have previously seen, a number of instructions go by more than one name. Such is the case for the unconditional jumps. The instructions, their required flag conditions, and their interpretations are as follows:

Instruction	*Flag Tested*	*Explanation*
JNC	CF = 0	jump if no carry
JAE	"	jump if above or equal
JNB	"	jump if not below
JC	CF = 1	jump if carry
JB	"	jump if below
JNAE	"	jump if not above or equal
JNZ	ZF = 0	jump if not zero
JNE	ZF = 0	jump if not equal
JZ	ZF = 1	jump if zero
JE	"	jump if equal
JNS	SF = 0	jump if no sign
JS	SF = 1	jump if sign
JNO	OF = 0	jump if no overflow
JO	OF = 1	jump if overflow
JNP	PF = 0	jump if no parity
JPO	"	jump if parity odd
JP	PF = 1	jump if parity
JPE	"	jump if parity even
JA	(CF or ZF) = 0	jump if above
JBNE	"	jump if not below or equal
JBE	(CF or ZF) = 1	jump if below or equal
JNA	"	jump if not above
JGE	(SF xor OF) = 0	jump if greater or equal
JNL	"	jump if not less
JL	(SF xor OF) = 1	jump if less
JNGE	"	jump if not greater or equal
JG	([SF xor OF] or ZF) = 0	jump if greater
JNLE	"	jump if not less or equal
JLE	([SF xor OF] or ZF) = 1	jump if less or equal
JNG	"	jump if not greater

Note that you must supply the target address in the jump instruction. Unlike JMP, we are not allowed to use register names or other fancy addressing mode operands. The nice thing about relative jumps is that they always jump to the correct location within the code segment, no matter where the code segment appears in memory.

■ **EXAMPLE 4.45** Let us look at a few ways conditional instructions may be used to test information.

Consider the following group of instructions, which determines if the AL register contains a lowercase ASCII letter.

```
CMP     AL,'a'
JB      NOTLOWER
CMP     AL,'z'+1
JNB     NOTLOWER
```

If AL does not contain the ASCII code of a lowercase letter (61H to 7AH for "a" or "z"), the processor will jump to NOTLOWER. If AL does contain a valid code, then neither of the conditional jumps will be taken, and the processor will execute the first instruction