**PRACTICAL # 05**

**OBJECT:**

Jumps and Branching in Assembly.

**THEORY:**

Unlike high level languages, branching in assembly is implemented using conditions and jumps to labels. In high level languages there are dedicated keywords for conditionals and loops, but in assembly we need to wire things up ourselves.

**Branching Instructions**

Two types of jumps in assembly are **unconditional** jumps and **conditional** jumps.

The unconditional jumps simply jump to the specified location without any condition. They resemble the **goto** statement in C. The conditional jumps use **CMP** statement for checking the condition upon. If the condition is satisfied, the jump occurs otherwise it doesn't. **CMP** opcode compares two operands and affects the flags register accordingly.

**Unconditional Jump (jmp)**

The unconditional jump transfers control unconditionally. It is similar to the goto statement in C.

main proc

mov ah, 2

**jmp skipd ; unconditional jump to label skipd**

mov dl, 'a' ;character a

int 21h

skipd:

mov dl, 'e' ;character e

int 21h

main endp

end main

The second statement in the code unconditionally jumps to the label name skipd. Thus the program only outputs **e**, instead of **ae**.

**Conditional Jump (jmp)**

The conditional jump is the jump that is executed based on some condition. There are many conditional jumps available in assembly language. They are used to implement if/else conditions and loops. The magic behind the scene that makes a jump possible is, the **CMP** performs arithmetic operation (subtraction) and modifies the contents of the **flags registers** accordingly. The conditional jump statement then makes the decision based on the status of flags register.

**Note**: There must not be any instruction between CMP statement and the conditional jump depending on it. If any other statement is put in between these two statements, it may modify flags register contents resulting in incorrect jump!

Some of the main conditional jumps along with their equivalents are listed here.

* JB ≡ JC ≡ JNAE
* JAE ≡ JNB ≡ JNC
* JE ≡ JZ
* JNE ≡ JNZ
* JBE ≡ JNA
* JA ≡ JNBE
* JP ≡ JPE
* JNP ≡ JPO
* JL ≡ JNGE
* JGE ≡ JNL
* JLE ≡ JNG
* JG ≡ JNLE

Where J = Jump, B = Below, C = Carry, N = Not, E = Equal, Z = Zero, A = Above, P = Parity, L = Less, G = Greater

The above list of conditional jumps shows equivalent jumps in each line. The first JB (Jump if Below) is the same as JNAE (Jump if Not Above or Equal) is the same as JC (Jump if Carry).

**Program:**

The program below finds greater number of two given numbers using conditional jump.

.model small

.stack 100h

.data

num1 db 7

num2 db 4

.code

main proc

;initiaze the data segment register

mov ax, @data

mov ds, ax

mov ah, 2 ; Output interrupt

;check for greater number

mov bh, num1

cmp bh, num2

jg num1greater ;conditional jump immediately follows cmp statment

mov dl, num2

add dl, 48

int 21h

jmp skipp ;unconditional jump

num1greater:

mov dl, num1

add dl, 48

int 21h

skipp:

mov ah, 4ch

int 21h

main endp

end main

**ACTIVITIES**

**Activity 1**

Build (Assemble and link) and run the above program, but take the two numbers input from user. You can build the program using MASM or Emu8086.

**Activity 2**

Write a program that finds greater of three numbers and prints that number. You may hardcode the value for three numbers. Note each of the numbers should be single digit.

**REVIEW QUESTIONS**

1. How a conditional jump makes jump decision?
2. Why is flags register important in the conditional jumping?
3. What does a label represent in program and why is it necessary in a jump statement?
4. If/else in assembly is accomplished using a combination of which types of jumps?