CIT 593 Homework 7: LC-4 Assembly

Due Date: Friday 11/11 @11:59pm via canvas upload ONLY

You will receive 1 extra credit point for each business day you turn this HW in early!

However only a maximum of 10 extra credit points will be awarded in total

The intention of this extra credit reward is to encourage you complete this HW BEFORE the midterm, to help you obtain a better understanding of the ISA.

READING: Chapters 6-7 of the book will be helpful, but realize they refer to the LC3 and we are using the LC4; if you are new to programming you will still find these chapters helpful.

Assigned Problems (to be done individually, NOT group work):

The problems assigned below require the use of the PennSim LC4 Assembler/Simulator. This assembler/simulator is written in java, which you can download from canvas (Files->Resources->LC4->PennSim->**PennSim.jar**) and run on your personal machine. You can then write the programs assigned below in a textfile, assemble and execute them using PennSim as shown in lecture. A manual with examples of how to assemble/load/run sample programs is also available on canvas called: **PennSimStartGuide.zip.** ***Consult the manual prior to posting piazza questions.***

1) IF STATEMENTS AND WHILE LOOPS IN ASSEMBLY

In this problem you will write a small program that tests whether a given integer is a prime number or not. Let's call the number we are testing A and let's assume that when the program starts, A is already in R0 (you'll need to load it with a script file)– at the end of your program R1 should contain 1 if the number is prime and 0 if it is not.

Here is the pseudo-code for the algorithm:

```
if (A <= 1) {
   PRIME_FLAG = 0;
   GOTO END
}

B = 2;
PRIME_FLAG = 1;   // assume # is prime until we prove otherwise

while (B*B <= A) {
   if (A % B == 0) { // checks if A is divisible by B
                     //note: % means "modulo division"
         PRIME_FLAG = 0;
         GOTO END
   }
   B = B+1
}

END
```

POINTS WILL BE DEDUCTED IF YOUR CODE IS NOT COMMENTED!

For this problem you should turn in two files: **prime.asm** and **prime_script.asm**

2) WORKING WITH DATA MEMORY IN ASSEMBLY

For this problem you will write a short assembly program that will compute the first 20 numbers in the Fibonacci sequence. Then verify that it assembles and loads in the PennSim simulator. A pseudo-code version of the required algorithm is shown below:

```
F_0 = 0
F_1 = 1;

i = 2;
while (i < 20) {
   F_i = F_(i-1) + F_(i-2);
   i = i+1
}
```

More specifically your program should write the first 20 Fibonacci numbers out to data memory starting at address x4000. At the end of your program the addresses x4000 through x4013 will contain the first 20 Fibonacci numbers starting at 0. You can look at the ***sum_numbers.asm***, from lecture #8, for an example of a program that accesses data memory.

Big Hint. You will probably find it convenient to maintain a variable that points to the address in data memory you are going to write to next and update this on each iteration. You may also find it useful to use the offset field in the LDR instructions to read values that are near to the current store address.

POINTS WILL BE DEDUCTED IF YOUR CODE IS NOT COMMENTED!

For this part you should turn in two files: fibonacci.asm and fibonacci_script.txt

CIT 593 Homework 7: LC-4 Assembly

3) EXTRA CREDIT (5 points): SUBROUTINES IN ASSEMBLY

For this problem you will write a short assembly subroutine that computes the integer square root of an argument that is passed to the function (called A in the pseudocode below), stores it in a variable B and then returns B after the subroutine is complete. You'll notice that if A is negative, the subroutine will return a -1.

```
SUBROUTINE SQR_ROOT (A) {
     B=0
     if (A >= 0) {
          while (B*B <= A) {
               B = B+1
          }
     }
     B = B-1
     RETURN B ;
}
```

For your implementation, call the subroutine: **SQR_ROOT**. Use register R0 to hold the input argument: **A**. Use register R1 to hold the return value: **B**. After implementing the subroutine, write additional code that "calls" your subroutine. *Make certain that if your code continues to run it doesn't accidentally run the SQR_ROOT subroutine again!* Test it initially with A=9 and check if your subroutine correctly computes 3 as the answer. Test your code with other #s to ensure it is working and ensure negative #s return a value of -1 from the subroutine.

POINTS WILL BE DEDUCTED IF YOUR CODE IS NOT COMMENTED!

For this part you should turn in two files: sqrt.asm and sqrt_script.txt

CIT 593 Homework 7: LC-4 Assembly

Directions on how to submit your work:

- Create a single zip file called: **LAST_FIRST_HW#.zip**
- The zip file should contain the 4 files named in this assignment.
- There should not be any sub-directories within your zip file.

As an example, I would turn in the following SINGLE zip file:

FARMER_THOMAS_HW07.zip

This single zip file would contain at least 4 files, and at most 6 files :

prime.asm
prime_script.txt
fibonacci.asm
fibonacci_script.txt
sqrt.asm
sqrt_script.txt

You will then upload ONLY 1 file to canvas: **FARMER_THOMAS_HW07.zip**

- DO NOT TURN IN ANY **.obj** or **PennSim.jar** files!

- Make certain that you submit the latest versions of your code.

- Submitting using any other compression type (.RAR, TAR, GZIP) will be rejected.

Paper/Email submissions will not be accepted for this assignment.