

(when a=0) <i>comp mnemonic</i>	c1	c2	c3	c4	c5	c6	(when a=1) <i>comp mnemonic</i>
0	1	0	1	0	1	0	
1	1	1	1	1	1	1	
-1	1	1	1	0	1	0	
D	0	0	1	1	0	0	
A	1	1	0	0	0	0	M
!D	0	0	1	1	0	1	
!A	1	1	0	0	0	1	!M
-D	0	0	1	1	1	1	
-A	1	1	0	0	1	1	-M
D+1	0	1	1	1	1	1	
A+1	1	1	0	1	1	1	M+1
D-1	0	0	1	1	1	0	
A-1	1	1	0	0	1	0	M-1
D+A	0	0	0	0	1	0	D+M
D-A	0	1	0	0	1	1	D-M
A-D	0	0	0	1	1	1	M-D
D&A	0	0	0	0	0	0	D&M
D A	0	1	0	1	0	1	D M

Figure 4.3 The *compute* field of the C-instruction. D and A are names of registers. M refers to the memory location addressed by A, namely, to Memory[A]. The symbols + and - denote 16-bit 2's complement addition and subtraction, while !, |, and & denote the 16-bit bit-wise Boolean operators Not, Or, and And, respectively. Note the similarity between this instruction set and the ALU specification given in figure 2.6.

dest part (see figure 4.4). The first and second d-bits code whether to store the computed value in the A register and in the D register, respectively. The third d-bit codes whether to store the computed value in M (i.e., in Memory[A]). One, more than one, or none of these bits may be asserted.

Recall that the format of the C-instruction is 111a cccc ccdd djjj. Suppose we want the computer to increment the value of Memory[7] by 1 and to also store the result in the D register. According to figures 4.3 and 4.4, this can be accomplished by the following instructions:

```
0000 0000 0000 0111    // @7
1111 1101 1101 1000    // MD=M+1
```