CSCI 212

Midterm Exam

Open Everything

Instructions:

This exam is open book and notes. It consists of a single program to be written in ARM Assembly . You may take the exam any time between now and end of day Tuesday (3/29/22). You can write the program using any editor, but the final product must compile and run on the Raspberry Pi (i.e., I should be able to compile and run it on the Pi 3B). Please submit the source code (i.e., all the .s files) and screenshots of your program output. Also, please do not write the code in C then submit the auto-generated assembler code from C. The code must be written in ARM assembler from scratch.

Rubric:

- Code compiles without any error (50%)

- Code runs and produces correct results for all test cases (40%). Points will be deducted if your code fail for some cases.

- Screen shot of the outputs (10%)

Write an ARM assembler program from scratch (**i.e., not autogenerate from C**) that generates random roll of a pair of dice some number of times (user input) and prints out a table showing how many times the sum 2 was rolled, 3 was rolled, etc... up to 12. You may use the random number generator (rand) to generate the random number. Please use the srand function to set the random number seed to the current time similar to the way we discussed in lecture. Your code should use the stack to store the number of times each sum is rolled, e.g. a stack location to store 2, a stack location to store 3, etc.

Ex: Your output should look something like this

>> input the number of times to roll the pair of dice: 2000

The frequency counts are:

Sum of the dice	Number of Occurences
2	50
3	60
4	95
5	120
6	200

7	300
8	
9	
10	ETC...
11	
12	

Your program should contain at a minimum the following functions:

main.s // Set up your stack and call the functions

get_rolls.s; // Get the number of times to roll the dice

roll_die.s; // roll a random number from 1 to 6 and return it

// You will need to call generate_random and roll_die twice to get the numbers for the two dice

update_count.s: // update the stack location containing the pair just rolled

// Put the function calls in a for-loop that runs n times and store the sum of the two dice onto the right location on the stack

print_table.s; // print the table as shown below

Makefile // write a makefile to compile your code

HINT:

This is an example of what the main function may look like (remember that you must write everything in ARM assembly):

```
main:
    @ Allocate the number of memory locations on the stack needed
    @ call get_roll()
    bl get_roll  @ return number of rolls in r0
    loop from 1 to num_rolls
        bl roll_die
        bl roll_die     // call roll_die twice, make sure you add the two numbers together
        @ Determine the location on the stack corresponding to the sum
        @ call update_count(address on the stack)
```

@ function update_count takes the stack address as input & increment it by 1

mov r0, stack address

bl update_count

repeat loop

@ call print_table(address of first stack location containing the sum)

@ Note that you should allocate consecutive stack locations to store the sum

@ so print_table just needs the address of the first location, it should iterate

@ through the rest.