

**Jcc—Jump if Condition Is Met**

<b>Opcode</b>	<b>Instruction</b>	<b>Description</b>
77 <i>cb</i>	JA <i>rel8</i>	Jump short if above (CF=0 and ZF=0)
73 <i>cb</i>	JAE <i>rel8</i>	Jump short if above or equal (CF=0)
72 <i>cb</i>	JB <i>rel8</i>	Jump short if below (CF=1)
76 <i>cb</i>	JBE <i>rel8</i>	Jump short if below or equal (CF=1 or ZF=1)
72 <i>cb</i>	JC <i>rel8</i>	Jump short if carry (CF=1)
E3 <i>cb</i>	JCXZ <i>rel8</i>	Jump short if CX register is 0
E3 <i>cb</i>	JECXZ <i>rel8</i>	Jump short if ECX register is 0
74 <i>cb</i>	JE <i>rel8</i>	Jump short if equal (ZF=1)
7F <i>cb</i>	JG <i>rel8</i>	Jump short if greater (ZF=0 and SF=OF)
7D <i>cb</i>	JGE <i>rel8</i>	Jump short if greater or equal (SF=OF)
7C <i>cb</i>	JL <i>rel8</i>	Jump short if less (SF<>OF)
7E <i>cb</i>	JLE <i>rel8</i>	Jump short if less or equal (ZF=1 or SF<>OF)
76 <i>cb</i>	JNA <i>rel8</i>	Jump short if not above (CF=1 or ZF=1)
72 <i>cb</i>	JNAE <i>rel8</i>	Jump short if not above or equal (CF=1)
73 <i>cb</i>	JNB <i>rel8</i>	Jump short if not below (CF=0)
77 <i>cb</i>	JNBE <i>rel8</i>	Jump short if not below or equal (CF=0 and ZF=0)
73 <i>cb</i>	JNC <i>rel8</i>	Jump short if not carry (CF=0)
75 <i>cb</i>	JNE <i>rel8</i>	Jump short if not equal (ZF=0)
7E <i>cb</i>	JNG <i>rel8</i>	Jump short if not greater (ZF=1 or SF<>OF)
7C <i>cb</i>	JNGE <i>rel8</i>	Jump short if not greater or equal (SF<>OF)
7D <i>cb</i>	JNL <i>rel8</i>	Jump short if not less (SF=OF)
7F <i>cb</i>	JNLE <i>rel8</i>	Jump short if not less or equal (ZF=0 and SF=OF)
71 <i>cb</i>	JNO <i>rel8</i>	Jump short if not overflow (OF=0)
7B <i>cb</i>	JNP <i>rel8</i>	Jump short if not parity (PF=0)
79 <i>cb</i>	JNS <i>rel8</i>	Jump short if not sign (SF=0)
75 <i>cb</i>	JNZ <i>rel8</i>	Jump short if not zero (ZF=0)
70 <i>cb</i>	JO <i>rel8</i>	Jump short if overflow (OF=1)
7A <i>cb</i>	JP <i>rel8</i>	Jump short if parity (PF=1)
7A <i>cb</i>	JPE <i>rel8</i>	Jump short if parity even (PF=1)
7B <i>cb</i>	JPO <i>rel8</i>	Jump short if parity odd (PF=0)
78 <i>cb</i>	JS <i>rel8</i>	Jump short if sign (SF=1)
74 <i>cb</i>	JZ <i>rel8</i>	Jump short if zero (ZF = 1)
0F 87 <i>cw/cd</i>	JA <i>rel16/32</i>	Jump near if above (CF=0 and ZF=0)
0F 83 <i>cw/cd</i>	JAE <i>rel16/32</i>	Jump near if above or equal (CF=0)
0F 82 <i>cw/cd</i>	JB <i>rel16/32</i>	Jump near if below (CF=1)
0F 86 <i>cw/cd</i>	JBE <i>rel16/32</i>	Jump near if below or equal (CF=1 or ZF=1)
0F 82 <i>cw/cd</i>	JC <i>rel16/32</i>	Jump near if carry (CF=1)
0F 84 <i>cw/cd</i>	JE <i>rel16/32</i>	Jump near if equal (ZF=1)
0F 84 <i>cw/cd</i>	JZ <i>rel16/32</i>	Jump near if 0 (ZF=1)
0F 8F <i>cw/cd</i>	JG <i>rel16/32</i>	Jump near if greater (ZF=0 and SF=OF)

## Jcc—Jump if Condition Is Met (Continued)

Opcode	Instruction	Description
0F 8D <i>cw/cd</i>	JGE <i>rel16/32</i>	Jump near if greater or equal (SF=OF)
0F 8C <i>cw/cd</i>	JL <i>rel16/32</i>	Jump near if less (SF<>OF)
0F 8E <i>cw/cd</i>	JLE <i>rel16/32</i>	Jump near if less or equal (ZF=1 or SF<>OF)
0F 86 <i>cw/cd</i>	JNA <i>rel16/32</i>	Jump near if not above (CF=1 or ZF=1)
0F 82 <i>cw/cd</i>	JNAE <i>rel16/32</i>	Jump near if not above or equal (CF=1)
0F 83 <i>cw/cd</i>	JNB <i>rel16/32</i>	Jump near if not below (CF=0)
0F 87 <i>cw/cd</i>	JNBE <i>rel16/32</i>	Jump near if not below or equal (CF=0 and ZF=0)
0F 83 <i>cw/cd</i>	JNC <i>rel16/32</i>	Jump near if not carry (CF=0)
0F 85 <i>cw/cd</i>	JNE <i>rel16/32</i>	Jump near if not equal (ZF=0)
0F 8E <i>cw/cd</i>	JNG <i>rel16/32</i>	Jump near if not greater (ZF=1 or SF<>OF)
0F 8C <i>cw/cd</i>	JNGE <i>rel16/32</i>	Jump near if not greater or equal (SF<>OF)
0F 8D <i>cw/cd</i>	JNL <i>rel16/32</i>	Jump near if not less (SF=OF)
0F 8F <i>cw/cd</i>	JNLE <i>rel16/32</i>	Jump near if not less or equal (ZF=0 and SF=OF)
0F 81 <i>cw/cd</i>	JNO <i>rel16/32</i>	Jump near if not overflow (OF=0)
0F 8B <i>cw/cd</i>	JNP <i>rel16/32</i>	Jump near if not parity (PF=0)
0F 89 <i>cw/cd</i>	JNS <i>rel16/32</i>	Jump near if not sign (SF=0)
0F 85 <i>cw/cd</i>	JNZ <i>rel16/32</i>	Jump near if not zero (ZF=0)
0F 80 <i>cw/cd</i>	JO <i>rel16/32</i>	Jump near if overflow (OF=1)
0F 8A <i>cw/cd</i>	JP <i>rel16/32</i>	Jump near if parity (PF=1)
0F 8A <i>cw/cd</i>	JPE <i>rel16/32</i>	Jump near if parity even (PF=1)
0F 8B <i>cw/cd</i>	JPO <i>rel16/32</i>	Jump near if parity odd (PF=0)
0F 88 <i>cw/cd</i>	JS <i>rel16/32</i>	Jump near if sign (SF=1)
0F 84 <i>cw/cd</i>	JZ <i>rel16/32</i>	Jump near if 0 (ZF=1)

### Description

Checks the state of one or more of the status flags in the EFLAGS register (CF, OF, PF, SF, and ZF) and, if the flags are in the specified state (condition), performs a jump to the target instruction specified by the destination operand. A condition code (*cc*) is associated with each instruction to indicate the condition being tested for. If the condition is not satisfied, the jump is not performed and execution continues with the instruction following the *Jcc* instruction.

The target instruction is specified with a relative offset (a signed offset relative to the current value of the instruction pointer in the EIP register). 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-bit or 32-bit immediate value, which is added to the instruction pointer. Instruction coding is most efficient for offsets of –128 to +127. If the operand-size attribute is 16, the upper two bytes of the EIP register are cleared, resulting in a maximum instruction pointer size of 16 bits.

## Jcc—Jump if Condition Is Met (Continued)

The conditions for each *Jcc* mnemonic are given in the “Description” column of the table on the preceding page. The terms “less” and “greater” are used for comparisons of signed integers and the terms “above” and “below” are used for unsigned integers.

Because a particular state of the status flags can sometimes be interpreted in two ways, two mnemonics are defined for some opcodes. For example, the JA (jump if above) instruction and the JNBE (jump if not below or equal) instruction are alternate mnemonics for the opcode 77H.

The *Jcc* instruction does not support far jumps (jumps to other code segments). When the target for the conditional jump is in a different segment, use the opposite condition from the condition being tested for the *Jcc* instruction, and then access the target with an unconditional far jump (JMP instruction) to the other segment. For example, the following conditional far jump is illegal:

```
JZ FARLABEL;
```

To accomplish this far jump, use the following two instructions:

```
JNZ BEYOND;
JMP FARLABEL;
BEYOND:
```

The JECXZ and JCXZ instructions differs from the other *Jcc* instructions because they do not check the status flags. Instead they check the contents of the ECX and CX registers, respectively, for 0. Either the CX or ECX register is chosen according to the address-size attribute. These instructions are useful at the beginning of a conditional loop that terminates with a conditional loop instruction (such as LOOPNE). They prevent entering the loop when the ECX or CX register is equal to 0, which would cause the loop to execute  $2^{32}$  or 64K times, respectively, instead of zero times.

All conditional jumps are converted to code fetches of one or two cache lines, regardless of jump address or cacheability.

### Operation

```
IF condition
  THEN
    EIP ← EIP + SignExtend(DEST);
    IF OperandSize = 16
      THEN
        EIP ← EIP AND 0000FFFFH;
    FI;
    ELSE (* OperandSize = 32 *)
      IF EIP < CS.Base OR EIP > CS.Limit
        #GP
    FI;
  FI;
```

## Jcc—Jump if Condition Is Met (Continued)

### Flags Affected

None.

### Protected Mode Exceptions

#GP(0)                    If the offset being jumped to is beyond the limits of the CS segment.

### Real-Address Mode Exceptions

#GP                      If the offset being jumped to is beyond the limits of the CS segment or is outside of the effective address space from 0 to FFFFH. This condition can occur if a 32-bit address size override prefix is used.

### Virtual-8086 Mode Exceptions

Same exceptions as in Real Address Mode