Course Name: Comp Arch LAb_____

Course Number and Section: **14:332:xxx:xx**

Experiment: [Experiment # [3] – C Memory Management and Introduction to RISC-V]

Lab Instructor:Mengmei Ye

Date Performed:October 12 2018

Date Submitted: October 26 2018

Submitted by: [Erika Jean-Pierre - 169008322]

Course Name: _____

Course Number and Section: **14:332:xxx:xx**

! 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

Static Variable	static
Local Variables	static
Global variables	static
Constants	code, static, stack
Machine Instructions	code
malloc()	Heap
String Literals	Static

Escercise 2

a.) An array of k integers

```
arr = (int*) malloc(sizeof (int) * k);
```

b.) A string str of length p

```
str = (char *) malloc(sizeof(char) * (p + 1));
```

c.) An n x m Matrix mat of integers initialized to zeros

```
mat = (int **) calloc(n, sizeof(int *));
    for (int i = 0; i < m; i++) {
        mat[i] = (int *) calloc(m, sizeof(int));
    }
```

Exercise 3

a) lw t0, 12(s0)

This code loads arr[3] into address xt0.

b) slli t1, t0, 2 #multiplies the number in address t0 by 4

add t2, s0, t1 # adds t1 to s0

lw t3, 0(t2) #loads the array at arr[t2] to t3

addi t3, t3, 1 #add one to immediately to t3 and store it in t3 so now t3 has a new value

sw t3, 0(t2) #stores back into memory

This code ultimately increments arr[t2[by 1

c) lw t0, 0(s0) # loads arr[0] into address t0

xori t0, t0, 0xFFF

addi t0, t0, 1

Sets xt0 to the two's complement og arr[0] also known as the negative of arr[0];

Exercise 4

s0 < s1	s0<= s1	s0 > 1
slt t0, s0, s1 bne t0, 0, label	slt t0, s1, s0 beq t0, 0, label	sltiu t0, s0, 2 beq t0, 0, label

Exercise 5

a. The register representing the variable k.

t0 is representing the variable k

b. The registers acting as pointers to the source and dest arrays.

t1 is acting as a pointer to the source array and t2 is acting as a pointer to the dest array as we can see in the main section.

c. The assembly code for the loop found in the C code.

d. How the pointers are manipulated in the assembly code.

Excercise 6

C	Riscv

// s0 -> a, s1 -> b
// s2 -> c, s3 -> z
int a = 4, b = 5, c = 6, z;
z = a + b + c + 10;
	addi s0 x0 4
	addi s1 x0 5
	addi s2 x0 6
	add s3 so s1
	add s3 s3 s2
	addi s3 s3 10

// s0 -> int * p = intArr;
// s1 -> a;
*p = 0;
int a = 2;
p[1] = p[a] = a;
	sw x0 0(s0)
	addi s1 x0 2
	sw s1 4(s0)
	slli t0 s1 2
	add t0 t0 s0
	sw s1 0(t0)

// s0 -> a, s1 -> b
int a = 5, b = 10;
if(a + a == b) {
a = 0;
} else {
b = a - 1; }
	add s0 x0 5
	addi s1 x0 10
	add t0 s0 s0
	bne t0 s1 else
	xor s0 x0 x0
	jal x0 exit
	else:
	addi s1 s0 -1
	exit:

computes s1 = 2^30
s1 =1
for (s0=0 , s0 < , s++) {
s1 *= 2;
}
	addi s0, x0, 0
	addi s1, x0, 1
	addi t0, x0, 30
	loop:
	beq s0, t0, exit
	add s1, s1, s1
	addi s0, s0, 1
	jal x0, loop
	exit

// s0 -> n, s1 -> sum
// assume n > 0 to start
int sum;
for(sum=0;n>0;sum+=n--);
	addi s1 s1 0
	loop:
	beq s0 x0 exit
	add s1 s1 so
	add s0 s0 -1

	jal x0, loop exit:

Excercise 7