* Computer Architecture and organisation *

* Binary - multiplication :-

Multiplicand (M) = 1010

multiplier (Q) = 0110

$\Rightarrow$

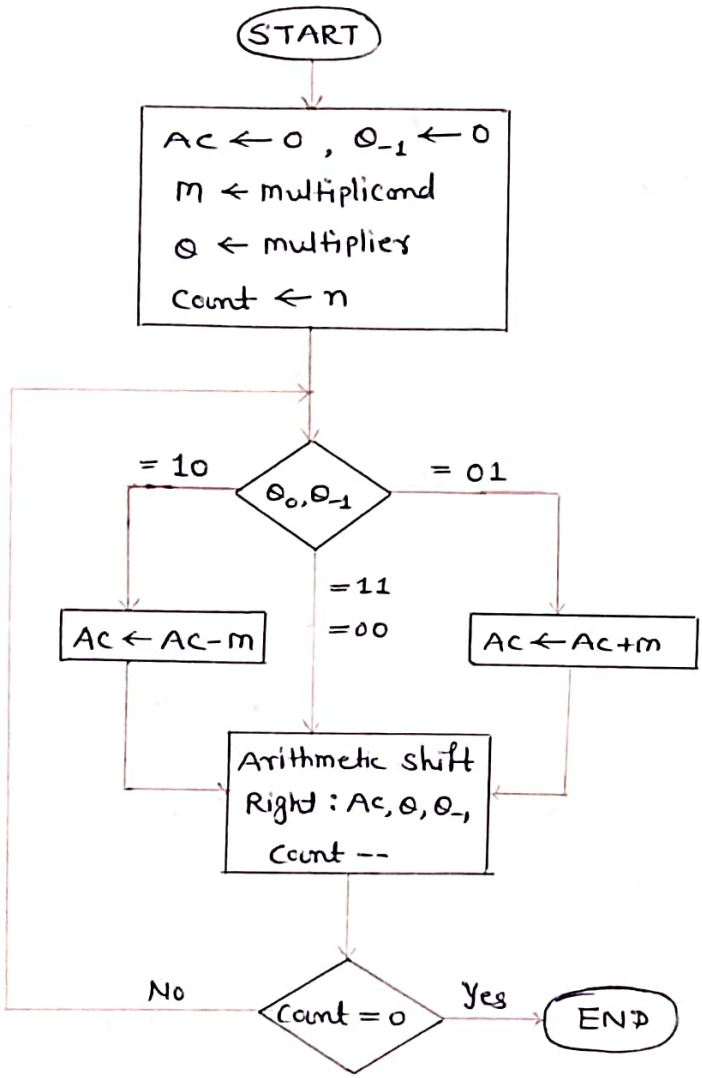
```
    1 0 1 0
  x 0 1 1 0
-------------
    0 0 0 0
  1 0 1 0 x
1 0 1 0 x x
0 0 0 0 x x x
-------------
0 1 1 1 1 0 0
```

Partial product

if |M| = n bits, |Q| = n bits

$\Rightarrow$ |M x Q| = 2n bits

* Booth's Algorithm :-



Exa - Multiply - 5 x (-6)

M = 5 = 0101 Count = 4

Q = -6 = 1010

-M = -5 = (1010 + 1) = 1011

OPR.	AC	Q	Q-1	
	0000	1010	0	
ASR	0000	0101	0	Count = 3
-M	1011			
	1011	0101	0	
ASR	1101	1010	1	Count = 2
+M	0101			
	0010	1010	1	
ASR	0001	0101	0	Count = 1
-M	1011			
	0100	0101	0	
ASR	0010	0010	1	Count = 0

Ans - 00101 0010

11101 0 (100010)$_2$

$\Rightarrow$ (011101 + 1)$_2$

= (011110)$_2$ $\approx$ 30

Exa - M = -7 = 1001

-M = 7 = 0111

Q = -3 = 1101

Count = 4

OPR	AC	Q	Q-1	
	0000	~~0000~~ 1101	0	
-M	0111			
	0111	1101	0	Count = 3
ASR	0011	1110	1	
+M	1001			
	1100	1110	1	Count = 2
ASR	1110	0111	0	
-M	0111			
	0101	0111	0	Count = 1
ASR	0010	1011	1	
ASR	0001	0101	1	Count = 0

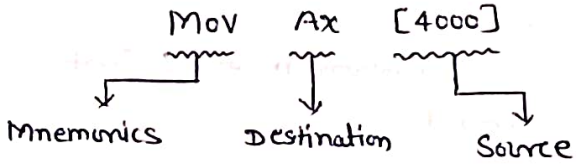
~~* System Bus :-~~

* Byte ordering (Endian mechanism)

- Default Data type in memory chip is Byte. So in the memory chip data is stored Byte wise.
- Default Data type in the CPU is 'word'. So in the CPU opⁿ are performed based on word format.

* When the word length of the processor is greater than 8 bit (16bit or 32bit) then multiple cell (multi Byte) Accessing to process the data simultaneously. So Endian mechanism is used.

Exa – (a) 16 bit processor –



Word length = 16 bit = 2 Byte

$$Ax \leftarrow \begin{cases} m[4000] \\ m[4001] \end{cases}$$

(b) 32 bit processor –

Mov R_L [4000]

Word length = 32 bit = 4B

$$R_L \leftarrow \begin{cases} m[4000] \\ \vdots \\ m[4003] \end{cases}$$

if data = 0X 1A 2B 3C 4D

R_L = 0X 1A 2B 3C 4D

OR

0X 4D 3C 2B 1A

2 Types – 1) Little Endian
2) Big Endian

(1) Little Endian :-

- Lower Address Contain Lower Byte & Higher Address Contain higher Byte.

OR

- Right to Left
- Store LSB / Little end at Lower memory Address

Exa – data = 0X 11215156

Starting Address = 1000

11	21	51	56

1000	56
1001	51
1002	21
1003	11

(2) Big Endian :-

- Lower Address Contain Highest Byte & Higher Address Contain Lower Byte.

OR

- Left to Right

Exa data = 0X 11 21 51 56

Starting Address = 1000

11	21	51	56

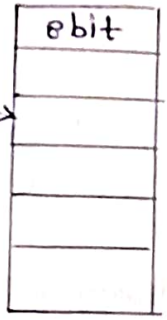
1000	11
1001	21
1002	51
1003	56

* Memory :-

(1) Word Addressable

1 cell = 1 word
= 16 bit /
32 bit /
64 bit

(2) Byte Addressable



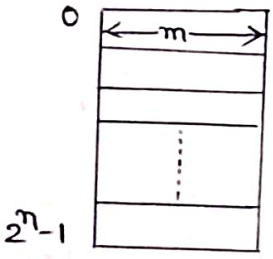
1 cell = 8 bit

Cell

Memory = $2^n \times m$

n: Number (#) of Address line

m: Number (#) of Data Line



$M = 2^2 \times 8$ bit

$n = 2$

$m = 8$

$2^{10} = 1024 = 1K$

2^{20} = 1 mega (1M)

2^{30} = 1 Giga (1G)

2^{40} = 1 Tera (1T)

2^{50} = 1 peta (1P)

Ex- 1K Byte -

$2^{10} \times 8$ bit

n = 10 bit A.L

m = 8 bit D.L

2^{10} memory Cells

64KB

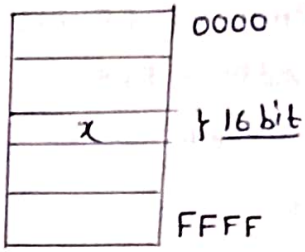
$= 2^6 \times 2^{10} \times 8$ bit

$= 2^{16} \times 8$ bit

n = 16 bit A-L

m = 8 bit Data line

2^{16} Cells



Hexadecimal – 0X , $( )_{16}$, $( )_H$

Exa - 0111 1010

$(7A)_H = 0X7A = (7A)_{16}$

* for n-bit processor –

- Byte Address – 8 bit
- word Address size – n bit

Q A processor can support a maximum memory of 4 GB, where the memory is word addressable (a word consists of 2 B).
The size of the address Bus.

Ans – Memory = 4GB

$= 2^2 \cdot 2^{30}$ B

$= 2^{31} \times 2$B

→ 31 bit Address

Q Consider a system which has 1024 K words. Each word has the size of 32 bits then what is the capacity of m/m in MB.

Ans – memory = 1024 K x 4B

$= 2^{22}$ B

= 4mB

* Instruction cycle :-

↳ The process required for each Instruction execution.

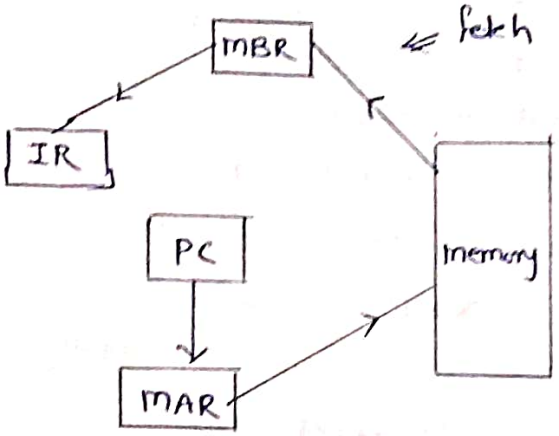
↳ 2 types - (1) Fetch cycle
(2) Execute cycle

(1) Fetch Cycle :-

- To fetch the instruction from memory to CPU(IR).
- At the End of fetch cycle, Pc is incremented, Now Pc denotes next instruction starting address.

(2) Execute cycle :-

- The objective of the execute cycle is to Execute the fetched Instruction.
  - → Decode
  - → Execute



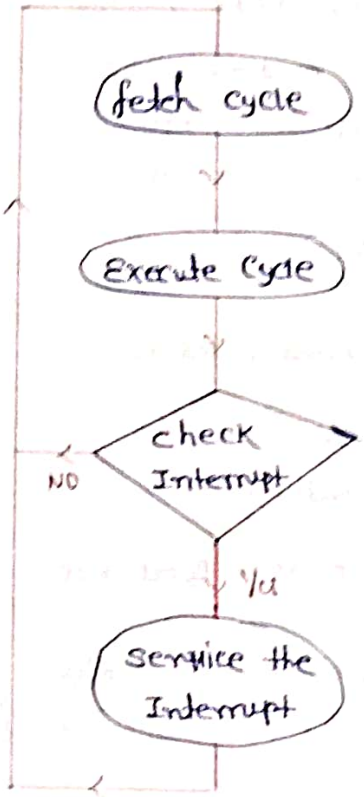
Steps :-

1) Instruction Address Calculation } fetch
2) Instruction fetch
3) Decoding } Decode
4) Operand Address Calculation
5) Operand fetch
6) Data processing } Execute
7) Result Storage

- Steps of fetch cycle :-
- At the beginning of Each Instn cycle the Processor fetches an Instn from m/m.
- Pc holds the address of the instruction to be fetched next.
- Pc ← Pc + 1
- fetched Instn Loaded into the IR.
- The processor interprets the instruction and Perform required action.

(2) Interrupt cycle :-

* When CPU encounter the Interrupt then after Completion of current Instruction Execution, Interrupt will be serviced.
* When CPU encounter the Interrupt, It Push Pc Value into stack (as a return Address & Control transfer to ISR (Interrupt sub routine)



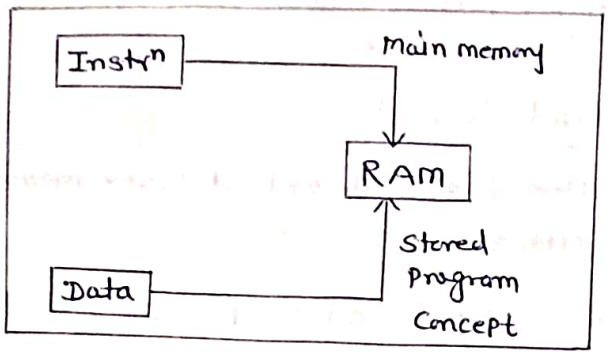
1000 I_1
1001 I_2
1002 I_3 → Interrupt occur during the Execution of I_3.
1003 I_4

→ 1003 will be stored in stack

* Computer Architecture :-

→ Based on Program Storage, Computer Architecture is of 2 Kinds :

(1) Von-Newmann Architecture (single memo. Arch.):-

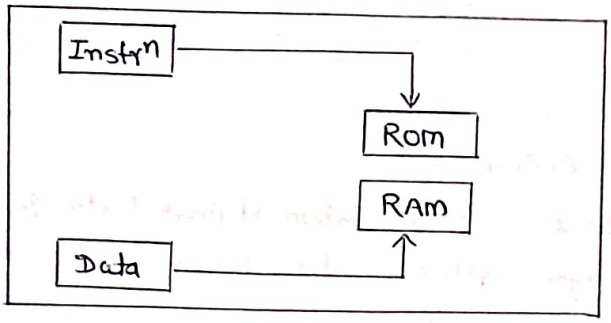


Exa:- DESKTOP Computer

8085 µp — 64 KB RAM

8086 µp — 1 MB RAM

(2) Harvard Arch. (Multi-memory Arch):-



→ Used by Embedded system

→ 8051 µ-Controller (4KB Rom, 128 B RAm)

* Components of Computer :-

1) CPU (ALU, CU, Register)
2) memory
3) I/o

CPU Registors —

(i) Program Counter (PC)
(Starting Address of Next Instrⁿ)

(ii) MAR (memory Address Registor):-
- Hold Address

(iii) MBR (memory Buffer Registor):-
- Hold Instrⁿ or Data

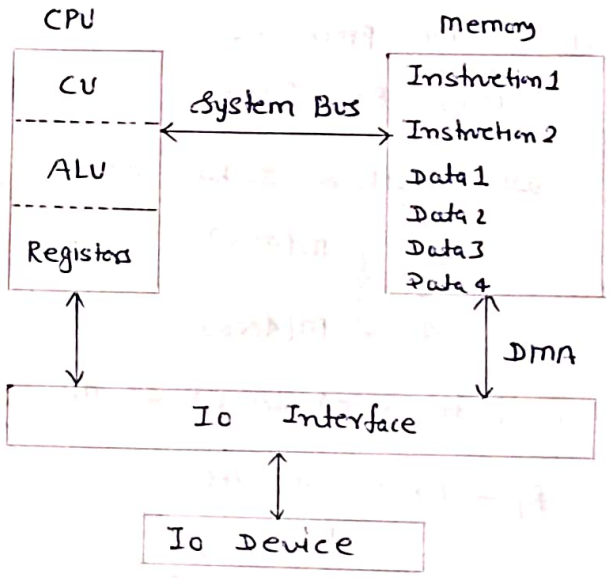
(iv) IR (Instruction Registor):-
- Contain Instruction currently being executed by CPU.

(v) AC (Accumlator):-
- Store Temporary result or First operand.

(vi) GPR (General Purpose Register)

(vii) SP (Stack Pointer)

(viii) PSW / flag Register



* Booth's Coding :- (Bit-Pair Method)

→ Transformation -

-1	1 ← 0
0	0 ← 0
0	1 ← 1
+1	0 ← 1

$(-57)_{10} = (1\ 1\ 0\ 0\ 0\ 1\ 1\ 1)_2\ (0)$ ← LSB

$\equiv (0\ -1\ 0\ 0\ +1\ 0\ 0\ \bar{1}.)$ → ASR

AC = AC - M ASR

AC = AC + M ASR

Q. M = 1010 1101 1011

Q = 1001 1010 1001

→ (i) Recorded multiplier

(ii) How many Arithmetic opr are required in the multiplication.

Ans - $Q = (1001\ 1010\ 1001)_2\ (0)$

$\Rightarrow (-1\ 0\ +1\ 0\ -1\ +1\ -1\ +1\ -1\ 0\ +1\ -1)$

+1 : AC = AC + M

-1 : AC = AC - M

total opr = 9

* Floating Point Representation :-

$\Rightarrow \pm M \times B^{\pm e}$

↳ Mantissa

→ IEEE 754 floating point Rep. :-

Sign (s)	Exponent (e)	Mantissa
1	7/8	24/23

(32-bit)

Sign (s)	Exponent (e)	Mantissa
1	11/12	52/51

* Biased Exponent (BE) = AE + Bias

↓

(Actual Exponent)

BE Field size	BE Range	Bias	Excess Bias
6-bit	-32 to 31	+31	
9-bit	-256 to 255	+255	
n-bit	-2^{n-1} to $2^{n-1}-1$	$+(2^{n-1}-1)$	2^{n-1}

Exa- $(0.01101)_2 \rightarrow (0.1101 \times 2^{-1})$

$(101.101)_2 \rightarrow (0.101101 \times 2^3)$

S	BE	Mantissa
(1)	(4 bits)	(5 bits)

$bias = 2^3 - 1 = 7$

$BE = 3 + 7 = 10 = (1010)_2$

Mantissa = 10110 1

0	1	0	1	0	1	0	1	1	0	1

↳ Explicit Normalization

Exa

S	BE	Mantissa
1	7	12

data = -13.25×2^{15}

$13.25 \equiv (1101.01)_2$

data = $-(0.110101) \times 2^{15}$

BE = 15 + 63 = 78

$= (1001110)_2$

BE = $(1001110)_2$

Mantissa = $(110101)_2$

sign = 1

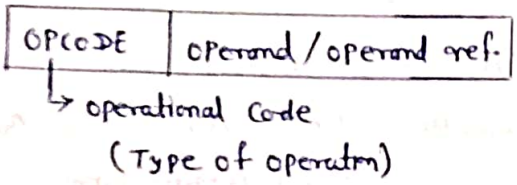
1 0110101 10011100000 0

$(B59C0)_{16}$

* Instruction :-

* Instruction is an binary sequence (binary code) which designed inside the processor to perform the task/ operation.

Binary sequences → Bind with - operation



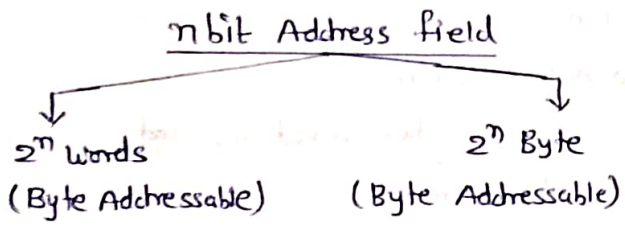
* n bit opcode can perform 2^n operation.

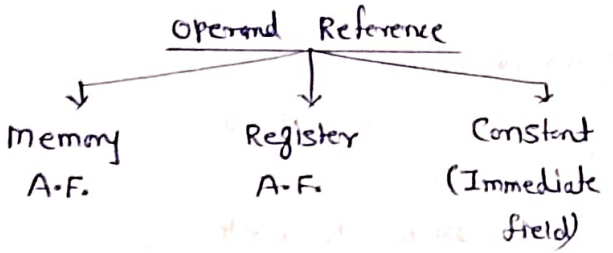
* Operand → 'Data'

Operand → Address of data Reference

* n bit Address field can represent 2^n memory cells.

Exa 1KB → 10 bit address





⇒1) If opcode bits is given then ...
Total # operation / Instruction.

Opcode bits → total # operations

1 bit	→	$2^1 = 2$
2 bit	→	$2^2 = 4$
4 bit	→	$2^4 = 16$
6 bit	→	$2^6 = 64$
n bit	→	2^n

= Is size

⇒2) total operations → # bits in opcode

50	→	$\lceil \log_2 50 \rceil = 6$ bit
70	→	$\lceil \log_2 70 \rceil = 7$ bit
500	→	$\lceil \log_2 500 \rceil = 9$ bit
135	→	$\lceil \log_2 135 \rceil = 8$ bit
N	→	$\lceil \log_2 N \rceil$

Q3 if memory is 256KB then what is mem. address field ?

Ans- Memory = 256KB = $(2^8 \cdot 2^{10})$ B

= 2^{18} B

Memory Address Field = 18 bit

Q4 if there are 25 Register then what is Reg AF (# bits Required to Represent Register) ?

Ans- # Register = 25

Reg. AF = $\lceil \log_2 25 \rceil$ = 5 bit

⑤ Consider a 32 bit hypothetical processor, byte Addressable memory. CPU accessing data word from memory starting address $(23C4)_H$

Memory Content =

$(BB)_H$ 23C2
$(99)_H$ 23C3
$(AA)_H$ 23C4
$(BB)_H$ 23C5
$(CC)_H$ 23C6
$(DD)_H$ 23C7

Ans ⇒ 32 bit ≡ 4B readed

↳ Ans = AA BB CC DD

* System Bus Configuration:-

(A) I/O Processor:- (IOP)

This Configuration uses:-

Common Control signal

- Common Address space, diffrent for memory and I/O
- Additional Hardware required for high performance CPU design.

⊕ → Parallel processing

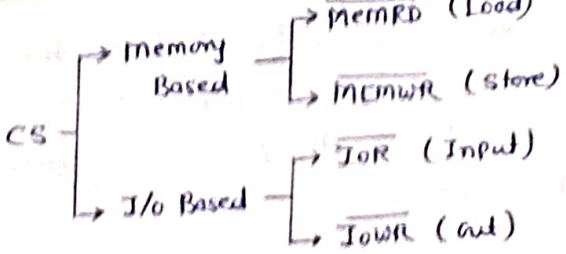
Reduced CPU Load - Iop manages IO independentally.

Faster Data Transfer - Uses DMA (Direct m/m Access) for Efficient data movement.

- Iop and CPU Transfer Data Seperately.

(B) Isolated I/O:-

→ seperate I/O and memory addresses
→ used only for I/O devices.
→ Uses $IO/\overline{M}$ signal to diffrentiate I/O
→ Slower dve to extra I/O Instructions
→ different Control signal for memory & I/O



$IO/\overline{M}$	$\overline{RD}$	$\overline{WR}$	
0	0	1	$\overline{MEMRD}$
0	1	0	$\overline{MEMWR}$
1	0	1	$\overline{IOR}$
1	1	0	$\overline{IOWR}$

(C) Memory-mapped I/O:-

→ It will Common Bus, different address Bus
→ Shared b/w memory & I/O
→ No special Control signals for I/O.
→ Uses Normal memory operations (LDA/STA)
→ Faster (direct memory operations)

(4) Dual Pin :-

$M/\overline{IO}$ OR $\overline{M}/IO$

↳ 0 I/O Read and write

→ 1 memory Read and write

* 8085 µP :-

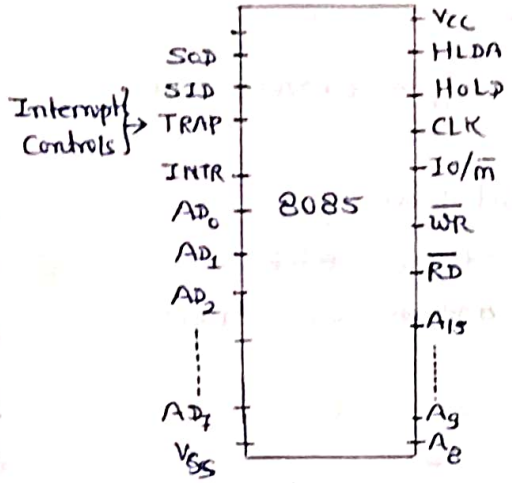
↳ 8 bit processor (8 bit data Bus)

→ 16 bit Address Bus

↳ (64 KB)

→ 40 Pins

→ $(AD_0 - AD_7)$ $(A_8 - A_{15})$



Exa - LDA [2040] → [2040_H | RD]

. Load to Accumulator from memory

(i) Send the address to memory.

ALE = 1

40 - $(AD_0 - AD_7)$

20 - $(A_8 - A_{15})$

(ii) Send the Control signal -

$IO/\overline{M} = 0$ (memory operation)

$\overline{RD} = 0$, $\overline{WR} = 1$

(iii) memory Location 2040_H Places data on the data Bus $(D_0 - D_7)$

8085 reads the data and stores it in the AC.

(iv) $\overline{RD} = 1$

* 8086 µP :-

↳ 16 bit processor (16 bit Data bus)

→ 20 bit Address Bus

↳ (1 MB memory)

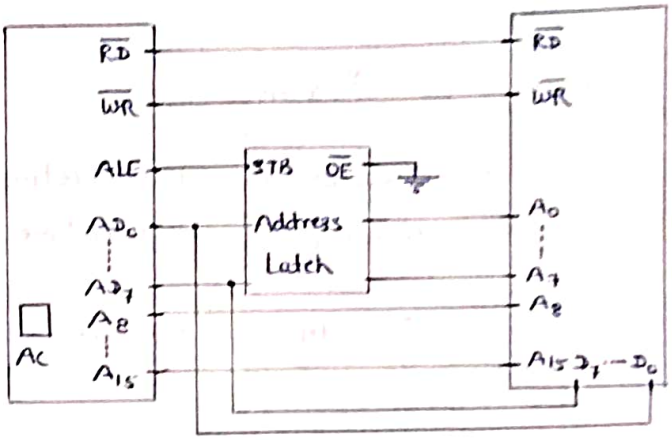
→ $(AD_0 - AD_{15})$ $(A_{16} - A_{19})$

→ $MN/\overline{MX}$ = 0 (Maximum mode)

↳ Enables multiuser multiprocessor operation

* Memory Interfacing :-

• This Concept used to map memory pins with CPU pins.



Store Pin - (STB) → 0 (Input line disable)

1 (Input line Enable)

OE (output enable) - 0 → output line Enable

1 → disable

Q 32MW memory, Word = 32 bits

The processor is enhanced with byte addressable, how many A_x Pins required?

Ans ⇒ Word size = 32 bits

= 4 B

memory = 32MW

$= 2^5 \times 2^{20} \times 2^2$ B

$= 2^{27}$ B

A_x Pins required = 27

*System Bus :-

- System Bus is a Collection of lines/wires which are used to provide the Communication between major Component of the Computer (CPU, memory, I/O) (1 Line Carry 1 bit at a time)

1) Address line (AL):-

- AL is used to Carry the Address towards memory and I/O.
- AL are Unidirectional.

(2) Data Line (DL) :-

- DL are used to Carry the Data (Binary Sequence)
- DL are Bidirectional.

(3) Control Line :-

- Control line Carries the Control signal.
- Control line individually unidirectional.

→ Address Bus – Collection of AL
→ Data Bus – Collection of DL
→ Control Bus – Collection of Control line

32 bit Processor	24 bit Address
Word size = 32 bit	Address line = 24 bit
Data line = 32 bit	Address Bus = 24 bit
Data Bus = 32 bit	MAR = 24 bit
MBR = 32 bit	AR = 24 bit
MDR = 32 bit	PC = 24 bit
GPR = 32 bit	
ALU = 32 bit	
AC = 32 bit	

* n AL ≡ 2^n memory cells
* 8085 µP – ($AD_0 - AD_7$ & $A_8 - A_{15}$)
  – 16 bit Address Line
  8086 µP – ($AD_0 - AD_{15}$ & $A_{16} - A_{19}$)
  – 20 bit Address Line.

* DL–
  8085 µP – Word length = 8 bit
  Operation performed on 8 bit data
  8086 µP – Word length = 16 bit

*Pins :-

(A) Active Low Pin:-
- Enable when input = 0
  → $\overline{Pin\_name}$
  → E.g. – $\overline{RD}$, $\overline{WR}$

(B) Active high Pin:-
- Enable when input = 1
  → Pin_name
  E.g. – INTR, HLDA etc.

(C) Time multiplexed pin:-
- This pin Carries multiple meaning But only one at a time.

E.g. – 8085 – ($AD_0 - AD_7$)
AP DP

Address Latch Enable → 1 Time multiplexed Pin – Address
ALE
→ 0 – Data

Word size = Number of DL = Data Bus width
⇓
Unit of Data According to CPU

Address Register = MAR = Address Bus width
(AR)

* Instruction Execution sequence :-

1) Implicit mode :- PC ← PC + step size

2) Explicit mode :- PC Loaded with the Effective Address.

Exa -

Instrⁿ	(word) Size	Address	
I_1	2	1000	+8
I_2	1	1008	+4
I_3	1	1012	+4
I_4	2	1016	+8
I_5	1	1024	

∵ 32 bit system

1 word = 32 bit = 4B

PC during I_4 = 1024

Q A CPU has 24 bit Instrⁿ which is Storing the program 300 words onwards which one PC is valid ?

(A) 400 (B) 500

✓(C) 600 (D) 700

Ans - 24 bit ≡ 3B

$PC = 300 + 3n$

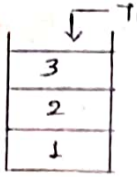
$n = 100$

$PC = 600$

* Instruction format :-

1) Stack Machine -

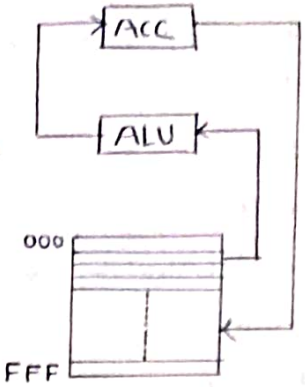
- Zero Address Instrⁿ (0A1 / 0AF)
- OPCODE



Pop, Por, Add, Push

(2) Accumulator machine :-

- One Address Input (1A1 / 1AF)
- OPCODE | Addr



Exa - | ADD | [3000] |

AC ← AC + M[3000]

(3) General Registers orgnization :-

* 3A1 / 3AF

OPCODE	A_1	A_2	A_3

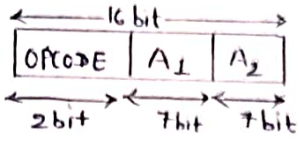
* 2A1 / 2AF

OPCODE	A_1	A_2

* → Register to Register Reference [RISC]
  → Register to memory Reference (Result storing reg)

Q Consider a hypothetical Processor, which support both 1 & 2 Add. Instruction. It Contains 128 word memory, where the word size is 16 bit, If one Instrⁿ is stored in 1 memory word & if there exists two 2 Add Instrⁿ. How many one Add. Instrⁿ Can be formulated ?

Ans



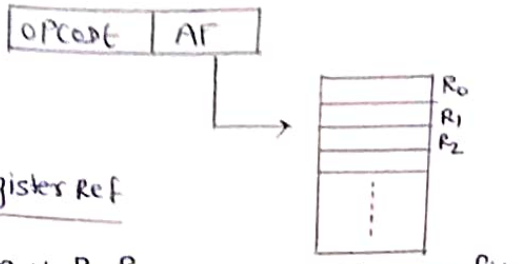
Opcode = 00, 01, 10, 11

No. of 1 Address Instruction = $2 \times 2^7 = 2^8$

= 256

(4) Register Direct Am:-

→ Register Address (Reg Name) is maintained in the AF of the Instrⁿ.



1 Register Ref

Eg. - MOV R_0 R_1

(5) Register Indirect Am:-

→ EA is stored in the Register and Operand are present in the memory.

E.g. - MOV R_1, @R_2 / MOV R_1, (R_2)

$R_1 \leftarrow m[R_2]$

(6) PC Relative Am:-

EA = Current PC Valve + AF / offset

EA = Updated / current PC Valve + Relative val AF / OFFset

(7) Base Register Am:-

EA = BR Valve + AF / OFFset

(8) Indexed Register Am:-

EA = XR-Valve + AF / OFFset

(9), (10) Auto Decrement / Increment Am:-

→ similar to Register Indirect Am in which Reg. valve Decreament / Increment respectively. EA = (R_i)+

EA = -(R_i)

(11) Implicit / Implied Am:-

→ In this Am operand are present in opcode itself.

CLC : Clear Carry (Cy = 0)

STC : set Carry (Cy = 1)

Exa -

Auto Increment -

ADD R_1, (R_2)+

$R_1 \leftarrow R_1 + m[R_2]$, $R_2 \leftarrow R_2 + 1$

Auto Decrement -

ADD R_1, -(R_2)

$R_2 \leftarrow R_2 - 1$, $R_1 \leftarrow R_1 + m[R_2]$

PC Relative Am -

Load A(PC) // AC $\leftarrow m[PC + A]$

Index Register -

ADD R_1, 20(R_2)

$R_1 \leftarrow R_1 + m[20 + R_2]$

Base Register Am -

ADD R_1, 20(R_2)

$R_1 \leftarrow R_1 + m[20 + R_2]$

EA = 20 + [R_2]

* Addressing modes:-

→ Am is a technique used to Calculate the Effective Address of the operand.

→ Location of the operand.

→ Effective Address (EA) is the Actual Address of the operand.

→ Types of Am:-

1) Immediate Am
2) Memory Direct (Absolute) Am
3) memory Indirect Am
4) Register direct Am
5) Register Indirect Am
6) PC-Relative Am
7) Base-Register Am
8) Index Register Am

} displacement Am

9) Implied / Implicit Am
10) Auto decrement Am
11) Auto Increment Am

→ Symbols of Am-

1) Immediate Am ⟶ I or #
2) Direct Am ⟶ [ ]
3) Indirect Am ⟶ @ or ([ ])
4) Indexed Am ⟶ Index Reg Name
5) Register Am ⟶ Reg Name

(1) Immediate Am:-

• Immediate Am is used to Access the Constant or to Initialize the Register or ~~the~~ Variable with value.

Eg MVI R_1 100

OR

MVI R_1, #100

MOV R_3 #7000

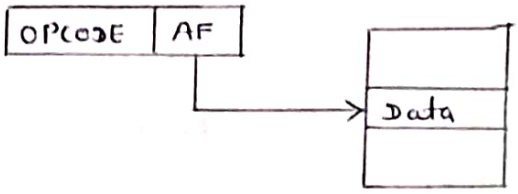
$R_3 \leftarrow 7000$

* Immediate Am Can not used as Designati
* Range of Constant is Limited size of AF

(2) Direct / Absolute Am:-

→ Operand are present in the memory & Instrn Contain the Effective address (EA)

→ 1 memory Reference for RD / WR.



E.g. - MOV R_1 [100]

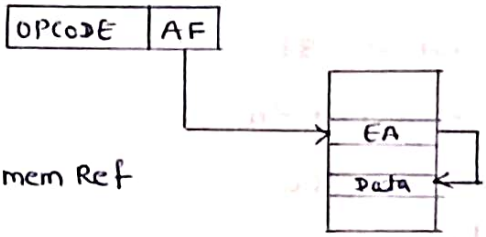
$R_1 \leftarrow M[100]$

ADD R_2 [2000]

$R_2 \leftarrow R_2 + M[2000]$

(3) Memory Indirect Am:-

→ Instruction Contain Address of Effective Address.



→ 2 mem Ref

E.g. - MOV R_0 @6000

$R_0 \leftarrow M[[6000]]$

ADD R_2 ([6000])

$R_2 \leftarrow R_2 + M[[6000]]$

* Flag Extra storage :-

(a) Conditional flag

(b) Control flags

1) Conditional flags :-

- Carry flag (Cy) :-

  Extra bit out of MSB → 1 (True)
  → 0 (False)

  Affected in all Arithmetic opr^n but Not in Increment, Decrement opr^n.

- Parity flag (P) :-
  - Even Parity flags
  - ALU Result Contains Even No. of 1
    - Set(1) (True)
    - Reset(0) (False)
  - during serial data Communication.

- overflow flag(o) :- [01 / 10]
  - Set(1) when the result of a signed operation is too large to fit.

- Zero flag (z) :-
  - Set(1) if ALU Result is zero.
  - Reset(0) false

- Auxiliary Carry flag (AC) :-
  - Used for BCD Numbers.
  - If after any arithmetic operation D_3 generates any Carry → Set(1)

- Sign flag (S) :-
  - if the MSB of ALU Result is 1.

    Set(1) → True

    Reset(0) → False

Exa -

(a) 1 0 0 0 1 1 0 0

1 1 0 0 0 1 0 0

1/0 1 0 1 0 0 0 0

0	0	0	1	0	1	0	1
S	Z		AC		P		CY

$= (15)_H$

→ overflow bit (o) = 1

(b) Control flags :-

- Trap flag :-

  → used for on chip debugging.

  → TF = 1; Single-step execution,
  (-1) (trap) Processor will execute one Instr^n at a time.

  → TF = 0; normal execution occurs
  (-1) (Enable mode)

- Interrupt flag :-

  IF = 1 (Enable Interrupt)

  IF = 0 (disable Interrupt)

- Direction flag :-

  DF = 0 : Processes strings from low to high memory (Auto-Increment mode)

  DF = 1 : Processes strings from high to Low memory (Auto-Decrement mode)

  STD - Set direction flag

  CLD - Clear direction flag

* The collection of different Instruction that the processor can execute is referred to as the processor's Instruction set (ISA).

Q. C = A+B
No. of machine Instruction

Ans. C = A+B

I_1 : Push A : Tos ← M[A]

I_2 : Push B : Tos ← M[B]

I_3 : ADD → Add to Tos

I_4 : Pop C M[C] ← Tos

⇒ 4 Instruction

Note → In stack machine, ALU operation (ADD, MUL etc) are zero Address Instrn.

But Data Transfer Instrn (Push & Pop) are not zero Address Instrn.

Q. X = (A+B) * (C+D)

Ans I_1 : Push A → 8 machine Instrn.

I_2 : Push B

I_3 : ADD

I_4 : Push C

I_5 : Push D

I_6 : ADD

I_7 : MUL

I_8 : Pop X

Consider a 32 bit Hypothetical processor 1 word opcode and 24 bit address

⇒ Opcode = 1 word

AF = 24 bit = 3B

1 word = 32 bit = 4B

Program Capacity –

(4+3) + (4+3) + 4 + (4+3) + (4+3) + 4 + 4 + (4+2)

= 47 B

Push, Pop } stack specific Load, Store } memory specific

In, Out } I/O specific MOV } General purpose

Q. C = A+B
Using Accumulator machine.

Ans – I_1 : Load A ; AC ← M[A]

I_2 : ADD B AC ← AC + M[B]

2 m

I_3 : STORE C M[C] ← AC

3 machine Instruction

Note – In AC-CPU one operand is implicit (present in Accumulator) & one operand is explicit (present in memory).

Q. X = (A+B) * (C+D)
Using AC-CPU

Ans – I_1 : Load A

I_2 : ADD B AC ← AC + M[B]

I_3 : STORE Temp[T] M[T] ← AC

I_4 : LOAD C

I_5 : ADD D ; AC ← AC + M[D]

I_6 : MUL T ; AC ← AC * M[T]

I_7 : STORE X ; M[X] ← AC

7 M/C Instructions
1 spills (Store the Intermediate)

Q. C = A+B
Using General Register organization.

Ans. ① Register – memory Reference –

I_1 : Load R_1 A ; R_1 ← M[A]

I_2 : ADD R_1 B ; R_1 ← R_1 + M[B]

I_3 : STORE C R_1 ; M[C] ← R_1

→ 3 M/C Instruction