Course Name: Computer Architecture Lab

Course Number and Section: **14:332:333:03**

Experiment: [Experiment # 4 = RISC-V Assembly]

Lab Instructor: Christos Mitropoulos

Date Performed: October 24 2018

Date Submitted: November 7 2018

Submitted by: Vancha Verma 173004061

Course Name: Computer Architecture Lab

Course Number and Section: **14:332:333:03**

! Important: Please include this page in your report if the submission is a paper submission. For electronic submission (email or Sakai) please omit this page.

-------------------------For Lab Instructor Use ONLY-------------------------

GRADE: _______________

COMMENTS:

Electrical and Computer Engineering Department
School of Engineering
Rutgers University, Piscataway, NJ 08854
ECE Lab Report Structure

Exercise 1:

1.Write a function triple in RISC-V that, when given an integer x, returns 3x.

**assume a0 is already set as input: x

```
triple:
add a3, x0, a0
addi a1, x0, 3
mul a2, a1, a3
```

2. Write a function power in RISC-V that takes in two numbers x and n, and returns xn . You may assume that n ≥ 0 and that multiplication will always result in a 32-bit number.

**assume that a0 and a1 are already preloaded with the values

```
power:
addi t0, x0, a0 #x
addi t1, x0, a1 #n
addi t1, t1, -1
bge x0, t1, end
mul t2, t0, t0

expo:
addi t1, t1, -1
bge x0, t1, end
mul t2, t2, t0
j expo
end:
```

Exercise 2: Comment each snippet with what the snippet does. Assume that there is an array, int arr[6] = {3, 1, 4, 1, 5, 9}, which starts at memory address 0xBFFFFF00, and a linked list struct (as defined below), struct ll* lst, whose first element is located at address 0xABCD0000. s0 then contains arr's address, 0xBFFFFF00, and s1 contains lst's address, 0xABCD0000. You may assume integers and pointers are 4 bytes and that structs are tightly packed.

```
struct ll {
     int val;
```

```
    struct ll* next;
}
```

```
1.  lw t0, 0(s0) #load arr[0]into t0
    lw t1, 8(s0) #load arr[2] into t1
    add t2, t0, t1 # t2 = t0+t1
    sw t2, 4(s0) #t2 = arr[1]
```

2.

```
add t0, x0, x0 #t0 = 0
loop:     slti t1, t0, 6 #shifts t1 by 6
          beq t1, x0, end #if t1 = 0, goes to label end
          slli t2, t0, 2 #shifts t0 left by 2 and stores in 2
          add t3, s0, t2 #t3 = s0 + t2
          lw t4, 0(t3) #t4 = arr[t3]
          sub t4, x0, t4 # t4 = -t4
          sw t4, 0(t3) # store new t4 in arr[t0]
          addi t0, t0, 1 #t0 = t0+1
          jal x0, loop #jump back to the label loop
end:
```

```
3. loop:  beq s1, x0, end #go to end if s1 is equal to 0
          lw t0, 0(s1) #load node value to t0
          addi t0, t0, 1 #t0 = t0+1
          sw t0, 0(s1) #store nre value of t0 in the node
          lw s1, 4(s1) #load next address
          jal x0, loop #jump and link to loop
end:
```

Exercise 4:

1. How do we pass arguments into functions?

Registers a0 - a7

2. How are values returned by functions?

a0 and a1 are used as return registers

3. What is sp and how should it be used in the context of RISC-V functions?

sp: stack pointer. The register can be used to save values of registers that can be overwritten at the end. Values of the register are restored at the end.

4. Which values need to be saved before using jal?

t0 - t6, a0 - a7, ra

5. Which values need to be restored before using jr to return from a function?

gp, s0-s11 and sp

Exercise 5:
Write a function sumSquare in RISC-V that, when given an integer n, returns the summation below. If n is not positive, then the function returns 0.

n^2 + (n − 1)^2 + (n − 2)^2 + (n − 3)^2 +. . . + 1^2

For this problem, you are given a RISC-V function called square that takes in an integer and returns its square. Implement sumSquare using square as a subroutine.

```
program:
addi sp, sp -12 #Move the stack pointer down
sw ra, 0(sp) #store values in the register
sw t0, 4(sp) #holds n
sw t1, 8(sp) #holds previous values sum
sw t2, 12(sp) #holds square
addi t0, x0, 4 #enter the n value, 4 in this case
addi t1, x0, 0
bge t0, x0, sumSquare
ble t0, x0, exit

sumSquare:
bge x0, t0, exit
jal ra, square
add t1, t1, t2 #add to old values
addi t0, t0, -1 #reduce n by 1, new n
jal x0 sumSquare

square:
mul t2, t0, t0 #squares n
jalr x0 0(x1)

exit:
add a0, t1, x0 #a0 = s1
lw ra, 0(sp) # Restore registers
lw t0, 4(sp)
lw t1, 8(sp)
lw t2, 12(sp)
addi sp, sp, 12 #Move the pointer back up

addi a1, a0, 0
addi a0, x0, 1
ecall # Print Result
```