

d1	d2	d3	Mnemonic	Destination (where to store the computed value)
0	0	0	null	The value is not stored anywhere
0	0	1	M	Memory[A] (memory register addressed by A)
0	1	0	D	D register
0	1	1	MD	Memory[A] and D register
1	0	0	A	A register
1	0	1	AM	A register and Memory[A]
1	1	0	AD	A register and D register
1	1	1	AMD	A register, Memory[A], and D register

Figure 4.4 The *dest* field of the *C*-instruction.

The first instruction causes the computer to select the memory register whose address is 7 (the so-called M register). The second instruction computes the value of $M + 1$ and stores the result in both M and D.

The Jump Specification The *jump* field of the *C*-instruction tells the computer what to do next. There are two possibilities: The computer should either fetch and execute the next instruction in the program, which is the default, or it should fetch and execute an instruction located elsewhere in the program. In the latter case, we assume that the A register has been previously set to the address to which we have to jump.

Whether or not a jump should actually materialize depends on the three *j*-bits of the *jump* field and on the ALU output value (computed according to the *comp* field). The first *j*-bit specifies whether to jump in case this value is negative, the second *j*-bit in case the value is zero, and the third *j*-bit in case it is positive. This gives eight possible jump conditions, shown in figure 4.5.

The following example illustrates the jump commands in action:

Logic

```
if Memory[3]=5 then goto 100
else goto 200
```

Implementation

```
@3
D=M      // D=Memory[3]
@5
D=D-A    // D=D-5
@100
D;JEQ    // If D=0 goto 100
@200
0;JMP    // Goto 200
```