Stacks – Exercise Solutions

1. Write mnemonics and machine code (for the Toy CPU) for the following tasks:

    a) A subroutine, starting at location 10 (hex) that sums memory locations, then stores the result after them. Assume that the calling routine initialised the register b with the address of the memory location of the summands, and the register c with the number of them. Don't forget the return.

Solution:

mnemonic	*address*	*machine code*
ld a 00	10	06, 00
add a [b]	12	24
add b 1	13	2E, 01
sub c 1	15	56, 01
jnz c 12	17	B6, 12
st a [b]	19	64
ret	1A	F8

    b) A main routine, starting at location 00, that calls the subroutine twice: once for summing 4 locations starting at 2A, and once for summing 6 locations starting at 38.

Solution:

mnemonic	*address*	*machine code*
ld b 2A	00	0E, 2A
ld c 04	02	16, 04
call 10	04	DE, 10
ld b 38	06	0E, 38
ld c 06	08	16, 06
call 10	0A	DE, 10

2. Your answer to 1 (a) will have used the a, b and c registers in its own calculations. Suppose this is inconvenient for the main program and it would be better not to lose the values that those three registers had when the subroutine was stored. There is a trick for doing this - the stack doesn't have to be used just for saving return addresses. Rewrite the subroutine so that it pushes the three registers at the start, saving their values on the stack, and then pops them back at the end so that the calling routine gets them back unchanged. Make sure you pop them in the right order!

Solution:

mnemonic	*address*	*machine code*
push a	10	C0
push b	11	C8
push c	12	D0
ld a 00	13	06, 00
add a [b]	15	24
add b 1	16	2E, 01
sub c 1	18	56, 01
jnz c 15	1A	B6, 15
st a [b]	1C	64
pop c	1D	F0
pop b	1E	E8
pop a	1F	E0
ret	20	F8

3. Convert some ordinary expressions into reverse Polish. Run them through with a stack to check they calculate the correct answer.

a. 3+4

Solution:

Reverse Polish: 3 4 +

symbol	*stack (top on right)*
	empty
3	3
4	3, 4
+	7

b. 2*3+4

Solution:

Reverse Polish: 2 3 * 4 +

symbol	*stack (top on right)*
	empty
2	2
3	2, 3
*	6
4	6, 4
+	10

c. 2+3*4

Solution:

Reverse Polish: 2 3 4 * +

symbol	*stack (top on right)*
	empty
2	2
3	2, 3
4	2, 3, 4
*	2, 12
+	14

d. SQRT(3*3+4*4) (SQRT is the square root function)

Solution:

Reverse Polish: 3 3 * 4 4 * + SQRT

symbol	*stack (top on right)*
	empty
3	3
3	3, 3
*	9
4	9, 4
4	9, 4, 4
*	9, 16
+	25
SQRT	5

4. See if you can follow through the shunting-yard algorithm (on Wikipedia) for the example in the slides: (5+2)*SQRT(x*x+y*y)+8.

Solution:

Output	*Stack (top on right)*	*Input*	*Comments*
no output yet	*empty stack*	(5+2)*SQRT(x*x+y*y)+8	
no output yet	(	5+2)*SQRT(x*x+y*y)+8	
5	(	+2)*SQRT(x*x+y*y)+8	
5	(+	2)*SQRT(x*x+y*y)+8	
5 2	(+	)*SQRT(x*x+y*y)+8	
5 2 +	*empty*	*SQRT(x*x+y*y)+8	Brackets are not output
5 2 +	*SQRT(	x*x+y*y)+8	Three steps in one here
5 2 + x	*SQRT(	*x+y*y)+8	Variables output immediately
5 2 + x	*SQRT(*	x+y*y)+8	
5 2 + x x	*SQRT(*	+y*y)+8	
5 2 + x x *	*SQRT(+	y*y)+8	* has higher precedence than +
5 2 + x x * y	*SQRT(+	*y)+8	
5 2 + x x * y	*SQRT(+*	y)+8	
5 2 + x x * y y	*SQRT(+*	)+8	
5 2 + x x * y y * +	*SQRT	+8	
5 2 + x x * y y * + SQRT *	+	8	*, SQRT both have higher precedence than +
5 2 + x x * y y * + SQRT * 8	+	*no input left*	
5 2 + x x * y y * + SQRT * 8 +	*empty*	*no input left*	Clear stack