target offset of the JMP instruction is ignored, and the new values loaded into CS and rIP are taken from the call gate or from the TSS.

Conditional Jump

- J*cc*—Jump if *condition*

Conditional jump instructions jump to an instruction specified by the operand, depending on the state of flags in the rFLAGS register. The operands specifies a signed relative offset from the current contents of the rIP. If the state of the corresponding flags meets the condition, a conditional jump instruction passes control to the target instruction, otherwise control is passed to the instruction following the conditional jump instruction. The flags tested by a specific J*cc* instruction depend on the opcode. In several cases, multiple mnemonics correspond to one opcode.

Table 3-6 shows the rFLAGS values required for each J*cc* instruction.

Table 3-6. rFLAGS for J*cc* Instructions

Mnemonic	Required Flag State	Description
JO	OF = 1	Jump near if overflow
JNO	OF = 0	Jump near if not overflow
JB JC JNAE	CF = 1	Jump near if below Jump near if carry Jump near if not above or equal
JNB JNC JAE	CF = 0	Jump near if not below Jump near if not carry Jump near if above or equal
JZ JE	ZF = 1	Jump near if 0 Jump near if equal
JNZ JNE	ZF = 0	Jump near if not zero Jump near if not equal
JNA JBE	CF = 1 or ZF = 1	Jump near if not above Jump near if below or equal
JNBE JA	CF = 0 and ZF = 0	Jump near if not below or equal Jump near if above
JS	SF = 1	Jump near if sign
JNS	SF = 0	Jump near if not sign
JP JPE	PF = 1	Jump near if parity Jump near if parity even
JNP JPO	PF = 0	Jump near if not parity Jump near if parity odd
JL JNGE	SF <> OF	Jump near if less Jump near if not greater or equal

Table 3-6. rFLAGS for J*cc* Instructions (continued)

Mnemonic	Required Flag State	Description
JGE JNL	SF = OF	Jump near if greater or equal Jump near if not less
JNG JLE	ZF = 1 or SF <> OF	Jump near if not greater Jump near if less or equal
JNLE JG	ZF = 0 and SF = OF	Jump near if not less or equal Jump near if greater

Unlike the unconditional jump (JMP), conditional jump instructions have only two forms—*near conditional jumps* and *short conditional jumps*. To create a far-conditional-jump code sequence corresponding to a high-level language statement like:

```
IF A = B THEN GOTO FarLabel
```

where FarLabel is located in another code segment, use the opposite condition in a conditional short jump before the unconditional far jump. For example:

```
      cmp   A,B               ; compare operands
      jne   NextInstr         ; continue program if not equal
      jmp   far ptr WhenNE    ; far jump if operands are equal
NextInstr:                    ; continue program
```

Three special conditional jump instructions use the rCX register instead of flags. The JCXZ, JECXZ, and JRCXZ instructions check the value of the CX, ECX, and RCX registers, respectively, and pass control to the target instruction when the value of rCX register reaches 0. These instructions are often used to control safe cycles, preventing execution when the value in rCX reaches 0.

Loop

- LOOP*cc*—Loop if *condition*

The LOOP*cc* instructions include LOOPE, LOOPNE, LOOPNZ, and LOOPZ. These instructions decrement the rCX register by 1 without changing any flags, and then check to see if the loop condition is met. If the condition is met, the program jumps to the specified target code.

LOOPE and LOOPZ are synonyms. Their loop condition is met if the value of the rCX register is non-zero and the zero flag (ZF) is set to 1 when the instruction starts. LOOPNE and LOOPNZ are also synonyms. Their loop condition is met if the value of the rCX register is non-zero and the ZF flag is cleared to 0 when the instruction starts. LOOP, unlike the other mnemonics, does not check the ZF flag. Its loop condition is met if the value of the rCX register is non-zero.

Call

- CALL—Procedure Call

The CALL instruction performs a call to a procedure whose address is specified in the operand. The return address is placed on the stack by the CALL, and points to the instruction immediately following