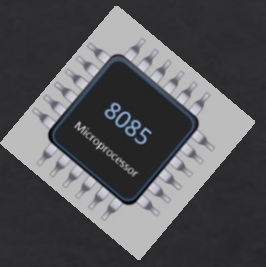


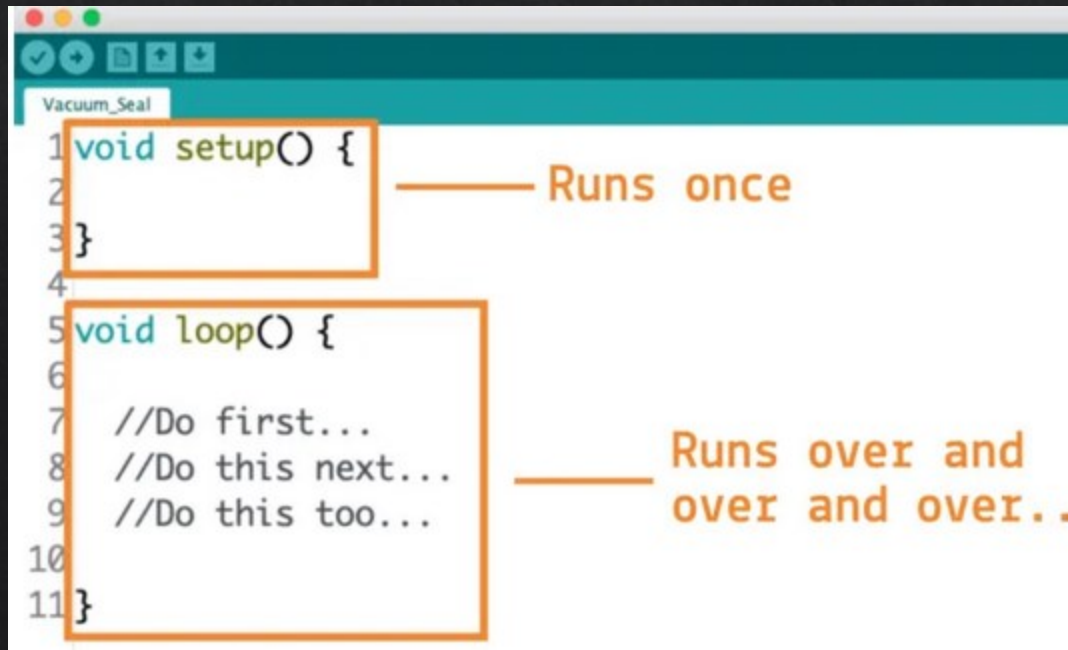
ETT210 MICROPROCESSOR AND MICROCONTROLLER PROGRAMMING

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING (EEE)

- PREPARED BY:

  Mr. Jean Claude TUYISENGE
- B.SC.IT, Msc.IT, Msc.IoT-ECS
- Tel:0782994921
- Email:claudenesta09@gmail.com

Lecturer three: Addressing modes and Instructions Se Of 8085 Microprocessor



Programming of 8085 and Its Interfacing

TOPICS

1. Introduction
2. Programming model of 8085
3. Addressing modes of 8085
4. Instruction set of 8085
5. Example Programs
6. Instruction & Data Formats of 8085

1. Introduction

- A microprocessor executes instructions given by the user
- Instructions should be in a language known to the microprocessor
- Microprocessor understands the language of 0's and 1's only

- This language is called **Machine Language**

- For e.g.

  01001111

- Is a valid machine language instruction of 8085
- It copies the contents of one of the internal registers of 8085 to another

A machine language program to add two numbers

```
00111110
  A
00000010
00000110
  B
00000100
10000000
```

;Copy value 2H in register

;Copy value 4H in register

;A = A + B

ASSEMBLY LANGUAGE OF 8085

- It uses English like words to convey the action/meaning called as MNEMONICS
- For e.g.

MOV	to indicate data transfer
ADD	to add two values
SUB	to subtract two values

Assembly language program to add two numbers

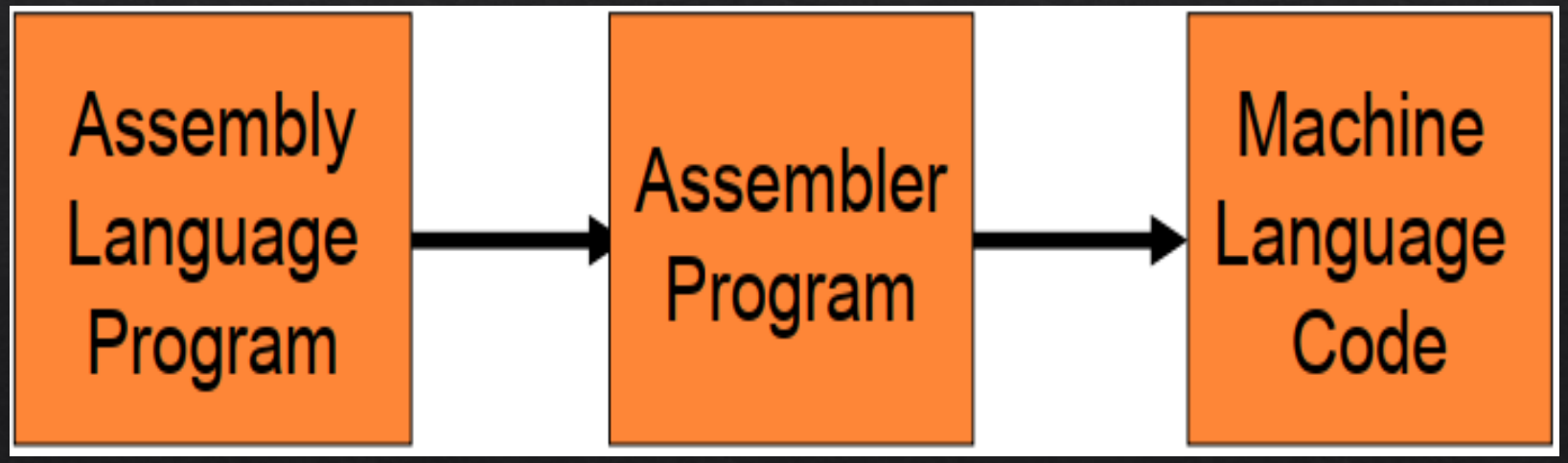
```
MVI A, 2H ;Copy value 2H in register A
MVI B, 4H ;Copy value 4H in register B
ADD B     ;A = A + B
```

Note:

- Assembly language is specific to a given processor
- For e.g. assembly language of 8085 is different than that of Motorola 6800 microprocessor

MICROPROCESSOR UNDERSTANDS MACHINE LANGUAGE ONLY!

- Microprocessor cannot understand a program written in Assembly language
- A program known as **Assembler** is used to convert a Assembly language program to machine language



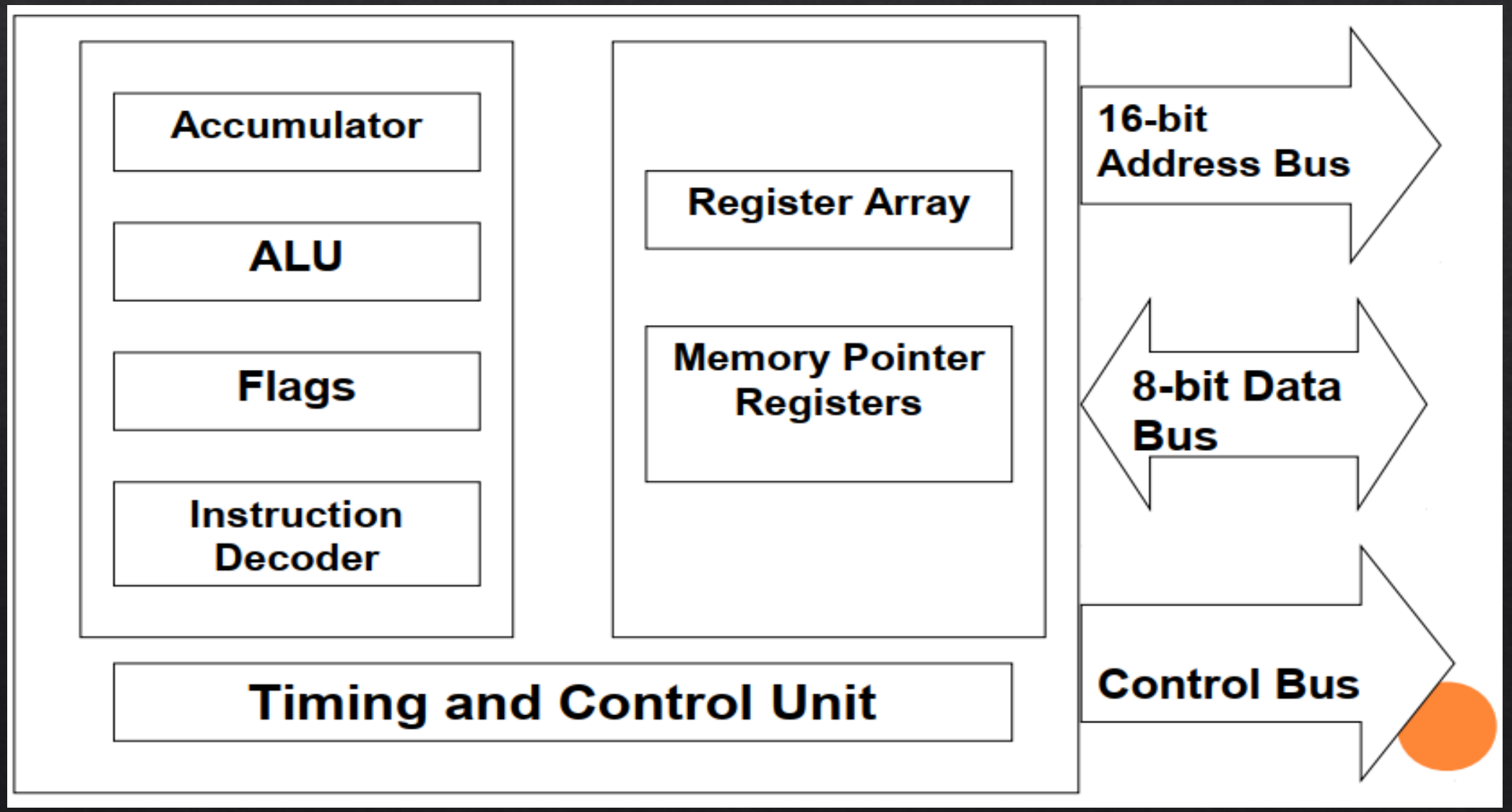
LOW-LEVEL/HIGH-LEVEL LANGUAGES

- Machine language and Assembly language are both
  - Microprocessor specific (Machine dependent)

    so they are called
  - Low-level languages

- **Machine independent** languages are called
  - High-level languages
  - For e.g. BASIC, PASCAL,C++,C,JAVA, etc.
  - A software called **Compiler** is required to convert a high-level language program to machine code

2. PROGRAMMING MODEL OF 8085



Accumulator (8-bit)

Flag Register (8-bit)

S Z AC P CY

B (8-bit)

C (8-bit)

D (8-bit)

E (8-bit)

H (8-bit)

L (8-bit)

Stack Pointer (SP) (16-bit)

Program Counter (PC) (16-bit)

8- Lines

Bidirectional

16- Lines

Unidirectional

OVERVIEW: 8085 PROGRAMMING MODEL

1. Six general-purpose Registers
2. Accumulator Register
3. Flag Register
4. Program Counter Register
5. Stack Pointer Register

1. Six general-purpose registers

- **B, C, D, E, H, L**
- Can be combined as register pairs to perform 16-bit operations (**BC, DE, HL**)

2. Accumulator – identified by name A

- This register is a part of ALU
- 8-bit data storage
- Performs arithmetic and logical operations
- Result of an operation is stored in accumulator

3. Flag Register

- This is also a part of ALU
- 8085 has five flags named
  - ✓ **Zero** flag (Z)
  - ✓ **Carry** flag (CY)
  - ✓ **Sign** flag (S)
  - ✓ **Parity** flag (P)
  - ✓ **Auxiliary Carry** flag (AC)

- These flags are five flip-flops in flag register
- Execution of an arithmetic/logic operation can **set** or **reset** these flags
- Condition of flags (set or reset) can be tested through software instructions
- 8085 uses these flags in decision-making process

4. Program Counter (PC)

- A 16-bit memory pointer register
- Used to sequence execution of program instructions
- Stores address of a memory location
  - where next instruction byte is to be fetched by the 8085

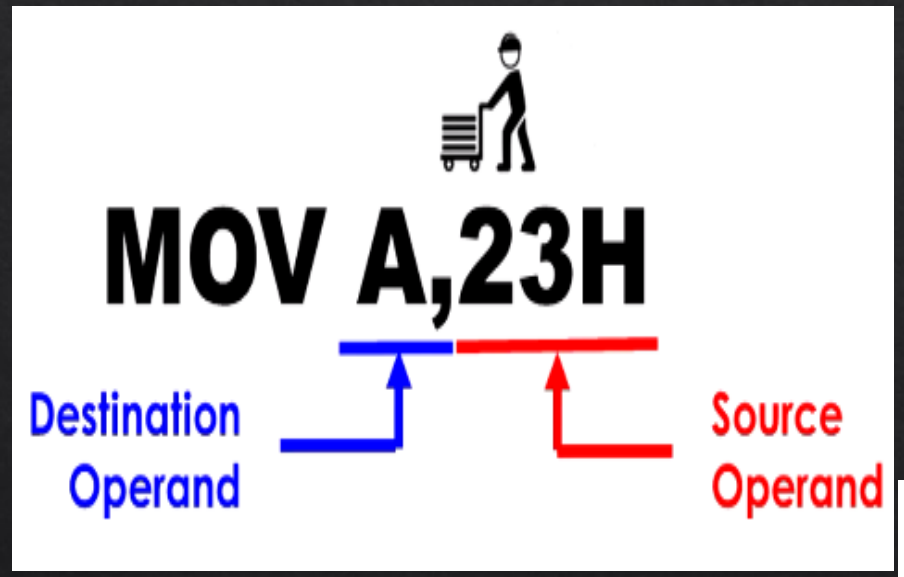
- when 8085 gets busy to fetch current instruction from memory
  - PC is incremented by one
  - PC is now pointing to the address of next instruction

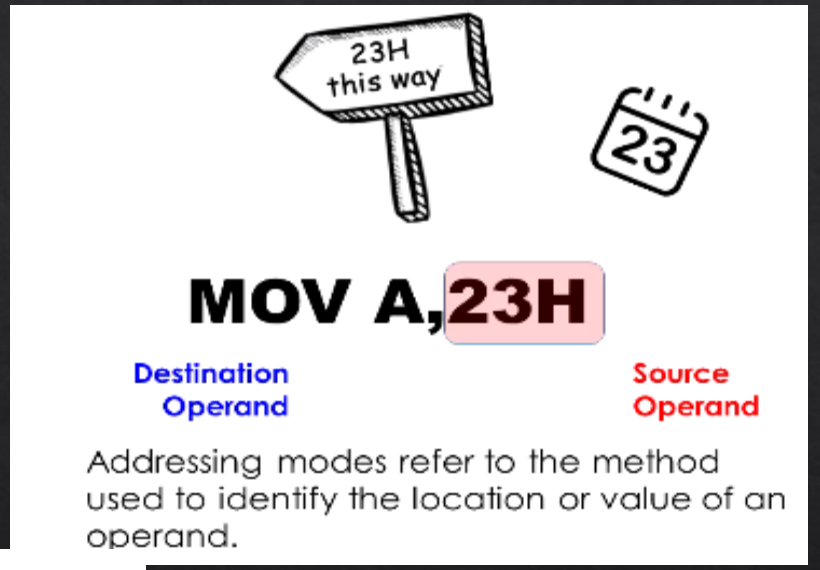
5. Stack Pointer Register

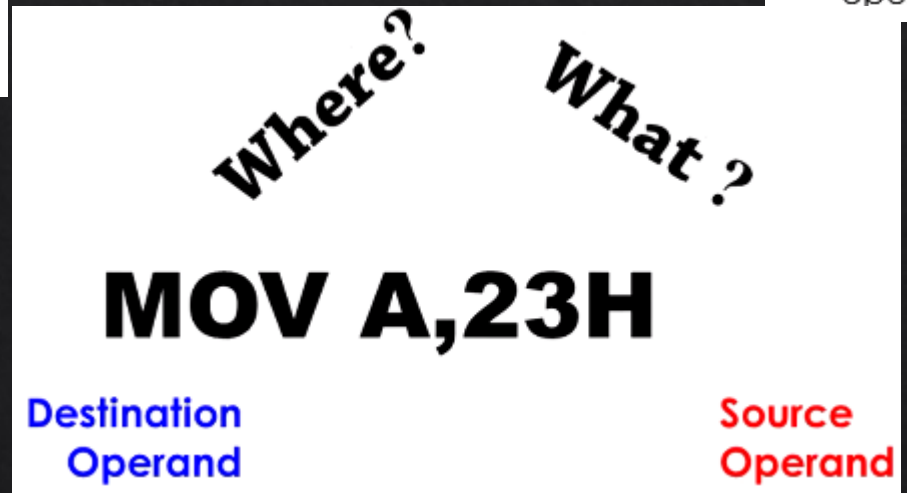
- a 16-bit memory pointer register
- Points to a location in **Stack** memory
- Beginning of the stack is defined by loading a 16-bit address in stack pointer register

4. Addressing mode of 8085

- Addressing modes refer to the method used to identify the location or value of a operand







- For 8085, they are the following addressing modes:

1. Immediate addressing.
2. Direct addressing.
3. Register addressing.
4. Indirect addressing.

1. Immediate addressing.

- Data is present in the instruction. Load the immediate data to the destination provided.
- ***Mnemonic:*** ***MVI*** R, #data
- The operand is the actual value to be used in the operation

- Examples:

  MOV A, #23H

  MOV R3,#0F4H

Mnemonics: These are the symbolic codes for either instructions or commands to perform a particular function. Eg MOV, ADD, SUB etc..

2. Direct addressing.

- Data is provided through the registers.
- ***Mnemonic:*** MOV Rd, Rs
- The operand is the address of the internal data memory

- Examples:

MOV A, 23H
MOV R1, 24H

MOV 26H, 33H
MOV 26H,#33H

3. Register addressing.

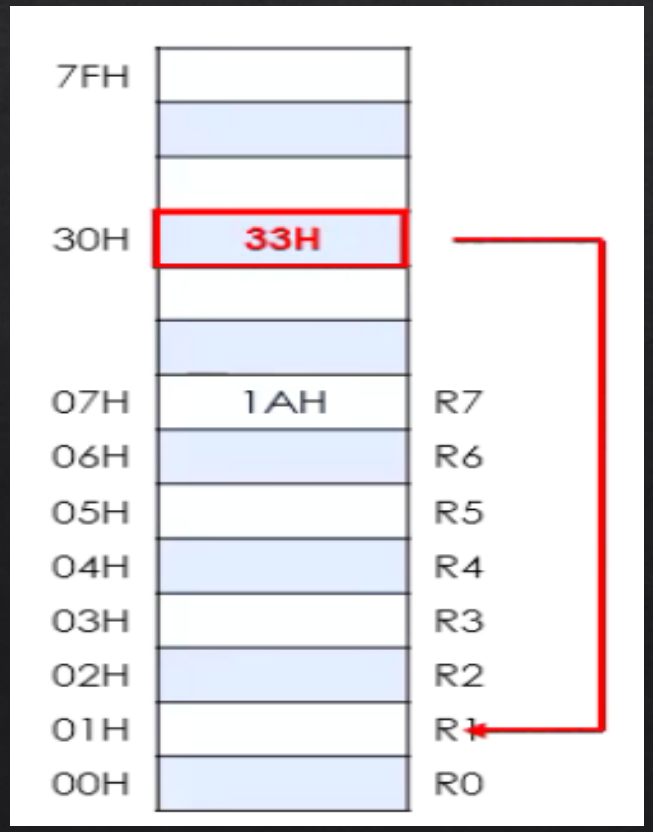
- Used to accept data from outside devices to store in the accumulator or send the data stored in the accumulator to the outside device.

- Accept the data from the port 00H and store them into the accumulator or Send the data from the accumulator to the port 01H
- The operand is a register

- Example:

MOV 26H,R7

MOV R1, 30H



4. Indirect addressing

- Register indirect addressing means that the address which will be used by the instruction is taken from the content of a register.

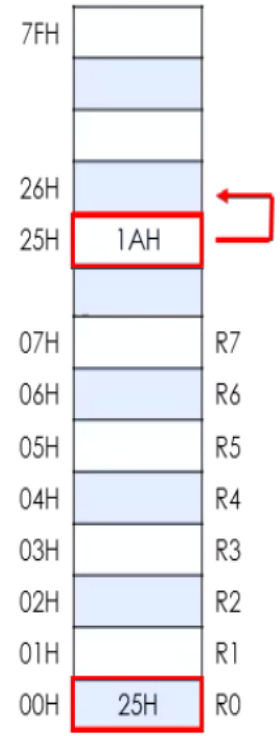
- Example:

MOV 26H,@R0

MOV @R1, 30H

MOV 26H,@R0

Mnemonic: MOV direct,@Ri



4. INSTRUCTION SET OF 8085

- An *instruction* is a command given to the microcomputer to perform a specific task or function on a given data. The entire group of instructions, called the *instruction set*, determines what functions the microprocessor can perform.

- **Consists of**
  - ✓ 74 operation codes, e.g. MOV
  - ✓ 246 Instructions, e.g. MOV A,B

- *Each instruction has two parts:*

- The first part is the task or operation to be performed. This part is called the "*opcode*" (operation code).

- The second part is the data to be operated on, and is Called the "*operand*".

- **8085 instructions can be classified as:**

1. Data Transfer (Copy)
2. Arithmetic
3. Logical and Bit manipulation
4. Branch
5. Machine Control

1. Data Transfer Instructions(copy)

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

Data Transfer Instructions

Opcode	Operand	Description
MOV	Rd, Rs M, Rs Rd, M	Copy from source to destination.

- This instruction copies the contents of the source register into the destination register.
- The contents of the source register are not altered.
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.
- **Example:** MOV B, C or MOV B, M

Data Transfer Instructions

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

- The 8-bit data is stored in the destination register or memory.
- If the operand is a memory location, its location is specified by the contents of the H-L registers.
- **Example:** MVI B, 57H or MVI M, 57H

Data Transfer Instructions

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
- **Example:** LDA 2034H

Data Transfer Instructions

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.
- The contents of either the register pair or the memory location are not altered.
- **Example:** LDAX B

Data Transfer Instructions

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

- This instruction loads 16-bit data in the register pair.
- **Example:** LXI H, 2034 H

Data Transfer Instructions

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.
- **Example:** LHLD 2040 H

Data Transfer Instructions

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of accumulator are copied into the memory location specified by the operand.
- **Example:** STA 2500 H

Data Transfer Instructions

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

- The contents of accumulator are copied into the memory location specified by the contents of the register pair.
- **Example:** STAX B

Data Transfer Instructions

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.
- **Example:** SHLD 2550 H

Data Transfer Instructions

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged with the contents of register E.
- **Example:** XCHG

Data Transfer Instructions

Opcode	Operand	Description
SPHL	None	Copy H-L pair to the Stack Pointer (SP)

- This instruction loads the contents of H-L pair into SP.
- **Example:** SPHL

Data Transfer Instructions

Opcode	Operand	Description
XTHL	None	Exchange H–L with top of stack

- The contents of L register are exchanged with the location pointed out by the contents of the SP.
- The contents of H register are exchanged with the next location (SP + 1).
- **Example:** XTHL

Data Transfer Instructions

Opcode	Operand	Description
PUSH	Reg. pair	Push register pair onto stack

- The contents of register pair are copied onto stack.
- SP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack.
- SP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack.
- **Example:** PUSH B

Data Transfer Instructions

Opcode	Operand	Description
POP	Reg. pair	Pop stack to register pair

- The contents of top of stack are copied into register pair.
- The contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags).
- SP is incremented and the contents of location are copied to the high-order register (B, D, H, A).
- **Example:** POP H

Data Transfer Instructions

Opcode	Operand	Description
OUT	8-bit port address	Copy data from accumulator to a port with 8-bit address

- The contents of accumulator are copied into the I/O port.
- **Example:** OUT 78 H

Data Transfer Instructions

Opcode	Operand	Description
IN	8-bit port address	Copy data to accumulator from a port with 8-bit address

- The contents of I/O port are copied into accumulator.
- **Example:** IN 8C H

Summary – Data transfer

- MOV Move
- MVI Move Immediate
- LDA Load Accumulator Directly from Memory
- STA Store Accumulator Directly in Memory
- LHLD Load H & L Registers Directly from Memory
- SHLD Store H & L Registers Directly in Memory

Summary Data transfer

- An 'X' in the name of a data transfer instruction implies that it deals with a register pair (16-bits);

- LXI Load Register Pair with Immediate data
- LDAX Load Accumulator from Address in Register Pair
- STAX Store Accumulator in Address in Register Pair
- XCHG Exchange H & L with D & E
- XTHL Exchange Top of Stack with H & L

Summary - Stack

- PUSH Push Two bytes of Data onto the Stack
- POP Pop Two Bytes of Data off the Stack
- XTHL Exchange Top of Stack with H & L
- SPHL Move content of H & L to Stack Pointer

I/O instructions

- IN Initiate Input Operation
- OUT Initiate Output Operation

EXAMPLES (COPY) OPERATIONS/INSTRUCTIONS

1. **Load** a 8-bit number 4F in register B	MVI B, 4FH
2. **Copy** from Register B to Register A	MOV A,B
3. **Load** a 16-bit number 2050 in Register pair HL	LXI H, 2050H
4. **Copy** from Register B to Memory Address 2050	MOV M,B
5. **Copy** between Input/Output Port and Accumulator	OUT 01H IN 07H

Programme

- Load memory location F300H with data 29H. Transfer the data in to register B.

- MVI A, 29H
- STA F300H
- MOV B, A
- HLT

2. Arithmetic instructions

- 1. **Addition** of two 8-bit numbers
  2. **Subtraction** of two 8-bit numbers
  3. **Increment/ Decrement** a 8-bit number

- For more details about Arithmetic instructions follow this link:

https://www.tutorialspoint.com/microprocessor/microprocessor_8085_arithmetic_instructions.htm

EXAMPLE ARITHMETIC OPERATIONS /INSTRUCTIONS

1. **Add** a 8-bit number 32H to Accumulator	ADI **32H**
2. **Add** contents of Register B to Accumulator	ADD **B**
3. **Subtract** a 8-bit number 32H from Accumulator	SUI **32H**
4. **Subtract** contents of Register C from Accumulator	SUB **C**
5. **Increment** the contents of Register D by 1	INR **D**
6. **Decrement** the contents of Register E by 1	DCR **E**

3. LOGICAL & BIT MANIPULATION OPERATIONS

1. **AND** two 8-bit numbers
2. **OR** two 8-bit numbers
3. **Exclusive-OR** two 8-bit numbers
4. **Compare** two 8-bit numbers
5. **Complement**
6. **Rotate** Left/Right Accumulator bits

- For more details about Logical instructions follow this link:

https://www.tutorialspoint.com/microprocessor/microprocessor_8085_logical_instructions.htm

MANIPULATION OPERATIONS/INSTRUCTIONS

1. Logically **AND** Register H with Accumulator	ANA **H**
2. Logically **OR** Register L with Accumulator	ORA **L**
3. Logically **XOR** Register B with Accumulator	XRA **B**
4. **Compare** contents of Register C with Accumulator	CMP **C**
5. **Complement** Accumulator	CMA
6. **Rotate** Accumulator Left	RAL

4. BRANCHING OPERATIONS

These operations are used to control the flow of program execution

1.Jumps

- Conditional jumps
- Unconditional jumps

2.Call & Return

- Conditional Call & Return
- Unconditional Call & Return

EXAMPLE BRANCHING OPERATIONS/ INSTRUCTIONS

1. **Jump** to a 16-bit Address 2080H if **Carry** flag is **SET**	JC **2080H**
2. Unconditional **Jump**	JMP **2050H**
3. **Call** a subroutine with its 16-bit Address	CALL **3050H**
4. **Return back** from the Call	RET
5. **Call** a subroutine with its 16-bit Address if **Carry** flag is **RESET**	CNC **3050H**
6. **Return** if **Zero** flag is **SET**	RZ

- For more details about branching instructions follow this link:

- https://www.tutorialspoint.com/microprocessor/microprocessor_8085_branching_instructions.htm

5. MACHINE CONTROL INSTRUCTIONS

- These instructions affect the operation of the processor. For e.g.

  - EI Enable Interrupt System
  - DI Disable Interrupt System
  - HLT Halt (stop program execution)
  - NOP No Operation (Do not perform any operation)

- For more details about Machine control instructions follow this link:

- https://www.tutorialspoint.com/microprocessor/microprocessor_8085_control_instructions.htm