Kristy Nguyen
Dr. Ed Jorgensen
CS 219-1002: Computer Organization
8 February 2021

CS 219 – Assignment #3

Purpose: Become familiar with the MIPS Instruction set, MIPS Architecture and QtSpim (the MIPS simulator).
Due: Monday (2/08)
Points: 100

Assignment:

1) Based on the provided program, answer the following questions: [20 pts, 2 pts each]

a) What is/are the MIPS instruction(s) that replace the pseudo-instruction:

```
li $t0, 42
ori $8, $0, 42
```

b) What is the MIPS opcode, in hex, for that instruction?
0x3408002a

c) What is the address, in hex, for that instruction (first occurrence)?
0x00400034

d) What is/are the MIPS instruction(s) that replace the pseudo-instruction:

```
li $t1, 65539
lui $1, 1
```

e) What is/are the MIPS instruction(s) that replace the pseudo-instruction:

```
li $t2, -42
lui $1, -1
```

f) What is/are the MIPS instruction(s) that replace the pseudo-instruction:

```
li $t3, -65539
lui $1, -2
```

g) What is/are the MIPS instruction(s) that replace the pseudo-instruction:

```
bge $t5, $t2, notMin
slt $1, $13, $10
beq $1, $0, 8
```

h) What is/are the MIPS instruction(s) that replace the pseudo-instruction:

```
ble $t5, $t3, notMax
slt $1, $11, $13
beq $1, $0, 8
```

i) What are the MIPS instruction(s) that replace the pseudo-instruction:

```
sub $t1, $t1, 1
```

```
addi $9, $9, -1
```

j) What are the MIPS instruction(s) that replace the pseudo-instruction:

```
add $t0, $t0, 4
addi $8, $8, 4
```

2) Based on the provided program, answer the following questions: [14 pts, 2 pts each]

a) What is the **$sp** register used for?

The $sp register is used to denote the most recently allocated address in a stack that shows where registers should be spilled or where old registers can be found.

b) What is the value, in hex, of the **$sp** register?

0x7ffff548

c) What is the **$pc** register used for?

The $pc or program counter register points to the next instruction to be executed and is automatically updated by the CPU after each instruction is executed.

d) What is the initial value, in hex, of the **$pc** register *before* any code is executed?

0x00000000

e) What is the initial value, in hex, of the processor status word (labeled 'Status')?

0x3000ff10

f) What is the initial value, in hex, of the **$epc** register?

0x00000000

g) What is the **$epc** register used for?

The exception program counter ($epc) contains the address of the instruction that caused the exception.

3) For the C statements below, what is the corresponding MIPS assembly code? Use a minimal number of MIPS assembly instructions. You may use pseudo-instructions. [12 pts, 3 pts each] *Note*, **a**, **b**, **c**, **d**, **f**, **i**, **x**, **y**, and **z** are declared, word-sized variables.

Note, **AR1** and **AR2** are declared, word-sized arrays with 100 elements each

a) **`x = a + b + c * 2;`**

```
lw   $t0, a
lw   $t1, b
lw   $t2, c
lw   $t3, x
mul  $t3, $t2, 2          # x = c * 2
add  $t3, $t3, $t0        # x = x + a
add  $t3, $t3, $t1        # x = x + b
```

b) **y = c * 3 + d / 5;**

```
lw   $t0, c
lw   $t1, d
lw   $t3, y
mul  $t3, $t0, 3          # y = c * 3
div  $t4, $t1, 5          # $t4 = d / 5
add  $t3, $t3, $t4        # y = y + $t4
```

c) **z = -a + AR1[29] + AR2[35];**

```
la   $t0, AR1
la   $t1, AR2
addu $t0, $t0, 116        # AR1[29]
addu $t1, $t1, 140        # AR2[35]
lw   $t2, z
lw   $t3, a
neg  $t2, $t3             #; z = -a
add  $t2, $t2, ($t0)      #; z = z + AR[29]
add  $t2, $t2, ($t1)      #; z = z + AR[35]
```

d) **f = AR2[AR1[i]+13];**

```
lw   $t0, f
la   $t1, AR1
la   $t2, AR2
lw   $t4, i
addu $t1, $t1, $t4        # $t1 = AR1[i]
add  $t5, ($t1), 13       # $t5 = AR1[i] + 13
addu $t2, $t2, $t5        # $t2 = AR2[$t5]
lw   $t0, ($t2)           # f = AR2[AR1[i]+13]
```

4) Given the below register values, what is the result, in register **$t2**, of the following operations on both **(a)** and **(b)**. [12 pts, 2 pts each]

(a)	$t0 = `0xAAAAAAAA`	$t1 = `0x12345678`

a)
```
sll $t2, $t0, 4
and $t2, $t2, -1
0xAAAAAAA0
```

b)
```
sll $t2, $t0, 4
or $t2, $t2, $t1
```

```
0xBABEFEF8
```

c)
```
sll $t2, $t0, 4
and $t2, $t2, $t1
0x02200220
```

d)
```
srl $t2, $t0, 2
and $t2, $t2, $t1
0x02200228
```

(b)	$t0 = `0xA5A5A5A5`	$t1 = `0x87654321`

e)
```
sll $t2, $t0, 4
or $t2, $t2, $t1
0xDF7F5B71
```

f)
```
sll $t2, $t0, 4
and $t2, $t2, -1
0x5A5A5A50
```

g)
```
sll $t2, $t0, 2
and $t2, $t2, -1
0x96969694
```

5) What is the shortest sequence of MIPS instructions that extracts three fields; bits 21-25, 16-20, and 11-15 (inclusive) from register **$a0**. The results should be placed it in the lower order portion of register **$s0** (for x bits), **$s1** (for y bits), and **$s2** (for z bits) registers (zero filled otherwise). [3 pts]

For example, the highlighted bits from **$a0**:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
?	?	?	?	?	?	x	x	x	x	x	y	y	y	y	y	z	z	z	z	z	?	?	?	?	?	?	?	?	?	?	?

Should be placed in the following locations of **$s0** and the rest set to 0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x

```
srl   $t0, $a0, 11      # remove bits 0-10
sll   $t0, $a0, 27      # remove bits 26-31
srl   $s0, $t0, 10      # remove bits 11-20, store bits 21-25 in x
sll   $s1, $t0, 5       # remove bits 21-25, store bits 11-20 in y
srl   $s1, $t0, 5       # remove bits 11-15, store bits 16-20 in y
sll   $s2, $t0, 10      # remove bits 16-25, store bits 11-15 in z
```

6) What is the shortest sequence of MIPS instructions that extracts op field value of bits 26-31 (inclusive) from register **$a1** and places it in the lower order portion of register **$t0** (zero filled otherwise). [3 pts]

```
srl $t0, $a1, 27      # remove bits 26-31, store bits in $t0
sll $t0, $t0, 26      # remove bits 0-25
srl $t0, $t0, 26      # set 0 on bits 0-25
```

7) Show the minimal sequence of MIPS instructions required for the C statements:

[9 pts, 3 pts each]

Note, the "|" is an "or" operation and the "**&**" is the "and" operation.
Note, all variables are word-sized (32-bits).
Note, multiple correct solutions possible, but must not exceed three instructions each.

a) **x = a | 1729;**

```
lw $t0, x
lw $t1, a
ori $t0, $t1, 1729
```

b) **y = b & 97;**

```
lw $t0, y
lw $t1, b
andi $t0, $t1, 97
```

c) **z = c | -1;**

```
lw $t0, z
lw $t1, c
ori $t0, $t1, -1
```

8) Describe what the following MIPS code computes. Assume the **$v0** is used for the output.

[8 pts]

```
     lw $a0, a
     lw $a1, b
     li $t0, 0
lp: beq $a1, 0, finish      # lp: if $a1 == 0, branch to finish
     add $t0, $t0, $a0      # $t0 = $t0 + a
     sub $a1, $a1, 1        # $a1 = b - 1
     b lp                   # branch to lp
finish:
     addi $t0, $t0, 10      # finish: $t0 = $t0 + 10
```

```
add $v0, $t0, $zero   # output ($v0) = $t0 + 0
```

9) Modify the provided program (**`mips.asm`**) to find the maximum, minimum, and integer average of the ***odd*** values in the list. *Note*, you should use a logical instruction to determine if the number is even or odd. Submit a copy of the source program and the QtSpim Log File (showing the Text Segment and Console). [19 pts]