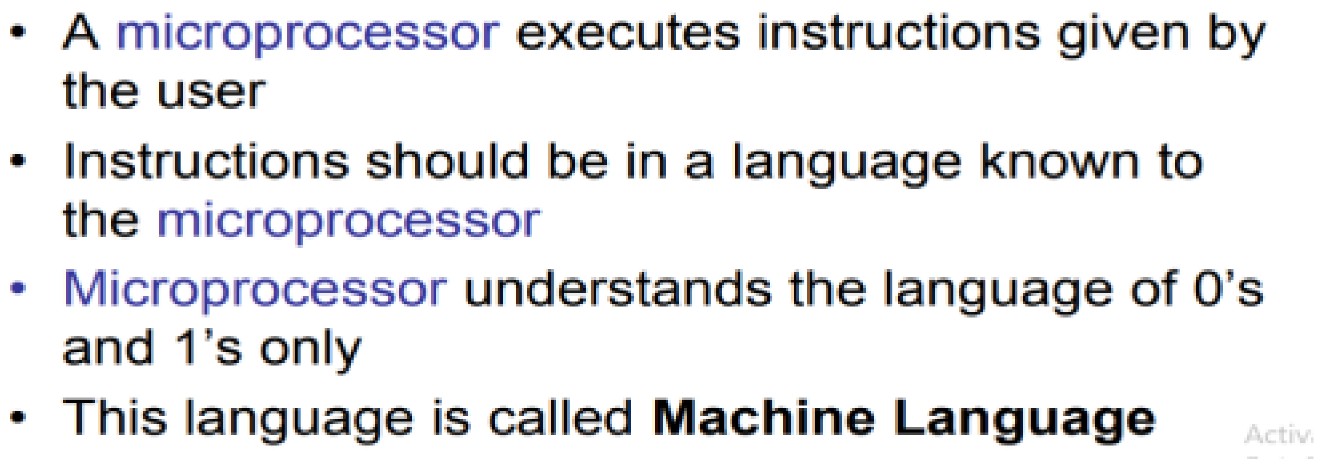
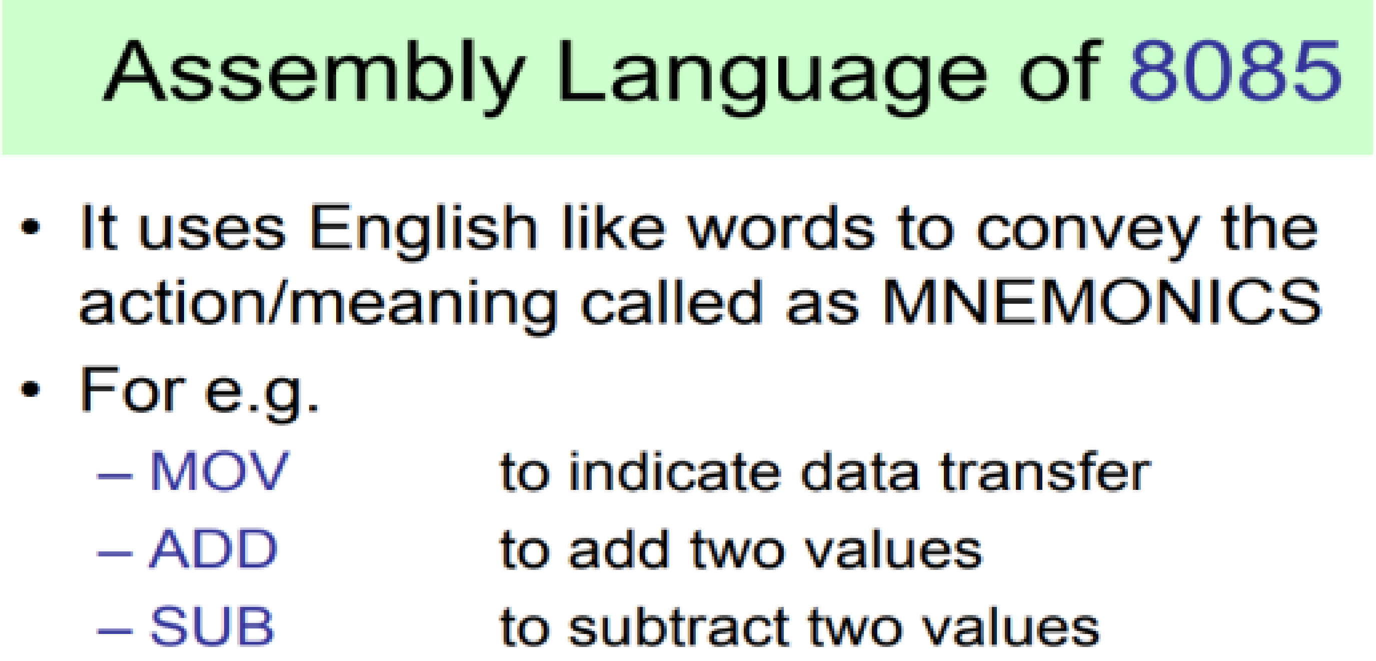
Unit-4

Introduction to Assembly language programming

Introduction:



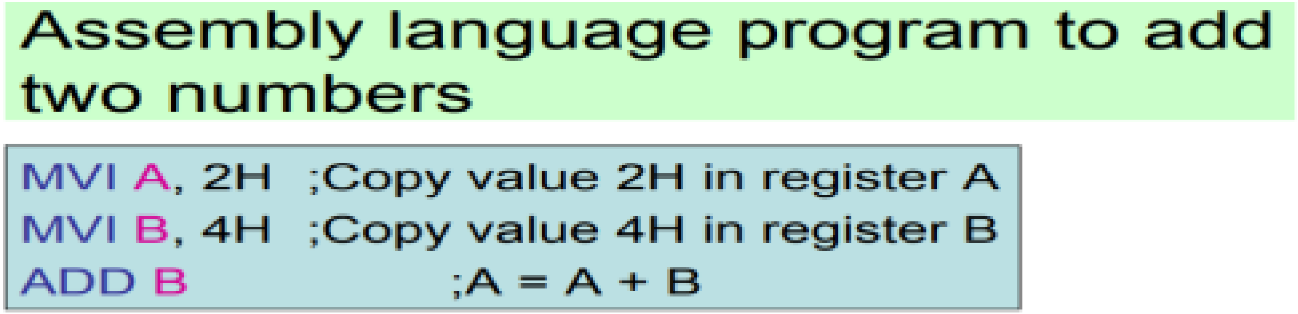
- A microprocessor executes instructions given by the user
- Instructions should be in a language known to the microprocessor
- Microprocessor understands the language of 0's and 1's only
- This language is called **Machine Language**



Assembly Language of 8085

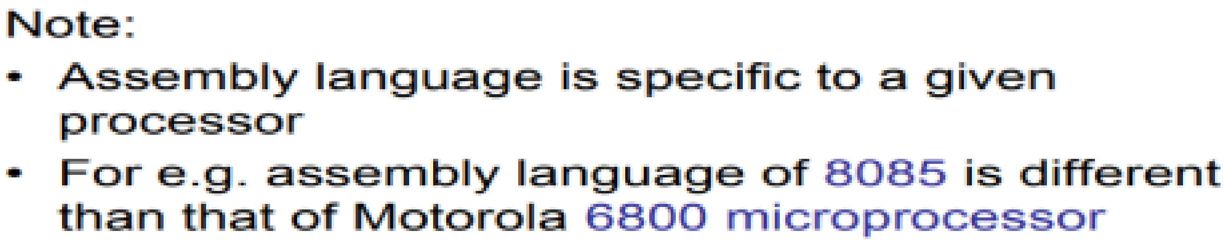
- It uses English like words to convey the action/meaning called as MNEMONICS
- For e.g.

– 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
```

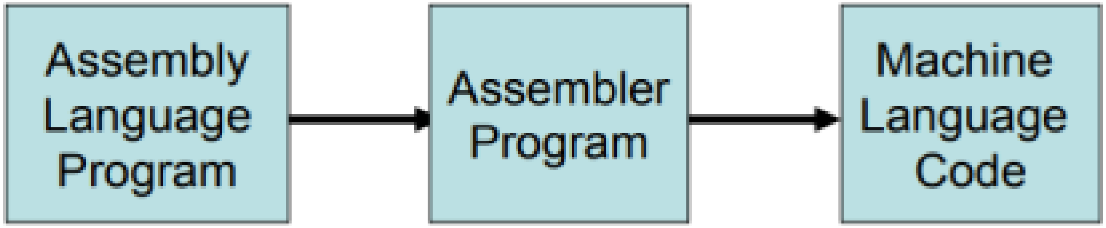


Note:

- 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

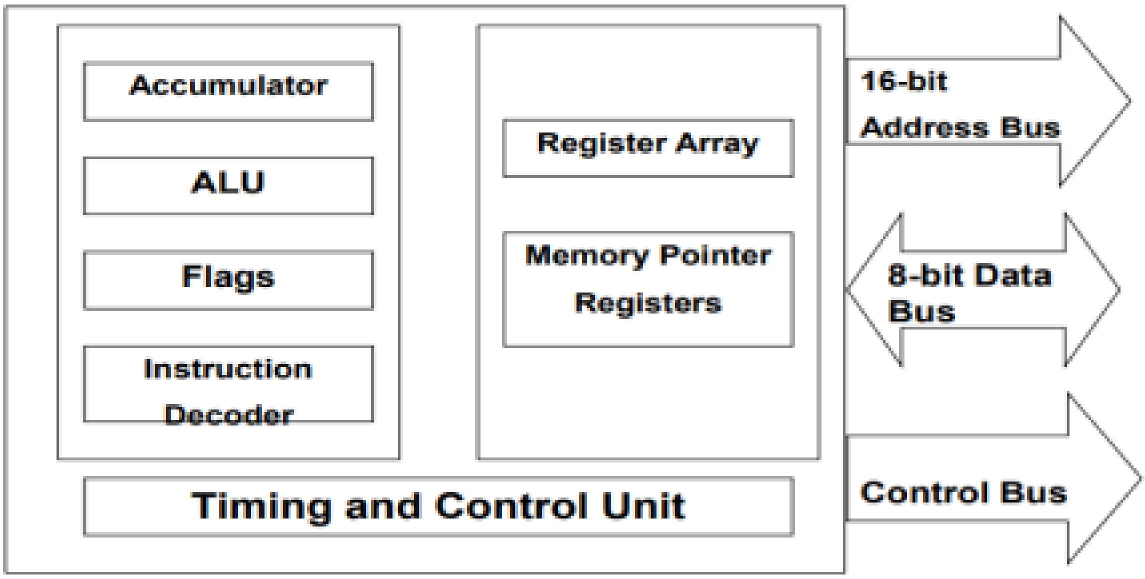


Low-level/High-level languages

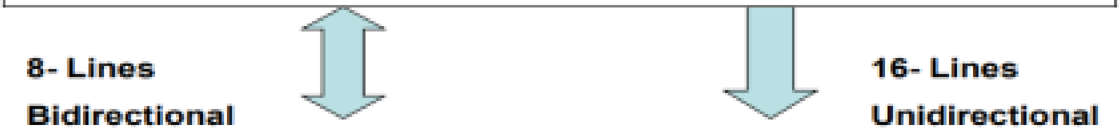
- Machine language and Assembly language are both
  - Microprocessor specific (**Machine dependent**)

    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

Programming model of 8085



<table>
<tr><td rowspan="2">Accumulator (8-bit)</td><td colspan="8">Flag Register (8-bit)</td></tr>
<tr><td>S</td><td>Z</td><td></td><td>AC</td><td></td><td>P</td><td></td><td>CY</td></tr>
<tr><td>B (8-bit)</td><td colspan="8">C (8-bit)</td></tr>
<tr><td>D (8-bit)</td><td colspan="8">E (8-bit)</td></tr>
<tr><td>H (8-bit)</td><td colspan="8">L (8-bit)</td></tr>
<tr><td colspan="9">Stack Pointer (SP) (16-bit)</td></tr>
<tr><td colspan="9">Program Counter (PC) (16-bit)</td></tr>
</table>



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

Instruction Set of 8085

2. Arithmetic Instructions:

1.Addition of two 8-bit/16-bit numbers
2. Subtraction of two-8-bit/16-bit numbers
3. Increment/Decrement a 8-bit number

Examples:

Addition:

ADD R A A + R [Add contents of R to the contents of Accumulator A and stored in A] ADD M A A + HL [Add contents of M(Memory pointer) i.e. register pair HL to the contents of Accumulator A and stored in A] ADI A A + data e.g. ADI 32H A A+32H [Add immediate 8-bit data 32H to the contents of A and stored in A] Subtraction:

SUB R A A – R [Subtract contents of R from the contents of Accumulator A and stored in A]

SUB M A A – HL [Subtract contents of M(Memory pointer) i.e. register pair HL from the contents of Accumulator A and stored in A] SUI 32H A A – 32H [Subtract immediate 8-bit data 32H from the contents of A and stored in A]

Addition with carry:

ADC R A A + R + C [Add contents of R to the contents of Accumulator A with carry and stored in A]

ADC M A A + HL + C [Add contents of M(Memory pointer) i.e. register pair HL to the contents of Accumulator A with carry and stored in A]

Subtraction with borrow:

SBB R A A – R – B [Subtract contents of R from the contents of Accumulator A with borrow and stored in A]

SBB M A A – HL – B [Subtract contents of M(Memory pointer) i.e. register pair HL from the contents of Accumulator A WITH borrow and stored in A]

Increment:

Example:

INR R R R + 1 [Increment the contents of register R by 1]

e.g. R = 06

R = 6 + 1 = 7

DCR R R R - 1 [decrement the contents of register R by 1]

e.g. R = 06

R = 6 - 1 = 5

Logical instructions:

AND

i) ANA R (Register) A R (Content of A is added with content of register) ii) ANA M A A ^ HL

iii) ANI data (8 bit) A A ^ data (8 bit)

ANA R

e.g.

A = 24

R =02

X	Y	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

A= 0010 0100

0000 0010

0000 0000

A =0

OR :

i) ORA R (Register) A R (Content of A is added with content of register) ii) ORA M A A HL

iii) ORI data (8 bit) A A data (8 bit)

ORA R

e.g. A= 26

R = 05

X Y OUTPUT

0 0 0

0 1 1

1 0 1

1 1 1

A = 0010 0110

0000 0101

0010 0111

A= 27

XOR

i) XRA R (Register) A A R

ii) XRA M A A HL

iii) XRI data (8 bit) A A-data (8 bit)

XRA R

e.g. A= 22

R = 04

X	Y	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

A = 0010 0010

0000 0100

0010 0110

A= 26

Compare:

Compare (register or memory) with accumulator (CMP R/M) –
This is a 1-byte instruction. It compares the data byte in the register or memory with the contents of accumulator.

1. If A less than (R/M), the CY flag is set and Zero flag is reset.
2. If A equals to (R/M), the Zero flag is set and CY flag is reset.
3. If A greater than (R/M), the CY and Zero flag are reset.

When memory is an operand, its address is specified by HL Pair.

Example:
Let register B contains data byte 62H and the accumulator A contains 57H.
Then, `1. Instruction- CMP B`

```
Before execution: A = 57, B = 62

After execution: A = 57, B = 62
```

Flags: As A less than B, thus CY is set and Z flag is reset.

`CY=1, Z=0`

2. **Compare immediate with accumulator (CPI 8-bit) –**
   This is a 2-byte instruction, the second byte being 8-bit data. It compares the second byte with the contents of accumulator.
   1. If A less than 8-bit data, the CY flag is set and Zero flag is reset.
   2. If A equals to 8-bit data, the Zero flag is set and CY flag is reset.
   3. If A greater than 8-bit data, the CY and Zero flag are reset.

   No contents are modified; however all remaining flags (S, P, AC) are affected according to the result of subtraction.

1. **Example:**
   Let the accumulator A contains C2H. Then,

```
Instruction- CPI C2H (Compare Immediate with Accumulator)

Before execution: A = C2, B = C2
```

```
After execution: A = C2, B = C2
```

Flags: As A equals to the data byte, thus Z is set and CY flag is reset.

```
CY=0, Z=1
```