j1 ($out < 0$)	j2 ($out = 0$)	j3 ($out > 0$)	Mnemonic	Effect
0	0	0	null	No jump
0	0	1	JGT	If $out > 0$ jump
0	1	0	JEQ	If $out = 0$ jump
0	1	1	JGE	If $out \geq 0$ jump
1	0	0	JLT	If $out < 0$ jump
1	0	1	JNE	If $out \neq 0$ jump
1	1	0	JLE	If $out \leq 0$ jump
1	1	1	JMP	Jump

Figure 4.5 The *jump* field of the *C*-instruction. *Out* refers to the ALU output (resulting from the instruction's *comp* part), and *jump* implies "continue execution with the instruction addressed by the A register."

The last instruction (`0;JMP`) effects an unconditional jump. Since the *C*-instruction syntax requires that we always effect *some* computation, we instruct the ALU to compute 0 (an arbitrary choice), which is ignored.

Conflicting Uses of the A Register As was just illustrated, the programmer can use the A register to select either a *data memory* location for a subsequent *C*-instruction involving M, or an *instruction memory* location for a subsequent *C*-instruction involving a jump. Thus, to prevent conflicting use of the A register, in well-written programs a *C*-instruction that may cause a jump (i.e., with some non-zero `j` bits) should not contain a reference to M, and vice versa.

4.2.4 Symbols

Assembly commands can refer to memory locations (addresses) using either constants or *symbols*. Symbols are introduced into assembly programs in the following three ways:

- *Predefined symbols:* A special subset of RAM addresses can be referred to by any assembly program using the following predefined symbols:
  - *Virtual registers:* To simplify assembly programming, the symbols `R0` to `R15` are predefined to refer to RAM addresses 0 to 15, respectively.
  - *Predefined pointers:* The symbols `SP`, `LCL`, `ARG`, `THIS`, and `THAT` are predefined to refer to RAM addresses 0 to 4, respectively. Note that each of these memory