Name: Cicelia Siu

Class: 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]
2. What is/are the MIPS instruction(s) that replace the pseudo-instruction:

**li $t0, 42**

ori $8, $0, 42

1. What is the MIPS opcode, in hex, for that instruction?

0x4308002a

1. What is the address, in hex, for that instruction (first occurrence)?

00400034

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

**li $t1, 65539**

lui $1, 1

ori $9, $1, 3

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

**li $t2, -42**

lui $1, -1

ori $10, $1, -42

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

**li $t3, -65539**

lui $1, -2

ori $11, $1, -3

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

**bge $t5, $t2, notMin**

slt $1, $13, $10

beq $1, $0, 8 (notMin - 0x004000b0)

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

**ble $t5, $t3, notMax**

slt $1, $11, $13

beq $1, $0, 8 (notMax-0x004000bc)

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

**sub $t1, $t1, 1**

addi $9, $9, -1

1. 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]

1. What is the **$sp** register used for?

$sp points to the first address in stack

1. What is the value, in hex, of the **$sp** register?

0x7ffffe08

1. What is the **$pc** register used for?

$pc points to the next instruction to be executed

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

0x00000000

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

0x3000ff10

1. What is the initial value, in hex, of the **$epc** register?

0x00000000

1. What is the **$epc** register used for?

has the address of the instruction when an exception occurs

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

1. **x = a + b + c \* 2 ;**

\*\*FOLLOWED PEMDAS\*\*

li $t0, 2

lw $t1, c

lw $t2, b

lw $t3, a

mul $t1, $t1, $t0

add $t1, $t1, $t2

add $t1, $t1, $t3

sw $t1, x

1. **y = c \* 3 + d / 5 ;**

\*\*FOLLOWED PEMDAS\*\*

lw $t0, c

li $t1, 3

lw $t2, d

li $t3, 5

mul $t0, $t0, $t1

div $t2, $t2, $t3

add $t0, $t0, $t2

sw $t0, y

1. **z = - a + AR1 [29]+ AR2 [35] ;**

la $t0, AR1

lw $t1, 116($t0) # 29\*4 =116 (AR1[0] = (t0))

la $t2, AR2

lw $t3, 140($t2) # 35\*4 = 140

lw $t4, a

sub $t4, $zero, $t4 # t4 = 0-a

add $t4, $t4, $t3

add $t4, $t4, $t1 # z = $t4

sw $t4, z

1. **f = AR2 [AR1[i] + 13];**

la $t0, AR1

li $t1, 4

lw $t2, i

mul $t1, $t1, $t2

add $t0, $t1, $t0 # address + offset for AR1

lw $t2, ($t0)

addi $t2, $t2, 13 #ARI[i] +13

la $t0, AR2

mul $t2, $t1, $t2 #(ARI[i] +13) \*4

add $t2, $t2, $t0 #address + offset

lw $t1, ($t0)

sw $t1, f

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** |

1. sll $t2, $t0, 4

and $t2, $t2, -1 # -1 = 1111 1111 1111 1111

#t2 = 0xAAAAAAA0 or 1010 1010 1010 1010 1010 1010 1010 0000

# -1 = 1111 1111 1111 1111 1111 1111 1111 1111

$t2 will equal 1010 1010 1010 1010 1010 1010 1010 00002

or 0xAAAAAAAA0

1. sll $t2, $t0, 4

or $t2, $t2, $t1

#0x AAAAAAA0 / 1010 1010 1010 1010 1010 1010 1010 0000

#$t1 = 0001 0010 0011 0100 0101 0110 0111 1000

t2 will equal 1011 1010 1011 1110 1111 1110 1111 10002

or 0xBABEFEF8

1. sll $t2, $t0, 4 #0xAAAAAAA0 / 1010 1010 1010 1010 1010 1010 1010 0000

and $t2, $t2, $t1 #$t1 = 0001 0010 0011 0100 0101 0110 0111 1000

t2 will equal 0000 0010 0010 0000 0000 0010 0010 00002

or 0x02200220

1. srl $t2, $t0, 2 #0x0AAAAAAAA / 0000 1010 1010 1010 1010 1010 1010 1010

and $t2, $t2, $t1 #$t1 = 0001 0010 0011 0100 0101 0110 0111 1000

t2 will equal 0000 0010 0010 0000 0000 0010 0010 1000

or 0x02200228

|  |  |  |
| --- | --- | --- |
| **(b)** | **$t0 = 0xA5A5A5A5** | **$t1 = 0x87654321** |

1. sll $t2, $t0, 4 #t2 = 0101 1010 0101 1010 0101 1010 0101 0000

or $t2, $t2, $t1 #t1 = 1000 0111 0110 0101 0100 0011 0010 0001

$t2 will equal 1101 1111 0111 1111 0101 1011 0111 0001

or 0xDF7F5B71

1. sll $t2, $t0, 4 #t2 = 0101 1010 0101 1010 0101 1010 0101 0000

and $t2, $t2, -1 # -1 = 1111 1111 1111 1111 1111 1111 1111 1111

$t2 wil equal 0101 1010 0101 1010 0101 1010 0101 0000

or 0x5A5A5A50

1. sll $t2, $t0, 2 #t2 = 10 0101 1010 0101 1010 0101 1010 0101 00

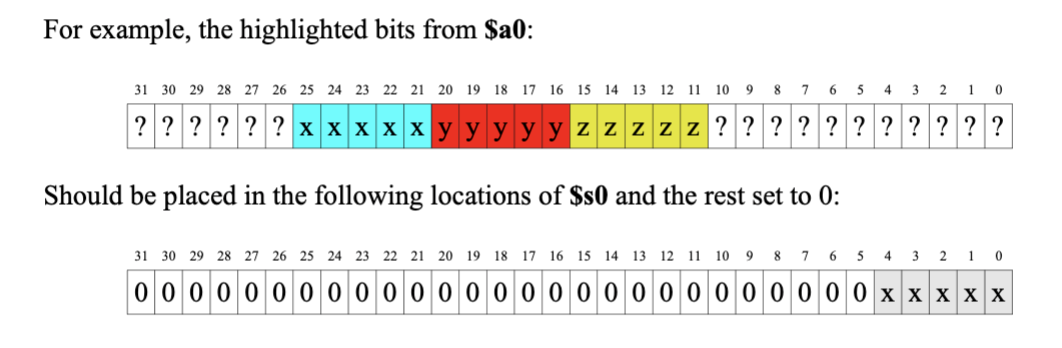
and $t2, $t2, -1 # -1 = 11 1111 1111 1111 1111 1111 1111 1111 11

$t2 will equal 1001 0110 1001 0110 1001 0110 1001 0100

or 0x96969694

5) What is the shortest sequence of MIPS instructions that extracts three fields; bits 21-25, 16-25, 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]



srl $a0, $a0, 11 #00000000000 ?????? xxxxx yyyyy zzzzz

andi $s2, $a0, 31 #00000000000 000000 00000 00000 11111

# places z bits into $s2 and the rest zeros

srl $a0, $a0, 5 # 00000 00000000000 ?????? xxxxx yyyyy

andi $s1, $a0, 31 #00000000000 000000 00000 00000 11111

# places y bits into $s1, and the rest are zeros

srl $a0, $a0, $a5 # 00000 00000 00000000000 ?????? xxxxx

andi $s0, $a0, 31 #00000000000 000000 00000 00000 11111

#places x bits into $s0 and the rest are zeros

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, 26

# xxxxxx ????? ????? ????? ???????????

# srl to get 00000 00000 00000 00000000000 xxxxxx

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, a

ori $t0, $t0, 1729

sw $t0, x

b) y = b & 97;

lw $t0, b

andi $t0, $t0, 97

sw $t0, x

c) z = c | -1;

lw $t0, c

ori $t0, $t0, -1 # ori is int signed

sw #t0, x

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 # jump to finish if b=0

add $t0, $t0, $a0 # t1 = 0->a

sub $a1, $a1, 1 # b-1

b lp

finish:

addi $t0, $t0, 10

add $v0, $t0, $zero

$t0 finds (b\*a)+10

$v0 outputs $t0

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]

mips.asm

qtSpimLogFile.txt