Experiment Number : (10)

Unsigned Jump Instructions

Object:

To understand the unsigned jump instructions and how can use it properly

Theory

Controlling the program flow is a very important thing, this is where your program can make decisions according to certain conditions. Jump instruction is the important instruction which that can control of the programs flow.

There are two main types of this instruction, and there are

1. **Unconditional JUMP instruction: this instruction is execute without any condition.**
2. **Conditional JUMP instruction: these instruction not execute until the condition is true**

Unconditional Jump Instructions:

The basic instruction that transfers control to another point in the program is unconditional jump instruction, and represented by:

JMP Label

To declare a Label in your program, just type the name and add":" to the end, label can any character combination but cannot start with a number.

For example here 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 **or**
X2: MOV AX, 2.

Short Conditional Jump Instructions:

Unlike JMP instruction that dose an unconditional jump, there are instructions that do a conditional jumps (jump only where some condition in act).
These instructions (Conditional jump instructions) contain also two main types and there are:

1. **Unsigned conditional Jump instructions.**
2. **Signed conditional Jump instructions.**

Unsigned Conditional Jump Instructions:

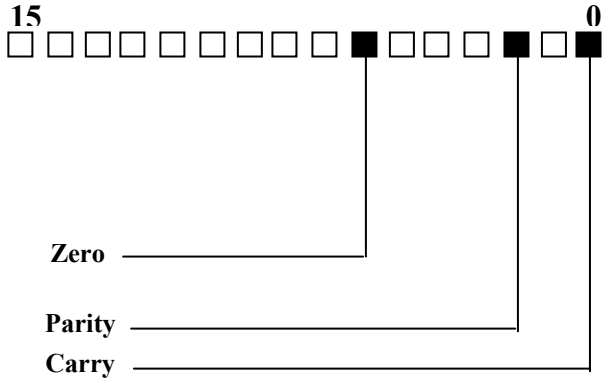
This instructions represented by:

1.	**JC Label**	**Jump if carry**	**CF = 1**
2.	**JPE Label**	**Jump if parity even**	**PF = 1**
3.	**JPO Label**	**Jump if parity odd**	**PF = 0**
4.	**JZ Label**	**Jump if zero**	**ZF = 1**
5.	**JE Label**	**Jump if equal**	**ZF = 1**
6.	**JA Label**	**Jump if above**	**CF = 0 & ZF = 0**
7.	**JB Label**	**Jump if below**	**CF = 1**
8.	**JAE Label**	**Jump if above or equal**	**CF = 0**
9.	**JBE Label**	**Jump if below or equal**	**CF = 1 OR ZF = 1**

Some of the above instructions can be negative as seen below:

1.	**JNC Label**	**Jump if not-carry**	**CF = 0**
2.	**JNZ Label**	**Jump if not-zero**	**ZF = 0**
3.	**JNE Label**	**Jump if not-equal**	**ZF = 0**
4.	**JNA Label**	**Jump if not-above**	**CF = 1 OR ZF = 1**
5.	**JNB Label**	**Jump if not-below**	**CF = 0**

All these instructions the sign number is neglect (I.E the negative number can't represented by above instructions)
All of the above instructions test the some of the flag and its:



Example:

Write 8086 program to find the largest number of DT_1. store the result into DT_2. DT_1 = 7,1,5,4,8,6,3,2,9

```
.DATA
DT1 DB 7, 1, 5, 4, 8, 6, 3, 2, 9
DT2 DB 0
.CODE
MOV AX, @DATA
MOV DS, AX
MOV SI, OFFSET DT1
MOV DI, OFFSET DT2
MOV BX, 0
MOV CX, 0008
MOV AL, [SI+BX]
L: INC BX
CMP AL, [SI+BX]
JA M
MOV AL, [SI+BX]
M:LOOP L
MOV [DI], AL
RET
```

Procedure

1. **Write 8086 program to compare between two numbers (X_1 & X_2) each one is 1 byte.**
2. **If X_1 is grater than X_2 store X_1 into DT_1.**
3. **If X_1 is equal X_2 store X_1 into DT_2.**
4. **if X_1 is smallest than X_2 store X_2 into DT_3.**
5. **Let X_1 = 0A & X_2 = 0C in hexadecimal mode**
6. **Execute the above program and find the results.**
7. **How can modify above program if the two number is stored into DT_4.**

Home Work:

1. **What are the results of the program below, explain the program flow**

```
.CODE
MOV AX, 5
MOV BX, 3
JMP CALC
BACK: JMP STOP
CALC:
ADD AX, BX
JMP BACK
STOP:
RET
```

2. **You have DT_1 = 1,5,7,2,10,4,8,4, write 8086 program to find the largest and smallest number of the DT_1 store the results (Largest) into DT_2, (Smallest) into DT_3.**
3. **Write 8086 program to compare between two number (X, Y) that stored in the stack, if the X is grater than Y store 1 in DT_1, if X is equal Y store 2 in DT_1 otherwise store 3 in DT_3.**
4. **Write a short note on:**

   JZ label, JE label
5. **Write a program to convert any small character to capital character stored in the location DT_1. Store the results into DT_2**

   DT1 = AaBbCcDd