2020-09-22 CH4 pt2

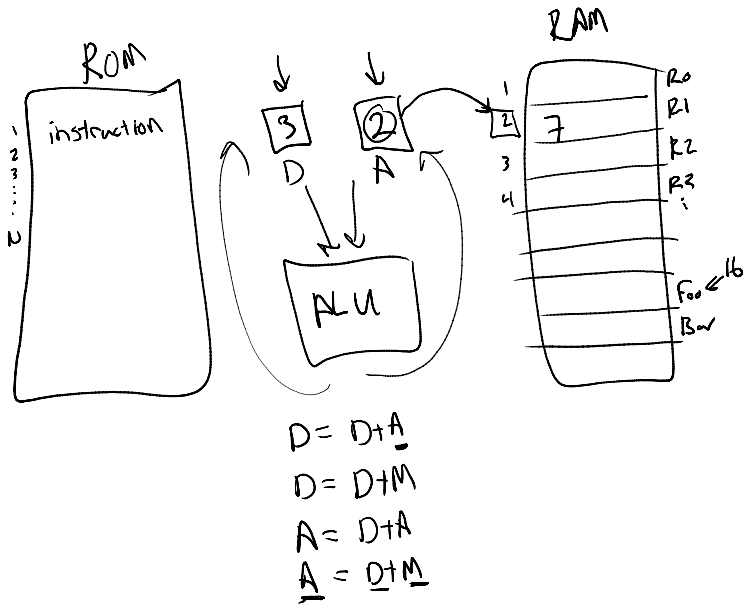
Thursday, September 24, 2020 7:12 AM

These bits instruct how to preset the x input		These bits instruct how to preset the y input		This bit selects between + / And	This bit inst. how to postset out	Resulting ALU output
zx	nx	zy	ny	f	no	out=
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x&y	if no then out=!out	f(x,y)=
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	x
1	1	0	0	0	0	y
0	0	1	1	0	1	!x
1	1	0	0	0	1	!y
0	0	1	1	1	1	-x
1	1	0	0	1	1	-y
0	1	1	1	1	1	x+1
1	1	0	1	1	1	y+1
0	0	1	1	1	0	x-1
1	1	0	0	1	0	y-1
0	0	0	0	1	0	x+y
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

Machine Language

(when a=0) ***comp*** *mnemonic*	c1	c2	c3	c4	c5	c6	(when a=1) ***comp*** *mnemonic*
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.



D = D+A

D = D+M

A = D+A

A = D+M

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."

- Jump specifications are used to manipulate the next instruction to be executed from the ROM (our program)
- Jumps are necessary for the construction of non-linear behavior
    - IF
    - WHILE
    - FOR
    - FUNCTIONS

Jump Pattern in Hack ASM

1. Compute some interesting result, store in D
2. Load some instruction value into A
3. Use a jump to determine whether or not the program should branch

```
D = M - D
@LessThan
D;JLE

(LessThan)
```