COA

★ Computer Architecture:-

- It is the external view of a computer by the programmer.

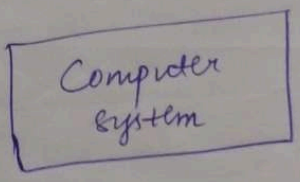
→ Instruction set
→ No. of bits used for data representation
→ I/O mechanisms
→ Addressing techniques

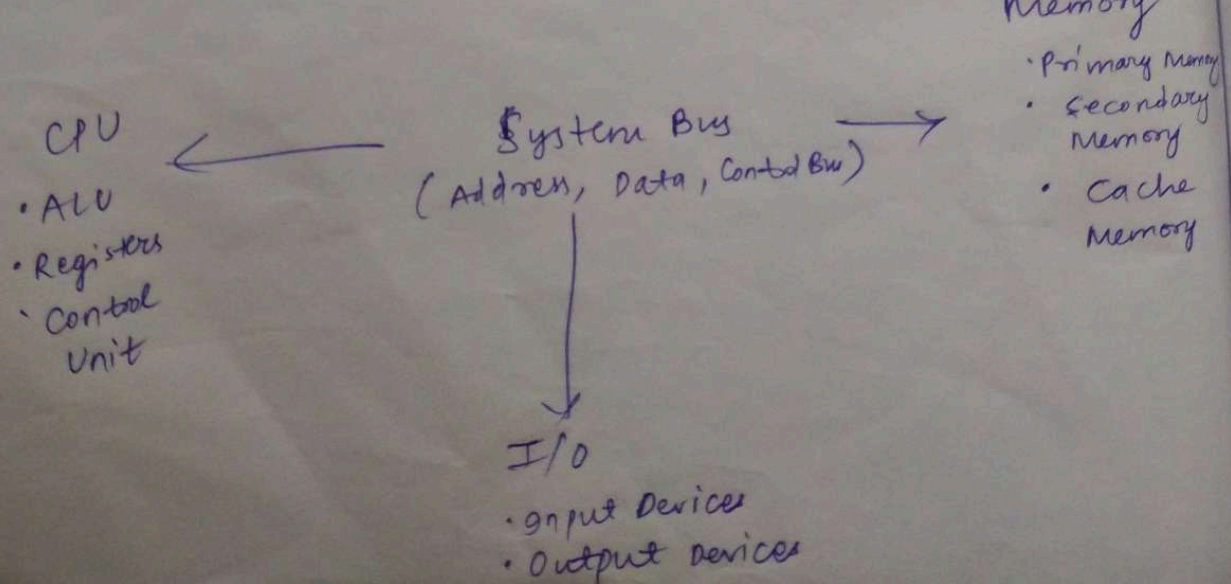
★ Computer Organization:-

- It is the internal view of a computer and the roles that ~~played~~ the internal components play during execution of a program.

→ Control signals
→ Interfaces
→ Memory technology

★





★ System Bus :-

- A Bus is a set of interconnecting lines used to carry information.

Three types :-

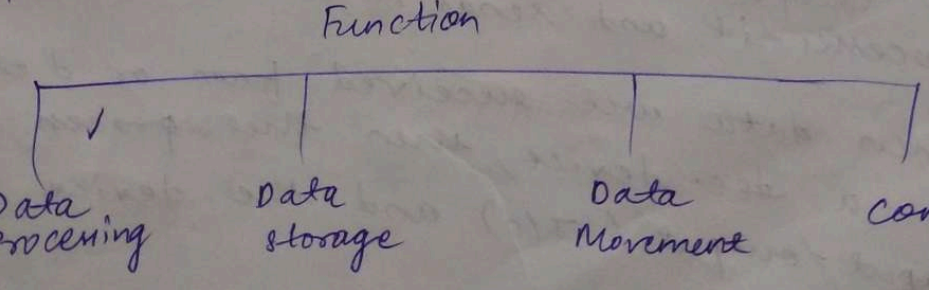
(1) Address Bus :- It carries the address for the operation.
Ex - If address bus is 16-bit, we can connect 2^{16} = 64 KB Memory.
→ Bigger the address bus, bigger is the memory.

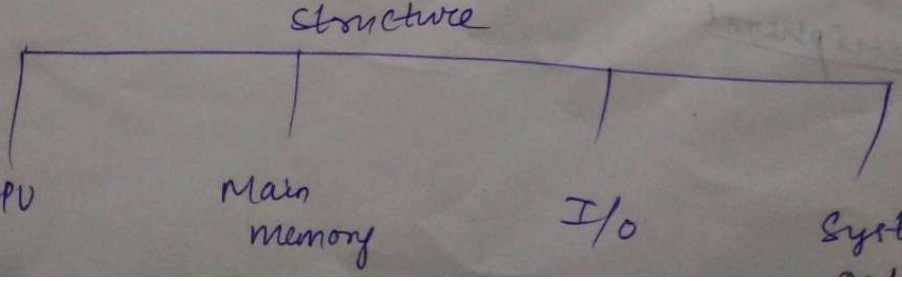
(2) Data Bus :- It carries data to and from the processor.
→ Bigger the data bus, faster the processor, as it can transfer more data in one cycle.

(3) Control Bus :- It carries control signals like RD, WR etc.
→ Determines the kind of operation that will be performed on the system Bus.

★ Structure and Function :-

- Structure is the way in which components relate to each other.
- Function is the operation of individual components as part of the structure.





ing :- computer.

- Data processing is the collection and manipulation of digital data to produce meaningful information.

→ Raw data collection
→ Data preparation
→ Data processing
→ output

(2) Data storage :-

- The computer is processing data on the fly (i.e. data comes in and gets processed and the results go out immediately).
- But the computer must temporarily store atleast those pieces of data which are worked on at any given moment.
- Files of data are stored for subsequent retrieval and update.

(3) Data Movement :-

- The computer takes the data from the source, processes it and sends it to the destination.
- When data are received from or delivered to a dev device, then the process is called input/output (I/O) and the device is called peripheral.

(4) Control :-

- The control unit manages the computer's resources.
- It coordinates the ~~response in~~ operation in response to instructions.

* Single Core Computer :- [structural components of computer]
  - CPU
  - Main Memory
  - I/O
  - System Interconnection [system Bus]

→ Major structural components of CPU :- [Processor].

(i) Control Unit :- Controls the operation of CPU.

(ii) Arithmetic and logic Unit (ALU) :- Performs the computer's data processing functions.

(iii) Registers :- Provides storage internal to the CPU.

(iv) CPU interconnection.

(v) ISU (Instruction sequence unit) :- Determines the sequence in which instructions are executed.

* Multi - core Computer :-

- Multi-core computer uses multiple processors. When multiple processors reside on a single chip, then the term multicore computer is used.
- Each processing unit (consisting of control unit, ALU, registers, cache) is called a core.

Vacuum Tubes :-

- A fundamental design approach first implemented on the IAS computer known as the stored-program concept.
- All of today's computers have this same general structure and function and are thus referred as Von Neumann machines.

→ IAS structure:-

(i) Main Memory
(ii) control Unit: I/O
(iii) CA (central Arithmetic)
(iv) CC (central control)

★ Structure of IAS:-

- set of registers (storage in CPU)
  (i) Memory Buffer Register (MBR)
  (ii) Memory Address Register (MAR)
  (iii) Instruction Register (IR)
  (iv) Instruction Buffer Register (IBR)
  (v) Program Counter (PC)
  (vi) Accumulator Register (AC)
  (vii) Multiplier quotient (MQ) register.

- A fundamental design approach first implemented on the IAS computer known as the stored-program concept.
- All of today's computers have this same general structure and function and are thus referred as von Neumann machines.

→ IAS structure:-

(i) Main Memory
(ii) control unit: I/O
(iii) CA (central Arithmetic)
(iv) CC (central control)

★ structure of IAS:-

- set of registers (storage in CPU)
  (i) Memory Buffer Register (MBR)
  (ii) Memory Address Register (MAR)
  (iii) Instruction Register (IR)
  (iv) Instruction Buffer Register (IBR)
  (v) Program Counter (PC)
  (vi) Accumulator Register (AC)
  (vii) Multiplier quotient (MQ) register.

(i) MBR :- Contains a data to be stored in memory or sent to the I/O unit.

(ii) MAR :- It stores the address of the memory from which data will be fetched to the CPU register.

(iii) IR :- Contains the instruction which is currently executed.

(iv) IBR :- Employed to hold the right hand instruction from a word in memory.

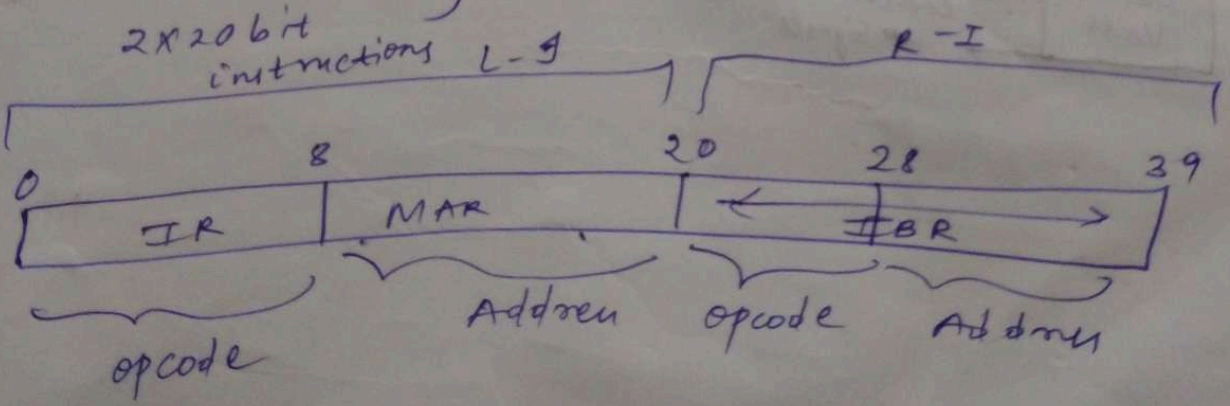
(v) PC :- Contains the address of the next instruction pair to be fetched from memory.

(vi) AC :- It stores intermediate ALU results.

(vii) MQ :- It is also employed to hold temporarily operands and results of ALU operation.

* IAS Memory Format :-

4096 x 40 bits] Main memory

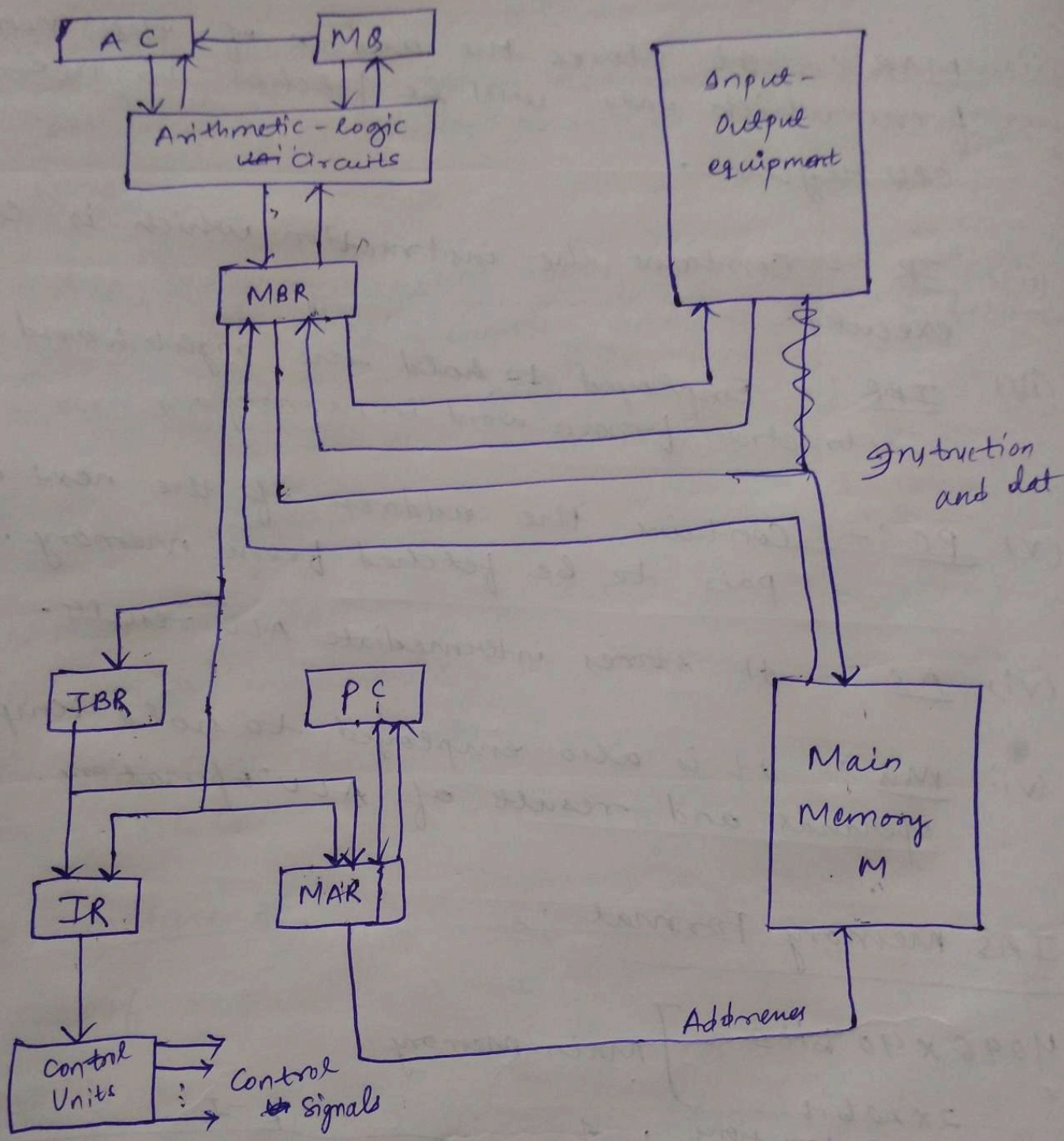


MBR (20:39) → IBR

MBR (0:7) → IR

MBR [8:11) → MAR

Instruction Cycle



1. Load M(X) 500, ADD M(X) 501 (PC=1)

MAR ← PC
MBR ← M[MAR]
IBR ← MBR[20:39]
IR ← MBR[0:7]
MAR ← MBR[8:19] STOR
MBR ← M[MAR]
AC ← MBR
IR ← IBR[0:7]
MAR ← IBR[8:19] LOAD
PC ← PC+1
~~← M[MAR]~~
MBR ← M[MAR]
AC ← AC+MBR

ADD

2. STOR M(X) 500, OTHER INSTRUCTION (PC=2)

MAR ← PC
MBR ← M[MAR
IBR ← MBR[20:39]
IR ← MBR[0:7]
MAR ← MBR[8:19] STOR
MBR ← AC
~~M ← M[MAR]~~
M[MAR] ← MBR

★ IAS Instruction set :-

Conditional branch	15 0000 1111	Jump +M(X, 0:19)	
	16 0001 0000	Jump +M(X, 20:39)	
Arithmetic	5 0000 0101	ADD M(X)	Add M(X) to AC, put result in AC
	7 0000 0111	ADD |M(X)|	Add |M(X)| to AC
	6 0000 0110	SUB M(X)	SUB M(X) from AC
	8 0000 1000	SUB |M(X)|	SUB |M(X)| from AC
	11 0000 1011	MUL M(X)	MUL M(X) BY MQ, most sig in AC, least sig in MQ
	12 0000 1100	DIV M(X)	DIV AC by M(X), q → MQ, r → AC
	4 0001 0100	LSH	Mul AC by 2
	21 0001 0101	RSH	DIV AC by 2
Address Modify	17 0001 0001	STOR M(X, 8:19)	
	19 0001 0011	STOR M(X; 28:39)	

a)

Address	Contents
08A	010FA210FB
08B	010FA 0F08D
08C	020FA210FB

		X	
010FA	0000 0001 LOAD M(X)	0FA 0FA	Transfer M(X) to AC
210FB	0010 0001 STOR M(X)	0FB 0FB	Transfer contents of acc. to memory loc. X
010FA	0000 0001 LOAD M(X)	0FA 0FA	Trans - . .
0F08D	0000 1111 Jump+M(X,0:19)	08D 08D	—
020FA	0000 0010 Load - M(X)	0FA 0FA	—
210FB	0010 0001 STOR M(X)	0FB 0FB	—

Address	Contents
08A	~~LOAD MCX~~ LOAD M(0FA) STOR M(0FB)
08B	LOAD M(0FA) Jump+M(08D)
08C	LOAD - M(0FA) STOR M(0FB)
08D	

Q) First, memory is accessed to fetch the instruction.
Next, memory is accessed to fetch the data.

★ Evolution of Computers :-

→ Moore's Law :-

- Increased density of components on chip.
- Gordon Moore - co-founder of Intel
- No. of transistors on a chip will double every ye

★ IBM 360 series :-

: 1964
: Replaced (and not compatible with) 7000 series
- First planned "family" of computers

→ Similar or identical instruction sets
→ Similar or identical O/s
→ Increasing speed
→ Increasing number of I/O ports
→ ~~Increasing me~~
→ Increased memory size
→ Increased cost

- Multiplexed switch structure.

★ MIPS Rate :-

$$\text{MIPS rate} = \frac{f}{CPI \times 10^6}$$

∮ CPI = Cycle per Instruction
f = Clock frequency.

04

- **Execution Time** $= T_c \sum_{i=1}^{\infty} C_c$

T_c = Cycle time

C_c = no. of cycles required for instruction execution.

N = No. of instruction in test program.

$T_c \sum_{i=1}^{\infty} C$

$T_c \sum_{i=1}^{\infty} C$

Module - 2

* Addition in 2's complement :-

Case I :- Both no's +ve, no overflow, 5 bit

```
 +6  ------->  0 0110
 +7  ------->  0 0111
 ---           ------
 13            0 1101
                 \___/
                  13
```

Case-II :- Both no's +ve, overflow, 5 bit

```
               1 11
 +9  ------->  0 1001
 +7            0 0111
 ---           ------
 16            1 0000
               ↓ Invalid
 as sign bit is -ve.
               ↓
             6 bit
              1 1 11
            0 01001
          + 0 00111
          ---------
            0 10000
              \___/
               16
```

Case-III :- +ve no > -ve no., 5 bit

```
  1
 1001
 + 1
 ----
 1010
```

```
 +13  --------------------->  01101
 -6   ------ 0110 --------->  11010
 ---                         +------
 +7                          1 00111
                                \__/
                                 +7
```

Case-4 [-ve no + +ve no]

$-15 \longrightarrow 01111 \xrightarrow{2's} 10001$

$+9 \longrightarrow 01001$

$-6 \longrightarrow 11010$

$0101 \xrightarrow{+1} 0110$

$11010 \xrightarrow{2's} 00110 = -6$

Case-5 :- Both no's are -ve, no overflow.

$-8 \longrightarrow 01000 \xrightarrow{2's} 11000$

$-4 \longrightarrow 00100 \xrightarrow{2's} 11100$

$-12 \longrightarrow 10100$

discard 1

$10100 \xrightarrow{2's} 01100 = -12$

→ subtraction also in 5-bit.

Q) 16 bits :-

+512

512 256 128 64 32 16 8 4 2 1

0000 0010 0000 0000

-29

1000000000011101

$\downarrow$

✓

$256 \times 2 = 512$

$256 + 128 = 384$

$384 + 64 = 448$

$448 + 32 = 480$

$480 + 16 = 496$

$496 + 8 = 504$

$16 + 8 = 24$

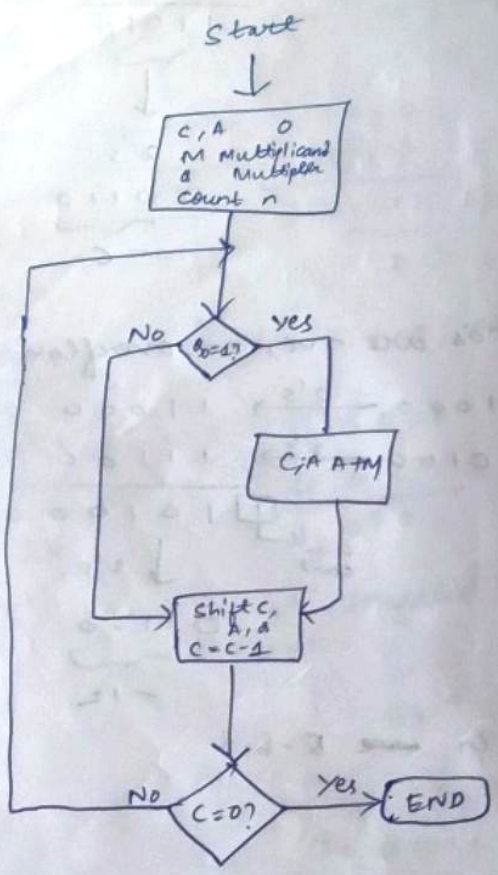
$24 + 4 = 28$

$28 + 1$

★ Multiplication :—

Unsigned Binary Multiplication

[Nothing means zero]



Product in A, Q

C	A	Q	M	operation
0	0000	1101	1011	Initial Values
0	1011	1101	1011	A = A+M
0	0101	1110	1011	Shift
0	0010	1111	1011	Shift
0	1101	1111	1011	A = A+M
0	0110	1111	1011	Shift
1	0001	1111	1011	A = A+M
0	0000	1111	1011	Shift

First cycle n=4

second cycle n=3

Third cycle n=2

Fourth cycle n=1

nothing means zero

* Multiplication of Signed Binary Numbers :- [Whenever Q is -ve, last multi. ↓ 2's comp. add.]

Case-I :- When multiplicand is -ve and multiplier is +ve.

(M) −5 → 0101 —2's→ 1011

(Q) ×7

−35

0000 + 1011 = 1011

0010 + 1011 = 1101

0110 + 1011 = 1001

0110 + 1011 = 10001 (discard)

1 0 1 1 0 1 1 0 1

discard

↓ 1's

0 0 1 0 0 0 1 0

↓ 2's

0 0 1 0 0 0 1 1

= −35

0101 → 1010 ↓ +1 = 1011

Case-II :- When both are -ve, [last mult → 2's → then add]

(M) −5 → 0101 —2's→ 1011

(Q) −7 → 0111 —2's→ 1001

35

```
  ① ① ① ①
    1 1 1 1 1 0 1 1
    0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0
  +   0 0 1 0 1
  ---------------
  [1] 0 0 1 0 0 0 1 1
```

discard.

= +35.

1101.1

↓1's

00100

↓2's

00101

Case-III :- When multiplicand is +ve, multiplier is -ve. [last mu → 2's → add]

(M) 5 → 0101

(Q) −7 → 0111 —2's→ 1001

−35

```
    0 0 0 0 0 1 0 1
    0 0 0 0 0 0 0 0
  +   0 0 0 0 0
    1 1 0 1 1
  -------------
  1 1 0 1 1 . 1 0 1
```

00101

↓1's

11010

↓2's

11011

Assign bit -ve.

↓1's

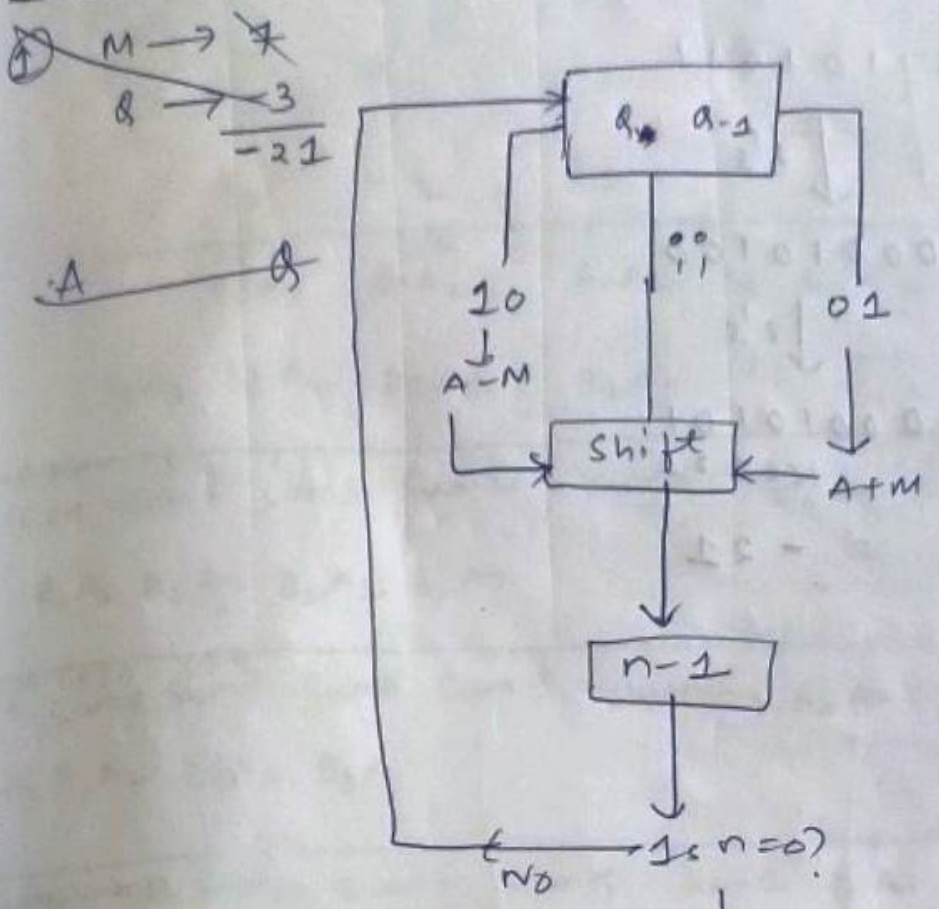
00100010

↓2's

00100011

= −35

★ Booth Algorithm :- [N] [sign get restored].



1) M → 7 → 0111 Yes
Q → -3 → 0011 → 2's → 1101
-21

A	Q	Q_{-1}	M	Initial Values	
0000	1101	0	0111	Initially	
1001	1101	0	0111	A ← A-M	First cycle n=4
1100	1110	1	0111	Shift	
0011	1110	1	0111	A ← A+M	Second cycle (n=3)
0001	1111	0	0111	Shift	
1010	1111	0	0111	A ← A-M	Third cycle n=2
1101	0111	1	0111	Shift	
1110	1011	1	0111	Shift	Fourth cycle n=1

0001
+1001
1010

1
1100
+0111
10011

1100
+1
1101

1000
+1
1001

0000
+1001
1001

Result = AQ

= 11101011

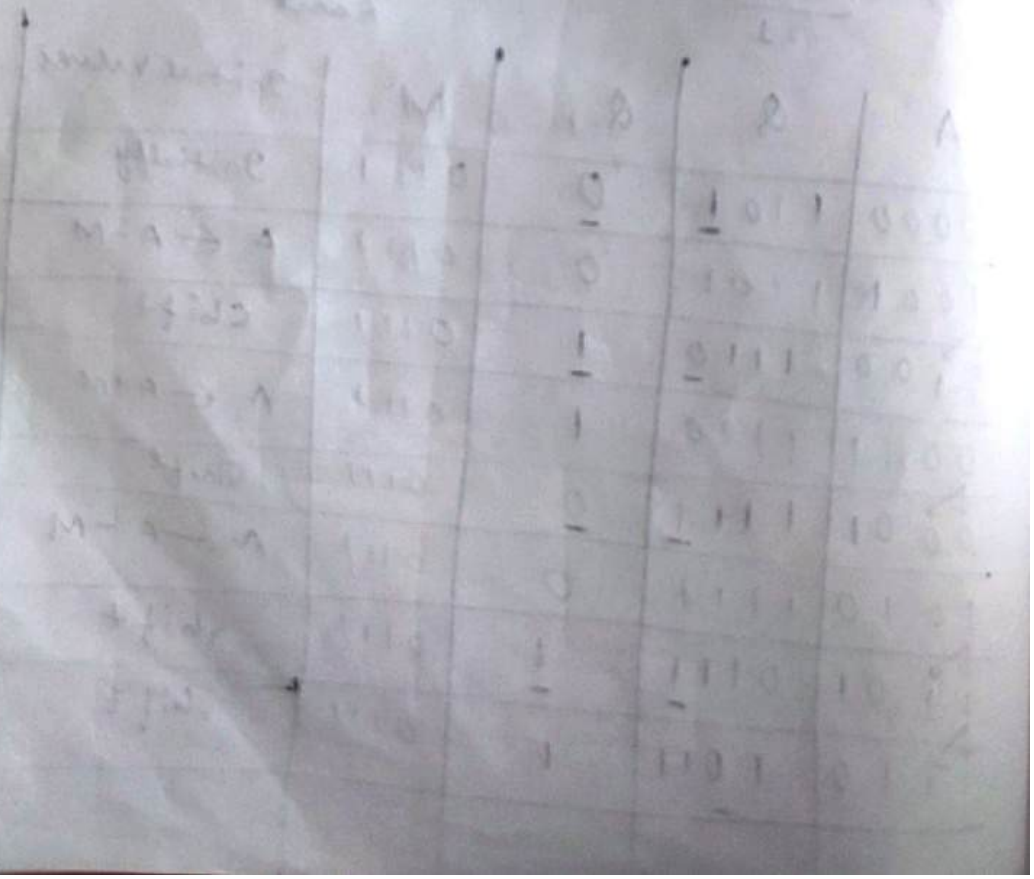
↓ 1's

00010100

↓ 2's

00010101

= -21



★ Array Multiplication :-

A_3 A_2 A_1 A_0

$\times$ B_3 B_2 B_1 B_0

CY2 CY1

B_0A_3 B_0A_2 B_0A_1 B_0A_0

B_1A_3 B_1A_2 B_1A_1 B_1A_0

CY7 CY6 CY5

CY4 Sum4 Sum3 Sum2 Sum1 B_0A_0

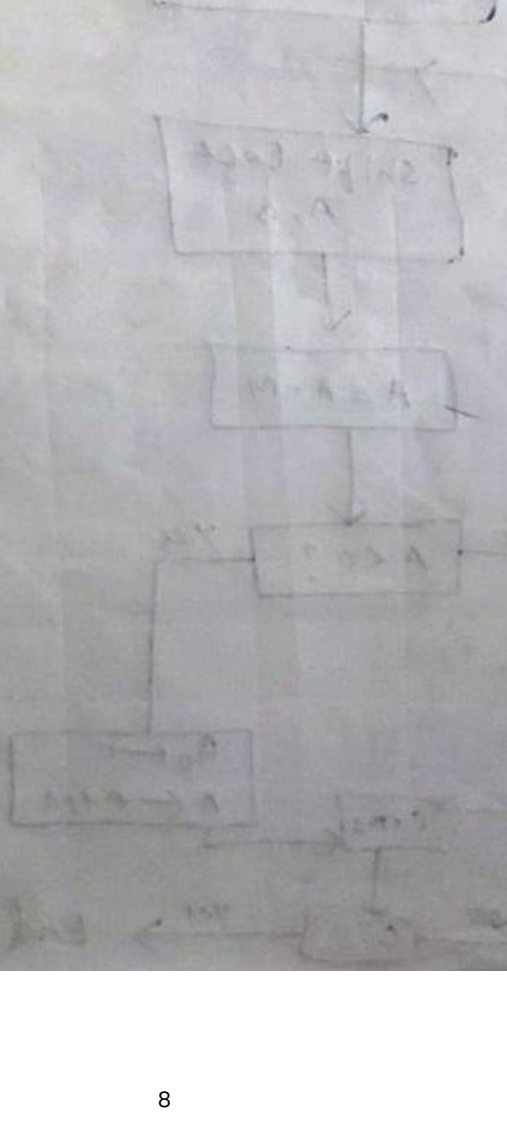
B_2A_3 B_2A_2 B_2A_1 B_2A_0

CY11 CY10 CY9

CY8 Sum8 Sum7 Sum6 Sum5 Sum1 B_0A_0

B_3A_3 B_3A_2 B_3A_1 B_3A_0

CY12 Sum12 Sum11 Sum10 Sum9 Sum5 Sum1 B_0A_0



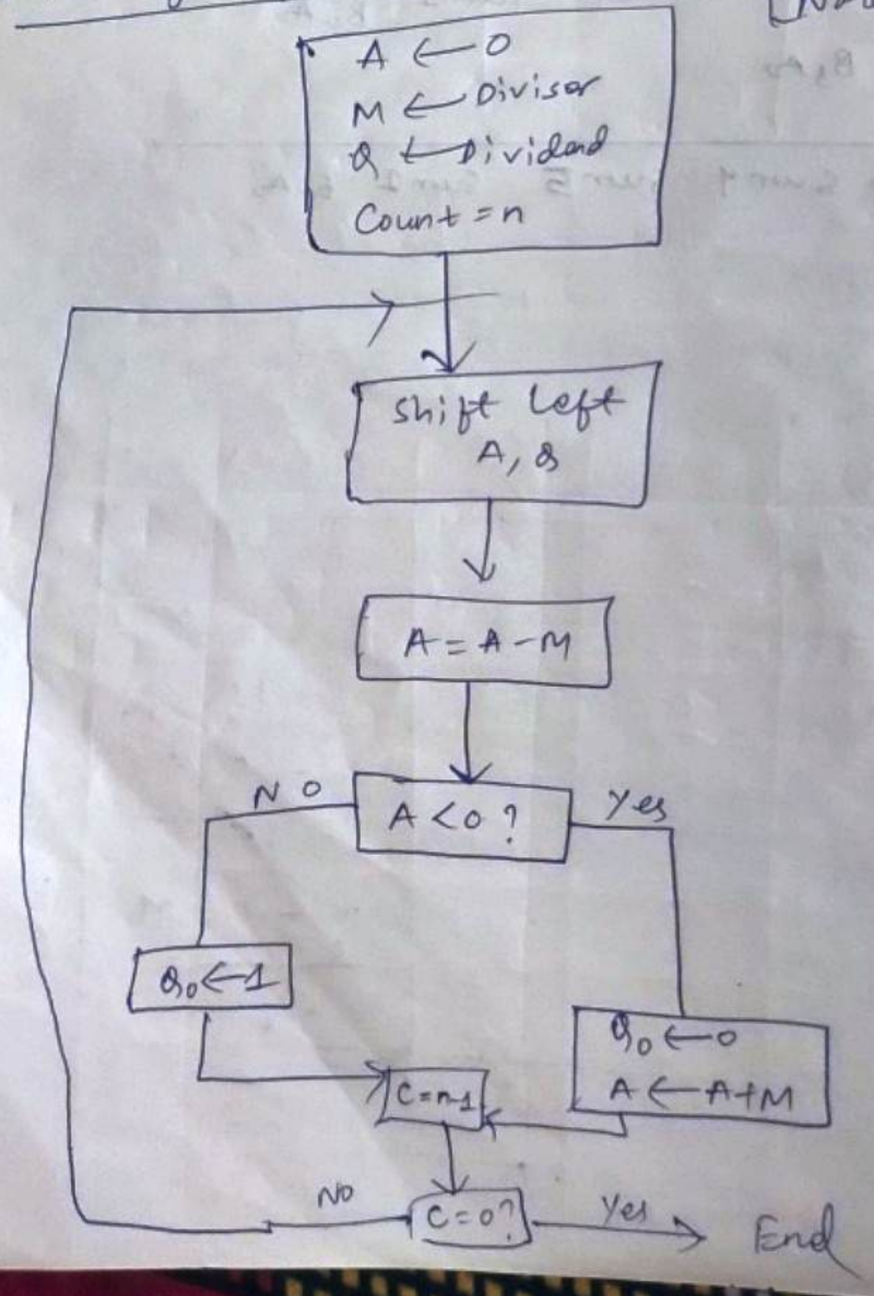
★ Division of Unsigned Binary Numbers :-

11 9
9

```
        00001
1011 ) 10010011
      -1011
       1110
      -1011
        1111
        1011
         100
```

★ Restoring Algorithm :- Unsigned Binary Division

[Nothing means zero]



Q) 7/3

Q = 0111

M = 0011 —2's→ 1101

Nothing means zero.

A	Q	operation
0000	0111	initially
0000	1110	shift left
1101	1110	A ← A − M (n=4)
0000	1110	$q_0 \leftarrow 0$, A ← A+M
0001	1100	shift left
1110	1100	A ← A − M (n=3)
0001	1100	$q_0 \leftarrow 0$, A ← A+M
0011	1000	shift left
0000	1000	A ← A − M (n=2)
0000	1001	$q_0 \leftarrow 1$
0001	0010	shift left
1110	0010	A ← A − M
0001	0010	$q_0 \leftarrow 0$, A ← A+M

R = 1 Q = 2

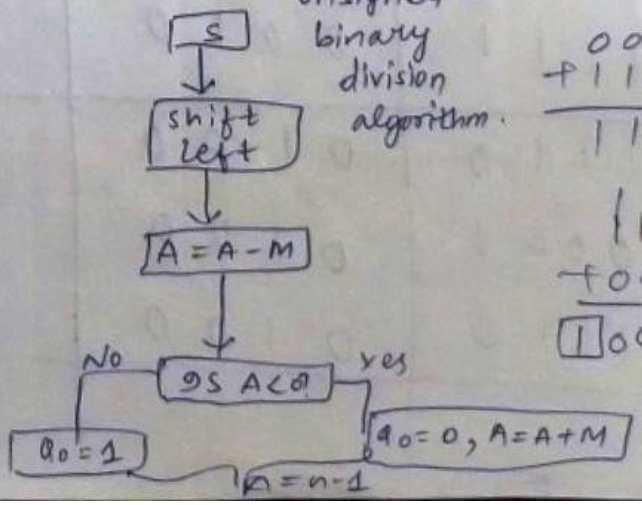
0011
1100
+ 1
1101

0001
+1101
1110

11
1110
+0011
1 0001

111
0011
+1101
1 0000

(Restoring) unsigned binary division algorithm.



0001
+1101
1110

1110
+0011
1 0001

★ Division of signed numbers using Restoring Algorithm :-

Algorithm :-

~~1) At each step~~

1) Left shift

2) If sign of A and M is the same then $A = A - M$ else $A = A + M$.

3) If sign of A remains the same or A and Q becomes zero, then quotient = 1.

4) If sign of A changes, then quotient = 0 ($q_0 = 0$) and A is restored.

Q1 $-\frac{7}{3}$ → Q → 0111 $\xrightarrow{2's}$ 1001

→ M → 0011 $\xrightarrow{2's}$ 1101

→ Nothing means zero

A	Q	Operation
~~0000~~	1001	~~Initi~~ A = Sign extension of Q ... n = 4
1111	0010	Shift left
0010	0010	A = A + M
1111	0010	$q_0 = 0$, Restored A
1110	0100	Shift left
0001	0100	A = A + M
1110	0100	$q_0 = 0$, Restord A

1100 1111 1111	1000 1000 1001	shift left A = A+M $Q_0 = 1$
1111 0010 1111	0010 0010 0010	shift left A = A+M $Q_0 = 0$, Restored A

Q = 2

2's

0001

R = 1

★ Non-Restoring Algorithm for Division :-

1) Left shift
2) A = A - M
3) If result = +ve, then $q_0 = 1$. [Next step - Sub]
4) If result = -ve, then $q_0 = 0$. [Next step - Add]

Q1 $\frac{7}{5}$ → Q → 0111
→ M → 0101 → 2's → 1011

Nothing means zero.

A	Q	Operation	
0000	0111	Initially	(n=4)
0000	1110	shift left	
1011	1110	A = A - M	(n=3)
1011	1110	$q_0 = 0$	
0111	1100	shift left	
0100	1100	A = A + M	n=2
0100	1101	$q_0 = 1$	
1001	1000	shift left	
1110	1000	A = A + M	n=1
1110	1000	$q_0 = 0$	
1101	0000	shift left	
0010	0000	A = A + M	n=0
0010	0001	$q_0 = 1$	
R=2	q=1		

★ IEEE Floating Point Representation :-

128 64 32 16 8 4 2 1 0 • 25 • 125 • 0625 • 03125

Q) Find the binary equivalent of 0.25.

Ans- Multiply the fraction by 2 until the fractional part becomes ~~in~~ whole.

$$\begin{array}{r} 0.25 \\ \times\ 2 \\ \hline 0.50 \\ \times\ 2 \\ \hline 1.00 \end{array}$$

. .5 .25

0 1

0.50
0.125
0.625

★ Solution is Normalization :-

$37.25_{10} =$

32 16 8 4 2 1 0 5 . 25 . 125 . 0625 . 03125

1 0 0 1 0 1 . 0 1

$1.00101\,01 \times 2^{5}$ → exponent

mantissa

32
4
1

0.625
+ 0.250
0.875
0.064

7.625

111.101 $= 1.11101 \times 2^{2}$

$0.3125_{10} = 0.0101_{2}$ $\quad 1.01 \times 2^{-2}$

Q) Find the IEEE FP representation of 40.15625

64 32 16 8 4 2 1 . .5 .25 .125 .0625

1 0 1 0 0 0 . 0 0 1 0 1

101000.00101

$1.0100000101 \times 2^{5}$ → exponent

mantissa

$127+5 = 132 \rightarrow 10000100$

exponent.

$01000010\ldots0$ → mantissa.

Q) -24.75

32 16 8 4 2 1 . .5 .25 .125 .0625 .03125

1 1 0 0 0 . 1 . 1

-11000.11

$-1.10001 1 \times 2^{4}$

$127+4 = 131$

10000011 → exponent.

$10001100\ldots0$

mantissa.

Q) 1/16 = 0.0625

64 32 16 8 4 2 1 . .5 .25 .125 .0625 .03125

0 . 0 0 0 1

0.0001 ~~x 2~~

1.00×2^{-4}

127 - 4 = 123

64
+32
96
+16
112
+2
120
+2
122
+1
123

1 1 1 1 0 1 1

exponent

~~0001 ... 0~~

~~mantissa~~

0000 ... 0

Mantissa

Q) Convert the following binary to decimal.

Exponent	Mantissa
10000011	10011000...0

$10000011 \rightarrow 131 = 127 + 4$

4 → Exponent

1.10011×2^4

11001.1×2^4

16 8 4 2 1 . 5

$(25.5)_{10}$

10.1100110×2^1

11001.1×2^4

★ Harvard vs Von Neumann architectures :-

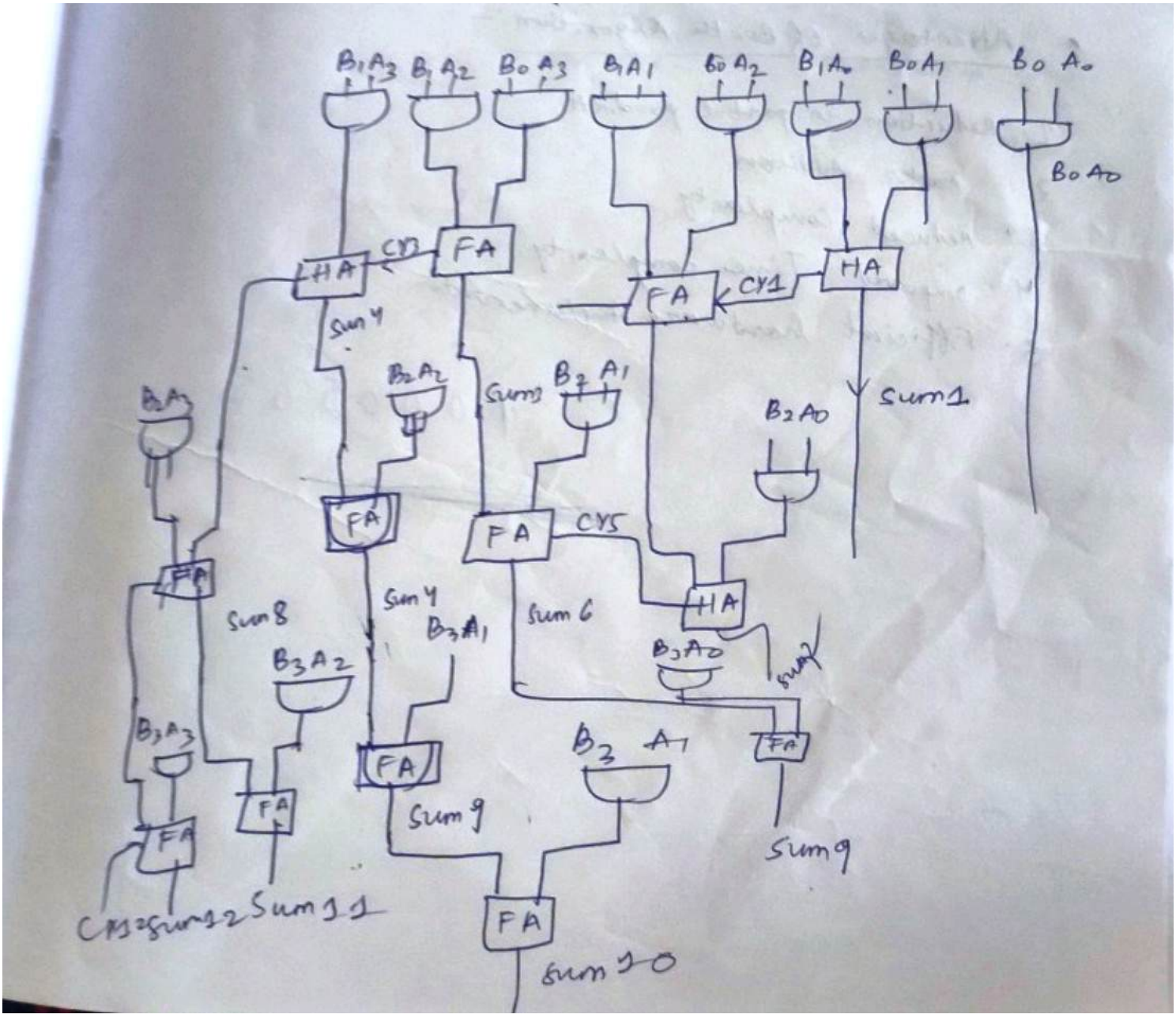
- Harvard and Von Neumann architectures are two fundamental computer architecture designs that differ in the way they handle the storage of program instructions and data.

→ Harvard Architecture :-

- computer's memory is divided into two separate and independend units.
- This separation allows the CPU to fetch instructions and access data simultaneously.
- Expensive
- One clock cycle

→ Von Neumann Architecture :-

- Also known as the "stored-program computer", uses a single memory unit that stores both instructions and data.
- It simplifies memory management.
- Cheaper
- Two clock cycles


B1A3 B1A2 B0A3 B1A1 B0A2 B1A0 B0A1 B0A0
B0A0
HA
CY3
FA
FA
CY1
HA
Sum4
Sum1
B2A2
Sum5
B2A1
B2A0
FA
FA
CY5
FA
HA
Sum8
Sum4
Sum6
B3A1
B3A0
Sum2
B3A2
B3A3
FA
FA
FA
FA
B3 A1
Sum9
Sum9
FA
Sum10

* CISC	RISC
• Emphasis on hardware.	• Emphasis on software
• Include multiclock complex instructions.	• Single clock.
• Memory to Memory	• Register to Register.
• Small code sizes	• Large code sizes
• Transistors used for storing complex instructions	• Spends more transistors on memory registers

* Advantages of Booth Algorithm:-

1. Reduction in partial products.
2. Faster Addition
3. Reduced Complexity
4. Improved Time complexity
3. Efficient hardware implementation

:1000060

Interrupt

★ The purpose of Interrupts :-

• I/O Organisations can be of 3 different types :-

(1) Programmed I/O (Polling / Daisy chain)

(2) Interrupt driven I/O.

(3) DMA

• Interrupts are useful when interfacing I/O devices with low ~~data~~-transfer rates, like a keyboard or a mouse, in which case polling the device wastes valuable procesing time.

→ Types of Interrupts :-

- Hardware Interrupts
- Software Interrupts
- Maskable and non-maskable interrupts.
- Vectored and non-vectored interrupts.
- Interrupt priority.

• Interrupt Vector :- Code loaded on the bus by the interrupting device that contains the Address (segment and offset) of specific interrupt service routine.

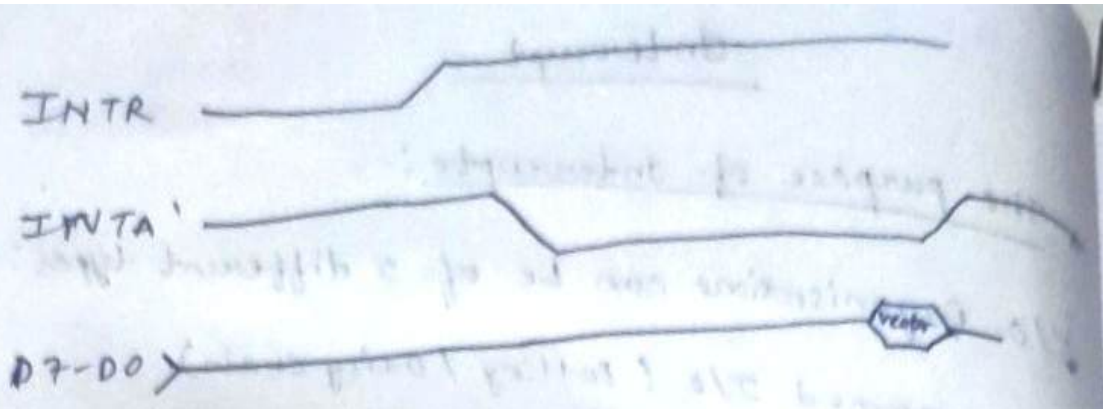
★ Interrupt Pins :-

• X86 Interrupt Pins.

(1) INTR :- Interrupt Request · Activated with logic 1. Level Triggered

(2) INTA : Interrupt Acknowledge · Activated with logic 0 Level Trigger

(3) NMI :- Non-maskable Interrupt · Must be at logic 1. Edge Triggered.



★ Interrupt Vectors :-

- The processor uses the interrupt vector to determine the address of the ISR of the interrupting device.
- The interrupt vector is a pointer to the Interrupt Vector Table.

Each entry in the Interrupt Vector Table is 4 bytes long:

→ The first two represent the offset Address and the last two ~~address~~ the segment address of the ISR.

- The Interrupt Vector Table occupies the address range from 00000H to 003FFH (1024 bytes)

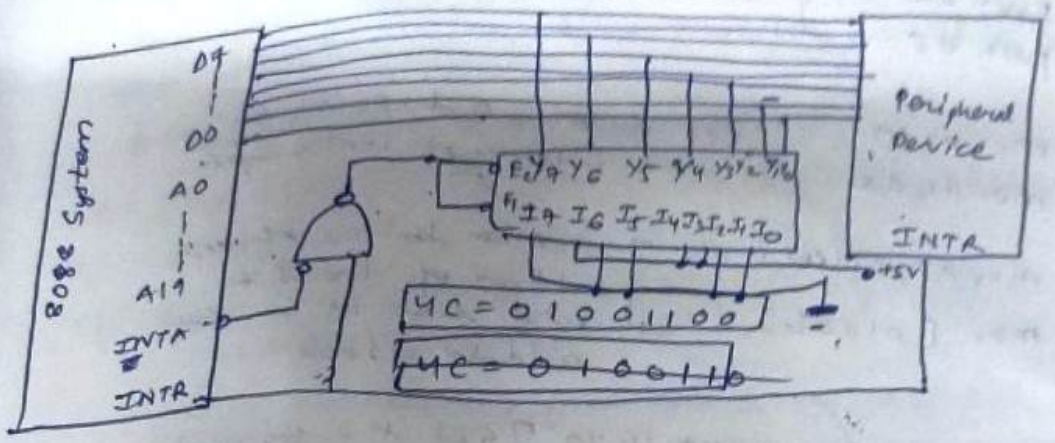
4×42
= 168 - 60 = 108

45H

Q) Draw a circuit diagram to show how a device with interrupt vector 4CH can be connected on an 8086 microprocessor system.

Ans - 4C - 01001100

A B C
1 1 1 2



- The peripheral device activates the INTR line.
- The processor responds by activating the INTA signal.
- The processor reads the data bus to get the interrupt vector.

Q) Using the Interrupt Vector Table shown below, determine the address of the ISR of a device with interrupt vector 42H.

A- Address in table = 4 × 42 + 1 = 108H
(multiply by 4 as each entry is 4 bytes).

- offset low = [108] = 2A , offset high = [109] = 33
- segment low = [10A] = 3C , segment high = [10B] = 7A

Address : 4A3C : 332A = 4A3C0 + 332A = 4D6EAH

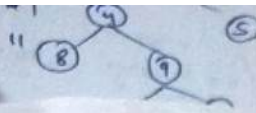
Q) Write a sequence of instructions that initialize vector 40H to point to the ISR "isr40".

Ans Address in table = 4×40 = 100H

```
push ax
push ds        } save registers in the stack.

mov ax, 0
mov ds, ax     } set ds to 0 to point to the interrupt vector table.

mov ax, offset isr40
mov [0100h], ax   ] Get the offset address of the ISR and store it in the address 0100h.

mov ax, segment isr40
mov [0102h], ax   ] Get the segment address of the ISR and store it in the address 0102h.

pop ds
pop ax         } Restore registers from the stack.
```

★ Interrupt Masking :-

- The processor can inhibit certain types of interrupts by use of a special interrupt mask bit.
- This mask bit is part of the flags / condition code register, or a special interrupt register.



★ H/W Interrupt Processing :-

- External interface sends an interrupt signal to the INTR pin.
- The CPU finishes the present instruction and checks the INTR pin.
- The contents of the flag registers are pushed onto the stack.
- ~~Both~~ the IF content is cleared.
- The contents of the CS and IP are pushed onto the stack.
- ~~Initiate~~ The interrupt vector contents are fetched, from (4xN) and placed into the IP.
- While returning from the IRET, the IP, CS and Flag registers are popped from the stack and return to ~~their~~ the interrupt.

→ Software instruction :-

- INT 0 → Divide Error.
- INT 1 → single step operation.
- INT 2 → NMI Pin
- INT 3 → setting a Breakpoint. (while implementing a breakpoint function in a system).
- INT 4 → Overflow.
- CLI - Clear Interrupt Flag. If IF → 0, then interrupts are disabled.
- STI - set Interrupt Flag. If IF is set to 1, then interrupt are enabled.
- IRET - Return from interrupt.

★ Programmable 82C59A :-

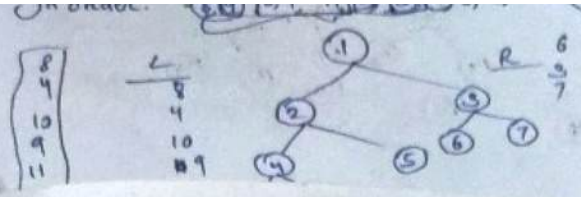
- To allow the 8086 to handle a variety of devices and priority structure, external devices are connected to the 82C59A.
- The 82C59A is programmable. The 8386 determines the priority scheme to be used by setting a control word in the 82C59A.

The following interrupt modes are available:

→ Fully nested :- The interrupt requests are ordered in priority from 0 through 7.

→ Rotating :- In some applications a no. of interrupting devices of equal priority.

→ Special mask :- This allows the processor to inhibit interrupts from certain devices.



* Control Unit Operation :-

- It is responsible for managing and coordinating the execution of instructions within the CPU.

* Microoperations :-

- Explain an instruction.

ADD R1, R2 → Load the contents of R1.
Load the contents of R2.
Add the contents of R1 and R2.
Store the final content in R1.

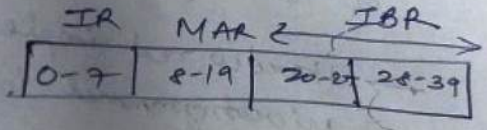
* Fetch :-

- MAR
- MBR
- PC
- IR

Fetch sequence :-

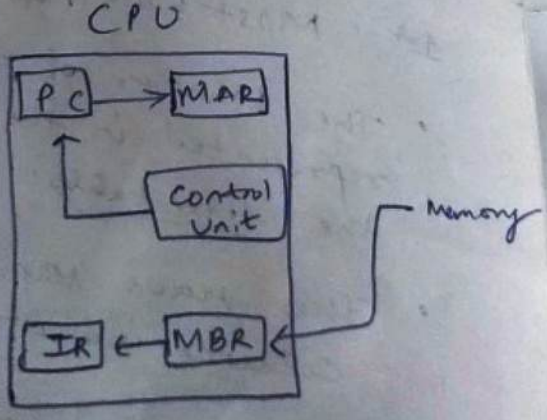
MAR ← PC
MBR ← M(Memory)
PC ← PC+1
IR ← MBR

→ Interrupt Cycle :-



MBR ← PC
MAR ← Save Address
PC ← interrupt service routine address
memory ← (MBR)

Q) ISZ X → increment and skip if zero.



MAR ← IR address
MBR ← memory
MBR ← MBR+1
Memory ← MBR
if (MBR) == 0 then PC ← PC+1

* Instruction Cycle code (ICC):-

- 00 - Fetch
- 01 - Indirect
- 10 - Execute
- 11 - Interrupt

* Types of Micro-Operation :-

- Transfer of data between registers.
- Transfer data from register to external
- Transfer data from external to register
- Perform arithmetic or logical ops.

* Control Unit Implementation :-

- Implementation of Control Unit is broadly of two types :-

→ Hardwired Implementation (RISC)
→ Microprogrammed implementation (CISC).

* Hardwired Control Unit Methods :-

- Most basic type of hardwired control unit.
- The behavior of the control unit is represented in the form of a table called the state table.
- The rows represent the T-states and the columns indicate the instructions.
- Each intersection indicates the control signal to be produced, in the corresponding T-state of every instruction.
- A circuit is then connected based on every column of this table, for each instruction.

→ Advantage :-

- It is the simplest method and is ideally suited for very small instruction sets.

→ Drawback :-

- As the number of instructions increase, the circuit becomes bigger and hence more complicated.

Q) Consider a single control signal, C_5 which causes data to be read from the external data bus into the MBR.

Ans- Let us define two new control signals, P and Q, that have the following interpretation:

P Q = 00 Fetch Cycle
P Q = 11 Interrupt Cycle
P Q = 10 Execute Cycle
P Q = 01 Indirect Cycle.

$\overline{P} \to 1$
$\overline{P} \to 0$.

- C_5 can be defined as :-

$$C_5 = \overline{P} \cdot \overline{Q} \cdot T_2 + \overline{P} \cdot Q \cdot T_2$$

(Fetch) (Indirect)

Q) Referring to the figures above, depict the control signals required for the execution cycle of LOAD 200 and STORE 300.

Ans

LOAD	STORE
t_1 MAR ← IR address (C_8)	t_1 MAR ← IR address C_8
t_2 MBR ← Memory (C_5)	t_2 MBR ← AC C_{11}
t_3 AC ← MBR (C_{10}) (load)	t_3 Memory ← MBR C_{12}. (store)

2) Express the Boolean expression for each of the control signals.

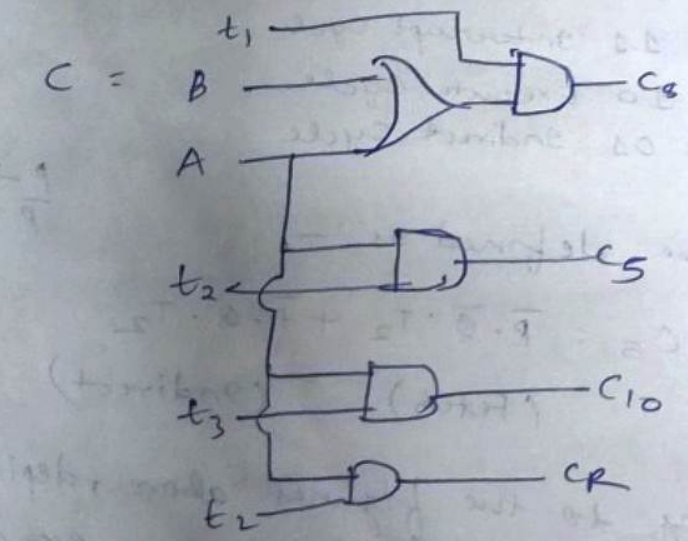
Ans LOAD → A and STOR - B

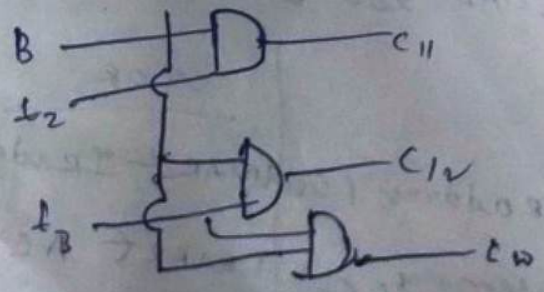
$C_8 = At_1 + Bt_1$
$C_5 = At_2$
$C_{10} = At_3$
$C_R = At_2$
} [2:5]

$C_{11} = Bt_2$
$C_{12} = Bt_3$
$C_W = Bt_3$

3) Implement the Boolean expressions for each of the control signals using minimum number of gates.

Ans- $C = B + A$





* Flow chart / Delay Element Method :-

1) Here the behaviour of the control unit is represented in the form of a flowchart.

2) Each step in the flowchart represents a control signal to be produced.

3) Once all steps are performed, the complete instruction gets executed.

4) Control signals perform Micro-operations, which require one T-states each.

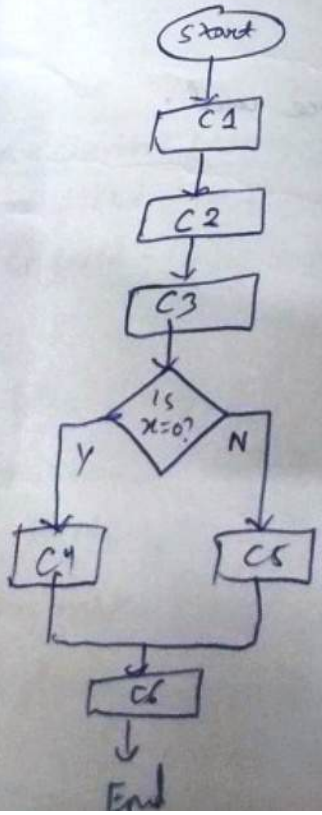
5) Hence, between every two steps of the flowchart, there must be a delay element.

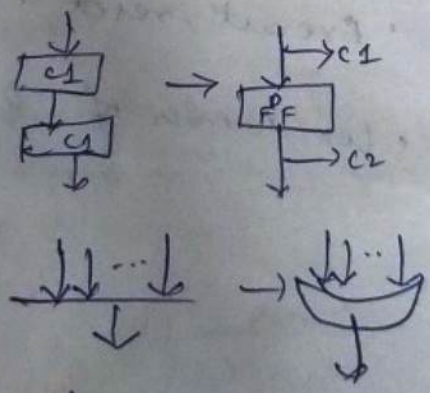
6) The delay is achieved by D-Flip-Flops. These D Flip-Flops are inserted between every ~~two~~ two consecutive control signals.

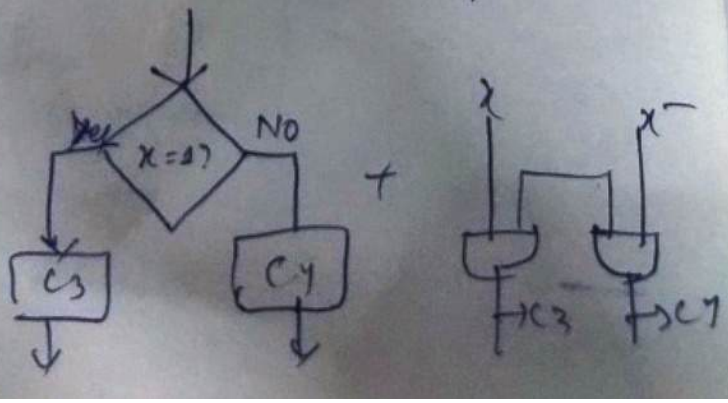
7) This method is called "one Hot Method".

8) On a multiple entry point, to combine two or more paths, we use an OR gate.

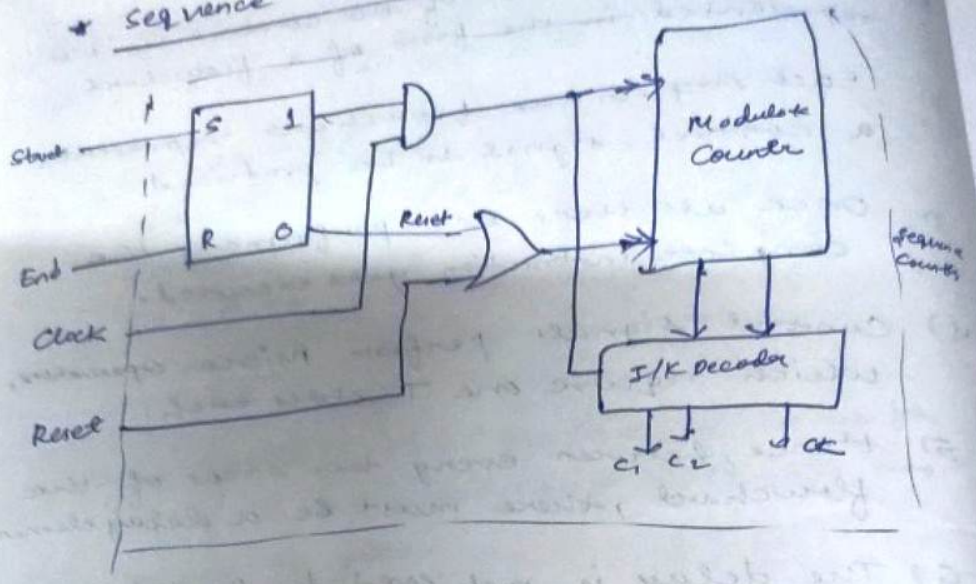
9) A decision box is replaced by a set of two complementing AND gates.







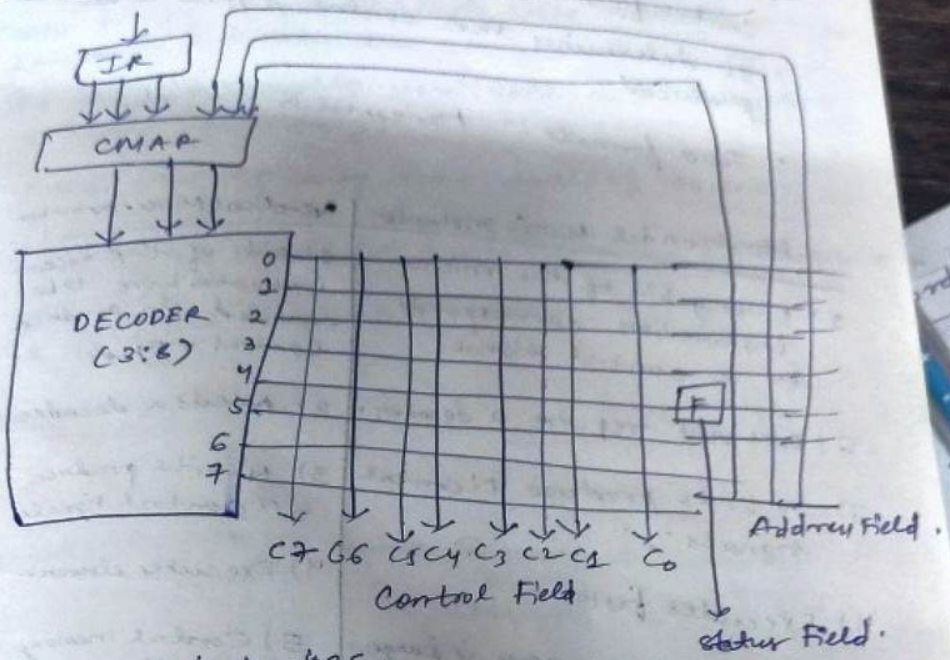
* Sequence counter method :-



- Most popular method.
- Logical approach of a flowchart.
- A flowchart is made and then converted to a circuit using AND and OR gates.
- We need a delay of 1 T-state between every two consecutive control signals.
- "Mod k" is used as output decoded.
- Execute, reset.
- Less number of D-Flipflops are used.

* Micro-programmed Control Unit implementation

* Wilkes' design for Micro programmed CU



- Uses micro-instructions.
- Performed by control-signals.
- It's called its micro-program.
- Address → IP → CMAR → Decoder → Address.

↓

Status Field.

CMAR - Control Memory Address Register.

Control field - indicates the control signals to be generated.

Address field - indicates the address of the next in.

Adv.

1) flexibility
2) Any change can be performed by simply changing the micro instruction.
3) This makes modification very easy.
4) Software can be easily debugged.

Drawbacks

1) control memory has to be present inside the processor.
2) This increases the cost of the system.
3) Adds an space to the control memory.

* Micro-instruction Format :-

- The main part of the micro-instruction is its control field.
- It determines the control signals to be produced.
- Two formats :- Horizontal or vertical.

Horizontal Micro-instruction	Vertical Micro-instruction
1) Every bit of the micro-instruction corresponds to a control signal.	1) Bits of the micro-instruction have to be decoded to produce control signals.
2) Does not require a decoder.	2) Needs a decoder.
3) N bits produce N control signals.	3) N bits produce 2^N control signals.
4) Executes faster.	4) Executes slower.
5) Control memory is large.	5) Control memory is small.

* Nano-Programming :-

- Combination of both vertical and horizontal form of the micro-instruction format is called Nano-Programming.
- We have a two level control memory.
- Horizontal - very wide → control memory - large.
- Vertical - narrow → control memory - small.
- Combination - Nano Programming.
- Two level control memory.
- First is fetched from main memory to IR.
- Using its opcode,

(3)

Memory

★ **Cache Memory :-**

- Cache memory is a small, high-speed volatile storage located between the CPU and RAM.
- It stores frequently used data and instructions to speed up processing.
- There are typically three levels of cache: L1, L2, L3.
- L1 being the closest to the CPU and the smallest but fastest.
- L3 being the largest but slower.

★ **Hierarchy List :-**

Processor ⟷ Cache ⟷ Primary Memory (RAM, ROM) ↔ Secondary Memory. (CD, DVD, Harddisk).

(1) **Registers :-**

- Registers are present inside the processor.
- They are basically a set of flip-flops.
- They ~~are~~ store data and addresses.
- They are very small in size typically just a few bytes.

(2) **Primary Memory :-**

- Main memory.
- Comprises of RAM and ROM.
- ROM is non-volatile. Typically ~~2MB~~ of 2MB-4MB in size.
- RAM is writable and is used for day-to-day operations. It's size is typically 4GB-8GB.

(3) Secondary Memory :-

- Increase the storage capacity, at low cost.
- Biggest component - Hard Disk.
- It is writeable as well as non-volatile.
- Size - 1 TB.

(4) CACHE Memories :-

- fastest form of memory as it uses SRAM (Static RAM).
- SRAM uses flip-flops hence is much faster than DRAM which uses capacitors.
- Size - 2MB - 8MB.
- If code and data are in the same cache then it is unified cache else its called split cache.
- L1 → Split Cache - 4 - 8 KB
- L2 → Unified Cache - 1 MB
- L3 → Unified Cache - 2 - 8 MB.

★ Memory Characteristics :-

1) Location :-

- On-chip - inside CPU
- Internal - on the motherboard. eg - RAM.
- External - connected to the motherboard. eg - Hard disk.

2) Storage Capacity :-

- Amount of data stored in the memory.
- NXM
  - N → No. of memory
  - M → Number of bits per memory location.

3) Transfer Modes :-

- Word Transfer :- If CPU needs some data, it will transfer only that amount of data.
- Block Transfer :- If CPU needs some data, it will transfer an entire block containing that data.

4) Access Modes :-

Memories can allow data to be accessed in two different ways :-

- Serial Access :- Here locations are accessed one by one in a sequential manner.
- Random Access :- Here all locations can be directly accessed in any random order.

5) Physical Properties :-

- Writeable - Contents of the memory can be altered.
- Non-writeable - Contents of the memory cannot be altered.
- Volatile - Contents of the memory are lost when power is switched off.
- Non-Volatile : Contents of the memory are retained when power is switched off.

6) Access Time

7) Reliability

8) Cost

9) Average Cost

10) Hit ratio

* Cache Memory Mapping :-

- Cache Memory Mapping techniques are used in computer architecture and design to determine how data is mapped to specific cache lines.

→ Direct Mapping (Direct - Mapped Cache) :-

- In a direct-mapped cache, each main memory block is mapped to a specific cache line.
- The mapping is determined by taking the least significant bits of the memory address and using them as the index to select a specific cache line.

→ Associative Mapping (Fully Associative Cache) :-

- each main memory can be placed in any cache line.

→ set - Associative :-

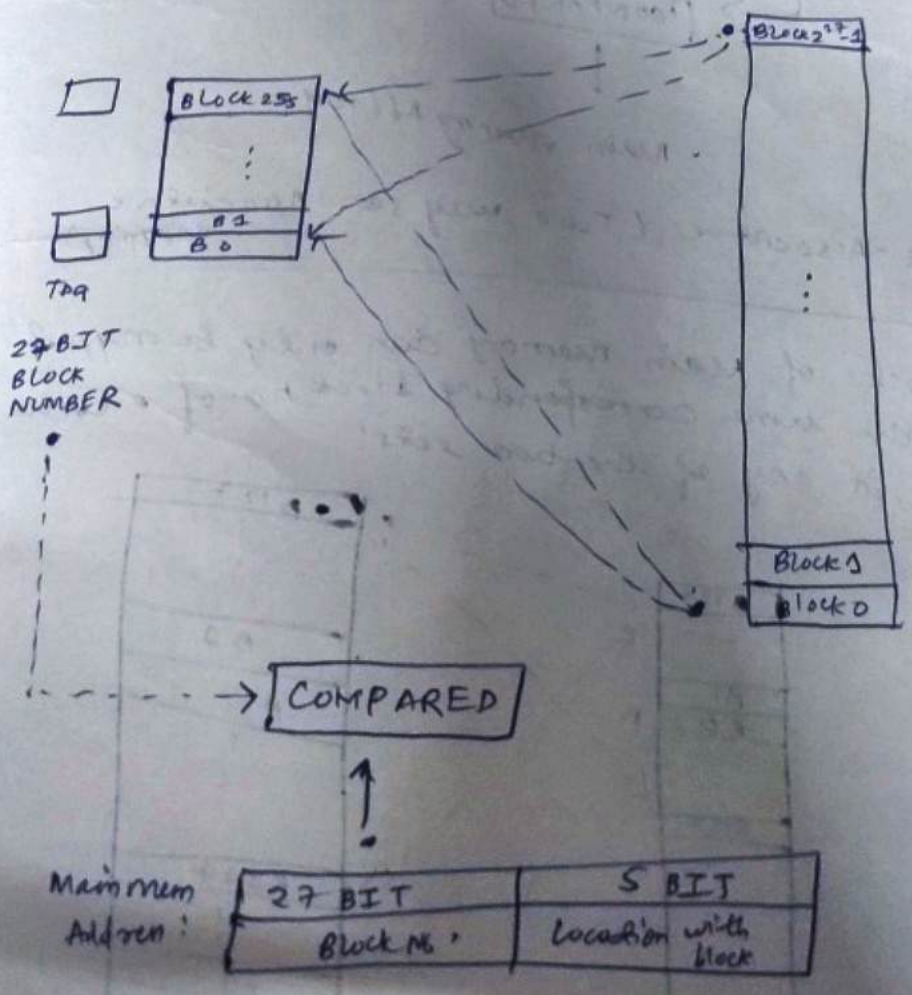
- the cache is divided into multiple sets.

* Associative Mapping (Fully Associative Mapping)

- During memory operations, Blocks are loaded from Main memory to cache Memory.
- Here, any block of main memory can be mapped at any available block of Cache Memory.
- Full caching is done.

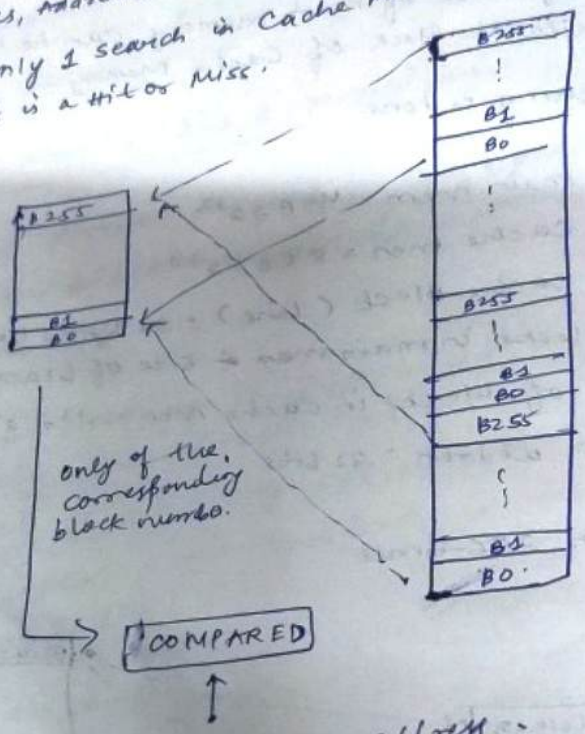
→ size of main mem = 4GB = 2^{32}
→ size of cache mem = 8 KB = 2^{13}
→ size of cache Block (Line) = 32 bytes (words) = 2^5
→ No. of blocks in main mem = size of block = 2^{27}
→ No. of blocks in cache mem = $2^{13} \div 2^5 = 2^8$
→ Main mem address = 32 bits.

- Searches for 256 times.



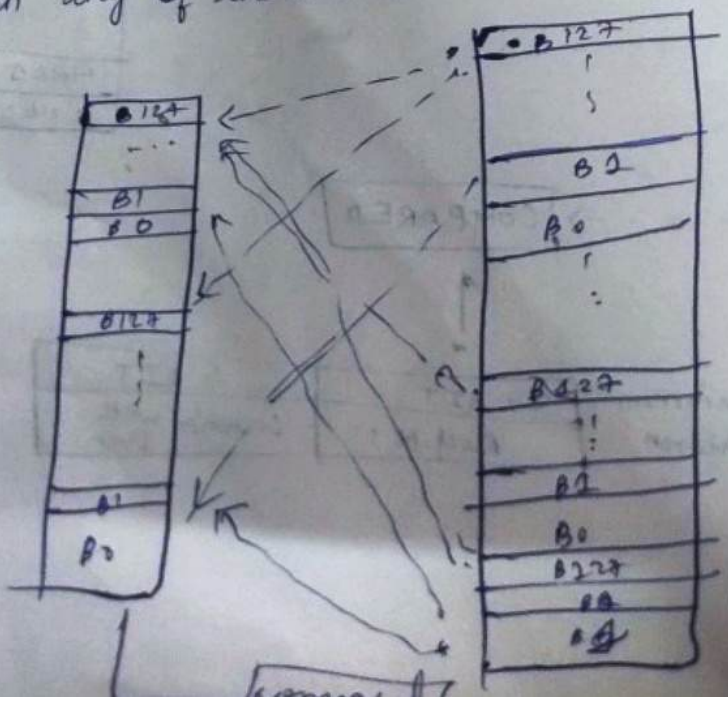
★ Direct Mapping (one way set Associative Mapping

- sizes, addresses all same as before.
- only 1 search in Cache Memory to know if it is a Hit or Miss.



★ Set-Associative (Two way set Associative Mapping).

- A block of Main memory can only be mapped into the same corresponding block no. of cache memory, in any of the two sets.



* Serial Communication (RS-232) :-

- RS-232, which stands for "Recommended Standard 232", is a standard for serial communication between devices.
- It defines the electrical characteristics and signals for serial communication between devices, such as computers, modems and other equipment.
- It is one of the oldest and most-widely used serial communication standards.

→ Key-Features of RS-232 :-

1. **Voltage Levels** :- RS-232 uses voltage levels to represent binary data. A logical "1" is typically represented by a negative voltage, while a logical "0" is represented by a positive voltage.

2. **Serial communication** :- Data is sent one bit at a time over a single communication line, making it a serial communication standard.

3. **Asynchronous Communication** :- RS-232 can operate in asynchronous mode, where the sender and receiver are not synchronized by a common clock signal.

4. **Handshaking** :- RS-232 supports various handshaking signals.

5. **Data Rate** :- RS-232 can support various data rates, commonly ranging from 300 to 115,200 bits per second.

6. **Cable Length** :- The max. cable length is relatively limited.

- Hardware Handshaking in RS-232 communication is a set of control signal that devices use to coordinate data transmission and reception.
- These handshaking signals help ensure that data is sent and received reliably.

→ some common handshaking signals in RS-232 communication include:

(1) RTS (Request to send)
(2) CTS (Clear to send)
(3) DTR (Data Terminal ready)
(4) DSR (Data set Ready).

- Software Handshaking in RS-232 involves two special characters for starting and stopping the communication.
- There characters are X-ON and X-OFF (Transmitter on and Transmitter OFF).
- When the receiver sends an X-OFF signal, the transmitter stops sending the data.
- The transmitter starts sending data only after it receives the X-ON signal.