Assignment-3

C to RISC-V

SUBMIT A PDF WITH YOUR CODE, SCREENSHOT FROM VENUS OUTPUT TAB, AND ANALYSIS IF YOU WISH TO WRITE.

Deadline: 17 September 2023 5:00 PM **[Maximum Marks:12x5 = 60]**

1. Printing a string via different methods. Suppose we have a print function that prints a character via assuming that the char is stored in the lower 8 bits of any of the 'a' registers. However the function below prints one character at a time.

    a. So now convert the following C code to RISCV while keeping the while loop structure.

```
void print_string(char *p){
    while(*p != '\0') {
        print(*p);
        p++;
    }
}
```

    b. Now do the same thing keeping the for loop structure

```
void print_string(char *p, int length){
    for(int i =0; i<length; i++) {
        print(*p);
    }
}
```

    c. Now repeat by keeping the do-while loop structure

```
void print_string(char *p){
    if(!(*p)) {
        return;
    }
    do {
        print(*p);
        p++;
    } while(*p);
}
```

2. Implement a recursive version for printing Fibonacci sequence in RISCV. Apply a well known trick to save intermediate results so that the resulting version is quicker. Provide an analysis of how you implemented this.

3. Write a RISCV code that will do the following function. Assume that the short is 16 bits wide. Understand that the main thing that you will have to implement here is to make sure that when the variable x reaches the limit, it will be reset to 0.

```
int next_half_int(){
        static short x=0;
        return x++;
}
```

4. Remember the linked list RISCV code discussed in class and slides. A sample of the same is here below:

```
struct node {
  int *arr;
  int size;
  struct node *next;
};
```

Now we wish to implement a apply function that applies a function f that is passed to map to all the elements of the list:

```
void apply(struct node *head, int (*f)(int)) {
        if (!head) { return; }
            for (int i = 0; i < head->size; i++) {
                    head->arr[i] = f(head->arr[i]);
             }
             map(head->next, f);
}
```

A sample file of most of the code is provided to you under map_lists.s. All you have to do is complete the map function body.

5. Consider a function f defined on the integer set {-3, -2, -1, 0, 1, 2, 3}. The function evaluations are as follows:

f(-3) = 11
f(-2) = 21
f(-1) = 45
f(0) = −3
f(1) = 12
f(2) = 62
f(3) = 16

Check the functions.s file provided. Finish function code with the restriction that YOU CANNOT USE ANY BRANCH OR JUMP INSTRUCTIONS. All the integers are defined in the .data section which you can use.