Lab # 9

# OBJECTive

Studying Transfer of control instructions (Conditional & Un-Conditional jumps).

# Theory

**Transfer of Control in Assembly Language:**

Control statements enable us to specify the flow of program control; i.e., the order in which the instructions in a program must be executed. They make it possible to make decisions, to perform tasks repeatedly or to jump from one section of code to another.

Flow of control through any given function is implemented with three basic types of control structures:

* Sequential: default mode. Sequential execution of code statements (one line after another)
* Selection: used for decisions, branching -- choosing between 2 or more alternative paths.
* Repetition: used for looping, i.e. repeating a piece of code multiple times in a row.

**Selection Control Structure:**

**Jump instructions** are used to change the flow of an Assembly language program. Jump instructions are two types:

### Unconditional Jump

The JMP instruction, whose syntax is:

JMP target

Transfers unconditionally control to the target location.

### Conditional Jump-Instructions

Conditional jumps are of the general form:

J***condition*** Statement Label

Where (i) ***condition*** is one, two, or three letters (ii) the Statement Label must in the current code segment and within 128 to +127 bytes from the conditional jump instruction.

#### How the CPU implements a conditional jump

Except for the JCXZ (Jump if the value in the CX register is zero) instruction, every conditional jump instruction must follow a status-flag modifying instruction, either immediately. It is the settings of the flags by this status-flag modifying instruction to which the conditional jump reacts to.

When a conditional jump is executed, the CPU checks the flags register. If the conditions for the jump (expressed as one or more status flag settings) are true, the CPU adjusts the IP register to point to the destination label, so that the instruction at this label will be executed next. If the jump condition is false, then the IP register is not altered; this means that the next sequential instruction will be executed.

**Note**: The conditional jump instructions DO NOT MODIFY the flags; they only react to the current flag values.

Example: **. . .**

SUB AX, BX

JZ L2 ; jump to L2 if the result is zero

**.**

**.**

L2**:**

### THE CMP (Compare) INSTRUCTION

The CMP instruction is used to compare either two signed numbers or two unsigned numbers.

* A signed number can be ***G***reater, ***L***ess, or ***E***qual to another signed number.
* An unsigned number can be ***A***bove, ***B***elow, or ***E***qual to another unsigned number.

The CMP instruction, whose syntax is:

CMP Operand1, Operand2

Compares two operands, and then sets or clears the following flags: AF, CF, OF, PF, and ZF. The instruction performs the subtraction:

Operand1 - Operand2

Without modifying any of its operands.

Note:

* The two operands must be of the same size.
* Both operands may not be memory locations at the same time.
* No operand may be a segment register.
* Operand1 may not be an immediate value.

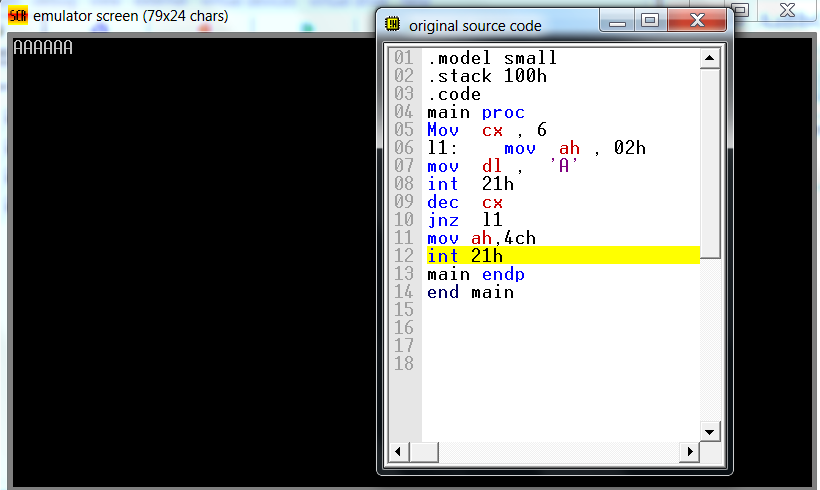
Conditional jumps can be classified into three: (1) Signed jumps, (2) Unsigned jumps, and (3) Single flag jumps.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **SIGNED JUMPS** | | **UNSIGNED JUMPS** | |
| condition | Equivalent condition | mnemonic | jump condition | mnemonic | jump condition |
| > | not ≤ | JG , JNLE | ZF = 0 and SF = OF | JA , JNBE | CF = 0 and ZF = 0 |
| ≥ | not < | JGE , JNL | SF = OF | JAE , JNB | CF = 0 |
| < | not ≥ | JL , JNGE | SF ≠ OF | JB , JNAE | CF = 1 |
| ≤ | not > | JLE , JNG | ZF = 1 or SF ≠ OF | JBE , JNA | CF = 1 or ZF = 1 |

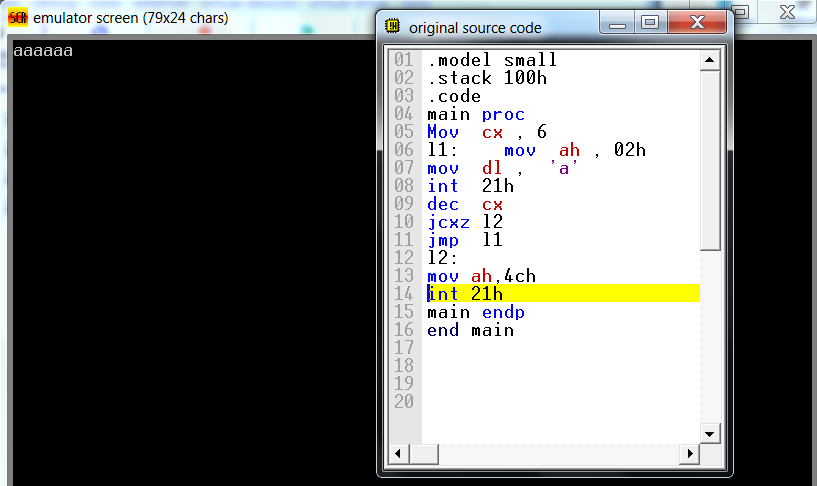
**Single flag jumps:**

| mnemonic | jump condition | description |
| --- | --- | --- |
| JE , JZ | ZF = 1 | Jump if equal |
| JNE , JNZ | ZF = 0 | Jump if not equal |
| JC | CF = 1 | Jump if carry |
| JNC | CF = 0 | Jump if no carry |
| JO | OF = 1 | Jump if overflow |
| JNO | OF = 0 | Jump if no overflow |
| JS | SF = 1 | Jump if sign negative |
| JNS | SF = 0 | Jump if sign is not negative |
| JP, JPE | PF = 1 | Jump if parity even, i.e., if there is an even number of 1 bits in the result. |
| JNP, JPO | PF = 0 | Jump if parity odd, i.e., if there is an odd number of 1 bits in the result. |

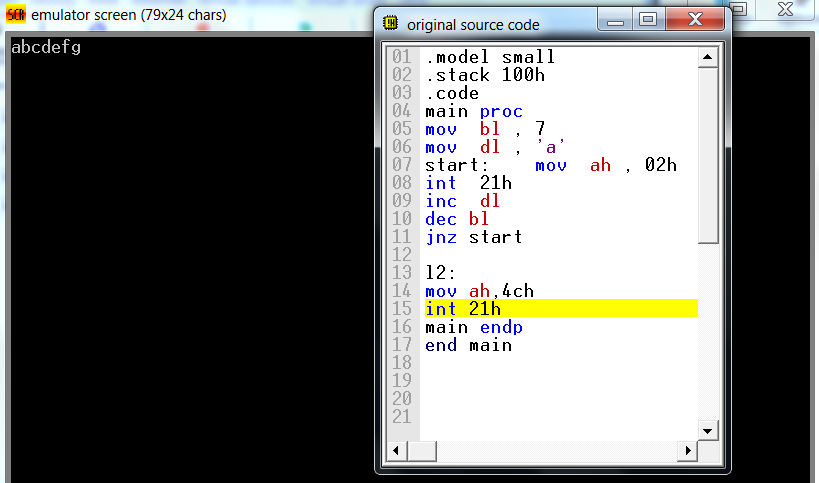
**Example**: Write a loop to display: AAAAAA



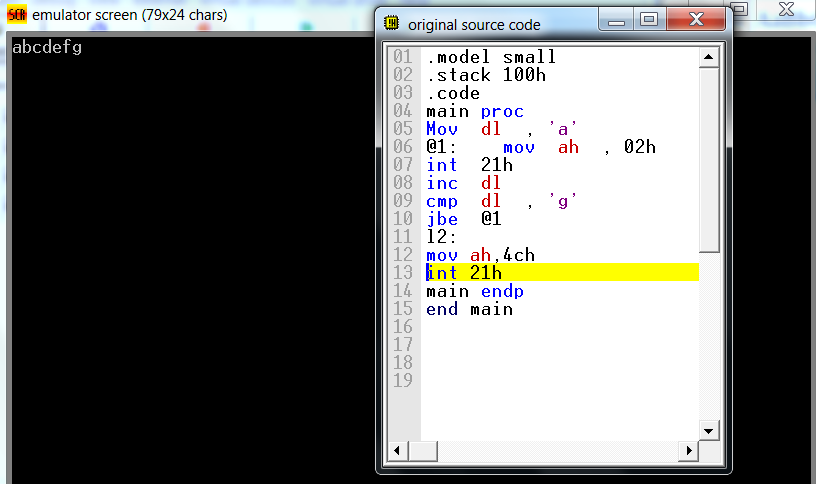
**an alternative solution is:** Write a loop to display: aaaaaa

****

**Example**: Write a loop to display: abcdefg



**an alternative solution is:**

****

**EXERCISE:**

**Task#1**: Display ‘\*’ 10 times each in new line (Use NEW\_LINE Macro)

**Task#2**: Display ASCII characters from A-Z and Z-A

**Task#3:** Display ASCII characters from a-z and z-a.

**Task#4:** Display ASCII characters from A-Z and a-z each on new line.