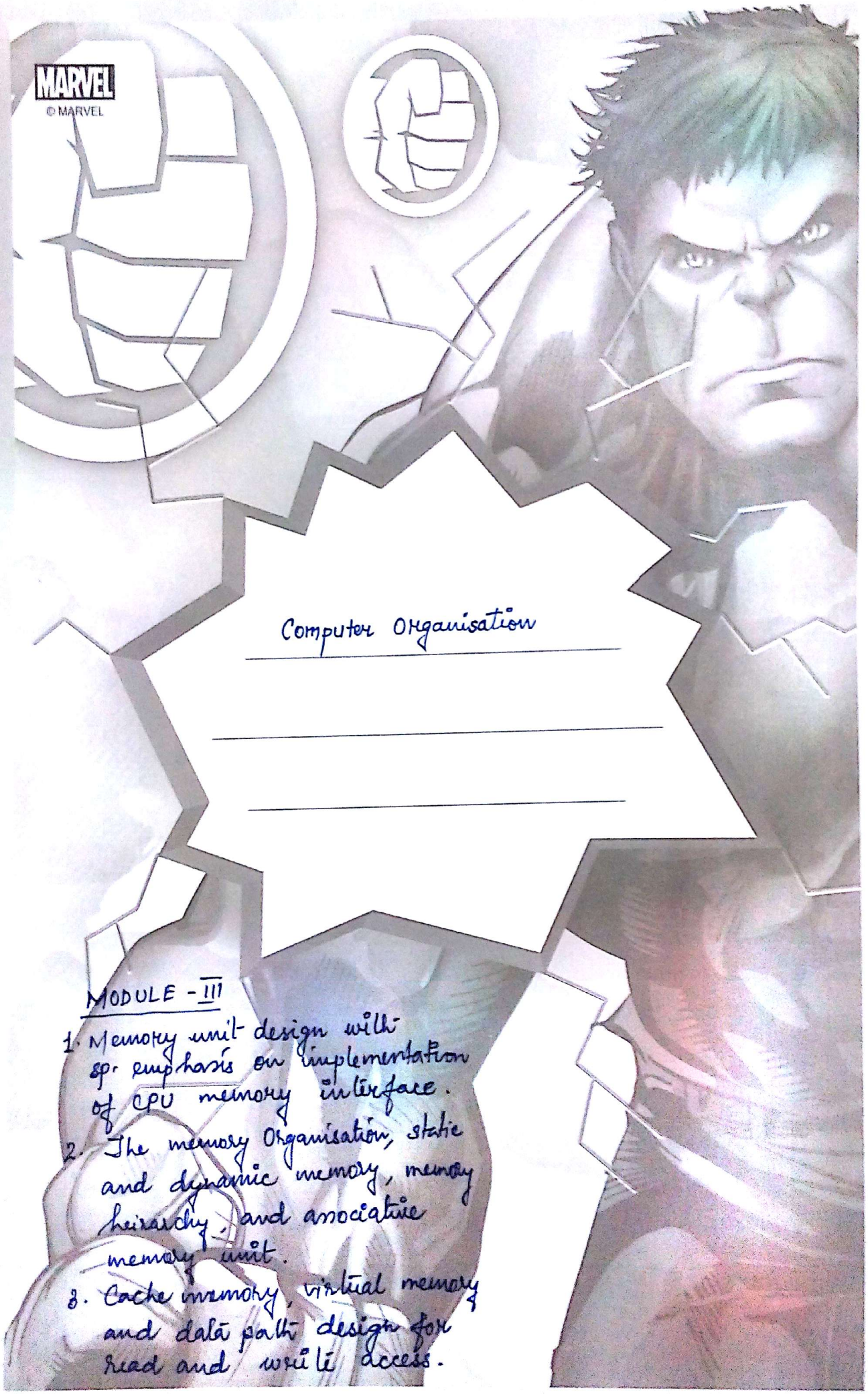


Computer Organisation

MODULE - III

1. Memory unit design with sp. emphasis on implementation of CPU memory interface.
2. The memory Organisation, static and dynamic memory, memory hierarchy, and associative memory unit.
3. Cache memory, virtual memory and data path design for read and write access.

Books

1. Computer Organisation
   - T.K. Ghosh
2. Computer organisation
   - M.M. Mano
3.
   - Hammalure (!!?)

1st August, 2017

Module - III

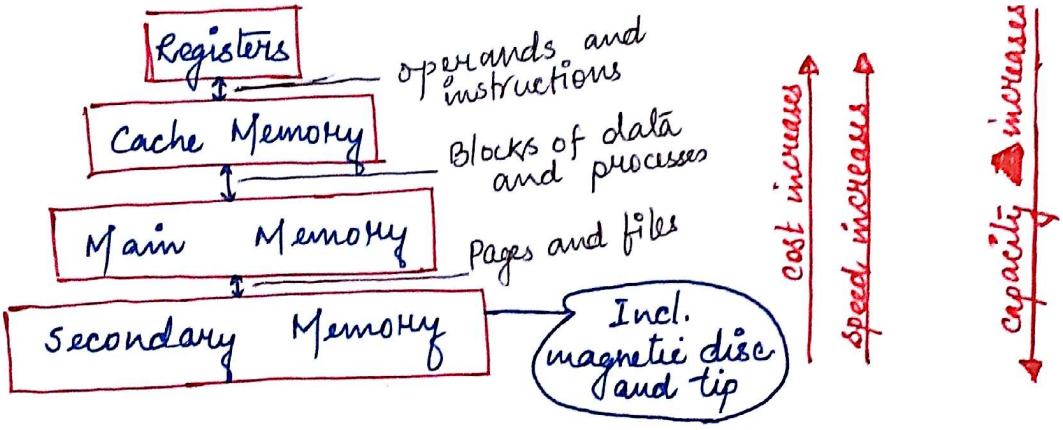
Parameters of Computer Memory Organisation

1. Capacity ⇒ represented as a m×n matrix
   where, m → no. of addresses
   n → word length

2. Speed ⇒ represented by three categories
   - cycle time (t_C)
   - access time (t_A)
   - recovery time (t_R)

3. Bandwidth (or Data Transfer Rate) ⇒

Memory H



Memory Access Methods

- Sequential
- Random
- Direct
- Associative

8th August, 2017

Properties of Memory

i) Inclusion
ii) Coherence
iii) Locality of Reference

INCLUSION

- If we call a memory level as i^{th} level, then the higher memory level is called $(i+1)^{th}$ level.

- $M_1 \subset M_2 \subset M_3 \subset M_4$

$M_1 \rightarrow$ Reg $\rightarrow (i-1)^{th}$
$M_2 \rightarrow$ cache $\rightarrow i^{th}$ (say)
$M_3 \rightarrow$ MM $\rightarrow (i+1)^{th}$
$M_4 \rightarrow$ SM $\rightarrow (i+2)^{th}$

COHERENCE

If a particular memory block's data is updated, then the later memory value is always used by the operating instruction.

LOCALITY OF REFERENCE

Types of locality –
i) Temporal locality (locality of time). eg: loop
ii) Spatial locality (locality of space). eg: array structure
iii) Sequential locality
eg: an instruction set

REGISTER

(i) INSTRUCTION REGISTER

(ii) DATA REGISTER → Involves with the I/O peripherals

(iii) ACCUMULATOR REGISTER → stores the newly calculated values

(iv) PROGRAM COUNTER/INSTRUCTION POINTER → keeps track of which data to be extracted from cache

(v) MEMORY ADDRESS REGISTER → [indirect data fetching by address]

(vi) MEMORY DATA REGISTER → Deals with internal Cache &/OR RAM

(vii) MEMORY BUFFER REGISTER → (interlinked)

↳ Values accessed by MAR are stored here at MBR

FUNCTIONS

- FETCH
- DECODE
- EXECUTE

CACHE MEMORY

(2) Speed:- (i) t_c (Cycle time)
(ii) t_A (Access ")
(iii) t_R (Recovery ")

$t_R = t_c + t_A$

• Memory Access Methods:-

(i) Sequential Access → used by or incorporated in Magnetic Tape.

(ii) Random " → RAM, ROM

(iii) Direct " → incorporated in Magnetic Disk.

(iv) Associative "

→ Combination of Sequential & ~~Direct~~ Random Access.

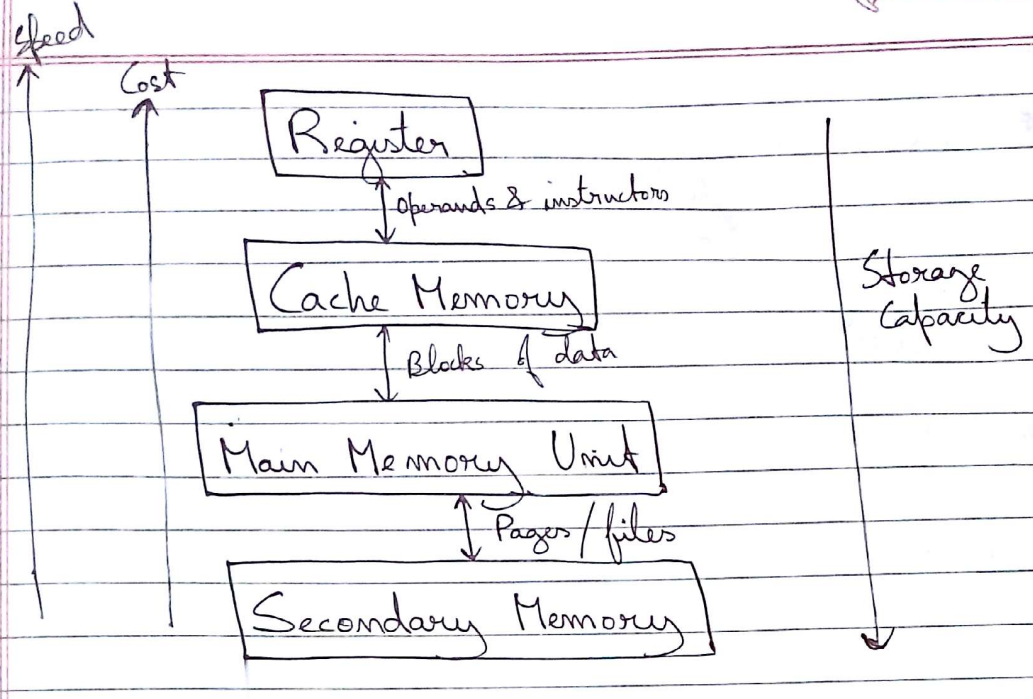
Ex:- If we want the 10th data (—

Sequential → skips 1 to 9 & then goes to 10th
Random → goes ~~directly~~ straight to 10.

→ If we want the 10th data in 4th disk in a magnetic disk, then by direct method, we randomly go to 4th disk & then sequentially goes to its 10th data.

• Memory Hierarchy:-

(P.T.O)



Speed ∝ Cost

8/8/17

Properties of memory :-

(i) Inclusions:-

(ii) Coherence (Cache) → if there are subsequent instructions using the same set of values, then the updated value will reflect in the next operation.

(iii) Locality of References:-

3 types of locality →
- (i) Temporal locality (ex. → loop)
- (ii) Spatial " (ex. → array)
- (iii) Sequential " (ex. → instruction set on a program)

Ex:- C = A + B
D = C / B
↳ here this value of C is updated after 1st instruction and this updated value is used here.

- Register :- Smallest unit of memory directly linked to processor and it stores current or ongoing value.

Types :

(i) Instruction register (IR)

(ii) Data register (DR)

(iii) Accumulator register (AR)

(iv) Program counter / Instruction pointer (PC/IP)

(v) Memory Address register (MAR)

(vi) " Data " (MDR)

(vii) Memory Buffer " (MBR)

DR → calculates the value of data & shows to the peripheral devices i.e the data comes out as output.

MDR → calculates the data & stores the data in memory for further calculations.

AC → stores the final result.

PC / IP → Cache stores the upcoming instructions to be performed, PC stores the order and sequence for upcoming instructions.

MBR → Stores the data.

123F → address
(A) → value
↳ in MBR

- Functions of Register :-

(i) Fetch
(ii) Decode
(iii) Execute.

- Main Memory :-

RAM (Re-writable)

ROM (non re-writable, can be erased forcefully only)
(~~boost~~ boot strap loader)

Schematic diagram of RAM

($2^9 = 512$)
9 bit adress bus →
Read/Write →
Chip select →

512 X 8 RAM

→ m x n
↔ 8 bit data bus

22/08/17

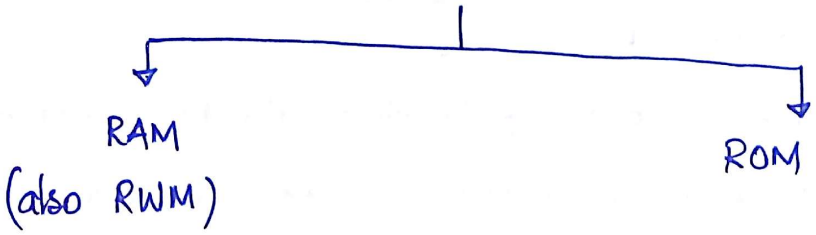
Construction of large memory

→ Heterogeneous
→ Homogeneous

→ ROM is directly connected to the system processor

RAM (application oriented)
→ (i) Horizon expansion (increasing of word length)
→ (ii) Vertical expansion (larger storage space).

MAIN MEMORY



m x m

H (m) — 1111

V (m) — — — —

512 x (2)

512 x (4)

IC required = $\frac{\text{no. of required}}{\text{no. available}}$

$= \frac{4}{2} = 2$

(i) Horizontal :- → (2^n) → address bus

IC1 → 512 x 2

IC2 → 512 x 2

↓

512 + 4 (o/p)

add. bus sorted.

a_0 a_1 a_2 ------ a_8

↓ ↓ ↓

A_0 A_1 A_2 - - A_8

(sorted)

(ii) Vertical :- 1024 + 2

→ 512 x (2)

→ 512 x (2)

512 + 512 = 1024 storage

(2) → output

select line (external)

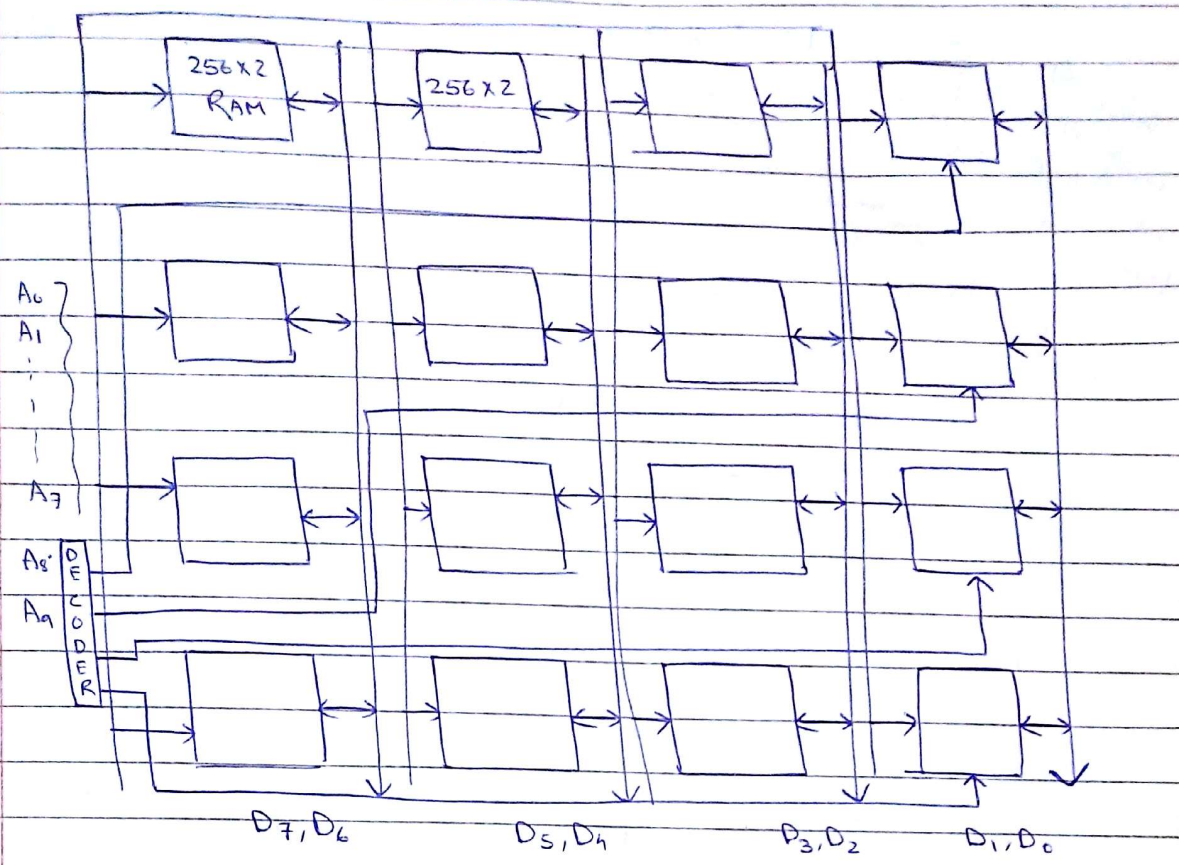
Q. Design a memory of capacity 1024 x 8 using 1 RAM on each RAM of size 256 x 2 ?

→ 1024 x 8

256 x 2

each → (4x4) matrix (thus many ICs)

(P.T.O)



Decoder

1024 → 2^{10}

10 → $A_0 - A_9$

256 → 2^8

8 → $A_0 - A_7$

2:4 i/p : o/p

3:8

xy	D_0	D_1	D_2	D_3
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

→ diagonal.

29/08/17

Q) Design a memory of capacity 1024 x 8, where RAM is of 512 x 8 and ROM also has the capacity of 512 x 8, and show the interconnection diagram between the CPU memory and the memory blocks.

(P.T.O)

When RAM and ROM are same size in question, then we can scale down the RAM size

1024 X 8

RAM → 512 X 8 → 1/4th. → 128X8

ROM → 512 X 8

Same size.

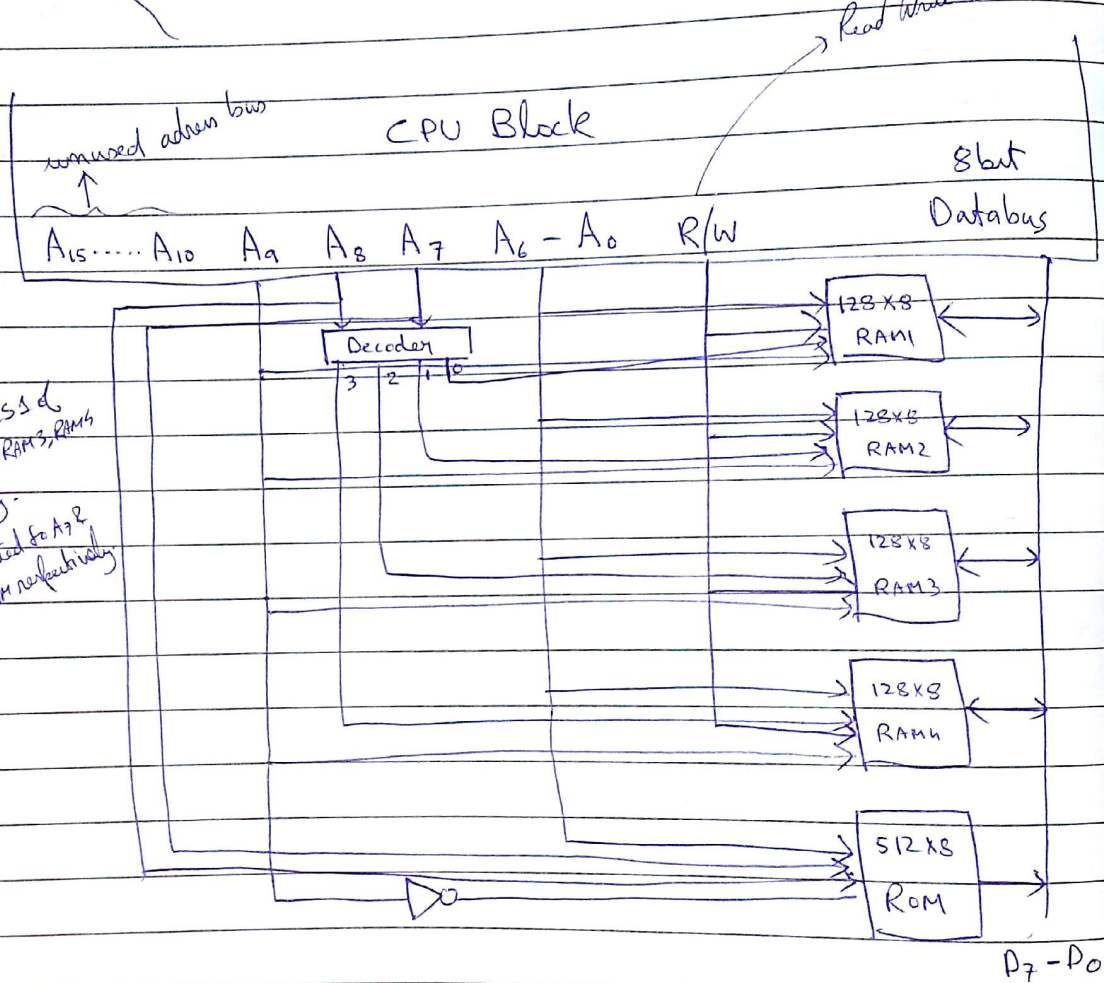
So, we will scale ~~RAM to~~ ~~128 X 8.~~ (1/4th)

4 IC's.

1024 → 2^{10} → $A_0 - A_9$

512 → 2^9 → $A_0 - A_8$

128 → 2^7 → $A_0 - A_6$



0, 1, 2, 3 is connected to CS1 of RAM1, RAM2, RAM3, RAM4 respectively.

A7 & A8 → connected to A7 & A8 of ROM respectively

Close up view of the 4 RAMs

RAM IC

$A_0 - A_6$
R/W
128 X 8 RAM
CS1
CS2.

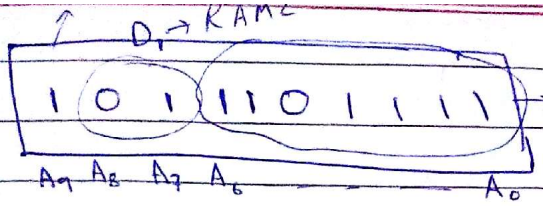
ROM IC

$A_0 - A_6$
A_7 512 X 8
A_8 ROM
CS

Close-up view of the ROM

1st selection :- RAM Vs. ROM.

2nd selection :- 4 RAM



00 → D_0
01 → D_1
10 → D_2
11 → D_3

1.9.17.

The internal organisation of 16X4 memory chip

16 X 4

2^4 → A_0 - A_3

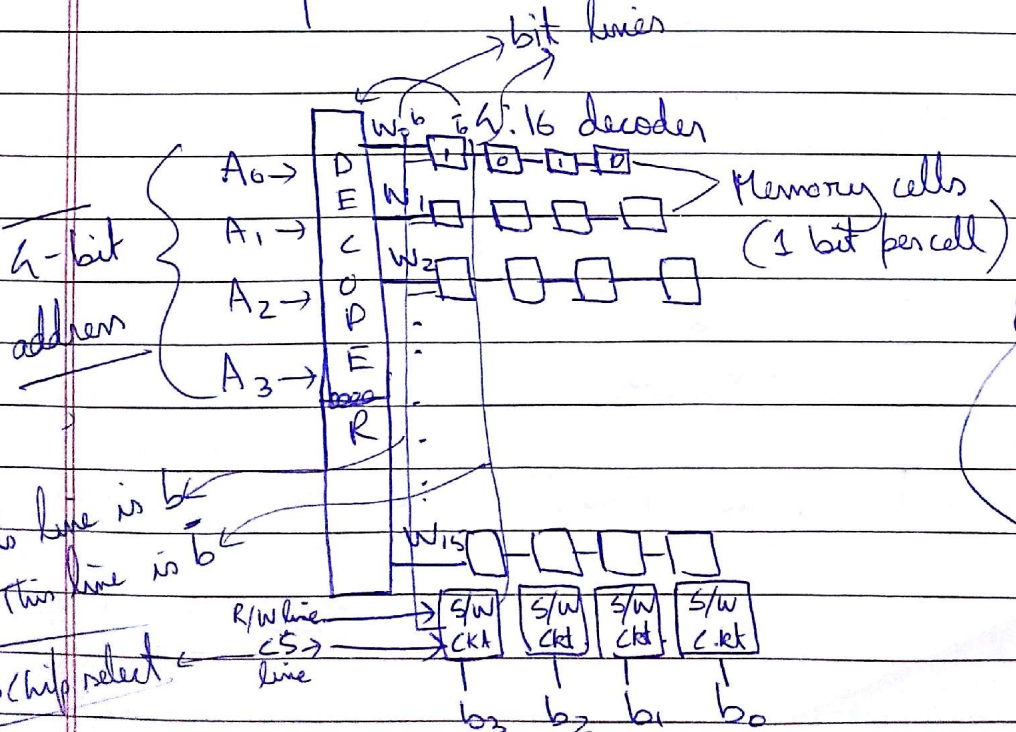
1 0 1 0 Word.

Bits

Lookup

The process of

$A_3 - A_0$	$D_3 - D_0$
0 0 0 0	0 1 0 1
0 0 0 1	0 1 1 1



$W_0 - W_{15}$ → Word line

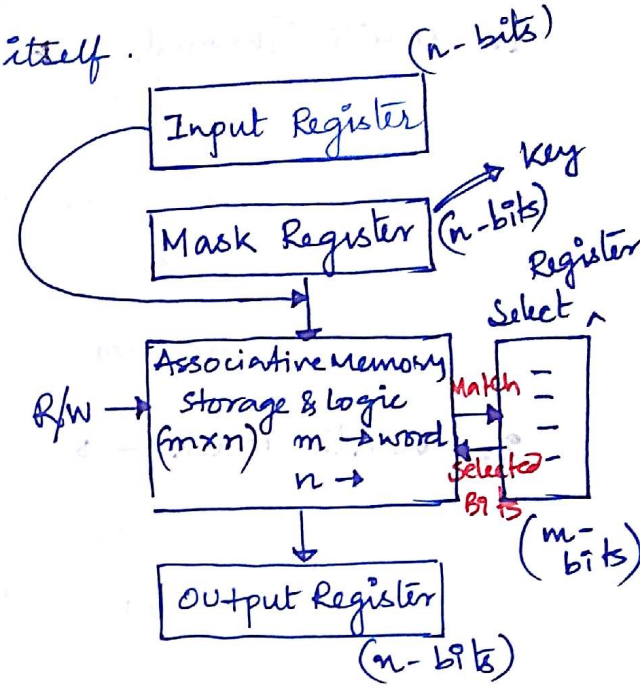
& Sense / Write Circuits is present for each bit column.

b → 1
$\bar{b}$ → 0

12th September, 2017

ASSOCIATIVE

- Data is searched by the data itself.
- Select register stores each bit corresponding to one word in the Associative Memory



Performance of Cache Memory:

cache hit → If the CPU searches for data, and finds it in the immediate cache memory, it is called a cache hit.

cache miss → if a cache hit fails, it is called a cache miss.

$$\text{cache hit ratio } (h_c) = \frac{\text{No. of hits}}{\text{Total no. of CPU searches}}$$

$$= \frac{\text{No. of hits}}{\text{No. of hits} + \text{No. of misses}}$$

cache access time (t_c)

main memory access time (t_m)

Data Writing Policies of Cache

1. Write Through → Updation of the data occurs by the CPU both in the cache and in the main memory both at the same time. i.e, for every iteration of a loop, the value of the variable will be modified.

2. Write Back → Updation occurs continuously in the cache, and only the final value is updated onto the main memory.

Dirty bit →

- For a two-level memory organisation;

$$t_{avg} = \underbrace{(h_c \times t_c)}_{\text{cache}} + \underbrace{[(1 - h_c) \times (t_c + t_m)]}_{\text{main memory}}$$

- For a three-level memory organisation;

$$t_{avg} = \underbrace{(h_c \times t_c)}_{\text{cache}} + \underbrace{[h_m \times (1 - h_c) \times (t_c + t_m)]}_{\text{main memory}} + \underbrace{[(1 - h_c) \times (1 - h_m) \times (t_c + t_m + t_s)]}_{\text{secondary storage memory}}$$

$$\text{Memory efficiency} = \frac{t_c}{t_{avg}}$$

Q1. For a two-level memory subsystem, $t_c = 162$ ns; $t_m = 960$ ns; $h_c = 0.9$; Calculate the total memory access time (average) and efficiency.

Q2. For a three-level memory subsystem having —

$t_c = 15$ ns; $t_s = 80$ ns; $h_c = 0.96$; $h_m = 0.9$. what should be the t_m to acheive the eff. access time of memory as 25 ns

Q3 A memory subsystem has the following specification:

$t_c = 50$ ns; $t_m = 500$ ns;

80% of the memory requests are 'read'.

$h_c = 0.9$ (for read access) [write Through is used]

Calculate t_{avg} for memory for only one read-cycle and find out the total average access time for both 'read' and 'write'.

Formula for

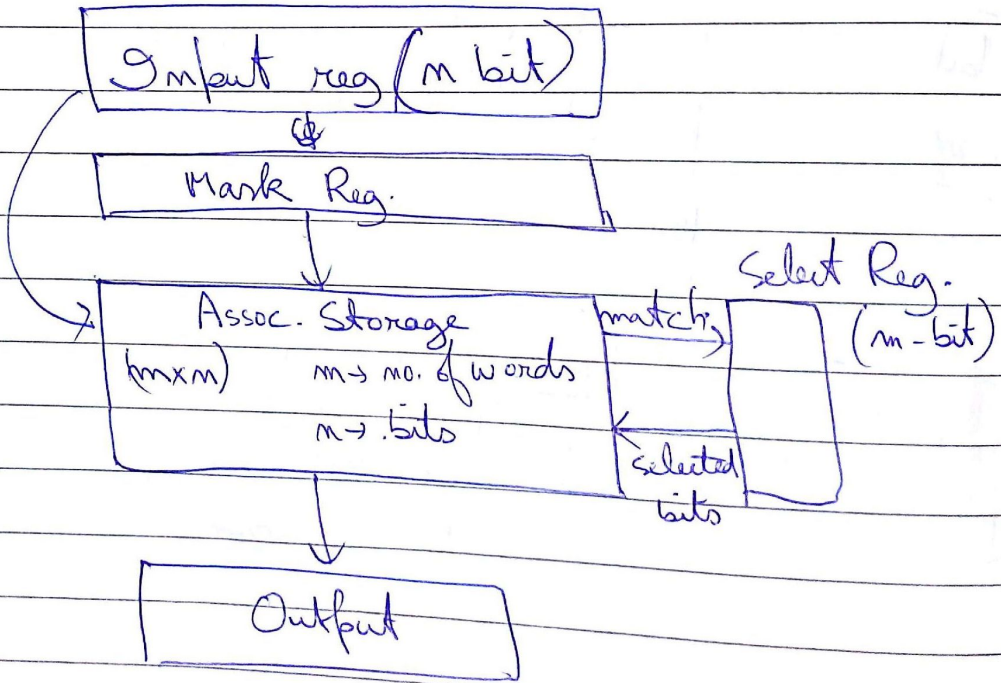
$$t_{avg} = \left[P_r \times t_{avg_{read}}\right] + \left[(1 - P_r) \times t_m\right]$$

Associative Memory :-

A → 3rd
A → 1st

ABC	Input
A	Mask Register / Key
XY	Associative memory (m x m)
XYZ	Selection Register (m bit size of each word)
PQR	Output. ~~register~~

Hardware :-



Input = 10101111 (n)

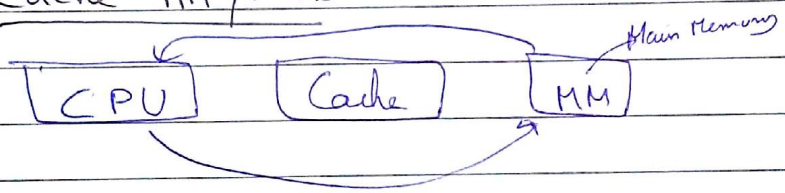
Mask/key = (1111) 0000 (n)

		Select Reg
Word 1 = 11011111	→	0 (no match)
Word 2 = (1010) 1011	→	1 (match)
Word 3 = 00011111	→	0 (no match)

1st 4 bits of words if matches with input's 1st 4 bits, then 1 (match) is assigned to select Reg.

12/7/17

Cache Hit/miss :-



$$\text{Cache hit ratio } (h_c) = \frac{\text{No. of hits}}{\text{Total no. of CPU references}}$$

$$h_c = \frac{\text{no. of hits}}{\text{no. of hits} + \text{no. of miss}}$$

$$80\% = \frac{80}{100} = 0.8$$

- Cache Writing policies :-

(i) Write through
(ii) Write back.

** Modifying bit / Dirty Bit

→ For a 2-level (cache & main memory) memory, the avg. memory access time :-

t_c = cache access time

t_m = main mem. access time

$$t_{avg} = \underbrace{(h_c \times t_c)}_{\text{(Cache)}} + \underbrace{(1-h_c) \times (t_c + t_m)}_{\text{(Main Memory)}} \quad \text{---} \quad (i)$$

→ For a 3-level memory (cache, main mem. & sec memory)

h_m = main memory hit ratio
t_s = sec memory access time

$$t_{avg} = (h_c \times t_c) + \left[(1-h_c) \times (t_c + t_m) \times h_m\right] + \left[(1-h_c)(1-h_m) \times (t_c + t_m + t_s)\right] \quad \text{(ii)}$$

Q1 A 2-level memory subsystem having $t_c = 160$ Nano seconds & $t_m = 960$ Ns. $H_c = 0.9$. Calculate the following:-

(a) avg. access time of memory

(b) Efficiency of memory subsystem $\left(\text{eff} = \frac{t_c}{t_{avg}}\right)$

Q2 A 3-level memory system having $t_c = 15$ Ns, $t_s = 80$ Ns has a cache hit ratio of 0.96 and main $h_m = 0.9$. What shd be the t_m to achieve the effective access time (t_{avg}) of 25 Ns.?

Ns = nano seconds

→ (1) Apply formula (i).

(a) $t_{avg} = (0.9 \times 160) + (1-0.9) \times (160+960)$

(b) "

→ (2) Apply formula (ii)

Q3 A hierarchial cache, main-memory subsystem has the following subsys specifications:-

$t_c = 50$ n.s — 80% of the memory cycle requests are for reading.
$t_m = 500$ n.s
$h_c = 0.9$ for reading. The system uses write through policy for data updation. Calculate the following.

(a) t_{avg} for only 1 memory ~~read~~ cycle. read cycle

Only for VTU exam → Syllabus for Mid Sem 1 (from TK Ghosh)

→ (i) Chapter 4 (excluding mapping)

(b) t_{avg} considering both read & write.

→ $t_{avg(read)} = [(0.9 \times t_c) + (1 - h_c) \times (t_c + t_m)] \times 0.8$

$P_R = 0.8$ (Probability for reading)

$t_{write} = (1 - P_R) \times t_m$.