* Steps to Run an Assembly language program:

i) Create the source program File(.ASM):

first we write the assembly program using any text editor and the file is saved with the extension .ASM.

ii) Assemble the program:

The MASM assembler translates the source file (.ASM) into an object file (.OBJ)

iii) Link the program:

The link program combines one or more object files and resolve addresses. It generates executable file (.EXE)

iv) Run the program:

The program runs and produces required Output.

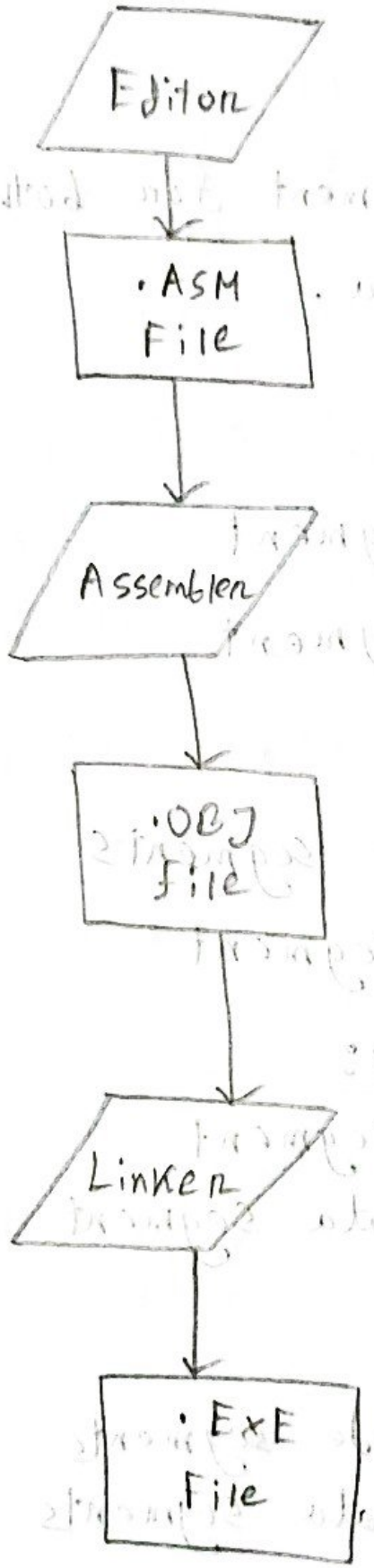

Fig: Creating and Running assembly code.

* Memory Models in Assembly language.

1) Tiny Model:

i) Only one segment for both code and data.

2) Small Model:

i) One code segment

ii) One data segment

3) Medium Model:

i) Multiple code segments

ii) One data segment

4) compact Model:

i) One code segment

ii) Multiple data segment

5) Large Model:

i) Multiple code segments

ii) Multiple data segments

* if-else ladder:

C code

```
int x=-5;
if (x>0 )
   Printf("Positive");
else if(x<0)
   Printf("Negative");
else
   Printf ("zero");
```

Assembly

```
.model small
.stack 100h
.data
msg_Pos db 'Positive $'
msg_neg db 'Negative $'
msg_zero db 'zero $'
x  dw  -5
.code
main proc
      mov  ax, @data
      mov  ds, ax

mov  ax, x  ; load x
cmp  ax, 0  ; compare with 0
jg   positive ; x>0 ?
jl   Negative ; x<0 ?
jmp  zero     ; else
```

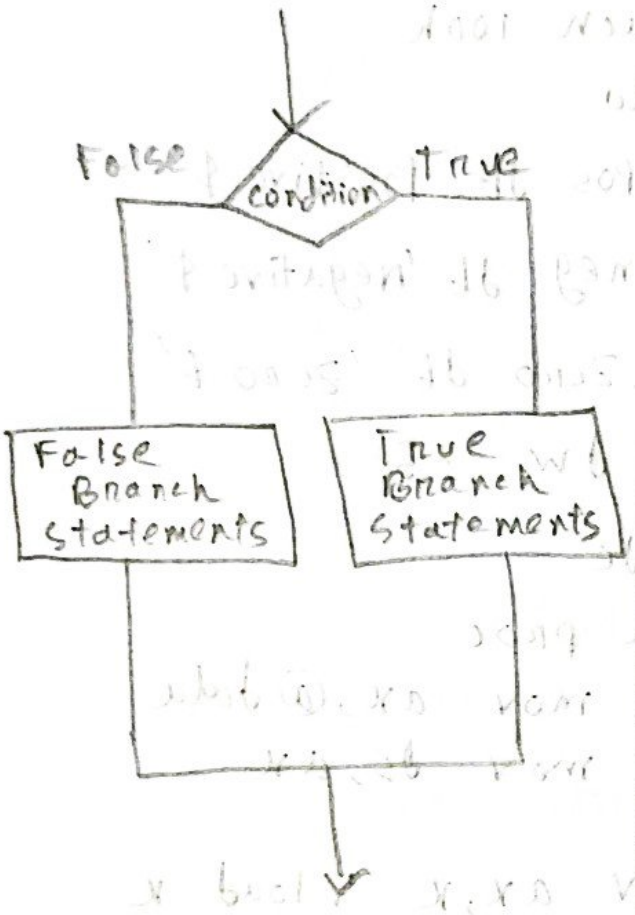

Fig: Flow chart of if..else.

```
Positive:
    mov ah, 9
    lea dx, msg_Pos
    int 21h
    jmp exit

Negative:
    mov ah, 9
    lea dx, msg_neg
    int 21h
    jmp exit

zero:
    mov ah, 9
    lea dx, msg_zero
    int 21h

exit:
    mov ah, 4ch
    int 21h
    main endp
end main
```

* 10 character conversion in Assembly:

```
.model small
.stack 100h
.data
msg1 db 'Enter 10 characters: $'
msg2 db 10,13, 'converted: $'
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg1
    int 21h
; loop starts from here.
    mov cx, 10
    read_loop:
        mov ah, 1
        int 21h
```

```
; check for lowercase
    cmp al, 'a'
    jl check_upper
    cmp al, 'z'
    jg check_upper
    sub al, 32
    jmp print

check_upper:
        cmp al, 'A'
        jl print
        cmp al, 'Z'
        jg print
        add 32

print:
        mov ah, 2
        mov dl, 10    ; new line
        int 21h
        mov dl, 13
        int 21h

        mov ah, 9
        lea dx, msg2
        int 21h
```

```
    mov  ah, 2
    mov  dl, al
    int  21h

    loop  Read_loop

    mov  ah, 4ch
    int  21h
    main  endp
end  main
```

— ✦ —

* Odd - Even:

```
    mov  al, num
    AND  al, 1
    JZ   even

odd:
    ; msg print odd

even:
    ; msg print even
```