Memory Managment

OS → pre-loaded

Resident Memory

a particular part of memory is dedicated to OS. (operating system)

OS → installed fence address

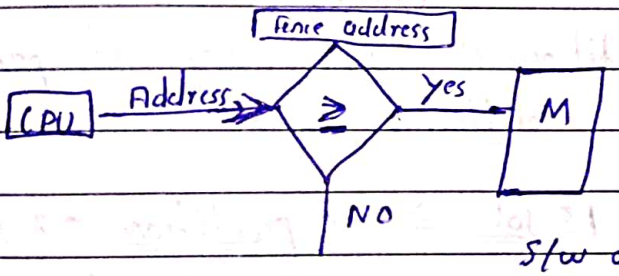
CPU → Address →

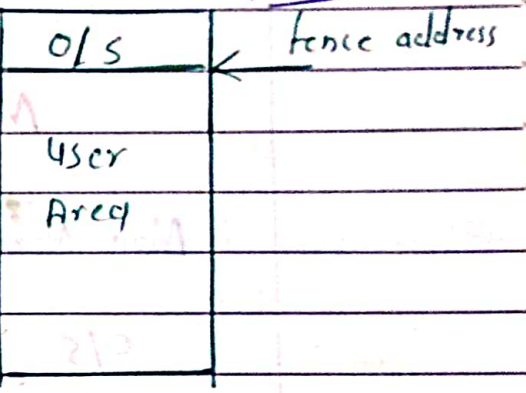
H/w based → Fence Register

Address → CPU

↓

TRAP



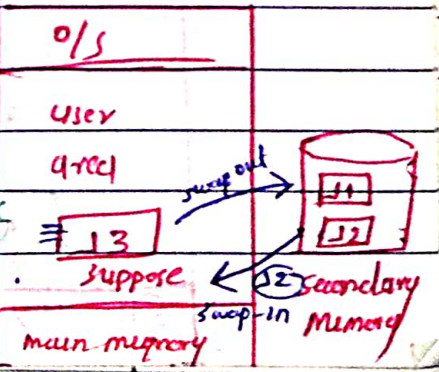


S/w or H/w ?

H/w → speed-up (fast)

S/w → slow-down →

Single-User



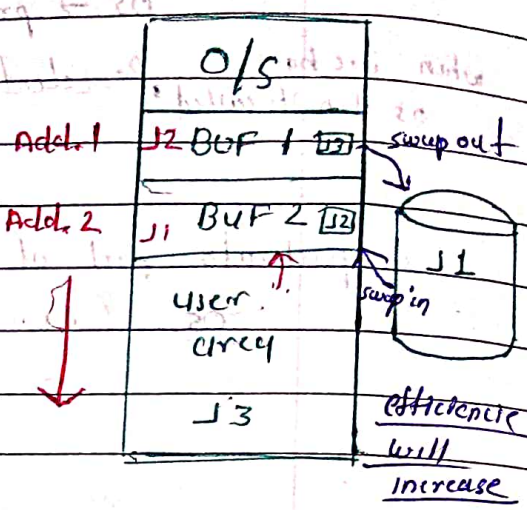
efficiency of CPU decrease

its own time

swap-out, swap-in → time taking

Moth User

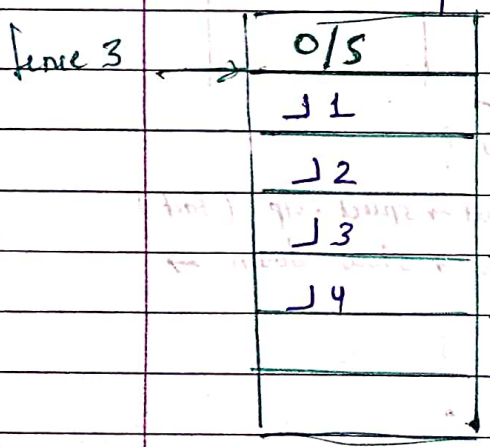
Date: _____ Page no: _____



- swap-in & swap-out done by address J3 & J1

Multiprogramming with fixed No. of Tasks (MFT)

Main Memory partitioned a number of parts different size



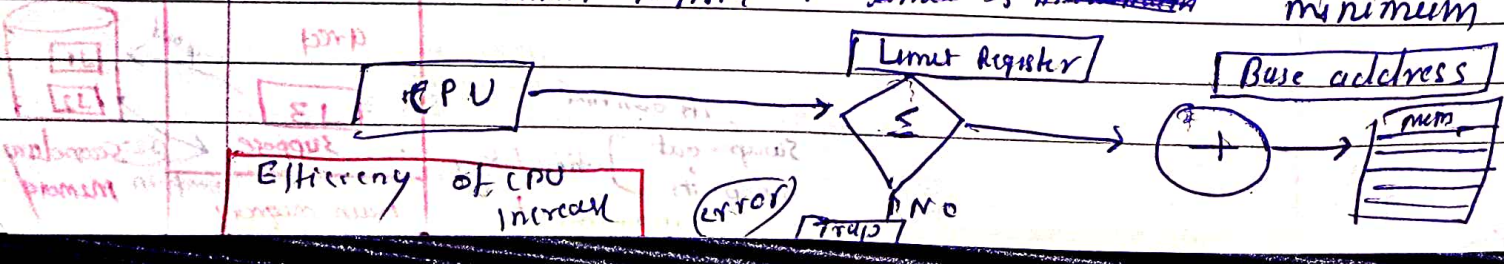
15 jobs ⇒ 15 partition → executed Simultaneously

lower bound Register {1} –

upper bound Register {10} –

1 - base Register → ~~minimum~~ maximum

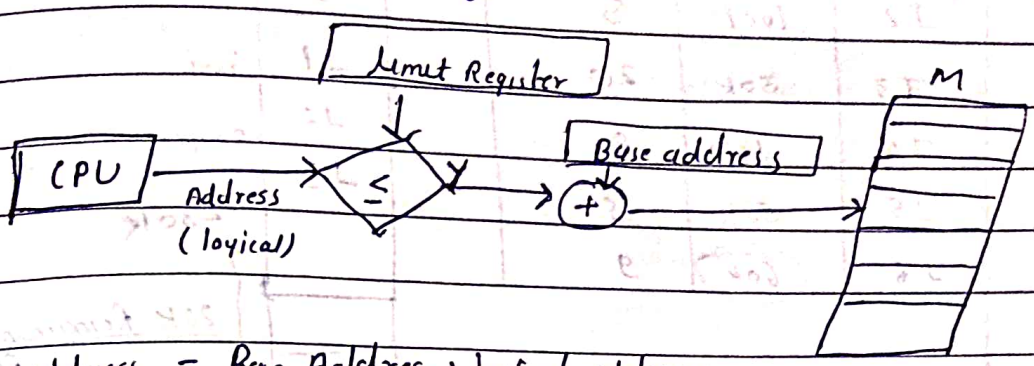
2 - limit Register → ~~limit~~ & ~~maximum~~ minimum



Date: Page no:

Lec-2

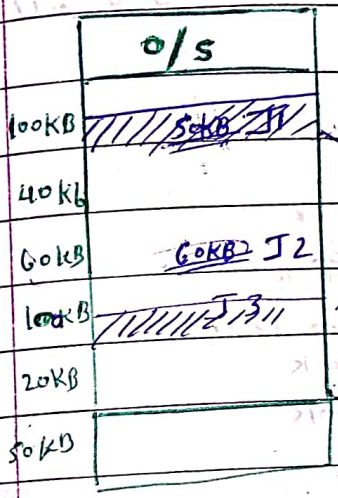
MFT ⇒ Multiprogramming with fix Number of Tasks



physical address = Base Address + logical address

J1 J2 J3

JOB - 50KB, 60KB, 80KB



different size

First fit Algorithm [FFA]

100KB → 50KB

[INTERNAL FRAGMENTATION]

3 50KB

20KB [INTERNAL FRAGMENTATION]

120KB External fragmentation

if Job ⇒ 120KB and I have max partition is 100KB so how can I alot

Best fit Algorithm

↓

complexity

High

Multiprograming with Variable number of Tasks [MVT] ⇒

MVT ⇒ We don't have any predetinceled partition rather the partition are created at the appropriate size when required.

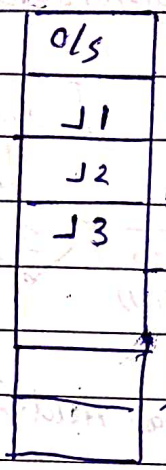
O/S

-40KB 256 KB - 40K for O/S

216K for you

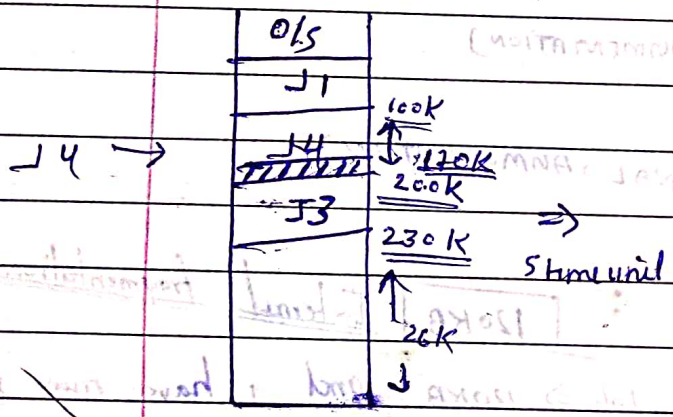
Date: Page no:

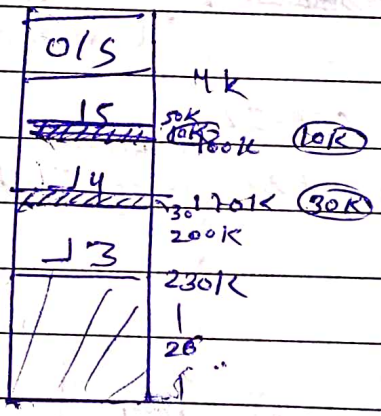
Job	size	Time
J1	60K	10
J2	100K	5
J3	30K	20
J4	70K	8
J5	50K	15
J6	60K	9



So we can't execute any another Job all are waiting

after 5 ~~minute~~ Time unit



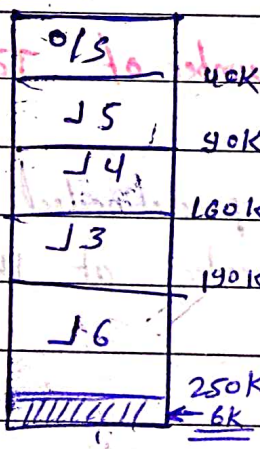


10K + 30K + 26K = 66K

but we can't execute J6

for that we use technology

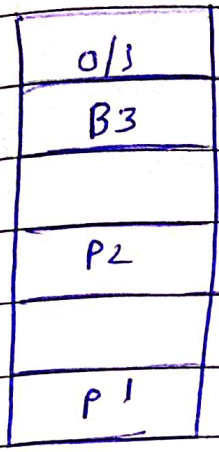
Compaction → Compact



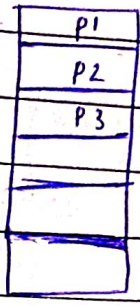
(we can't use it because its size is very less)

Date: _____ Page no: _____

Paging



Job - 30K (devide the job also equal size)



⇒ process in devide in equal size is called page.

Page

Topic for mid

⇒ CPU, Number system, Memory managment

mcq - 30 marks

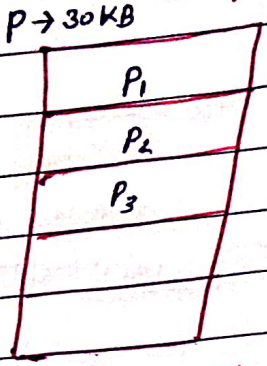
9 - MCQ

21 - discriptive

paging

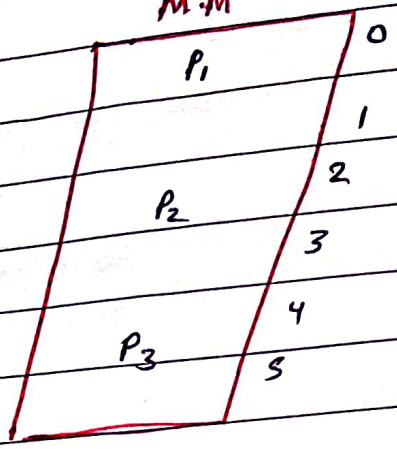
process / job

P → 30 KB



page (1KB)

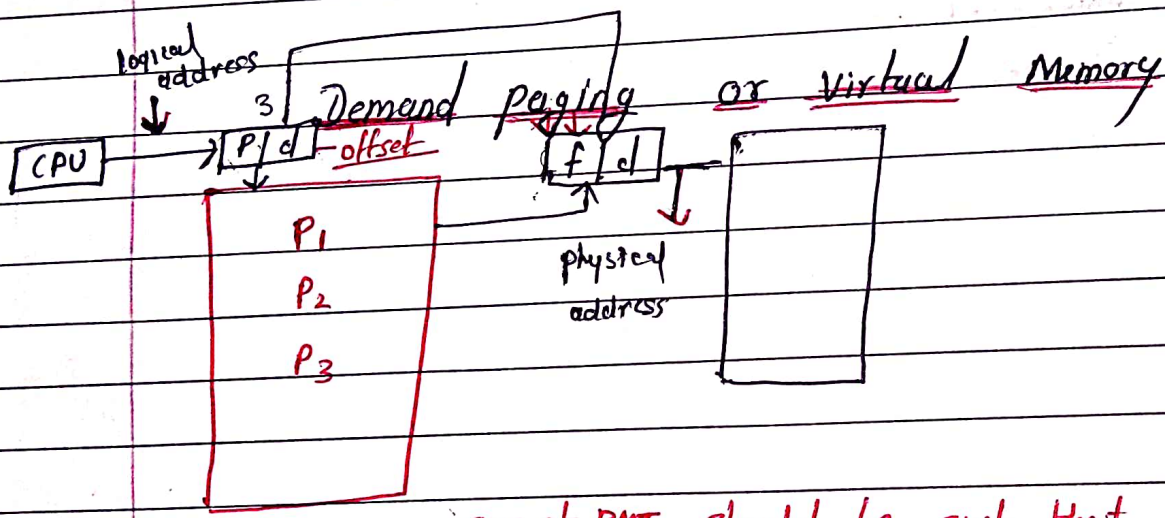
M.M



Ram (1KB)

size of pages and frames are always same

Demand paging or Virtual Memory



PMT

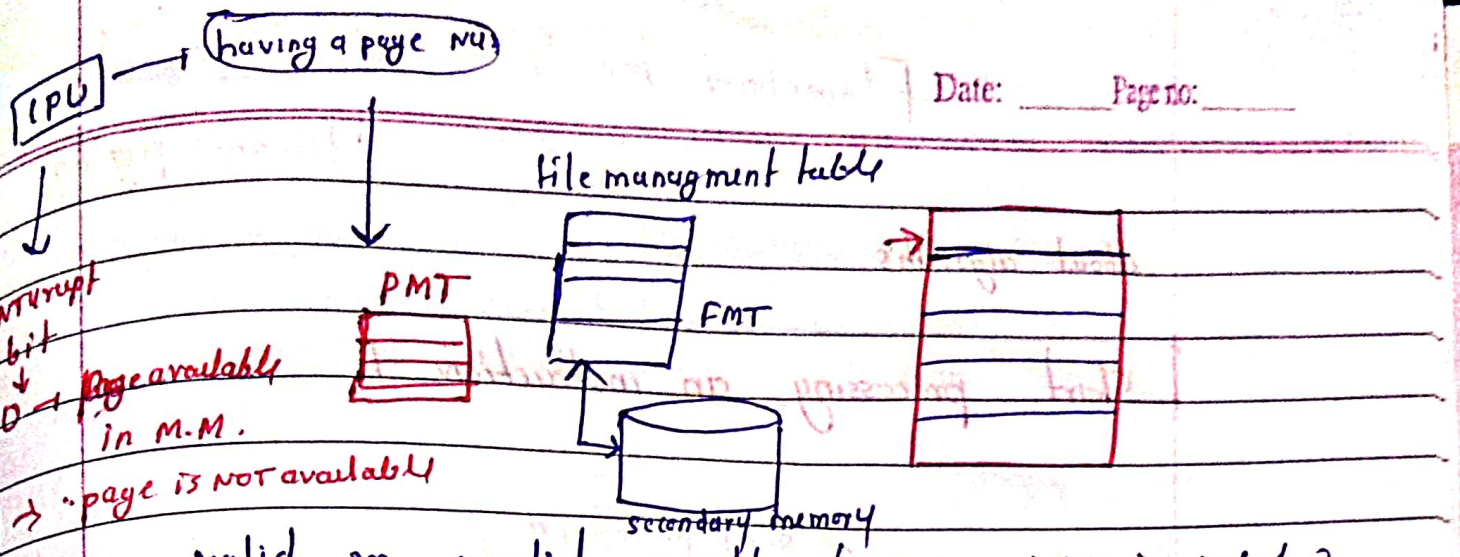
↓

page managment Table

size of PMT should be such that accomodate all the pages.

paging ⇒ all the pages accomodate in PMT.

Demand paging ⇒ we will not put all the pages in main memory.



valid or invalid results (page available is valid)

⇒ page is not available

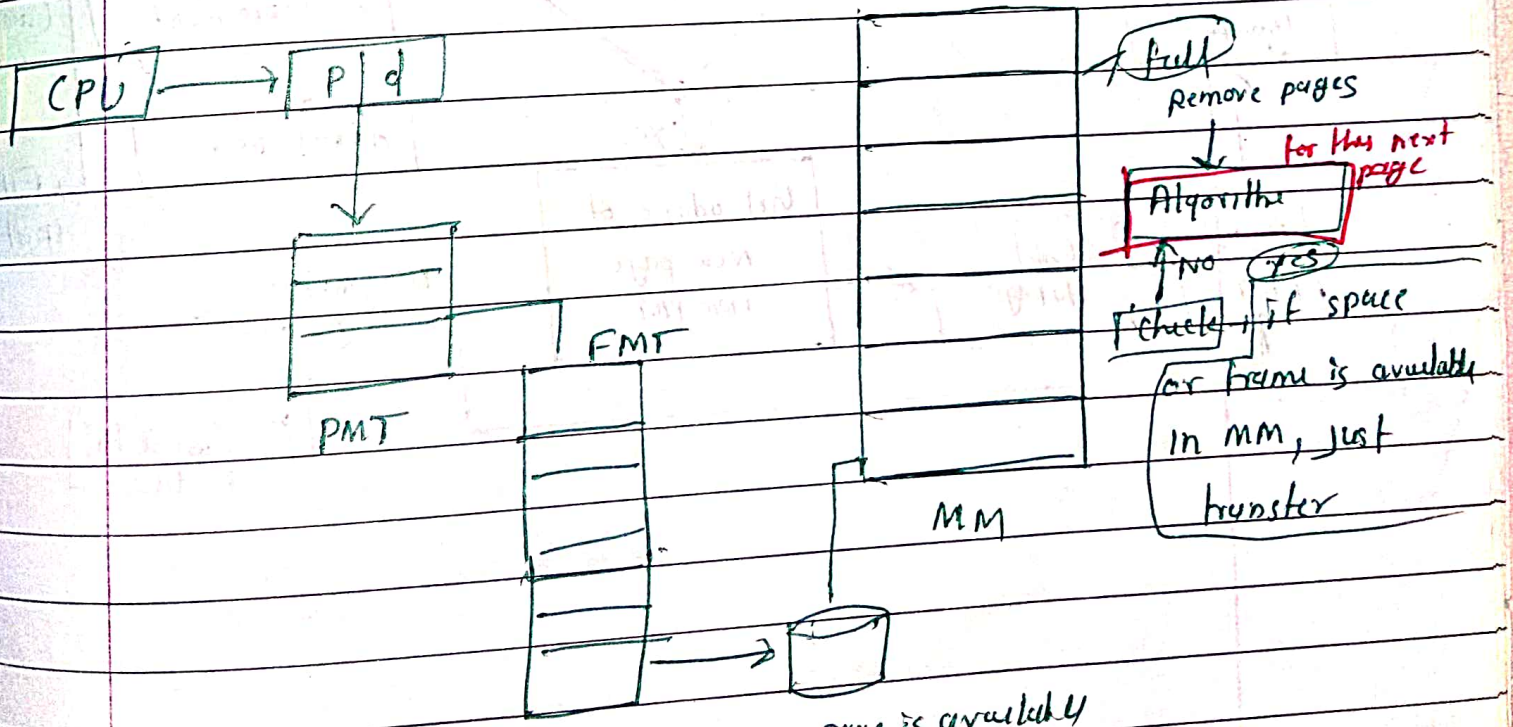
↓

interupt bit - 1

page fault

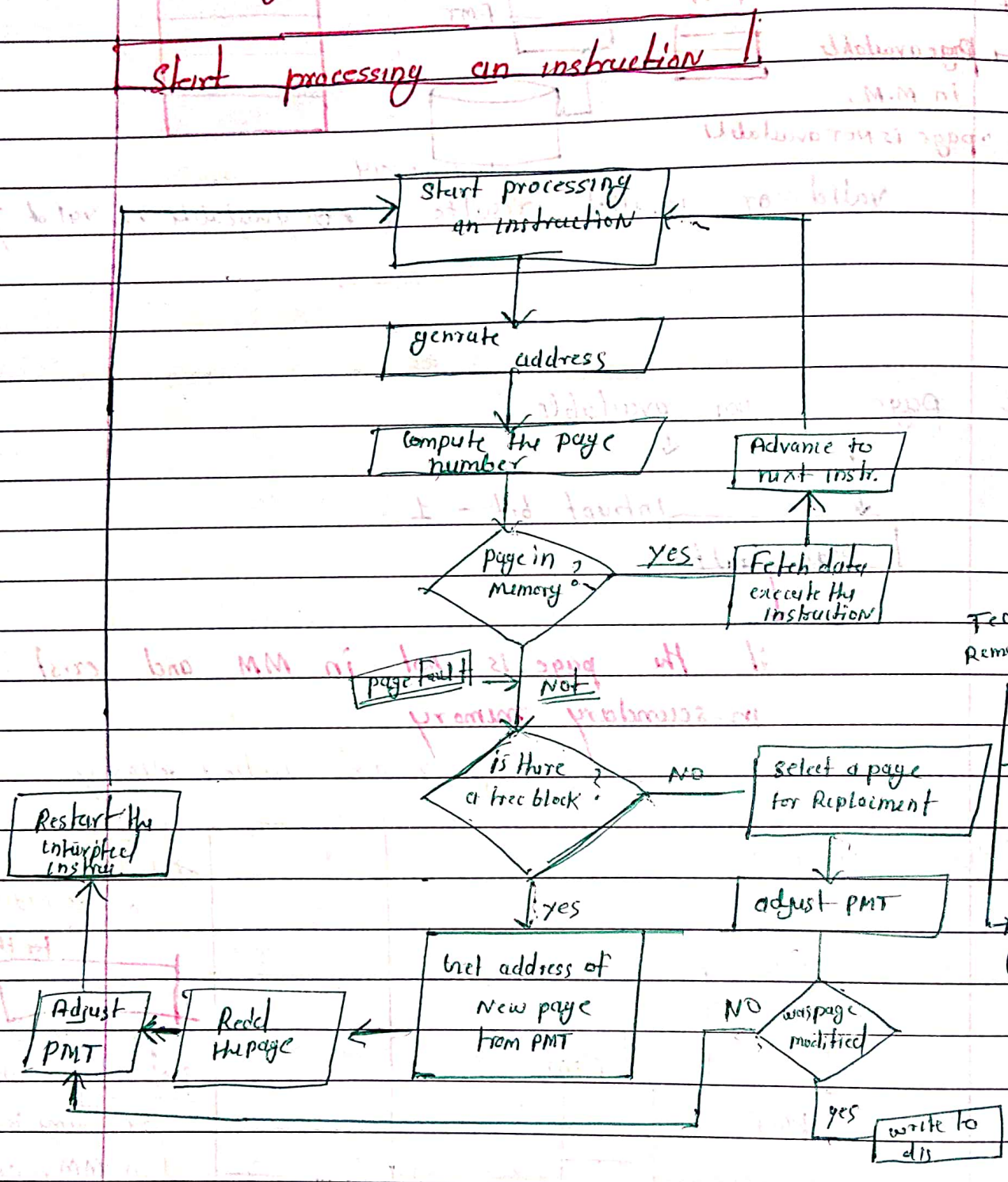
if the page is not in MM and exist in secondary memory

Demand paging or virtual memory



Flowchart for Virtual Memory or demand paging

about algorithm's



page Replacment Algorithm

Date: _____ Page no: _____

page Replacment Algorithms →

1. optimal
2. LRU
3. FIFO

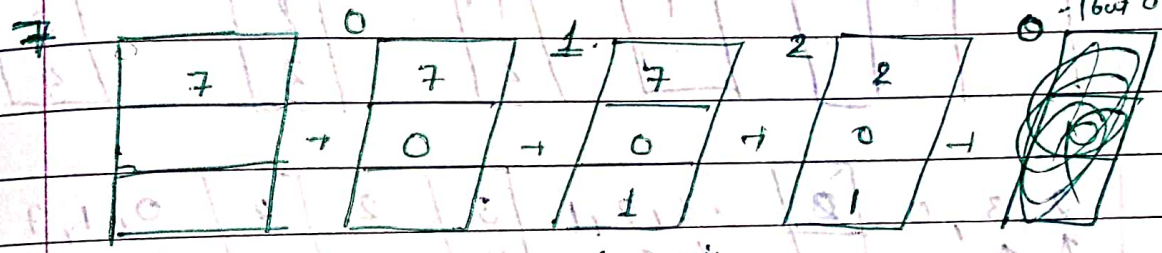
FIFO (First in first out)

pages

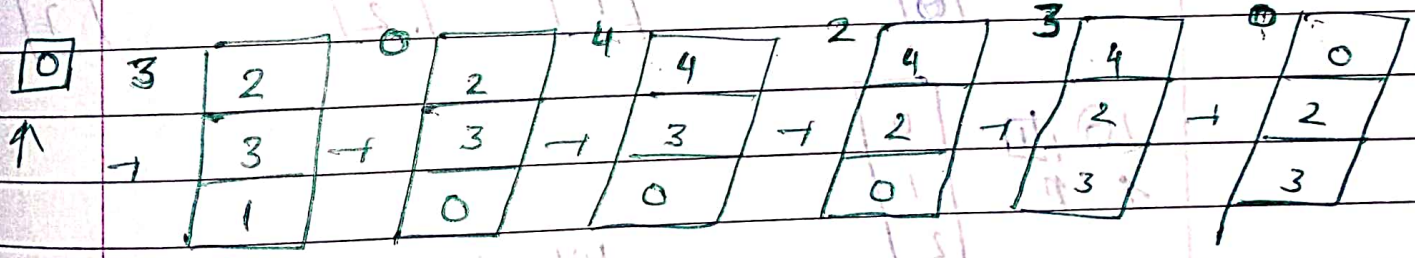
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

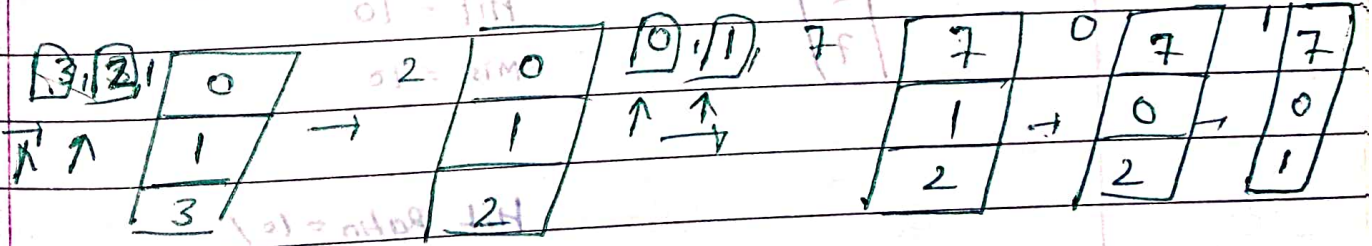
Block size - 3 [considered free]

first in first out so 2 replace 7



memory is full and need to Replace





Hit = 5

Miss = 15

20 pages

~~Hit Ratio = 5/20~~

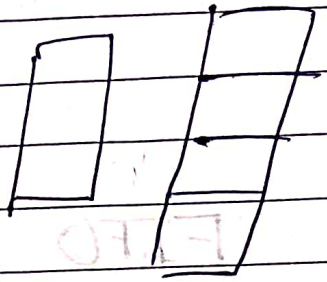
Miss Ratio = 15/20

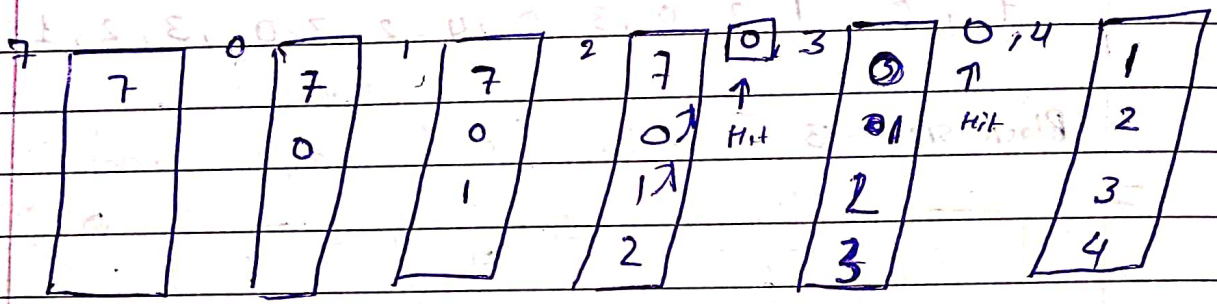
Date: Page no:

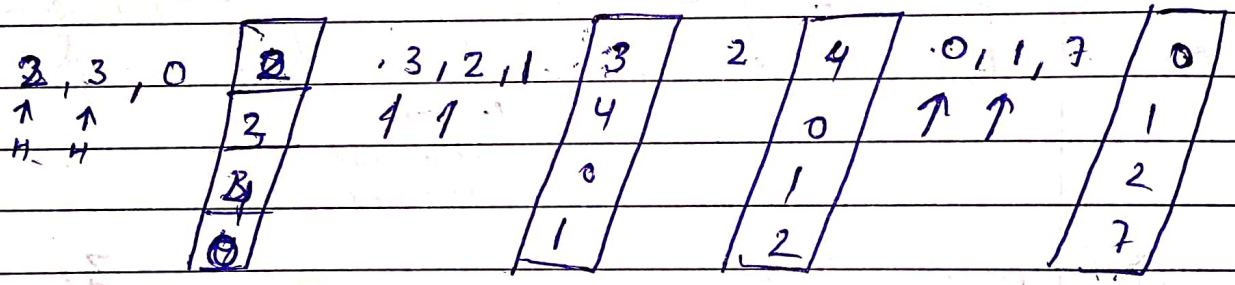
Ex 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 2, 1, 2, 0, 1, 7, 0, 1

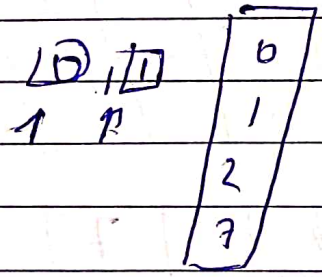
Page Replacement algorithm

→ FIFO
→ frame = 3 (free)
→ frames = 4 (free)









Hit = 10

Miss = 10

Hit Ratio = 10/20

Mis Ratio = 10/20

FIFO - Belady's Anomaly - problems in FIFO Technique

Date: Page no:

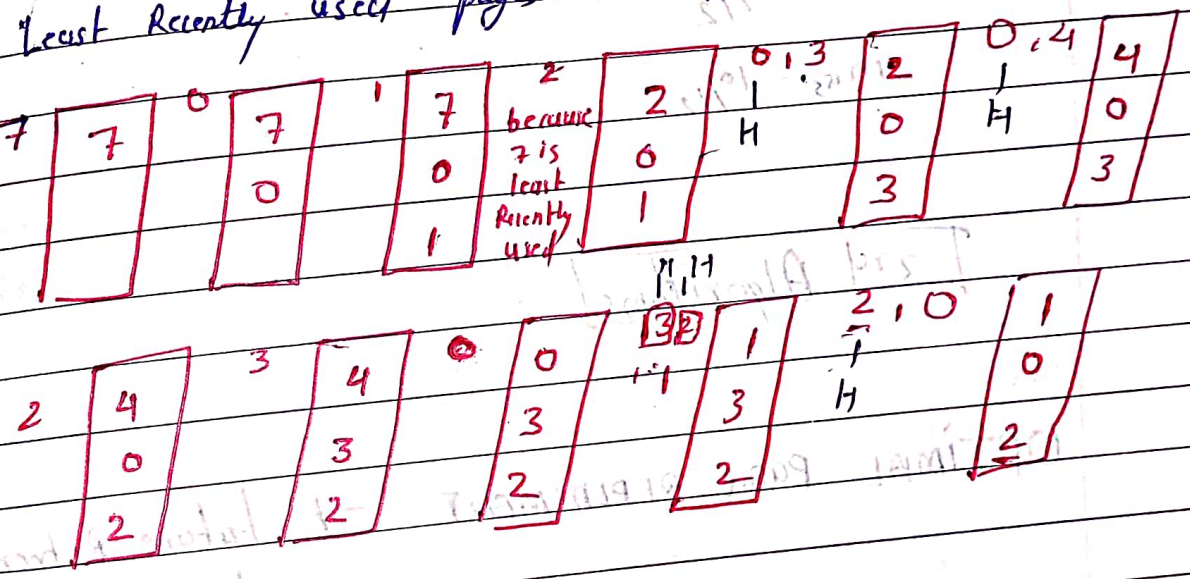
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 find Hit & Miss Ratio by frame 3 and 4

2nd Algorithm

Least Recently Used [LRU]

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Least Recently used pages for Replacment (3)



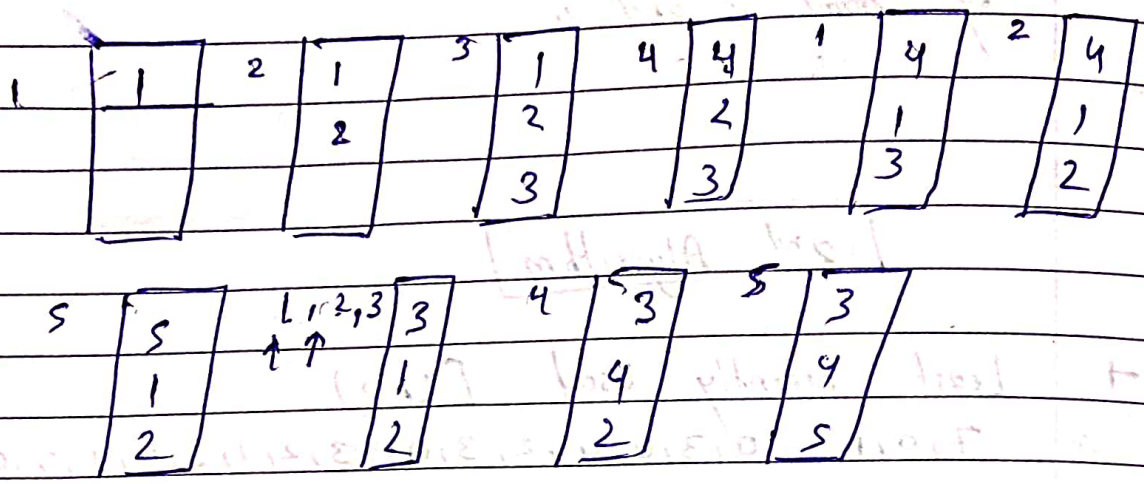
Hit = 8/20

Miss = 12/20

Ashish patidar | So optimal is best

Date: Page no:

=> 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



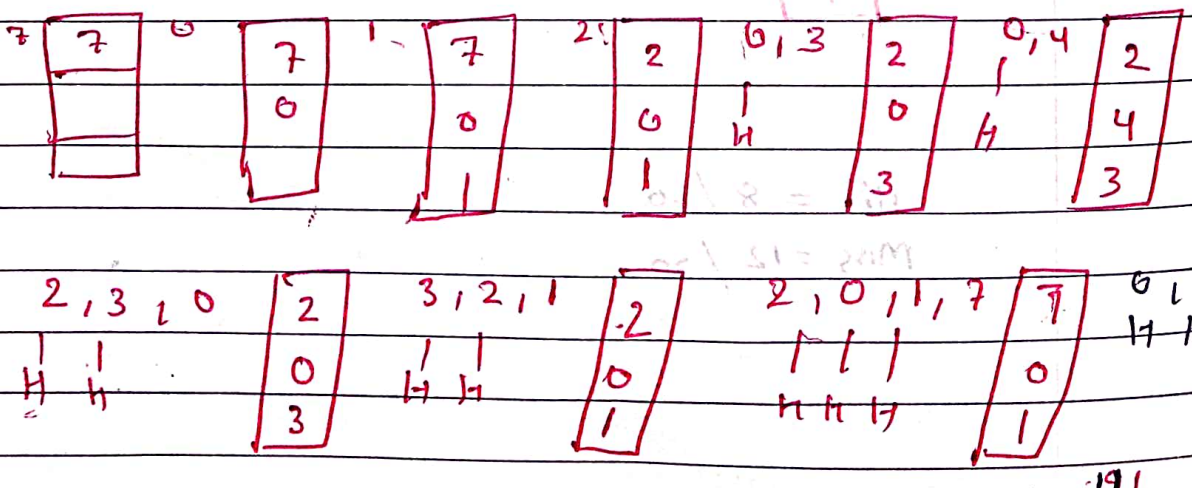
Hit - 2/12

Miss - 10/12

3rd Algorithms

OPTIMAL PAGE REPLACMENT → future Refrence for Replacment

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



H = 12/20

M = 8/20

Date: _____ Page no: _____

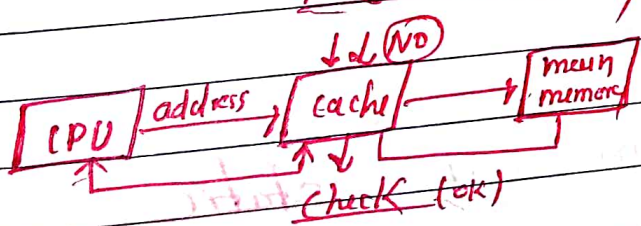
Memory managment

main memory
- → Buffer
- → Fixed number of Tasks
- → Variable number of Tasks
- → pagging
- → symentation [No. of functions]
- → Demand pagging [virtual memory]
- → Replacment Algorithms [LRU, FIFO, optimal]

CPU → speed of CPU very high.

Fastest — cache → small
Fast — main memory → pagging (similar)
Slow — secondary memory → Large in com. of cache
→ very large memory

Cache Memory



Cache Architecture ⇒

1. Associative cache
2. direct Mapping cache
3. set - Associative cache

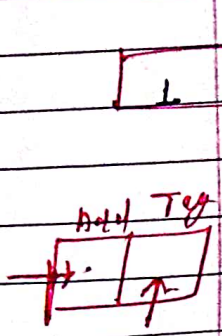
3- Hardware is required

Direct → H/w less → lose the flexibility

Associative → H/w is more → flexibility

Date: Page no:

1. Associative Cache =>



Address line - 16 bit

16 bit maximum memory $= 2^{16} = 64KB$

64 KB – MM

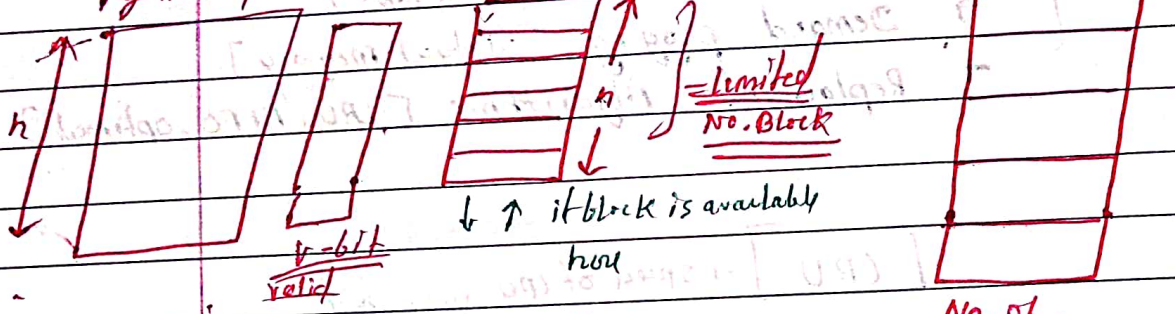
block ⇒ 8 byte size

⇒ 8K number

⇒ 16 – size.

block size

Size – No. of block



for each block, we have Tag memory

valid bit = 0 or 1

invalid valid

paging → main memory

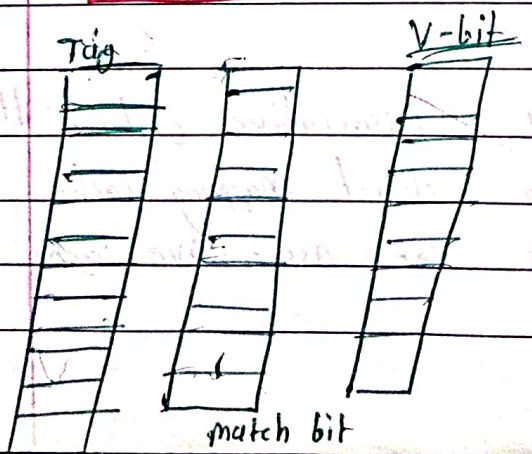
cache

CPU → address

Hardware implementation

Associative cache static

Arg. Reg



Date: Page no.

Adj. Reg

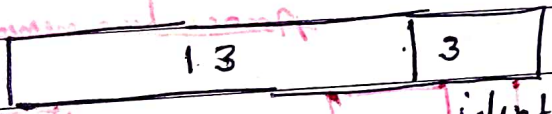
$\Rightarrow 2^{16} = 64K$

$\Rightarrow 2 \times 2 \times \ldots \Rightarrow 65536 / 1024$

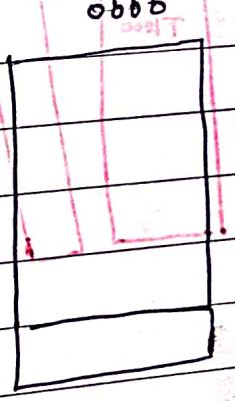
$1K = 1024$

$\Rightarrow 64K$

Direct Map



identification



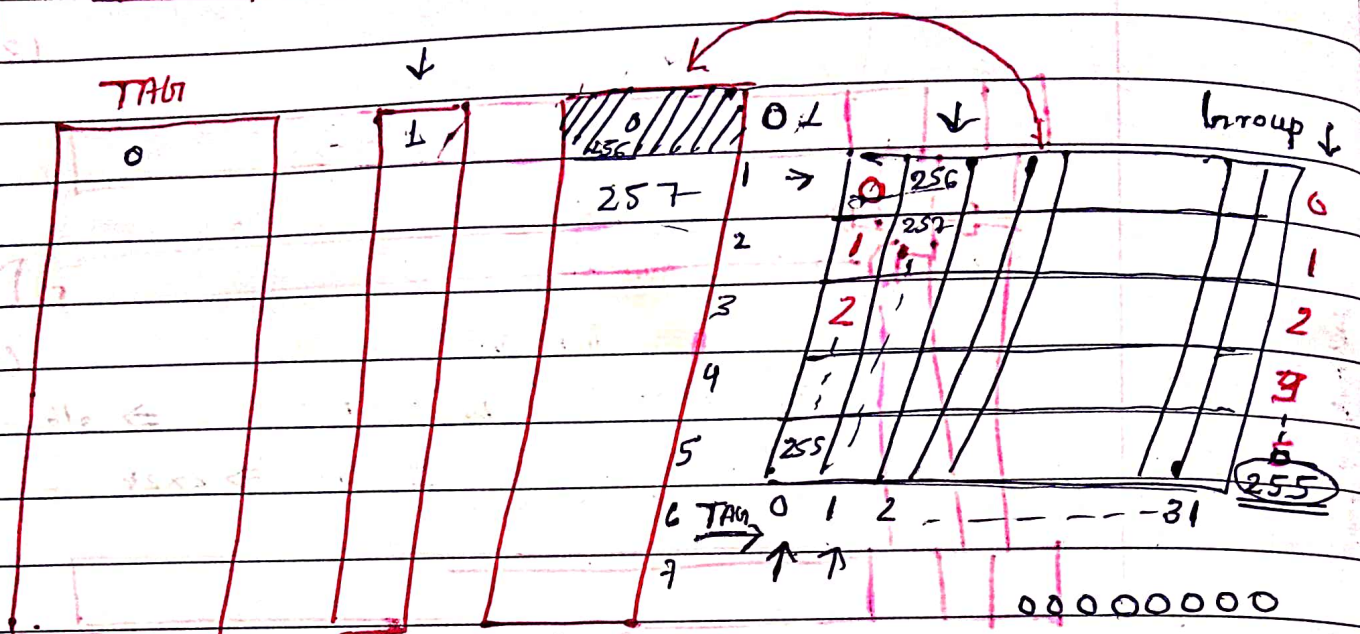
Date: Page no.

Direct Map

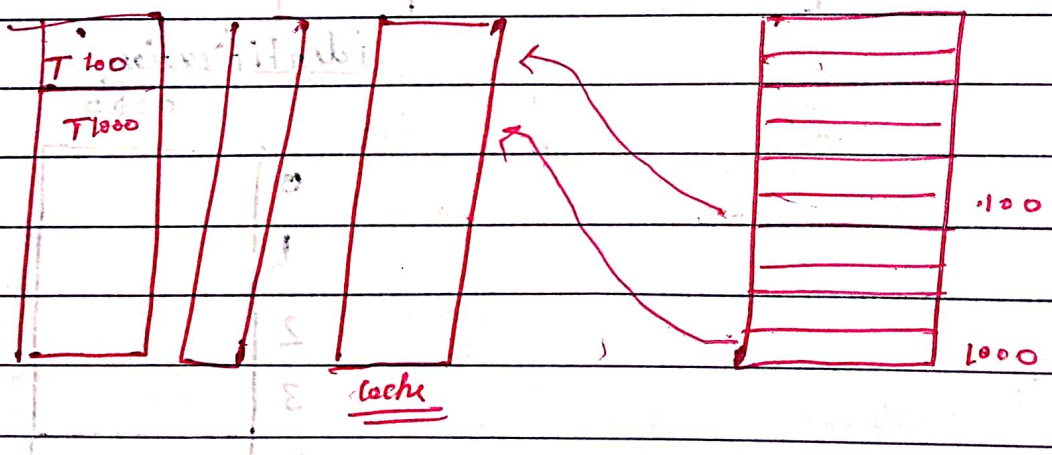
Associative

Tag	Byte

Tag	Group	Byte
5	8	3

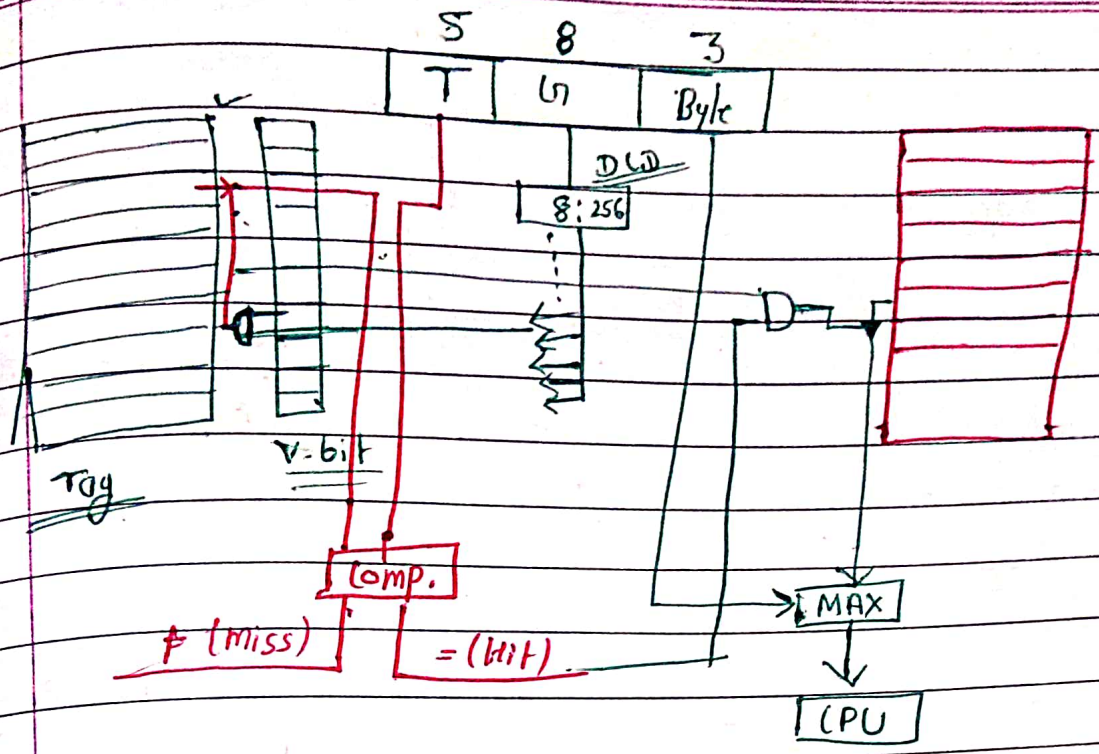


Associative memory

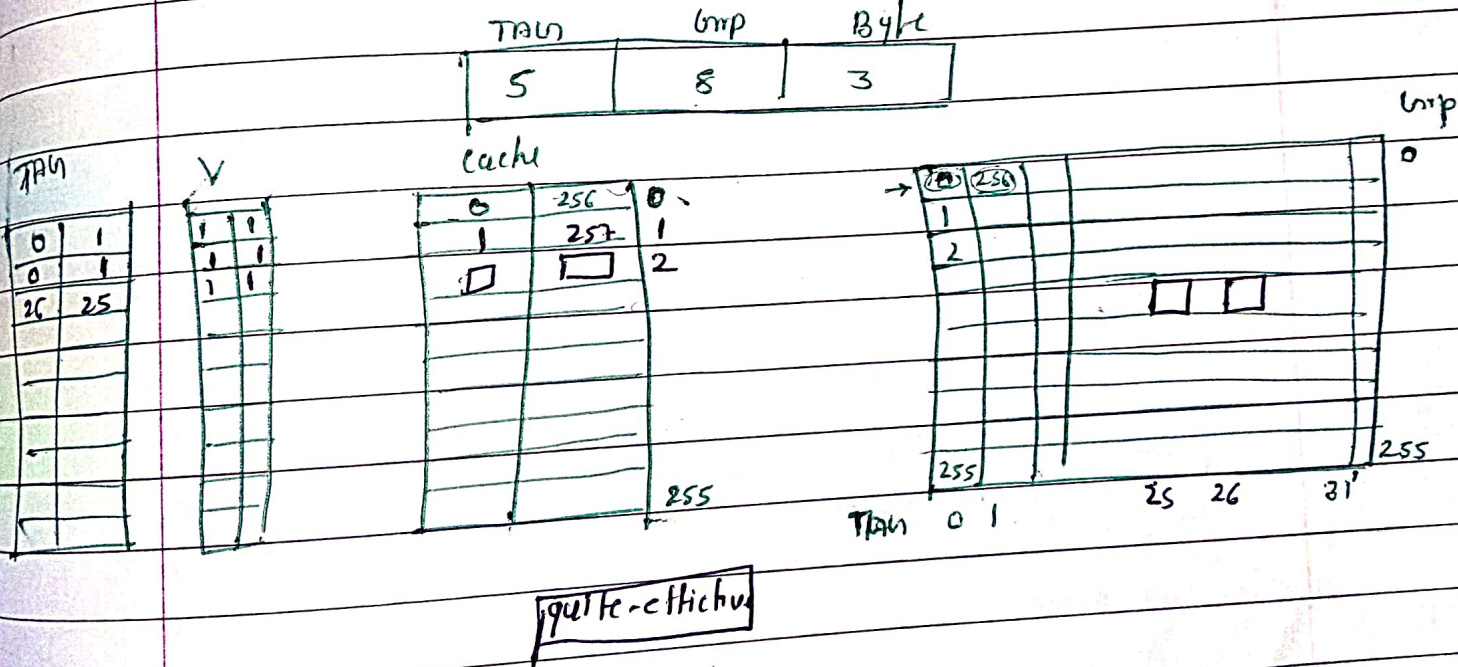


Direct Mapping

Date: _____ Page no: _____



Set - Associative [2-way set associative]



fully-effective