Relative Offsets of Conditional Jumps

Conditional jumps use a *relative offset* to locate their jump targets. In 32-bit mode, a 32-bit signed relative offset is used if the jump is more than 127 bytes before or 127 bytes beyond the current location counter. (The *current location counter* is the address of the instruction following the current one because the CPU increments the instruction pointer prior to executing the current instruction.) In the following example, label L1 is located 189 bytes (000000BDh) beyond the location counter, so the target address field is 32 bits. The opcode is 0F 84:

```
Offset      Encoding          ASM Source Code
00000000    0F 84 000000BD    jz L1
```

On the other hand, if the jump is to a location within the range −128 to +127 bytes from the current location counter, a single-byte (signed) offset is encoded in the instruction operand. In the following example, the JZ instruction at offset 0 is encoded as **74 03**. The opcode is 74 and the relative offset is 03. (NOP stands for the no-operation instruction.) The address following JZ is 00000002, so the CPU adds 00000003 to 00000002, producing 00000005 (the offset of label L2):

```
Offset      Encoding    ASM Source Code
00000000          74 03       jz L2
00000002          90          nop
00000003          90          nop
00000004          90          nop
00000005                      L2:
```

Similarly, the next example shows a backward jump (negative offset). The offset after the jump is 00000005, so 0FBh (−5) is added to 00000005, producing offset 0 (the offset of label L1):

```
00000000    L1:
00000000    90          nop
00000001    90          nop
00000002    90          nop
00000003    74 FB       jz L1
00000005
```

16-Bit Mode

In 16-bit mode, the jump target must be within −128 to +127 bytes from the current location counter. The same limitation on ranges applies to the LOOP, LOOPZ, and LOOPNZ instructions. If a jump in a 16-bit mode program exceeds the range permitted by a signed byte offset, MASM generates a *relative jump out of range* error. Assuming that instructions have an average length of 3 bytes, you can put approximately 40 instructions inside a loop before encountering an error. To avoid the error, jump to an unconditional jump instruction (which has a 16-bit range).