Lecture 4: Branching and Looping

Table of contents

0.1 Objectives

1 Branching

- By default, the CPU loads and executes programs sequentially.
- A branch, or transfer of control, is a way of altering the order in which statements are executed.

- There are two basic types of transfer:

    1. **Unconditional transfer**, in which a transfer is occurred unconditionally.
    2. **Conditional transfer**, in which a transfer is occurred based on a certain condition.

1.1 Instruction Pointer

- The EIP, or instruction pointer , register contains the address of the next instruction to be executed. Certain machine instructions manipulate EIP, causing the program to *branch* to a new location. These instructions are JMP, Jxx, LOOPxx, CALL, RET, or IRET.

1.2 `JMP` Instruction

Syntax

```
JMP    <destination>
```

- The JMP instruction causes an unconditional transfer to a destination address.
- The destination (target) operand specifies the address of the instruction being jumped to. This operand can be an immediate value, a general-purpose register, or a memory location.

    In this course, we only consider the immediate value for the destination operand.

1.2.1 Relative vs Absolute Offset Address

- The destination address, within the instruction stream, can be **relative offset** or **absolute offset**
- A **relative offset** is a signed *displacement* to the current value of the EIP.

$$\text{Offset Address} = \text{EIP} + \text{DEST}$$

- An **absolute address** is an offset from the base of the code segment (i.,e., offset from address 0 of a segment).

$$\text{Offset Address} = \text{DEST}$$

- Relative address can be specified in either of the following ways

Address format	Description
`rel8`	A constant value in the range from -128 to 127 bytes.
`rel16`	A relative address within the same code segment.
`rel32`	A relative address within the same code segment.

  - A relative offset (`rel8`, `rel16`, or `rel32`) is generally specified as a label in assembly code, but at the machine code level, it is encoded as a signed 8-, 16-, or 32-bit immediate value.

- Absolute address can be specified as an address in general-purpose register or an address specified using memory addressing mode.
- Thus, the JMP instruction takes the following types (for this course):

```
JMP   rel8          ; relative address
JMP   rel16         ; relative address (not supported in 64-b
JMP   rel32         ; relative address
JMP   reg/mem16     ; absolute address (not supported in 64-b
JMP   reg/mem32     ; aboslute address (not supported in 64-b
```

1.2.2 Types of Jumps:

1. **Near Jump**: A jump to instruction within the current code segment (intra-segment jump). Here the displacement is either `rel16`, `rel32`, `reg/mem16`, `reg/mem32`.
2. **Short jump**: A near jump where the jump is limited to -128 to +127 from the current EIP value. Here the displacement is `rel8`.
3. **Far Jump**: A jump to an instruction located in a different segment than the current code segment but at the same privilege level (inter-segment jump) (NOT covered).

   There is no difference in the coding for a relative short jump and for a relative near jump. The assembler uses a short jump if the displacement is within the small range in order to generate more compact code. A near jump is used automatically if the displacement is more than 128 bytes away.

1.2.3 Example

```
      mov   ax, 0
top:
      mov   bx, 5
      add   bx, 1
      .
      .
      .
      jmp   top
```

1.3 LOOP Instruction

Syntax

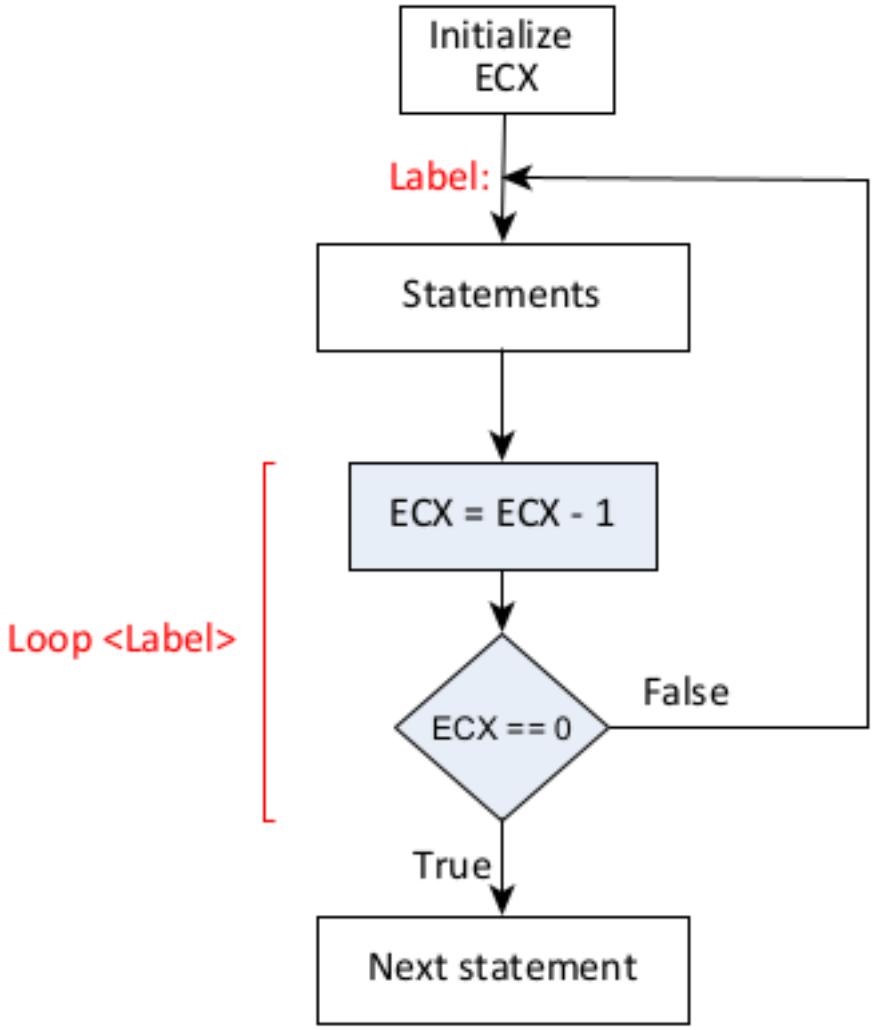
```
        LOOP    <destination>
```

Rule

- The destination operand must be rel8.

- The loop instruction repeats a block of statements a specific number of times (determined by ECX register).
- ECX is automatically used as a counter and is decremented each time the loop repeats.
- Example

```
      mov   ax, 1
      mov   ecx, 5
top:
      add   ax, ax
      loop  top
```


Initialize ECX
Label:
Statements
ECX = ECX - 1
Loop <Label>
ECX == 0
False
True
Next statement