UNIVERSITY OF
BIRMINGHAM

M.Sc. CS / AI & CS – Term 1
Computer Systems
MIPS, Stacks, RPN Expressions [Solutions]

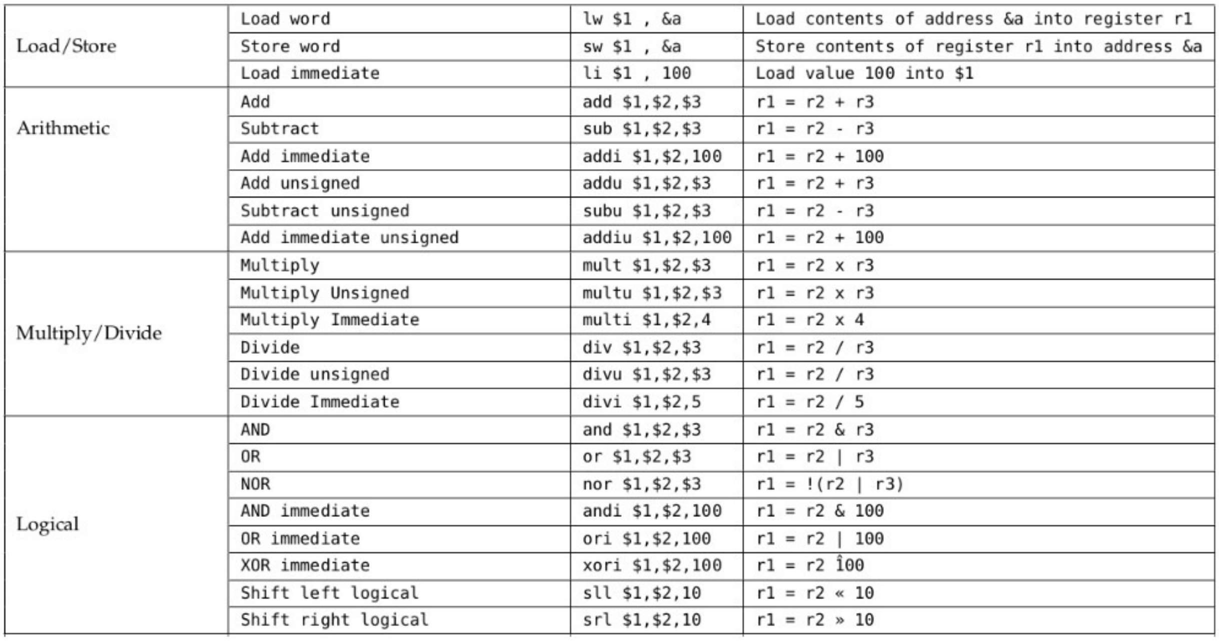
Exercise#1: Given the MIPS Instruction set, convert the following MIPS assembly code into Java-like code. Note that “bgt” instruction is for “branch greater than” and is used to jump to a label, if a register value is more than the provided constant. You are expected to consider the following register assignment while writing Java code.

MIPS Register	Java Variable
$1	i
$2	tmp
$3	sum

MIPS Assembly Code

```
    and $1,$1,$0
    and $2,$2,$0
    and $3,$3,$0
label:
    bgt $1,14,exit
    mult $2,$1,2
    add $3,$3,$2
    addi $1,$1,1
    j label
exit:
```

MIPS Instruction Set



Load/Store	Load word	lw $1 , &a	Load contents of address &a into register r1
	Store word	sw $1 , &a	Store contents of register r1 into address &a
	Load immediate	li $1 , 100	Load value 100 into $1
Arithmetic	Add	add $1,$2,$3	r1 = r2 + r3
	Subtract	sub $1,$2,$3	r1 = r2 - r3
	Add immediate	addi $1,$2,100	r1 = r2 + 100
	Add unsigned	addu $1,$2,$3	r1 = r2 + r3
	Subtract unsigned	subu $1,$2,$3	r1 = r2 - r3
	Add immediate unsigned	addiu $1,$2,100	r1 = r2 + 100
Multiply/Divide	Multiply	mult $1,$2,$3	r1 = r2 x r3
	Multiply Unsigned	multu $1,$2,$3	r1 = r2 x r3
	Multiply Immediate	multi $1,$2,4	r1 = r2 x 4
	Divide	div $1,$2,$3	r1 = r2 / r3
	Divide unsigned	divu $1,$2,$3	r1 = r2 / r3
	Divide Immediate	divi $1,$2,5	r1 = r2 / 5
Logical	AND	and $1,$2,$3	r1 = r2 & r3
	OR	or $1,$2,$3	r1 = r2 | r3
	NOR	nor $1,$2,$3	r1 = !(r2 | r3)
	AND immediate	andi $1,$2,100	r1 = r2 & 100
	OR immediate	ori $1,$2,100	r1 = r2 | 100
	XOR immediate	xori $1,$2,100	r1 = r2 1̂00
	Shift left logical	sll $1,$2,10	r1 = r2 « 10
	Shift right logical	srl $1,$2,10	r1 = r2 » 10

Solution:

```
tmp = 0;
sum = 0;
for (int i = 0; i < 15; i++){
        tmp = i * 2;
        sum = sum + tmp;
}

OR

sum = 0;
for (int i = 0; i < 15; i++){
        sum = sum + i * 2;
}
```

Exercise#2: Suppose a letter means push and an asterisk means pop in the following sequence. Give the sequence of values returned by the pop operations when this sequence of operations is performed on an initially empty LIFO stack.

A U E * * I R * * K * *

E A S * Y * Q U E * * * S T * * * I O * N * * *

EURIKA

SYEUQTSAONIE

Exercise#3: Suppose that an intermixed sequence of push and pop operations are performed. The push operation pushes the integers 0 through 9 in order; the pop operations print out the return value. Which of the following sequences could not occur?

(a) 4 3 2 1 0 9 8 7 6 5

Push 0, Push 1, Push 2, Push 3, Push 4

Pop 4, Pop 3, Pop 2, Pop 1, Pop 0 => 4 3 2 1 0

Push 5, Push 6, Push 7, Push 8, Push 9

Pop 9, Pop 8, Pop 7, Pop 6, Pop 5 => 9 8 7 6 5

So we can get this output: **4 3 2 1 0 9 8 7 6 5**

(b) 4 6 8 7 5 3 2 9 0 1

Push 0, Push 1, Push 2, Push 3, Push 4

Pop 4 => 4

Push 5, Push 6

Pop 6 => 6

Push 7, Push 8

Pop 8, Pop 7, Pop 5, Pop 3, Pop 2 => 8, 7, 5, 3, 2

Push 9

Pop 9, Pop 1, Pop 0 => 9, 1, 0

So we cannot get this output: **4 6 8 7 5 3 2 9 0 1**

(c) 2 5 6 7 4 8 9 3 1 0

Push 0, Push 1, Push 2

Pop 2 => 2

Push 3, Push 4, Push 5

Pop 5 => 5

Push 6

Pop 6 => 6

Push 7

Pop 7, Pop 4 => 7, 4

Push 8

Pop 8 => 8

Push 9

Pop 9, Pop 3, Pop 1, Pop 0 => 9 3 1 0

So we can get this output: **2 5 6 7 4 8 9 3 1 0**

(d) 4 3 2 1 0 5 6 7 8 9

Push 0, Push 1, Push 2, Push 3, Push 4

Pop 4, Pop 3, Pop 2, Pop 1, Pop 0 => 4 3 2 1 0

Push 5

Pop 5 => 5

Push 6

Pop 6 => 6

Push 7

Pop 7 => 7

Push 8

Pop 8 => 8

Push 9

Pop 9 => 9

So we can get this output: **4 3 2 1 0 5 6 7 8 9**

Exercise#4: Convert the following infix expressions to postfix (RPN) expressions using the Shunting Yard algorithm. With the help of a Stack and using the RPN Expression evaluation scheme, evaluate them as well.

(a) 4 + (6 * 2) - 15

Input	Stack	Output
4	Empty	4
+	+	4
(	+ (	4
6	+ (	4 6
*	+ (*	4 6
2	+ (*	4 6 2
)	+	4 6 2 *
-	-	4 6 2 * +
15	-	4 6 2 * + 15
	Empty	**4 6 2 * + 15 -**

(b) 4 * (4 + 10 / 2.5 – 8)

Input	Stack	Output
4	Empty	4
*	*	4
(	* (	4
4	* (	4 4
+	* (+	4 4
10	* (+	4 4 10
/	* (+ /	4 4 10
2.5	* (+ /	4 4 10 2.5
-	* (-	4 4 10 2.5 / +
8	* (-	4 4 10 2.5 / + 8
)	*	4 4 10 2.5 / + 8 -
	Empty	**4 4 10 2.5 / + 8 - ***

(c) 17 + 21 * 3 / 7 + 21

Input	Stack	Output
17	Empty	17
+	+	17
21	+	17 21
*	+ *	17 21
3	+ *	17 21 3
/	+ /	17 21 3 *
7	+ /	17 21 3 * 7
+	+	17 21 3 * 7 / +
21	+	17 21 3 * 7 / + 21
	Empty	**17 21 3 * 7 / + 21 +**

Exercise#5: Run the above RPN expressions through with a stack to check they calculate the correct answer.

(a)

Input	Stack
4	4
6	4 6
2	4 6 2
*	4 12
+	16
15	16 15
-	**1**

(b)

Input	Stack
4	4
4	4 4
10	4 4 10
2.5	4 4 10 2.5
/	4 4 4
+	4 8
8	4 8 8
-	4 0
*	**0**

(c)

Input	Stack
17	17
21	17 21
3	17 21 3
*	17 63
7	17 63 7
/	17 9
+	26
21	26 21
+	**47**