Module III : Process. org^{n} [30 mks]

1. Exp Basic CPU org^{n} & register org^{n}
2. Exp. different Inst^{n} format
3. What is Inst^{n} cycle
4. Exp soft wired (µp reg) & Hardwired control Unit method.
5. SN on micro inst^{n} sequencing & execution
6. SN on Nanoprogramming
7. Explain flynn's classification
8. Explain diff pipeline Hazards.
9. Explain diff. stages of pipelining.

Module 1 and 2

1. Explain von-nuemann architecture in detail
2. Explain Evolution of computer / Generation of compi
3. Diff bet^{n} computer org^{n} & Computer Arch^{n}
4. Exp Architecture of 8086
5. Exp flag register of 8086.
6. Write SN on segmentation of 8086
7. Write SN on memory banking of 8086
8. How physical address is generated of 8086

} Any one

9. Exp addressing modes of 8086.
10. Exp diff pins of 8086
11. Exp minimum mode of 8086
12. Exp maximum mode of 8086
13. Diff between min & max mode

} Any one

14. SN on 8288 Bus controller
15. SN on mixed Language prog.
16. Explain concept of procedure & macro with example
17. Explain various string of 8086.
18. Program of 8086

Module VI - I/O organization [15 to 20 m]

1. SN on I/O module.
2. SN on I/O processor [8089]
3. Exp programmed I/O data transfer technique.
4. Exp Interrupt driven I/O data transfer tech.
5. SN on DMA.

Module V Memory organization [20 to 25 m]

1. Exp memory characteristics & measure parameters
2. SN on memory Hierarchy.
3. Concept of cache memory & L1, L2, L3 concept
4. Exp various memory mapping techniques.
5. S.N on cache coherency.
6. S.N on Interleaved & Associative memory.
7. S.N on prob. on page replacement algo.
8. Classification of primary & secondary mem.

[25 to 30m] Module IV Data representation & Arithmatic Algo

1. Problem on Booth's algo
2. —"— restoring division algo.
3. —"— NON.-"—
4. —"— or SN IEEE 754 floating pt format
5. Add & shift multiplication algo
6. 2's complement problem.

Module 1 - Ov

Explain C
Architectu
1. Compu
attribut
progra
2. comp
operati
that

8. 1. Eg
Inc
bit
2

Module 1 - Overview of computer Archr & Orgn

1) Explain Computer Organisation and Computer Architecture.

A 1. Computer architecture refers to the attributes of system visible to programmer.

2. Computer Organisation refers to operational units and their interconnection that realize the architectural specification.

B 1. Eg of computer architecture attribute include the instruction set, the number of bits used to represent various data types and I/O mechanism.

2. Eg of computer organization attribute include such as control signal, interface between computer and peripherals.

C 1. Computer Architecture design issue whether the instruction set will have multiply instruction or not?

2. Computer Organization issue whether instruction will be implemented by special multiply or by mechanism that makes repeated use of add units of system

* Many manufacture offer same architecture design but with different organization.

2) Explain von-neumann architecture.

Note: History of computer

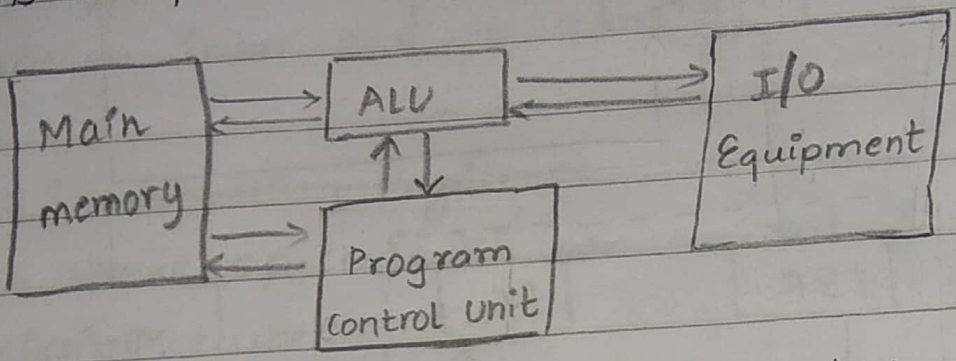
The first generation computer (Vaccum Tube)

ENIAC (Electronic Numerical Integerator and computer)

- It is designed by John brothers. It is first general purpose electronic digital computer The project was ~~for~~ response to USA war time need
- The Army's Balistics Research Lab & trajectory tables for new weapons, Eckert proposed to build a general computer using vaccum tubes used for BRL application
- In 1943 work begon ENIAC. The resulting computer is of wieght 30 tons and containing more than 18000 vaccum tubes and occupying 15000 sq.ft space and capable of 5000 additions/sec. The drawback of ENIAC was that it had to be programming manually by setting switches & plug and unplugged the cables.

Von neumann model

As was mentioned the task of entering and altering programs for ENIAC was extremely tedious. In 1946, von neumann and his collegues began the design of new stored program computer reffered as IAS computer.



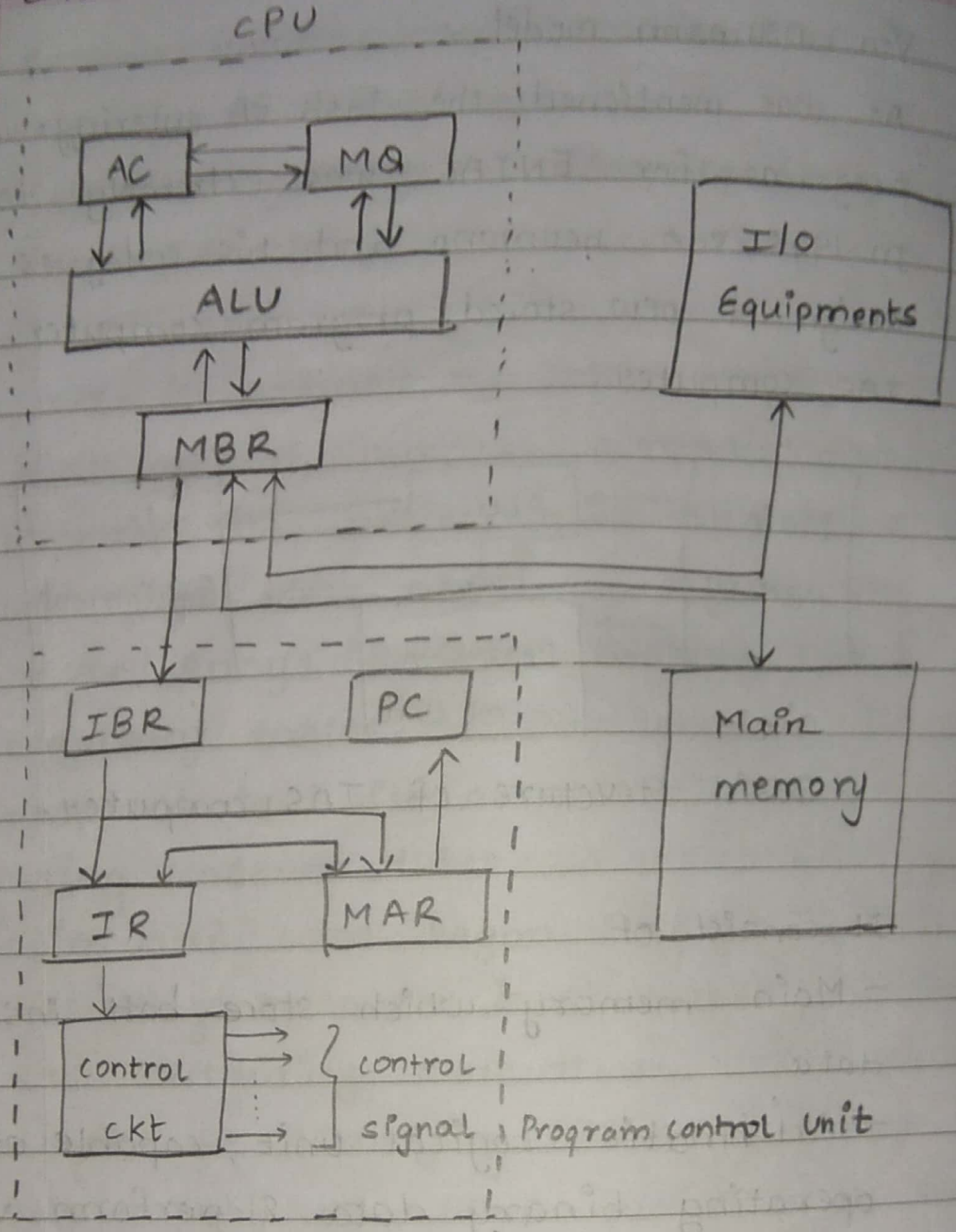
Basic Structure of IAS computer.

It consist of

- Main memory, which store both instruction & data
- Arithmatic Logical unit, capable of operating binary data & perform various arithmatic and logical operation
- Control unit, which interprets the instruction in memory.
- An I/O Equipment, operated by control unit

With rare exception, all of today's computer have this same general structure and function.

Extended structure of IAS computer



The memory of IAS consist of 1000 storage location & 40 bits each.
(Both instruction and data stored)
The control unit fetch the instruction from memory executing them. In figure, control unit and CPU contains storage location of register defined as follows:

MBR (Memory Buffer Register)
It is also called as MDR. It contains data from memory.

MAR (Memory Address Register)
It specifies the address of data to be written from and to memory.

IR (Instruction Register)
It stores op-code of ~~con~~ current instruction being executed.

IBR (Instruction Buffer Register)
It is used to store the data temporarily

PC (Program Counter)
It contains address of next instruction, which fetch from memory.

AC (Accumulator) and MQ (Multiplier Quiotent)
Accumulator stores the ~~operation~~ operand (data) and result of ALU for eg the result of mutiplying two 40 bit number is 80 bits hence MSB goes to AC and LSB goes to MQ after execution

Perform unsigned multiplication using Add and shift method.

1. 11 X 13

11 → 1 0 1 1

13 → x 1 1 0 1

1 1 0 1 1

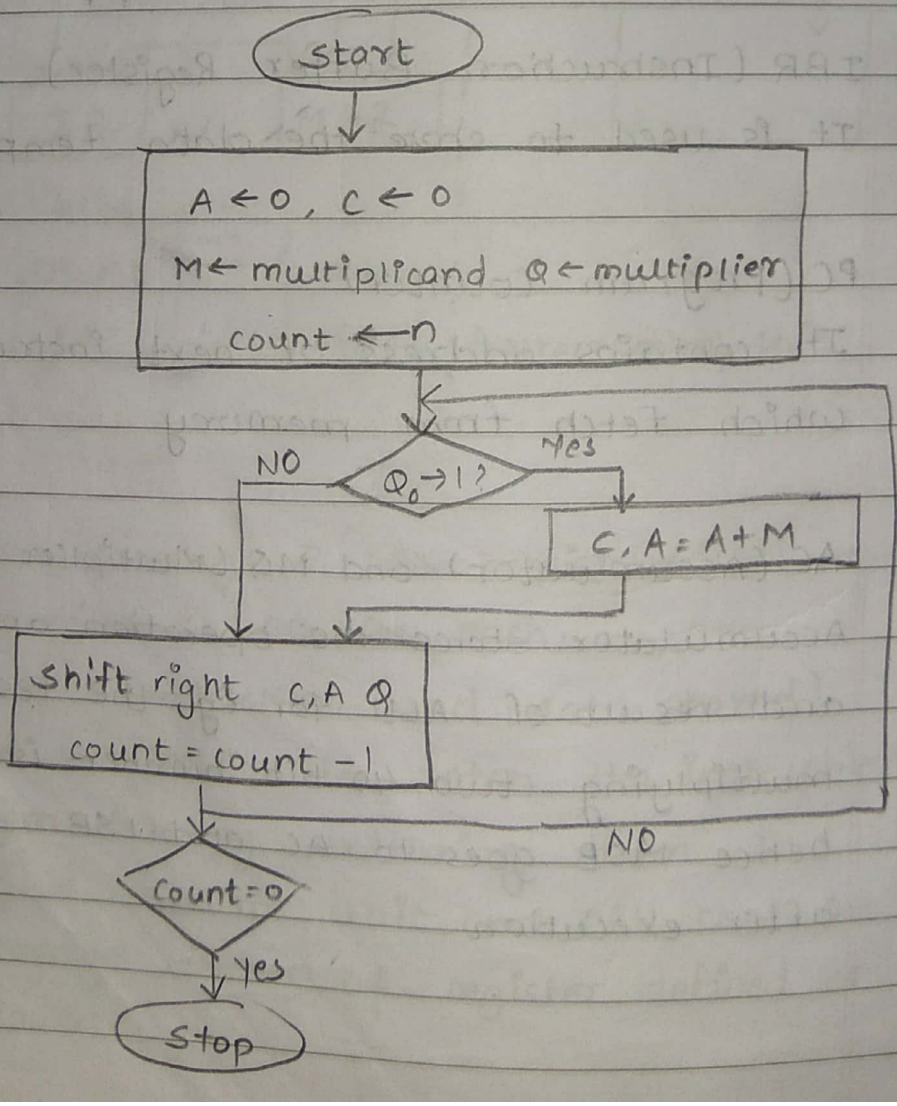
1 0 0 0 0 x

1 1 0 1 1 x x

1 0 1 1 x x x

1 0 0 0 1 1 1 1

11 X 13 → 143



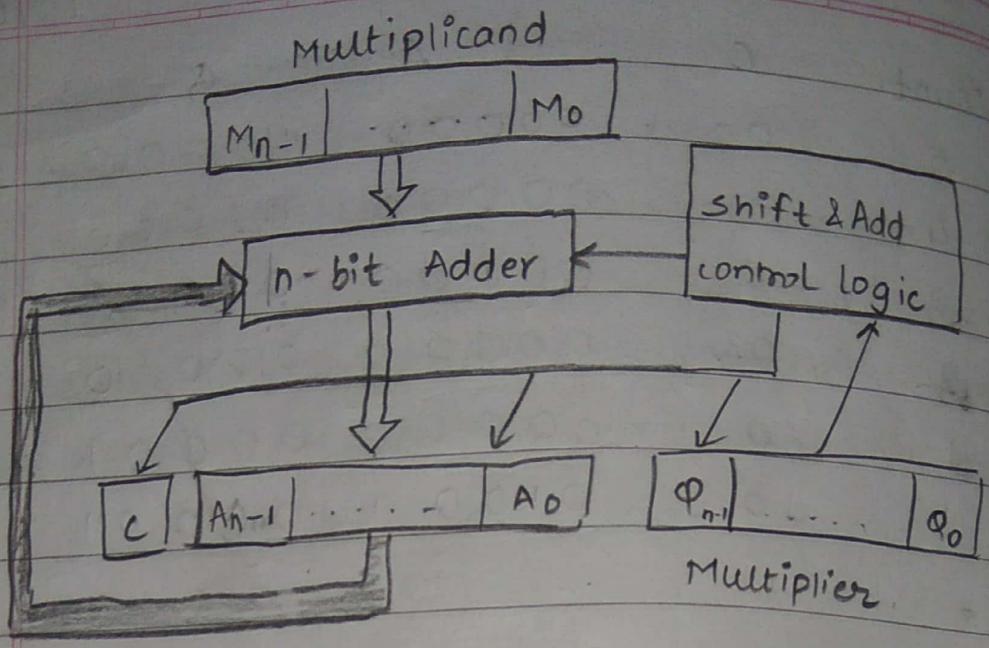
9X16

Count	C	A	Q
5	0	00000	10000
4	0	00000	01000
3	0	00000	00100
2	0	00000	00010
1	0	00000	00001
	0	01001	00001

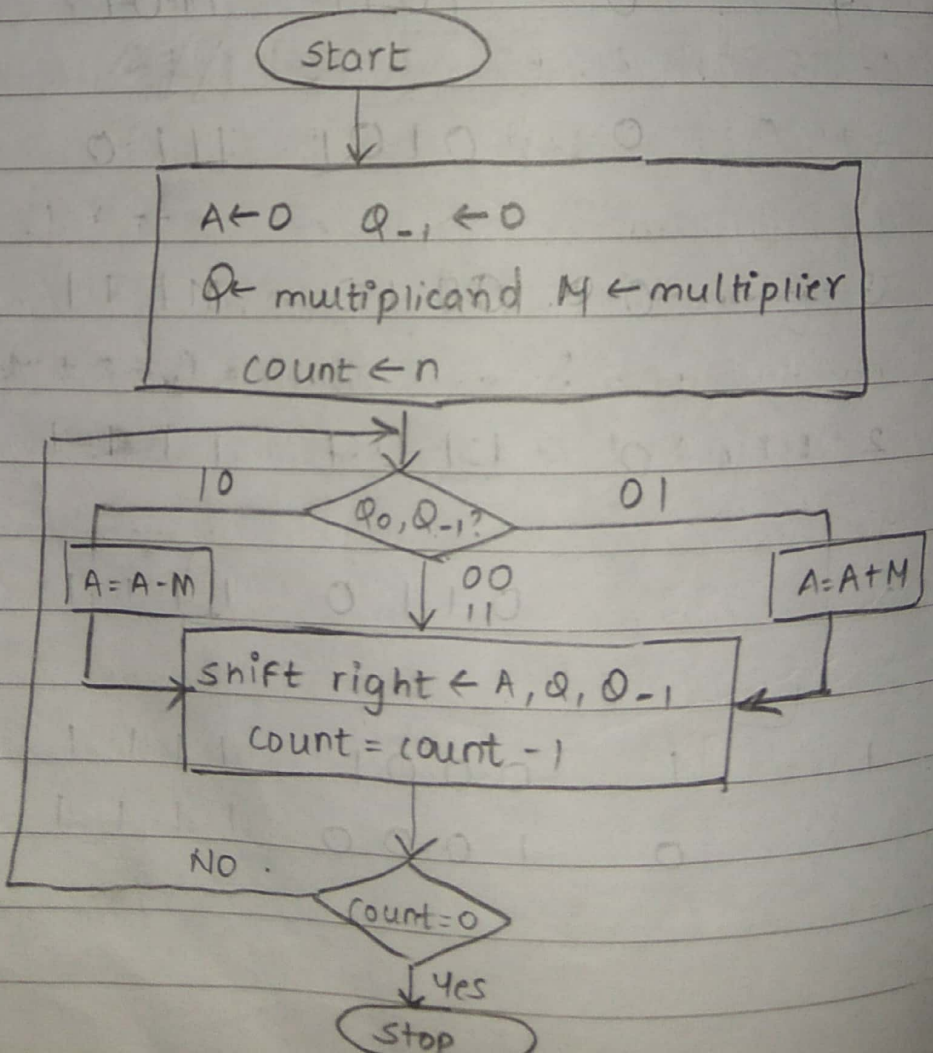
1001
10000

13 X 11

Count	C	A	Q
	0	0000	1101
4	0	1011	1101
	0	0101	1110
3	0	0010	1111
2	0	1101	1111
		0110	1111
1	1	0001	1111
	0	1000	1111



Booth's Algorithm (signed) for 2's complement method.



3 × 7.

cycle	A	Q	Q_{-1}	comment
	0 0 0 0	0 0 1 1	0	Initial
4	1 0 0 1	0 0 1 1	0	$A = A - M$ ($Q_0Q_{-1} = 10$)
3	1 1 0 0	1 0 0 1	1	shift right
2	1 1 1 0	0 1 0 0	1	shift right
	0 1 0 1	0 1 0 0	1	$A = A + M$
1	0 0 1 0	1 0 1 0	0	SR
	0 0 0 1	0 1 0 1	0	SR

Solve 4 × 7 ~~for~~ B using Both algorithm.

Count	A	Q	Q_{-1}	Comment
	0000	0 1 0 0	0	Initial
4	0 0 0 0	0 0 1 0	0	SR
3	0 0 0 0	0 0 0 1	0	SR
2	1 0 0 1	0 0 0 1	0	$A = A - M$
	1 1 0 0	1 0 0 0	1	SR
1	0 0 1 1	1 0 0 0		$A = A + M$
	0 0 0 1	1 1 0 0	0	SR

4 → 0100

7 → 0111

A = A + (−M)

−7 → 0111

↓

1000

+ 1

1001

1100

+ 0111

Scanned by CamScanner

3×-7.

Count	A	Q	Q-1	Comment
	0000	0011	0	Initial.
4	0111	0011	0	A = A-M
	0011	1001	1	SR
3	0001	1100	1	SR
2	1010	1100	1	A = A+M
	1101	0110	0	SR
1	1110	1011	0	SR

$3 \rightarrow 0011$

$-7 \rightarrow 0111$

1000

$+ 1$

1001

$A = A + (-M)$

$= A + (1-7)$

$1110 + 0111 = 0001$

$0001 + 1001 = 1010$

Ans.

$11101011 \rightarrow 00010100 \rightarrow 10101$

$(-3) \times 7$.

count	A	Q	Q-1	comment
	0000	0111	0	Initial.
4	0011	0111	0	A = A-M
	0001	1011	1	SR
3	0000	1101	1	SR
2	1101	1100	1	A = A+M
	1110	1110	0	SR
1	1111	0111	0	SR

0011

1100

$+ 1$

1101

0111

1000

$+ 1$

1001

$+ 0111$

0000

1101

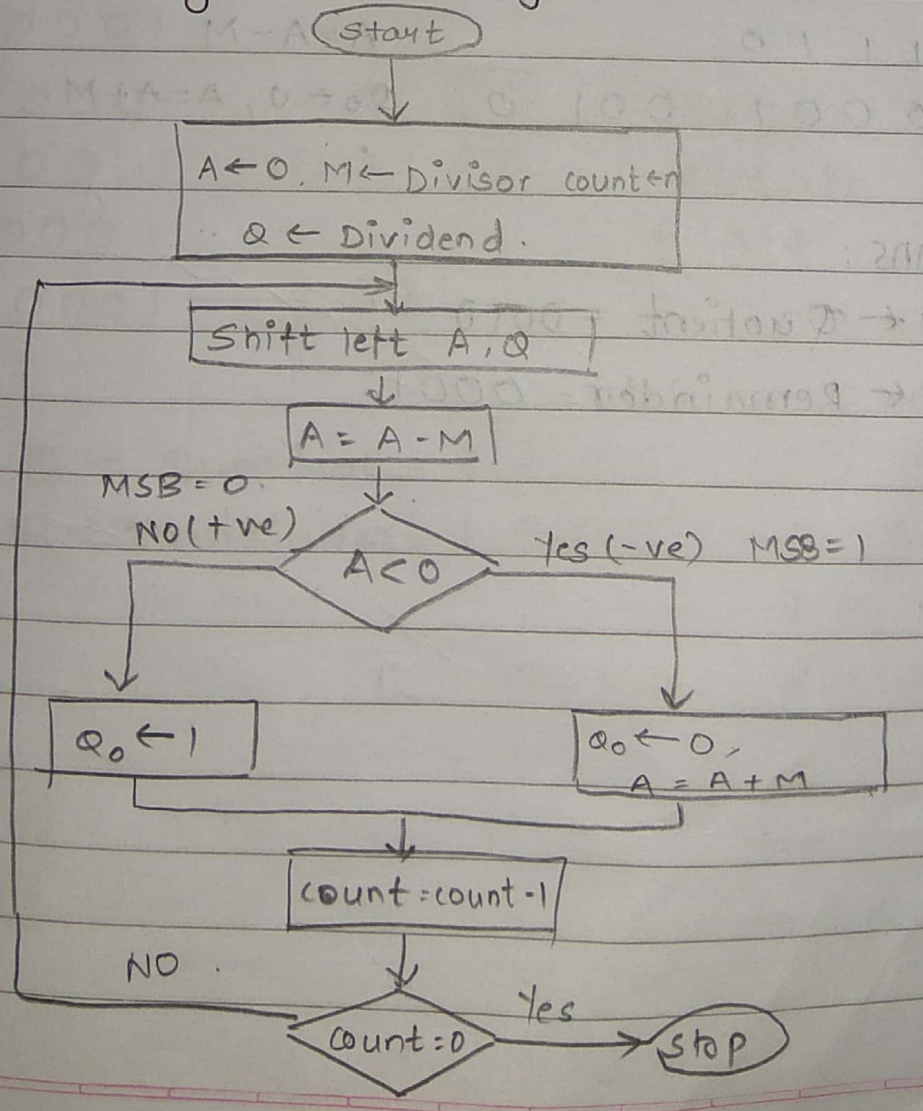
solve (-3)x(-7) using Both algorithm.

count	A	Q	Q_{-1}	Comment
	0000	1001	0	Initial.
4	0011	1001	0	A = A-M
	0001	1100	1	SR
3	1110	1100	1	A = A+M
	1111	0110	0	SR
2	1111	1011	0	SR
1	0010	1011	0	A = A-M
	0001	0101	0	SR
0				

Ans = 21

0001
+1101
1101

Restoring division algorithm.



Q ← Divident = 7

Q Solve 7 ÷ 3 M ← Divisor = 3.

Count	~~Coun~~ A	Q	Comment	
	0000	0111	Initial	0000
4	0000	1111	Shift left	+ 1101
	1101		A = A − M	1101
	0000	1111	$Q_0 \leftarrow 0$, A = A + M	0001
3	0001	1110	Shift left	1101
	1110		A = A − M	1110
	0001	1110	$q_0 \leftarrow 0$, A = A + M	
2	0011	1100	Shift left	
	0000		A = A − M	
	0000	1001	$Q_0 \leftarrow 1$	0001
1	0001	0011	Shift left	1101
	1110		A = A − M	1110
	0001	0010	$Q_0 \leftarrow 0$, A = A + M.	

Ans:

Q ← Quotient = 0010

A ← Remainder = 0001.

Q. Solve $7 \div 2$

Q ← Divident = 7

M ← Divisor = 2

count	A	Q	Comment
	0000	0111	Initial.
4	0000	111 □	shift left
	1110		A = A - M
	0000	111 [0]	$Q_0 \leftarrow 0$, A = A + M
3	0001	110 [1]	shift left
	1111		A = A - M
	0001	110 [0]	$Q_0 \leftarrow 0$, A = A + M
2	0011	100 □	shift left
	0001		
	0001	100 [1]	$Q_0 \leftarrow 1$
1	0011	001 □	shift left
	0001		A = A - M
	0001	0011	

$$\begin{array}{r} 0010 \\ 1101 \\ +\quad 1 \\ \hline 1110 \end{array}$$

$$\begin{array}{r} 1110 \\ +0010 \\ \hline 0000 \end{array}$$

$$\begin{array}{r} 0001 \\ +1110 \\ \hline 1111 \end{array}$$

$$\begin{array}{r} 0011 \\ +1110 \\ \hline 0001 \end{array}$$

$$\begin{array}{r} 0011 \\ +1110 \\ \hline 0001 \end{array}$$

Q ← Quotient = 3

R ← Remainder = 1.

Q. Solve $16 \div 5$

Q ← Divident = 16

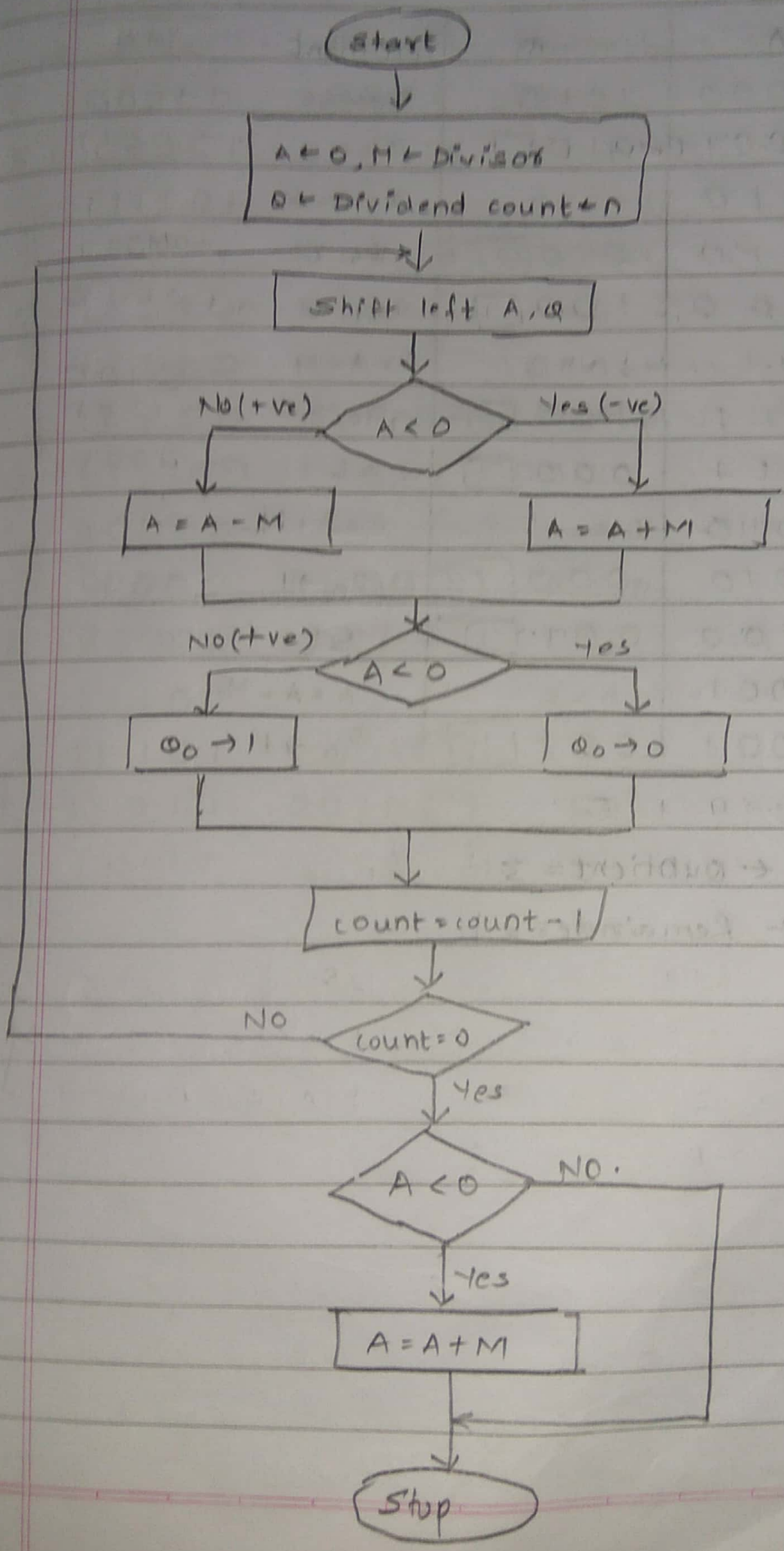
M ← Divisor = 5

count	A	Q	comment
	000000	010000	
6	000000	100000	Shift left
	111011		A = A - M
	000000	100000	$Q_0 \leftarrow 0$, A = A + M
5	000001	000000	Shift left
	111100		A = A - M
	000001	000000	$Q_0 \leftarrow 0$, A = A + M
4	000010	000000	SL
	111101		A = A - M
	000010	000000	$Q_0 \leftarrow 0$, A = A + M
3	000100	000000	SL
	111111		A = A - M
	000100	000000	$Q_0 \leftarrow 0$, A = A + M
2	001000	000000	SL
	000111		A = A - M
	000111	000001	$Q_0 \leftarrow 1$
1	001110	000010	SL
	000001		A = A - M
	000011	000011	$Q_0 \leftarrow 1$

Q ← Quotient = 3

R → Remainder = 1

Non restoring division algorithm (unsigned)



Solve $10 \div 3$

$Q \leftarrow 10$ $\quad M \leftarrow 3$.

Count	A	Q	Comment.
	0000	1010	Initial
4	0001	010□	SL
	1110		A = A − M
	1110	0100	$Q_0 \rightarrow 0$
3	1100	100□	SL
	1111		A = A + M
	1111	1000	$Q_0 \rightarrow 0$
2.	1111	000□	SL
	0010		A = A + M
	0010	0001	$Q_0 \rightarrow 1$
1	0100	001□	SL
	0001		A = A − M
	0001	0011	$Q_0 \rightarrow 1$

0011
1100
$+\ \ 1$
1101

1100
$+0011$
1111
$+0011$
0010

0100
$+1101$
0001

$Q_0 \leftarrow$ Quotient $= 3$

$A \leftarrow$ Remainder $= 1$

10011

Solve 19 ÷ 4

Count	A	Q	comment
	00000	10011	Initial.
5	00001	0011 □	SL
	11101		A = A − M
	11101	0011 0	$Q_0 \leftarrow 0$
4	11010	0110 □	SL
	11110		A = A + M
	11110	0110 0	$Q_0 \leftarrow 0$
3	11100	1100 □	SL
	00000		A = A + M
	00000	1100 1	$Q_0 \leftarrow 1$
2	00001	1001 □	SL
	11101		A = A − M
	11101	1001 0	
1.	11011	0010 □	SL
	11111		

00100
11011
+ 1
11100

11010
+00100
11110

11100
00100
00000

00001
+11100
11101

11011
+00100

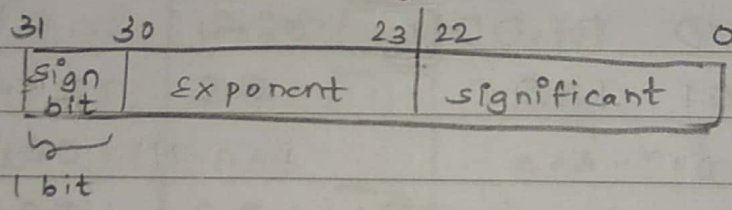
IEEE 754 floating point representation.

1) 32 bit format (Single precision) 127
2) 64 bit format (Double precision) 1023

Note: $(1.f) \times 2^{e-127}$

It includes.
- Sign bit (-ve = 1 and +ve = 0)
- Exponent
- Significant (Mentissa)

1) 32 bit form (single ~~bit~~ precision format)



Eg. (1) - 209.125.

Step 1. Convert into binary

2	209	
2	104	1
2	52	0
2	26	0
2	13	0
2	6	1
2	3	0
	1	1

1

$.125 \times 2 = .250 \rightarrow 0$

$.250 \times 2 = .500 \rightarrow 0$

$.500 \times 2 = 1.000 \rightarrow 1.$

$(209.125) = (11010000 1.001)_2$

Step 2: findout True exponent.

$1101000001.001 \times 2^0$

$1.101000001001 \times 2^7$

Here 7 is True Exponent.

Step 3: Exponent = Bias component + True Exponent

$= 127 + 7$

$= 134$

2	134	
2	67	0
2	33	1
2	16	1
2	8	0
2	4	0
2	2	0
	1	0

$(134)_{10} = (10000110)_2$

Step 4.

sign bit	Exponent	significant
1	10000110	1010001001

8086 Microprocessor.

80X86 family overview.

chip / µp	Year	Data Bus (Bit)	Address Bus (Bit)
4004	1971	4	8
8008	1972	8	8
8080	1974	8	16
8085	1977	8	16
8086	1978	16	20
8088	1978-79	8	20
80186	1982	16	20
80286	1983	16	24
80386 DX	1986	32	32
80386 SX	1988	16	24
80486 DX	1989	32	32
80486 DX	1989	32	32

Features of 8086 µP

(a) It is an 16-bit microprocessor i.e 8086 µP can read or write or perform any arithmatic and logical operation on 16-bit data at a time

(b) It is 40 pin IC or device.

(c) It is capable of executing about 0.33 MIPS (Million Instⁿ / sec)

(d) It has 16 bit data bus.

(e) It has 20-bit address ~~bus~~ lines so 8086 can access upto 2^{20} = 1 MB of memory.

(f) At a time it can work with four 64-KB segment i.e CS, DS, ES and SS.

(g) It ~~regd~~ reqd +5V supply

(h) It works on max 10MHz clock freq & min 5MHz clock freq.

(i) It contains 16 bit flag register.

(j) It support multiprocessing

(k) It can operate in single mode (minimum mode) and multiprocessor mode (maximum mode)

(l) It provides ~~66~~ 6 byte of inst^n queue for pipelining.

(m) It has memory bank concept.

(n) It has 256 s/w interrupts.

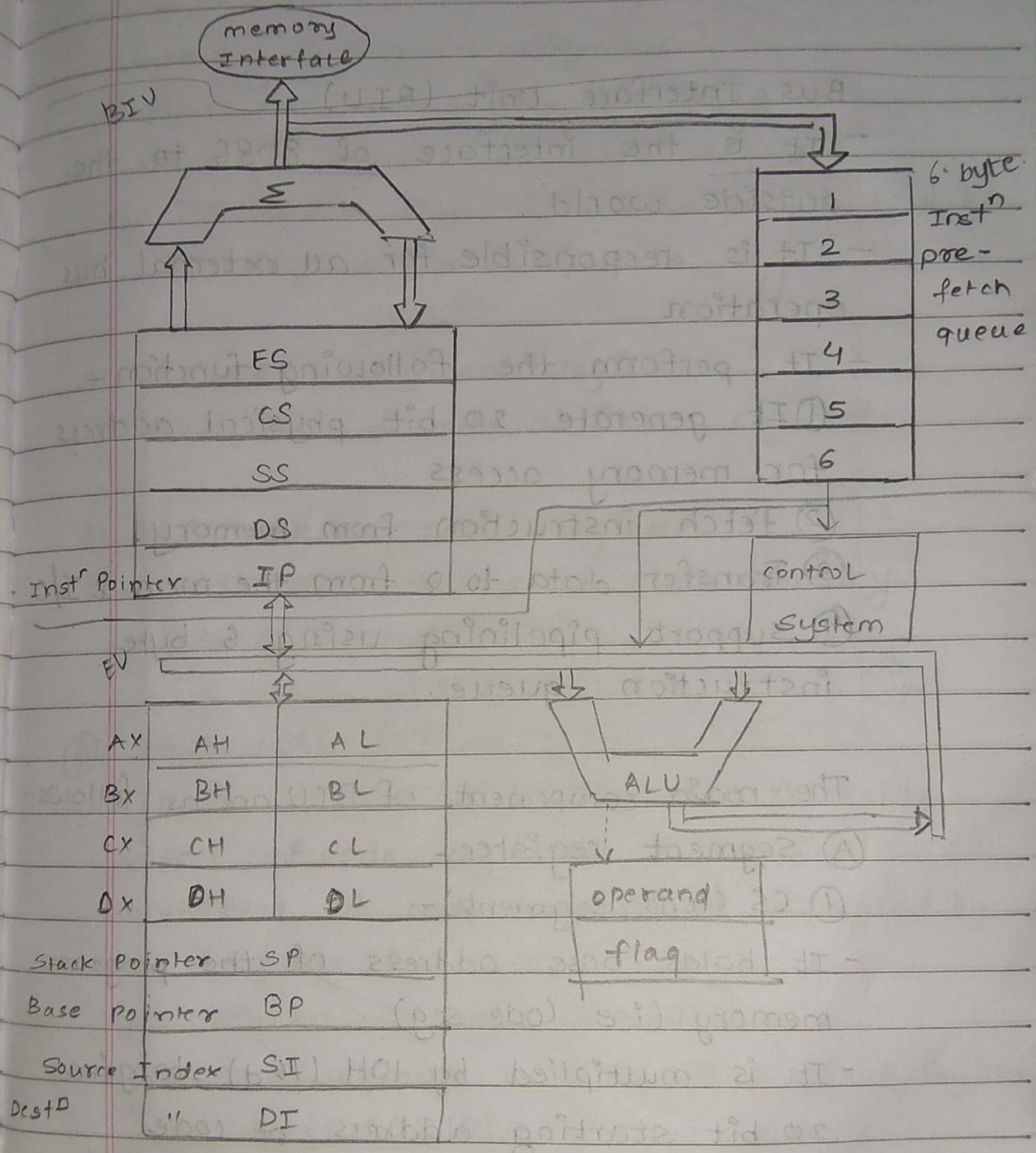
CS → IP

DS → BX/SI

ES → DI

SS → SP/BP

Architecture of 8086 µp V.V. Imp (10M)



8086 is a 16 bit microprocessor. It can perform 16 bit internal processing of data. It is internally divided into two independent functional unit i.e BIU

(Bus Interface Unit) and EU (Execution Unit)

Bus Interface Unit (BIU)

- It is the interface of 8086 to the outside world.
- It is responsible for all external bus operation
- It perform the following function -
  (1) It generate 20 bit physical address for memory access.
  (2) fetch instruction from memory.
  (3) Transfer data to & from the mem & I/O
  (4) support pipelining using 6 byte instruction queue.

The main component of BIU are as follows.

(A) Segment register -

(1) CS (Code segment) -

- It holds base address of the program memory (i.e Code seg)
- It is multiplied by 10H (16 d) to give 20 bit starting address of code segment

e.g If CS = 1234H then cs x 10H
= 12340 H → starting address of CS.

2. DS (Data segment) register:
- Hold base address for the ~~memory~~ data memory (i.e Data segment)
- It is multiplied by 10H to give starting address of data segment

eg. DS = 1111h then DS x 10H = 11110H will be starting address of data seg.

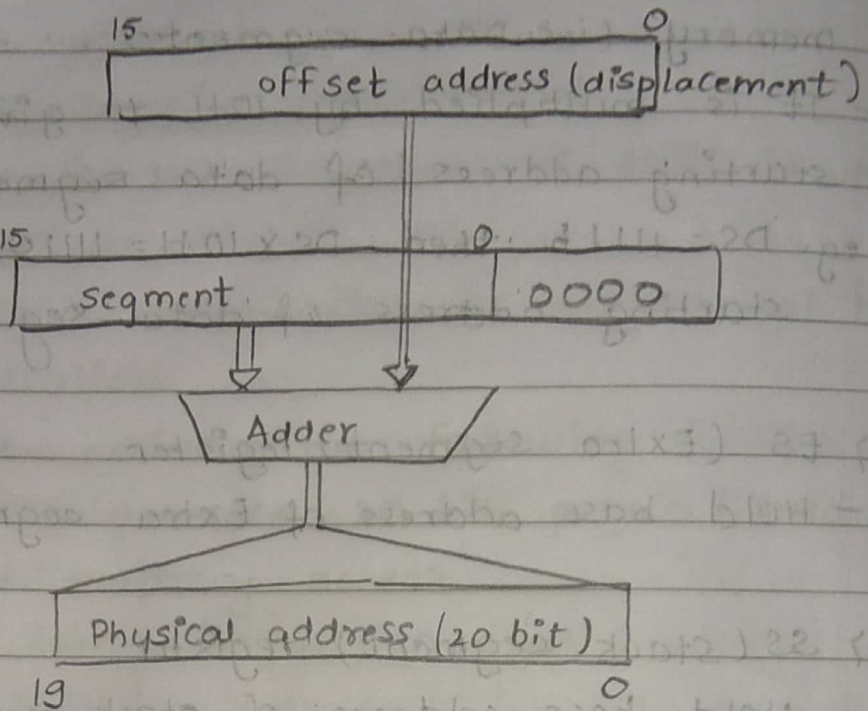
3) ES (Extra segment) register.
- Hold base address of Extra segment.

4) SS (Stack segment) register.
- Hold base address of stack memory (Stack Segment)

(B) Instruction Pointer (IP)-
- It is a 16 bit register and It holds offset of code segment
- Address of code segment is calculated by (CS x 10) + IP

i.e If CS = 1000 and IP = 2222
= (segment register x 10) + offset register
= (1000 x 10) + 2222
= 12222h 20 bit physical address of program

C) 20 bit physical address generation mechanism:



(D) 6 byte prefetch Queue -

- It is a 6 byte FIFO RAM used to implement pipelining.
- The BIU fetch the next "six instruction Byte" from code segment into queue.
- To fetch inst^n bytes atleast 2-bytes of queue must be empty
"fetching the next instruction while executing the current inst^n is called pipelining"
- pipelining fails when a branch occurs
- During a branch the prefetch queue is cleared & the reload with the prefetched inst^n bytes from the new location

F1	E1	F2	E2	F3	E3	F4

Non pipelined processor (i.e 8085)

t →

F1	E1	E2	E3	E4	E5
	F2	F3	F4	F5	

pipelined processor (i.e 8086)

2) Execution Unit.

- It fetches instruction from the instruction queue.
- It performs arithmatic, logical, decision making operations.
- It ~~access~~ sends request signals to BIU to access the external module.

The main component of EU are -

(A) General purpose registers:

- 8086 has four 16 bit general purpose register i.e AX, BX, CX and DX.
- These are available to programmer for arithmatic and logical operations.
- Each of these registers can be divided into two 8-bit register such as AH-AL, BH-BL, CH-CL and DH-DL.

(B) Special purpose register

(I) Stack Pointer (SP)

- It is 16 bit register used to hold offset

address of the top of the stack in the stack segment.

- It is used by inst^n like PUSH, POP, CALL, RET etc

(II) Source Index (SI)

- It is 16 bit register used to hold offset address of data segment

(III) Destination Index (DI) -

- It is 16 bit register used to hold offset of Extra Segment.

(IV) Base pointer (BP)

It is 16 bit register to hold offset address of stack memory.

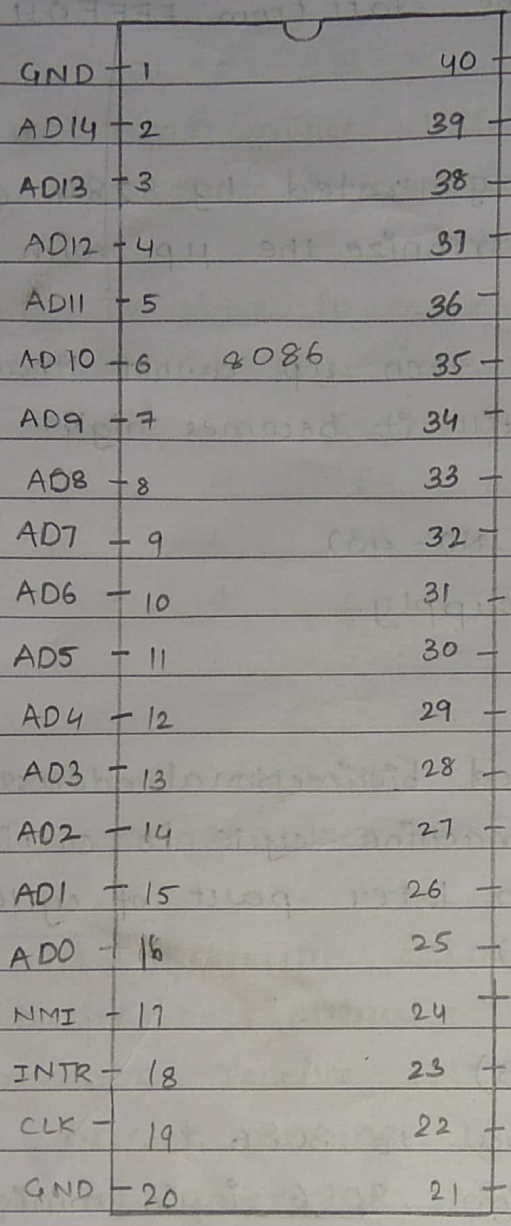
(C) ALU:

It can perform 8 bit or 16 bit arithmatic and logical operation.

(D) Operand

- It is a 16 bit register used to hold the operand or data temporarily
- It is not available for programmer.

Pin configuration of 8086.



1) CLK
- This is input line.
- An External clock generator IC 8284 provides the clock signal
- 8086 required, single phase 33% duty cycle.

2) RESET (P no-21)
- This is Reset input line
- The 8284 clock generator provides it
- It clears flag reg & Instruction Queue

- It also clear DS, ES, SS and IP and set all bits
- Hence after Reset 8086 start from FFFF0H.

3) READY (PNO-22)
- This If signal is also generated by 8284 clock gene and used to synchronize the µp with slower peripherals.
- If READY is low then µp cannot transfer dat with peripherals, till it becomes high.

4) Vcc (Pno-1 & 20, P.NO-40)
- used for power supply.

5) AD0 - AD15
- These are multiplexed bidirectional address/data Bus
- During start of machine cycle it carries address (A0 - A15) and in later part of cycle it carries data (i.e D0 - D15)

6) MN/$\overline{MX}$ (P.No-33)
- This is input signal to 8086.
- If the signal is high, 8086 is in minimum mode i.e If uniprocessor mode & if the signal is Lo 8086 is in maximum mode, i.e In multiproccesor mode.

1) $\overline{TEST}$ (PNO-23)
- It is active LOW input line dedicated for 8087 co-processor.
- In maximum mode whenever the co-processor is bu It makes the pin HIGH.
- If $\overline{TEST}$ pin is High, the µp (8086) enter into w state it becomes low.

* The min mode connected to GND.

A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3

- These are time multiplexed & status lines.
- During T_1 - state it carries upper 4 bit of Address bus to identify memory segment
- During T_2 - T_4 state it carries status signal (S_3 to S_6)
- Hex S_5 is interrupt enable signal.

S_4	S_3	segment
0	0	ES
0	1	SS
1	0	CS or None
1	1	DS.

9. INTA (Interrupt Request)
This is a maskable, level triggered, Low priority interru

10. NMI $\overline{INTA}$ (Interrupt Acknowledge)
- It is active low output.
- When processor receive INTR signal, the processor complete current machine cycle & Acknowledge the Interrupt by generating this signal.

11. NMI (Non maskable Interrupt)
- This is non-maskable, edge triggered, High priority Interrupt.

12. $\overline{BHE}/S_7$ (Bus High Enable/Status)
- The bus High Enable signal is used to indicate the transfer of data over High order (D_8 to D_{15}) data bus as shown in table ~~shown~~ below.

$\overline{BHE}$	A_0	
0	0	Whole word from Both (Even & Odd) address
0	1	Upper byte from odd address ($D_8 - D_{15}$)
1	0	Lower byte from Even address ($D_0 - D_7$)
1	1	None

- $\overline{BHE}$ in conjuction with A_0 determines whether a byte or word will transfer from or to memory.

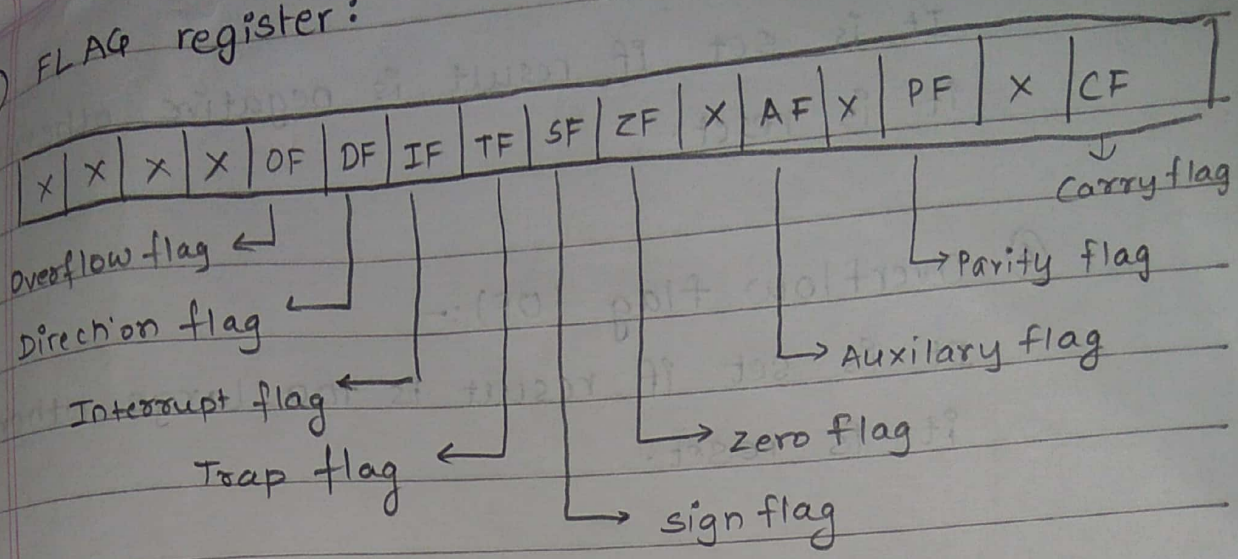
3) ALE (Address Latch Enable)
- It will indicate the availability of address on $AD_0 - AD_{15}$ if ALE is High.
- When $AD_0 - AD_{15}$ have valid address ($A_0 - A_{15}$) at that time this pin is High & when $AD_0 - AD_{15}$ have valid data ($D_0 - D_{15}$) at that time this pin is Low.

14) $\overline{WR}$ (write)
- It is active low signal is issued by the processor to write data to memory or I/O device.

15) $\overline{RD}$ (read)
This is active low output signal, when 8086 read data from memory or I/O device.

16) M/$\overline{IO}$
- This is used by processor to distinguish memory & I/O device.
- When this signal is High, memory is accessed & when this signal is low the I/O device is accessed.

(E) FLAG register:



- It has 9 flags. These flags are of two types 6 status (condition) flag & 3 control flag
- The control flags are used to control certain operation.

* Status flag:

① Carry flag (CF):
It is set whenever carry out of MSB (i.e D7 Bit), otherwise it is reset.

② Parity flag (PF).
It is set if result has even parity (1's) otherwise reset.

③ Auxiliary carry (CF)
It is set if carry is generated between D_3 and D_4 bit otherwise it is reset.

④ Zero flag (ZF): It is set if result is zero otherwise it is reset.

⑤ Sign flag (SF):
It is set if result is negative otherwise it is reset.

⑥ Overflow flag (OF):-
It is set if result is too large otherwise it is reset.

* control flag
① Trap flag (TF)-
- It is used to set Trace mode ie start single stepping mode.
- It is used for Debugging purpose.

② Interrupt flag (IF):-
- It is used for Enable & disable the interrupt.

③ Direction flag (DF):-
- If it is set the SI and DI are in auto decrement mode during sring operation otherwise in auto increment mode.

$M/\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Action.
1	0	1	Mem. Read
1	1	0	Mem. Write
0	0	1	I/o Read
0	1	0	I/o write.

~~So, S, ,S2~~

Addressing mode

1) Register Addressing mode

eg: MOV AX, BX.

2) Immediate addressing mode

eg: MOV AL, 02H

3) Direct Memory addressing mode

eg: MOV [5000], AL

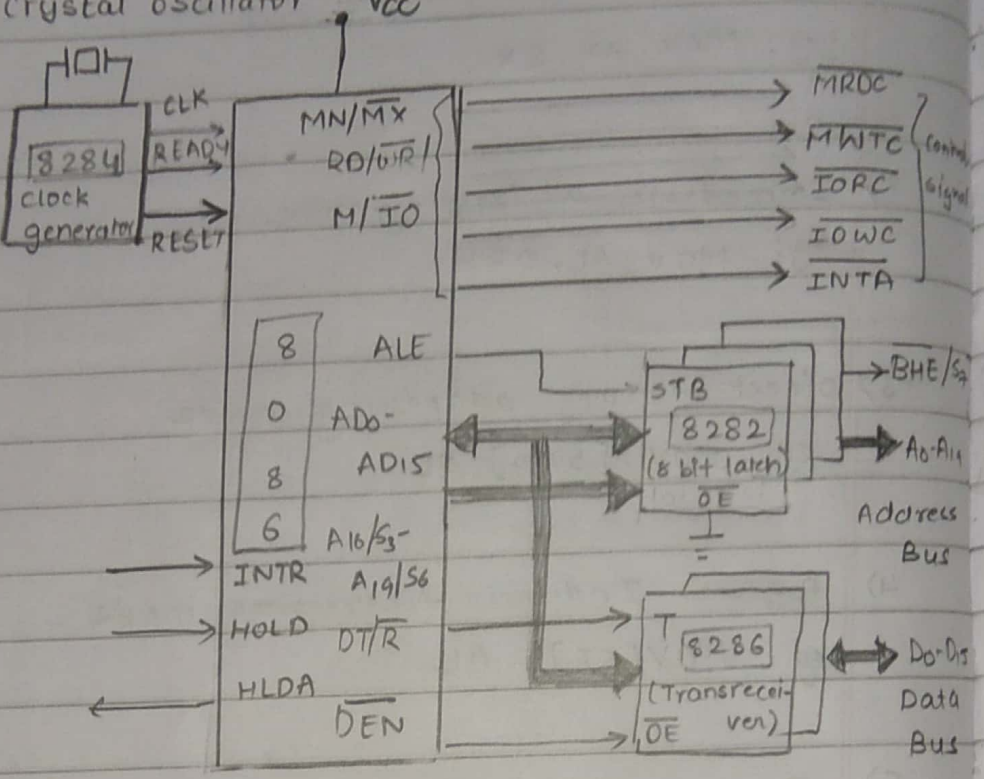
4) Register Indirect Addressing mode

eg: MOV[SI], AL

5) Implied Addressing mode

eg: STC

8086 Minimum Mode.



- 8086 works in minimum mode when MN/$\overline{MX}$=1 (Vcc)
- In minimum mode 8086 is ~~on~~ the only processor in the system
- As shown in ckt diagram clock, Ready & Reset signals are ~~processor~~ provided by the 8284 clock generator
- Address from the address bus latch into the 8282 (8 bit latch). Three such latches are needed as address bus is 20 bit
- The ALE for latch is given by 8086 itself
- The ALE of 8086 connected to strobe of 8282
- The databus is driven through 8286, 8 bit transreciever is required as data bus is 16 bit.

- Transrecievers are enable through $\overline{DEN}$ signal while direction of data is controlled by the $DT/\overline{R}$ signal.
- Both $\overline{DEN}$ & $DT/\overline{R}$ are given by 8086 itself
- $\overline{DEN}$ is connected to $\overline{OE}$ & $DT/\overline{R}$ is connected to T of 8286.

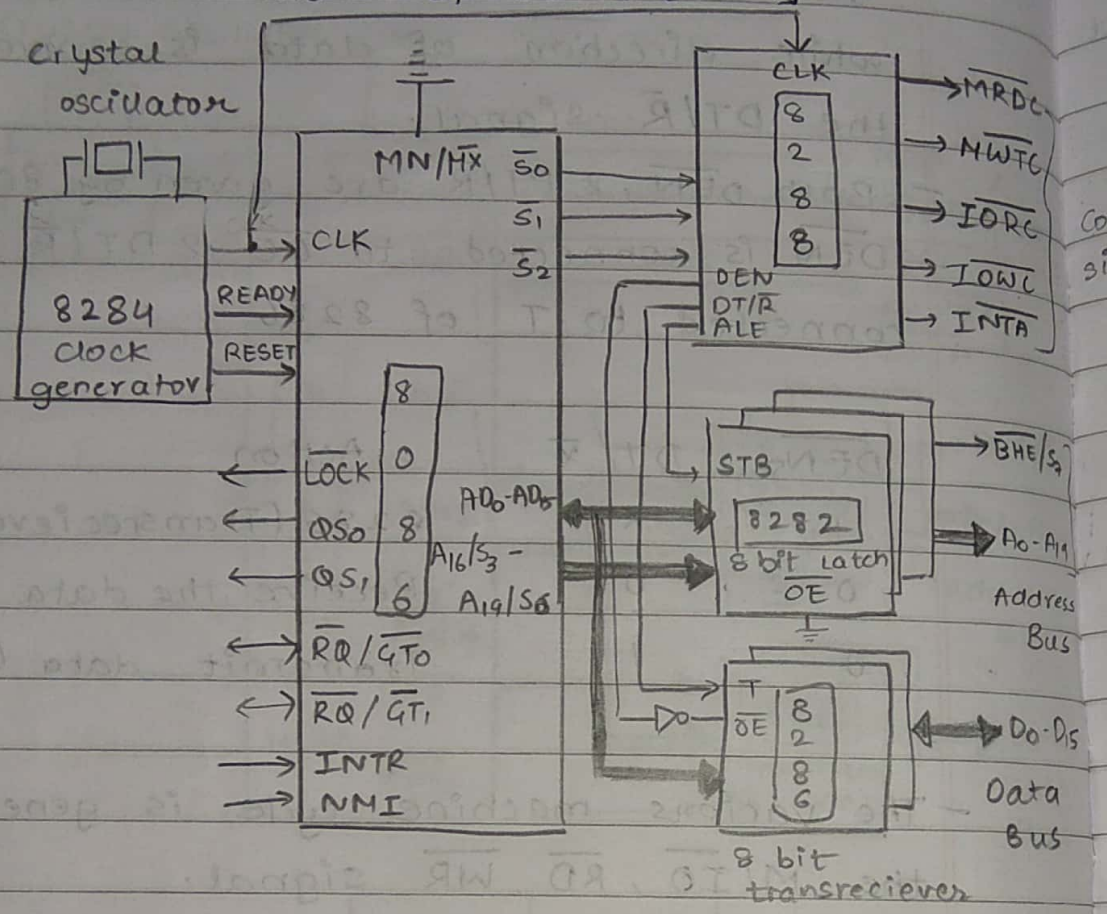
$\overline{DEN}$	$DT/\overline{R}$	Action
1	X	8286 (Transreciever) disable
0	0	Receive the data (Read)
0	1	Transmit data (write).

- The various machine cycle is generated by the $M/\overline{IO}$, $\overline{RD}$, $\overline{WR}$ signal.

$M/\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Action.	Signal
1	0	1	Memory reads	$\overline{MRDC}$
1	1	0	Memory write	$\overline{MWTC}$
0	0	1	I/O reads	$\overline{IORC}$
0	1	0	I/O Write.	$\overline{IOWC}$

- $M/\overline{IO}$, $\overline{RD}$, $\overline{WR}$ are given by 8086 itself
- DMA is done through HOLD & HLDA signal
- $\overline{INTA}$ is given by 8086 in response to INTR by 8086.

Maximum mode of 8086 [10 m]



- 8086 works in maximum mode when MN/$\overline{MX}$ = 0 (Gnd)
- In max mode, there is atleast one more processor in the system besides 8086.
- As shown in ckt diagram clock, Ready, & Reset signals are provided by 8284 (Clock generator)
- The most significant part of max. mode is 8288 (Bus controller)
- Instead of 8086 the bus controller (8288) provides various control signals
- Address from the address bus latches into 8282 (8 bit address latch)

- Three such ~~buses~~ latches needed as address bus is 20 bit.
- The ALE for 8 bit Latch is given by 8288 Bus controller
- The ALE is connected to STB of 8282
- The data bus is driven through 8286 (Data Transceiver)
- Two such transceivers are needed, as the data bus is 16 bit.
- The transceivers are Enabled through the DEN signal. while direction of data is controlled by DT/$\overline{R}$
- Both DEN & DT/$\overline{R}$ are given by 8288 Bus controller
- DEN is connected to $\overline{OE}$ & DT/$\overline{R}$ is connected to T of 8286

DEN	DT/$\overline{R}$	Action
0	X	8286 disable
1	0	Recieve data
1	1	Transmit data

- The various machine cycles are initiated by 8288 bus controller using $\overline{S_0}$, $\overline{S_1}$ & $\overline{S_2}$

	$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Status	8288 Active output
	0	0	0	Interrupt ACK	$\overline{INTA}$
I/O	0	0 (Read)	1	Read I/O port	$\overline{IORC}$
I/O	0	1 (write)	0	Write I/O port	$\overline{IOWC}$
	0	1	1	Halt	None
	1	0	0	Instruction fetch	$\overline{MRDC}$
	1	0 (Read)	1	Mem. Read	$\overline{MRDC}$
	1	1 (write)	0	Mem. Write	$\overline{MWTC}$
	1	1	1	Inactive	None

- DMA is done through $\overline{RQ}$ / & $\overline{GT}$ lines of 8086.
- $\overline{INTA}$ is given by 8086 , in response to INTR.

Instruction set of 8086.

* Arithmatic Instruction

1> ADD / ADC destination, source

eg I] ADD AL, 25H ; AL ← AL + 25H

II] ADD BL, CL ; BL ← BL + CL

III] ADC CX, DX ; CX = CX + DX + carry

BE	AE
CX = 1000H	CX = 3001H
DX = 100DH	DX = 2000H
carry = 1	carry = 0

2) SUB/SBB destination, source.
It's similar to ADD/ADC except that it does subtraction.
(SBB ⇒ Subtract with Borrow)

3) INC dest;
Add 1 to specified dest
I] INC SI

BE	AE
SI = 2002	SI = 2003

II] INC BL

4) DEC dest
It is similar to INC except it does decrement by 1.

5] MUL source (8 bit / 16 bit) - unsigned multiplication
- If source is 8 bit, it is multiplied with AL & ~~except~~ result is stored in AX.
- MUL affects AF, PF, SF, ZF

eg. I) MUL BL; AX = AL * BL.
II) MUL BX; DX - AX = AX * ~~BE~~ BX;
operation.

BE	AE
AX = 2222H	DX = 0000
BX = 4444H	AX = 3333
DX = 7777H	

AX AX
AX * BX = 0000 3333

6) IMUL source (8 bit signed / 16-bit multiplication)
same as MUL except, Here it can do signed operation.

7) DIV source (unsigned division)

eg: I) DIV BL;

operation:

AX divide by BL
AL ← Quotient
AH ← Remainder

```
        4444
3333 ) 1111 2222
       ----
        5555
```

II) DIV BX;
{DX : AX} / BX
AX ← Quotient
DX ← Remainder.

BE	AE
DX = 1111 H	AX = 4444 H
AX = 2222 H	DX = 5555 H
CX = 3333 H	

8) IDIV source (signed Division)

9) NEG destination

- This instruction forms the 2's complement of destination & source back to dest^n.

eg. NEG AL

BE	AE
AL = 35 H	AL = CB H
AL = 35 H	0011 0101
	↓ 1's
	1100 1010
	+ 1
	1100 1011

10) CMP dest, source

→ This instruction compare source with destination

→ The source & dest. must be of same size.

→ the result of this subtraction is NOT stored anywhere instead of flag bits are affected.

→ All condition flags are updated

eg: CMP CL, BL.

11) CBW [Convert signed Byte to signed word]

- This instruction copies sign of byte into all the bits of AH

eg. CBW

BE AE

AX = xxxx xxxx 1001 0001 AX = 1111 1111 1001 0001

signed bit.

12) CWD (convert word to double word)

BE AE

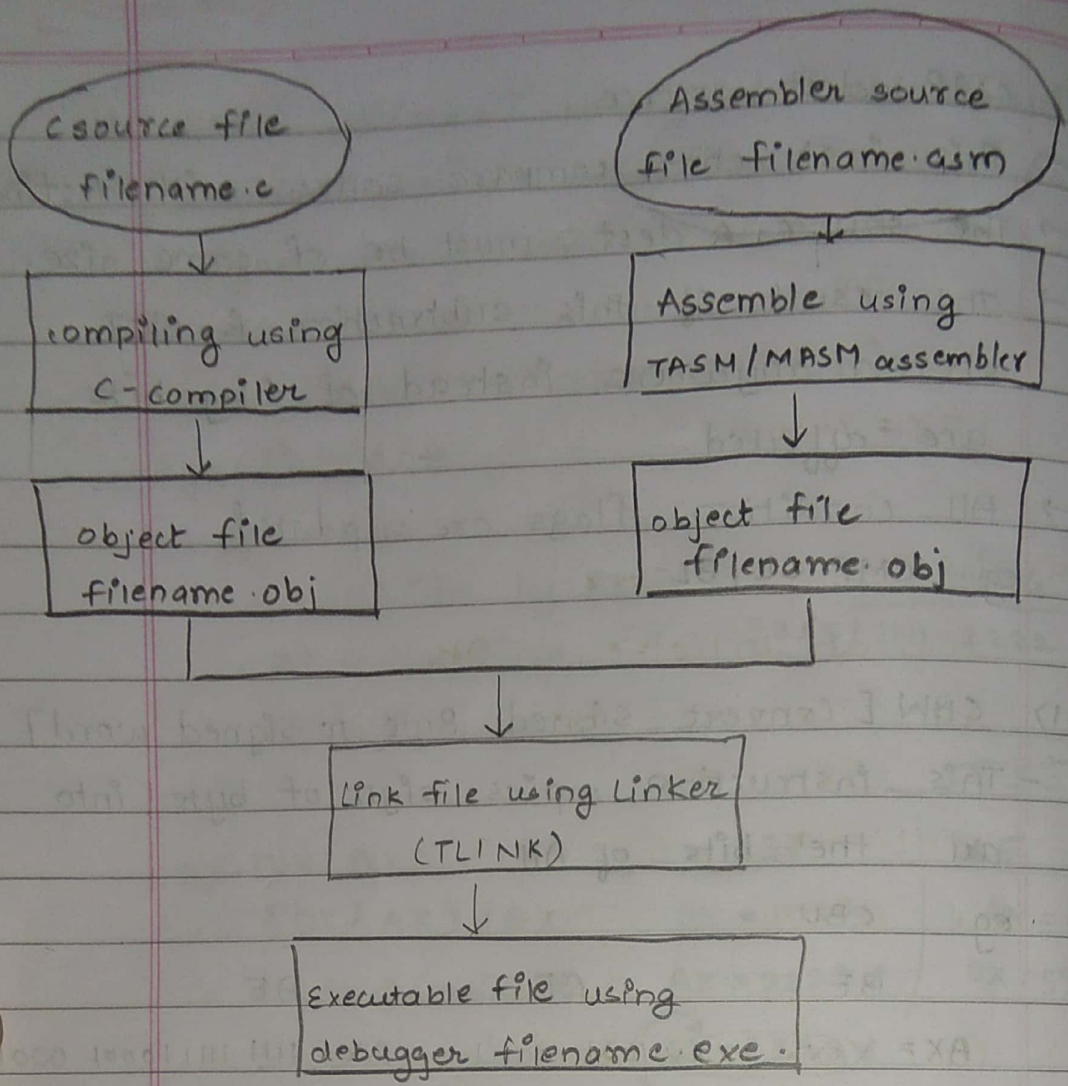
AX = 10011000 1111 0011 AX = 1001 1000 1111 0011

DX = xxxx xxxx xxxx xxxx DX = 1111 1111 1111 1111

SN on mixed language programming

→ Here we need to call written in other language as mixed language programming

→ Involve a call to functions (procedure) & macro.



Logical Group of Instruction.

1) NOT destination - operation 1's complement

eg. NOT AX

BE	AE
AX = 1234 H	AX = EDCB H
0001 0010	0011 0100
1110 1101	1100 1011
E D	C B

2) AND dest, source

Logical ~~bis~~ bitwise AND

eg AND AX, BX

BE	AE
AX = 1234H	AX = 1024H
BX = 7126 H	BX = 7126 H

3) OR dest, source
Logical bitwise OR
eg OR AX, BX
BE AE
AX = 1234H AX = 7336H
BX = 7126H BX = 7126H

4) EXOR dest, source
logical bitwise EXOR
eg EXOR AX, BX
BE AE
AX = 1234H AX = 6312H
BX = 7126H BX = 7126H.

5) TEST dest, source
- Logical Bitwise ANDing
- But it will not change content of destination reg & only affect condition flag like AND instruction.

shift and Rotate Instruction.

1) SHL/SAL (shift logical/Arithmatic Left)
eg SHL AX, 01H SHL dest, count
BE AE
AX = C123H AX = 8246H
1100 0001 0010 0011

1 1000 0010 0100 0110 ←
8 2 4 6

2) SHR (Shift Right) SHR dest, count

eg SHR AX, 01

BE AE

AX = C123H AX = 6091H

1100 0001 0010 0011

0110 0000 1001 0001

3) SAR (Shift Arithmatic Right)

SAR dest, count

BE AE

AX = C123H AX = E091H

1100 0001 0010 0011

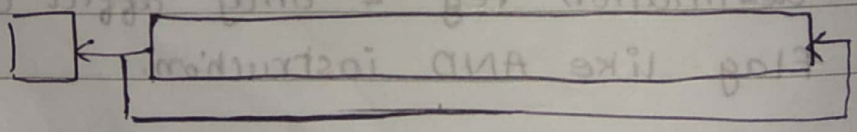
1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 → 1

E 0 9 1

ROtate

1) ROL (Rotate left)

ROL dest, count



BE AE

AX = C123H AX = 8247H

0 1100 0001 0010 0011

1 1000 0010 0100 0111

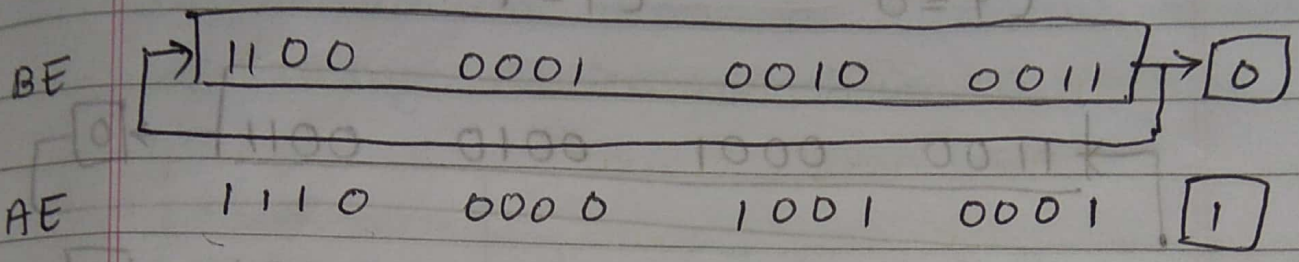
8 2 4 7

2) ROR (Rotate right

ROR dest , count

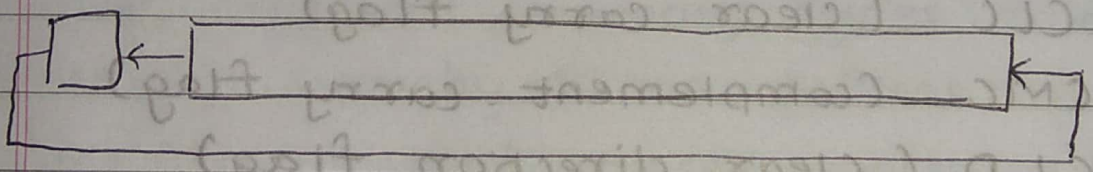
BE AE

AX = C123 H AX = E091 H



RCL (Rotate left with carry)

RCL dest , count



BE AE

AX = C123 H AX = 8246

CY = 0 CY = 1

0 1100 0001 0010 0011

1 1000 0010 0100 0110

8 2 4 6

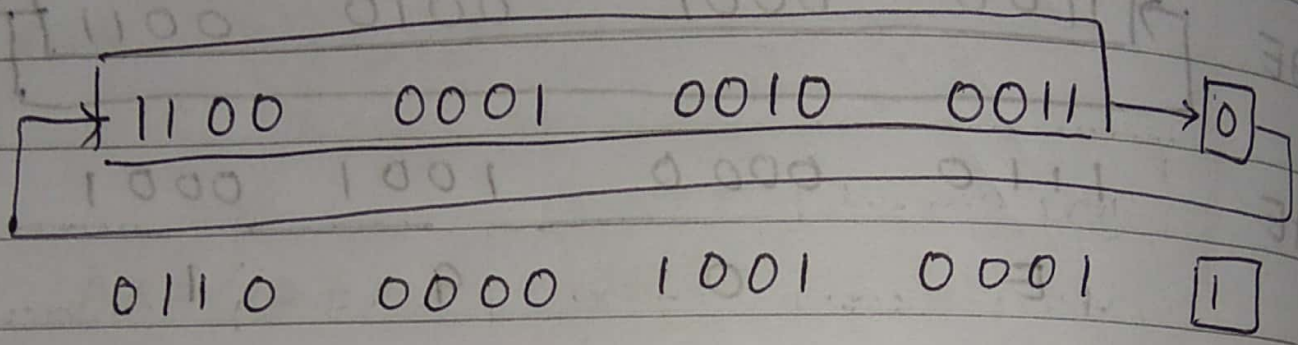
RCR (ROtate right with carry)

RCR dest, count

BE AE

AX = C123H AX = 6091H

CY = 0 CY = 1



Processor control Instruction.

STC (Set carry flag)

CLC (clear carry flag)

CMC (complement carry flag)

CLD (Clear direction flag)

STD (set Interrupt flag)

STI (set Interrupt flag)

CLI (Clear Interrupt enable)

NOP (No operation)

Memory Organization.

Q7 Explain memory hierarchy in detail.

There are different memory system, which have different storage capacity & different level of performance. They are organized to form a multilevel hierarchy of storage device as shown in figure.

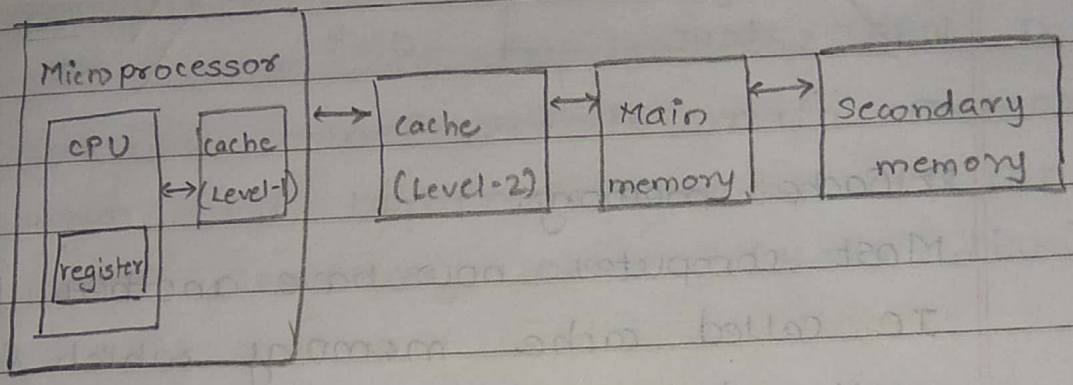


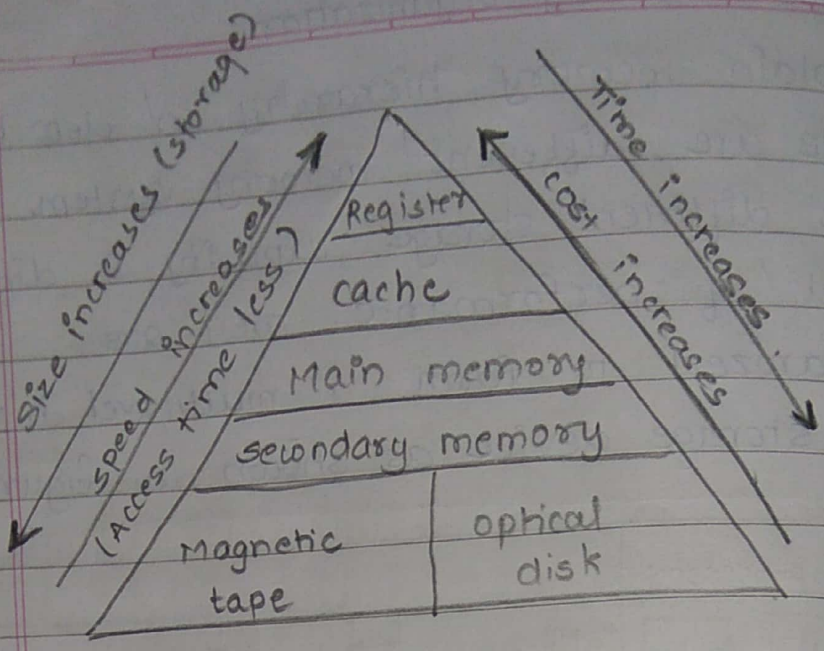
Fig: Memory Hierarchy

The different memory types involved in this hierarchy is as follows

1) CPU register (Internal processor memory)

There are small sets of High Speed register, flip-flop, which have operating speeds of computer and such High Speed memory together with ALU & control unit integerated into single chip & termed as CPU. They usually form a general purpose register (GPR's) for storing data as it is processed.

They form scratch pad memory eg. in 8086 its AX, BX, CX, DX etc.



2) Cache memory.

Most computer now have another level of IC called cache memory which is positioned logically between CPU & main memory. A cache memory storage capacity is lesser than main memory ~~A cache memory~~ but faster than main memory. Some cache memory resides on same IC called L1 cache and some residue other than CPU called L2 & L3 cache.

3) Main memory (Primary memory)

CPU directly access their location in the main memory. These are used for program and data storage during operation. It is read/write volatile memory. Compare to CPU register access speed is slower (but it is

larger than cache memory and fact that it is seperated from CPU.

4) secondary memory
This is much larger in size but slower than main memory and used for permanent storing prog & large data files, which are required by CPU. This data is transferred to main memory when CPU required. Because CPU cannot access secondary memory (Hard disk) directly.

Q. Explain different memory characteristics.

Ans. The important characteristics of any memory can be based some parameters. as follows.

i) Location of memory-

① CPU - The memory that have storage, placed inside the CPU is registers & High speed cache (Level-1) memory. It is read-write volatile memory.

② Primary memory - (Internal memory) The main memory is called as primary memory. It is next to cache memory & It is called as read-write volatile memory.

(3) Secondary (External) memory.
External memory consists of pe peripheral storage devices like HDD. It is next to main memory & It is non-volatile memory.

2) Capacity (size):
The capacity or size of memory is typically expressed in terms of number of bytes. External memory is typically huge & it is expressed in ~~terms of~~ terms of megabyte & gigabytes & Nowadays in Tera bytes.

3) Unit of Transfer
For internal memory the unit of data transfer is equal to number of data lines into & out of data module (i.e D_0 to D_n) The units of data transfer may be of 1-bytes, 2-bytes, 4-bytes etc at a time (or in one cycle) depends on data lines of CPU and I/O device.

4) Access method.
There are different types of access method used to access a particular memory location from memory. These are as follows.

— Random Access
If storage location access in any order and access time is independent of the physical location being accessed i.e to access any memory location of memory it takes same amount of time eg. RAM

— Serial Access -
Memories whose storage location can be accessed in certain pre-determined sequence is called serial access memory. In serial access memory, the location are accessed by continually moving storage location. Here time required to access any location depends on position of memory location.

— Semi-random access.
Storage devices like Hardware disk, floppy disk & CD ROM contains many rotation circular storage tracks. Here track can be access randomly, but access within each track is serial. In such case access mode is semi random.

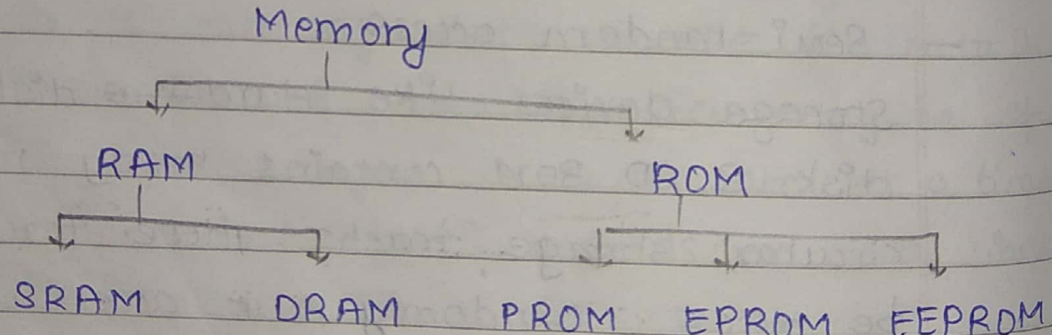
5) Performance measure.
which measures speed of memory & parameters are as follows.

- Access time -
This is the time from the instant that address is presented to memory to the instant that data ~~have~~ is available for use.

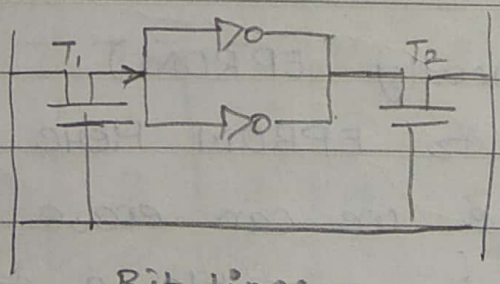
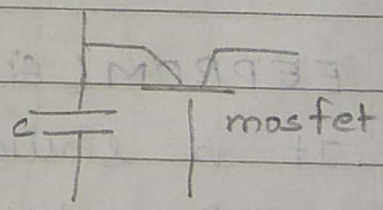
- Memory cycle time
This consist of access time plus any additional time required before second access can commence.

- Transfer rate
This is the rate at which data can transfered into or out of a memory.

6) Physical type - (Physical classification)



- RAM (Random Access memory)
It is read-write access memory. It is ~~available~~ volatile memory. It is two type ie SRAM & DRAM.

	SRAM (Static Ram)	DRAM (Dynamic RAM).
1.	It is faster than DRAM.	It is slower than SRAM.
2.	It is built-up of flip flop & transistors	It is built up for capacitors & mosfet.
3.	Refreshing ckt is not required.	~~Resh~~ Refreshing ckt is not required.
4.	Cost is more	Cost is less.
5.	Used for implementing cache memory.	Used for implementing main memory.
6.	It consist of more no. of components.	It consists of less no of components.
7.		

ROM (Read only memory)
It is used for permanent storage It is a non-volatile memory. It can be PROM, EPROM, EEPROM

PROM (Programmable ROM)
The contents of ROM cannot be altered once it is write. But it is less expensive. It is non volatile ~~exa~~
eg. toys, which contain ROM.

EPROM (Erasable PROM)
Here write operation is also possible before write operation all storage cell must be erased by using ultra-violate radiation light ~~the~~ Here selective ~~ee~~ erasing is not possible It is less expensive than EEPROM.

EEPROM (Electrically EPROM)
It is similar to EPROM Here by passing voltage we can erase data Here selective is possible & cost is more than EPROM.

7) Physical characteristics.

volatile / non-volatile erasable / Non-erasable.

Page Replacement techniques: (Cache memory)

① FIFO (First in First Out)

Time	1	2	3	4	5	6	7	8	9	10	11	12
page	2	3	2	1	5	2	4	5	3	2	5	2
frame	2	2	2*	2	5	5	5	5*	3	3	3	3
		3	3	3	3	2	2	2	2	2*	5	5
				1	1	1	4	4	4	4	4	2
	miss	miss	Hit	miss	miss	miss	miss	Hit	miss	Hit	miss	miss

Hit ratio = $\frac{\text{Hit}}{\text{Hit+Miss}} \times 100 = \frac{3}{12} \times 100 = 25\%$

page fault = 9

② LRV Algorithm (least recently used)

Time	1	2	3	4	5	6	7	8	9	10	11	12
page	2	3	2	1	5	2	4	5	3	2	5	2
frame	2	2	2*	2	2	2*	2	2	3	3	3	3
		3	3	3	5	5	5	5*	5	5	5*	5
				1	1	1	4	4	4	2	2	2*
	Miss	Miss	Hit	Miss	Miss	Hit	M	Hit	Miss	Miss	Hit	Hit

Hit ratio = $\frac{5}{12} \times 100 = 0.43 = 43\%$

3) optineal

Time	1	2	3	4	5	6	7	8	9	10	11	12
Page	2	3	2	1	5	2	4	5	3	2	5	
	2	2	2*	2	2	2*	4	4	4	4	4	
		3	3	3	3	3	3	3	3*	2	2	
				1	5	5	5	5*	5	5	5*	
	M	M	H	M	M	H	M	H	H	M	H	H

Hit ratio = $\frac{6}{12} \times 100 = 50\%$.

Eg 1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	2
	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	
	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0*	0	7	1	
		0	0	0	0*	3	3	3	2	2	2	2	2*	1	1	1	1*	1	0	
			1	1	1	1	0	0	0	3	3	3*	3	3	2	2	2	2	2	
	M	M	M	M	H	M	M	M	M	M	M	H	H	M	M.	H	H	M	M	

Hit = $5/20 = 25\%$ Page fault = 15

	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	
		0	0	0	0	0	0*	0	0	3	3	3*	3	3	3	0	0	0	0*	
			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	
	M	M	M	M	H	M	H	M	M	M	M	H	H	H	H	M	H	M	H	

Hit = $8/20 = 40\%$. Page fault = 12

	7	7	7	2	2	2	2	2	2*	2	2	2	2*	2	2*	2	2	7	7	
		0	0	0	0*	0	0*	4	4	4	0	0	0	0	0	0*	0	0	0	
			1	1	1	3	3	3	3	3*	3	3*	3	1	1	1*	1	1		
				H	M	H	M	H	H	M	H	H	M	H	H	H	M	H		

Hit = $\frac{11}{20} = 55\%$. Page fault = 9.

Module 6. I/O organization.

- I/O Module — 5m
- I/O processor (8089) — 10m
- Data transfer technique — 10m
  - Programmed I/O
  - Interrupt driven
  - DMA

Explain different I/O data transfer technique.

There are three I/O data transfer techniques

① Programmed I/O

- It is a useful method for computer where hardware cost needs to be minimized.
- Entire I/O is handled by CPU with the help of small software without any additional Hardware.
- It is based on the concept of busy waiting before I/O is performed.
- CPU constantly check the status of I/O devices.
- If I/O device is not ready, CPU simply waits in the loop for the I/O device to become ready.
- CPU performs the following steps.

① Read the I/O device status bit

② Test the status bit to determine the device is ready to begin data transfer operation.

3 If device is not ready, return to step ① otherwise proceed with data transfer

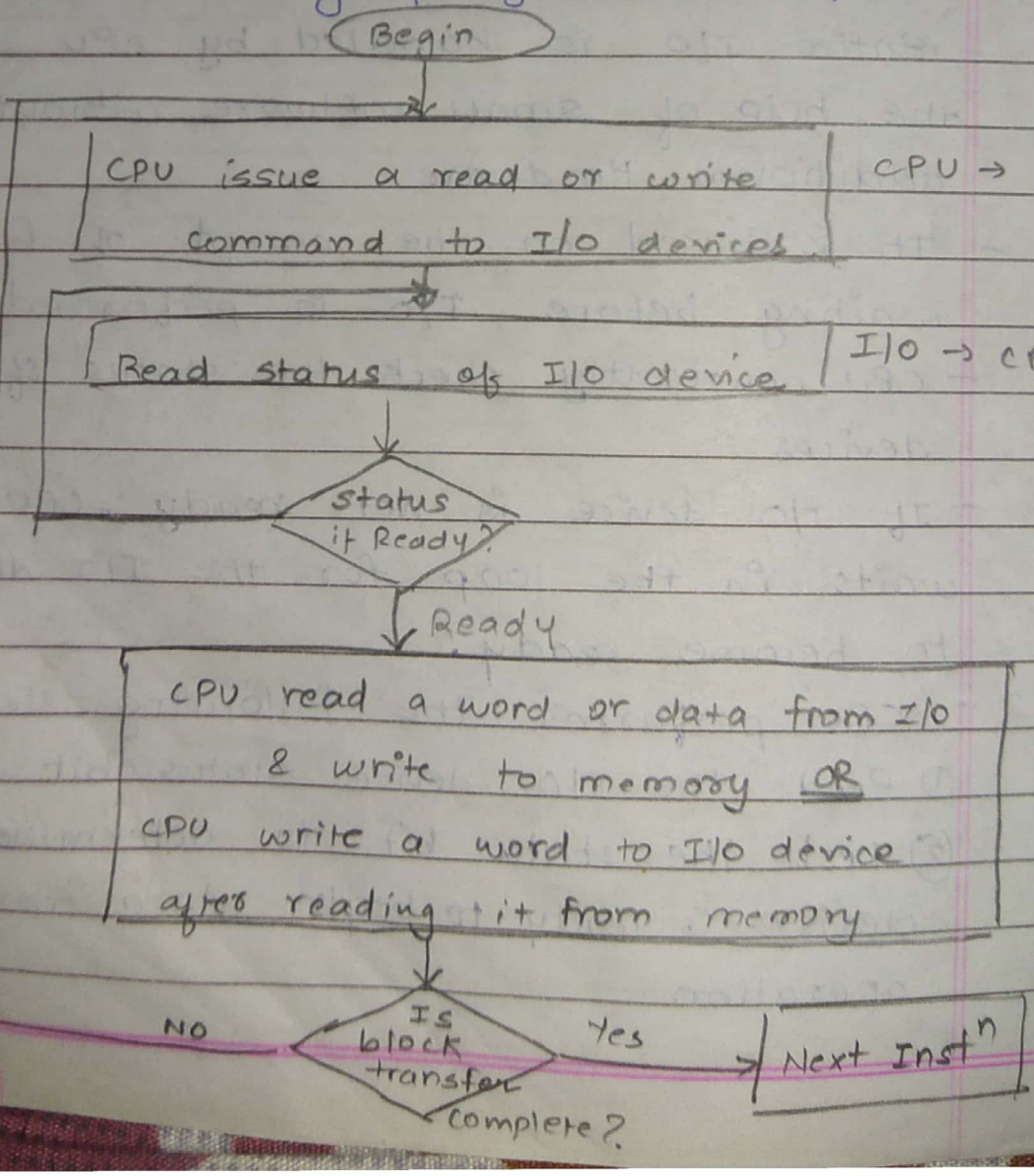
Advantages :-

1. It is very simple technique
2. It is cost effective technique.

Disadvantages:

1. Max time CPU has to wait for I/O device hence ~~very~~ very time Consuming technique.

Flowchart: ~~technique~~ transfer a block of data using programmed I/O.



PAGE NO :
DATE :

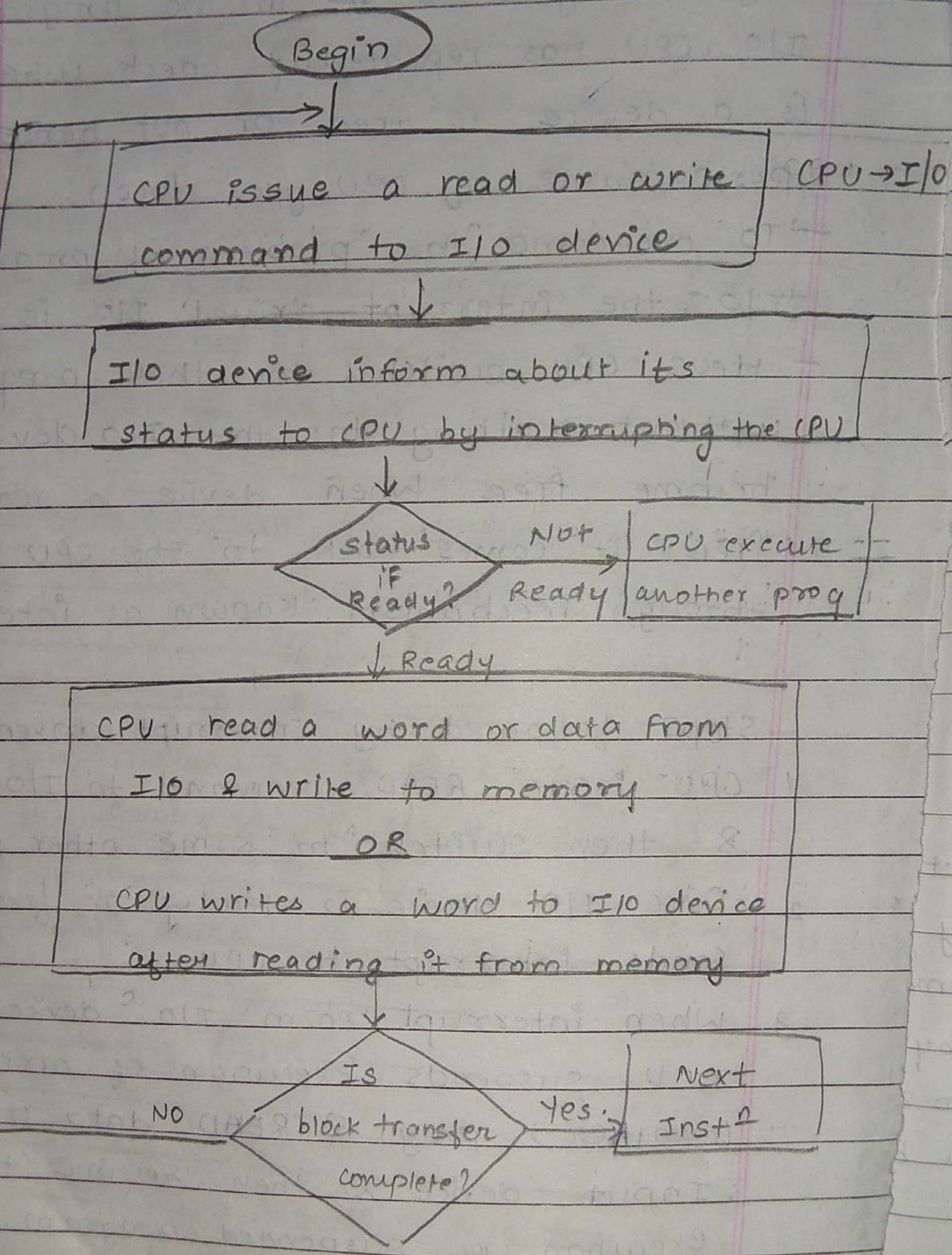
Interrupt driven I/O

- Major drawback of programmed I/O is busy waiting since ~~the~~ in the programmed I/O, CPU has repeatedly check whether is a device is free or not hence performance of CPU goes down.
- To overcome the problem of programmed I/O, the interrupt drived I/O is used.
- Here CPU switches to other prog without waiting for the I/O devices to become free. When device is ready or free, it comes back to the CPU through mechanism known as interrupt.

Steps followed by Interrupt driven I/O

1. CPU issue READ command to I/O device & then switch to some other program
2. Once I/O device is ready or free it sends interrupt to CPU.
3. When interrupt from I/O device comes CPU suspends execution of current (another) prog & READ data from Input device & then resume the execution of suspended program.

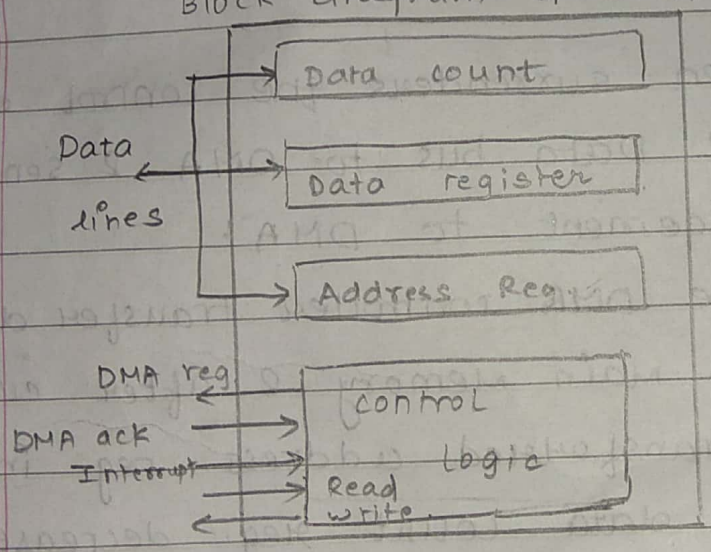
flowchart : transfer a blocks of data using interrupt driven I/O



SN of DMA [Direct Memory Access]

- There are some drawback of programmed I/O & Interrupt driven I/O. Both required participation of the CPU to transfer data between memory & devices. Hence the CPU become busy in performing in I/o transfer.
- To transfer large block of data at high speed without or less CPU intervention, an alternative method can be used. A special controller is provided to allow transfer of block of data directly between I/O devices & Main memory.

Block diagram of DMA.



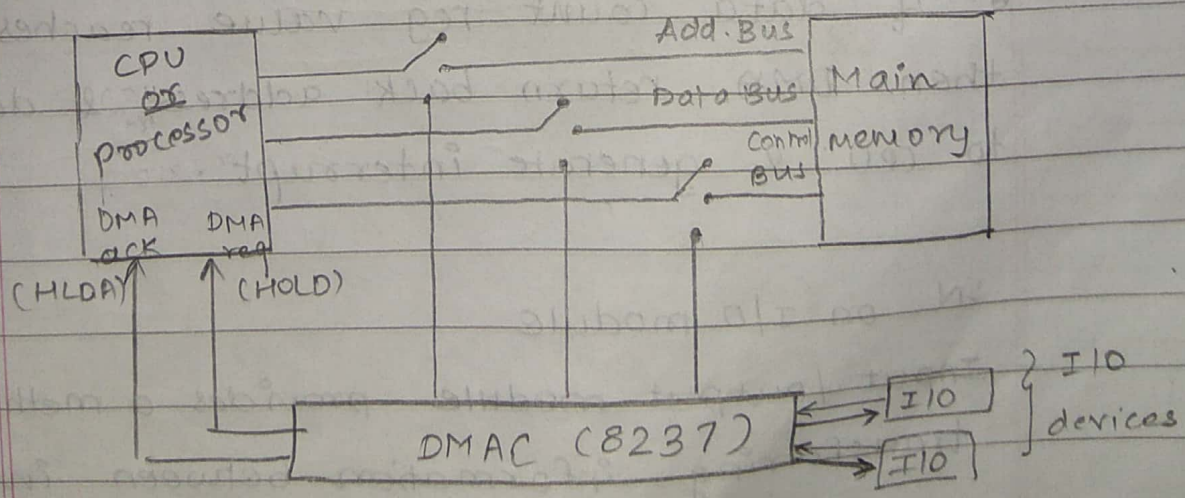


Fig: working of DMA

DMA working:

DMA data transfer proceeds as follow:

1. The CPU initially load the DMA register i.e Address register & data count with their initial values. where Data count ~~address reg~~. contains number of bytes to be transfered to or from memory & Address reg. contain starting memory location of data transfer
2. The I/O device need the service, generate request on any one channel on DMA.
3. Then DMA request for that I/O device to CPU.
4. CPU then surrenders the control of address & Data bus to DMA & send acknowledgement to DMA.
5. Now the DMA controller transfer directly to & from main memory & after every byte of transferred address reg. increment by one & data count reg. decrease by one.
6. If data count reg. value reaches to zero then DMA return back address & data bus to CPU & generate interrupt.

SN on I/O module

- Input/output module provides a method for transferring information between internal storage & external I/O devices.

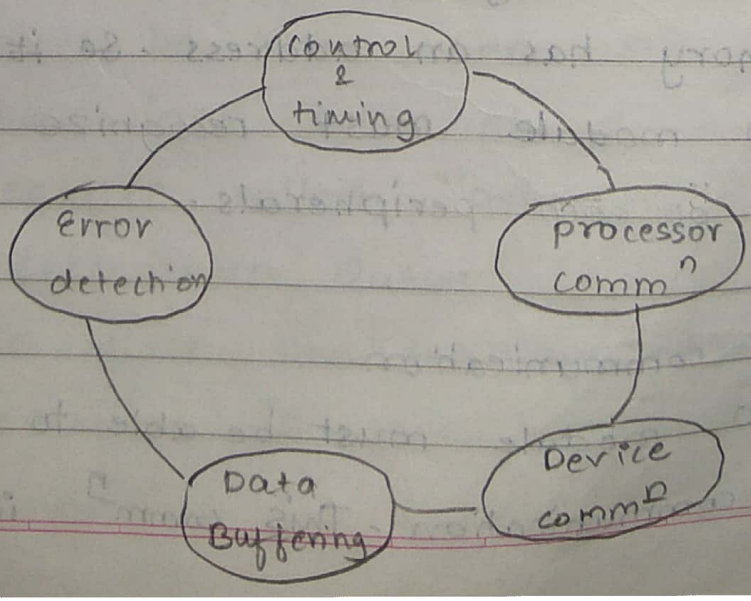
- Peripheral connected to computer need special communication link is to resolve the difference that exist between the computer & each peripherals.
- The major differences are -

1. peripherals are electromechanical & electromagnetic devices & their manner of operation is different from the operation of the CPU & memory, which are ~~electric~~ electronic devices. Therefore, a conversion of signal may be req^{d}.
2. The data transfer rate of ~~pherip~~ peripherals is slower than CPU, hence synchronization mechanism req^{d}.
3. Data codes & formats in peripherals are differ from the word format in CPU & memory

Hence I/O module:

- Interface to CPU & memory.
- Interface one or more peripherals.

* various functions of I/O modules.



1. Control & timing :

- The co-ordinate the flow of traffic between resources & external devices.
- To transfer data from external ~~but~~ devices to processor involve the following sequence of steps :
  a. The processor check status of attached device (I/O device)
  b. The I/O module return the device status.
  c. If device is ready to transmit, the processor transfer the data by means of I/O module

2. CPU communication :

- CPU comm^n involve following.
  a. Data - Data are exchanged between the processor & I/O module over data bus.
  b. Status reporting - Because peripherals are so slow, it is important to know the status & I/O module.
  c. Address recognition - Just as each word of memory has an address. ~~So it is~~ Thus an I/O module must recognize unique address of each peripherals.

3. Device communication.

- The I/O module must be able to perform device communication - This comm^n involves

commands. status info & data.

4. Data Buffering.

- Rate of data transfer to/from CPU is normally higher than I/O device hence I/O module buffers data so that CPU & I/O device send/receive at its rate.

5. Error detection

Must detect & correct or report error

- Types of error during
- Data Error, transmission
- Mechanical malFunction

M3 - Q. Explain pipelining process concept & different stages of pipeline.

- Pipelining is an effective way of organizing parallel activity in computer system. The actual ~~feeling~~ fetching of inst^n from memory is a major bottleneck in instruction speed.
- computers have ability to fetch inst^n from memory in advanced so that they would be there when they are needed.
- These inst^n were stored in a set of register called "pre fetch" buffer or "Instruction Queue"
- In effect pre-fetching dividing inst^n execution in many parts & each one is handled by seperate unit, all of which

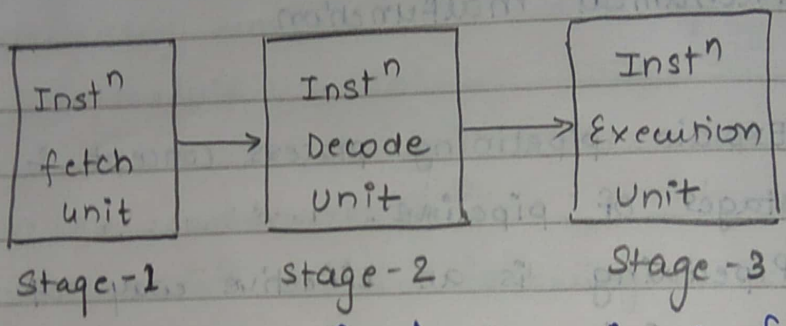
run parallel & each part is called stage
- Pipeline stages.

M stage instructions pipeline can overlap the processing of upto M - inst^n. So it is desirable to use more than 2 stages to maximize inst^n throughput.

Examples:

1. Three stage pipeline

In 3 stage pipeline, one inst^n has three units.



Stage-1 : Inst^n fetch unit, it fetch inst^n from memory & placed it in buffer or queue unit it needed.

Stage 2 : Inst^n Decode until, it decode the inst^n & determine its type and what operand (data) it need.

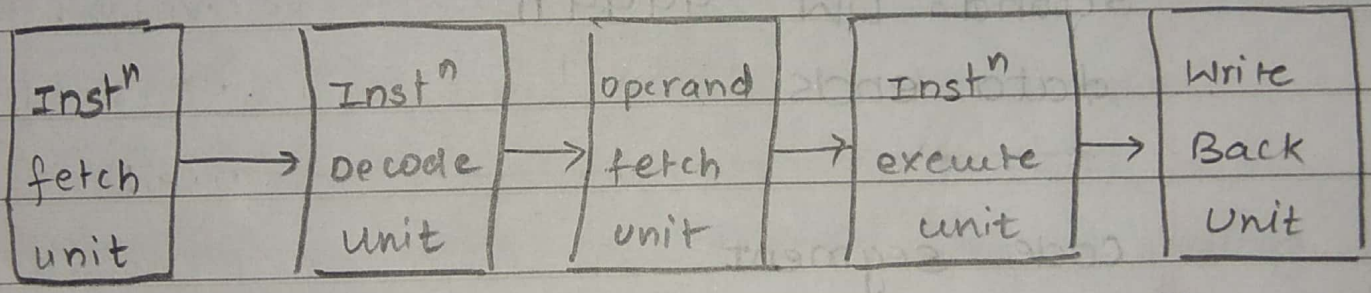
stage 3 : It actually execute the inst^n with the help of ALU.

Stage	Instruction					
1	I_1	I_2	I_3	I_4		
2		I_1	I_2	I_3	I_4	
3			I_1	I_2	I_3	I_4 → Time
	t_1	t_2	t_3	t_4	t_5	t_6

Pipelining - fetching the next instⁿ while decoding & executing current instⁿ is called as pipelining.

a. During time unit (t1), fetch - 1 from mem.
b. During time unit (t2), stage -2 decode I1 & stage -1 , fetch I_2 from memory.
c. During time unit (t3), stage 3 execute I1, stage 2 decodes I_2 and stage 1 fetch I3. and so on.

2. Five stage pipeline:



Stage 1

Stage 2

stage 3- Read operands (data) either from memory or register

Stage 4-

Stage 5- It write result back to proper register or address.

Program of 8086.

a. Addition of 8 bit numbers / 16 bit

```
code segment
assume CS: code
Start:
        MOV AL, 12h     / MOV AX, 1111h
        MOV BL, 64h     / MOV BX, 2222h
        ADD AL, BL      / ADD AX, BX.
End start
code ends.
```

b.

```
data segment
first   DW  6666h
second  DW  9999h
data ends

code segment
assume  CS: code , DS: data,
start:
    MOV AX, data
    MOV DS, AX
    MOV AX, first
    MOV BX, second
    ADD AX, BX
end start
code ends.
```

Q. Explain different pipeline Hazards

→ In general there are three major difficulties that cause the ~~information~~ instruction pipeline to deviate from its normal opp oprn

a. Resource conflict.

b. Data Hazards (Data dependency conflict)

c. Branch difficulties.

a. Resource conflict :- A resource conflict is a competition of the same resource at the same time. Example of resources includes memories, cache, buses, ports, functional unit like ALU.

- To understand concept of resource conflict let us take eg

eg: $x = y + z$

Above equation is executed by following instruction.

I1 : $R_1 \rightarrow Y$

I2 : $R_2 \rightarrow R_1 + Z$

I3 : $x \rightarrow R_2$

Time	1	2	3	4	5	6	7	8
I_1	FI	DI	OF	EX				$R_1 \leftarrow Y$
I_2		FI	DI	OF	EX			$R_2 \leftarrow R_1$
I_3			X	X	FI	DI	OF	EX $x \leftarrow R_2$

As we see,

- During clock cycle 3, I1 is fetching operand (OF) and no other inst^n can access memory

during cycle 3 & same with I2.
- Inst^n - 3 (I3) is delay by 2 cycle as it cannot fetch Inst^n as memory is being access by other inst^n.
- Thus resource dependency can detoriate overall performance of pipeline execution
- Above problem can be solved by using seperate inst^n & data memory.

b. Data Hazards (Data dependency conflict)
- It arises when an inst^n depends on result of previous inst^n but this result is not yet available.

for eg. I1 : D = B + C
I2 : X = D + 2

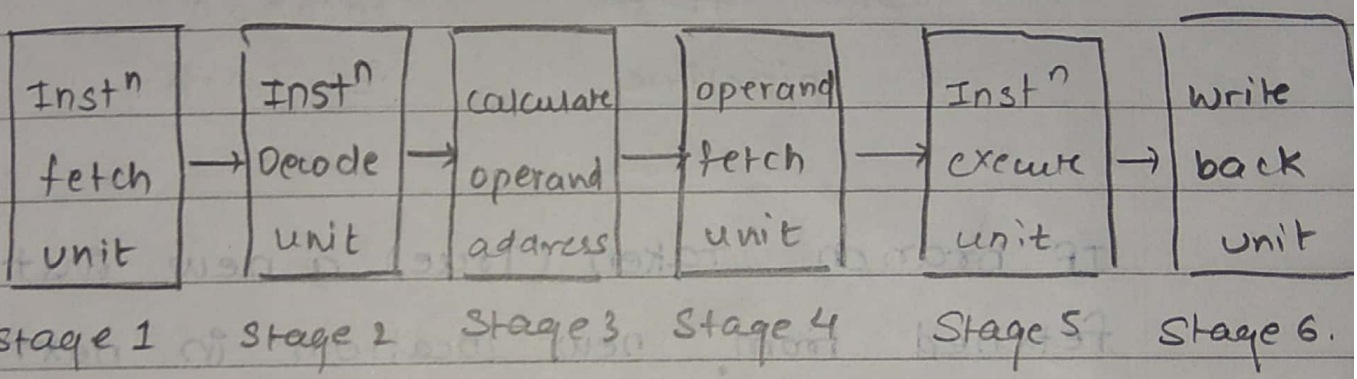
Here Inst^n 1 modifies the variable - D, Inst^n - 2 uses the value of variable - D, Thus, there is data dependency between two inst^n due to variable - D and both inst^n simultaneously in pipeline can not be executed.

c. Branch difficulties.
- In the absence of branch inst^n all the four stages operate on different inst^n.
- In case of Branch inst^n as it is decided after decode address in clock cycle the transfer from Inst^n fetch (IF) to decode inst^n (DI) of inst^n is halted until the execution of branch inst^n is completed.

BCD addition.

```
MOV AX, 9999H
MOV BX, 9999H
ADD AL, BL
DAA
ADC AH, BH
DAA
```

Six Stage application



Stage 1

Stage 2

Stage 3 - It will calculate effective address of operand (data)

Stage 4

Stage 5

Stage 6.

Time / Instn	1	2	3	4	5	6	7	8	9	10	11	12	13	14
I1	IF	ID	OF	EX										
I2		IF	ID	OF	EX									
I3			IF	ID	OF	EX								
I4				IF	.....		IF	ID	OF	EX				
I5								IF	ID	OF	EX			

Previous Instn is flushed from pipeline

New instn branch location is fetch.

If branch taken (True) a new instn is fetched from new location in next cycle
If branch not taken (False) the instn fetched previously can be used.

Q. Exp flynn's classification in detail.

→ -Various computer classification scheme have been formed according to architecture of parallel computer.

- In 1966, Michael J. flynn based his classification scheme on the multiplicity of Instn stream & data streams in computer.
- On this basis digital computers may be classified into four categories.
- The term stream is used to denote a sequence of items (Instn or data) as

executed or operated upon by single processor.

- Instruction stream is a sequence of Instⁿ as executed by machine.
- Data stream is a sequence of data including input partial or temporary results called for previous result of Instruction.
- Categories of flynn's are :

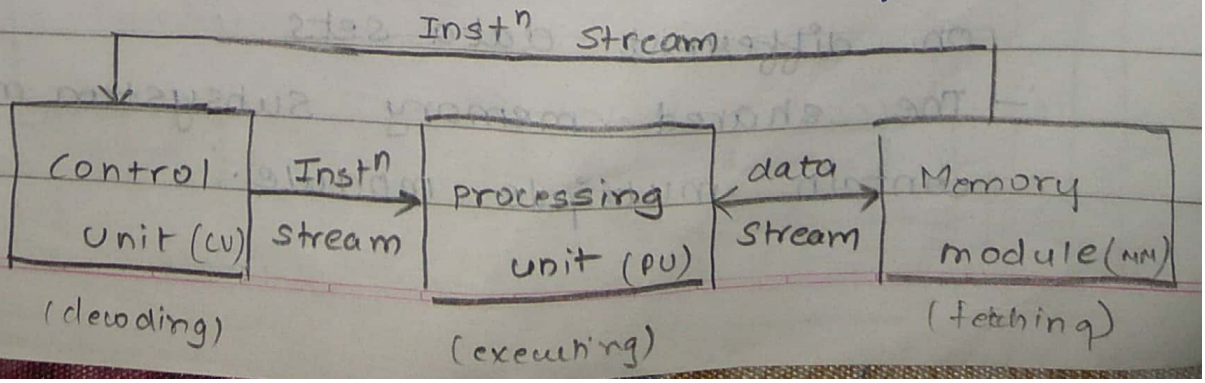
1. SISD (single Instruction stream - single data stream)
2. SIMD (single Instⁿ stream - Multiple data stream)
3. MISD (Multiple —"— . single —"—)
4. MIMD (Multiple —"— . Multiple —"—)

Conceptually only three types. System components are needed, control unit (CU), processing unit (PU) & memory module (MM)

- The Instⁿ & data are fetched memory module are decoded by control unit & further sends to processing unit for execution.

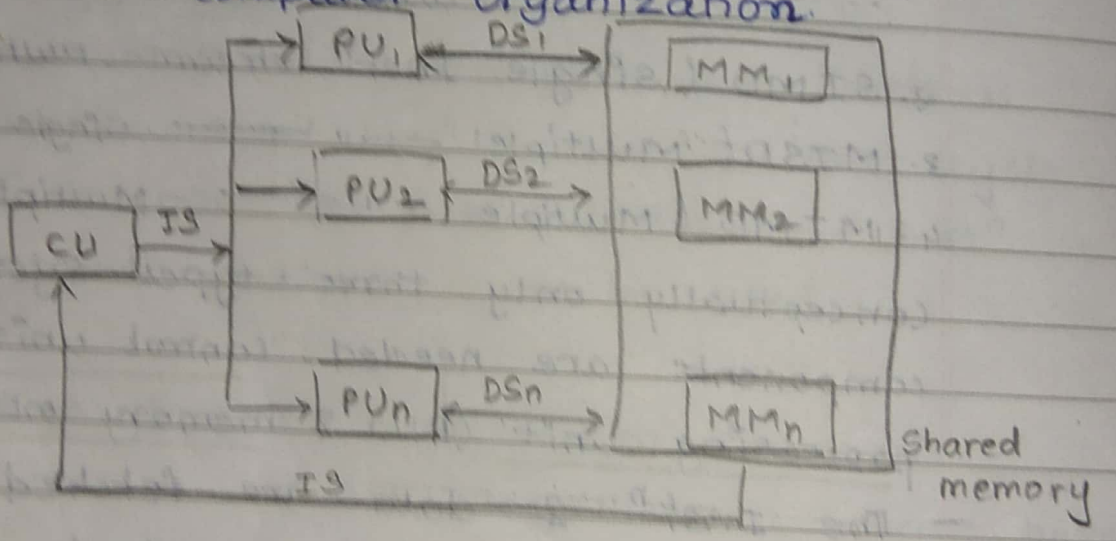
1. SISD computer organization.

Most serial computers are of SISD type.



- In SISD inst^n are executed seriaily
- They can be overlapped their execution Stage with Fetch stage like 8086
- Inst^n fetched from memory module that is decoded by control unit to generate decoded inst^n stream. The processing unit like ALU performs calculation on this. Inst^n stream & data stream fetch to & from memory.

2. SIMD computer Organization



- This category corresponds to array processor
- There are multiple processing unit supervised by single control unit.
- All processing unit receive same instruction broadcast from control unit but operanted on different data sets
- The shared memory subsystem may contain multiple modue.

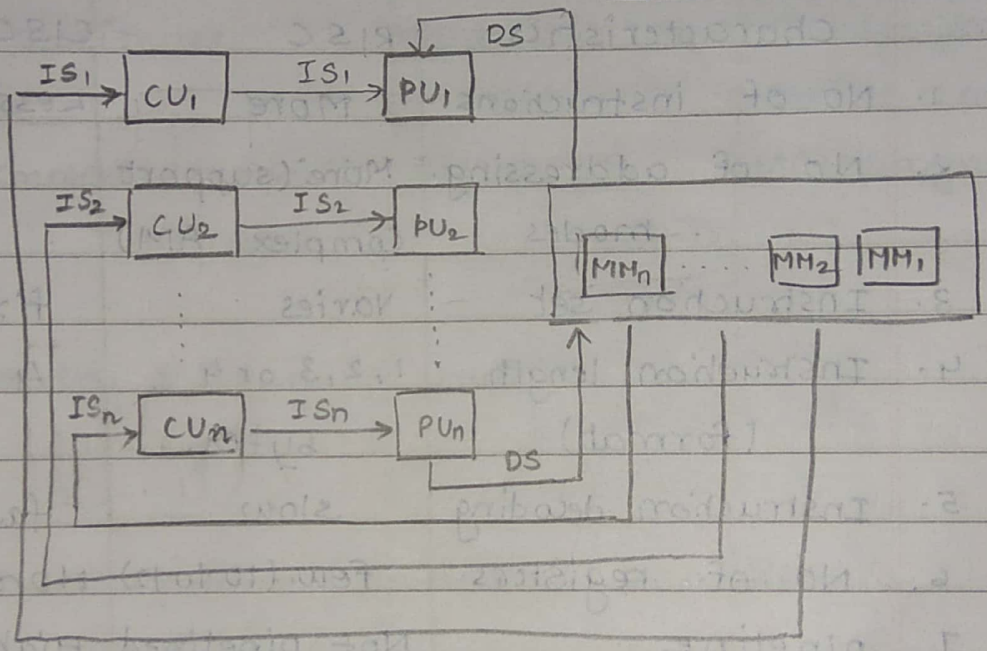
3. MISD Computer Orgⁿ.

— There are n-processing unit can receiving distinct Instⁿ stream over same data stream.

— The result of one processor (PU) passed to next processor in micropipes.

- The concept of this organization is not implemented.

— This is also known as 'systolic Array'



4. MIMD Computer Organization.

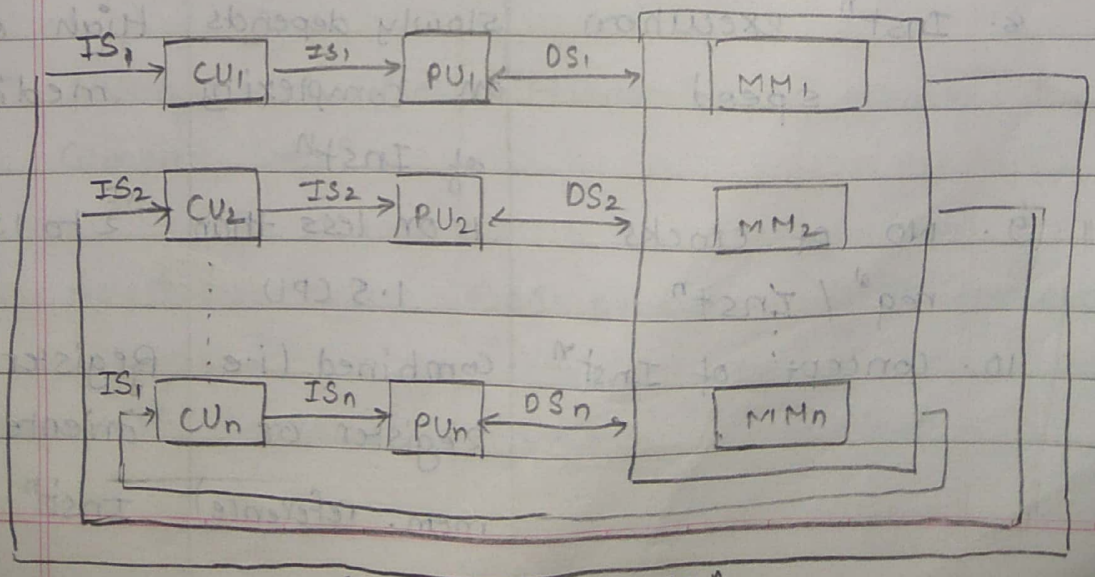


fig: MIMD computer orgⁿ

- Most multiprocessor system classified in this categories ,now-a-days
- It is also called as multiple SISD organization.

Module 3 control Unit

Differentiate RISC V/S CISC [5-10 m]

~~RISC~~ ~~CISC~~ .

RISC → Reduced Instruction set computer

CISC → complex Instruction set computer

Characteristics	RISC	CISC
1. No of instructions	More	Less
2. No of addressing modes	More (support Complex AM)	Less
3. Instruction set	varies	fixed.
4. Instruction length (format)	1, 2, 3 or 4 bytes	4 bytes
5. Instruction decoding	slow	fast
6. NO of registers	few (10 to 12)	More (50 to 10
7 pipeline	Not pipelined or less pipelined	Highly pipelined
8. Instⁿ execution speed	slowly depends on complexity of Instⁿ	High or medium
9. NO of clocks reqᵈ / Instⁿ	1 or less than 1.5 CPU	2 to 15 CPI
10. concept of Instⁿ	Combined (i.e Register or mem. reference)	Register oriented Instⁿ

11.	Memory Access	Frequently	Rarely
12.	compiler	Simpler	complicated
13.	Inst^n Execution (control Unit)	By microprogrammed	By Hardwired
14.	Example	ARM 8051 Intel i860	8085, 8086. Intel x86 family
15	cost	Less	More

Diff bet Hardwired & microprogrammed CPU

Attributes	Hardwired control Unit	Microprogrammed control Unit.
1. Implementation Approach	Sequential circuit	Microprogramming
2. Decoding & Sequencing Logic	complex	Easy.
3. Speed of Execution	fast	slow
4. Cost Of Implementation	More	Less
5. Ability to handle Complex Inst^n	Difficult	Easier
6. Application	RISC processor	CISC processor
7 Inst^n set size	Less	More.

8.	Control memory	Absent	present
9.	Control function	Implemented by Hardware	Implemented by S/w
10.	Flexibility	Not flexible, difficult to modify for new instⁿ	Flexible, New instⁿ can be easily added.
11.	Ability to support different as & diagnostic feature	very difficult (unless anticipated during design)	Easy.

Explain design of control unit with respect to Hardwired & microprogrammed (softwired) approach.

To execute instⁿ, a computer processor must generate a control signal in proper sequence This sequence of action can either by processor's software or by hardware There are 2 different approach used for control unit design

1. Hardwired control Unit
2. Microprogrammed control Unit.

1. Hardwired Control Unit

A hardwired control unit is a combination circuit which gets a set of input & transform

them into set of control signals
- Hardwired control unit provide Highest speed
- RISC's are implemented with Hardwired CU.
- If the instruction is complex, implementing Hardwired CU is difficult in the case microprogrammed CU are used.
- The CU are fixed logic circuit to intrepre inst^n & generate control signal for that.

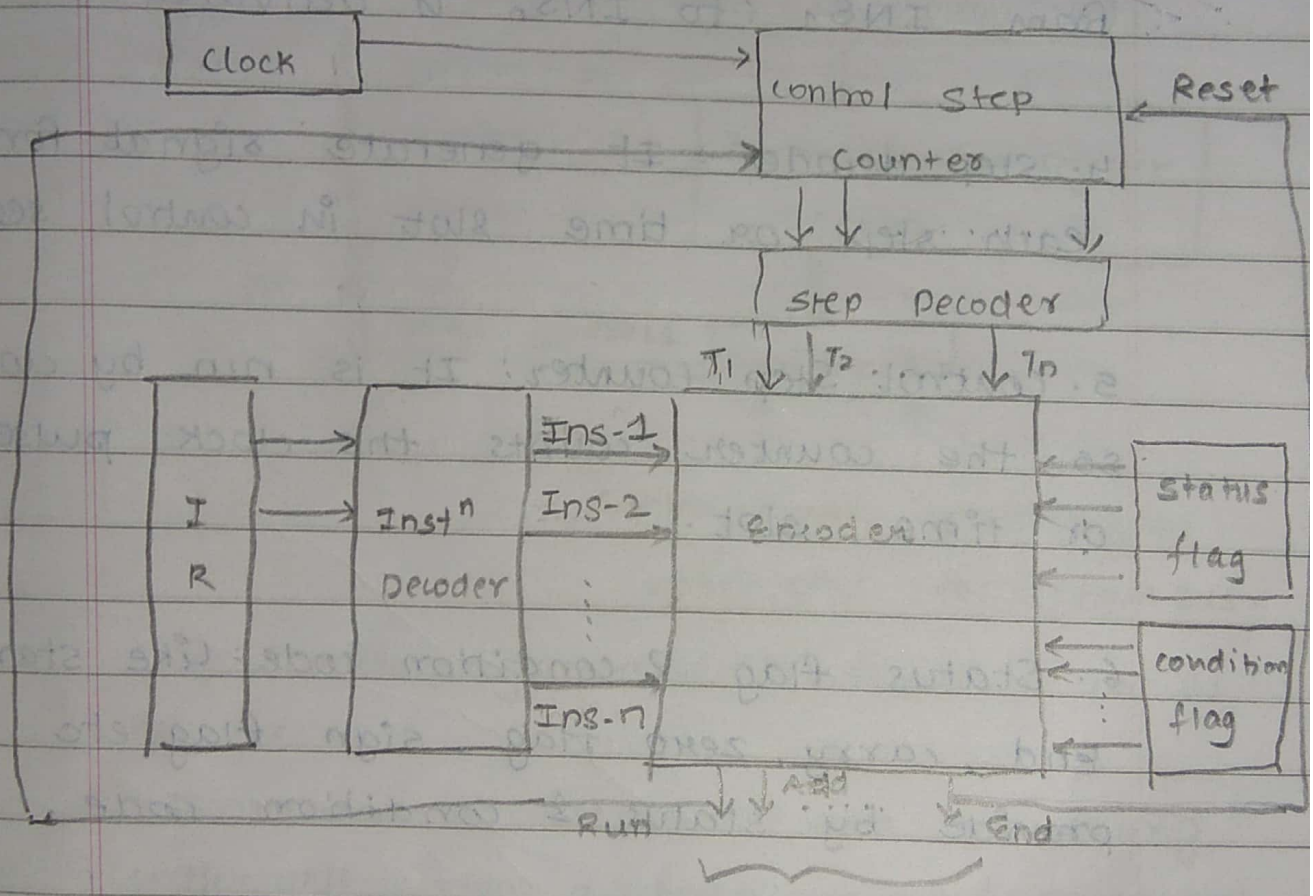


Fig: Hardwired control Unit Organization

It has different elements

1. Clock : With the help of clock the control unit maintain time. This is referred as processor cycle time.

2. Instruction Register (IR) : It store opcode of current inst^n the CU make the use of this opcode and will perform diff. action for aifferent action.

3. Instruction Decoder (ID): As usual the IR is input to decoder for any instruction loaded by IR, one of the output lines from INS_1 to INS_n is activated.

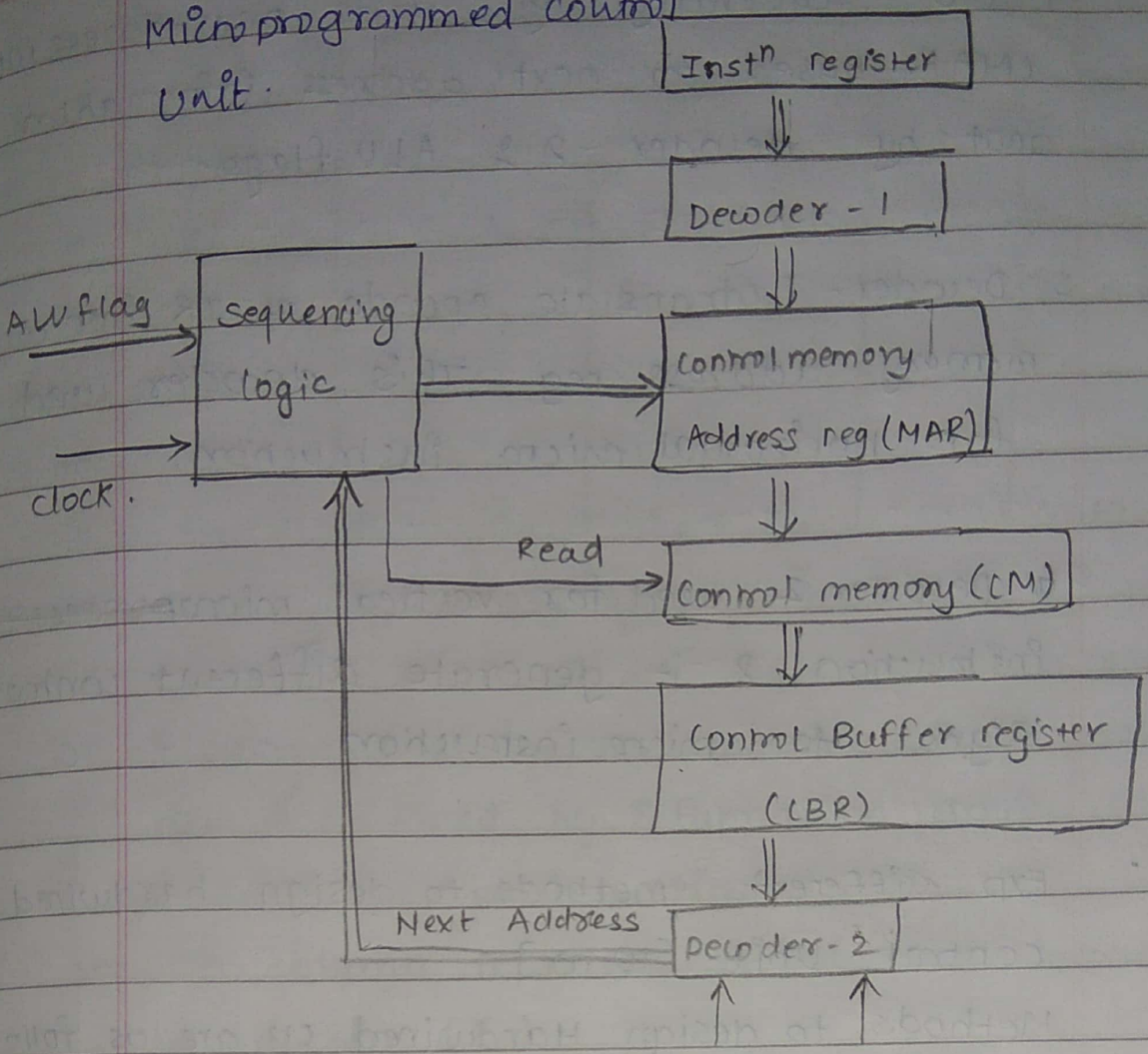
4. Step decoder : It generate signal for each step or time slot in control sequence

5. Control Step Counter: It is run by clock so the counter counts the clock pulse or time slot.

6. Status flag & condition code: Like start, end, carry, zero flag, sign flag etc provide by status & condition code.

7. Encoder: All input signal to the encoder block should be ~~considered~~ combined to generate different control signals.

Microprogrammed control Unit.



The microprogrammed CU function are as follows:

1. To execute an instruction, the sequencing logic unit issue a Read command to control memory.
2. The instruction whose address is specified by CMAR is Read by CM & stored in CBR
3. The CBR used to generate control signals & Next Address information for sequencing logic unit.

4. The sequencing logic units new address into CMAR based on next address information sent by decoder-2 & ALU flags.

5. Decoder-1 translate opcode of IR into control memory address reg. This decoder used for horizontal micro instruction.

6. Decoder 2 used for vertical micro~~processor~~ instruction & it generate different control signal for micro instruction.

- Exp different methods to design hardwired control Unit [10 m]

Methods to design Hardwired CU are as follows

1. State table method.
2. Delay Element method.
3. Sequence counter method.

1. State table method.

In this approach it is necessary to construct a state table for the control unit. The state table for this implementation derived from the flow chart directly, like any finite state machine representation

- Finite state machine are of two types.

(A) Mealy type

(B) Moore type

(A) Mealy type.

State	Input			
	I_1	I_2	.....	I_n
S_1	$S_{1,1}, O_{1,1}$	$S_{1,2}, O_{1,2}$		$S_{1,m}, O_{1,m}$
S_2	$S_{2,1}, O_{2,1}$	$S_{2,2}, O_{2,2}$		$S_{2,m}, O_{2,m}$
S_3				
⋮				
S_n	$S_{n,1}, O_{n,1}$	$S_{n,2}, O_{n,2}$		$S_{n,m}, O_{n,m}$

Table : Mealy type state table

Here the rows of the state table represent internal state of the machine. These states are determined by information stored in machine at various time unit.

- The column represent combination applied to machine (I_j)
- The entry of row S_i and column I_j from $S_{i,j}$ and $O_{i,j}$
- Where $S_{i,j}$ is the next state of machine that result from the application of input combination I_j
- Where $O_{i,j}$ denotes the output signal appears whenever machine is in the state S_i with Input I_j applied.

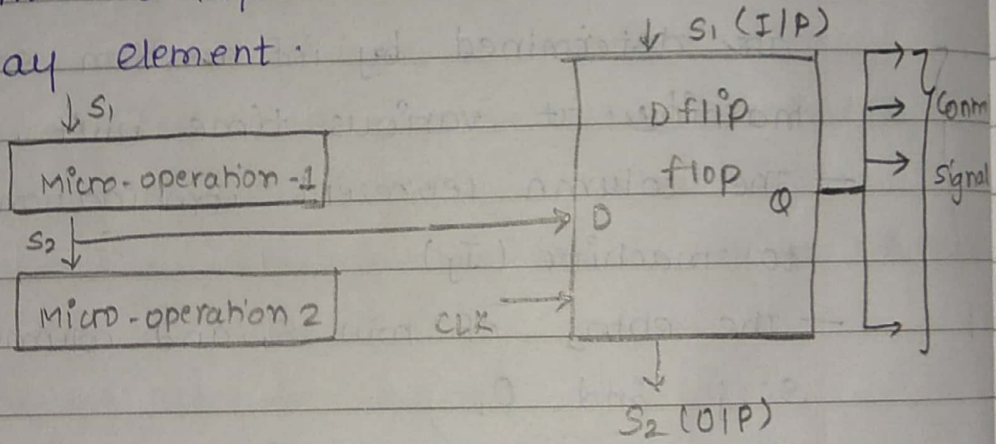
B. Moore type:

State	I_1	I_2	.....	I_m	output
S_1	$S_{1,1}$	$S_{1,2}$		$S_{1,m}$	O_1
S_2	$S_{2,1}$	$S_{2,2}$		$S_{2,m}$	O_2
⋮	⋮	⋮		⋮	⋮
S_n	$S_{n,1}$	$S_{n,2}$		$S_{n,m}$	O_n

2. Delay Element method:

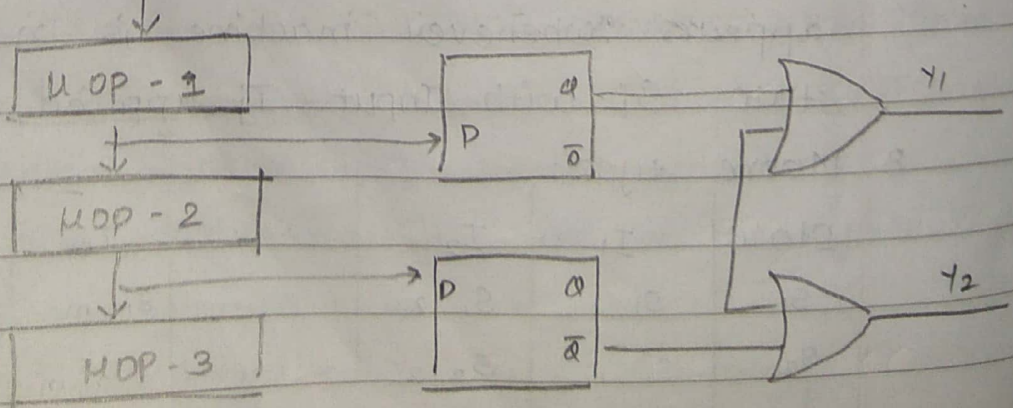
The control signals or groups of control signals from the control units are activated in a proper sequence. The control unit using delay elements can be make from flowchart that specifies required control signal sequence. These are simple rule to derive control circuit from the flowchart.

A. Rule 1: Two successive micro-operations required delay element. The control signals are taken from the input & output lines of the delay element.

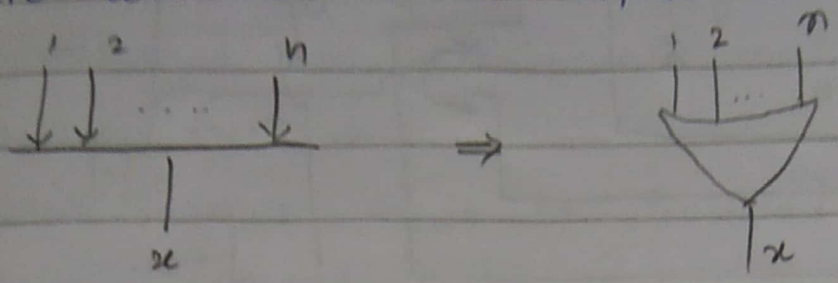


B) RULE - 2

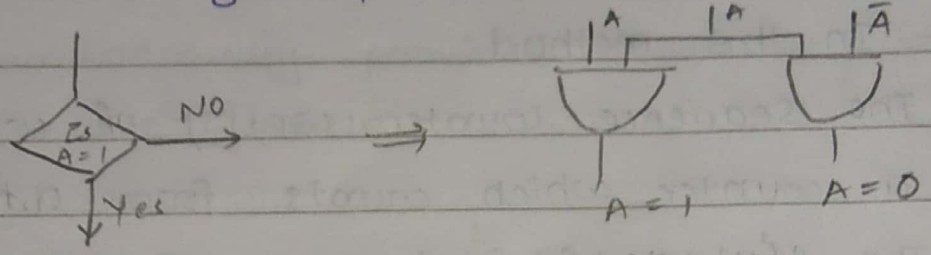
The signal that activate some control signals are ORed to get common output signal.



C. Rule 3: When n-lines in the flowchart merge to common point x then these lines are connected to 'n' input OR gate.

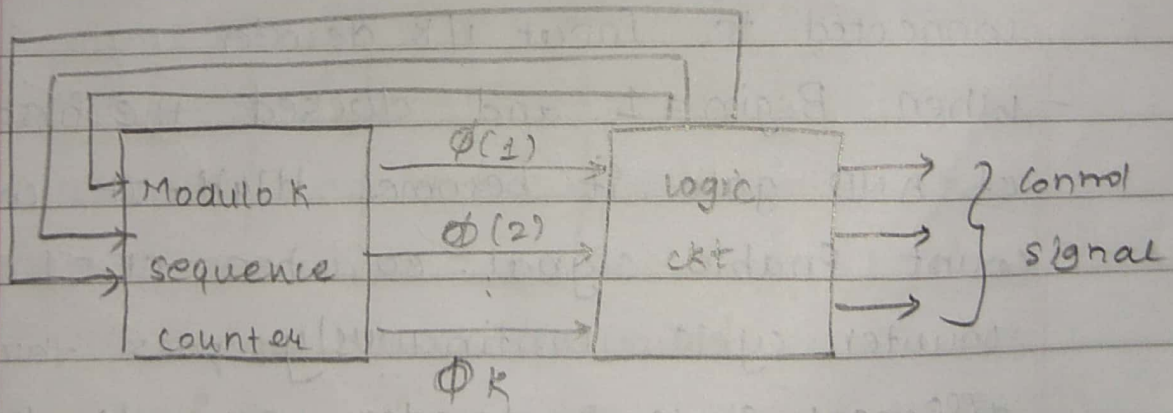


D. Rule 4: A decision box can be implemented by 2-AND gates. The input of AND gates derived by input A & complement of A

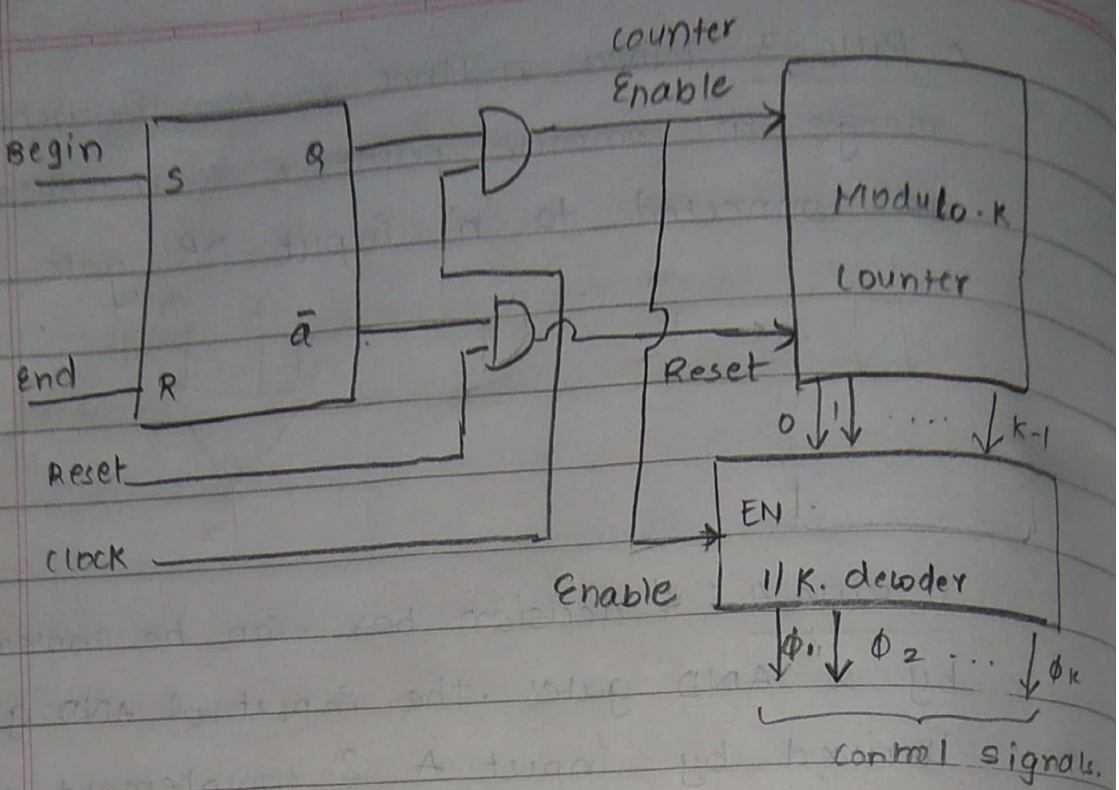


3. Sequence Counter Method:

- It shows a modulo-k sequence counter cycling through k-different state.



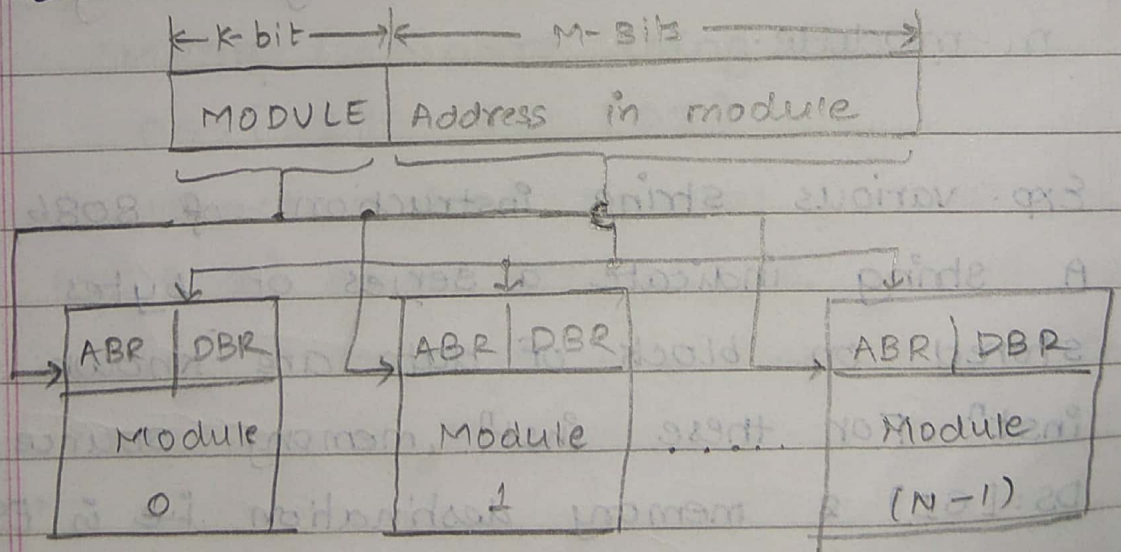
The logic circuit of counter take these k-output as input along with some addition input used for counting purpose to generate some control signals.



- In this method.
- The sequence counter consist of counter. It is a counter which counts from 0 to K-1 state
- The 1/K decoder and input signals ~~from~~ are used to control the operation of modulo-K counter i.e reset, clock, count Enable.
- The output ~~of~~ of modulo-K counter is connected to input 1/K decoder.
- When Begin = 1 and clocked the first input of AND gate, it becomes HIGH. It will set count Enable signal. so when CE = 1, the counter cycle continuously pass through different state & decoder. generate K-phase(Φ) signal of output.
- when Begin = 0 or End = 1, clock not ready to counter.
- when Reset = 1, it will off the counter

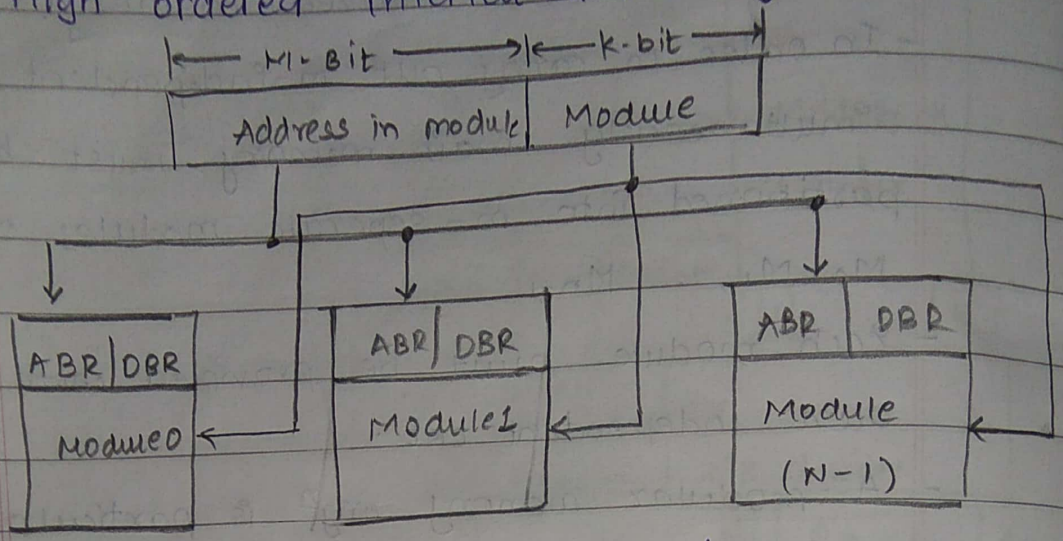
- ~~When~~ SN on Interleaved memory.
- In order to carry out m-independent access simultaneously, main memory must be partitioned into m-seperate modules or banks $M_0, M_1, \ldots M_{n-1}$
- Each module must be provided with its own independent ckt.
- A modular memory org^n is particularly useful in multiprocessor to a common memory. Different processor can access memory simultaneously provided that they reference seperate module.

Low order interleaved



low order interleaved
(consecutive word in a module)

High ordered interleaved memory.



High Ordered Interleaved
(Consecutive word in consecutive module)

— Inorder for this memory orgⁿ to used efficiently the memory references generated by processor should be distributed every among the m-module.

Exp. various string instruction of 8086 [10m]
A string indicate a series of bytes or word stored on block of data are known as string instⁿ. For these instⁿ, memory source is DS:[SI] & memory destination is ES:[DI] After performing string operation SI or DI are automatically incremented (or decremented by 1 or 2 depending on byte or word operation. If the direction flag (DF) is reset then automatic incrementing & vice-versa.

1. STOSB. (store String Byte)

ES: [DI] ← AL

- Store content of AL reg. into m.l pointed by ES:[DI]

2. LODSB (load string Byte)

AL ← DS: [SI]

- Load content of m.l pointed by DS:[SI] to AL reg
- After string operation, if DF is reset then the DI is ~~decremented~~ incremented by 1 otherwise it is decremented by 1

3. MOVSB [Move String Byte]

ES: [DI] ← DS: [SI]

- Move the content of m.l pointed by DS:[SI] to ES: [DI]

4. CMPSB (compare string Byte)

DS: [SI] - ES: [DI]

- Subtract content of m.l pointed by ES: [DI] from DS: [SI] & content of m.l will not change any flags are affected i.e compare operation.

5. SCASB (Scan string Byte)

AL - ES: [DI]

- subtract content of m.l pointed by ES: [DI] from AL only flags will affect

1. STOSW
2. LODSW
3. MOVSW
4. MOVSW
5. SCASW.

COA

* XLAT :

MOV into AL, the content of memory location in Data segment, whose effective address is formed by the sum of BX & AL

Eg: XLAT ; AL ← DS : [BX + AL]

; i.e IF DS = 1000, BX = 0200, AL = 03

; 10000 (DS x 10)

\+ 0200 (BX)

03 (AL)

10203 ∴ AL ← [10203H]

* LAHF

load AH with lower byte of flag reg

* SAHF

- Stores the content of AH into lower bytes of flag reg.

* PUSH source -

eg. 1. PUSH CL; SS : [SP - 1] ← CL

2. PUSH CX; SS : [SP - 1] ← CH

SS : [SP - 2] ← CL

* POP source

eg 1. POP CL ; CL → SS : [SP]

SS : [SP + 1]

2. POP CX CL ← SS : [SP + 1]

CH ← SS : [SP + 2]

SS : [SP + 3]

* AAA [ASCII Adjust for Addition]
(for AX reg only)
- In ASCII codes, 0 to 9 are represented by 30 to 39.
- It will convert content of AL in ASCII after addition.

eg. MOV AL, 04H
MOV BL, 03H
ADD AL, BL
AAA

* AAS (ASCII Adjust for subtraction)
* AAM (BCD Adjust after multiplication)
* AAD (Binary adjust before Division)

(MS)

Explain various memory mapping techniques of cache memory.

- As there are fewer cache blocks [lines] than main memory block, an algorithm is needed for mapping main memory blocks into cache lines (blocks)
- Further a mean is needed for determining which main memory block currently occupies a cache lines. The choice of the mapping function required for that mapping technique play very important rule.
- Three techniques can be used
1. Direct
2. Associative
3. set associativeness

1. Direct mapping.

— The simplest technique, known as direct mapping maps, each block of main memory into only one possible cache line.

— The mapping expressed as

$$i = j \text{ modulo } m.$$

where,

i = cache Block number

j = main memory Block number

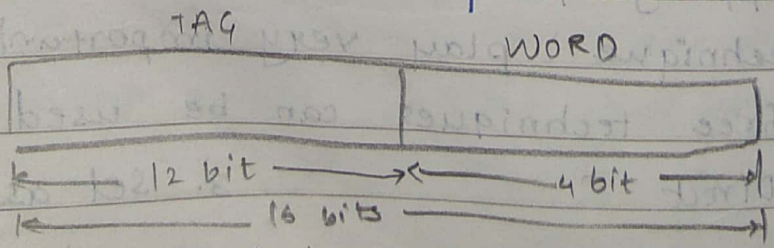
m = number of block (lines) in the cache memory.

2. Associative mapping

— Associative mapping overcome the disadvantage of direct mapping by permitting each main memory block to be loaded each into any block of cache memory.

— In this case, the cache control logic interprets a memory simply as a Tag & a word field. The tag field unique identifies a block of main memory.

— To determine whether block is in cache, the cache control logic simultaneously examine every line (Block) tag of match.



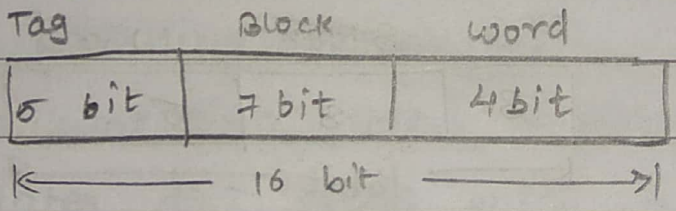
TAG = No of Block in main memory.

WORD = No. of Bytes in Blocks.

3. Set Associative memory

- Set associative mapping is a compromise that exhibits the strength of both the direct & associative approaches which reducing their disadvantages
- This is referred as k-way set associative mapping
- For set associative mapping, each word map into all the cache lines in the specific set,
- So that main mem. block B0 maps into set and so. on.

Q. Assume main mem. size is 64 KB & cache memory consist of 128 Blocks of 16 bytes each Find Tag, Block, word of direct mapping.



Given main mem = 64 KB

$= 2^6 \times 2^{10} = 16$ bit

Word = 16 Bytes $= 2^4$

BLOCKS = No. of Block in cache $= 128 = 2^7$

Tag = $\frac{\text{Total No. of Block in main mem}}{\text{Total No. of Blocks in cache memory.}}$

(15) Cache mapping technique.

let main memory consist of 2048 Blocks of 32 words each & cache memory coutain 256 Blocks.

1. Find TAG, Block,coord using direct mapping
2. find TAG & WORD of fuly associative mapping
3. find TAG, SET, WORD using 4-way set associative technique.

Given: 1. No of Blocks in main memory = 2048

2. Block Size = 32 Bytes.

3. Size of main memory = 2048×32

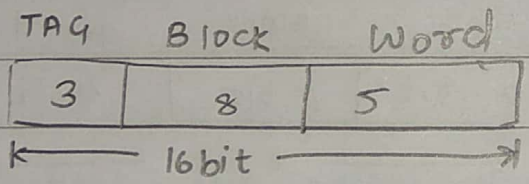
$= 2^{11} \times 2^5 = 2^{16}$

$= 64$ KB

∴ 16 bit is req^d to access main memory

4. No of BLOCKS in cache mem = 256

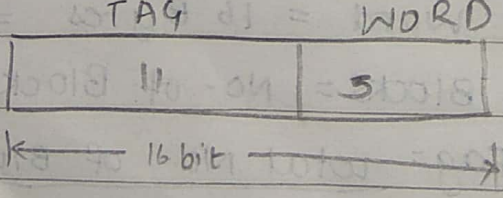
A.



Word = 32 bytes = 2^5

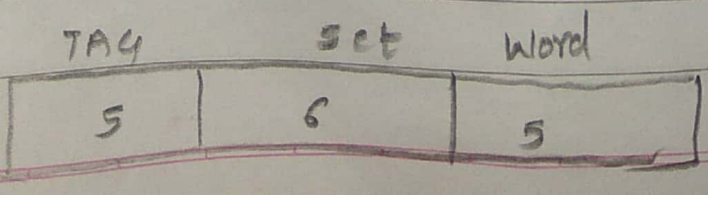
Block = No of block in cache mem i.e $256 = 2$

B.



C. For 4-way associative means

No of Blocks in set = $\frac{\text{No of Blocks in cache}}{4}$



(115)

SN on cache coherency.

In multiprocessor systems, another Bus master can take over control of system. This Bus master could write data into a main memory blocks which are ~~otd~~ already held in the cache of another processor. When it happens the data in the cache no longer match those held in main memory. creating ~~incos~~ inconsistency. These are different method to cache coherency.

1. Bus watching (snooping)
2. Hardware transparency
3. Non cacheable memory
4. Cache flushing.

1. Bus watching (snooping)

In Bus watching, cache memory controller invalidates the cache entry, if another master writes to a location in shared memory which also resides in the cache memory.

2. Hardware transparency:

In this access of all devices of the main memory are routed through the same cache or by copying all cache write both to the main memory & to all other cache that share same memory.

3. Non cacheable memory - In this partitioned main memory into a cacheable & non cacheable memory. By designing shared memory as non cachoable memory coherency can be maintained, since shared memory is never copied into cache.

4. Cache flushing. To avoid data inconsistency, a cache flush write any altered data to the main memory, & cache before the system flushed the shared memory.