Assembly language

Basic Structure

```
.model small          ; memory model (one code + data segment)
.stack 100h           ; stack size (256 bytes)

.data                 ; data segment

  ; variables

.code                 ; code segment
main proc
    mov ax, @data
    mov ds, ax        ; initialize DS

    ; program starts from here


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

Common DOS Functions

Ah	Use
1	Read character (AL)
2	Print character (DL)
9	Print $ terminated string
4ch	Exit program

Flags register

i) CF – carry Flag
ii) PF – Parity Flag
iii) AF – Auxiliary Carry
iv) ZF – Zero Flag
v) SF – Sign Flag
vi) OF – overflow Flag

Signed Jumps (Use after CMP)

i) Greater — JG

ii) Greater/equal — JGE

iii) Less — JL

iv) Less/equal — JLE

v) Equal — JE

vi) Not Equal — JNE

[Jmp] ↓ To initial level jump

— x —

C code

i)
```
if (x>y)
    a=1;
else
    a=2;
```

Assembly

```
cmp x, y
jg go_to_greater
mov a, 2
jmp if_ended
go_to_greater:
mov a, 1
if_ended:
```

Nested If-Else

c code

```
if(a>b)
    if(a>c)
        max=a;
    else
        max=c;
else
    if (b>c)
        max=b;
    else
        max=c;
```

assembly

```
cmp a,b
jle else1
cmp a,c
jle setc1
mov max,a
jmp done
setc1:
    mov max,c
    jmp done
else1:
    cmp b,c
    jle setc2
    mov max,b
    jmp done
setc2:
    mov max,c
done:
```

switch case

C

```
switch(choice)
{
  case 1 : A;
  case 2 : B;
  case 3 : C;
  default : D;
}
```

assembly

```
cmp choice,1
je case1
cmp choice,2
je case2
cmp choice,3
je case3
jmp default

case 1:
  ; code A
  jmp end_switch
case 22:
  ; code B
  jmp end_switch
case 3:
  ; code C
  jmp end_switch
default :
  ; code D
end_switch.
```

For loop

C code

```
for(i=0;i<10;i++)
    print(i);
```

Assembly

```
mov cx, 10 ; loop count
mov bx, 0 ; i = 0
for_loop:
    mov ah, 2
    mov dl, bx
    int 21h
    inc bx
    loop for_loop
```

while loop

C

```
while(x<5)
    x++;
```

Assembly

```
while_start:
    cmp x, 5
    jge while_end
    inc x
    jmp while_start
while_end.
```

Do - while

C

```
do {  x++;
} while (x<5);
```

Assembly

```
do_start:
    inc x
    cmp x,5
    jl do_start
```

some instructions

i) Reading one character:

```
mov ah,1
int 21h
; AL = char
```

ii) Printing one character:

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

iii) Printing string:

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

Print a string

```
.model small
.stack 100h
.data
    msg db 'Hello, Assembly world! $'
.code
main proc
        mov ax, @data
        mov ds, ax

        mov ah, 9
        lea dx, msg
        int 21h

        mov ah, 4ch
        int 21h

main endp
end main
```