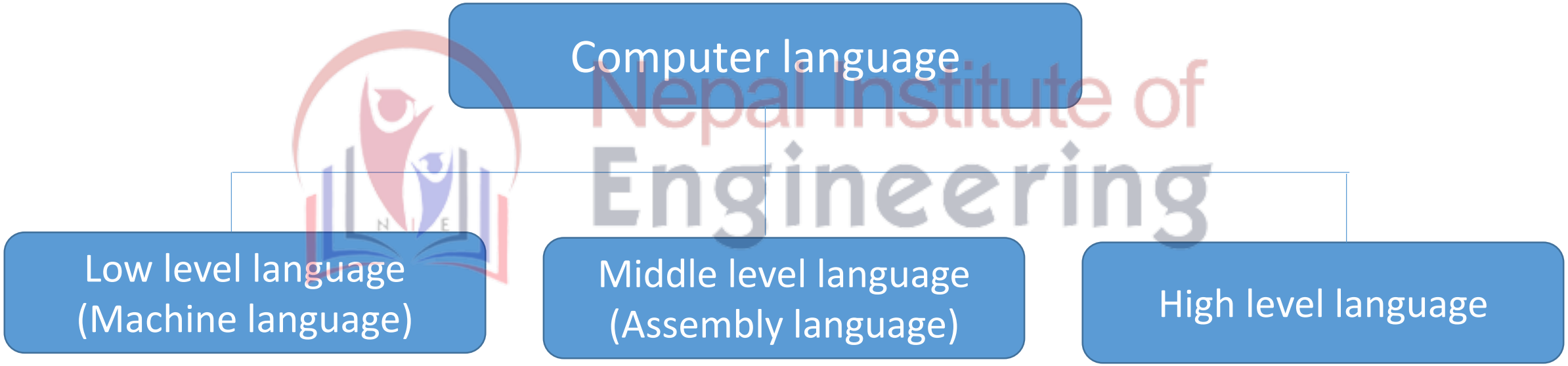
Assembly language Programming

Compiled by :- Er. Nagendra Karn

Computer language :-

- Over the years, computer languages have been evolved from low level to high level languages. In the earliest days of computer only binary language was used to write programs. The computer languages are classified as follows:-



Use 1's & 0's to create Instructions (e.g. binary)

use mnemonics to create instructions (Assembly)

similar to human language (C,C++, java)

Assembly language

- Each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities.
- Each family of processors has its own set of instructions for handling various operations such as getting input from keyboard, displaying information on screen and performing various other jobs. These set of instructions are called 'machine language instructions'.
- A processor understands only machine language instructions, which are strings of 1's and 0's. However, machine language is too obscure and complex for using in software development.
- the low-level assembly language is designed for a specific family of processors that represents various instructions in symbolic code and a more understandable form.

Assembly language

- An assembly language is the most basic programming language available for any processor. With assembly language, a programmer works only with operations that are implemented directly on the physical cpu.
- Assembly languages generally lack high level conveniences such as variables and functions, and they are not portable between various families of processor.
- They have same structure and set of command as machine language, but allow programmer to use name instead of numbers.
- Assembly language is specific to a given processor. E.g. assembly language of 8085 is different that of Motrola 6800 microprocessor.

Assembly language

- Microprocessor can't understand program written in assembly language. A program known as assembler is used to convert assembly language program to machine language.

- Assmebly language program to add two number :-
  - MVI A,2H ; copy the value 2H in register A
  - MVI B,4H ;copy value 4H in register B
  - ADD B ; A= A+B

Advantage of Assembly language :-

- The symbolic programming of Assembly Language is easier to understand and saves a lot of time and effort of the programmer.
- It is easier to correct errors and modify program instructions.
- Assembly Language has the same efficiency of execution as the machine level language.

Disadvantage of Assembly language :-

- It is machine dependent. A program written for one computer might not run in other computers with different hardware configuration.
- If you are programming in assembly language, you must have detailed knowledge of the particular microcomputer you are using.
- Assembly language programs are not portable.

Instruction set of 8085

- An instruction is a command given to the microprocessor to perform specific operation on data.
- An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions that a microprocessor supports is called **Instruction Set**.
- 8085 has **246** instructions. Each instruction is represented by an 8-bit binary value. These 8-bits of binary value is called **Op-Code** or **Instruction Byte**.
- Basically we are having five groups of instructions :-
    a. Data transfer instruction
    b. Arithmetic instructions
    c. Logical instructions
    d. Branching instructions
    e. I/O & Machine Control instructions

Data transfer instructions :-

- 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.
- Example: MOV, MVI

Instruction	Operation
MOV R1,R2	R1←R2
MOV M,R	M←R
MOV R,M	R←M
MVI R,FFH	R←FFH
MVI M,8 bit data	M←8 bit data
LXI SP,16 bit data	SP← 16 bit data

Arithmetic Instruction :-

- These instructions performs the operation like addition, subtraction, increment and decrement. e.g. ADD, SUB, INR, DCR

Instruction	Operation	Instruction	Operation	Instruction	Operation
ADD R A=30H, R=20H	A←A+R A=50H	ADC R CY=1	A←A+R+CY A=51H	INR R R=20H	R←R+1 R=21H
ADD M A=30H, HL=A1H	A←A+M A=D1H	ADC M CY=1	A←A+M+CY A=D2H	INR M HL= 20H	M←M+1 HL=21H
ADI 8 bit data A=30H, DATA=A2H	A←A+8 bit data A=D2H	ADC 8 bit data CY=1	A←A+8 bit data+CY A=D3H	DCR R R=20H	R←R-1 R=19H
SUB R A=30H, R=10H	A←A-R A=20H	SBB R CY=1	A←A-R-CY A=19H	DCR M HL=20H	M←M-1 HL=19H
SUB M A=40H, HL=20H	A←A-M A=20H	SBB M CY=1	A←A-M-CY A=19H	INX rp INX B(20H)	rp←rp+1 B=21H
SUI 8 bit data A=40H, DATA=10H	A←A-8 bit data A=30H	SBI 8 bit data CY=1	A←-A-8 bit data-CY A=29H	DCX rp DCX B(20H)	rp←rp-1 B=19H
				DAD rp DAD B	HL←HL+rp

Logical Instruction :-

- These instructions performs logical operations on data stored in register and memory. The logical operations are : AND, OR, Rotate, Compare, Complement .

example ANA, ORA, RAR, RAL, CMP, CMA

Instruction	Operation	Instruction	Operation			Instrcution	Operation
ANA R	A←A^R	CMP R	A-R	A>R	CY=0Z=0	CMA	A← $\overline{A}$
				A=R	CY=0 Z=1		
				A<R	CY=1 Z=0		
ANA M	A←A^M	CMP M	A-M			CMC	CY← $\overline{CY}$
ANI 8 bit data	A←A^8 bit data	CPI 8 bit data	A-8bit data			STC	CY←1
ORA R	A←AvR	RLC Without carry	10100011	01000111			
ORA M	A←AvM	RRC Without carry	10100011	11010001			
ORI 8 bit data	A←Av8 bit data	RAL With carry	0 10100011	1 01000110			
XRA R	A←A⊕R	RAR With carry	0 10100011	1 01010001			
XRA M	A←A⊕M						
XRA 8 bit data	A←A⊕8 bit data						

Branching Instruction :-

- Branching instructions refer to the act of switching execution to a different instruction sequence as a result of executing a branch instruction. The three types of branching instructions are: Jump, Call and Return.
- Jump :- conditional, Unconditional
- Call :- conditional, unconditional
- Return :- conditional, unconditional

Conditions	Value	Jump	Call	Return
NZ	Z=0	JNZ	CNZ	RNZ
Z	Z=1	JZ	CZ	RZ
C	C=1	JC	CC	RC
NC	C=0	JNC	CNC	RNC
PE	P=1	JPE	CPE	RPE
PO	P=0	JPO	CPO	RPO
P (Plus)	S=0	JP	CP	RP
M (Minus)	S=1	JP	CM	RM

Control Instructions :-

- The control instructions control the operation of microprocessor. Examples: HLT, NOP, EI (Enable Interrupt), DI (Disable Interrupt).
- **STACK** :-

1. PUSH :- push two bytes of data into stack.
2. POP :- pop two bytes of data from stack.
3. HTHL :- exchange top of the stack, with HL.
4. SPHL :- move contents of HL to stack.

- **I/O :-**

5. IN :- initiate input operation (e.g. IN 8 bit data :- A←8 bit data)
6. OUT :- initiate output operation (e.g. OUT 8 bit data :- A →8 bit data)

Machine related :-

7. EI (enable interrupt) :- all interrupts are enabled.
8. DI (disable interrupt) :- RST 7.5, 6.5, 5.5, & INTR will be disabled.
9. HLT (halt) :- microprocessor is halted.
10. NOP :- No operation.

8085 Program :-

- 1. Program to add two 8-bit numbers.
- *Add* numbers *05H & 13H and display result in output port 03H.*

```
MVI A,05H    //Move data 05H to accumulator
MVI B,13H    //Move data 13H to B register
ADD B        //Add contents of accumulator and B register
OUT 03H      //Transfer result to output port 03H
HLT          //Terminate the program.
```

Input: A=05H B=13H **Output**: (port 03H) = 18H

- **2. Program to add two 8-bit numbers.**
- *Add numbers from memory location 2050H & 2051H and store result in memory location 2055H.*

```
LDA 2051H          //Load contents of memory location 2051 to accumulator
MOV B,A            //Move contents of accumulator to B register
LDA 2050H          //Load contents of memory location 2050 to accumulator
ADD B              // Add contents of accumulator and B register
STA 2055H          //Store contents of accumulator in memory location 2055H
HLT                //Terminate the program.
```

Input:

Memory location	**Data**
2050H	45H
2051H	53H

Output:

Memory location	**Data**
2055H	98H

3. Program to subtract two 8 bit numbers.

subtract numbers 25H & 12H and displays result in output port 01H.

```
MVI A,25H          //Move data 05H to accumulator
MVI B,12H          //Move data 13H to B register
SUB B              //Add contents of accumulator and B register
OUT 01H            //Transfer result to output port 01H
HLT                //Terminate the program.
```

Input: A=25H B=12H **Output**: (port 03H) = 13H

4. Program to subtract two 8 bit numbers.

Subtract numbers from memory location 2050H & 2051H and store result in memory location 2055H.

```
LDA 2051H          //Load contents of memory location 2051 to accumulator
MOV B,A            //Move contents of accumulator to B register
LDA 2050H          //Load contents  of memory location 2050 to accumulator
SUB B              // Add contents of accumulator and B register
STA 2055H          //Store contents of accumulator in memory location 2055H
HLT                //Terminate the program.
```

Input:

Memory location	**Data**
2050H	65H
2051H	53H

Output:

Memory location	**Data**
2055H	12H

5. Program to find 1's complement of a number.

- Input number from memory location 2013H and store result in memory location 2052H.

```
LDA 2013H     //Load contents from memory location 2013H to accumulator
CMA           //Complement contents of accumulator
STA 2052H     //Store result in memory location 2052H
HLT           //Terminate the program.
```

Input:

Memory location	Data
2013H	12H

Output:

Memory location	Data
2052H	EDH

6. Program to find 2's complement of a number.

Input number from memory location 2013H and store result in memory location 2052H.

```
LDA 2013H     //Load contents from memory location 2013H to accumulator
CMA           //Complement contents of accumulator
ADI 01H       //Add 01H to the contents of accumulator
STA 2052H     //Store result in memory location 2052H
HLT           //Terminate the program.
```

Input:

Memory location	Data
2013H	12H

Output:

Memory location	Data
2052H	EEH

7. Program to right shift 8 bit numbers.

Shift an eight-bit data four bits right. Assume data is in memory location 2051H. Store result in memory location 2055H.

```
LDA 2051H    //Load data from memory location 2051H to accumulator
RAR          //Rotate accumulator 1-bit right
RAR
RAR
RAR
STA 2055H    //Store result in memory location 2055H
HLT          //Terminate the program.
```

8. Program to left shift 8 bit numbers.

Shift an eight-bit data four bits left. Assume data is in memory location 2051H. Store result in memory location 2055H.

```
LDA 2051H    //Load data from memory location 2051H to accumulator
RAL          //Rotate accumulator 1-bit left
RAL
RAL
RAL
STA 2055H    //Store result in memory location 2055H
HLT          //Terminate the program.
```

- **Program to add two 16-bit numbers.**
- Add numbers 1124H & 2253H and store result in memory location 2055H & 2056H.

```
LXI H,1124H      //Load 16-bit data 1124H to HL pair
LXI D,2253H      //Load 16-bit data 2253H to DE pair
MOV A,L          //Move contents of register L to Accumulator
ADD E            //Add contents of Accumulator and E register
MOV L,A          //Move contents of Accumulator to L register
MOV A,H          //Move contents of register H to Accumulator
ADC D            //Add contents of Accumulator and D register with carry
MOV H,A          //Move contents of Accumulator to register H
SHLD 2055H       //Store contents of HL pair in memory address 2055H & 2056H
HLT              //Terminate the program.
```

Input:

Register pair	Data
HL	65H
DE	53H

Output:

Memory location	Data
2055H	77H
2056H	33H

10. Program to add two 16 bit numbers :-

Input first number from memory location 2050H & 2051H and second number from memory location 2052H & 2053H and store result in memory location 2055H & 2056H.

```
LHLD 2052H      //Load 16-bit number from memory location 2052H & 2053H to HL pair
XCHG            //Exchange contents of HL pair and DE pair
LHLD 2050H      //Load 16-bit number from memory location 2050H & 2051H to HL pair
MOV A,L         //Move contents of register L to Accumulator
ADD E           //Add contents of Accumulator and E register
MOV L,A         //Move contents of Accumulator to L register
MOV A,H         //Move contents of register H to Accumulator
ADC D           //Add contents of Accumulator and D register with carry
MOV H,A         //Move contents of Accumulator to register H
SHLD 2055H      //Store contents of HL pair in memory address 2055H & 2056H
HLT             //Terminate the program.
```


Input:

Register pair	Data
2050H	33H
2051H	45H
2052H	24H
2053H	34H

Output:

Memory location	Data
2055H	57H
2056H	79H

11. Program to subtract two 16 bit numbers :-

- Subtract number 1234H from 4897H and store result in memory location 2055H & 2056H.

```
LXI H,4567H   //Load 16-bit data 4897H to HL pair
LXI D,1234H    //Load 16-bit data 1234H to DE pair
MOV A,L        //Move contents of register L to Accumulator
SUB E          //Subtract contents of Accumulator and E register
MOV L,A        //Move contents of Accumulator to L register
MOV A,H        //Move contents of register H to Accumulator
SBB D          //Subtract contents of Accumulator and D register with borrow
MOV H,A        //Move contents of Accumulator to register H
SHLD 2055H     //Store contents of HL pair in memory address 2055H & 2056H
HLT            //Terminate the program.
```

Input:

Register pair	**Data**
HL	4879H
DE	1234H

Output:

Memory location	**Data**
2055H	63H
2056H	36H

12. Program to subtract two 16 bit numbers :-

- Input first number from memory location 2050H & 2051H and second number from memory location 2052H & 2053H and store result in memory location 2055H & 2056H.

```
LHLD 2052H      //Load 16-bit number from memory location 2052H & 2053H to HL pair
XCHG            //Exchange contents of HL pair and DE pair
LHLD 2050H      //Load 16-bit number from memory location 2050H & 2051H to HL pair
MOV A,L         //Move contents of register L to Accumulator
SUB E           //Subtract contents of Accumulator and E register
MOV L,A         //Move contents of Accumulator to L register
MOV A,H         //Move contents of register H to Accumulator
SBB D           //Subtract contents of Accumulator and D register with carry
MOV H,A         //Move contents of Accumulator to register H
SHLD 2055H      //Store contents of HL pair in memory address 2055H & 2056H
HLT             //Terminate the program.
```


Input:

Register pair	Data
2050H	78H
2051H	45H
2052H	24H
2053H	34H

Output:

Memory location	Data
2055H	54H
2056H	11H

13. Program to Multiply two 8 bit numbers :-

- **Multiply 06 and 03 and store result in memory location 2055H.**

```
MVI A,00H
MVI B,06H
MVI C,03H
X: ADD B
DCR C
JNZ X
STA 2055H
HLT
```

14. Program to divide to 8-bit numbers.

Divide 08H and 03H and store quotient in memory location 2055H and remainder in memory location 2056H.

```
MVI A,08H
MVI B,03H
MVI C,00H
X: CMP B
JC Y
SUB B
INR C
JMP X
Y: STA 2056H
MOV A,C
STA 2055H
HLT
```

- **Program to find greatest among two 8 bit numbers.**
- Input numbers from memory location 2050H & 2051H and store greatest number in memory location 2055H.

```
LDA 2051H
MOV B,A
LDA 2050H
CMP B
JNC X
MOV A,B
X: STA 2055H
HLT
```

- **Program to find smallest among two 8 bit numbers.**
- Input numbers from memory location 2050H & 2051H and store smallest number in memory location 2055H.

```
LDA 2051H
MOV B,A
LDA 2050H
CMP B
JC X
MOV A,B
X: STA 2055H
HLT
```

- **Program to find whether the number is even or odd.**
- Input number from memory location 2050H and store result in 2055H.

```
LDA 2050H
ANI 01H
JZ X
MVI A,0DH
JMP Y
X: MVI A,0EH
Y: STA 2055H
HLT
```

- **Program to count number of 1's in a given number.**
- Input number from memory location 2050H and store result in 2055H.

```
LDA 2050H
MVI C,08H
MVI B,00H
X: RAR
JNC Y
INR B
Y: DCR C
JNZ X
MOV A,B
STA 2055H
HLT
```

- Find sum of numbers from 1 to 10.

```
MVI B,01H
MVI C,0AH
MVI A,00H
X: ADD B
INR B
DCR C
JNZ X
STA 2055H
HLT
```

- Display all odd numbers from 1 to 10.

```
LXI H,2050H
MVI B,01H
MVI C,0AH
X: MOV M,B
INX H
INR B
INR B
DCR C
DCR C
JNZ X
HLT
```

- **Find the sum of 5 numbers in array.**

```
LXI H,2050H
MVI C,O5H
MVI A,00H
X: MOV B,M
ADD B
INX H
DCR C
JNZ X
STA 2060H
HLT
```

02	03	01	02	04

- **Find the smallest number in array.**

```
LDA 2200H
MOV C,A
LXI H,2201H
MVI A,00H
X: CMP M
JC Y
MOV A,M
Y: INX H
DCR C
JNZ X
STA 2300H
HLT
```