COMSM1302 In-Class Test 2 Reference Sheet

32:	space	56:	8	80:	P	104:	h	127:	DEL
33:	!	57:	9	81:	Q	105:	i	128:	newLine
34:	"	58:	:	82:	R	106:	j	129:	backSpace
35:	#	59:	;	83:	S	107:	k	130:	leftArrow
36:	$	60:	<	84:	T	108:	l	131:	upArrow
37:	%	61:	=	85:	U	109:	m	132:	rightArrow
38:	&	62:	>	86:	V	110:	n	133:	downArrow
39:	'	63:	?	87:	W	111:	o	134:	home
40:	(	64:	@	88:	X	112:	p	135:	end
41:	)	65:	A	89:	Y	113:	q	136:	pageUp
42:	*	66:	B	90:	Z	114:	r	137:	pageDown
43:	+	67:	C	91:	[	115:	s	138:	insert
44:	,	68:	D	92:	/	116:	t	139:	delete
45:	-	69:	E	93:	]	117:	u	140:	esc
46:	.	70:	F	94:	^	118:	v	141:	f1
47:	/	71:	G	95:	_	119:	w	142:	f2
48:	0	72:	H	96:	`	120:	x	143:	f3
49:	1	73:	I	97:	a	121:	y	144:	f4
50:	2	74:	J	98:	b	122:	z	145:	f5
51:	3	75:	K	99:	c	123:	{	146:	f6
52:	4	76:	L	100:	d	124:	|	147:	f7
53:	5	77:	M	101:	e	125:	}	148:	f8
54:	6	78:	N	102:	f	126:	~	149:	f9
55:	7	79:	O	103:	g			150:	f10
								151:	f11
								152:	f12

Hack character set reference (taken from Nisan and Schocken).

Command	Pops	Computes	Comment
`add`	2 values	$x + y$	Integer addition
`sub`	2 values	$x - y$	Integer subtraction
`neg`	1 value	$-y$	Arithmetic negation
`and`	2 values	$x \& y$	Bitwise AND
`or`	2 values	$x \mid y$	Bitwise OR
`not`	1 value	$!y$	Bitwise NOT
`eq`	2 values	$x == y$	Test equality
`gt`	2 values	$x > y$	Test greater than
`lt`	2 values	$x < y$	Test less than

Hack VM arithmetic/logical operation reference (see week 9 video 2).

A-instruction

(symbolic): `@`*xxx* (*xxx* is a decimal value ranging from 0 to 32767, or a symbol bound to such a decimal value)

(binary): `0` *v v v v v v v v v v v v v v v* (*v v...* *v* = 15-bit value of *xxx*)

C-instruction

(symbolic): *dest* `=` *comp* `;` *jump* (*comp* is mandatory. If *dest* is empty, the `=` is omitted; If *jump* is empty, the `;` is omitted)

(binary): `111`*accccccdddjjj*

comp		*c*	*c*	*c*	*c*	*c*	*c*
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	M	1	1	0	0	0	0
!D		0	0	1	1	0	1
!A	!M	1	1	0	0	0	1
-D		0	0	1	1	1	1
-A	-M	1	1	0	0	1	1
D+1		0	1	1	1	1	1
A+1	M+1	1	1	0	1	1	1
D-1		0	0	1	1	1	0
A-1	M-1	1	1	0	0	1	0
D+A	D+M	0	0	0	0	1	0
D-A	D-M	0	1	0	0	1	1
A-D	M-D	0	0	0	1	1	1
D&A	D&M	0	0	0	0	0	0
D|A	D|M	0	1	0	1	0	1
a == 0	*a* == 1						

dest	*d*	*d*	*d*	Effect: store *comp* in:
null	0	0	0	the value is not stored
M	0	0	1	RAM[A]
D	0	1	0	D register (reg)
DM	0	1	1	D reg and RAM[A]
A	1	0	0	A reg
AM	1	0	1	A reg and RAM[A]
AD	1	1	0	A reg and D reg
ADM	1	1	1	A reg, D reg, and RAM[A]

jump	*j*	*j*	*j*	Effect:
null	0	0	0	no jump
JGT	0	0	1	if *comp* > 0 jump
JEQ	0	1	0	if *comp* = 0 jump
JGE	0	1	1	if *comp* ≥ 0 jump
JLT	1	0	0	if *comp* < 0 jump
JNE	1	0	1	if *comp* ≠ 0 jump
JLE	1	1	0	if *comp* ≤ 0 jump
JMP	1	1	1	unconditional jump

Hack assembly and instruction set reference (taken from Nisan and Schocken).

Keyword	Addresses	Usage
`SP`	0	[Address of the topmost stack value] + 1.
`LCL`	1	Stores base address of `local` segment.
`ARG`	2	Stores base address of `argument` segment.
`THIS`	3	`pointer 0` (i.e. base address of `this` segment).
`THAT`	4	`pointer 1` (i.e. base address of `that` segment).
`R5`–`R12`	5–12	`temp` segment (max size 8).
`R13`–`R15`	13–15	Temporary variables for VM translator (if needed).
N/A	16–255	`static` segment (max size 240).
N/A	256–2047	Reserved for the stack, including `local` and `argument` segments (max size 1792 combined).
N/A	2048–16383	"Heap" memory for other purposes. Can be allocated to `this` or `that` segments.
`SCREEN`	16384–24575	Memory-mapped output to screen.
`KBD`	24576	Memory-mapped input from keyboard.

Standard memory map from Hack VM to Hack assembly (see week 9 video 4).