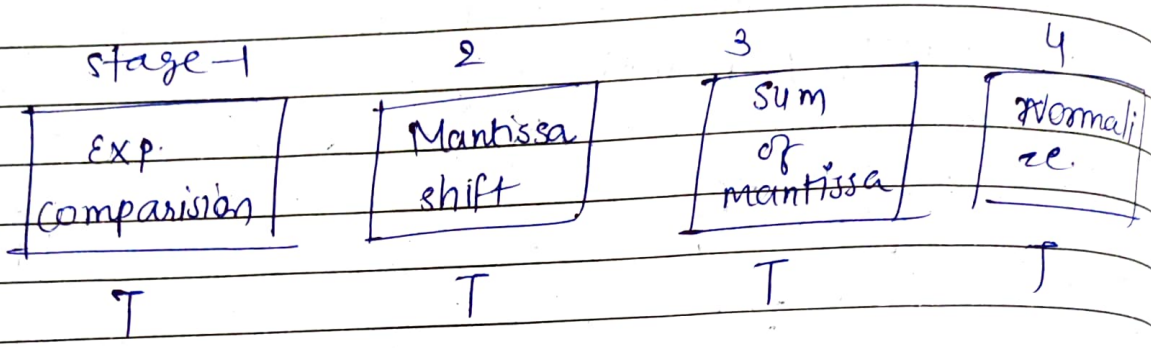
DATE:
PAGE:

sessional - III

→ for addition of two floating numbers.
- compare of exp
- shifting of mantissa
- summing the mantissa
- Normalize the result.



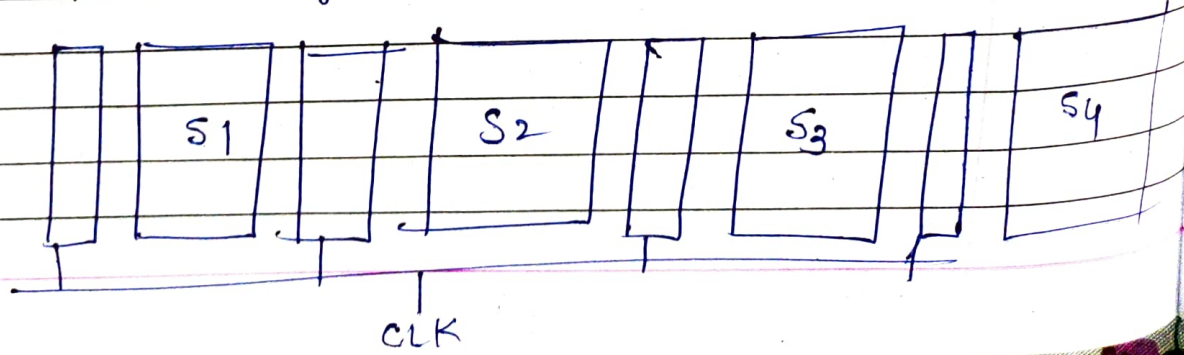
total delay = 4T

→ while x_1, y_1 is in mantissa shift we can put x_2, y_2 in another (exp comp) phase.

→ while two numbers are in one phase, other two numbers we can put in ~~seco~~ another stage (phase).

∴ using pipelining = 6T (4T+T+T=6T for adding 3 numbers)
without pipelining = 12T

To avoid collision buffer regs are placed bet^n two stages.



DATE:
PAGE:

→ delay = max{TPi} + TR ← buffer reg.

→ Throughput = $\frac{1}{\text{Max}\{TPi\}+TR}$

→ speedup factor = $\frac{\text{Non pipelining proc}}{\text{pipelined system.}}$

→ Non-pipelined proc. → N no. of processes 1 operation is completed in 4T time.
- Non pipelined system = 4NT.

→ for pipelined system = 4T + (N−1)T
= (3+N)T

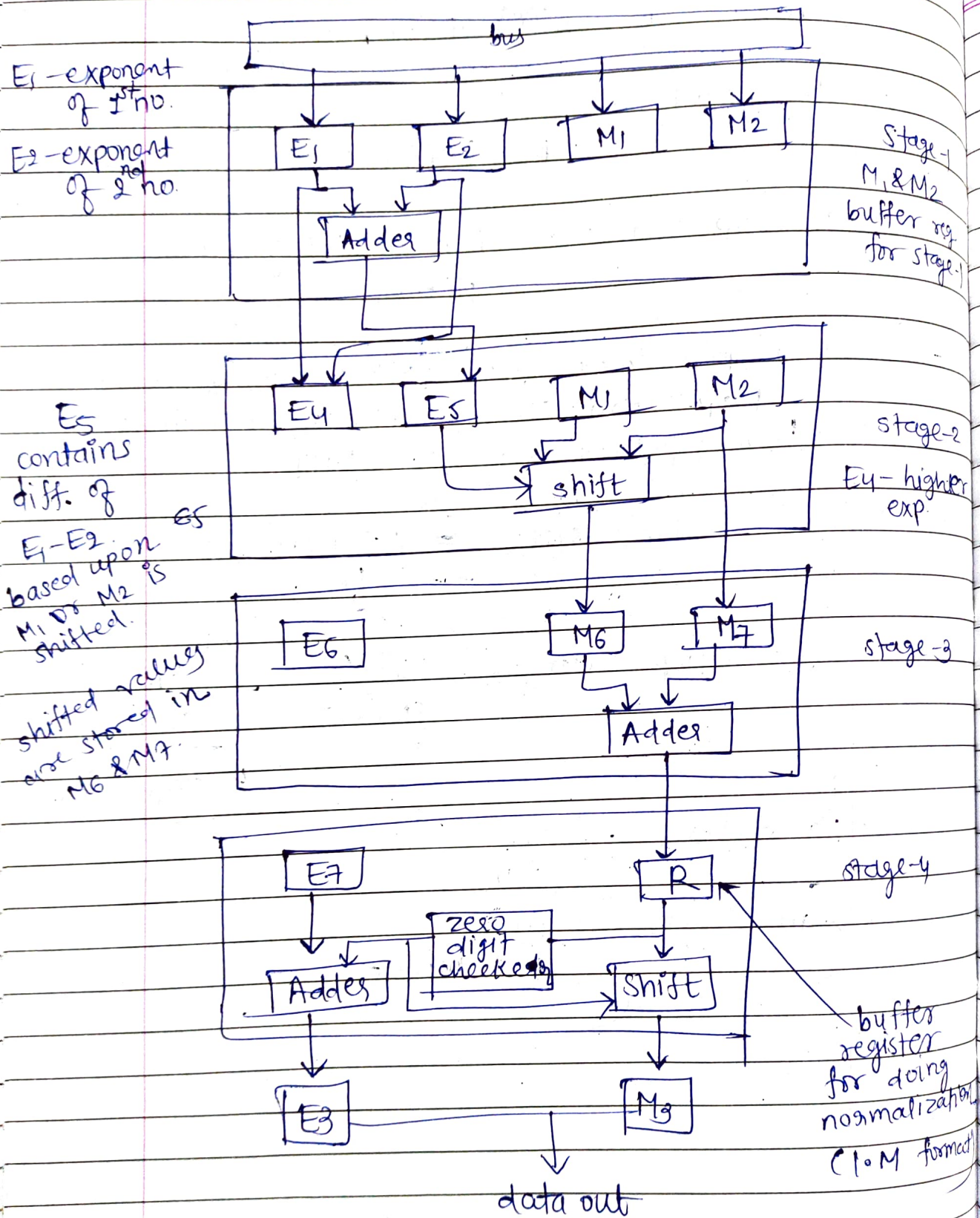
∴ speedup factor = $\frac{4NT}{(N+3)T}$ = $\boxed{\frac{4N}{N+3}}$

4 stages

if 5 stages, = $\frac{5N}{N+4}$.

m stages = $\frac{mN}{N+(m-1)}$

4 Stage pipeline



Advantage: speed is high

Drawback: because of buffer registers cost will be increase

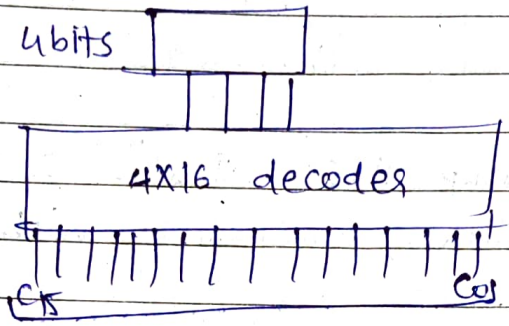
Encoding of Micro Instruction

Inst^{n}s are of 2 types.

i) Horizontal

ii) Vertical.

In Vertical format, 16 Inst^{n}s ⟹ 4 bits



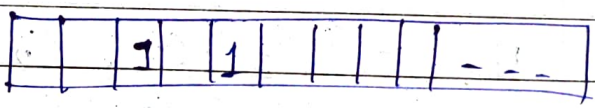
at a time only 1 is high.

In register file

two selection lines should be provided at same time.

but using vertical, we can only activate one signal at a time.

In horizontal,



we can activate one or more signal at a time.

Ex

μInst^{n}s	control signals
I_1	c_1, c_2, c_3, c_4, c_5
I_2	c_1, c_3, c_4, c_6
I_3	c_5, c_6
I_4	c_4, c_5, c_8
I_5	c_7, c_8
I_6	c_1, c_8, c_9
I_7	c_3, c_4, c_8
I_8	c_1, c_2, c_9

Have to group these μInst^{n}s so that it should be ~~ex~~ mutually exclusive with each other

$M_1 = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9\}$

M_2 = only c_1 & c_7 can be grouped because it is ~~not~~ mutually exclusive

$= \{c_1c_7, c_2c_6, c_2c_7, c_2c_8, c_3c_7, c_3c_9, c_4c_7, c_4c_9, c_5c_7, c_5c_9, c_6c_7, c_6c_8, c_6c_9, c_7c_9\}$

$M_3 = \{c_2c_6c_7, c_2c_6c_8, c_3c_7c_9, c_4c_7c_9, c_5c_7c_9, c_6c_7c_9\}$

Group: c_1c_7, $c_2c_6c_7$, $c_2c_6c_8$, $c_3c_7c_9$, $c_4c_7c_9$, $c_5c_7c_9$, $c_6c_7c_9$

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
C_1C_7	✓						✓		
$C_2C_6C_7$		✓				✓	✓		
$C_2C_6C_8$		✓				✓		✓	
$C_3C_7C_9$			✓				✓		✓
$C_4C_7C_9$				✓			✓		✓
$C_5C_7C_9$					✓		✓		✓
$C_6C_7C_9$						✓	✓		✓

Find minimal covers class

- Above rounded are minimal covers class.

C_1C_7 – 1 bit to indicate only C_1 because C_7 is covered into anothee classes as well.

$C_2C_6C_8$ – 2 bits

00 – None

01 – C_2, 10 – C_6, 11 – C_8

$C_3C_7C_9$ – 2 bits

00 – None

01 – C_3, 10 – C_7, 11 – C_9

$C_4C_7C_9$ – 1 bit for C_4 because C_7 C_9 already covered.

$C_5C_7C_9$ – 1 bit for C_5

1	2 bits	2 bits	C_4	C_5
C_1	$C_2C_6C_8$	$C_3C_7C_9$		

I1 = 1 0 0 1 1 include C_1, C_2, C_3, C_4, C_5

Ex.

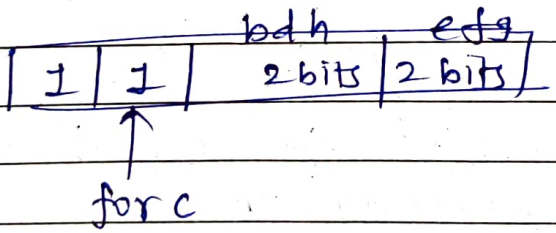
μInst^{n}s	Control signals
I_1	a b c g
I_2	a c e h
I_3	a d f
I_4	b c f

$M_1 = \{a, b, c, d, e, f, g, h\}$

$M_2 = \{bd, be, bh, cd, de, dg, dh, ef, eg, fg, fh, gh\}$

$M_3 = \{bde, bdh, dgh, efg, fgh\}$

	a	b	c	d	e	f	g	h
a	✓							
cd			✓	✓				
bde		✓		✓	✓			
bdh		✓		✓				✓
dgh				✓			✓	✓
efg					✓	✓	✓	
fgh						✓	✓	✓



Memory Organization

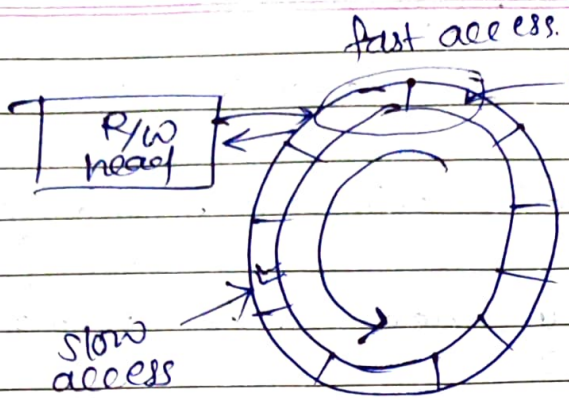
- Address lines → n → 2^n
- proc: only knows primary memory
- secondary mem. >> primary mem. size
- Only activate programs are there into the primary mem.
- Cache memory are placed bet^n proc. & main memory
- Secondary mem. – usec
  primary mem – (5 to 10 times slower than proc)
- DRAM are cheaper than SRAM
- primary → c/s is high & access time is low
  secondary → c/s is low & access time is high
  ↓
  cost/bit
- write cycle takes more clock cycle than read cycle

Access Modes

1) Serial Access
2) Random Access
3) SemiRandom Access – Hard disk / CD drive used in

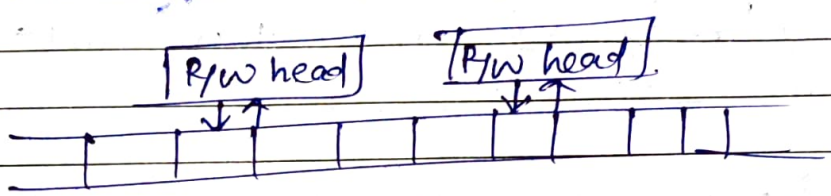
DATE:
PAGE:

① Serial Access Mode.



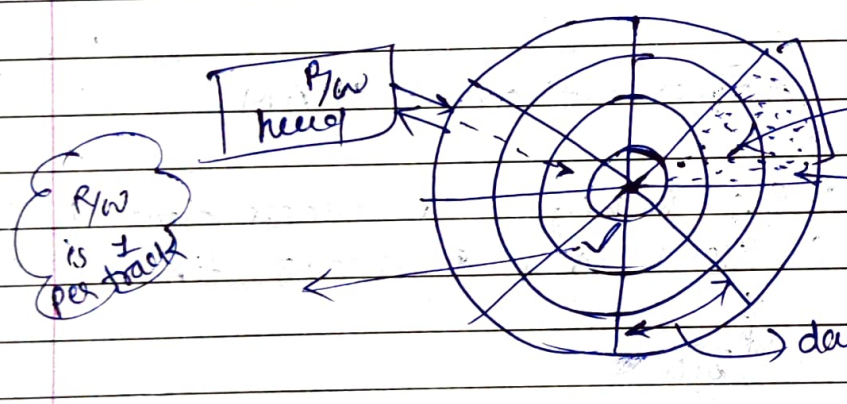
depends upon current position of R/W head.

②



data is whereves it will take same amount of time to access.

③

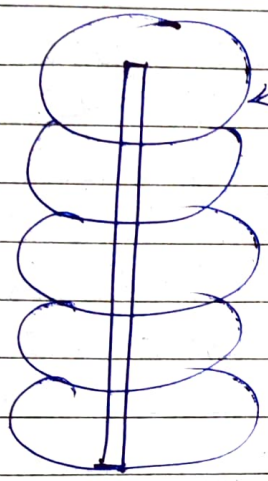


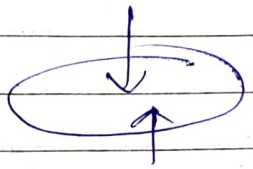
- time taken to move R/W head on a given track is known as "seek time".
- R/W head is under given sector disk has to rotate → it is known as Rotational delay.
- disk controller is used to read data or write data with bus takes time → known as Access time

- Avg. Rotational delay = $\frac{\text{Rotational delay}}{2}$

- In semiac random access, we don't have seek time only we have rotational delay & access time

⇒





data is on both the surface

← In this, 10 surfaces are there

← on both surface there would be R/W head.

track-0 on Pladder 1] → makes
track-0 on Pladder 2] cylinder 0
- - - - Pladder 5

Ex! Consider a Hard disk, with 4 surfaces, 64 tracks/surface, 128 sectors per track, 256 bytes per sector.

i) Find capacity of the Harddisk.

→ Capacity = 8 Mega Byte
= 4 * 64 * 128 * 256 bytes

ii) Disk is rotating at 3600 rpm (rotation per minute). What is a data transfer rate?

→ Data transfer rate = No. of Rotations/sec * track capacity * No. of surfaces

$= \frac{3600}{60}$ rps $=$ 60 rotations per sec

$$DTR = \frac{\text{capacity}}{60 * 128 * 256 * 4} = 7.5 \text{ MB/sec}$$

iii) Average access time = Avg. rotational delay
= 8.3 msec
= ~~3600~~
2

Rotational latency $= \frac{1}{60}$ sec $= 16.67$ msec
(60 ← rotations/sec)

Average Rotational latency $= \left(\frac{16.67}{2}\right) = 8.3$ msec

- **Classification Of Memory Systems.**

(a)

Volatile	Non-volatile
- stored data is lost when the power is switched off	- stored data is retained even when the power is switched off
- ex CMOS static mem, CMOS dynamic memory	- ex ROM, Magnetic disk, CDROM / DVD

(b)

Random-access	Direct / Sequential Access
- R/w time is indep^{t} of the memory location being accessed.	- stored data can only be accessed sequentially in a particular order

- Ex CMOS memory (RAM and ROM)	- Ex Magnetic tape - part of the access is sequential and part is random - direct access or semi-random access

(c)

ROM	RAM
- data is stored in permanent or semi-permanent.	- data access time is the same indep^{n} of the location
- data written during manufacture or in the laboratory.	- data once written are retained as long as power is on. - used in main/cache mem.
- Ex ROM, PROM, EPROM	- Ex SRAM, DRAM.

Sol: 2

No. of tracks = 500
No. of sectors/track = 100
No. of bytes/sector = 500
Time taken by the head to move from one track to adjacent track = 1 ms
Rotation speed = 600 rpm = 0.1 sec
taken
What is the avg time for ~~transformi~~ transferring 450 bytes from the disk?

→ Avg. time to transfer =
Avg. seek time + Avg. rotational delay
+ Data transfer time.

- Now, Avg seek time = $\frac{\Sigma(0+1+2+\ldots+499)}{500}$

= 249.5 ms

time taken to move from,
1st track to 1st track = 0 ms
1st track to 2nd track = 1 ms, 2 ms, 3 ms, ... 499 ms

- Avg. Rotational Delay = $\frac{0.1}{2}$ = 50 ms

- Data transfer time :
In 1 ~~track~~ rotation we can read data on one track = 100 * 500 ← tracks
= 50,000 B

Now, 50000 B → 0.1 sec
250 B → $\frac{250 \times 0.1}{50,000}$ = 0.5 ms

∴ Therefore, ATT = 249.5 + 50 + 0.5
= 300 ms

Ex A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces and 1000 cylinders. Address of the sector is given as a triple (c,h,s), where c = cylinder number, h = surface number, s = sector number. Thus, 0th sector is addressed as (0,0,0) the 1st sector as

(0,0,1) and so on. the address <400, 16, 29> corresponds to the sector number

→ <400, 16, 29> represents 400 cylinders are passed. (0-399) and thus, for each cylinder 20 surfaces (10 platters * 2 surface each) and each cylinder has 63 sectors per surface.

Hence, we have passed 0-399
= 400 * 20 * 63 sectors + In 400th cylinder we have passed 16 surfaces (0-15) each of which again contains 63 sectors per cylinder so
16 * 63 sectors + Now on the 16th surface we are on the 29th sector. so sector no.
= 400 * 20 * 63 + 16 * 63 + 29
= 505037

Ex Consider the data given in the previous question, The address of the 1039th sector is

→ 1 cylinder have 63 * 20 = 1260 no. of sectors
(10 platters * 2 surfaces)

Here, 1260 > 1039 → so it would be in 1st cylinder (cylinder no. = 0)

Now, surface no. (head no.) = 1039 / 63 = 16

sector no. = 1039 % 63 = 31 sector no.

So, (C,h,s) = (0, 16, 31)

DATE:
PAGE:

- **Characteristic / Types of Memory**

1. Destructive Read out RAM.

-whenever Read op. is done with content of memory data will be lost. to return data, same them is copied in Buffer register & after reading it needs to write in that place.

2. Dynamic - data is stored as charges in capacitor, periodically dynamic RAM needs to be refressed.

3. Volatile - temporary data.

- In Non-destructive RAM - we did not have to write after read operation.

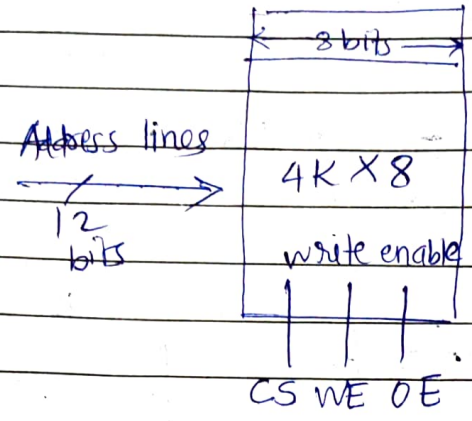
- Access time (t_A) - time which memory takes to receive request & give output is known as access time.

- Time betⁿ 2 successive Read or 2 successive write operation is called cycle time (t_H).

- cycle time & access time may not be same.

- t_A (Access time) is same as cycle time in static memory.

RAM IC :



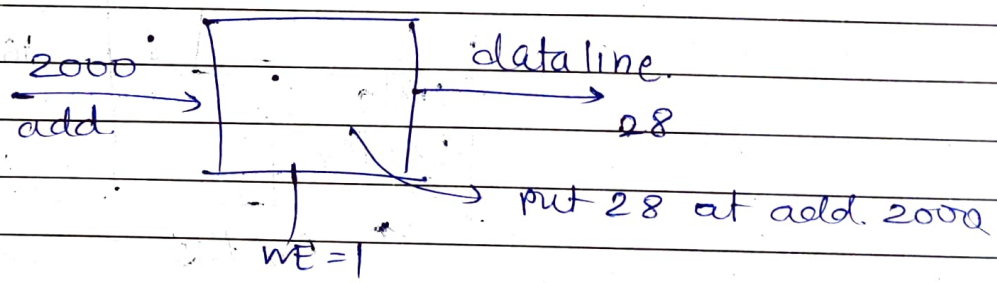
- $4K = 2^2 \times 2^{10} = 2^{12}$

4096 diff. memory location

↳ 12 address lines are required

- LDA 2000, load into Ac.
  ↳ memory read.

* WE – Write Enable input
when WE = 1 indicates it has to perform memory write.



WE = 0, read op.

* OE – output enable
whenever data bus is ready to receive or send data.

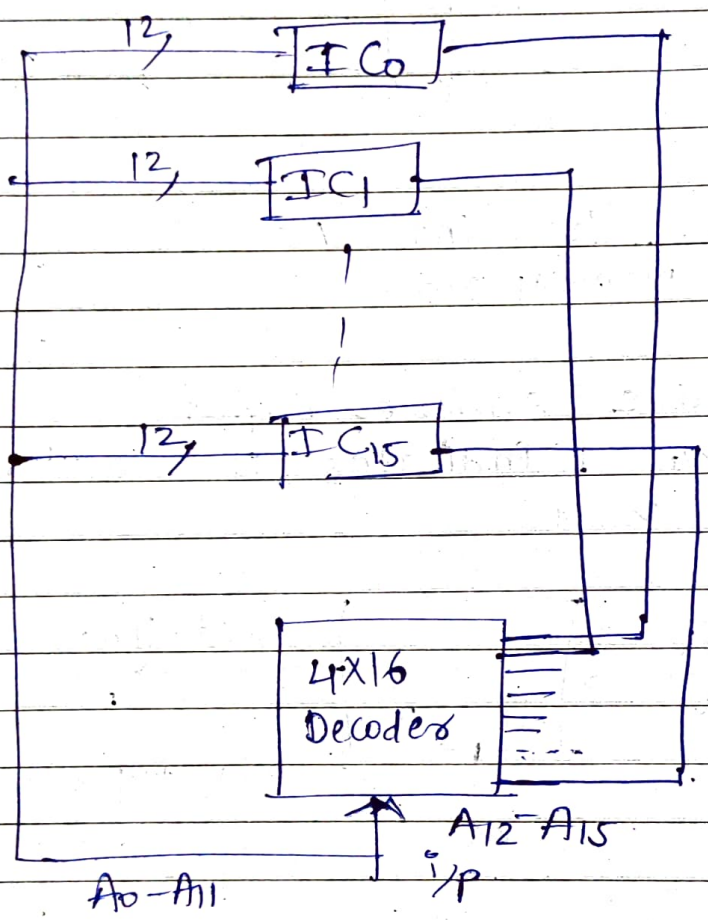
OE = 0, Bus is not ready to receive or send data

size of 8085 microprocessor = 2^{16} = 64 KB
16 add. lines $\frac{2^{10} \cdot 2^6}{K}$ ↳ 64

DATE:
PAGE:

Que I want to design 8085 µProc. using 4K x 8 memory

– 16 add. lines → 4K X 8 IC are required to make 64KB



12 add. lines are provided (A0 – A11)

16 o/p s are connected with 16 ICs.

Que OFFFH

A15 A12 A11-A8 A7-A4 A3-A0

0000

↳ IC0 chip selected → on this chip FFF mem location (last loc.)

DATE:
PAGE:

Que. 1010 H.

A_{15} A_{14} A_{13} A_{12} A_{11} A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0

0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0

↓ IC1 selected

↓ 12 addresses in it is selected

Que For 32 KB

↳ 8 IC required.

↓

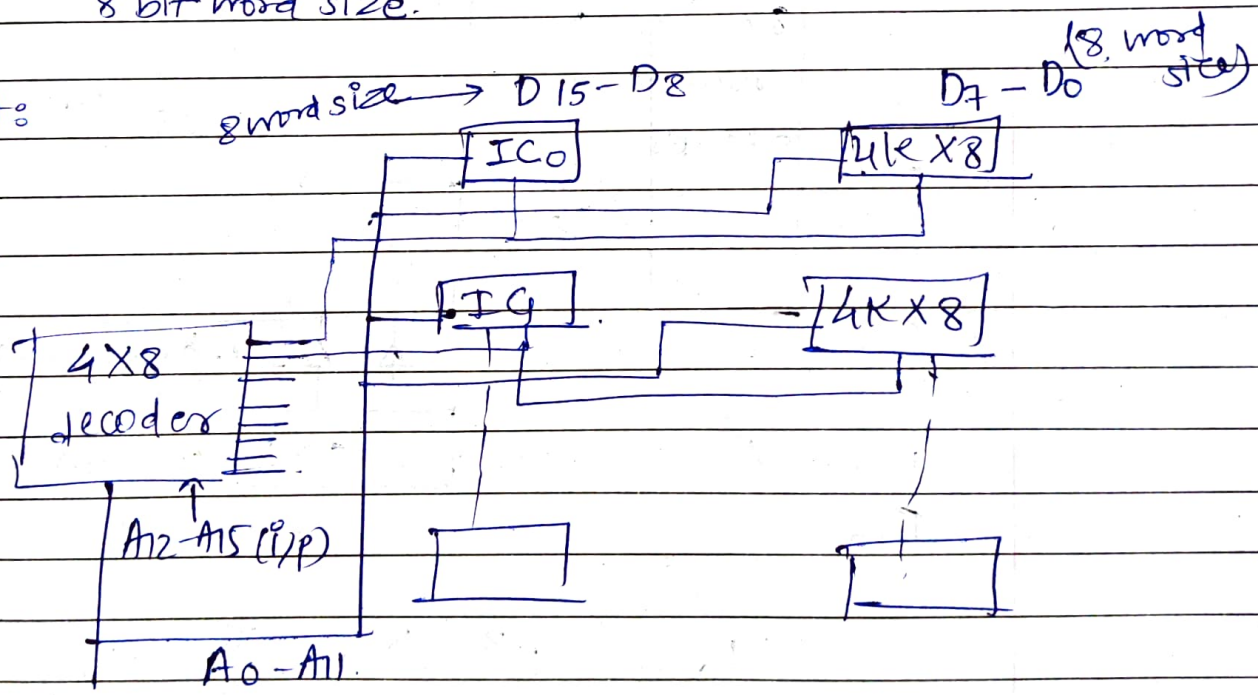
3 X 8 decoder required, 3 input $A_7 - A_5$

Que 4K X 8 memory but require size = 16 bits

↓

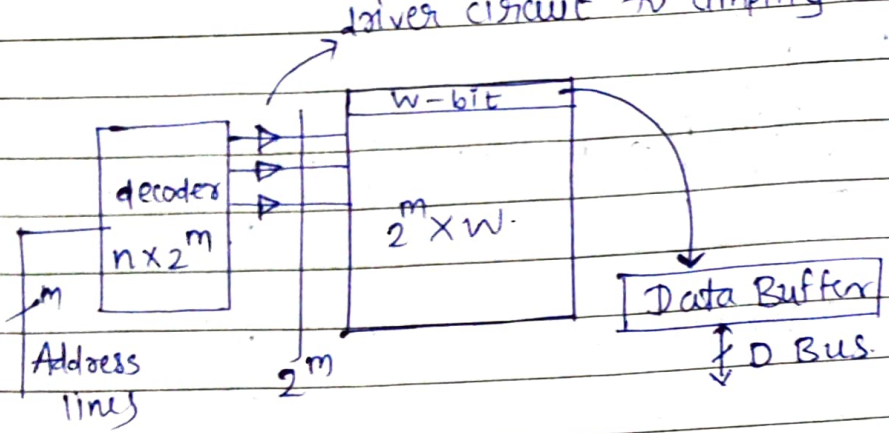
8 bit word size.

Improvement:



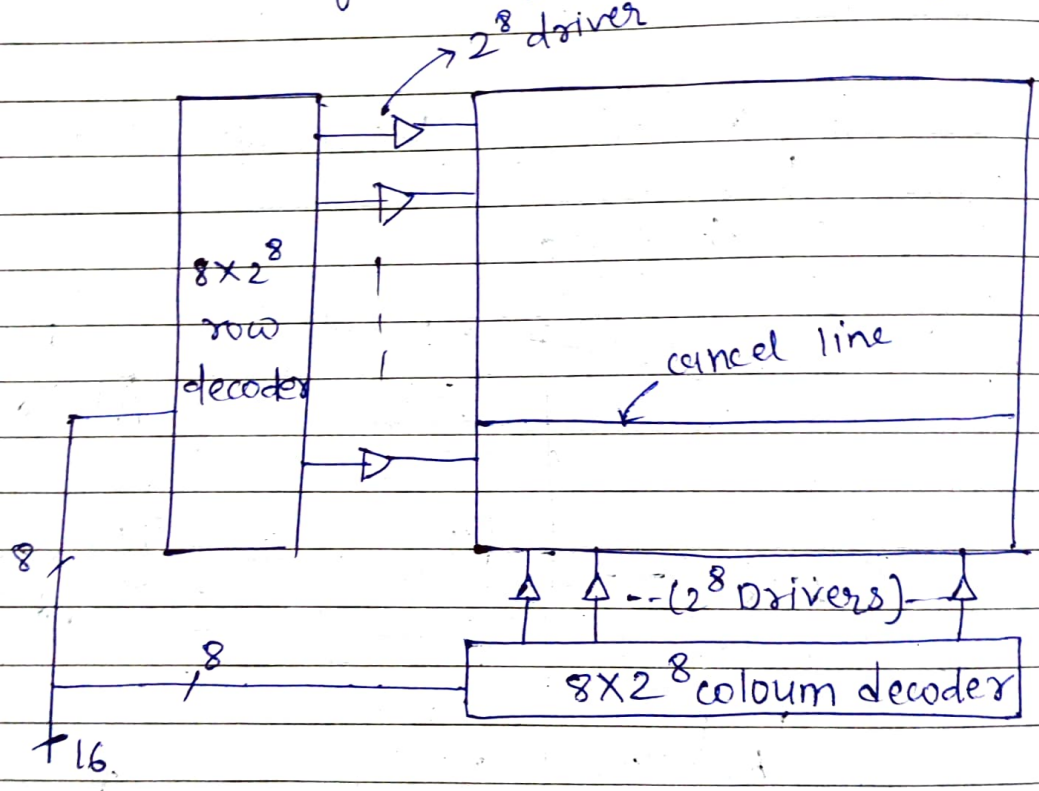
1 column 2 ICs each of 8 bit word size

= 8 + 8 = 16 bit word size



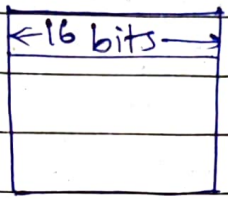
1-D form of Arrangement of Memory

- 2-D form of Arrangement of Memory.



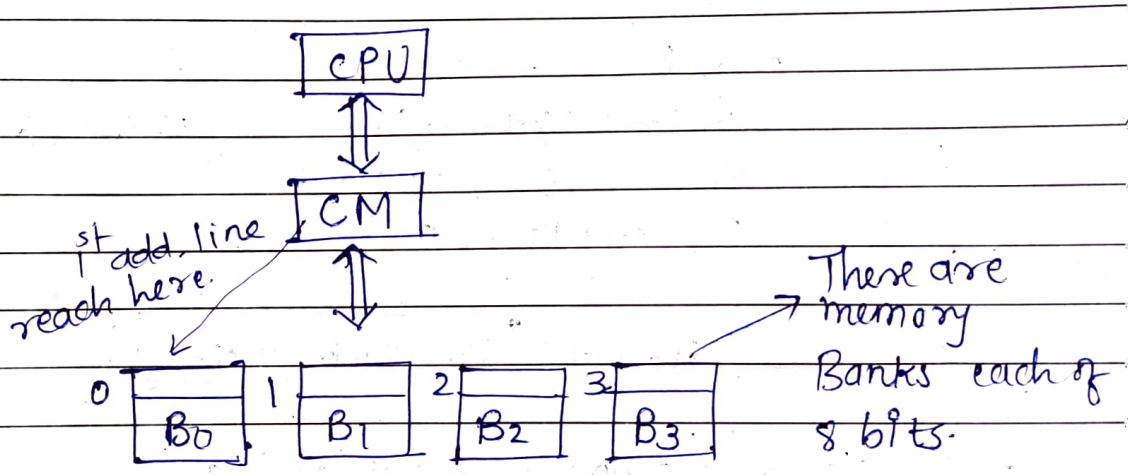
Total = $2^8 + 2^8 = 2^{16}$ Drivers are required

- 2-D form reduces no. of circuit required.
- It is recommended to use 2-D form.
- To increase the speed of memory.



wordsize = 16 bits
But databus is of 8 bits
so, databus first access the 1st 8 bits And then other 8.

- To access 16 bits more size of proc. is required.
- Disadvantage: cost of circuit increase.



- B0 start executing, B1, B2, B3 are free.
- At same time, 2nd Address is placed at B1 which is free, processing speed increase.

(1) Assume that there are latency, it takes 1 clock cycle would be taken to reach the address from CPU to main memory, main memory access time = 15 clock cycle, transfer time 1 clock cycle from memory CPU. find time to catch 4 words without memory interleaving.

→ 1 word size = 15 + 1 + 1

4 words = 17 × 4

= 68 clock cycle.

But using Memory Banks, in case of not using memory interleaving.

for B_0 = 17

B_1 → 15 + 1 = 16. 16 × 3 = 48

B_2 → 15 + 1 = 16

B_3 → 15 + 1 = 16

48 + 17 = 65 clock cycles

→ Using memory interleaving

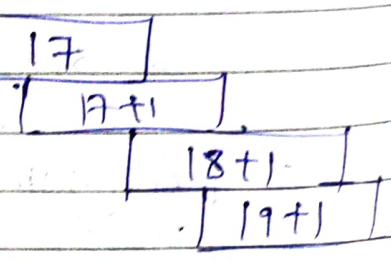
B_0 = 17

B_1 = 1

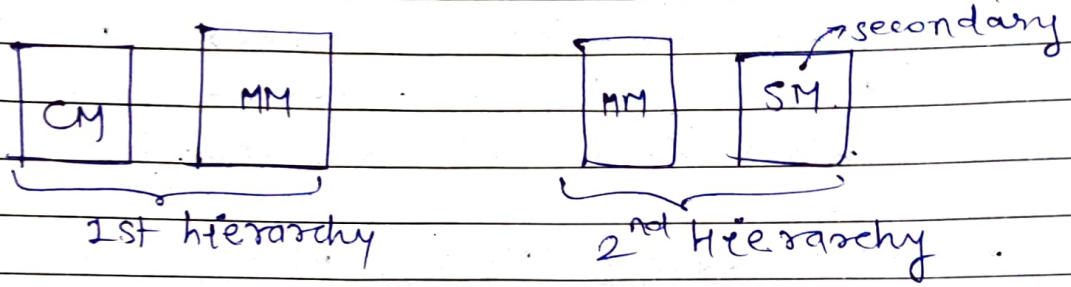
B_2 = 1

B_3 = 1

Total = 17 + 1 + 1 + 1 = 20 clock cycle



- level - 1 memory nearest to memory (L_3)
- cost per bit is very much less for higher order than lower order memory (L_1)
- storage of lower level less than higher level memory
- Access time of lower level memory is less than that of higher level memory



- Required data should be present in fastest memory (CM).
- whatever is present in MM is not present in CM
- Hardware Controller would take care of block transfer from MM to CM.
- In OS, we have memory management unit which would take transfer from SM to MM.

Virtual Memory

- In assembly language, the data is stored in physical memory.
- User would find that it is having very ~~to~~ vast memory

- 10 programs running in processor (CPU) then all 10 programs should be present in MM, processor only knows MM.

- If size of prog. is too large, instead of placing all in MM at same time, that prog. is divided in no. of Blocks. One of the block place in MM, MM execute that Block (B1) & after executing, remove that Block, new Block is added in MM (B2), these $B_1, B_2, \ldots$ Blocks needs to map in SM which forms virtual Memory.

- The address (Blocks) present in MM are physical address.

- MM is build using dynamic RAM, requires period refreshing, CM is Build using static RAM and it is nearer to processor, Therefore CM is more faster than MM.

- Prediction Of Addresses : locality of Reference.

address		
1000	MVI A,10	
1002	MOV B,A	→ due to prediction it assume that next inst^n is at very next add
1003	MVI C,10	
1004	loop: DCR C	
1007	JNZ loop	

- That type of locality where it predicts that it needs to be placed at very next add. is called spatial locality.
  ↳ consecutive inst^n are placed at consecutive address.

- For loops case, entire loops should be copied from higher (secondary) level to lower level (MM).

```
for loop
{
  stat 1
  stat 2
  ---
}
```

all statements needs to be copied.

- In temporal locality, in case of loop, entire loops need to be locality.

- **Cost and Performance (Measure)**

Assume 2-level memory

$$\text{Cost } C = \frac{C_1 S_1 + C_2 S_2}{S_1 + S_2}$$

where C_1 = cost/bit of Memory M_1
S_1 = Storage capacity of M_1
C_2 = cost/bit of M_2
S_2 = storage capacity of M_2

→ Hit ratio : prob. of Reference word if it is present in physical memory

$$H = \frac{N_1}{N_1 + N_2}$$

where N_1 = no. of references of M_1

N_2 = no. of references of M_2.

(word) in MM.

if it is not present it is miss.

miss $M = 1 - H$.

max. Hit or Miss is 1. (max. prob.)

→ Access time : time to access the content of memory

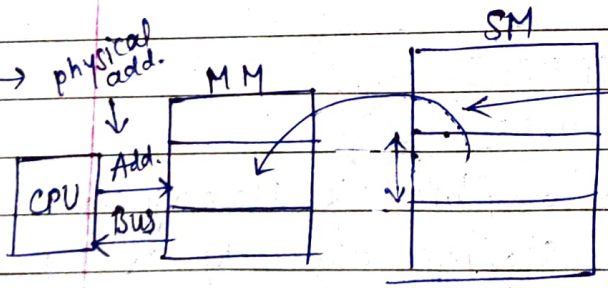
$$t_a = H t_{a_1} + (1-H) t_{a_2} \quad \text{—— (1)}$$

Avg. access time.

if word is present $H = 1$.

$$t_a = t_{a_1}$$

if miss ($H = 0$) → $t_a = t_{a_2}$



copied whole block if it is not present in MM, which is required by CPU (by virtual add.)

Proc always only knows MM. SM is like IO device. proc. does not know about SM.

t_{a_2} = Block transfer time + Access time of MM

$$t_{a_2} = t_B + t_{a_1}$$

put into (1).

$$t_a = H t_{a_1} + (1-H)(t_B + t_{a_1})$$
$$= H t_{a_1} + t_B + t_{a_1} - H t_B - H t_{a_1}$$

$$\therefore \boxed{t_a = t_{a_1} + (1-H) t_B}$$

→ Access ratio $\boxed{r = \frac{t_{a_2}}{t_{a_1}}}$

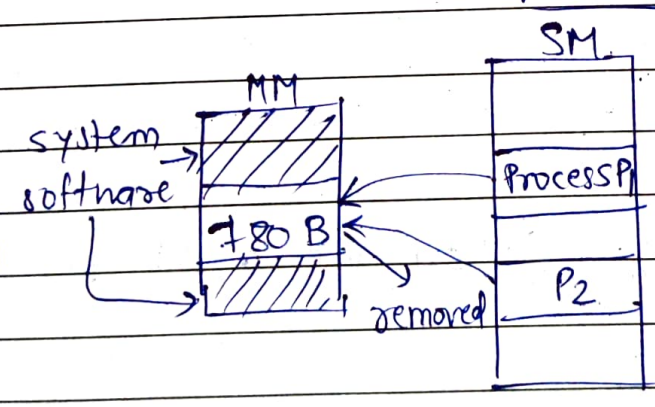
→ Access efficiency (How far it is from the avg. access time)

$$e = \frac{t_{a_1}}{t_a}$$
$$= \frac{t_{a_1}}{H t_{a_1} + (1-H) t_{a_2}}$$

$$\therefore \boxed{e = \frac{1}{r + (1-r)H}}$$

→ Utilization factor $U = \frac{\text{Useful space}}{\text{Total space}}$

$$\boxed{u = \frac{Su}{S}}$$



once P1 is over it is removed from MM and another P2 is placed.

entire memory is not utilized every time

Reasons

P1 requires = 780 , P2 = 550

① space ($P_1 - P_2$) is wasted → this is known as fragm

(anticipation)

- due to prediction if word w is required it would also copied next add w+1 but it might possible that proc. don't require it so it wastes the space of MM. at that time no other block can use that space of MM.

because of spacial locality.

- Instead of word by word ~~copy~~ copy, block by block copy is good

② Prediction may go wrong

③ part of the MM is utilized by system s/w so it can't be useful.

- Address translation

done by

1) - Programmer (Assembly lang. Prog.)
   - Requires when programmer writes prog in ALP.

   - If we use kit, we have to provide starting add., opcode of Inst^{n}s.

2) Compiler

3) loader

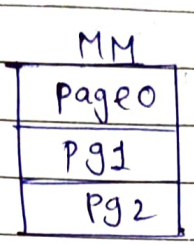
4) Run the prog. (when we right recursion)

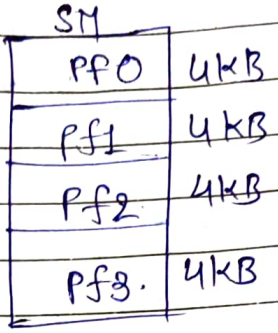
variable

fixed size block - is also known as pages.

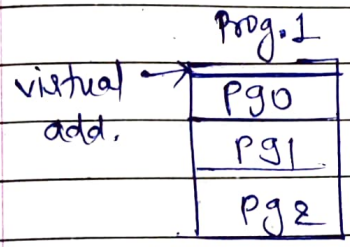
~~variable~~ fixed size blocks. - known as page frames

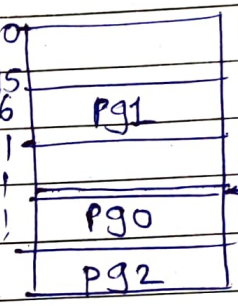
same size





MM partioned into page frames, page can be accomoded in ~~page~~ frames.





physical add. of first byte of Pg0 add. ⇒ 48.

- Page table: per proc one page table would be maintained.

(virtual) Av	(physical) Ap
0	3
1	1
2	4
pages	page frames

physical add. = 48.

virtual add = 1 ⟹ physical add. = 49.

virtual add. = 0 ⟹ physical add. = 48.