

BUILD A FREE WEBSITE
OF YOUR OWN ON

TRIPOD

[\[Up \]](#) [\[Docs Home \]](#)

JNZ - Jump if Condition Is Met

[\[Back \]](#) [\[Next \]](#)

[*](#) [A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#) [?](#)

Code	Mnemonic	Description
77 cb	JA rel8	Jump short if above (CF=0 and ZF=0)
73 cb	JAЕ rel8	Jump short if above or equal (CF=0)
72 cb	JB rel8	Jump short if below (CF=1)
76 cb	JBE rel8	Jump short if below or equal (CF=1 or ZF=1)
72 cb	JC rel8	Jump short if carry (CF=1)
E3 cb	JCXZ rel8	Jump short if CX register is 0
E3 cb	JECXZ rel8	Jump short if ECX register is 0
74 cb	JE rel8	Jump short if equal (ZF=1)
7F cb	JG rel8	Jump short if greater (ZF=0 and SF=OF)
7D cb	JGE rel8	Jump short if greater or equal (SF=OF)
7C cb	JL rel8	Jump short if less (SF<>OF)
7E cb	JLE rel8	Jump short if less or equal (ZF=1 or SF<>OF)
76 cb	JNA rel8	Jump short if not above (CF=1 or ZF=1)
72 cb	JNAЕ rel8	Jump short if not above or equal (CF=1)
73 cb	JNB rel8	Jump short if not below (CF=0)
77 cb	JNBE rel8	Jump short if not below or equal (CF=0 and ZF=0)
73 cb	JNC rel8	Jump short if not carry (CF=0)
75 cb	JNE rel8	Jump short if not equal (ZF=0)
7E cb	JNG rel8	Jump short if not greater (ZF=1 or SF<>OF)
7C cb	JNGЕ rel8	Jump short if not greater or equal (SF<>OF)
7D cb	JNL rel8	Jump short if not less (SF=OF)
7F cb	JNLE rel8	Jump short if not less or equal (ZF=0 and SF=OF)
71 cb	JNO rel8	Jump short if not overflow (OF=0)
7B cb	JNP rel8	Jump short if not parity (PF=0)
79 cb	JNS rel8	Jump short if not sign (SF=0)
75 cb	JNZ rel8	Jump short if not zero (ZF=0)
70 cb	JO rel8	Jump short if overflow (OF=1)
7A cb	JP rel8	Jump short if parity (PF=1)
7A cb	JPE rel8	Jump short if parity even (PF=1)
7B cb	JPO rel8	Jump short if parity odd (PF=0)

78 cb	JS rel8	Jump short if sign (SF=1)
74 cb	JZ rel8	Jump short if zero (ZF = 1)
0F 87 cw/cd	JA rel16/32	Jump near if above (CF=0 and ZF=0)
0F 83 cw/cd	JAЕ rel16/32	Jump near if above or equal (CF=0)
0F 82 cw/cd	JB rel16/32	Jump near if below (CF=1)
0F 86 cw/cd	JBE rel16/32	Jump near if below or equal (CF=1 or ZF=1)
0F 82 cw/cd	JC rel16/32	Jump near if carry (CF=1)
0F 84 cw/cd	JE rel16/32	Jump near if equal (ZF=1)
0F 84 cw/cd	JZ rel16/32	Jump near if 0 (ZF=1)
0F 8F cw/cd	JG rel16/32	Jump near if greater (ZF=0 and SF=OF)
0F 8D cw/cd	JGE rel16/32	Jump near if greater or equal (SF=OF)
0F 8C cw/cd	JL rel16/32	Jump near if less (SF<>OF)
0F 8E cw/cd	JLE rel16/32	Jump near if less or equal (ZF=1 or SF<>OF)
0F 86 cw/cd	JNA rel16/32	Jump near if not above (CF=1 or ZF=1)
0F 82 cw/cd	JNAЕ rel16/32	Jump near if not above or equal (CF=1)
0F 83 cw/cd	JNB rel16/32	Jump near if not below (CF=0)
0F 87 cw/cd	JNBE rel16/32	Jump near if not below or equal (CF=0 and ZF=0)
0F 83 cw/cd	JNC rel16/32	Jump near if not carry (CF=0)
0F 85 cw/cd	JNE rel16/32	Jump near if not equal (ZF=0)
0F 8E cw/cd	JNG rel16/32	Jump near if not greater (ZF=1 or SF<>OF)
0F 8C cw/cd	JNGЕ rel16/32	Jump near if not greater or equal (SF<>OF)
0F 8D cw/cd	JNL rel16/32	Jump near if not less (SF=OF)
0F 8F cw/cd	JNLE rel16/32	Jump near if not less or equal (ZF=0 and SF=OF)
0F 81 cw/cd	JNO rel16/32	Jump near if not overflow (OF=0)
0F 8B cw/cd	JNP rel16/32	Jump near if not parity (PF=0)
0F 89 cw/cd	JNS rel16/32	Jump near if not sign (SF=0)
0F 85 cw/cd	JNZ rel16/32	Jump near if not zero (ZF=0)
0F 80 cw/cd	JO rel16/32	Jump near if overflow (OF=1)
0F 8A cw/cd	JP rel16/32	Jump near if parity (PF=1)
0F 8A cw/cd	JPE rel16/32	Jump near if parity even (PF=1)
0F 8B cw/cd	JPO rel16/32	Jump near if parity odd (PF=0)
0F 88 cw/cd	JS rel16/32	Jump near if sign (SF=1)
0F 84 cw/cd	JZ rel16/32	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 to 0s, resulting in a maximum instruction pointer size of 16 bits.

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 2³² 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.

Operands	Bytes	Clocks	
near8	2	1	PV
near16	3	1	PV

cycles apply to jump and no jump

Flags

ID	<i>unaffected</i>	DF	<i>unaffected</i>
VIP	<i>unaffected</i>	IF	<i>unaffected</i>
VIF	<i>unaffected</i>	TF	<i>unaffected</i>
AC	<i>unaffected</i>	SF	<i>unaffected</i>
VM	<i>unaffected</i>	ZF	<i>unaffected</i>
RF	<i>unaffected</i>	AF	<i>unaffected</i>
NT	<i>unaffected</i>	PF	<i>unaffected</i>
IOPL	<i>unaffected</i>	CF	<i>unaffected</i>
OF	<i>unaffected</i>		



