IBADAT INTERNATIONAL UNIVERSITY ISLAMABAD

Name	Roll No	LAB No .	Date
HAMZA RIAZ	5121323017	09	06/1/2025

Submitted to

MS.SARIA NOREEN

Subject

COAL(LAB)

Department

BS SOFTWARE ENGINEERING
(3RD SEMESTER)

EXPERIMENT 09
BRANCH INSTRUCTION-II

To understand the concepts of 8086 branch instructions in assembly language.

Program Flow Control

Controlling the program flow is a very important thing, this is where your program can make decisions according to certain conditions.

Unconditional Jumps

The basic instruction that transfers control to another point in the program is **JMP**. The basic syntax of **JMP** instruction:

```
JMP label
```

To declare a *label* in your program, just type its name and add "**:**" to the end, label can be any character combination but it cannot start with a number, for example here are 3 legal label definitions:

```
label1:
label2:
a:
```

Label can be declared on a separate line or before any other instruction, for

```
example: x1:
MOV AX, 1
x2: MOV AX, 2
```

Here is an example of **JMP**

```
instruction: ORG 100h
MOV AX, 5 ; set AX
to 5. MOV BX, 2 ;
set BX to 2.
JMP calc ; go to 'calc'.
back: JMP stop ; go
to 'stop'. calc:
ADD AX, BX ; add BX to AX.
JMP back ; go
'back'. stop:
RET ; return to operating system.
END ; directive to stop the compiler.
```

Of course there is an easier way to calculate the sum of two numbers, but it's still a good example of **JMP** instruction.

As you can see from this example **JMP** is able to transfer control both forward and backward. It can jump anywhere in current code segment (65,535 bytes).

Short Conditional Jumps

Unlike **JMP** instruction that does an unconditional jump, there are instructions that do a conditional jumps (jump only when some conditions are in act). These instructions are divided in three groups, first group just test single flag, second compares numbers as signed, and third compares numbers as unsigned.

Jump instructions that test single flag:

JZ , JE	Jump if Zero (Equal).	ZF = 1	JNZ, JNE
JC , JB, JNAE	Jump if Carry (Below, Not Above Equal).	CF = 1	JNC, JNB, JAE
JS	Jump if Sign.	SF = 1	JNS
JO	Jump if Overflow.	OF = 1	JNO
JPE, JP	Jump if Parity Even.	PF = 1	JPO
JNZ , JNE	Jump if Not Zero (Not Equal).	ZF = 0	JZ, JE
JNC , JNB, JAE	Jump if Not Carry (Not Below, Above Equal).	CF = 0	JC, JB, JNAE
JNS	Jump if Not Sign.	SF = 0	JS
JNO	Jump if Not Overflow.	OF = 0	JO
JPO, JNP	Jump if Parity Odd (No Parity).	PF = 0	JPE, JP

There are some instructions that do that same thing, that's correct, they even are assembled into the same machine code, so it's good to remember that when you compile **JE** instruction - you will get it disassembled as: **JZ**.
Different names are used to make programs easier to understand and code.

Here's a program written in 8086 assembly language that compares two variables and displays which one is greater using conditional jumps. This example demonstrates the use of conditional branch instructions like JG (jump if greater) and JL (jump if less).

LAB TASK

Assembly Program to Compare Two Variables

```
; Program to compare two variables and display which one is greater

ORG 100h    ; Set origin for COM program

; Define variables
MOV AX, 5   ; Variable 1: AX = 5
MOV BX, 8   ; Variable 2: BX = 8

; Compare AX and BX
CMP AX, BX  ; Compare AX with BX
JG AX_Greater  ; If AX > BX, jump to AX_Greater
JL BX_Greater  ; If AX < BX, jump to BX_Greater

; If AX == BX
```

```
MOV DX, OFFSET MsgEqual
JMP DisplayMessage  ; Jump to display message

AX_Greater:
MOV DX, OFFSET MsgAXGreater
JMP DisplayMessage  ; Jump to display message

BX_Greater:
MOV DX, OFFSET MsgBXGreater

DisplayMessage:
MOV AH, 09h         ; DOS interrupt for displaying string
INT 21h             ; Call interrupt

RET                 ; Return to OS

; Define output messages
MsgAXGreater DB 'AX is greater than BX$', 0
MsgBXGreater DB 'BX is greater than AX$', 0
MsgEqual DB 'AX is equal to BX$', 0

END
```

Explanation

Variable Initialization:

1. `AX` and `BX` are used as the variables to be compared. Their values are set to `5` and `8`, respectively.

Comparison and Conditional Jumps:

1. The `CMP` instruction compares `AX` and `BX`.
2. If `AX > BX`, the `JG` instruction jumps to the `AX_Greater` label.
3. If `AX < BX`, the `JL` instruction jumps to the `BX_Greater` label.
4. If neither condition is met (i.e., `AX == BX`), it continues to display an equality message.
2.

Displaying Messages:

1. The message corresponding to the condition is displayed using the DOS interrupt `INT 21h` with function `09h`.

RET Instruction:

1. The `RET` instruction returns control to the operating system after execution.

Tasks Completed

- Comparison of two variables.
- Conditional branching to display the result.
- Use of unconditional and conditional jumps.

Conclusion

This program demonstrates the basic usage of `CMP`, `JG`, and `JL` instructions for conditional branching in 8086 assembly. It illustrates how the program can make decisions based on comparisons and execute specific code paths accordingly.