computer Architecture	computer organization
① computer architechture is concerned with the way hardware componeht are connected together to form system	① computer organization is concerned with structure and behaviour of computer ~~archite~~ system as seen by user
② It acts as interface between hardware & software	② It deals with the component of connection in system
③ while designing the complex system architecture is considered first	③ An organization is done on the basis of architecture
④ computer Architecture deals with high level design issue	④ Computer organization deals with low-level design issue.
⑤ Architecture involves logic(instruction set, addressing mod)	⑤ Organization involves physical components (circuit design, Adders)
⑥ Focus on what a system does	⑥ Focus on How of implementating the system.

Page No.:
Date:
YOUVA

① Hardwired control unit vs microprogrammed control unit.

parameter	Hardware control unit	microprogrammed control unit.
control signal	① Generated using Hardwire	① Gremated using software
Structure	② based on Hard wore so that its rigid	② based on software so that it is flexible.
Processor	③ RISE	③ CISE
Execution speed	④ Fast	④ slow
modification	⑤ Redegining	⑤ Reprogramming
cheap Area	⑥ more	⑥ less
Intruction set	⑦ ~~small~~ less and easy	⑦ more and complex
Pipeline	⑧ small and efficient	⑧ long and Ineffieient
debugging	⑨ Not easy	⑨ easy

Q Difference between ~~Encodecoden~~ and incoder

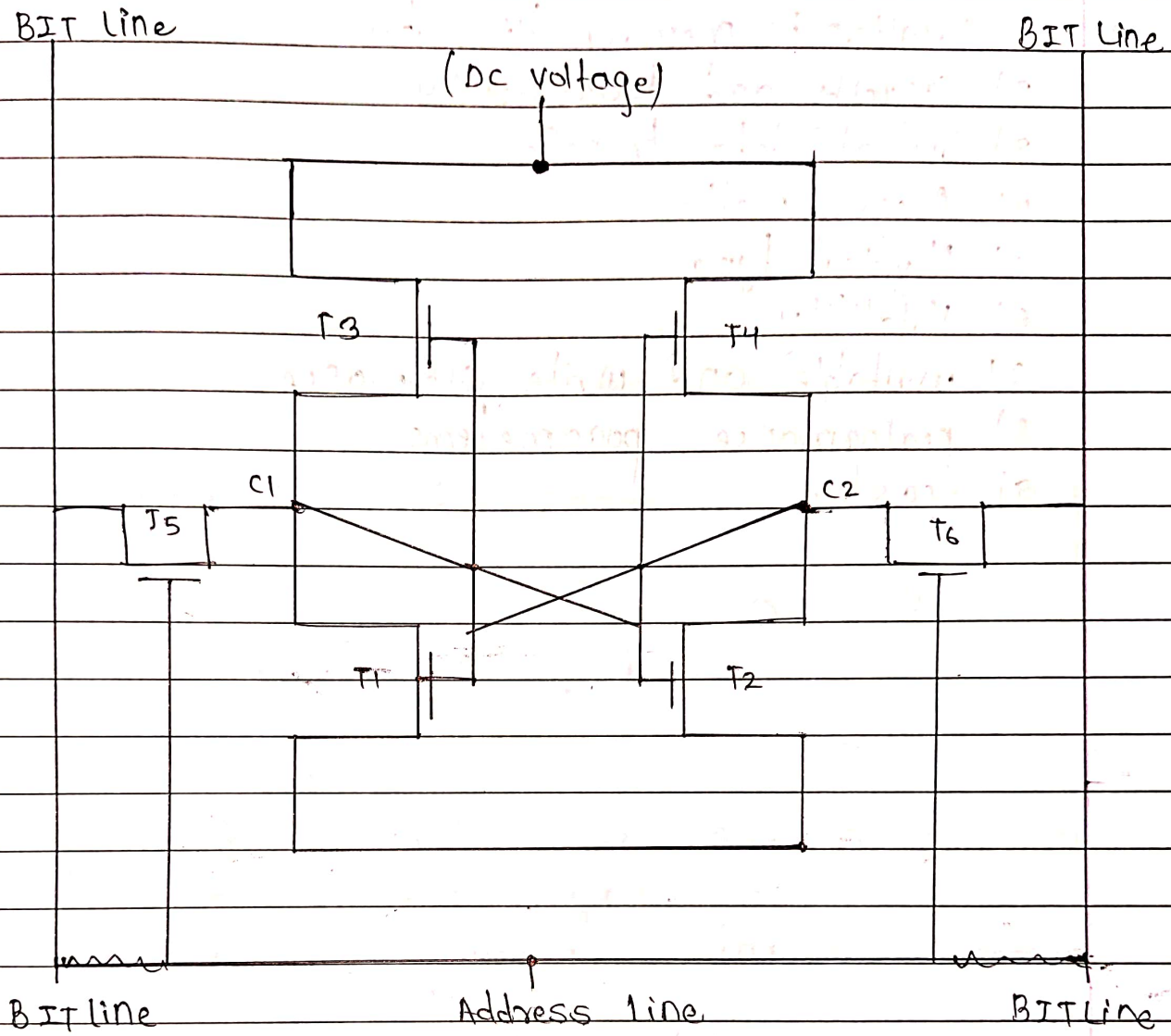
features	Encoder	Decoder
① function	convert ex data into coded or compressed format	convert the coded data into its back to original form.
② Input	original data or Signals	Encoded data or signals
③ output	Encoded & compressed ddta	original ~~and~~ data or signals
④ Direction	from original format to encoded form	from encoded Format to original Form
⑤ Example	~~binary~~ octal to ~~octal~~ binary ~~binary~~ to ~~decimal~~ ~~binary~~ decimal to ~~Hexade~~ binary Hex	Binary to octal Binary to decimal ~~decimal to binary~~ Hex binary to
⑥ Application	Datata compression, signal conversion	Data recovery signal interpretation
⑦ operation	simple	complex.

Page No.:
Date:
YOUVA

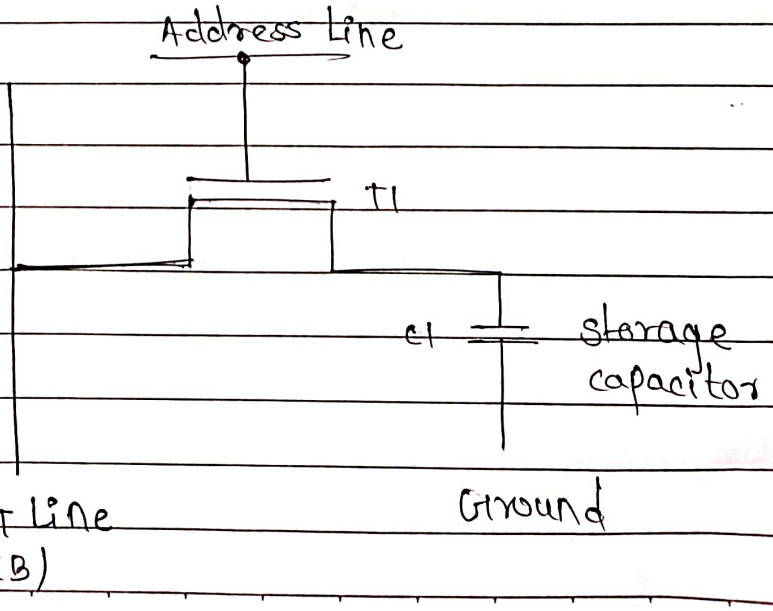
Q

SRAM	DRAM
① SRAM : static random access memory	① DRAM : Dynamic random access memory
② Transistors are used to store information in SRAM	③ Capacitor is used to store data in DRAM
③ It has less storage capacity	⑧ It has large storage capacity
④ SRAM are low density devices	④ DRAM are high density devices
⑤ SRAM used in cache memories	⑤ DRAM are used in main memory
⑥ SRAM is expensive	⑥ DRAM are cheaper
⑦ Power consumption is more in SRAM	⑦ Power consumption is less in DRAM
⑧ SRAM structure is complex than DRAM	⑧ DRAM structu is simplex than SRAM
⑨ Not need periodic refreshment to maintain data	⑩ It required periodic refreshment to maintain data
⑩ SRAM is faster	⑩ DRAM is slower

SRAM Diagram :



DRAM Diagram :



Q List and Explain characteristics of memory

→
1) location in memory Hierachy
2) capacity and addressability
3) Unit of data transfer
4) Access method
5) Physical type
6) Volatility
7) writable and write sublenance
8) performance parameters
9) cost.

There are other physical types of memories such as Magnetic Access/ Recording type (ex. Hard disks) or Optical Access/Recording type (ex. CD, DVD, Blu-Ray Disks)

(f) Volatility

It is the characteristics that identifies the capacity of the memory to retain or hold its data contents when the power to memory is turned off. Based on this parameters we find the memories to be :

- **Volatile :** The memory that loses its data contents of its locations (or not able to retain) is called as Volatile Memory.
- **Non-Volatile :** The memory that holds on to the data contents of its locations (or is able to retain) is called as Non-Volatile Memory.

In case of Non-volatile memories, data can be retained of a specific amount of time. This property of Non-volatile memories is called as **Data Retainance**. For example EEPROM is said to have Data Retainance of 10 years.

(g) Writable and Write Cycle Sustenance

Not all memory technologies allow writing Data to the memory or writing data multiple times to the memory. Therefore, Non-volatile memories, usually ROMs are classified based on whether they are writable or not. They are referred to as Read Mostly memory if they are Writable. Such memories show different number of Write cycles that they can sustain. This property is called as Write Cycle Sustenance. For Example, OTPROMs can be written only once where as Flash ROM can be written 100,000 times.

(h) Performance Parameters

These characteristics are the parameters on which the performance of the memory is determined and measured. These parameters are discussed in detail, in the next section 10.6.

(i) Cost

It's a commercial characteristic of the memory and determines its implementation feasibility. This aspect is discussed in detail, in the next section 10.6.

These characteristics are :

a. Location in Memory Hierarchy
b. Capacity and Addressability
c. Unit of Data Transfer
d. Access Method
e. Physical type
f. Volatility
g. Writable and Write Cycle Sustenance
h. Performance Parameters
i. Cost

(a) Location in the Memory Hierarchy

It deals with the location of the specified memory in the memory hierarchy of the computer system. There are different possible locations for the memories :

- **CPU :** CPU consists of many CPU registers which is the Processor's own private memory storage space for storing most required or hand elements of data references, instruction references and addresses. Many times, it is referred to as L0 cache of the system.
- **Cache Memory :** The high-speed memory that is introduced in the system organization with processor proximity and having the intention of reducing the average wait-states in the CPU machine cycles, is called as Cache Memory. Generally high-speed memories qualify to become cache memories. Currently up to 3 levels – Level-1 (L1), Level-2 (L2) and Level-3 (L3) – of cache memories are used in the computer systems.
- **Main Memory :** It is the bulk memory of the system. All the live data and program (instructions) elements are stored in the main memory. Main memory and Cache memory are directly addressable by the processor or CPU and therefore form the Primary Memory of the system.
- **External or secondary :** It comprises of secondary storage devices like hard disks, magnetic tapes. The CPU doesn't access these devices directly. It uses device controllers to access secondary storage devices.

(b) Capacity and Addressability

The capacity of the memory or memory device is represented in terms of: i) Word size and ii) Number of words or Addressability.

- **Word size :** Words are expressed in terms of no. of bits or bytes (8 bits). A word can however mean any number of bytes. Commonly used word sizes are 1 byte (8 bits), 2 bytes (16 bits) and 4 bytes (32 bits) or 8 bytes (64 bits). It is also called as the data width of the memory.
- **Addressability or Number of words :** This specifies the number of words available in the particular memory or how many words that are addressable in the memory.

Total capacity of the memory is Word Size X No. of Words. For example, if a memory device is specified as 8K x 16. This means that the device has a word size of 2 bytes (16 bits) and a total of 8192 (8K) words in memory.

(c) Unit of Transfer

It is the maximum number of bits that can be accessed (read or written into) by the memory at a time. In case of main memory or memories used as primary memory, it is generally same as its word size. In the case of external memory, unit of transfer may not be same as that of its word size. It is usually larger than the word size and it is referred to as blocks.

(d) Access Methods

It is the fundamental characteristic of the memory or memory device. It is the method or order in which memory elements can be accessed. There are three types of access methods:

1. **Random Access :** If storage locations in a particular memory device can be accessed in any order i.e. The access time of the memory is independent of the memory address or location being accessed. Any location in such memory requires exactly same to access. This type of memory is said to have a **Random Access** mechanism. Usually, most of the semiconductor memories and primary memories are Random Access in nature.
2. **Sequential or Serial Access :** If memory locations in the memory device, can be accessed only in a specific pre-determined sequence or order, then the access method is called as **Sequential or serial access.** Many secondary storage devices like magnetic tapes exhibit Sequential access to its contents.
3. **Semi random or Pseudo Random Access :** Memory devices such as Magnetic Hard disks and Optical disks use this type of access method. Here each disc surface has an independent read/write head thus each surface can be accessed randomly but access within the surface is sequential in nature as tracks and sector arranged can be accessed in serial order. Therefore the access is not fully or truly random in nature. This type of access is called as **Semi random or Pseudo Random Access.**

Module 5

(e) Physical type

This type is essentially based on the technology by which the memory is constructed. The memories implemented with different IC technologies are Semiconductor memories. Semiconductor memories are further classified based on their Write characteristics as RAM – Random Access Memory (usually of Read/Write nature) and ROM – Read Only Memory.

erased like EPROM. It is Therefore, an intermediate between EPROM and EEPROM. Though individual bytes cannot be erased a section of memory cells can be erased at a time without affecting the contents of the entire memory.

☞ Comparison of DRAM and SRAM

UQ. 10.3.2 Differentiate between SRAM and DRAM. **MU - Q. 5(b)(ii), Dec. 17, 5 Marks**

Table 10.3.2 : Comparison of DRAM and SRAM

Observation	DRAM	SRAM
Memory cell construction	Simple and small	Complex and large
Cell Density (No of cells per unit given area)	More	Less
Cost	Less expensive	Costlier as compared
Refresh circuitry	Required	Not Required
Speed	Slight slower as compared	Faster as compared
Usage	Main memory	Cache

⏭ 10.4 MEMORY HIERARCHY

UQ. 10.4.1 Describe the memory hierarchy in the computer system.
MU - Q. 1(c), May 18, Q. 1(D), Dec. 18, 5 Marks

UQ. 10.4.2 Explain in details Memory Hierarchy with examples.
MU - Q. 4(a), May 14, 6 Marks

- The performance of the general purpose computers used mostly in day-to-day life can be improved by providing additional storage beyond the main memory. Single memory is not enough to store all the programs and continuously generated data. Also, not all the data and programs are required simultaneously.
- Single memory implementation technology cannot support the system requirements of high storage and faster execution. Hence, a combination of memory technologies can be used to store active programs and data together in faster smaller memories, whereas, save the rest of the data in low cost slightly slower huge memory devices. This technique is termed as **memory hierarchy**.
- The faster smaller memories consist of registers, cache and main memories. Magnetic disks, CDs, DVDs, tapes, etc. form the auxiliary or low cost slower memories. A typical memory hierarchy pyramid is shown in Fig. 10.4.1.

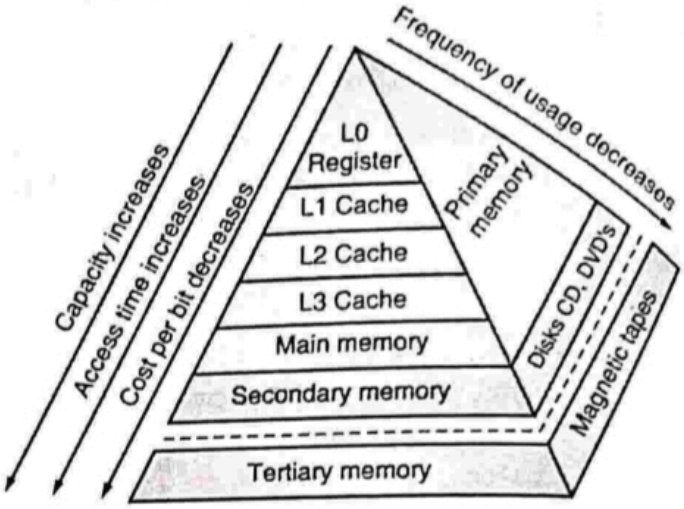


Fig. 10.4.1 : Memory Hierarchy or Organization

- As one goes down the hierarchy, following observations occur :
  1. Increase in the capacity.
  2. Decrease in the cost per bit.
  3. Increase in the access time.
  4. Decrease in the frequency of usage of the memory by the processor.
- Thus, smaller, costlier, faster memories are supplemented by bigger, cheaper, slower memories. This is termed as memory hierarchy. The key to success for the hierarchy is that the frequency of access also reduces as one goes down the pyramid.

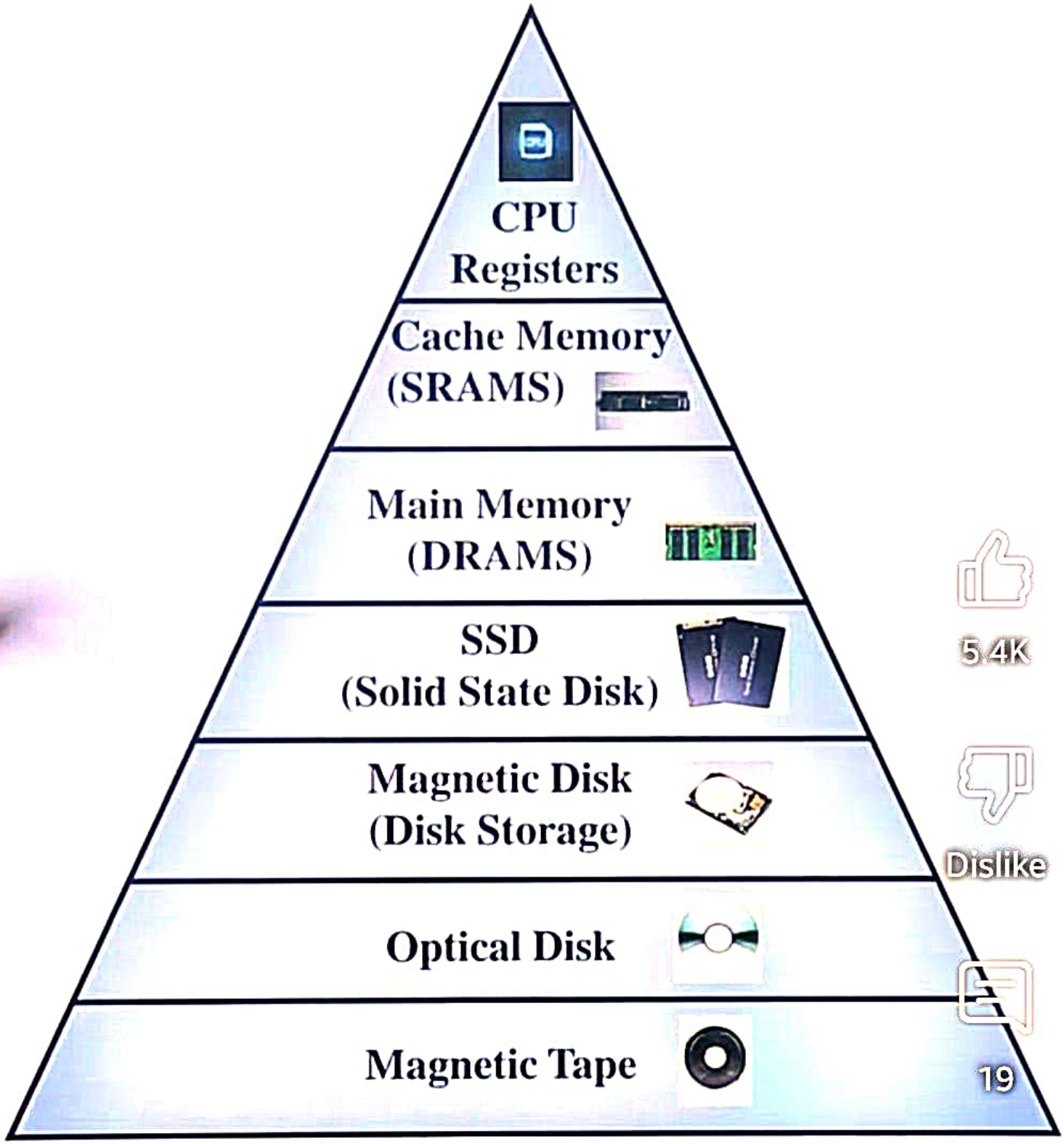
⏭ 10.5 MEMORY CHARACTERISTICS

UQ. 10.5.1 List different memory organization , characteristics.
MU - Q. 3(c), May 14, 8 Marks, Q. 1(c), Dec. 15, Q. 1(a), Dec. 17, 5 Marks

UQ. 10.5.2 Describe the characteristics of Memory.
MU - Q. 4(b), Dec. 16, 10 Marks

UQ. 10.5.3 Write notes on the characterizes of memory. **MU - Q. 6(b), May 16, 8 Marks**

The Memories or Memory devices are measured for their features and their performance based on different characteristics.



5.4K
Dislike
19
Selection
Write
Eraser
Clean
Ruler
Shape
Form
Mind map
Note
Move
Grid
Share

Q Explain memory mapping techniques

→ ① Direct mapping

- How it works :
  - memory is divided into blocks, and the cache is divided into the lines
  - Each block of memory is mapped to exactly one cache line using the formula.

cache line = (Block address) mod (Number of cache lines)

  - If another block maps to the same line, it replaces the existing block.

- Advantages :
  - simple, easy to implement and inexpensive
  - fast because the mapping is direct
  - requires less hardware for implementation

- Disadvantages :
  - High conflict misses because multiple blocks can map to the same cache line.
  - Not efficient for for program with frequent overlapping block access.

- System Use case :

Systems with predictable memory access pattern, such as embedded systems.

- **Diagram :**

at a time only one block number present in cache line

0	0 2 4 6
1	1 3 5 7

(8 words)
(capacity)

0	0 1 2 3
1	4 5 6 7
2	8 9 10 11
3	12 13 14 15
4	16 17 18 19
5	20 21 22 23
6	24 25 26 27
7	28 29 30 31

(32 words capacity)

- Number of blocks = $\frac{32}{4}$ = 8 blocks
- Number of cache lines = $\frac{8}{4}$ = 2 lines
- Block = 4 words
- **formula :** cache lineno : $\left(\frac{\text{main memory}}{\text{block}}\right)$ mod (No. of lines in cache memory)

$\therefore$ (0) mod 2
= 0

$\therefore$ ~~(0) mod (2)~~ $\therefore$ (1) mod (2)
~~= 0~~ = 1

- Physical address

5 bit		
2 bit	1 bit	2 bit
Tag	Line no	Boffset

Date:

② Fully Associative mapping :-

* How It works

- memory blocks can be loaded into any line of the cache
- Each cache line has tag that stores the memory block number
- when accessing data, the tag of every cache line is searched to find the required block
- If the cache is full, a replacement polacy (eg. least recentaly used) determines which block to replace.

* Advantages :

- Eliminates conflict misses because any block can go into any line
- maximizes cache utilization and improve cache hits rates.

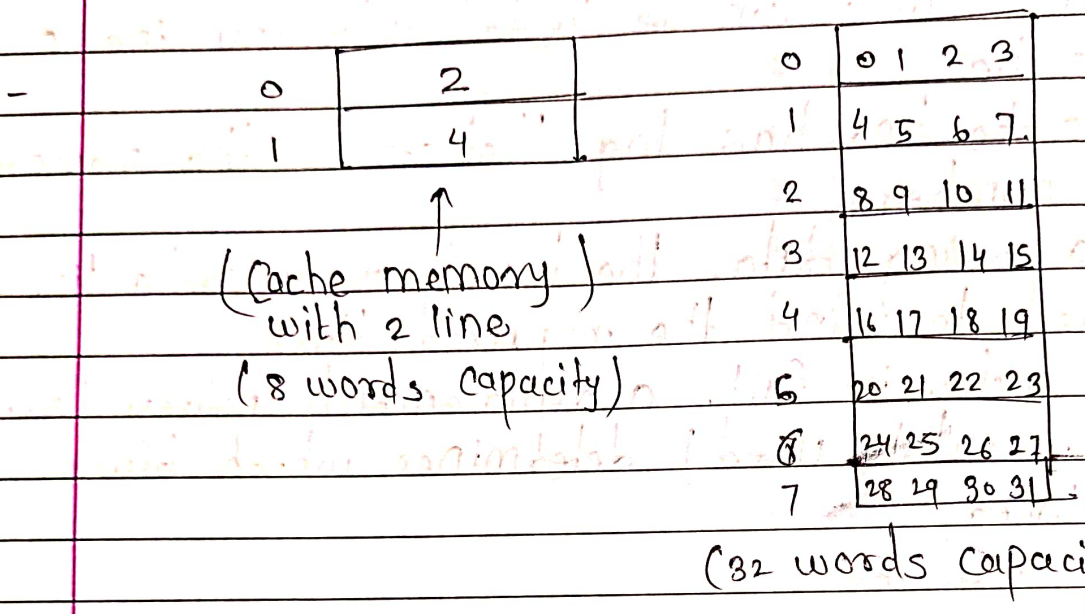
* Disadvantages :

- Expensive
- Complex dues to need of to seach all caches lines
- slower compared to direct mapping due to tag maching process

* Use case :

- High performance systems, such as servers or processors handling unpredictable workloads.

- Diagram : eg.



- (2) 0 1 0 | 8 9 10 11

(4) 1 0 0 | 16 17 18 19

tag field — blocks consists data.

- cache hit condition :

0 1 0 | 1 1

checking with tag field — content in block

- cache miss condition.

1 0 1 | 0 0

checking with tag field — content in block

- Physical address :

5 bit

3 bit	2 bit
Tag	Block offset

③ Set- Associative mapping

* How it works

- The cache is divided into sets, with each set containing multiple lines
- A block of memory maps to specific set using the formula

Set Index = (Block address) mod (Number of sets)

- within the set, a block can occupy any line. A tag is used to identify the block
- when the set is full, a replacement policy determines which block to evict.

* Advantages :

- Reduces conflict misses compared to direct mapping
- Balances the simplicity of direct mapping and flexibility of fully associative mapping.

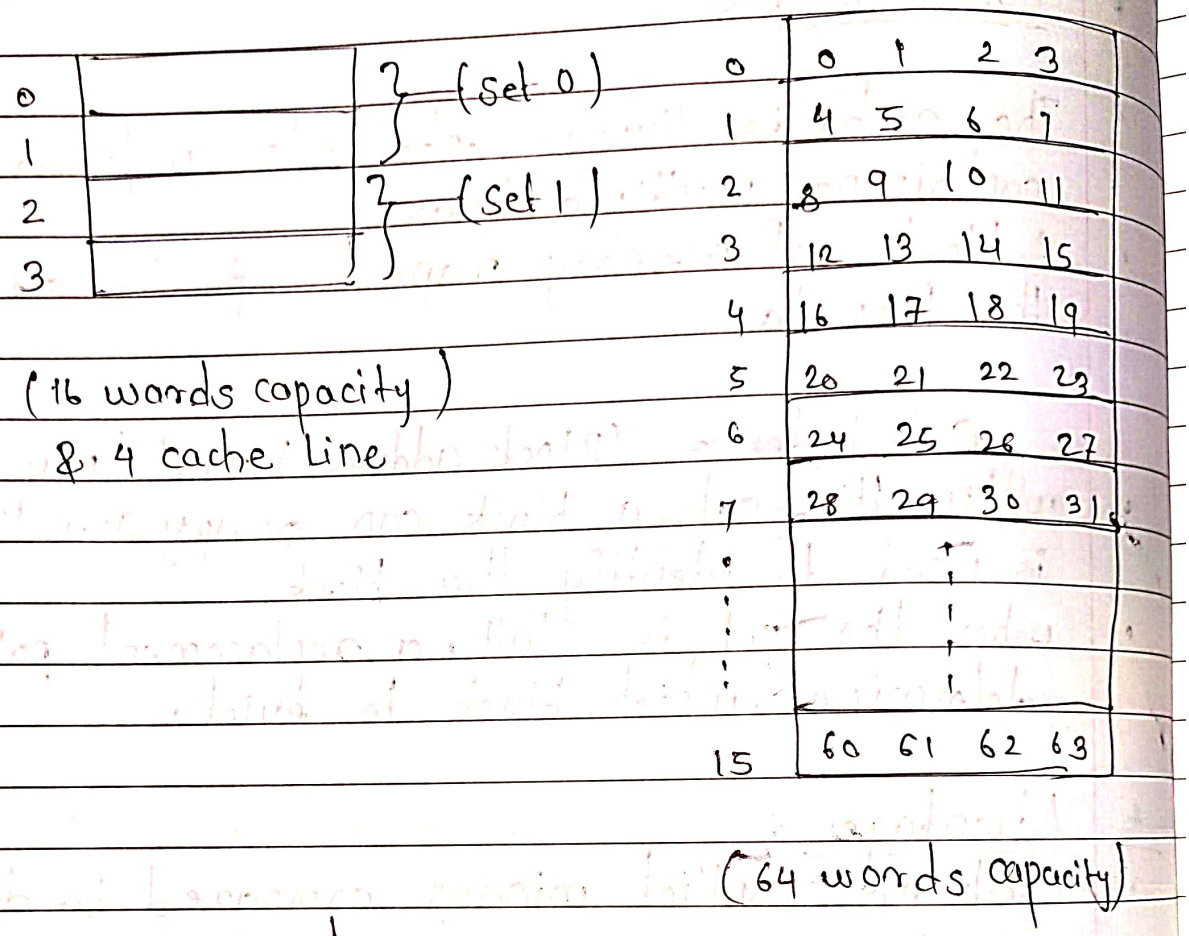
* Disadvantages :

- slightly more complex than direct mapping
- Requires additional logic for set management and replacement policies.

* Use case :

- common in modern CPU for balancing performance and cost.

* Diagram : eg.

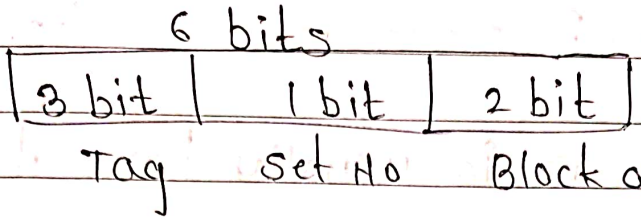


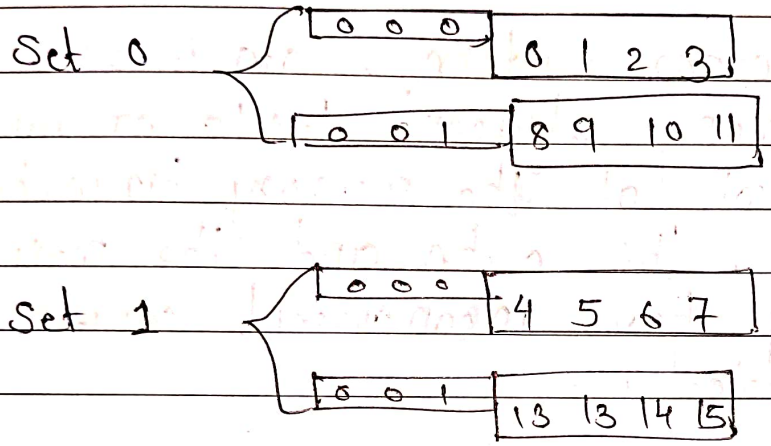
- Blocks = 4 words
- Number of blocks = $\frac{64}{4}$ = ~~4 Blot~~ 16 Blocks
- Number cache line = $\frac{16}{4}$ = 4 lines
- 2way set associativity we

0 mod 2 = 0
1 mod 2 = 1
2 mod 2 = 0
3 mod 2 = 1
4 mod 2 = 0

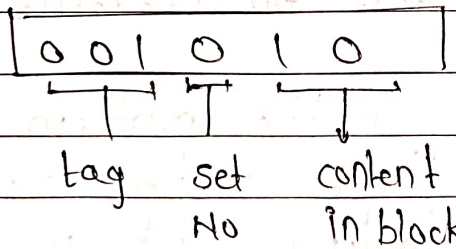
Date: YOUVA

- physical address :

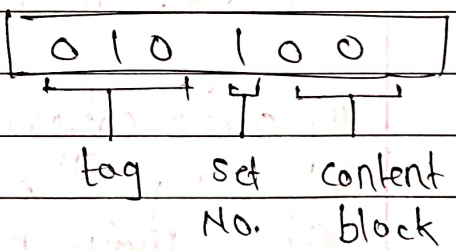




* cache hit condition



* cache miss condition



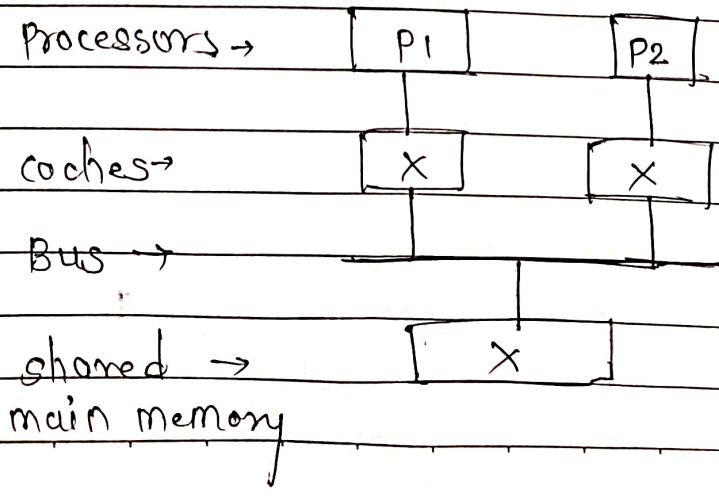
(4) ~~Segment mapping~~
Q Cache coherence

→ 1) for higher performace in a multiprocessor sysmem
- each processor is usually have its own cache

2) In a multiprocessor system, data inconsistenty may occur among adjacent levels or within the same level of the memory hiraachy
- for example, the cache and the main memory may have inconsistent copies of the same object.

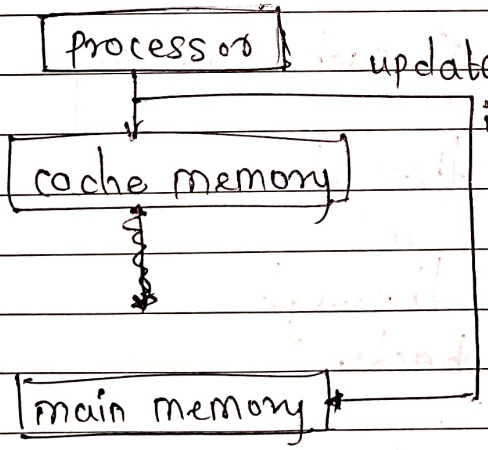
3) As multiple processor operate in parallel, and independentaly multiple caches may process different copies of the same memory block, this creates cache ~~he~~ coherence problem

4) cache coherence refers to the problem of keeping the data in these caches consistent. The main problem is dealing with writes by a processor.

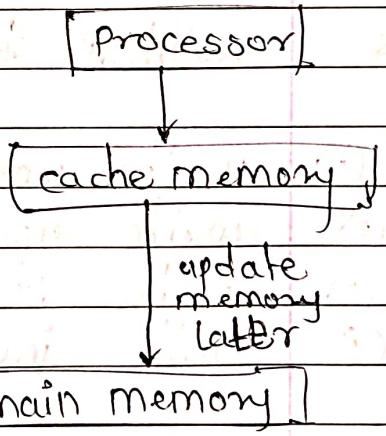


Cache write policies

- **write through**: In the write through policy, all data written to the cache is also written to memory at the same time (ensuring that main memory is always valid).
- **write back**: In the write back policy, when data is written to a cache a dirty bit is set for the affected block. The modification block is written to memory only when the block is replaced



(write through)



(write back)

* Solution for coherance of cache memory

1. write update - write through
2. write update - write back
3. write invalidate - write through
4. write invalidate - write back.

1. write update - write through
   1. processor writes new data to its cache
   2. for write update : the updated data is brodcast to other caches, which update their copies
   3. for write throug : The updated data is immediately written to main memory

(2) write update and - write back

(1) processor writes new data in its caches
(2) write update : the updated data is brodcast to other caches
caches holding the same data update their copies
(3) write Back : the updated data is written to main memory only when the cache block is evicated (Not immedielly)

(3) write invalid & - write through :-

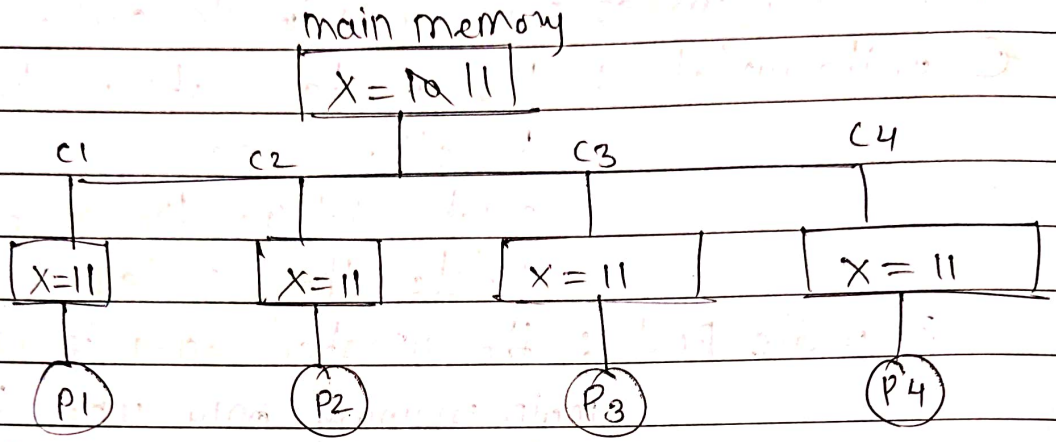
(1) Processor writes new data to its cache
(2) write Invalidate : The cache sends an invalidation message to other caches holding the same data
. other caches invalidate their copies of the data
(3) write through : the updated data is immediately written to the main memory
(4) consistency check : main memory now contains the latest value, ensuring synchronization.

(4) write invalid - write back :-

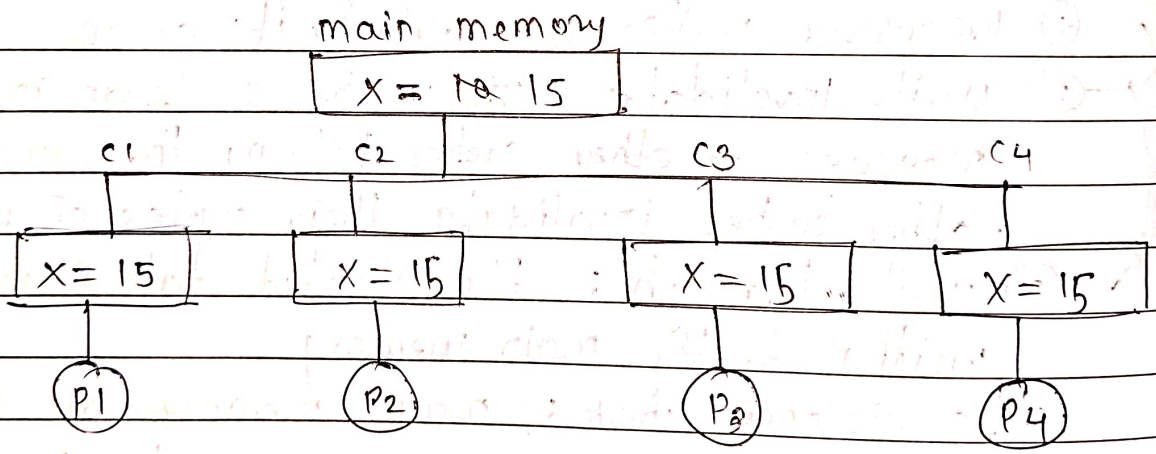
(1) processor write new data to its cache.
(2) write invalidate : the cache sends an invalidation message to other caches holding the same data, other caches invalidate their copies of data.
(3) write back : the update data is written to main memory only when the modified cache block is evicted.
(4) cosistent check : cache coherance protocol ensures synchronization between caches and main memory

(Diagrams for 4 solutions)

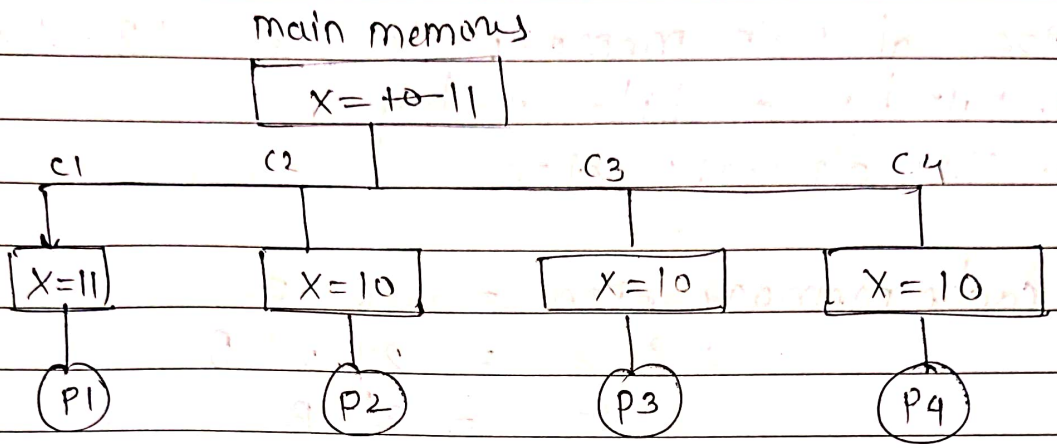
(1) write update - write through



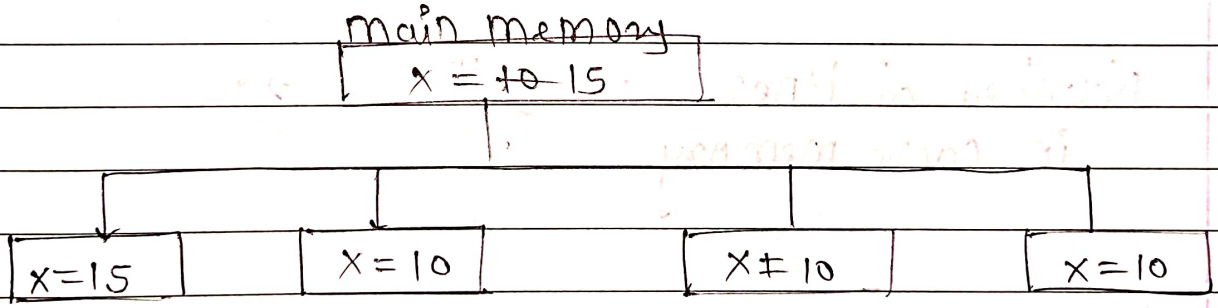
(2) write update - write back



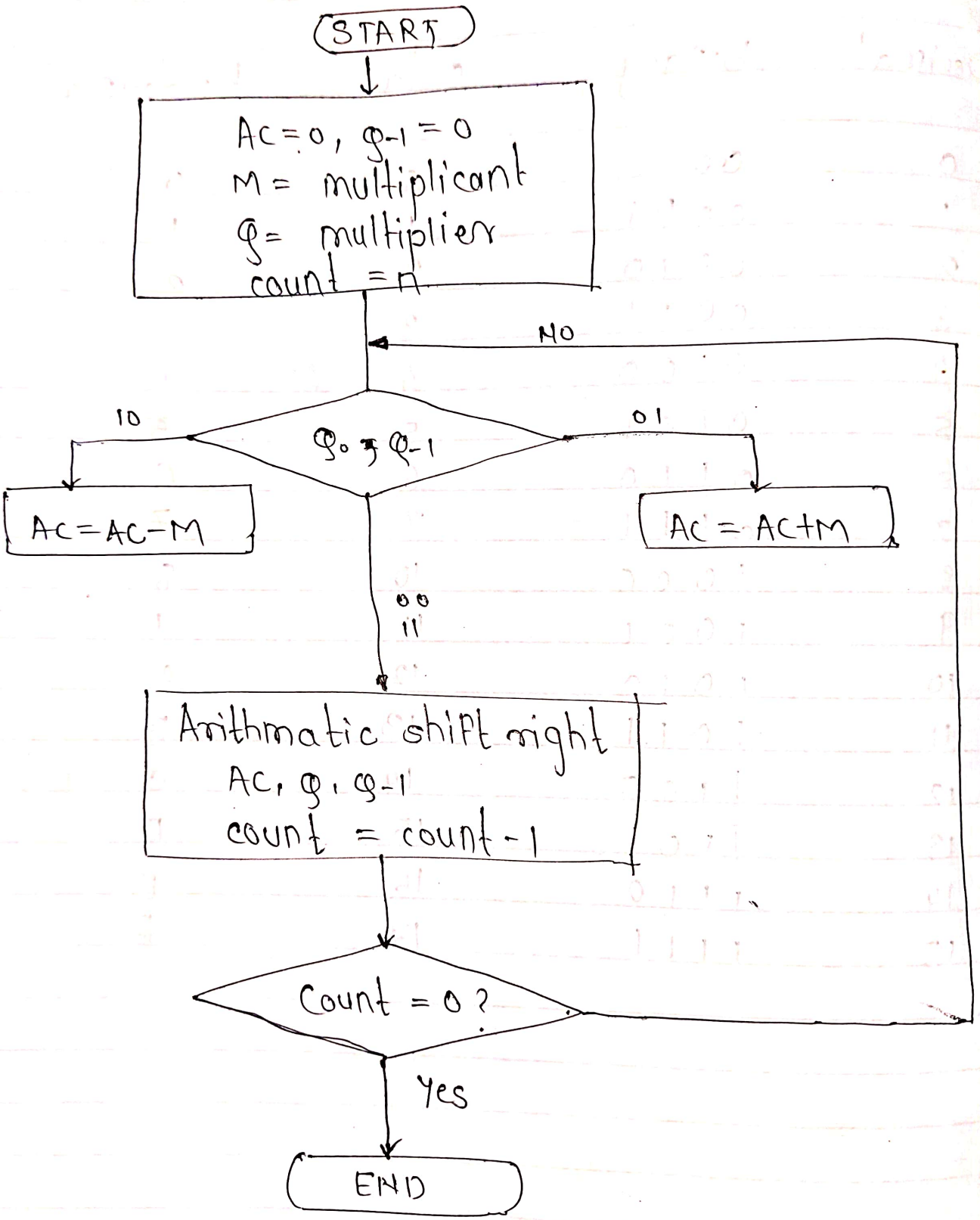
(3) write invalidate - write through



(4) write invalidate - write back

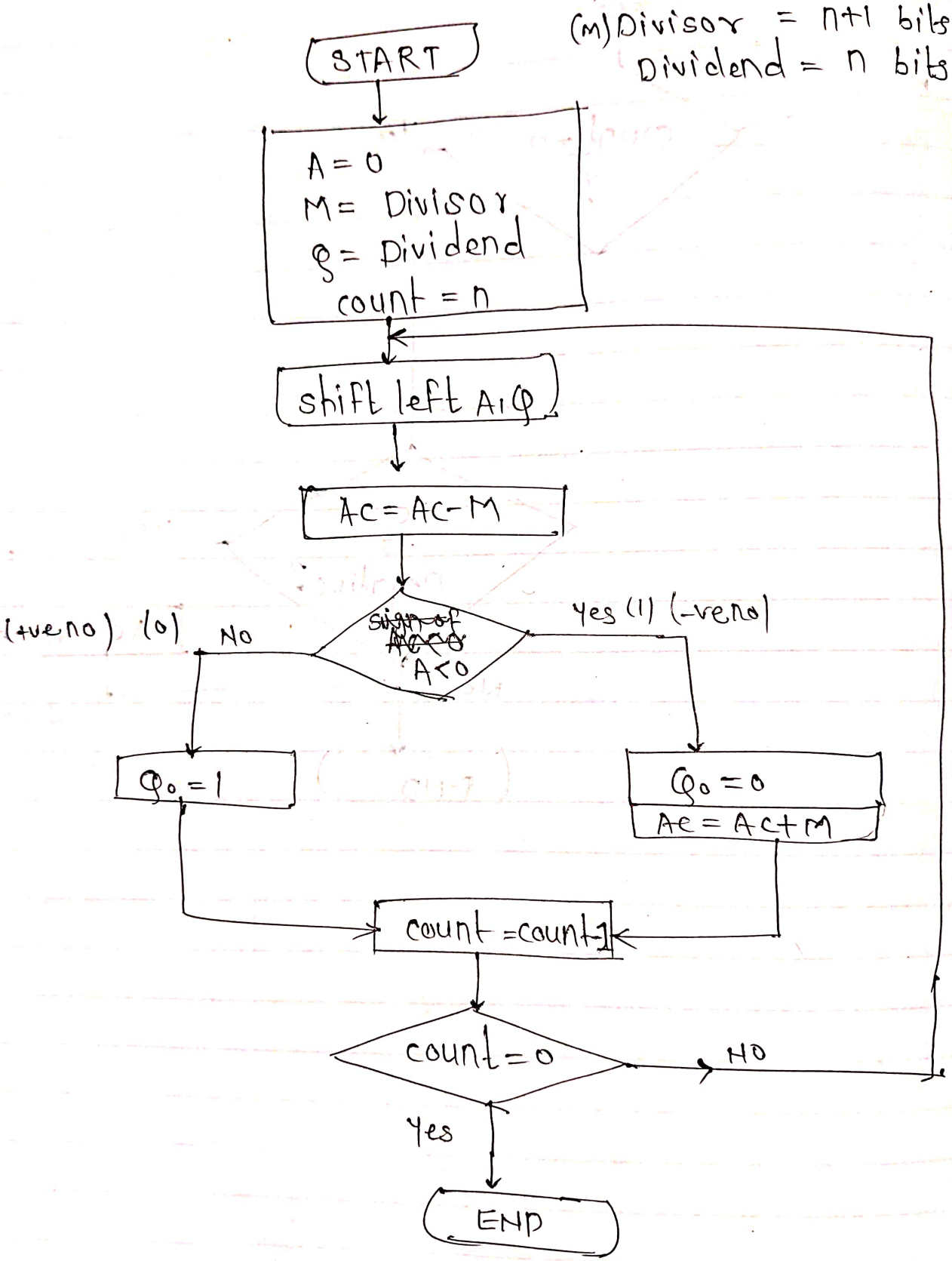


Booths algorithm



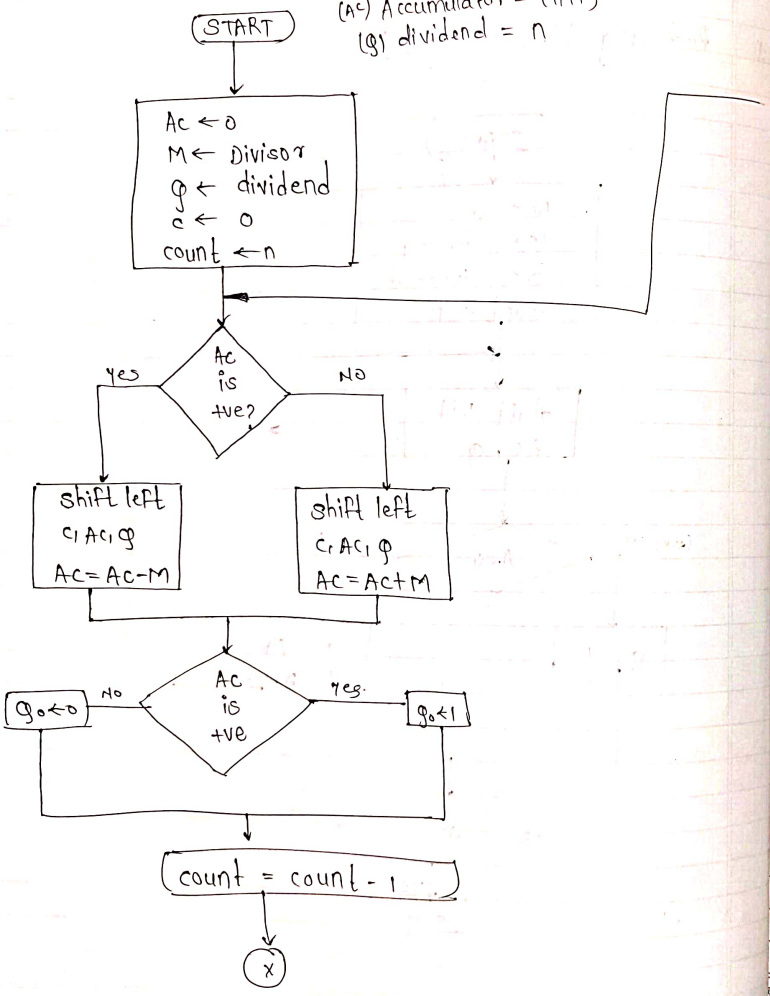
Restoring & Non-Restoring algorithm

Accumulator = n+1 bits
(M) Divisor = n+1 bits
Dividend = n bits



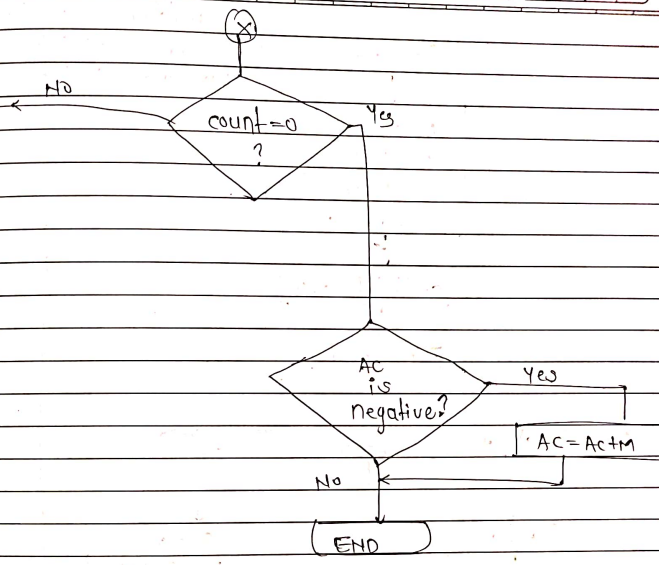
Non-Restoring Division algorithm :-

(M) Divisor = (n+1)
(AC) Accumulator = (n+1)
(Q) dividend = n



EXPERIMENT : No.

Page No.
Date



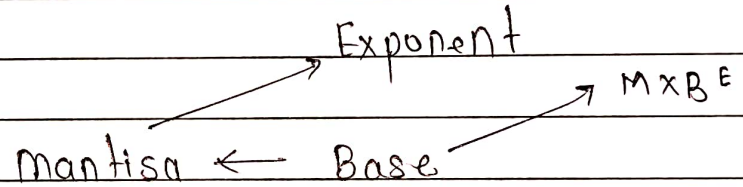
Teacher's Sign. :

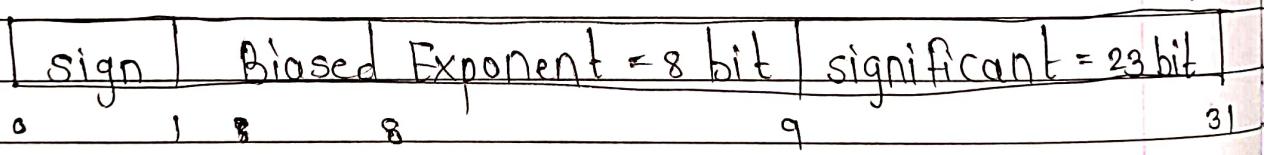
IEEE 754 standard Floating point representation

single & double Precision also.

→ IEEE standard for floating point arithmatic (IEEE - 754) is technical standard for floating point computation.

- Formal Representation :
Three numbers are associated with a floating point numbers
1) A mantisa (M)
2) Exponent (E)
3) Base (B)





In typical representation of floating point number exponent is biased and mantisa is normalised.

a) Biased exponent : The exponent fields needs to be represent both positive and negative exponent is represent using an excess - 128 code Adding 128 in the biased exponent.

b) Normalize mantisa :- A binary floating point number is represented in a normalized form, that is number is in form of ± 0 (significant starting with non-zero bit) x 2 ± (exponent value)

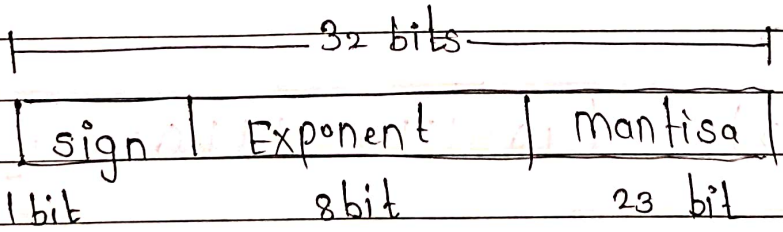
For example,

Binary number = 11.101
Normalized form = 11101 x 2^{-3}

Number	mantissa	Base	Exponent
3×10^{6}	3	10	6
110×2^{8}	110	2	8
6132.784	6132	10	-3

- single precision : (32 bits) (128)

format :- $(1.N)\ 2^{E-127}$



- Double precision (64 bits) (1024)

Format :- $(1.N)\ 2^{E-1023}$

64 bits

sign	Exponent	mantissa
1 bit	11 bit	52 bits

Q Explain half and full adder with truth table.

① Half Adder (2 bit Adder) :

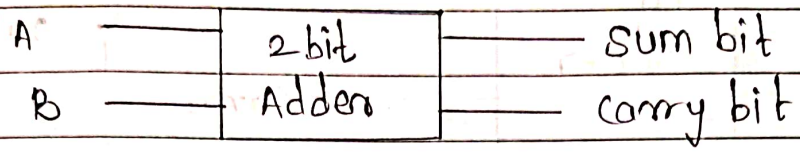
↳ combination of circuit (logic gates)

A	B	output bit (Sum bit)	Carry bit
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

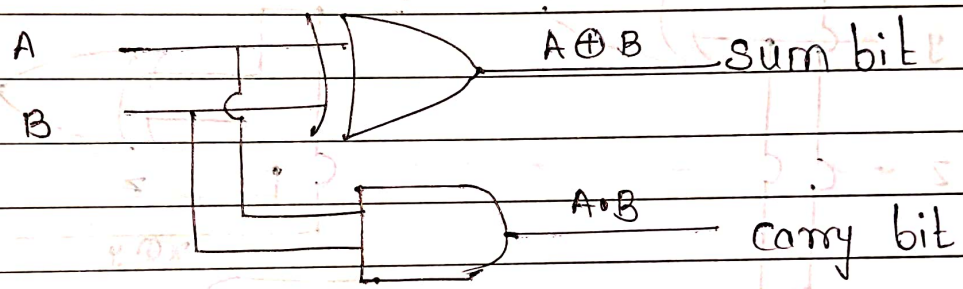
for sum bit we use an (X-OR) logic gate

for carry bit we use (AND) logic gate.

* Block diagram :



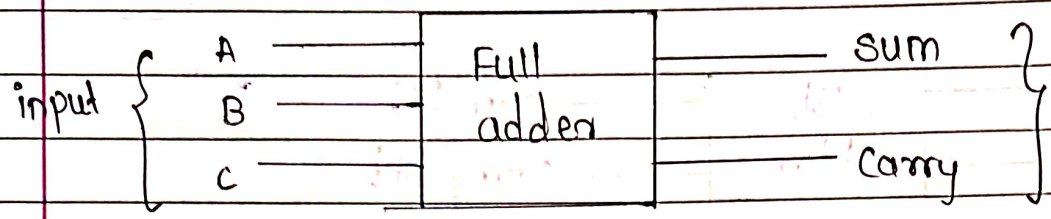
* Circuit diagram : (Logical diagram



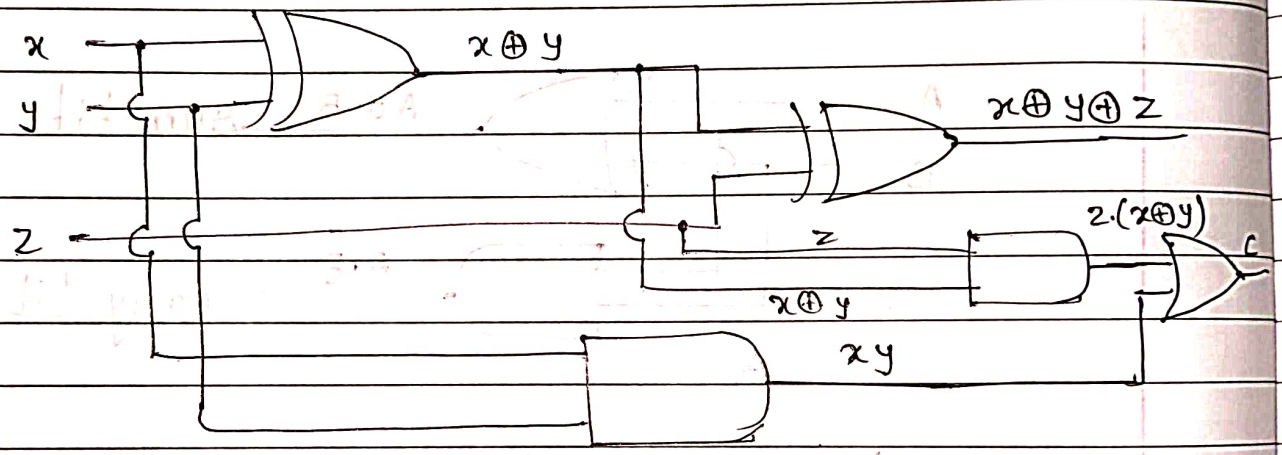
Full Adder (3 bit adder) :

Inputs			outputs	
A	B	C	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

* Block diagram:



* logical diagram :-



Defination: A combination of logical circuit that computes the addition of three binary digits is called as full adder

- ~~the addition of three binary digits. ie~~

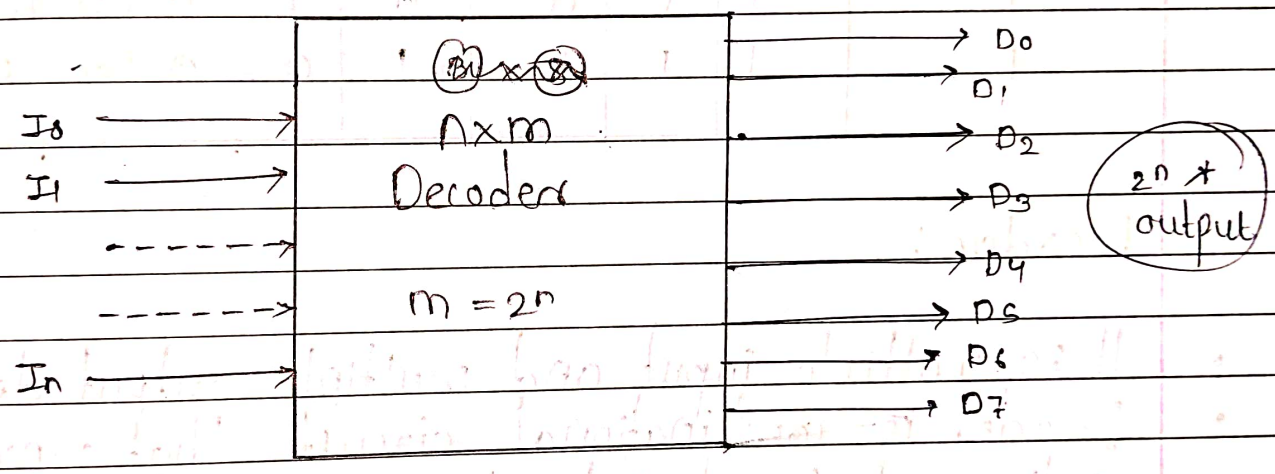
* at a time only one output is active

M T W T F S S
Page No.:
Date:
YOUVA

Decoder :-

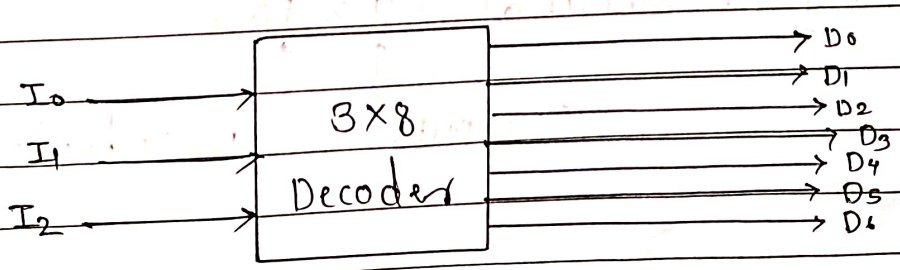
* It is a combinational circuit.
  ↳ Logical circuit.
* It has n input and 2^{nd} 2^n output.
* At a particular time one output line is active

* Block diagram :



n input

* Example Diagram :



Enable	Inputs			Outputs							
E	A_2	A_1	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Decoder:

- It is multiple input and multiple output device
- Decoder is combinational circuit that converts n lines of input into the 2^n lines of output.
- Application of Decoders are converting binary code to other code like
  - Binary to octal (23)
  - Binary to Hexadecimal (24)
  - Binary to decimal (4×10)

④ inputs ↑ ↑ ⑩ outputs

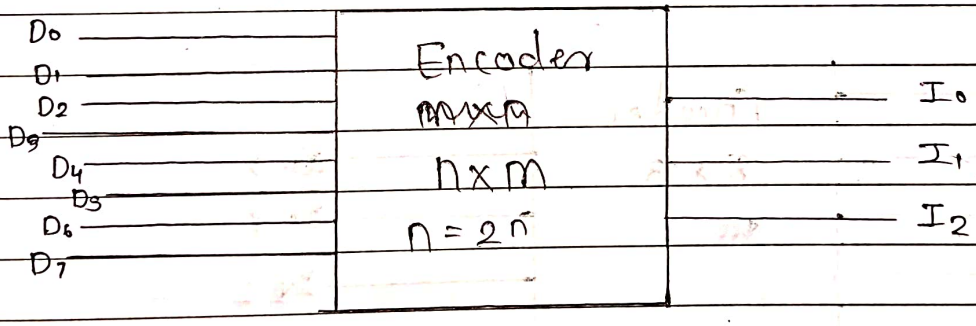
* ata time only one input line is active

M T W T F S S
Page No.:
Date:
YOUVA

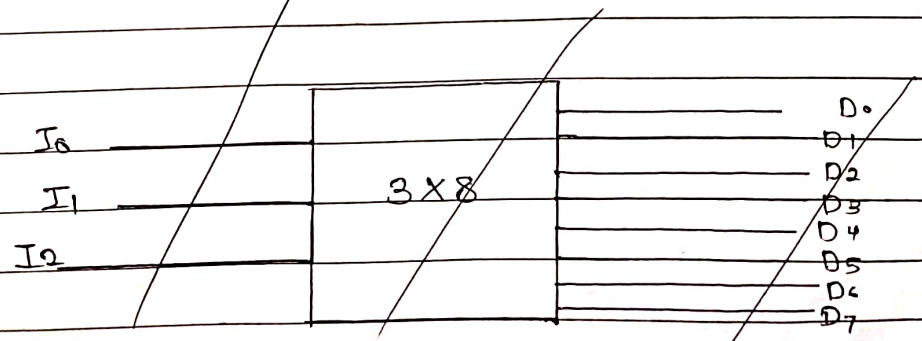
Encoder :

- An Encoder is combinational circuit that converts binary information in the form of 2^N input lines into N output lines which represent N bit code for the input. for simple encoders, it is assumed that only one input line is active at a time.
- Application of ~~Dec~~ Encoders are converting other code into binary
  - Octal to binary
  - Hexadecimal to binary
  - Decimal to binary

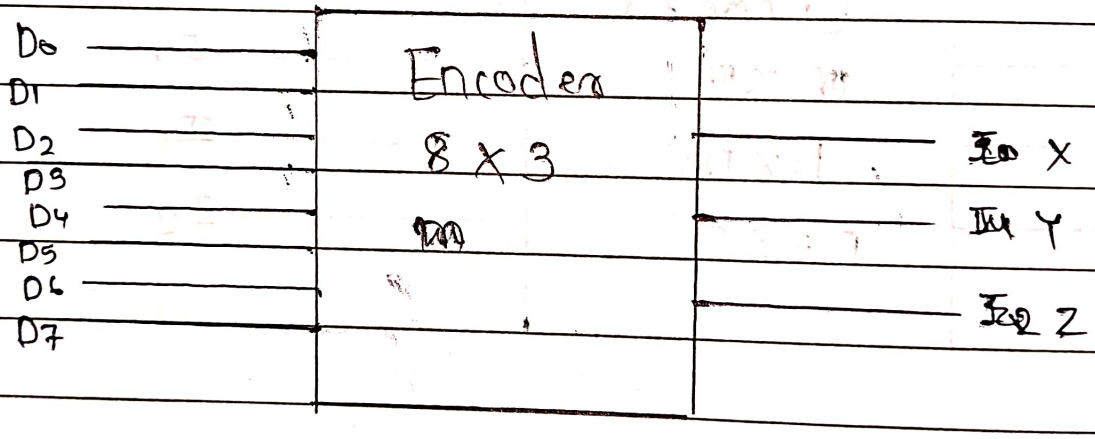
* Block diagram :



* Example diagram :



* Example Diagram :



* Truth table : Encoder

Enable (E)	Inputs								outputs		
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	X	Y	Z
1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	1	0	0	0	0	1	1
1	0	0	0	1	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0	0	1	0	1
1	0	1	0	0	0	0	0	0	1	1	0
1	1	0	0	0	0	0	0	0	1	1	1

Truth tables

Page No.:
Date:
YOUVA

Incod: Decoder table ①

Incoder table ②

① Decoder table

Input pattern			Active output
C	B	A	
0	0	0	Y_0
0	0	1	Y_1
0	1	0	Y_2
0	1	1	Y_3
1	0	0	Y_4
1	0	1	Y_5
1	1	0	Y_6
1	1	1	Y_7

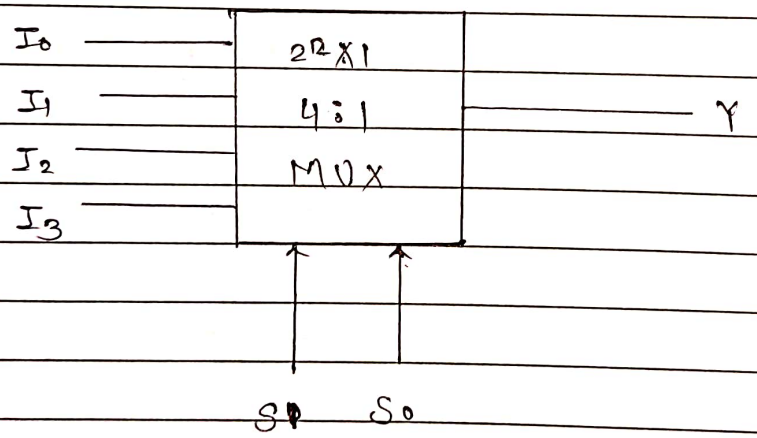
② Incoder table

Input ~~pattern~~ line active	~~Active output~~ Pattern		
	Y_2	Y_1	Y_0
I_0	0	0	0
I_1	0	0	1
I_2	0	1	0
I_3	0	1	1
I_4	1	0	0
I_5	1	0	1
I_6	1	1	0
I_7	1	1	1

Multiplexer : (2^n input) → (1 output) (switch jaisa kaam karta hai)

- A multiplexer is combination circuit that has 2^n input lines and a single output line.
- A multipler is an electronic switch that connects 1 out of 'n' inputs to an output.
- It is functionally comple i.e all boolean functions can realize use one multiplexer without any other gates.

* Diagram :

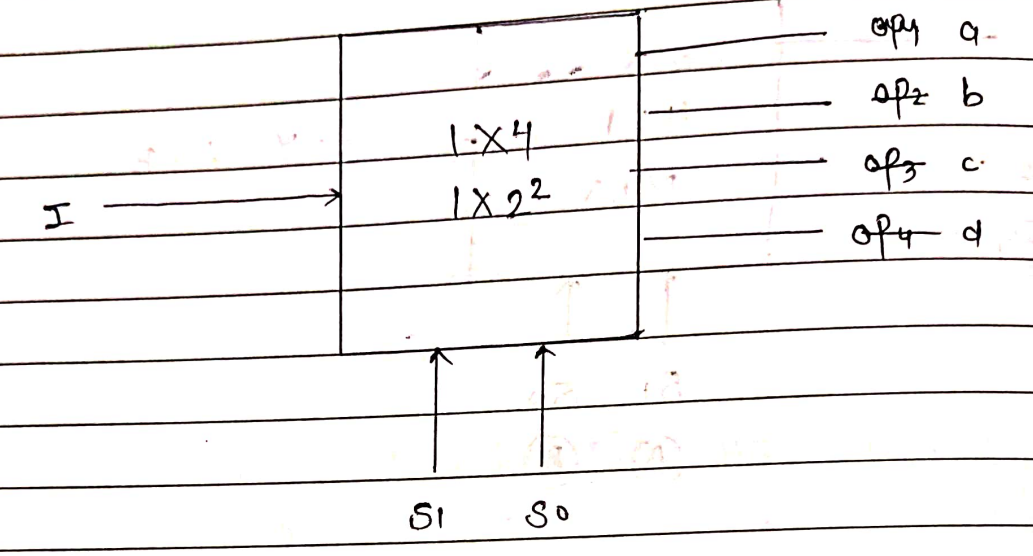


*

S1	S0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Demultiplexer : (1) : (2^n) ← output

Input



S_1	S_0	op_1 a	op_2 b	op_3 c	op_4 d
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

* A demultiplexer is digital circuit that takes a single input signal and directs it to one of multiple output lines. It's the opposite of multiplexer, which has multiple inputs and one output.

KGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCEKGCE

Q4 Write note on :

1] **Gray Code :** (i) Gray code system is a binary number system in the which two successive pair of numbers differs in only one bit.

(ii) It is also known as Reflected code (RBC) or cyclic code.

(iii) Binary number is converted to gray code to reduce switching operation.

(iv) Today gray code are widely used to Facilitate the error correction in digital communication such as cable TV system

(v) It is unweighted code that means it does not depends on positional value of digit and also called unit distance code or minimum error code.

(vi) It is basically work by EX-OR operation.

* steps to solve binary to gray :

step (1) : Record the MSB as it is
step (2) : Add MSB to next bit, record the sum of and neglect the carry.
step (3) : Reapeat the process.

Ex.)
1 0 1 1
↓ ↘ ↓ ↘ ↓ ↘ ↓
1 1 1 0

1110

2] **BCD Codes : Binary Coded decimal**

i) The binary coded decimal (BCD) is a type of binary code used to represent a decimal number in an equivalent binary form using (8421) It is an ee 6 bit code therefore 64 different symbol represents

ii) BCD is not another number system like binary, octal, decimal and hexa-decimal.

iii) It is the fact the decimal system with each digit encoded in it's binary equivalent.

Ex.> $(183)_{10}$ to BCD

1 → 0001
8 → 1000
3 → 0011

0001 1000 0011

• $(000110000011)_{BCD}$

3.] Excess - 3 code :

i) The excess 3 code (or XS3) is a non-weighted code used to express decimal numbers. It is self complementary binary coded decimal

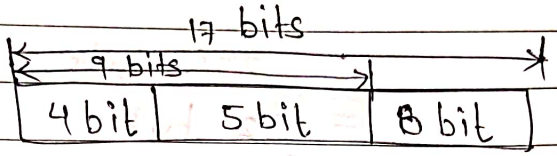
ii) It is written using BCD code as, excess -3 code is equal to BCD +3.

ex.> 1001 to Excess - 3

```
   1 0 0 1   — (9)10
(+) 0 0 1 1   — (3)10
   -------
   1 1 0 0
```

Q4 consider a 2 way set associative mapped cache of size of main memory is 128 KB. find
(a) Number of bits in tag
(b) Tag directory size

→



Set size = 2 = 2^1

cache memory size
~~Cache memory~~ = 16 KB = $2^4 \cdot 2^{10}$ bytes
= 2^{14} bytes

main memory size
~~block size~~ = 128 KB
= $2^7 \cdot 2^{10}$ bytes
= 2^{17} bytes

block size = 256 bytes = 2^8 bytes

No. of main Memory block = $\frac{\text{Size of main memory}}{\text{Block size}} = \frac{2^{17}}{2^8} = 2^9$

No. of lines cache memory = $\frac{\text{cache memory size}}{\text{Block/Line size}} = \frac{2^{14}}{2^8} = 2^6$

No. of set = $\frac{\text{Number of cache line}}{\text{set size}} = \frac{2^6}{2^1} = 2^5$

→ (a) ~~tag~~ Number of bits in tag = total memory - set no - B/L offset
= 4 bit

→ (b) Tag directory size = No. of lines × No. of tag bit

= 2^6 × 4

= 256 bits

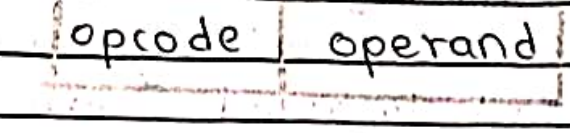
= 32 bytes

Date :

Q.5. Explain the various addressing modes.

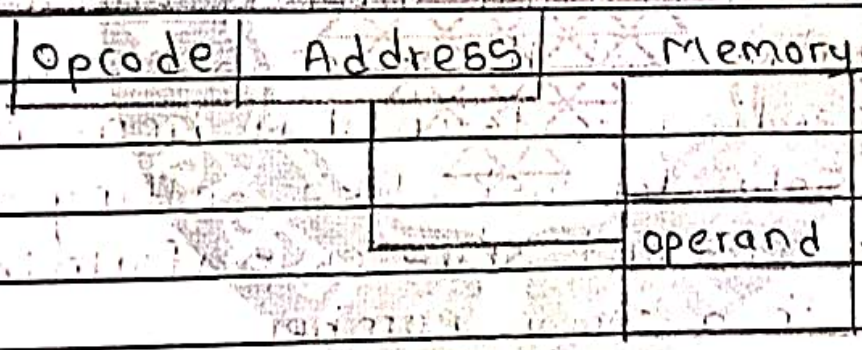
→

i) Immediate Addresing mode



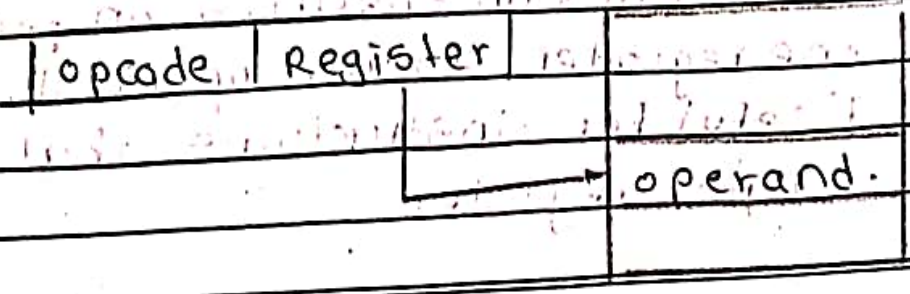
- The operand value is present as a part of the instruction.
- Used for accessing constants and set values of variables.
- It requires no memory reference for accessing the operand.

ii) Direct Addressing Mode



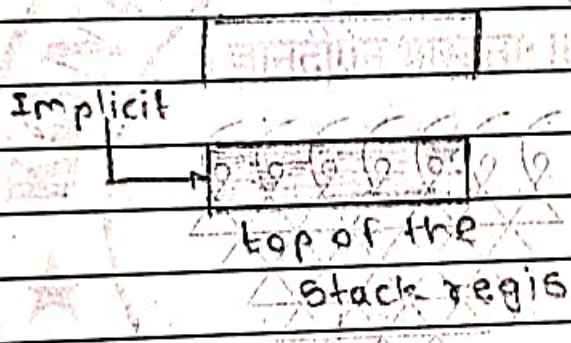
- The effective address of the operand is present as a part of the instruction.
- It requires only one memory reference
- It can access limited range of address.

iii) In Register Addressing mode



- The operands are placed in the register and the address of register is indicated in the instruction
- No memory reference hence it's a fast transfer, small address field.
- It has limited no of registers which requires greater management efforts on programmer side.

iv) Stack Addressing Mode



- An implicit addressing mode is in which no memory is mentioned.
The address of the operand is always the top of the stack

Q1 Explain & Draw Von-Neumann architecture and explain in brief

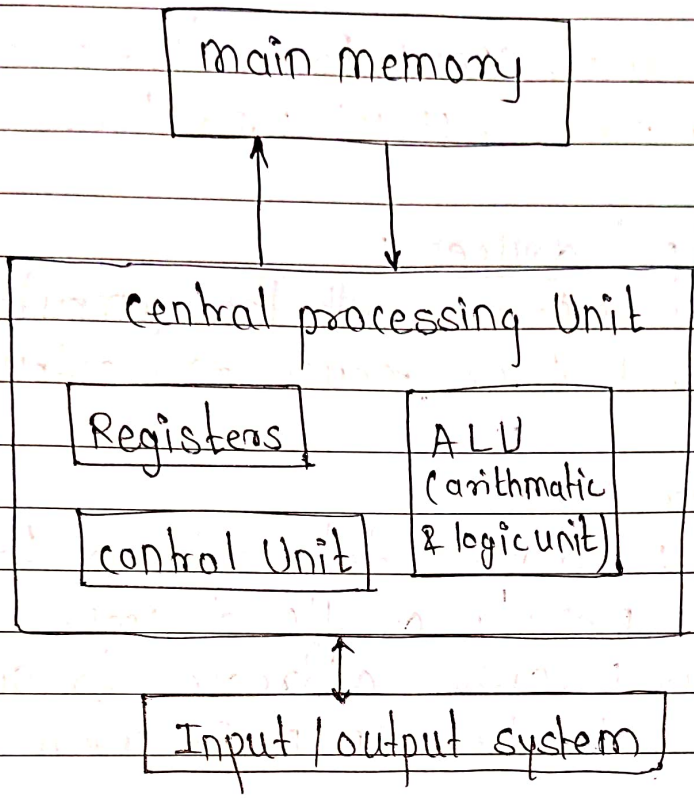
→ (1) The von Neuman architecture is a computer design model where a single memory storage both instruction and data. §

(2) It consists of a CPU (with ALU, control unit, and registers), memory, and I/O devices interconnected by buses. The ~~architecture~~

(3) ~~We~~ The architecture works ~~on~~ by fetching, decoding, executing and storing instructions sequentially, ~~and~~

(4) Its simplicity and flexibility have made it the foundation of modern computing, but it suffers from "von Neuman bottleneck" where data transfer is limited by a shared bus system.

⑤ Diagram :



⑥ Key components :

ⓐ Central processing unit (CPU) :
- Arithmatic logic unit (ALU) :- Executes arithmatic and logical operations
- control unit (CU) : Directs the operation of the computer by feching, decoding and executing instructions.
- Registers : small, high speed storage locations within the CPU used for temporary data storage

ⓑ Memory :
- stores both instructions (program code) and data in the same memory space. This is called the stored program concept.

- Divided into → primary memory (RAM)
  → secondary memory (ROM)

- primary memory (RAM) for active process & operation
- secondary memory (ROM) long term data storage

③ Input/output devices :
Allow communication with the computer for data input (eg. keyboard mouse) and output (eg monitor)

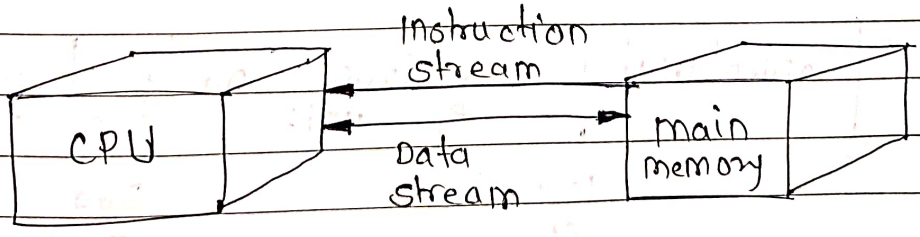
④ Bus system :
- Data Bus : transfer data between the CPU and memory or I/O devices.
- Adress Bus : carries addresses to specify the location in main memory
- control Bus : Sends control signals between CPU, memory and peripherals.

Page No.:
Date:
YOUVA

Q. Explain flynn's classification

→ The processor or CPU has primary functionality of processing the data by executing programs. program is sequence of instructions. Both programs and data are stored in the memory when the processor or CPU executes the program it is needed to ~~Fetchi~~ fetch instruction in the program from the memory. once the instructions is fetched it is decoded and executed in the processor or CPU. while executing it reads the input data operands from memory and it stores output data operands to the memory.

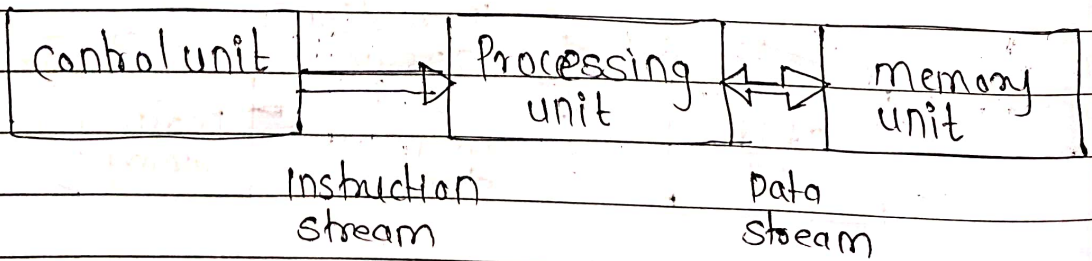
- The flow or movement of instructions from memory to the CPU is called as instruction stream
- The two-way flow of input and output data operands between the CPU and the memory is called as data stream.
- Therefore, there are two types of stream observed: instruction stream and Data stream



- Based on the multiplicity of the instruction and data streams, the computers are classified into four different types. This classification or taxonomy is called as Flynns classification.

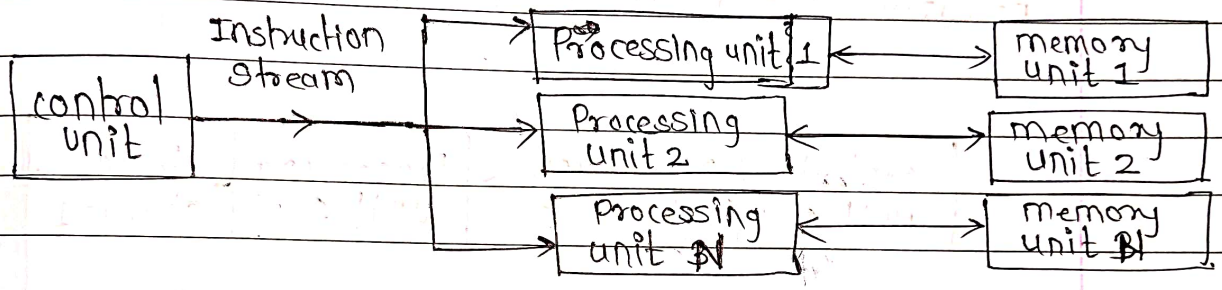
- The four types are
1. SISD : single instruction stream, single Data stream
2. SIMD : single instruction stream, multiple Data stream
3. MISD : Multiple instruction stream, single Data stream
4. MIMD : Multiple Instruction stream, multiple Data stream.

1. SISD : single instruction stream, single Data Stream
- An SISD computing system is a uni-processor machine which is capable of executing a single instruction, operating on a single data stream.
- single instruction : only one instruction stream is being acted on by CPU during any one cycle
- single data : only one data stream is being used as input during any on clock cycle
- conventional single-processor von-Neumann computer are classified as SISD systems.
- It is serial (non parallel) computer



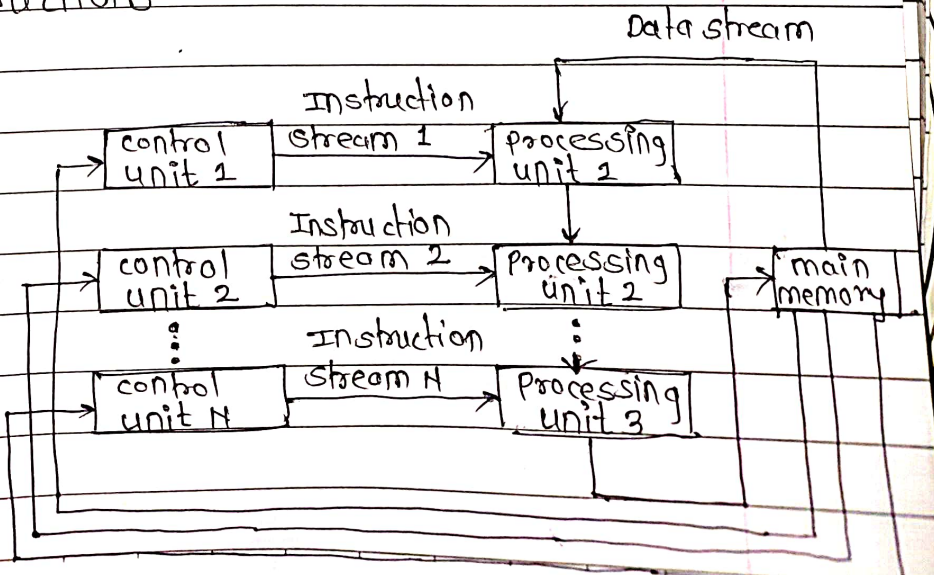
② SIMD

- In SIMD, one instruction controls multiple simultaneous execution of number of processing elements on lock step basis
- Each processor unit has associate data memory for execution of instruction.
- scientific computing based on SIMD Architecture
- Example of this model are vectors & array



③ MISD :

- In MSD, Here multiple processor executes different instruction with single data stream.
- This architecture is not implemented commercially
- Example :- lets have ~~da~~ same data A to be used in multiple instructions
  - A = sin A
  - B = cos A
  - C = tan A



④ MIMD:

- In MIMD, Here multiple instruction processor executes different instructions with example data stream.
- shared memory in MIMD system is symmetric multiprocessor system
- Different processor takes instruction and data from common shared memory
- with the use of buses, multiple processor executes program along with shared memory