Section C

1. Digital Electronics → 20 Q
2. Computer Arch → 15 Q
3. Microprocessor → 15 Q + 3 = 18/20 Q

Section – C
Computer Architecture

Trainer :- Sohail Inamdar

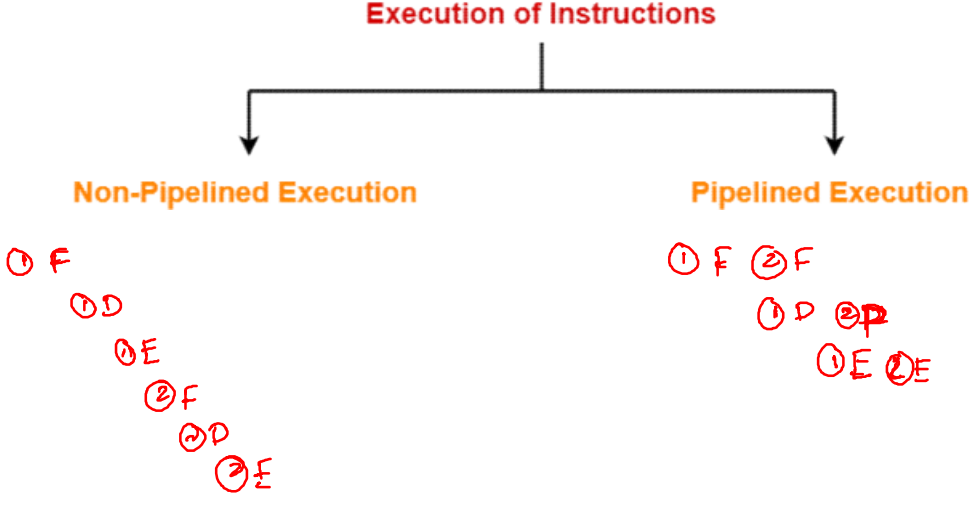
Email :- sohail.inamdar@sunbeaminfo.com

Computer Architecture

fetch, Decode, Execute

- A program consists of several number of instructions.
  - These instructions may be executed in the following two ways

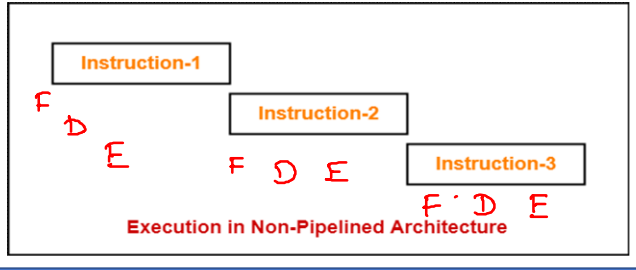
mov A, B



- **Non-Pipelined Execution**

- In non-pipelined architecture, All the instructions of a program are executed sequentially one after the other.
- A new instruction executes only after the previous instruction has executed completely.
- This style of executing the instructions is highly inefficient.
- Example:
- Consider a program consisting of three instructions.
- In a non-pipelined architecture, these instructions execute one after the other as
- If time taken for executing one instruction = t, then-
- Time taken for executing 'n' instructions = n x t

t = 0.5 sec
n = 3
= n x t
= 0.5 x 3 = 1.5 sec



Execution in Non-Pipelined Architecture

→ F, D, E

- No. of cycle required to execute = K x N
- Where K à Number of stages N à Number of Instruction

- For Example :
- If we have 5 number of instruction and 4 number of stages then total how many cycle are required in non-pipelining ?

no. of cycles = K × N
= 4 × 5
= 20

- **Pipelined Execution-**
  - In pipelined architecture, Multiple instructions are executed simultaneously.
  - Pipelining is the process of accumulating instruction from the processor through a pipeline. It allows storing and executing instructions in an orderly process.

- **Four-Stage Pipeline-**
  - In four stage pipelined architecture, the execution of each instruction is completed in following 4 stages-

    - Instruction fetch (IF)
    - Instruction decode (ID)
    - Instruction Execute (IE)
    - Write back (WB)
    - Read back (RB)

Computer Architecture

	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7	Stage 8
Instruction 1	IF	ID	IE	WB				
Instruction 2		IF	ID	IE	WB			
Instruction 3			IF	ID	IE	WB		
Instruction 4				IF	ID	IE	WB	
Instruction 5					IF	ID	IE	WB

No. of cycle required to execute = K +(N -1)

Where K → Number of stages

N → Number of Instruction

For Example :

If we have 5 number of instruction and 4 number of stages then total how many cycle are required in pipelining ?

K + (N - 1)

4 + (5 - 1)

= 8

- Advantages of Pipelining
  - The cycle time of the processor is reduced.
  - It increases the throughput of the system.
  - It makes the system reliable.

- Disadvantages of Pipelining
  - The design of pipelined processor is complex and costly to manufacture.

K = 3, N = 4

Non-pipelining

K x N = 3 x 4

= 12

Pipelining

K + (N-1) = 3 + (4-1)

= 6

Section – C
Microprocessor 8085

Microprocessor 8085

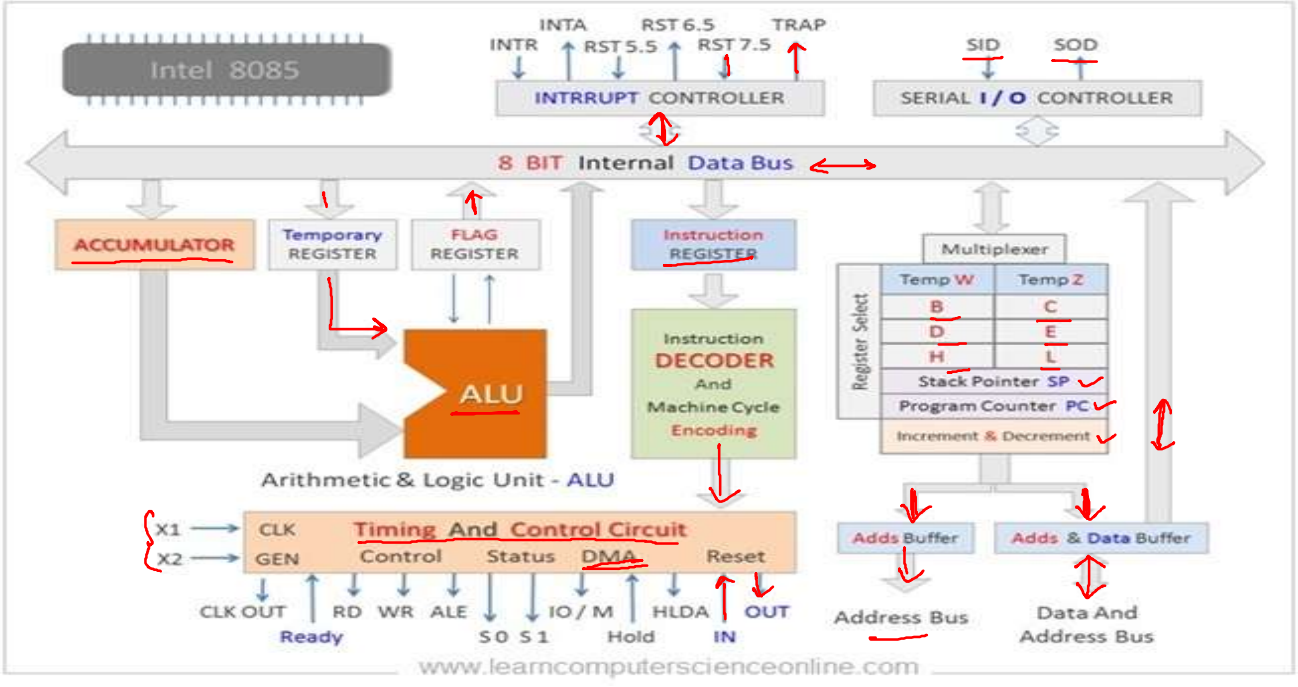
- 8085 architecture generally called as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor which was introduced by Intel in the year 1976 using NMOS technology.
- It has the following configuration −
- 8-bit data bus.
- 16-bit address bus, which can address up to 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3 MHZ single phase clock

BC, DE HL
16bit

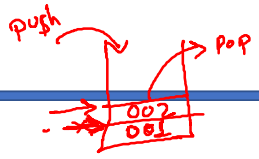
Microprocessor 8085

- **Bus Structure in 8085**
  - There are three buses in Microprocessor: 1. Address Bus 2. Data Bus 3. Control Bus
- **Data Bus**
  - The data bus width is 8-bit i.e. 2^8 = 256 combination of binary digits, identified as D0–D7. Data bus is bidirectional since it carries data in binary form between microprocessor and other external units such as memory. It is used to transmit data. As it is 8-bit wide then largest number is 11111111.
- **Address Bus**
  - 8085 microprocessor contains 16-bit address bus i.e. 2^{16} = 65536 = 64KB memory location it can access, identified A0 – A15. Address bus is unidirectional. The address bus carries addresses from microprocessor to the memory or other devices. The higher order address lines are A8 – A15 and the lower order lines A0 – A7 are multiplexed with the eight bits data lines (D0 – D7)..

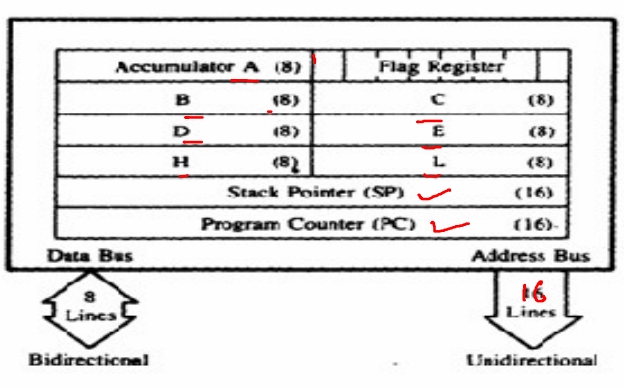
Microprocessor 8085



- **Accumulator**
  - It is an 8-bit register used for general purposes. It also helps in arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU. The data is stored in this register. Arithmetic and logic unit.
  - It is helpful in arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.
- **General purpose register.**
  - There are 6 general purpose registers in 8085 processor, i.e. B, C, D, E, H & L. Each register can hold 8-bit data.
  - These registers can work in pairs in order to hold 16-bit data and their pairing combination looks like B-C, D-E & H-L.
- **Program counter.**
  - It is a type of 16-bit register used to store the address of the instructions that is to be executed. Whenever each instruction get fetched from program counter its store value increases by 1.



- **Stack pointer**
  - It is also a 16-bit register works like stack, which is always incremented/ decremented by 2 during push & pop operations.
- **Temporary register**
  - It is an 8-bit register, as the name suggests it holds the temporary data of arithmetic and logical operations. W and Z are temporary register.



Flag register

- It is an 8-bit register which has five 1-bit flip-flops, which contains either 0 or 1 based on the result data that is stored in the accumulator.

D7	D6	D5	D4	D3	D2	D1	D0
S	Z		AC		P		C

- Sign (S)
- Zero (Z)
- Auxiliary Carry (AC)
- Parity (P)
- Carry (C)

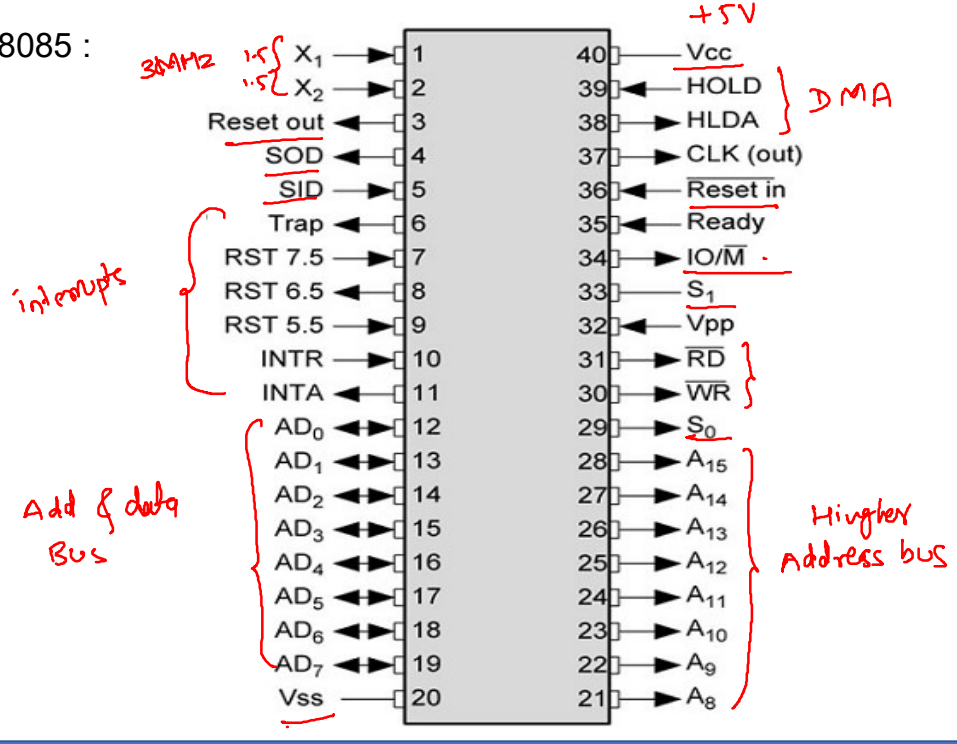
- **Instruction register and decoder**
- The instructions that is fetched from the memory. Instruction decoder decodes the information present in the Instruction register.

- **Timing and control unit**

  - It supplies timing and control signal to the microprocessor to perform certain operations.
  - Following are the timing and control signals, which control external and internal circuits −
  - Control Signals: READY, RD', WR', ALE
  - Status Signals: S0, S1, IO/M'
  - DMA Signals: HOLD, HLDA
  - RESET Signals: RESET IN, RESET OUT

- 8085 is a 40pin IC. Works on +5v power supply with 3MHz frequency. 8085 signals are classified into six groups:
- Address bus
- Data bus
- Control & status signals
- Power supply and frequency signals
- Externally initiated signals § Serial I/O signals

- Pin Diagram of 8085 :



- **Address bus**
- A15-A8, it carries the most significant 8-bits of memory/IO address.
- **Data bus**
- AD7-AD0, it carries the least significant 8-bit address and data bus.
- **Control and status signals**
- These signals are used to recognize the nature of operation. There are 3 control signal and 3 status signals. Three control signals are RD, WR & ALE.
- **RD** − This signal signifies the selected IO or memory device that is to be read and helps in accepting data readily available on the data bus.
- **WR** − This signal indicates that the data on the data bus is to be written into a selected memory or IO location.
- **ALE** − It is Address Latch Enable signal. It is a positive going pulse produced when a new operation is started by the microprocessor. When the pulse goes high, it indicates address. When the pulse goes down it indicates data.

- Three status signals are IO/M, S0 & S1.

- **IO/M**

- This signal is used to distinguish between IO and Memory operations, i.e. when it is high indicates IO operation and when it is low then it indicates memory operation.

- **S1 & S0**

- These signals are used to identify the type of current operation.

- **Power supply**

- There are 2 power supply signals − VCC & VSS. VCC indicates +5v power supply and VSS indicates ground signal.

- There are 3 clock signals, i.e. X1, X2, CLK OUT.

- **X1, X2** − A crystal is connected at these two pins and is used to set frequency of the internal clock generator. This frequency is internally divided by 2.

- **CLK OUT** − This signal is used when the system clock for devices connected with the microprocessor.

- **Interrupts & externally initiated signals**
  - Interrupts are the signals produced by external devices to request the microprocessor to perform a specific task. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR.
- **INTA** − It is an interrupt acknowledgment signal.
- **RESET IN** − This signal is used to reset the microprocessor by setting the program counter to zero.
- **RESETOUT**− This signal is used to reset all the connected devices when the microprocessor is reset.
- **READY** − This signal indicates that the device is ready to send or receive data. If READY is low, then the CPU has to wait for READY to go high.
- **HOLD** − This signal gives hint that another master is requesting the use of the address and data buses.
- **HLDA** (HOLD Acknowledge) − It indicates that the CPU has received the HOLD request and it will relinquish the bus in the next clock cycle. HLDA is set to low after the HOLD signal is removed

- **Serial I/O signals**
- There are 2 serial signals, i.e. SID and SOD and these signals are used for serial communication.
- **SOD** (Serial output data line) – According to the SIM instruction the output SOD is set/reset.
- **SID** (Serial input data line) - Whenever the data instruction is executed the data on this line is loaded into accumulator (AC)

- **Interrupts in 8085**
- Interrupts are the signals that are sent by the external devices to the microprocessor to perform a particular task or work in request format. There are 5 interrupt signals, i.e. TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR.
- Interrupt are classified into following groups based on their parameter −
- **Vector interrupt** − In this type of interrupt, the interrupt address is known to the processor.
- For example: RST7.5, RST6.5, RST5.5, TRAP.
- **Non-Vector interrupt** − In this type of interrupt, the interrupt address is not known to the processor so, the interrupt address require to send externally by the device to perform interrupts.
- For example: INTR.
- **Maskable** interrupt − In this type of interrupt, you can neglect the interrupt by giving some instructions to the program.
- For example:RST7.5, RST6.5, RST5.5

- **Non-Maskable interrupt** − In this type of interrupt, interrupts cannot be disabled by writing some instructions into the program.
  - For example: TRAP.
- **Software interrupt**− In this type of interrupt, as it is software the programmer has to give the instructions to the program in order to execute the interrupts. There are 8 software interrupts in 8085, i.e. RST0, RST1, RST2, RST3, RST4, RST5, RST6, and RST7.
- Hardware interrupt − There are 5 interrupt pins in 8085 used as hardware interrupts, i.e. TRAP, RST7.5, RST6.5, RST5.5, INTA.
  - Note − TRAP has the highest priority, then RST7.5.

- **TRAP**

- It is called as a non-maskable interrupt, which has the highest priority among all interrupts. It is subroutine calls which are forced by the microprocessor when it identifies any interruption in the instructions. So, when the power failure occurs, it executes as ISR and sends the data to backup memory. This interrupt transfers the control to the location 0024H.

- **RST7.5**

- It is called as maskable interrupt, which has the second highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 003CH address.

- **RST 6.5**

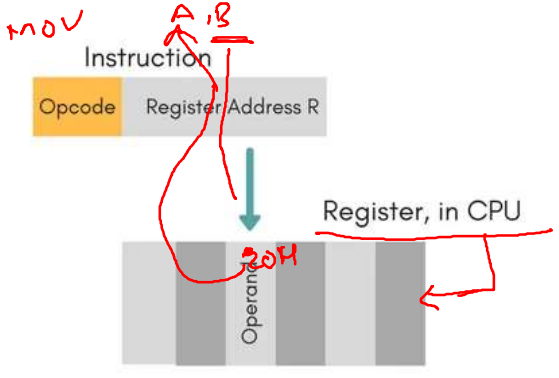
- It is a maskable interrupt, having the third highest priority among all interrupts. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 0034H address

- **RST 5.5.**
- It is a maskable interrupt. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.
- **INTR**
- It is a maskable interrupt, having the lowest priority among all interrupts. It can be deleted by resetting the microprocessor.

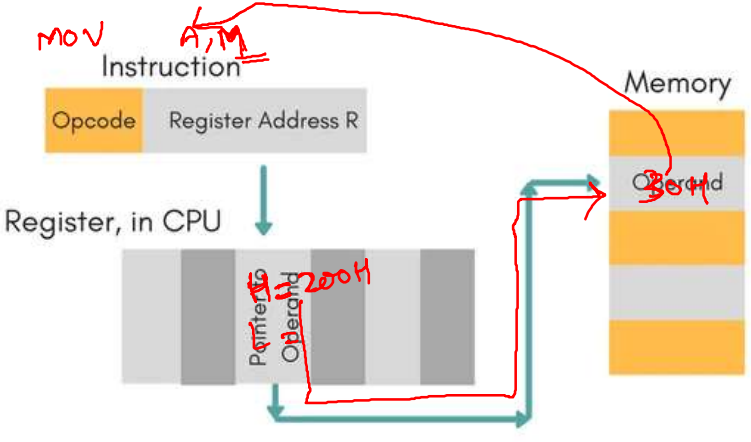
- **Addressing Modes and Instruction Cycle**
- The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction.
- **Types of Addressing Modes**
- **Immediate Mode**
- In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.
- For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here. MVI B, 20H: means 20H is copied into register B.

Register Mode

- In this mode, the data is transferred or copied from one register to another.
- In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.
- For example: MOV A, B: means data in register B is copied to register A



- **Register Indirect Mode**
- In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.
- For example: MOV A,M : the memory data addressed by HL pair is moved to A register.



DEC , B

- **Auto Increment/Decrement Mode**
- In this the register is incremented or decremented after or before its value is used.
- **Direct Addressing Mode**
- In this mode, effective address of operand is present in instruction itself.
- Single memory reference to access data.
- No additional calculations to find the effective address of the operand.
- For Example:
- LDA 5000K: the data at address 5000K is copied to register B..

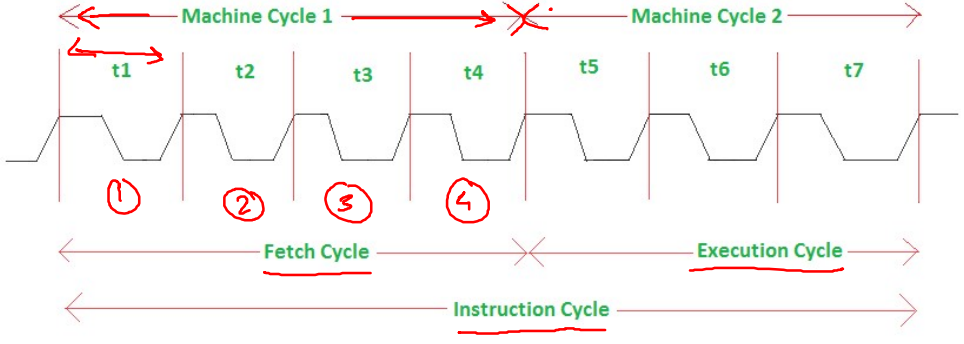
LDA , 5000K

- **Implied addressing mode**
- In this mode need not require any operand, the data is specified by the op code itself.
- For example: CMP, RAL, RAR.

RAL

- 8085 instruction can be classified as
- **Data Transfer Instruction**
- This group of instructions copy data from a location called a source to another location called a destination, without modifying the contents of the source. The term data transfer is used for copying function
- **Arithmetic Instruction**
- These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.
- **Logical Instruction**
- These instructions perform various logical operations with the contents of the accumulator.
- **Branching Instruction**
- This group of instructions alters the sequence of program execution either conditionally or unconditionally.
- **Machine control Instruction**
- These instructions control machine functions such as Halt, Interrupt, or do nothing.

- **Instruction Cycle:** The time taken by the processor to complete the execution of an instruction.
- **Machine Cycle:** The time required to complete one operation, accessing either the memory or I/O device. A machine cycle consists of three to six T-states.
- **T-State:** Time corresponding to one clock period. It is the basic unit to calculate execution of instructions or programs in a processor



Instruction cycle in 8085 microprocessor

To execute a program, 8085 performs various operations as:

- Opcode fetch (4T)
- Memory read (3T)
- Memory write (3T)
- I/O read (3T)
- I/O write (3T)
- Loop (7 or 10T)

Programs on 8085

- MVI D, 8BH
- MVI C,6FH
- INR C
- MOV A,C
- ADD D
- CMA
- OUT PORT1
- HLT
  STOP

D = 8BH

C = 6FH

INR C → increment C + 1H

```
  0110 1111
+ 0000 0001
-----------
  0111 0000
  C=7  0H
```

A = 70H

```
A = 70H =  0111 0000
D = 8BH = +1000 1011
-------------------
          1111 1011
            F    B
```

A ←

A = FBH

CMA → complement

1's comp

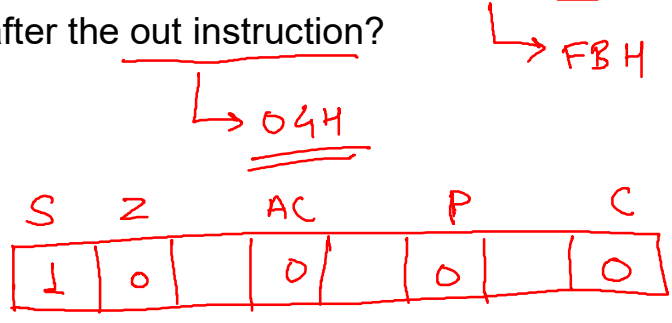
A = FBH = 1111 1011

1's comp

```
0000 0100
   0    4
```

A = 04

1) From the above program what is output after Add operation ?and status of all flags?
2) What will the result after the out instruction?



JC – jump if carry is generated

```
MVI A,8FH
ADI 72H
JC DISPLAY
OUT PORT1
HLT

DISPLAY: XRA A
OUT PORT1
HLT
```

A = 8FH

ADI 72H 0111 0010

A 8FH 1000 1111

1) 0000 0001

A = 01H

CY = 1

XRA → XOR with A.

A ⊕ A

0000 0001

⊕ 0000 0001

0000 0000

A = 00H

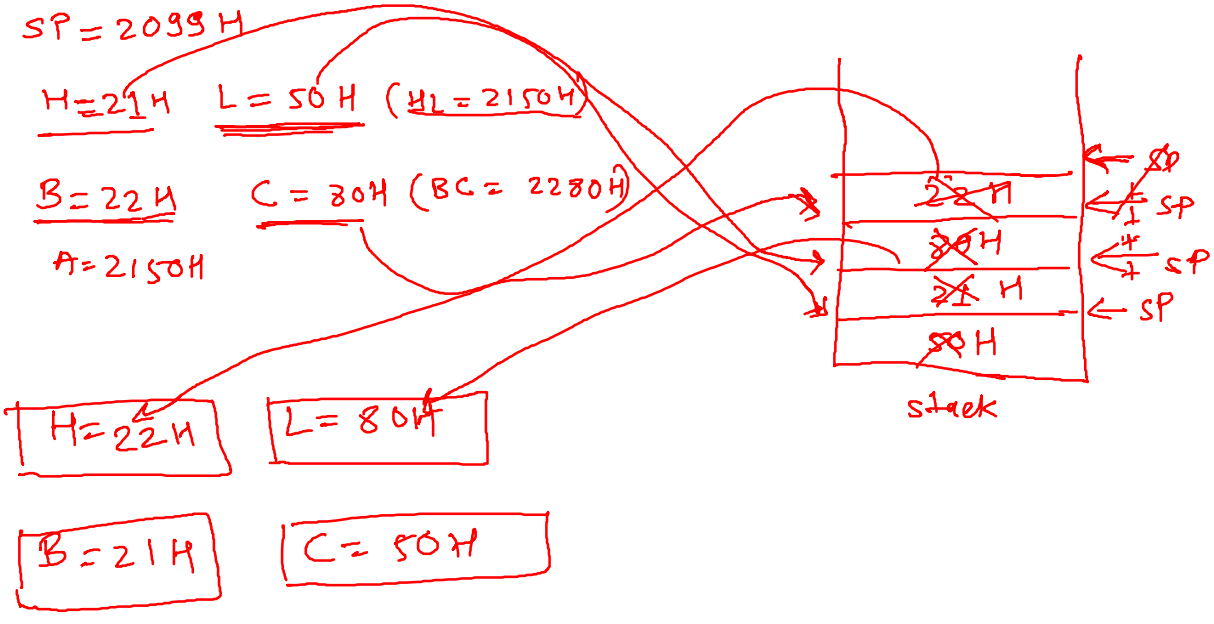
1) What is output after Add operation ? what is status of the carry flag?

2) Jump instruction in the program is conditional or unconditional?

3) What is the value displayed on the screen?

LXI → load immediate

- LXI SP, 2099H
- LXI H, 2150H
- LXI B, 2280H
- MOV A,M → HL
- PUSH H
- PUSH B
- POP H
- POP B
- HLT



- 8-bit data is stored at memory location 2050H (8 bit data =73H)

LDA 2050H → 2050H (Address) → 73H (Data)

A = 73H

MOV B,A

B = 73H

STC

STC → Set carry

CY = 1

CMC

CMC → compliment carry

CY = 0

RAR = Rotate A right / A = 73H

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	CY
A=73=	0	1	1	1	0	0	1	1	0
	0	0	1	1	1	0	0	1	1

A = 39H

XRA B

A = 39 = 0011 1001

B = 73 = ⊕ 0111 0011

0100 1010

A = 4AH

STA → Store Acc 3050

3050	4AH

```
MVI A,50H
MVI B,20H
CMP B
MOV A,B
HLT
```

→ A = 50H

B = 20H

CMP B

A > B cy=0 , z=0

A = B z=1 cy=0

A < B cy=1 , z=0

A = 20H

H.W

- At 3000H location 05H data is stored

LDA 3000H

CMA

STA 3001H

ADI 01H

STA 3002H

HLT

- **Data Transfer instruction**
- **MVI** - Move immediate
- MVI Rd, data -Transfers 'data' value into register Rd.
- Ex: MVI A,50H
- **MOV** – Move from one register to another register
- MOV Rd, Rs -Here Rd is destination register, Rs is source register. Transfers data in Rs into Rd. Rs content will not be changed.
- Ex: MOV B,A
- **IN port-address** – This instruction reads data from IO device connected at specified port address and loads accumulator with that data.
- Ex: IN 10H
- OUT port-address – This instruction sends content of accumulator to IO device connected at specified port address.
- Ex: OUT 05H

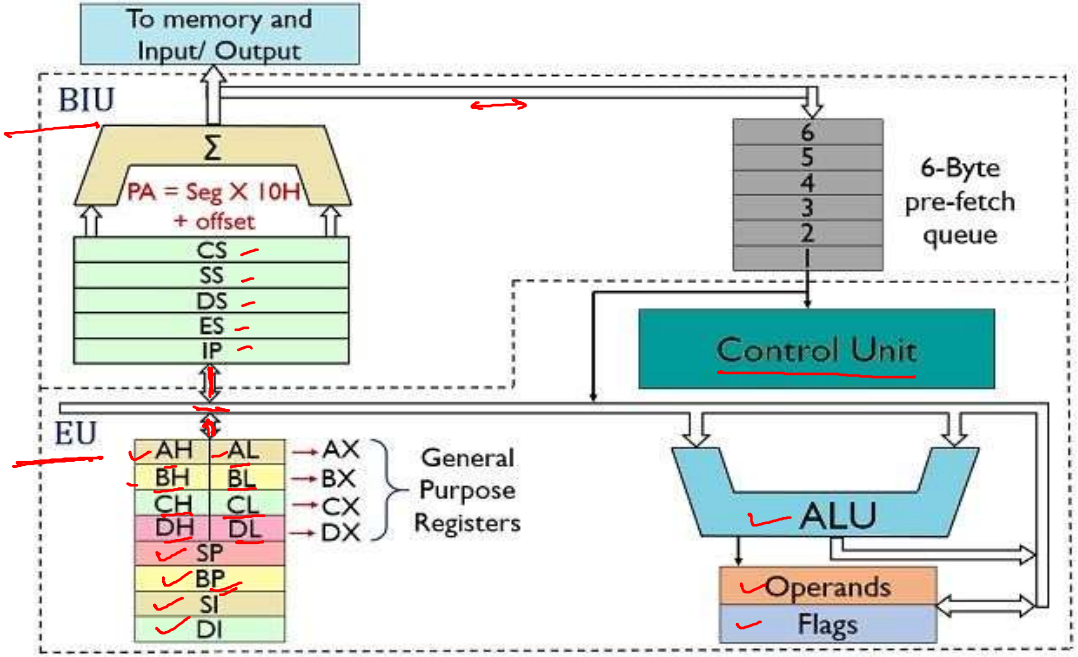
- **LXI**- Load Immediate data into register pair
- LXI Rp, data -This instruction loads 16-bit immediate data into specified register pair(Rp) Example: LXI H,2050H
- **LDAX**-Load accumulator indirect
- LDAX B/D - This instruction loads Accumulator with memory location pointed by content of B or D Example : LDAX B,2030H
- **LDA**-Load accumulator direct
- LDA 16-bit address - This instruction loads accumulator with the content of memory location specified in the instruction.
- Ex: LDA 2005H
- **STAX**-Store accumulator indirect
- STAX B/D -This instruction stores the content of accumulator into memory location pointed by B or D register pair.
- Example: STAX B

- Arithmetic Instructions
- **ADD**-Add Register to Accumulator
- This instruction performs addition of content of register or memory location pointed by register pair HL and stores the result in accumulator.
- Example: ADD B
- **ADI**- Add immediate data
- ADI data - This instruction adds 8-bit “data” to accumulator and places the result in accumulator.
- Example: ADI 50H.
- **SUB**- subtract register from accumulator
- This instruction subtracts content of register R or content of memory location pointed by HL register pair from accumulator and stores the result in accumulator.
- Example: SUB B
- **SUI**- subtract immediate data
- SUI data - This instruction subtracts 8-bit “data” from accumulator and stores the result in accumulator. Example: SUI 34H

8085 → Data bus → 8 bit
8086 → 16 bit

Address
8085 – 16 = 2^{16} = 64 KB
8086 → 20 – 2^{20} = 1 MB

- **Architecture 8086**
- The Architecture of 8086 supports a 16-bit ALU, a set of 16-bit register and provides segmented memory addressing capability, a rich instruction set, powerful interrupt structure, fetched instruction queue for overlapped fetching and execution etc.
- The complete architecture of 8086 can be divided into two parts
- **Bus Interface Unit (BIU)**
- BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.
- **Execution Unit (EU)**
- EU contains Control circuitry, Instruction decoder, ALU, Registers, Flag register



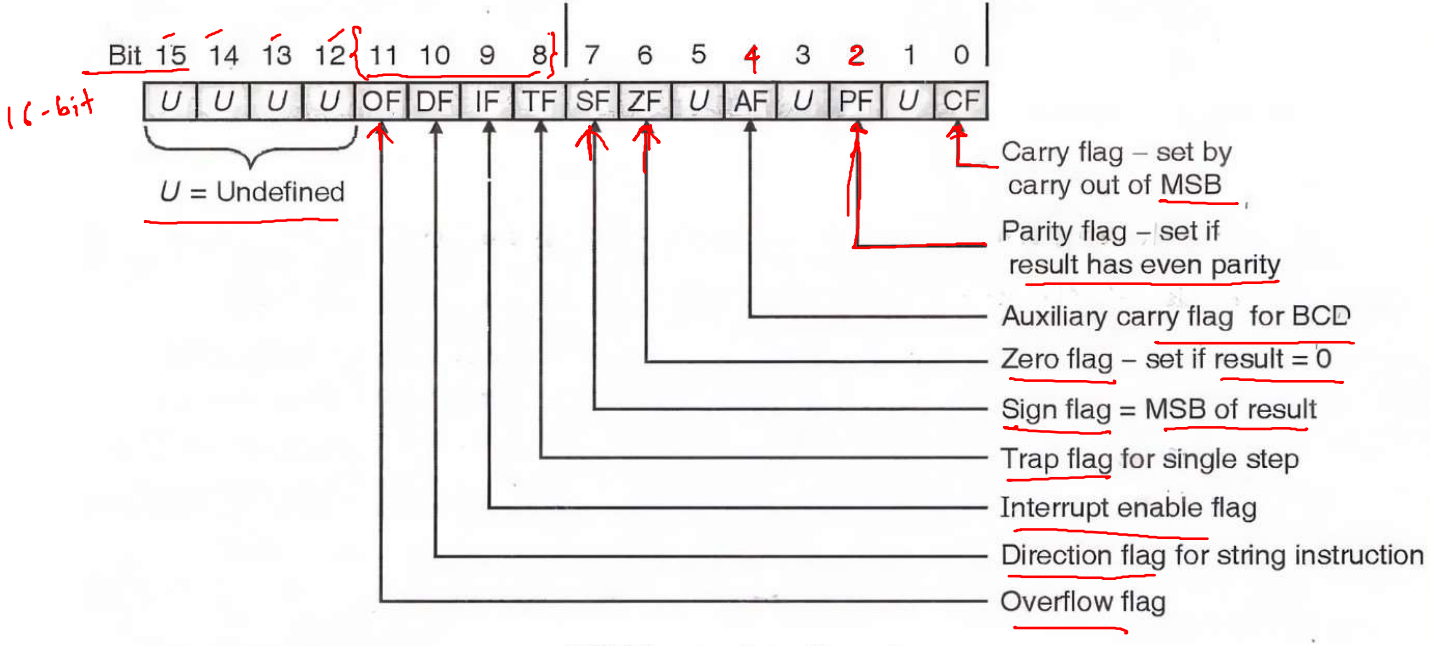
Block Diagram of 8086 Microprocessor

Electronics Desk

- **EU (Execution Unit)**
- Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU.
- **ALU**
- It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.
- Instruction pointer
- It is a 16-bit register used to hold the address of the next instruction to be executed.
- **Base Pointer**
- BP can hold offset address of any location in the stack segment. It is used to access random locations of the stack.
- **Source Index**
- It holds offset address in Data Segment during string operations.
- **Destination Index**
- It holds offset address in Extra Segment during string operations

- **General purpose register**
  - There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16 bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.
- **AX register** − It is also known as accumulator register. It is used to store operands for arithmetic operations.
- **BX register** − It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** − It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** − This register is. used to hold I/O port address for I/O instruction.
- **Stack pointer register**
  - It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

- **Flag Register**
  - It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups − Conditional Flags and Control Flags.
- Conditional Flags
  - It represents the result of the last arithmetic or logical instruction executed. List of conditional flags are
- Carry flag − This flag indicates an overflow condition for arithmetic operations.
- Auxiliary flag − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
- Parity flag − This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- Zero flag − This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- Sign flag − This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- Overflow flag − This flag represents the result when the system capacity is exceeded

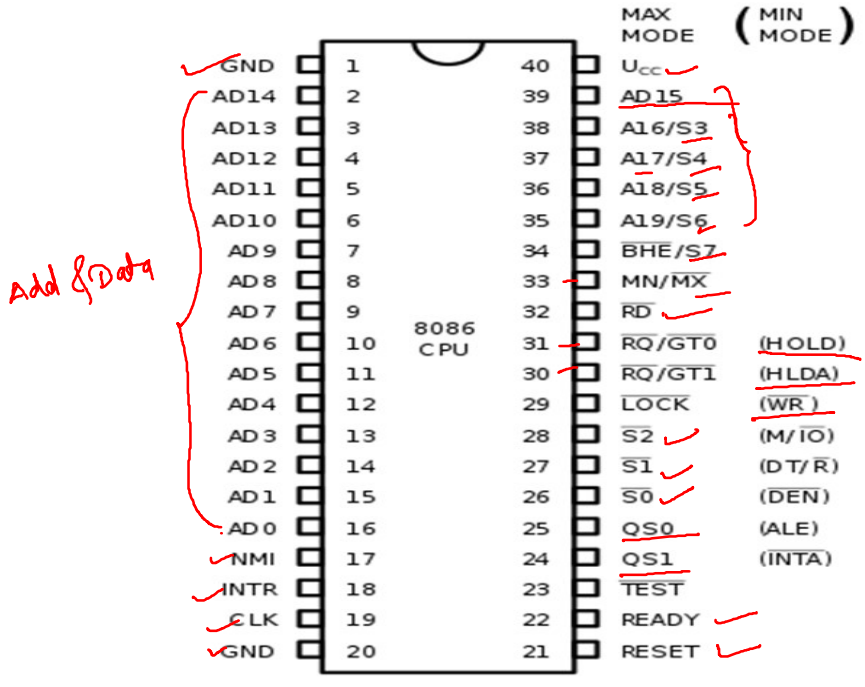


8086 flag register format

- **Control Flags**
- Control flags controls the operations of the execution unit.
- List of control flags are
- **Trap flag** − It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** − It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa

- **BIU (Bus Interface Unit)**
- BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.
- **Instruction queue.**
- BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.
- Fetching the next instruction while the current instruction executes is called pipelining.

- **Segment register**
- IU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations.
- **Code Segment register:**
- CS is 16-bit register holds the base address for the Code Segment. All programs are stored in the Code Segment and accessed via the IP.
- **Data Segment register:**
- DS is 16-bit register holds the base address for the Data Segment.
- **Stack Segment register:**
- SS is 16-bit register holds the base address for the Stack Segment.
- **Extra Segment register:**
- ES is 16-bit register holds the base address for the Extra Segment


MAX MODE
(MIN MODE)
GND 1
AD14 2
AD13 3
AD12 4
AD11 5
AD10 6
AD9 7
AD8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
GND 20
8086 CPU
40 Vcc
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 BHE/S7
33 MN/MX
32 RD
31 RQ/GT0 (HOLD)
30 RQ/GT1 (HLDA)
29 LOCK (WR)
28 S2 (M/IO)
27 S1 (DT/R)
26 S0 (DEN)
25 QS0 (ALE)
24 QS1 (INTA)
23 TEST
22 READY
21 RESET
Add & Data

- **Addressing modes**
- **Immediate addressing Mode** - the data is provided in the instruction directly.
- MOV CX,4374H Direct Addressing Modes – the instruction operand specifies the memory address where data is located.
- MOV BL, [4374H]
- **Register Addressing Mode** – copy the data from one to another register.
- MOV CX, AX
- **Register Indirect Addressing Modes** - Instruction specifies a register containing an address, where data is located. This addressing mode work with SI, DI, BX and BP register.
- MOV AL, [BX] or MOV AL, CS: [BX]
- **Register Relative Addressing Modes-** Effective address is formed by adding an 8-bit or 16-bit displacement with contents of any one of register BX, BP, SI and DI
- MOV AX,50H[BX][SI]

- Instruction Set of 8086
- Data transfer instruction
- Arithmetic and Logical instruction
- Branching Instruction
- Loop Instruction
- Machine control Instruction
- Flag manipulation Instruction
- Shift & rotate Instruction
- String Instruction