Course Name: Yu Liu

Course Number and Section: **14:332:333:01**

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

Lab Instructor: Ali Essam

Date Performed: 10/29/2018

Date Submitted: 11/12/2018

Submitted by: Yu Liu 173001088

Course Name: ________________

Course Number and Section: **14:332:333:01**

! 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:

1. RISC – V Function:

```
a.  Triple: add a0, a0, a0
                jr ra

b.  power:  li  t0,  0
                addi  t1,  a0,  0
                addi  a0,  x0,  1
    loop:      bge  t0,  a1,  end
                mul  a0,  a0,  t1
                addi  t0,  t0,  1
                jal  x0,  loop
    end:       jr  ra
```

2. RISC – V Arrays and Lists:

```
struct II {
          int val:
          struct ll* next;
}
```

```
1.  lw  t0,  0(s0)       loads arr[0] into register t0
    lw  t1,  8(s0)       loads arr[2] into register t1
    add  t2,  t0,  t1    sets t2 equal to t0 + t1
    sw  t2, 4(s0)        sets arr[1] equal to the value of t2

                         sets arr[1] to arr[0] + arr[2]
```

```
2.          Add  t0,  x0,  x0          register t0 = x0 = 0
   loop:    slti  t1,  t0, 6           sets t1 to 1 if t0 < 6, 0 otherwise
            beq  t1,  x0,  end         branches to the end if t1 is 1 for t0 >= 6
            slli  t2,  t0,  2          sets t2 to t0, t0 is shifted twice to the left or t0 * 4
            add  t3,  s0,  t2          set t3 = arr[t0] + t2. s0 is arr[t0]
            lw  t4,  0(t3)             load t3 into register t4. t3 is arr[t0]
            sub  t4,  x0,  t4          set t4 = 0 – t4. So t4 is basically negative.
            sw  t4,  0(t3)             stores the new t4
            addi  t0,  t0,  1          t0 = t0 + 1. t0 is incremented by 1 so the next
                                       element in the array
            jal  x0,  loop             jump back to the loop label
   end:
```

```
3. loop:    beq s1, x0, end            branch to end if struct pointer is x0 or null
```

```
        lw t0, 0(s1)          load value of s1 into t0
        addi t0, t0, 1        increment t0 by 1 and set that as the new t0
        sw t0, 0(s1)          store the new t0 with the value of the node
        lw s1, 4(s1)          load the address of the next element into s1
        jal x0, loop          jump back to the loop label
end:
```

3. RISC V Calling Conventions:

a. we pass arguments into functions by using the argument registers from a0 to a7.

b. Values are returned by functions through a0 and a1.

c. sp is “stack pointer” and it is used to create more space and add to free space. The stack pointer is used to save and restore the value of registers that maybe overwritten.

d. Registers a0 to a7, t0 to t6, and ra need to be saved before using jal.

e. Register sp, gp, and s0 to s11 need to be restored before using jr.

4. Writing RISC V Functions:

```
sumSquare:  addi  sp,  sp  -12        create space 3 words on the stack for each stack uses 4
                                      byte
            sw  ra,  0(sp)            stores the return address
            sw  s0,  4(sp)            stores the stack pointer to register to s0
            sw  s1,  8(sp)            store register s1
            add  s0,  a0,  x0         set s0 equal to a0 + x0
            add  s1,  x0,  x0         set s1  equal to x0 + x0 for x0 = 0  s1 = 0
     loop:  bge  x0,  s0,  end        branch if s0 is not positive
            add  a0,  s0,  x0         a0 = s0 + x0, x0 = 0 so a0 = s0
            jal   ra,  square         call the function square
            add  s1,  s1,  a0          s1 = s1 + s0
            addi  s0,  s0,  -1        decrement s0 by 1
            jal  x0,  loop            jump back to the loop label
      end:  add   a0,  s1,  x0        set a0 to s1 because x0 = 0
            lw   ra,  0(sp)            restore ra
            lw   s0,  4(sp)            restore s0
            lw   s1,  8(sp)            restore s1
            addi  sp,  sp,  12        free space of the stack for the 3 word
            jr   ra                   return to the caller
```