CDA 3103 Computer Organization Homework

The problems in this assignment will be divided into two parts: Part A and Part B. You can select to complete this assignment with a partner (pair programming) or independent.

- If you select to complete this assignment independently, you only need to complete the problems in Part A and you can use problems in Part B for extra exercise.
- If you select pair programming, one student only needs to submit answers to problems in Part A and the other student only needs to submit the answers to problems in Part B.

How the pair programming works:

- Let us call two students in one team as student A and student B.
- Student A is the primary developer for problems in Part A and student B is the primary developer for problems in Part B.
- Student A should submit the final answers for Part A while student B should submit the final answers for Part B.
- For each problem, students A and B should use one computer for assembly code developing. The primary student develops the code and explains the code to the other student. If two students have different opinions, please continue further discussion, and learn from each other or other resources until come to the final conclusion.

For the whole assignment, pseudo-instructions are not allowed except "`j target_label`" and "`jr ra`". One suggestion for assembly programming problems is that you can include comments to one or a block of instructions.

Section I: Problems

Part A:

1. (3 points): Please circle your choice: Pair programming or Independent.
   If your choice is pair programming, please specify the name of your partner: and which part of problems for your submission:__________________.

2. (30 Points) Write a RISC-V assembly function to search a specified integer in an integer array. The function should take the base address of the array, the number of elements in the array, and the specified integer as function arguments. The function should return the index number of the first array entry that holds the specified value. If no array element is the specified value, it should return the value -1.

3. (12 points) Consider a RISC-V assembly function **`func1`**. **`func1`** has three passing arguments stored in registers **`a0`**, **`a1`** and **`a2`**, uses temporary registers **`t0-t3`** and saved registers **`s4-s10`**. **`func1`** needs to call **`func2`** and other functions may call **`func1`** also. **`func2`** has two passing arguments stored in registers **`a0`** and **`a1`**, respectively. In **`func1`**, after the program returns to **`func1`** from **`func2`**, the code needs the original values stored in registers **`t1`** and **`a0`** before it calls **`func2`**.

(a) How many words are the stack frames of function **func1**?

(b) Indicate which registers are stored on the stack of **func1**.

4. (55 Points) Consider the following C code snippet.

```
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
int findMinimum(int arr[], int N)
{
    // variable to store the index of minimum element
    int min_idx = 0;
    int min_E = arr[min_idx];
    // Traverse the given array
    for (int i = 1; i < N; i++) {
      // If current element is smaller than min_idx then update it
        if (arr[i] < min_E) {
            min_idx = i;
            min_E = arr[min_idx];
        }
      }
    return min_idx;
}
/* Function to sort an array using selection sort*/
 void selectionSort(int arr[], int n)
 {
     int i, min_idx;
     // One by one move boundary of unsorted subarray
     for (i = 0; i < n-1; i++)
     {
         // Find the minimum element in unsorted array
         min_idx = findMinimum(&arr[i], n-i);
         // Swap the found minimum element with the first element
         if(min_idx != 0)
             swap(&arr[min_idx+i], &arr[i]);
     }
 }
```

(a) [45 points] Implement the C code snippet in RISC-V assembly language. Use **s0 and s1** to hold the variable **i, and min_idx** in the function **selectionSort**. Be sure to handle the stack pointer appropriately. Clearly comment on your code.

(b) [10 points] Assume that the **selectionSort** is the function called. Draw the status of the stack before calling **selectionSort** and during each function call. Indicate stack addresses and names of registers and variables stored on the stack; mark the location of **sp**; and clearly mark each stack frame. Assume the **sp** starts at **0x8000**.

Section I: Problems

Part B:

1. (3 points): Please circle your choice: Pair programming or Independent.
   If your choice is pair programming, please specify the name of your partner: and which part of problems for your submission:_________________.

2. (30 Points) Write a RISC-V assembly function to find the length of a string. The function should take the base address of the string and return the length of the string.

3. (12 points) Consider a RISC-V assembly function **func1**. **func1** has five passing arguments stored in registers **a0, a1**, **a2**, **a3** and **a4**, uses temporary registers **t0-t2** and saved registers **s0-s6**. **func1** needs to call **func2** and other functions may call **func1** also. **func2** has three passing arguments stored in registers **a0-a2**. In **func1**, after the program returns from **func2,** the code needs the values stored in registers **t0**, **t1**, **a0** and **a1** before it calls **func2**.

   (a) How many words are the stack frames of function **func1**?

   (b) Indicate which registers should be stored on the stack of **func1**.

   4. (55 Points) Write a program called **print_Tri_Sqr** to a triangle or a square of stars ("*") to the monitor. The function **print_Tri_Sqr** has two passing arguments **a0** and **a1**. **a0** stores the print option (**a0 = 0**, print a square; **a0 = 1**, it prints a triangle) and **a1** stores the size of the shape.

```
* * * * *          *
* * * * *          * *
* * * * *          * * *
* * * * *          * * * *
* * * * *          * * * * *
```

   Write a subroutine **print_Sqr** to print a square and write a subroutine **print_tri** to print a triangle. Both subroutines **print_sqr** and **print_tri** have one single argument **a0**, which stores the size of the shape and use another subroutine **starline** that writes a line of a given number of stars. The **starline** subroutine is given as follows:

```
# a0 = size = n
starline:    add t0, zero, zero        # t0 = 0
             add t1, a0, zero          # t1 = size
```

```
            addi a7, zero, 11
            addi a0, zero, '*'
Loop_line:  bge t0, t1, Exit_line
            ecall
            addi t0, t0, 1
            j Loop_line
Exit_line:  addi a0, zero, '\n'
            ecall
            jr ra
```

(a) [45 points] Implement functions **print_Tri_Sqr**, **print_sqr** and **print_tri** in RISC-V assembly language. Be sure to handle the stack pointer appropriately. Clearly comment your code.

(b) [10 points] Assume that the **print_Tri_Sqr** is called two times. The user prints a square, then prints a triangle. Draw the status of the stack before calling **print_Tri_Sqr** and during each function call for these two function calls. Indicate stack addresses and names of registers and variables stored on the stack; mark the location of **sp**; and clearly mark each stack frame. Assume the **sp** starts at `0x8000`.

```
ADDI a0, x0, x0

ADDI a1, x0, 8

JAL ra, print_Tri_Sqr

ADDI a0, x0, 1

ADDI a1, x0, 5

JAL ra, print_Tri_Sqr
```

Section II: Submission Requirements

The following requirements are for electronic submission via Canvas.

- Your solutions must be in a single file with a file name yourname-module4-assignment-II.
- Upload the file by following the link where you download the homework description on Canvas.
- If scanned from hand-written copies, then the writing must be legible, or loss of credits may occur.
- Only submissions via the link on Canvas where this description is downloaded are graded. Submissions to any other locations on Canvas will be ignored.