## ASSEMBLER DIRECTIVES.

Assembler directives tell the assembler to do something other than creating the machine code for an instruction. In assembly language programming, the assembler directives instruct the assembler to

1. Process subsequent assembly language instructions
2. Define program constants
3. Reserve space for variables

***The following are the widely used 8051 assembler directives.***

**ORG (origin)**

The ORG directive is used to indicate the starting address. It can be used only when the program counter needs to be changed. The number that comes after ORG can be either in hex or in decimal.

**Eg: ORG 0000H ;Set PC to 0000.**

**EQU and SET**

EQU and SET directives assign numerical value or register name to the specified symbol name.

EQU is used to define a constant without storing information in the memory. The symbol defined with EQU should not be redefined.

SET directive allows redefinition of symbols at a later stage.

**DB (DEFINE BYTE)**

The DB directive is used to define an 8 bit data. DB directive initializes memory with 8 bit values. The numbers can be in decimal, binary, hex or in ASCII formats. For decimal, the 'D' after the decimal number is optional, but for binary and hexadecimal, 'B' and 'H' are required. For ASCII, the number is written in quotation marks ('LIKE This).

```
DATA1:  DB   40H          ; hex
DATA2:  DB   01011100B    ; b i n a r y
DATA3:  DB   48           ; decimal
DATA4:  DB   ' HELLO W'   ; ASCII
```

**END**

The END directive signals the end of the assembly module. It indicates the end of the program to the assembler. Any text in the assembly file that appears after the END directive is ignored. If the END statement is missing, the assembler will generate an error message.

# ASSEMBLY LANGUAGE PROGRAMS.

1. **Write a program to add the values of locations 50H and 51H and store the result in locations in 52h and 53H.**

```
ORG 0000H          ; Set program counter 0000H
MOV A,50H          ; Load the contents of Memory location 50H into A ADD
ADD A,51H          ; Add the contents of memory 51H with CONTENTS A
MOV 52H,A          ; Save the LS byte of the result in 52H
MOV A, #00         ; Load 00H into A
ADDC A, #00        ; Add the immediate data and carry to A
MOV 53H,A          ; Save the MS byte of the result in location 53h
END
```

2. **Write a program to store data FFH into RAM memory locations 50H to 58H using direct addressing mode**

```
ORG 0000H          ; Set program counter 0000H
MOV A, #0FFH       ; Load FFH into A
MOV 50H, A         ; Store contents of A in location 50H
MOV 51H, A         ; Store contents of A in location 5IH
MOV 52H, A         ; Store contents of A in location 52H
MOV 53H, A         ; Store contents of A in location 53H
MOV 54H, A         ; Store contents of A in location 54H
MOV 55H, A         ; Store contents of A in location 55H
MOV 56H, A         ; Store contents of A in location 56H
MOV 57H, A         ; Store contents of A in location 57H
MOV 58H, A         ; Store contents of A in location 58H
END
```

3. **Write a program to add two 16 bit numbers stored at locations 51H-52H and 55H-56H and store the result in locations 40H, 41H and 42H. Assume that the least significant byte of data and the result is stored in low address and the most significant byte of data or the result is stored in high address.**

```
ORG 0000H          ; Set program counter 0000H
MOV A,51H          ; Load the contents of memory location 51H into A
ADD A,55H          ; Add the contents of 55H with contents of A
MOV 40H,A          ; Save the LS byte of the result in location 40H
MOV A,52H          ; Load the contents of 52H into A
ADDC A,56H         ; Add the contents of 56H and CY flag with A
MOV 41H,A          ; Save the second byte of the result in 41H
MOV A,#00          ; Load 00H into A
ADDC A,#00         ; Add the immediate data 00H and CY to A
MOV 42H,A          ; Save the MS byte of the result in location 42H
END
```

4. **Write a program to store data FFH into RAM memory locations 50H to 58H using indirect addressing mode.**

```
ORG 0000H            ; Set program counter 0000H
MOV A, #0FFH         ; Load FFH into A
MOV RO, #50H         ; Load pointer, R0-50H
MOV R5, #08H         ; Load counter, R5-08H
Start:MOV @RO, A     ; Copy contents of A to RAM pointed by R0
INC RO               ; Increment pointer
DJNZ R5, start ; Repeat until R5 is zero
END
```