Course Name: COMPUTER ARCHIT LAB

Course Number and Section: 14:332:333:02

Experiment: Lab 3 Report

Lab Instructor: Haolin Jiang

Date Performed: 3/5/25

Date Submitted: 3/26/25

Submitted by: Chance Reyes 225006531

Exercise 1 Memory Allocation in C [15 pts]

1.1

```c
#include <stdio.h>

#include <stdlib.h>

int main() {
    int *arr = (int *)malloc(6 * sizeof(int));
    int k = 6;

    arr[0] = 11;
    arr[1] = 22;
    arr[2] = 33;
    arr[3] = 44;
    arr[4] = 55;
    arr[5] = 66;

    for(int i = 0; i < k; i++){
        printf("%d ", arr[i]);
    }
    printf("\n");

    arr = (int *)realloc(arr, 4 * sizeof(int));
    k = 4;

    arr[0] = 10;
    arr[1] = 20;
    arr[2] = 30;
    arr[3] = 40;

    for(int i = 0; i < k; i++){
        printf("%d ", arr[i]);
    }

    free(arr);
    return 0;
}
```

Result:

```
re\Labs\Lab 3> ./exercise1
11 22 33 44 55 66
10 20 30 40
```

Exercise 2 A Minimal RISC-V Program Tutorial [25 pts]

2.1

Register x5: becomes 0x00000003

Register x6: becomes 0x0000000e then 0x00000011

Register x7: becomes 0x00000012

Register x10: becomes 0x00000090

2.2

```
li x5, 0x10000100 # li: load instant value. x5 stores the beginning address of the array

addi a0, x0, 2
sw a0, 0(x5) # store 2 into memory 0x10000100; arr[0]
addi a1, x0, 15
sw a1, 4(x5) # store 15 into memory 0x10000104; arr[1]
addi a2, x0, 0
sw a2, 8(x5) # store 0 into memory 0x10000108; arr[2]
addi a3, x0, 0
sw a3, 12(x5) # store 0 into memory 0x1000010C; arr[3]

add a2, a0, a1 # add the entries; a2 = a0 + a1;

srl a3, a2, a0 # logical right shift; a3 = a2 >> a0;


sw a2, 8(x5) # store the result of the addition
sw a3, 12(x5) # store the result of the right shift
```

Result:

0x1000010c	04
0x10000108	11
0x10000104	0f
0x10000100	02

Exercise 3 Environment Calls in Venus [25 pts]

3.1

The program outputs the integer 82, because the function code x10 was set to print an integer. If we change line 3 to addi x10, x0, 11, the function code is now set to print a character. The program then prints the ASCII value 82 which is R.

3.2

```
addi x1, x0, 46 # char a1_c1 = '.'; ASCII 46
addi x2, x0, 15 # int a1_n3 = 15;
addi x3, x0, 9 # int a1_n4 = 9;
sub x4, x2, x3 # int a1_n5 = a1_n3 - a1_n4;
addi x5, x3, -3 # int a1_n6 = a1_n4 - 3;

addi x10, x0, 1 # function code for print_int
add x11, x0, x4 # passes a1_n5 as argument for print_int
ecall # prints a1_n5 as the integer 6

addi x10, x0, 11 # function code for print_char
add x11, x0, x1 # passes a1_c1 as argument for print_char
ecall # prints a1_c1 as the ASCII character "."

addi x10, x0, 1 # function code for print_int
add x11, x0, x5 # passes a1_n6 as argument for print_int
ecall # prints a1_n6 as the integer 6
```

Result:

```
6.6
```

Exercise 4 [35 pts]

4.1

By adding a jump statement to “main:” in the program, the program will keep looping and print "ECE333 Sp24" continuously until interrupted.

4.2

```
.data
    arr: .word 11, 22, 33, 44, 55, 66
.text
    main:
        la x5, arr # load base address of arr into x5
        addi x6, x0, 6 # load number of elements in arr into x6
        addi x4, x0, 4 # sizeof(word) = 4 bytes
```

```
        addi x7, x0, 0 # stores sum of elements in arr

        addi x8, x0, 0 # int i = 0;
        loop: # for(int i = 0; i < 6; i++)
            bge x8, x6, end # i>=6;

            mul x9, x8, x4 # calculate offset
            add x13, x5, x9 # set address arr[i] into x13
            lw x12, 0(x13) # load address arr[i] into x12
            add x7, x7, x12 # sum += arr[i]

            addi x8, x8, 1 # i++;
            j loop
        end:

        addi x10, x0, 1 # function code for print_int
        add x11, x0, x7 # passes sum in x7 as argument for print_int
        ecall # prints sum
```

Result:

```
231
```

4.3

```
#ex4_in_riscv.s
.data
#[here your code]Initialize variable output_string: "c is equal to "
    output_string: .asciiz "c is equal to "
.text
    main:
        addi x14, x0, 12 #Initialize variable a
        addi x15, x0, 4 #Initialize variable b
        addi x16, x0, 0 #Initialize variable c
        jal x1, ex_4 #Jump to ex_4, and return back to this position after ex_4()
is finished

        add x16, x13, x0 #Load the returned value from Ex_4 in x16; c = ex_4(a,b)

        #[here your code]Print output_string
        addi x10, x0, 4 #function code for print_string
        la x11, output_string #set argument for print_string as output_string
        ecall
```

```
    addi x11, x16, 0 #Store value of x15 in x11 for print in ecall
    addi x10, x0, 1 #Store 1 in x10 for printing integers using ecall
    ecall
    j exit

ex_4:
#[here your code]Initialize res and i
    addi x5, x0, 100 #Initialize res
    addi x6, x0, 0 #Initialize i

while:
#[here your code]Calculate res += a and i = i + 1
    sub x5, x5, x14 #res -= a
    addi x6, x6, 1 #i = i + 1

#[here your code]If i < b, jump to while
    blt x6, x15, while

#[here your code]Place value of res in x13 for return
    add x13, x5, x0

    jalr x0, 0(x1) #Return back to caller

exit:
```

Result:

> c is equal to 52