the number of page faults in page replacement algorithms.

Ans → B

39 - GATE 2002 Question Multi Level Paging

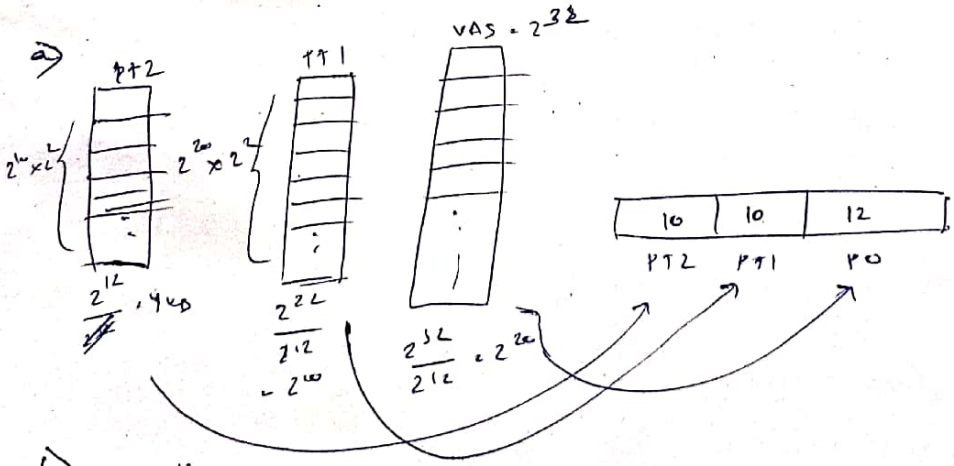
VA = 32, PA = 32, Byte Addressable

PS = 4KB. Two level PT. PTE = 4B

a) Diagram showing VA to PA conversion

b) Page table entries in each page?

c) How many bits are available for storing protection and other info in each PTE.



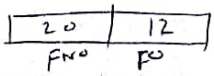
b) $\frac{PS}{PTE} = \frac{2^{12}}{2^2} = 2^{10}$ = 1K entries in each page.

c) PTE = 32 bits PAS = 2^{32} # no of frames = $\frac{2^{32}}{2^{12}} = 2^{20}$

∴ Protection bits = 32 − 20 = 12 bits

PS = 2^{12}

PO = 12



≈ 32 − 12 = 20

32 − 20 = 12

40 - GATE 2008 IT Question

1) Dirty → b) Write-Back Policy
2) R/W → d) Page Protection
3) Reference → c) Page Protection
4) Valid → a) Page Initialization

(Code Segment) → R X (Read & execute)
(Data Segment) → R W (Read & Write)
(Stack) → R W (")

41 - GATE 2013 Question on Multilevel Paging

VA = 46 bits, PA = 32 bits, 3 level paging.
First level page table has exactly one page.
PTE is 32 bits. What is the size of the page in this computer?

PS = P Bytes No of Pages = $\frac{2^{46}}{P}$ No of entries in one page = $\frac{P}{4}$ (1st PT)

$\frac{2^{46}}{P} * 4 \leq P$

$\Rightarrow P^2 \geq 2^{48}$

$\Rightarrow P^2 \geq 2^{24}$

PS = 2^{24} = 16 MB

2nd PT → $P/4 * P/4 = P^2/16$

3rd PT → $P^2/16 \times P/4 = P^3/64$

total no. of pages → $P^3/64$

VAS = $(P/4)^3 * P$

$\therefore 2^{46} = \frac{P^4}{64}$

$\Rightarrow P^4 = 2^{46} \times 2^6$

$\Rightarrow P^4 = 2^{52} \Rightarrow P = (2^{52})^{1/4}$

$\Rightarrow P = 2^{13} = 8$KB

42 - Finding Optimal Page Size

VAS = 4 GB
PS = 4 KB
Pages = $\frac{VAS}{PS} = \frac{4GB}{4KB} = 1M$

PS = 4 MB
Pages = $\frac{VAS}{PS} = \frac{4GB}{4MB} = 1K$

- PS ↑ PTS ↓ / PS ↓ PTS ↑
  - ↓
  - Small overhead
  - Page can be wasted → (P/2) → half of the page

Page size = P bytes, PTE = e bytes
On average VAS = S bytes

Overheads → $\underbrace{\left(\frac{P}{2}\right)}_{\text{Size wasted}} + \underbrace{\left(\frac{S}{P}\right) * e}_{PT}$

$$\frac{dO}{dp} = \frac{1}{2}\phi - \frac{S}{p^2} * e = 0$$

$$\Rightarrow \frac{S}{p^2} * e = \frac{1}{2}$$

$$\Rightarrow 2Se = p^2 \Rightarrow p = \sqrt{2Se}$$

↓

Best possible page size so that overhead is minimal

S	e	p (Best size)
4 KB	8 B	
1 MB	2 B	
256 MB	32 B	

$\rightarrow \sqrt{2 \times 2^{12} \times 2^3} = \sqrt{2^{16}} = 2^8 = 256\ B$

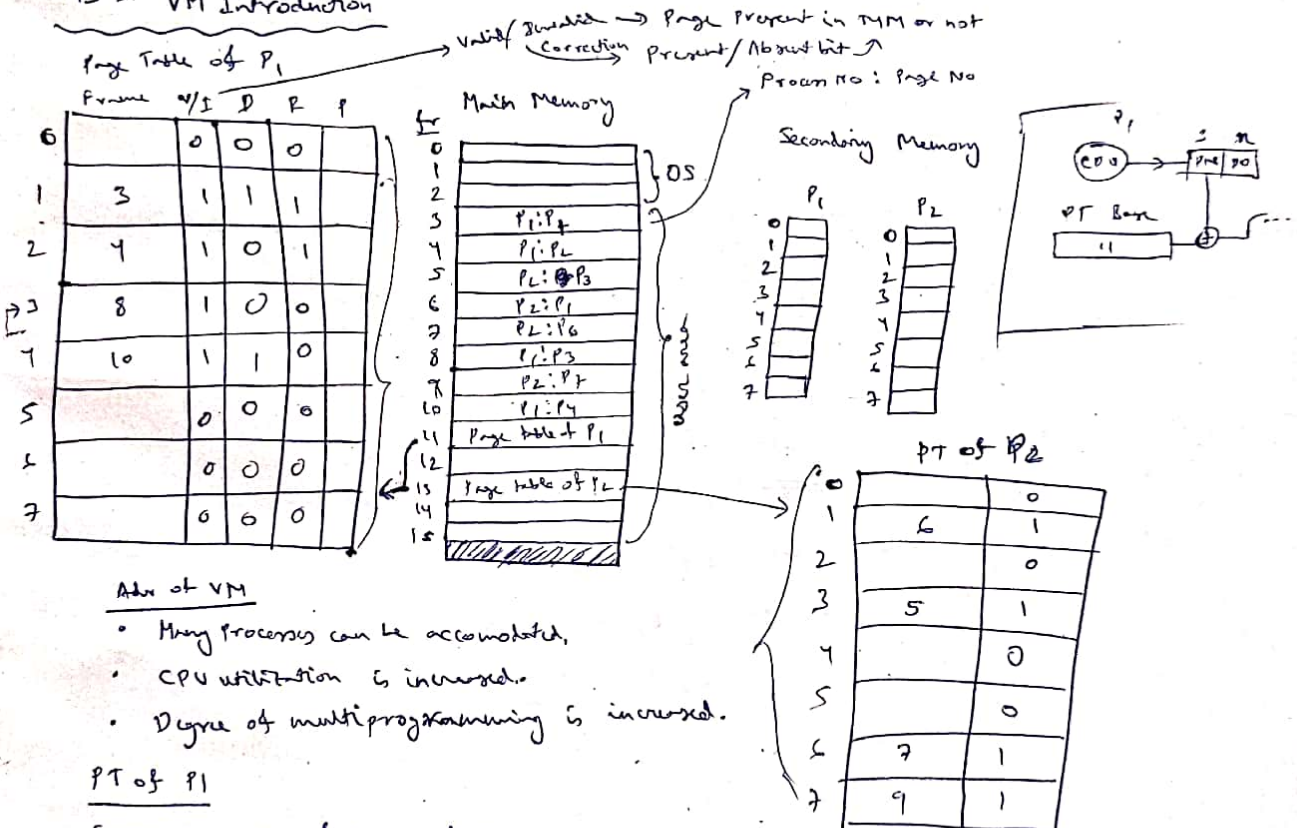
$\rightarrow \sqrt{2 \times 2^{20} \times 2^1} = \sqrt{2^{22}} = 2^{11} = 2\ KB$

$\rightarrow \sqrt{2 \times 2^{30} \times 2^5} = \sqrt{2^{36}} = 2^{22} = 4\ MB$

Process divided into n sections

then total wastage = $\frac{np}{2} + \left(\frac{S}{p} * e\right)$

43 – VM Introduction



Adv of VM

- Many processes can be accomodated.
- CPU utilization is increased.
- Degree of multiprogramming is increased.

PT of P1

- Frame → Consists of Page No of MM
- P/A → ~~Valid~~/Present / ~~Invalid~~/Absent bit → Page is present in MM or not → ~~Present~~ Present → 1, Absent → 0 (may keep)
- D → Dirty bit/Modified bit → A page after getting loaded into the MM ~~has~~ was modified or not. Mainly used for Write back policy so that modification or update can be ~~not~~ reflected on Secondary Memory.
- R → Reference bit → In the last clock cycle, what are the pages that are referred. So, ~~replace~~ this is mainly needed in replacement techniques to take decision whether to replace a page or not.
- P → Protection bit → Whenever trying to access MM, and trying to write something on the code, so the protection will catch it & check its permissions. i.e. Read/Write/Execute (R/W/X).

44 - TLB

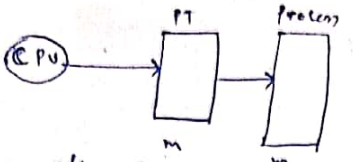
- Paging & Page Table is invented to decrease the wastage of memory due to external fragmentation.

VAS = 4GB, PS = 4KB, Pages = 1M

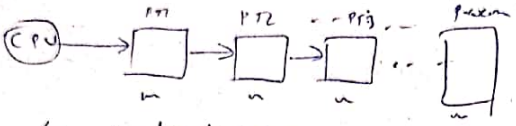
External Fragmentation
↓
Paging
↓
Page Table (Large Space)
↓
Multilevel Paging

Overhead time increased

- PT → Too large → Static Partitioning → Internal Fragmentation



∴ Effective memory access time = 2 × m (m → time needed to read one word from mem)



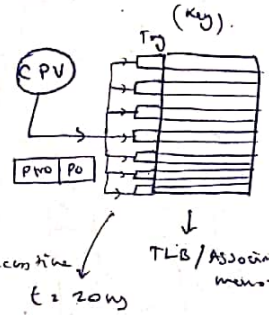
(∵ = No of levels × m + m

Base of PT → Register → Help context ~~switch time~~ Switch time

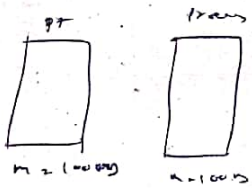
New type of memory → Cache → Faster than MM

↳ Cheaper than Registers & more density than registers

This concept is called TLB (Translation Lookaside Buffer)



- TLB is very small, so it may not hold all the PT, It only stores frequently accessed PTs.



(80%) TLB Hit → 20ns → process → (20 + 100) = 120ns

(20%) TLB Miss → (20 + 100) = 120ns → process → (120 + 100) = 220ns

Effective Memory Access time = 0.8 (20 + 100) + 0.2 (20 + 100 + 100)

= 16.0 + 80.0 + 4.0 + 20.0 + 20.0

= 140 ns

$$P(t+m) + (1-P)(t+m+m)$$

P → hit probability; (1−P) → miss probability

Multilevel PT → $P(t+m) + (1-P)(t+km+m)$

k → no. of levels

- Mention Process Id/PID or Flush the TLB to keep track of processes & avoid confusion about Pg no.

45 - GATE 2008 Question on TLB

A paging scheme uses TLB. A TLB access takes 10 ns and main memory access takes 50 ns. What is the effective access time (in ns) if the TLB hit ratio is 90%, and there is no page fault?

a) 54, b) 60, c) 65, d) 75

$$EMAT = 0.9(10+50) + 0.1(10+50+50)$$
$$= 54 + 11 = 65 \text{ ns} \rightarrow C.$$

46 - Numericals on TLB

TLBA	MA	TLBH	PT Levels	EMAT
20 ns	100 ns	80%	1	___
20 ns	100 ns	80%	2	___
20 ns	100 ns	80%	3	___
20 ns	100 ns	___	1	130 ns
___	100 ns	60%	1	160 ns
20 ns	___	50%	1	170 ns

* Assume No Page fault.

i) $EMAT = 0.8(20+100) + 0.2(20+100+100)$
$= 96 + 44 = 140$ ns

ii) $EMAT = 0.8(20+100) + 0.2(20 + 2*100 + 100)$
$= 96 + 64 = 160$ ns

iii) $EMAT = 0.8(20+100) + 0.2(20+3*100+100)$
$= 96 + 84 = 180$ ns

iv) $130 = p(20+100) + (1-p)(20+100+100)$
$= 120p + 220 - 220p$
$\Rightarrow 100p = 90 \Rightarrow p = \frac{90}{100} \Rightarrow p = 90\%$

v) $160 = 0.6(t+100) + 0.4(t+100+100)$
$\Rightarrow 160 = 0.6t + 60 + 0.4t + 80$
$\Rightarrow t = 20$ ns

vi) $170 = 0.5(20+m) + 0.5(20+m+m)$
$\Rightarrow 170 = 10 + 0.5m + 10 + 0.5m + 0.5m$
$\Rightarrow 1.5m = 150$
$\Rightarrow m = \frac{150}{1.5} \times 10 \Rightarrow m = 100$ ns

47 - TLB Summary

TLB access time = t units

Main memory access time = m units

TLB miss rate = p

∴ Effective memory access time (EMAT) =

~~p(t+m)+m~~

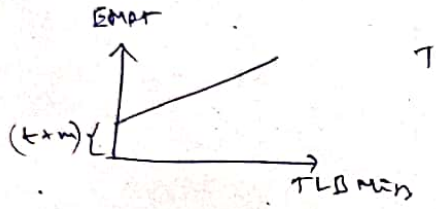
$= p(t+m+m)+(1-p)(t+m)$

$= p(t+2m)+(1-p)(t+m)$

$= pt + 2pm + t + m - pt - pm$

$= pm + t + m$

∴ EMAT = $pm + (t+m) \Rightarrow y = ax + c$



TLB hit = $t+m$

For multilevel paging → PT Level = k

EMAT = $p(t + km + m) + (1-p)(t+m)$

$= t + m + pkm$

48 - Page Fault

- Demand Paging :→ Don't load any page until required.
- Page Fault :→ Referring to a page which is not present in MM.
- Page Fault → Interrupt → User mode $\xrightarrow{(CS)}$ kernel mode → page replacement (Context Switching)

↪ kernel mode $\xrightarrow{(CS)}$ user mode → process again started → Page hit

MA - ns
CS - ns
copy - nxc
CS - nxc
} time

Page Fault ↑ Total time ↑

- Thrashing :- If no. of page faults is almost equal to no. of page references, then maximum time will be wasted by doing copy from HDD to MM, this is called thrashing.

solution

better page replacement technique ⟶ less thrashing → less execution time

∴ EMAT = P.(Service time or Page fault time + memory access) + (1−P).(memory access time)

Page fault rate → Service time or Page fault time (+ memory access)

Page hit rate

- time: ms → ns → negligible
- TLB Hit / Miss
- Page Hit / Fault

49 – GATE 2001 Question on Page Fault

Let the page fault service time be 10 ms in a computer with average memory access time being 20 ms. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory?

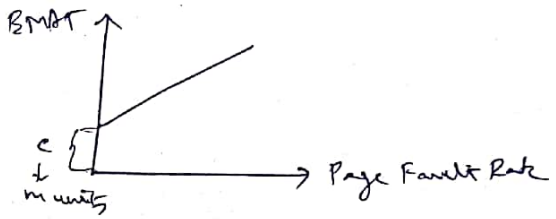
A) 21 ms, B) 30 ms, C) 23 ms, D) 35 ms

EMAT = P(PS + MA) + (1−P)(MA) ∵ (MA = m)

= P(PS) + Pm + m − Pm

= P(PS) + m

y = mx + c



∴ EMAT = P(PS) + m

= $\frac{1}{10^6}$ × (10 ms) + 20 ms

= 20 ms + 10 ms = 30 ms → B

50 – GATE 98 Question on Page Fault

If an instruction takes 'i' (μs) micro seconds and a page fault takes an additional 'j' msec, the effective instruction time if on the average a page fault occurs every k instructions is:

a) i + (j/k) b) i + j * k c) (i+j)k d) (i+j) * k

EMAT = P(PS) + m

= $\frac{1}{k}$. j + i = i + (j/k) → a

51 – GATE 2000 Question on Page Fault

Suppose the time to service a page fault is on average 10 msec, while memory access takes 1 msec. Then a 99.99% hit ratio results in an average memory access time of – a) 1.9999 msec b) 1 msec c) 9.999 msec d) 1.999 msec

∴ EMAT = P(PS) + m → = (0.01%) * 10 msec + 1 ms

P = (1 − 99.99) = 0.01% → = 0.0001 * 10 ms + 1 ms

= 100 * 10 msec + 1 ms = 1000 ms + 1 ms

0.0001 = 10^{-4}

10 ms = 10×10^{-3} ms → 10^{-4} ms

= 1 ms + (10^{-4} * 10^{4}) ms

= 1 ms + 1 ms = 2 ms → d

EMAT = $p(ps + m) + (1-p)(m)$

$= p(ps) + pm + m - pm$

$= p(ps) + m$

1.99 $\Rightarrow$ EMAT $= p(ps) + (1-p)m \Rightarrow m$ is negligible

$= 0.0001\,(10 \times 10^{-3}) + (9999)(10^{-6})$

$= 10^{-6} + 0.9999 \times 10^{-6} = 1.9999$ ms

52 - GATE 2007 Question on page fault and dirty bit

A demand paging system takes 100 time units to service a page fault and 300 time units to replace a dirty page. Memory access time is 1 time unit. The probability of page fault is 'p'. In case of page fault the probability of page being dirty is also 'p'. It is observed that the average access time is 3 time units. Then value of p is - A) 0.194, B) 0.233, C) 0.514, D) 0.981

- $PS = (1-p)(100) + p(300) = 100 + 200p$
  (1-p: no dirty, p: dirty)
- $m = 1$ unit
- EMAT $= m + (p)(PS)$

$3 = 1 + p(100 + 200p)$

$3 = 1 + 100p + 200p^2 \Rightarrow 200p^2 + 100p - 2 = 0$

$$p = \frac{-100 \pm \sqrt{(100)^2 - 4(200)(-2)}}{2(200)} = 0.0194$$

53 - GATE 2003 Question on TLB and Paging

- A processor uses 2-level paging VA = PA = 32 bits. Byte addressable.

VA		
10	10	12

PTE = 4 Bytes. TLB has hit rate of 96%. Cache has hit rate of 90%. Main memory access time is 10ns, Cache access time is 1ns and TLB access time is 1 ns.

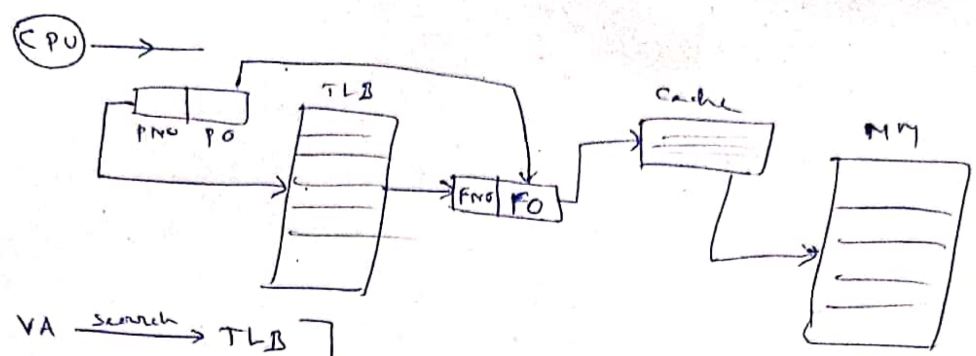
Q) Assuming no page faults, the average time taken to access a VA is approximately (to the nearest 0.5 ns)

a) 1.5 ns b) 2 ns c) 3 ns d) 4 ns

[TLB is also cache but TLB $\rightarrow$ Page Table entries & Cache $\rightarrow$ data from the process.]

i) Convert VA $\rightarrow$ PA

ii) Getting the word from MM using PA.



[VA —search→ TLB]
[PA —search→ Cache]
Not found search → MM

~~EMAT~~ = Average time taken to access VA

$= (VA \to PA) + (\text{Fetch the word from MM/process})$

- $VA \to PA$ ↓ Help of TLB
- Fetch the word ↓ Help of cache first

$= \left(t + (1-P_t)(k * m)\right) + \left(c + (1-P_c)(m)\right)$

- t → TLB hit
- $(1-P_t)(k*m)$ → TLB miss
- c → Cache hit
- $(1-P_c)(m)$ → Cache miss

$= (1ns + 0.04 * 2 * 10ns) + (1ns + 0.1 * 10ns)$

$= 3.8\,ns \to$ d) $4ns$

CPU → Cache → MM → SM

If Page fault is there → $t + (1-P_t)(k*m) + (c + (1-P_c) * (m + P_f(PS)))$

- P_f → Page fault rate
- PS → PF service time

TLB → PTB
Cache → Data of Process

Q) Suppose the process has only the following pages in VAS: Two contiguous code pages starting at virtual address 0x00000000, two contiguous data pages starting at VA 0x00400000, and a stack page at VA 0xFFFFF000. The amount of memory required for storing page tables of this process?

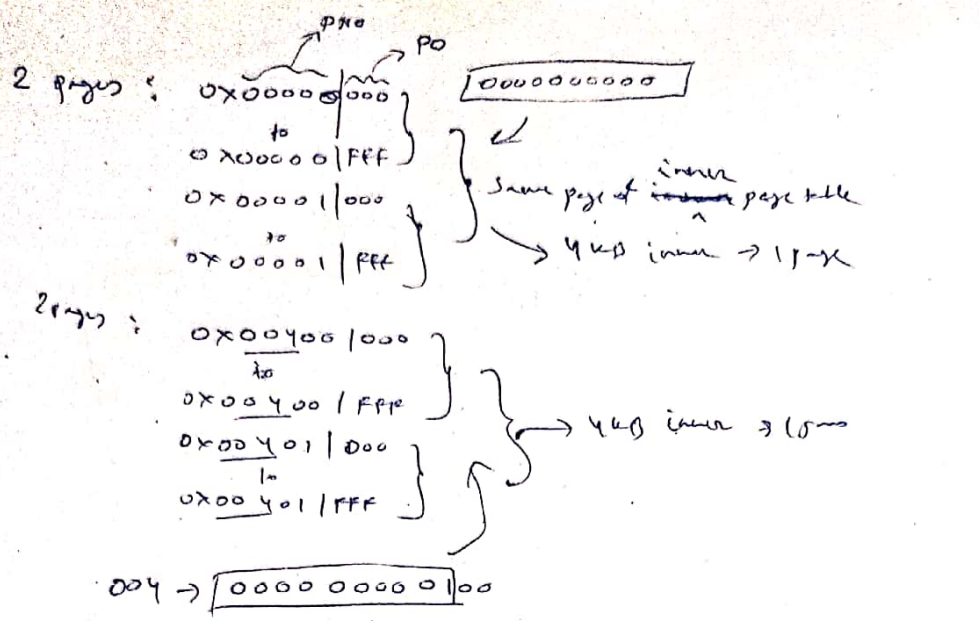
a) 8 KB, b) 12 KB, c) 16 KB, d) 20 KB

$PS = 2^{12} = 4\,KB$

No of Pages $= 2^{20}$ ⇒ $PTS = 2^{20} \times 4B = 2^{22} = 4\,MB$

Outermost PT $= 2^{10}$ entries

↳ size $= 2^{10} \times 4B = 2^{12}B = 4KB$



54 - GATE 2004 Question on TLB and Page Fault

Consider a system with a 2-level paging scheme in which a regular memory access takes 150 nsec and servicing a page fault takes 8 msec. An average instruction takes 100 nsec of CPU time and two memory accesses. The TLB hit rate is 90% and Page fault rate is one in every 10,000 instructions. What is the effective average instruction execution time?

a) 645 nsec, b) 1050 nsec, c) 1215 nsec, d) 1230 nsec

EAIT = CPU Time + 2 * EMAT

= 100 + 2 * EMAT

EMAT = (VA → PA) + (Access the byte from PA)

= $\left(t + (1-p_t)(2*m)\right) + \left(m + (1-p_f)(p_t + p_s)\right) + (p_f * p_s)$

= $0 + (0.1 * 2 * 150) + (150 + (\frac{1}{10^4}) \cdot (8 \times 10^{-3}))$

= 30 nsec + 150 nsec + $\frac{1}{10000} \times 10^{-3} \times 8$

= (30 + 150 + 800) nsec = 980 nsec

EAIT = 100 + 2 * 980

= 2060 nsec

- If no two memory access –

= 100 + 980 = 1080 nsec → B → 1056 nsec

55 – GATE 2014 Question on TLB

Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 msec to search the TLB and 80 msec to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in msec) is _____.

EMAT = $t + (1-P_t).(m) + m + (P_t * P_s)$

↳ Do not consider

= 10 ms + 0.4 * 80 ms + 80 ms

= 10 ms + 32 ms + 80 ms

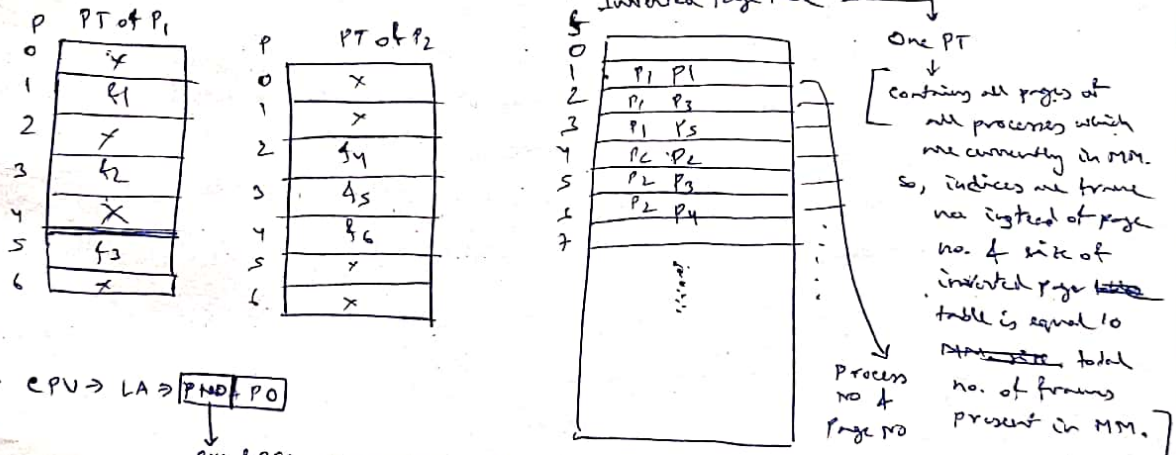
= 122 msec.

EMAT = $P_t(t+m) + (1-P_t)(t+m+m)$

= $P_t t + P_t \cdot m + t + 2m - P_t t - 2P_t m$

= $t + 2m - P_t \cdot m$ = 122 msec

56 – Inverted Page Table



- CPU → LA ⇒ | PNO | PO |

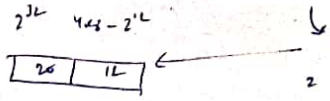
PNO & PID → search all entries in (IPT) → takes lot of time

- Not widely popular (IPT) → lot of time (Search)

↓ It will be faster ← If maintained in TLB

PA = 32 bits, PS = 4 KB, IPTE = 4 B

Size of IPT (IPTS) = ?



↳ No of frames in MM * PTE

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

57 - Importance of frame allocation and page replacement

Virtual Memory

[These two factors have impact on system performance.]

- Frame Allocation
  - Minimum requirement → defined by the architecture
    (Min req of pages - due to bad design i.e if the no. of page req is set to less num then Thrashing rate will be high)
  - Maximum requirement → defined by the process
  - choose a no between this
- Page Replacement

both approaches

- Equal Allocation → 30 frames & 3 process → 30/3 = 10 frames to each process
  (not good, if different processes has different no. of page requirements)
- Weighted Allocation → Frames are allocated to process according their weightage.
  $P_1 \Rightarrow 20$, $P_2 \Rightarrow 30$, $P_3 \Rightarrow 50$
  no of frames = 10
  $P_1 = \frac{20}{100} \times 10 = 2$ frames
  $P_2 = \frac{30}{100} \times 10 = 3$ frames
  $P_3 = \frac{50}{100} \times 10 = 5$ frames
- Dynamic Allocation

• These two methods are static

$P_i \Rightarrow S_i$

$S = \sum_{i=1}^{n} S_i$

"F"

$$P_i = \left(\frac{S_i}{S}\right) F$$

Depending on the priorities, frames will be allocated to processes. If a higher priority process wants more frames, then frames will be forcefully taken from lower priority processes & assign them.

• Page Replacement :-

i)

- Local
  - It only replaces pages of assigned frames to process.
  - Better than global i.e doesn't increase thrashing
- Global
  - It replaces pages from any frames of MM.
  - Thrashing ↑ Performance ↓

[A process whose page is already being thrashed is responsible for replacement of other process due to global access & total thrashing rate will be high.]

• Local Page Replacement Policies

(At first when MM is empty demand paging is used.)

[Reference string → CPU generates a sequence of addresses during execution of a process.] → 0 1 4 5 7 - - -
(used for need of sequence of pages)
Page no.

58 - Page Replacement Algorithm

- **Optimal** :→ Replace the page which will not be ~~refft~~ referred for longest. (not practical)
- **Least Recently Used (LRU)** :→ Replace the page which is least recently used or has not been referred for a long time.
- **FIFO** :→ First one has to go out first from MM.

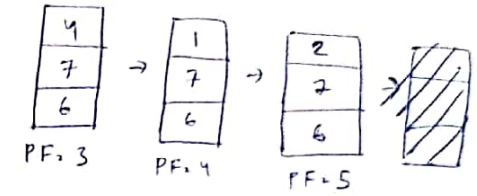
[Optimal is not practical & can't be implemented, but in theoritical aspect, when a new page replacement algorithm is ~~invented~~ Proposed, then that is measured against optimal as optimal gives least no of page faults.]

57 - Questions on Page Replacement algorithms

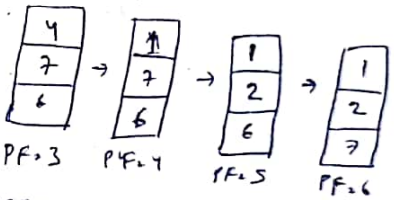
Find the Page Faults

14) 4, 7, 6, 1, 7, 6, 1, 2, 7, 2, F = 3 →

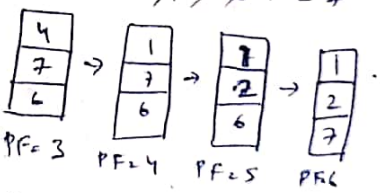
optimal



LRU



FIFO ~~4~~ ~~7~~ ~~6~~ 1 2 7



∴ Optimal = 5
LRU = 6
FIFO = 6

25) LO- - - - - - - - algorithm 2

Decimal
0100 ← 01 → Page No

01, 02, 04, 04, 05, 05, 05, 01, 02, 02, 02, 03, 03

F = 1 → optimal = LRU = FIFO

∀ PF = ~~3 for~~ 7 for all PR algorithms

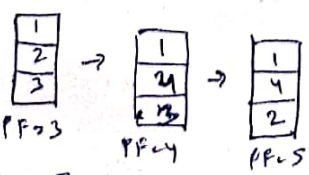
hit rate/PF rate = $\frac{7}{13}$, Hit rate = $\frac{6}{13}$

01 - - - - - - - - algorithm 3

09)

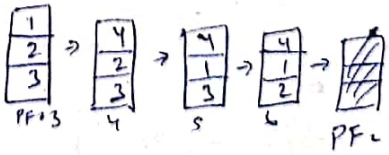
1 2 3 4 4 1 3 2 4 1 F = 3

optimal

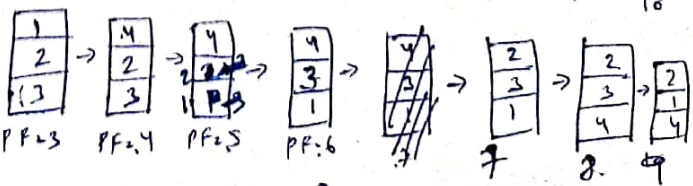


PFR = $\frac{5}{10}$, HR = $\frac{5}{10}$.

FIFO ~~1~~ ~~2~~ ~~3~~ 4 1 2



LRU



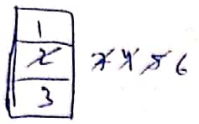
PF = 9, PFR = $\frac{9}{10}$

62- .. - - - - algorithms 4

Q7

1, 2, 1, 3, 7, 4, 5, 6, 3, 1 F = 3

Optimal



PF = 7

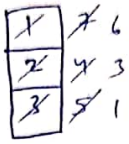
PFR = 7/10

LRU

PF = 9

PFR = 9/10

FIFO



PF = 9

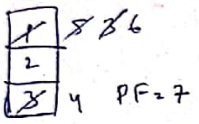
PFR = 9/10

63- - - - - - - - algorithms 5

17.

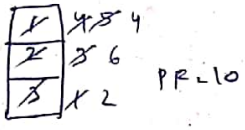
1 2 3 4 2 1 5 3 2 4 6 F = 3

Optimal



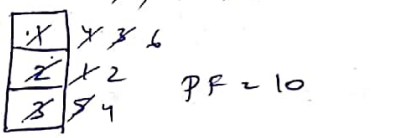
PF = 7

LRU



PF = 10

FIFO



PF = 10

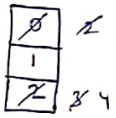
(PF $\xrightarrow{\text{min}}$ no of different pages)

L4- Belady's Anomaly

0 1 2 3 0 1 4 0 1 2 3 4

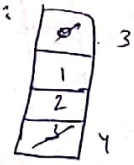
Optimal

F=3:



PF = 7

F=4:



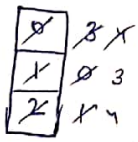
PF = 6

F = 3	F = 4
PF↑	PF↓

∴ F↑ PF↓

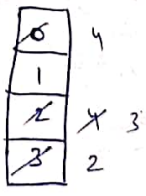
LRU

F=3:



PF = 10

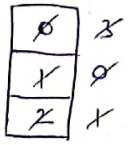
F=4:



PF = 8

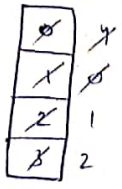
FIFO

F=3:



PF = 9

PF=4:

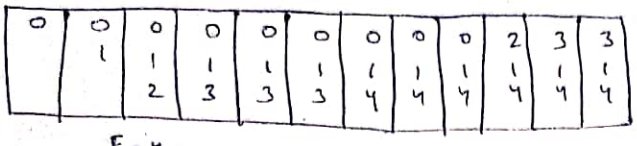


PF = 10

65 - Stack Algorithms

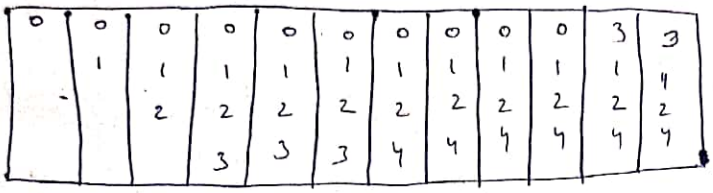
0 1 2 3 0 1 4 0 1 2 3 4

Optimal F=3



∴ Pages(3) ⊆ Pages(4)

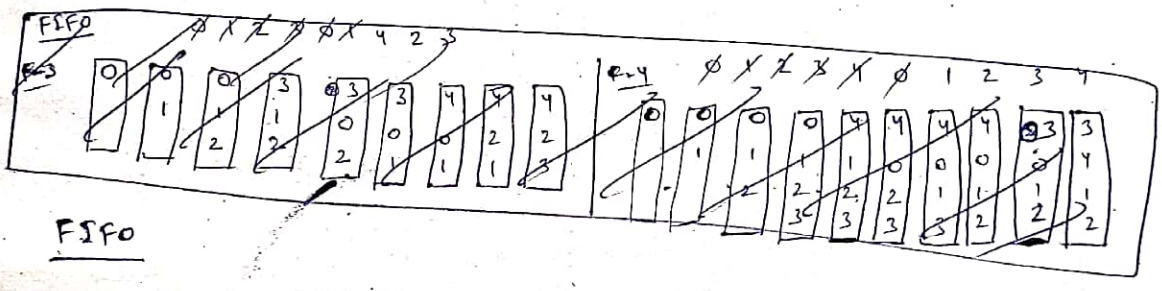
F=4



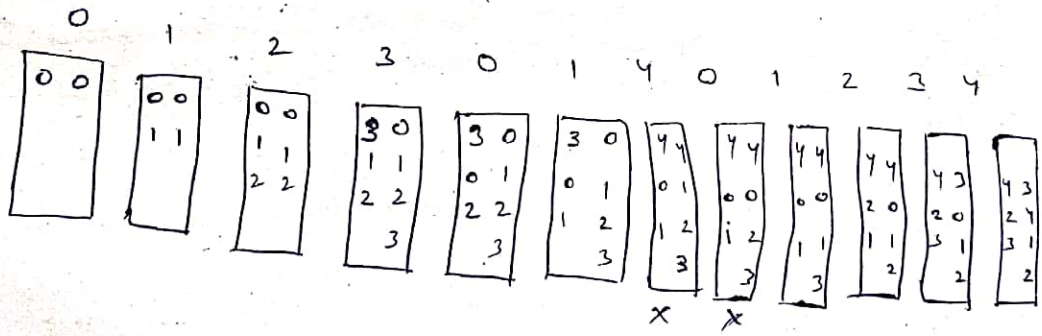
Stack Property :→ If there are some no. of pages in main memory with n frames & then is increased the ~~frames~~ no. of frames by 1 then the same pages that were present for n frames, must have present for (n+1) frames.

$$Pages(n) \subseteq Pages(n+1)$$

∴ Optimal PR is a stack algorithm so doesn't fall into belady's anomaly.



FIFO



LRV

Do it yourself → follows stack property

66 - TY Question on LRV LFU and FIFO

A memory page containing a heavily (frequently) used variable that was initialized very early and is in ~~cont~~ constant use is removed when _____ is used.

a) LRU
b) FIFO
c) LFU
d) None

Ans → B.

(* - LRV & its variations are widely used in most OS.)

C7 – GATE 2007 Question on FIFO

Consider the following statements:

P: Increasing the number of page frames sometimes increases page faults in FIFO.

Q: Some programs do not exhibit the locality of reference.

- P & Q are true, but Q is not valid reason for P.

C8 – GATE 2010 question on FIFO

A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur?

a) 196, b) 192, c) 197, d) 195

F = 4 → 100 distinct → PF = 100

↳ same 100 in reverse order SC = 100 − 4 = 96

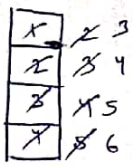
↳ 100 + 96 = 196 → Ans → A

C9 – Some interesting behaviour of optimal page replacement algorithm

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

F = 4

Optimal →



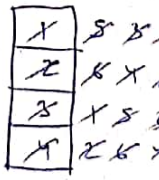
PF = 12

→ When a loop is present in the code, it is beneficial when optimal PRT is used & it is achieved by replacing most recently used pages in a consecutive manner (MRU).

MRU →

PF = 12

LRU (Worst Case) ↳ LRU = FIFO

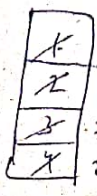


PF = 24

1 2 3 4 5 6 7 8 9 9 8 7 6 5 4 3 2 1

Optimal

Optimal = LRU = FIFO



PF = 14

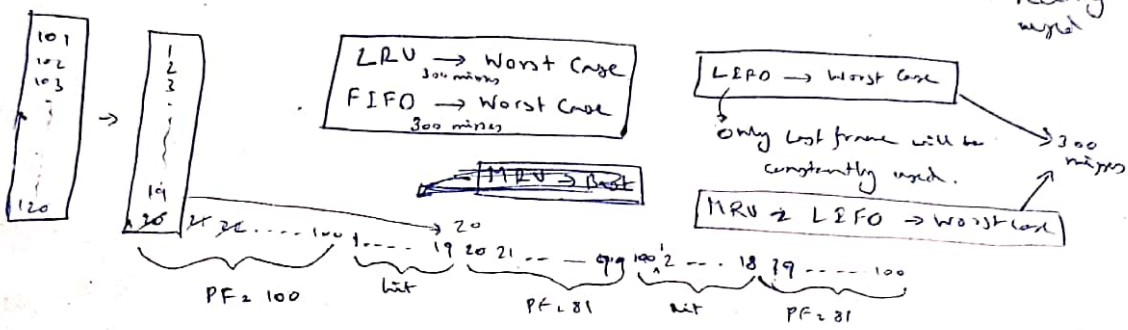
MRU → Worst case → only 1 hit

Repetation = Worst for LRU

70 - GATE 2014 Question on LRU, MRU, LIFO, FIFO and Optimal

A computer has 20 physical page frames which contain pages numbered 101 through 120. Now a program accesses the pages numbered 1, 2, ... 100 in that order and repeats the access sequence thrice. Which one of the following page replacement policies experience the same number of page faults as optimal page replacement policy for this program.

a) Least Recently Used, b) First in First out, c) Last in First out, d) most recently used



∴ Total PF = 100 + 81 + 81 = 262

Total Page Hit = 19 + 19 = 38

∵ Question → initially all frames empty → Demand paging is used

∵ LIFO = MRU = Best Case → PF = 262
↳ Hit = 38

71 - Working Set Algorithm

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

Working Set Algorithm :→ The nearest future is the close approximation of the recent past.

Window size → A = 4

Working set → $w = \{1\}$ A=1, $w = \{1,2\}$ A=2, $w = \{1,2,3\}$ A=3, $w = \{1,2,3\}$ A=4, $w = \{2,2,3\}$ A=4, $w = \{1,2,3,4\}$ A=4

$w = \{1,2,4\}$ A=4, $w = \{1,2,4\}$ A=4, $w = \{1,2,4\}$ A=4, $w = \{1,2,4\}$ A=4, $w = \{1,2,4,5\}$ A=4

$w = \{1,2,5\}$ A=4, $w = \{1,2,5\}$ A=4, $w = \{1,2,4,5\}$ A=4, $w = \{1,5,4\}$ A=4, $w = \{1,2,4\}$ A=4

- Whenever there is a PF, or a process is blocked & next time when it is going to be loaded, then we will not use demand paging. We will load the process with the working set, so that PF will be less.

Avg Frame requirement = $\frac{\text{Sum of all frame size}}{\text{total no}}$

- This algorithm is used in dynamic allocation & deallocation.
- Another variation is PF decreasing/limiting algorithm.

22

(add) c c d b c e c e a d

Victim → Loosing frames
Gainer → Adding frames

w = {c, d, e} | w = {c, d, e, c} w = {c, b, e, c}
A = 4 | w = {c, d, e, c}, w = {d, e, c}
| w = {b, c, d}, w = {b, c, d}
| w = {b, c, d, e}, w = {b, c, e}
| w = {c, e}, w = {a, c, e}, w = {a, c, d, e}

∴ Avg Fr req
= (4+3+3+3+3+3 +4+3+3+2+3+4) / 10

• 72 – Page Replacement Algorithms Implementation 1

• Optimal :→ It cannot be implemented practically.

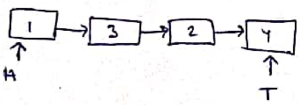
• Not Recently Used :→ Find page which is not recently used & replace it. Referred bit is used along with clock cycle to know which page is referred or not & replaced. Modified bit can also be used to notify page modification.

R	M	
0	0	→ no ref & no mod
0	1	→ not ref, but modified
1	0	→ referred
1	1	→ referred & modified

not ref in last clock cycles, referred in previous clock cycles
priority or ordering of replacement

73 – Page Replacement Algorithms Implementation 2

• FIFO :→ It is implemented as Queue using Linked List.



• Second Chance :→ Almost FIFO but R bit (Refer bit) is used whenever there is a need of replacement. It is better than FIFO. Whenever a page is known for no replacement, it brings the page to the end of the list & add it. This is unnecessary & main disadvantage.

• Clock Page Replacement :- Variation of / modification of second chance. It is maintained as a circular singly linked list. Only pointer movement is enough if R = 1 (bit). It runs in a clockwise manner.

74 – Page Replacement Algorithms Implementation 3

• LRU :→ Instead of one bit, multiple bits are used. Initially all bits are zero for all pages. Whenever a page has been referred in one clock then the MSB bit is set to 1 & after every one clock these bits are shifted to right by one. These bits specify the reference bits for that no. of clock cycles. So if all one 1's then the page is referred mostly in last few clock cycles. The more no. of bits = the more information can be stored. Less no. of 1's = Least recently used page.

[Most implementation use 64 bits to represent these]

• Diff b/w LRU & Not Recently used is no. of bits.

Scanned with CamScanner

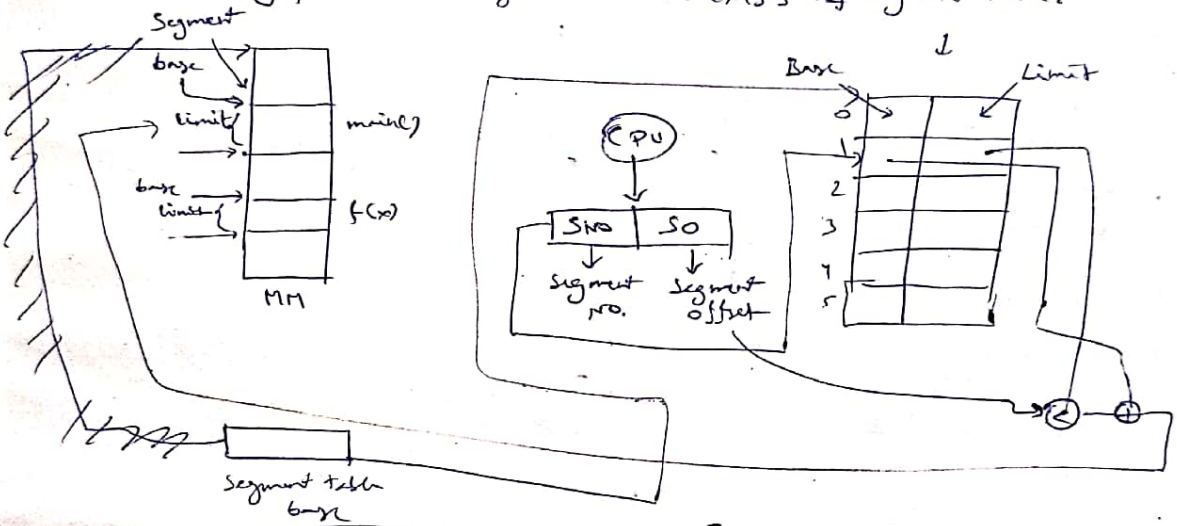
- **LFU** :- With every page a counter is associated & whenever a page is referred then its counter is incremented.

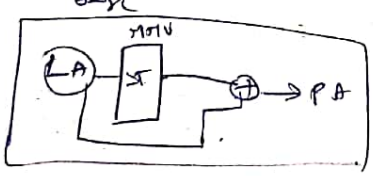
  Higher Value → MFU

  Lower Value → LFU

75 - Segmentation

- Dividing the program or process into pages might sometime lead to division which might not be meaningful. The paging boundary might ~~divide~~ try to divide the process in such a way that some useful information might fall in different ~~pla~~ places, which are supposed to be together.
- To be meaningful for the users, process is divided into segments. Each segment might hold a function or data or stack part. These segments are loaded into main memory as a ~~whole~~ whole. Segments are not further divided into many parts. Like Page Table there exists ~~eg~~ Segment Table.



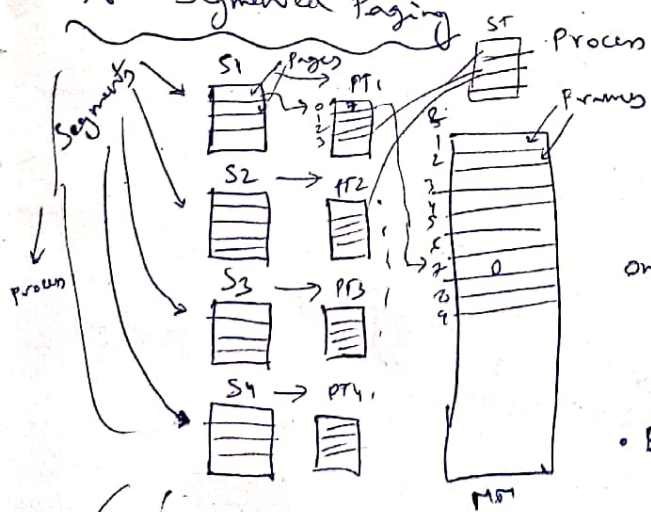


[In some OS, Segmentation is combined with paging]

Segmentation → User view

Paging → Implementation point of view

76 - Segmented Paging



↳ Page = Frame

[Segments → user point of view
Paging → Implementation point of view]

one PT for each segment

No of Pages in one segment = no of entries in its corresponding PT

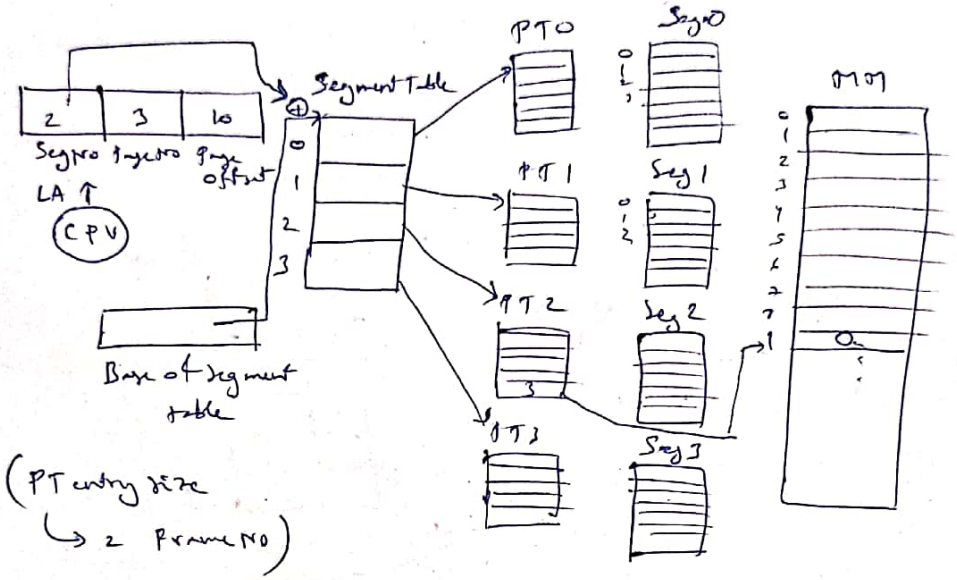
- Entry of PT containing frame no (FNO) in which corresponding page of the segment is present in MM.

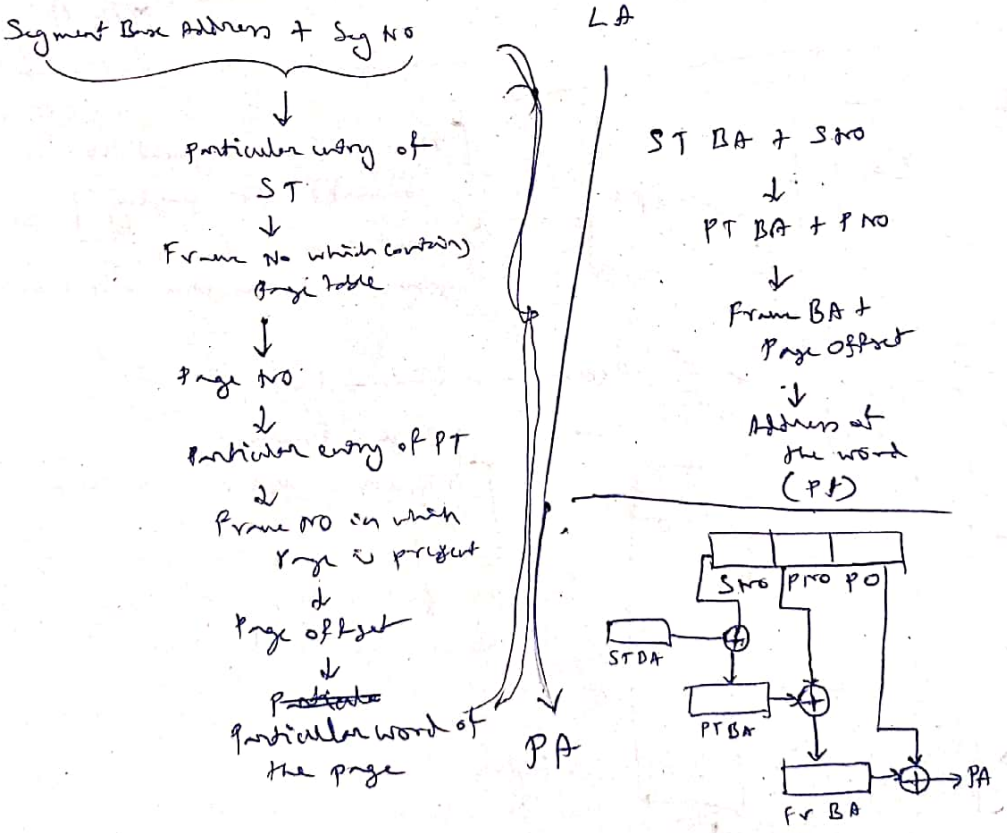
(↳ S1[0] → PT1[0] = 7 → MM[7, 0])

↳ One PT of each segment must fit in one frame each.

↳ 4 ~~Pages~~ PT → 4 frames

- To maintain the list of all the frame numbers in which the page tables are loaded, one more table is used i.e called Segment table. For entire process there is only one segment table or ST.
- ST contain pointing to the Page Tables.
  (Frame No)
- ST [Frame No] → PT → PT [Frame No] → Page



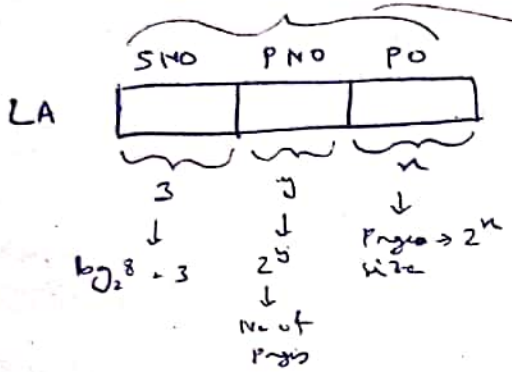


77 – GATE 99 Question on Segmented Paging

- PAS = LAS = 2^{16} B Byte addressable

VAS is divided into 8 non-overlapping equal sized segments. The MMU has a segment table, each entry of which contains BA of the PT for the segment. PT are stored in the MM and consist of 2 byte PTE.

a) What is the minimum page size (in bytes) so that the page table for a segment requires at most one page to store it. Assume that page size is a power of '2'.



$\therefore 3 + n + y = 16$

$\Rightarrow n + y = 13$

∵ PTS = No of entries * size of PTE

$\Rightarrow 2^n = 2^y * 2$

$\Rightarrow 2^n = 2^{y+1}$

$\therefore n = y+1$

∵ $n + y = 13$

$\Rightarrow y + 1 + y = 13 \Rightarrow 2y = 12 \Rightarrow y = 6 \quad \therefore n = 7$

∴ Page size = $2^n = 2^7 = 128$ B