J*cc*—Jump if Condition Is Met

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
77 *cb*	JA *rel8*	Valid	Valid	Jump short if above (CF=0 and ZF=0).
73 *cb*	JAE *rel8*	Valid	Valid	Jump short if above or equal (CF=0).
72 *cb*	JB *rel8*	Valid	Valid	Jump short if below (CF=1).
76 *cb*	JBE *rel8*	Valid	Valid	Jump short if below or equal (CF=1 or ZF=1).
72 *cb*	JC *rel8*	Valid	Valid	Jump short if carry (CF=1).
E3 *cb*	JCXZ *rel8*	N.E.	Valid	Jump short if CX register is 0.
E3 *cb*	JECXZ *rel8*	Valid	Valid	Jump short if ECX register is 0.
E3 *cb*	JRCXZ *rel8*	Valid	N.E.	Jump short if RCX register is 0.
74 *cb*	JE *rel8*	Valid	Valid	Jump short if equal (ZF=1).
7F *cb*	JG *rel8*	Valid	Valid	Jump short if greater (ZF=0 and SF=OF).
7D *cb*	JGE *rel8*	Valid	Valid	Jump short if greater or equal (SF=OF).
7C *cb*	JL *rel8*	Valid	Valid	Jump short if less (SF≠ OF).
7E *cb*	JLE *rel8*	Valid	Valid	Jump short if less or equal (ZF=1 or SF≠ OF).
76 *cb*	JNA *rel8*	Valid	Valid	Jump short if not above (CF=1 or ZF=1).
72 *cb*	JNAE *rel8*	Valid	Valid	Jump short if not above or equal (CF=1).
73 *cb*	JNB *rel8*	Valid	Valid	Jump short if not below (CF=0).
77 *cb*	JNBE *rel8*	Valid	Valid	Jump short if not below or equal (CF=0 and ZF=0).
73 *cb*	JNC *rel8*	Valid	Valid	Jump short if not carry (CF=0).
75 *cb*	JNE *rel8*	Valid	Valid	Jump short if not equal (ZF=0).
7E *cb*	JNG *rel8*	Valid	Valid	Jump short if not greater (ZF=1 or SF≠ OF).
7C *cb*	JNGE *rel8*	Valid	Valid	Jump short if not greater or equal (SF≠ OF).
7D *cb*	JNL *rel8*	Valid	Valid	Jump short if not less (SF=OF).
7F *cb*	JNLE *rel8*	Valid	Valid	Jump short if not less or equal (ZF=0 and SF=OF).
71 *cb*	JNO *rel8*	Valid	Valid	Jump short if not overflow (OF=0).
7B *cb*	JNP *rel8*	Valid	Valid	Jump short if not parity (PF=0).
79 *cb*	JNS *rel8*	Valid	Valid	Jump short if not sign (SF=0).

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
75 *cb*	JNZ *rel8*	Valid	Valid	Jump short if not zero (ZF=0).
70 *cb*	JO *rel8*	Valid	Valid	Jump short if overflow (OF=1).
7A *cb*	JP *rel8*	Valid	Valid	Jump short if parity (PF=1).
7A *cb*	JPE *rel8*	Valid	Valid	Jump short if parity even (PF=1).
7B *cb*	JPO *rel8*	Valid	Valid	Jump short if parity odd (PF=0).
78 *cb*	JS *rel8*	Valid	Valid	Jump short if sign (SF=1).
74 *cb*	JZ *rel8*	Valid	Valid	Jump short if zero (ZF ← 1).
0F 87 *cw*	JA *rel16*	N.S.	Valid	Jump near if above (CF=0 and ZF=0). Not supported in 64-bit mode.
0F 87 *cd*	JA *rel32*	Valid	Valid	Jump near if above (CF=0 and ZF=0).
0F 83 *cw*	JAE *rel16*	N.S.	Valid	Jump near if above or equal (CF=0). Not supported in 64-bit mode.
0F 83 *cd*	JAE *rel32*	Valid	Valid	Jump near if above or equal (CF=0).
0F 82 *cw*	JB *rel16*	N.S.	Valid	Jump near if below (CF=1). Not supported in 64-bit mode.
0F 82 *cd*	JB *rel32*	Valid	Valid	Jump near if below (CF=1).
0F 86 *cw*	JBE *rel16*	N.S.	Valid	Jump near if below or equal (CF=1 or ZF=1). Not supported in 64-bit mode.
0F 86 *cd*	JBE *rel32*	Valid	Valid	Jump near if below or equal (CF=1 or ZF=1).
0F 82 *cw*	JC *rel16*	N.S.	Valid	Jump near if carry (CF=1). Not supported in 64-bit mode.
0F 82 *cd*	JC *rel32*	Valid	Valid	Jump near if carry (CF=1).
0F 84 *cw*	JE *rel16*	N.S.	Valid	Jump near if equal (ZF=1). Not supported in 64-bit mode.
0F 84 *cd*	JE *rel32*	Valid	Valid	Jump near if equal (ZF=1).
0F 84 *cw*	JZ *rel16*	N.S.	Valid	Jump near if 0 (ZF=1). Not supported in 64-bit mode.
0F 84 *cd*	JZ *rel32*	Valid	Valid	Jump near if 0 (ZF=1).
0F 8F *cw*	JG *rel16*	N.S.	Valid	Jump near if greater (ZF=0 and SF=OF). Not supported in 64-bit mode.
0F 8F *cd*	JG *rel32*	Valid	Valid	Jump near if greater (ZF=0 and SF=OF).
0F 8D *cw*	JGE *rel16*	N.S.	Valid	Jump near if greater or equal (SF=OF). Not supported in 64-bit mode.
0F 8D *cd*	JGE *rel32*	Valid	Valid	Jump near if greater or equal (SF=OF).

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
0F 8C *cw*	JL *rel16*	N.S.	Valid	Jump near if less (SF≠ OF). Not supported in 64-bit mode.
0F 8C *cd*	JL *rel32*	Valid	Valid	Jump near if less (SF≠ OF).
0F 8E *cw*	JLE *rel16*	N.S.	Valid	Jump near if less or equal (ZF=1 or SF≠ OF). Not supported in 64-bit mode.
0F 8E *cd*	JLE *rel32*	Valid	Valid	Jump near if less or equal (ZF=1 or SF≠ OF).
0F 86 *cw*	JNA *rel16*	N.S.	Valid	Jump near if not above (CF=1 or ZF=1). Not supported in 64-bit mode.
0F 86 *cd*	JNA *rel32*	Valid	Valid	Jump near if not above (CF=1 or ZF=1).
0F 82 *cw*	JNAE *rel16*	N.S.	Valid	Jump near if not above or equal (CF=1). Not supported in 64-bit mode.
0F 82 *cd*	JNAE *rel32*	Valid	Valid	Jump near if not above or equal (CF=1).
0F 83 *cw*	JNB *rel16*	N.S.	Valid	Jump near if not below (CF=0). Not supported in 64-bit mode.
0F 83 *cd*	JNB *rel32*	Valid	Valid	Jump near if not below (CF=0).
0F 87 *cw*	JNBE *rel16*	N.S.	Valid	Jump near if not below or equal (CF=0 and ZF=0). Not supported in 64-bit mode.
0F 87 *cd*	JNBE *rel32*	Valid	Valid	Jump near if not below or equal (CF=0 and ZF=0).
0F 83 *cw*	JNC *rel16*	N.S.	Valid	Jump near if not carry (CF=0). Not supported in 64-bit mode.
0F 83 *cd*	JNC *rel32*	Valid	Valid	Jump near if not carry (CF=0).
0F 85 *cw*	JNE *rel16*	N.S.	Valid	Jump near if not equal (ZF=0). Not supported in 64-bit mode.
0F 85 *cd*	JNE *rel32*	Valid	Valid	Jump near if not equal (ZF=0).
0F 8E *cw*	JNG *rel16*	N.S.	Valid	Jump near if not greater (ZF=1 or SF≠ OF). Not supported in 64-bit mode.
0F 8E *cd*	JNG *rel32*	Valid	Valid	Jump near if not greater (ZF=1 or SF≠ OF).
0F 8C *cw*	JNGE *rel16*	N.S.	Valid	Jump near if not greater or equal (SF≠ OF). Not supported in 64-bit mode.

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
0F 8C *cd*	JNGE *rel32*	Valid	Valid	Jump near if not greater or equal (SF≠ OF).
0F 8D *cw*	JNL *rel16*	N.S.	Valid	Jump near if not less (SF=OF). Not supported in 64-bit mode.
0F 8D *cd*	JNL *rel32*	Valid	Valid	Jump near if not less (SF=OF).
0F 8F *cw*	JNLE *rel16*	N.S.	Valid	Jump near if not less or equal (ZF=0 and SF=OF). Not supported in 64-bit mode.
0F 8F *cd*	JNLE *rel32*	Valid	Valid	Jump near if not less or equal (ZF=0 and SF=OF).
0F 81 *cw*	JNO *rel16*	N.S.	Valid	Jump near if not overflow (OF=0). Not supported in 64-bit mode.
0F 81 *cd*	JNO *rel32*	Valid	Valid	Jump near if not overflow (OF=0).
0F 8B *cw*	JNP *rel16*	N.S.	Valid	Jump near if not parity (PF=0). Not supported in 64-bit mode.
0F 8B *cd*	JNP *rel32*	Valid	Valid	Jump near if not parity (PF=0).
0F 89 *cw*	JNS *rel16*	N.S.	Valid	Jump near if not sign (SF=0). Not supported in 64-bit mode.
0F 89 *cd*	JNS *rel32*	Valid	Valid	Jump near if not sign (SF=0).
0F 85 *cw*	JNZ *rel16*	N.S.	Valid	Jump near if not zero (ZF=0). Not supported in 64-bit mode.
0F 85 *cd*	JNZ *rel32*	Valid	Valid	Jump near if not zero (ZF=0).
0F 80 *cw*	JO *rel16*	N.S.	Valid	Jump near if overflow (OF=1). Not supported in 64-bit mode.
0F 80 *cd*	JO *rel32*	Valid	Valid	Jump near if overflow (OF=1).
0F 8A *cw*	JP *rel16*	N.S.	Valid	Jump near if parity (PF=1). Not supported in 64-bit mode.
0F 8A *cd*	JP *rel32*	Valid	Valid	Jump near if parity (PF=1).
0F 8A *cw*	JPE *rel16*	N.S.	Valid	Jump near if parity even (PF=1). Not supported in 64-bit mode.
0F 8A *cd*	JPE *rel32*	Valid	Valid	Jump near if parity even (PF=1).
0F 8B *cw*	JPO *rel16*	N.S.	Valid	Jump near if parity odd (PF=0). Not supported in 64-bit mode.
0F 8B *cd*	JPO *rel32*	Valid	Valid	Jump near if parity odd (PF=0).
0F 88 *cw*	JS *rel16*	N.S.	Valid	Jump near if sign (SF=1). Not supported in 64-bit mode.

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
0F 88 *cd*	JS *rel32*	Valid	Valid	Jump near if sign (SF=1).
0F 84 *cw*	JZ *rel16*	N.S.	Valid	Jump near if 0 (ZF=1). Not supported in 64-bit mode.
0F 84 *cd*	JZ *rel32*	Valid	Valid	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 J*cc* 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.

The conditions for each J*cc* 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 J*cc* 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 J*cc* 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 JRCXZ, JECXZ and JCXZ instructions differ from other J*cc* instructions because they do not check status flags. Instead, they check RCX, ECX or CX for 0. The register

checked is determined by the address-size attribute. These instructions are useful when used at the beginning of a loop that terminates with a conditional loop instruction (such as LOOPNE). They can be used to prevent an instruction sequence from entering a loop when RCX, ECX or CX is 0. This would cause the loop to execute 2^{64}, 2^{32} or 64K times (not zero times).

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

In 64-bit mode, operand size is fixed at 64 bits. JMP Short is RIP = RIP + 8-bit offset sign extended to 64 bits. JMP Near is RIP = RIP + 32-bit offset sign extended to 64-bits.

Operation

```
IF condition
    THEN
        tempEIP ← EIP + SignExtend(DEST);
        IF OperandSize = 16
            THEN tempEIP ← tempEIP AND 0000FFFFH;
        FI;
    IF tempEIP is not within code segment limit
        THEN #GP(0);
        ELSE EIP ← tempEIP
    FI;
FI;
```

Protected Mode Exceptions

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

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.
#UD	If the LOCK prefix is used.

Virtual-8086 Mode Exceptions

Same exceptions as in real address mode.