Recursion in Assembly

Programming Lab #6
CEG 3310/5310: Computer Organization

PURPOSE

In this lab you will learn how to implement recursive subroutines in assembly.

ASSIGNMENT

You will be implementing recursive subroutines in assembly. A common exercise in recursive programming is displaying the Fibonacci Sequence. The Fibonacci sequence is defined as:

F(n) = F(n-1) + F(n-2)

Where, F(0) = 0 and F(1) = 1

For example, the first 7 numbers in the Fibonacci Sequence are:

F(0)	F(1)	F(2)	F(3)	F(4)	F(5)	F(6)
0	1	1	2	3	5	8

Notice how F(2) = F(1) + F(0) = 0 + 1 = 1 and how F(6) = F(5) + F(4) = 5 + 3 = 8.

You will have to implement a recursive subroutine that returns the desired number in the Fibonacci Sequence. An example output of your completed lab should look like the following:

```
Please enter a number n: 9

F(14) = 34
```

Another example is:

```
Please enter a number n: 6

F(10) = 8
```

This can be accomplished by implementing a recursive function, with a properly implemented general and base case. The general case would be F(n) = F(n-1) + F(n-2) and the base cases are F(0) = 0 and F(1) = 1. Notice, in order to calculate F(n), you must call F(n-1) and F(n-2), this will be implemented using recursion. For a value of n = 3, your subroutines will be call each other in this manner:

```
fibonacci(3) calls fibonacci(2) + fibonacci(1)

fibonacci(2) calls fibonacci(1) + fibonacci(0)

fibonacci(1) returns 1 to fibonacci(2) and fibonacci(3)

fibonacci(0) returns 0 to fibonacci(2)

fibonacci(2) returns 1 + 0 = 1 to fibonacci(3)

fibonacci(3) returns 1 + 1 = 2 to main()

main() displays “F(3) = 2” to the user
```

For reference, to implement this in C code, the fibonacci function would look like:

```
int fibonacci(int n)
{
    if(n == 1)
    {
        return 1;
    }
    else if(n == 0)
    {
        return 0;
    }
    else
    {
        return fibonacci(n-1) + fibonacci(n-2);
    }
}
```

IMPLEMENTATION

Write your assembly code in the “lab-6.asm” file provided:

- Implement the main function in assembly.
- Implement the fibonacci function as an LC3 assembly subroutine. This function must be recursive, that is, it calls itself until the base case is reached.
- Maintain a proper runtime stack. This runtime stack should allow for multiple fibonacci subroutine calls, passing inputs to each function call, and returning outputs properly without losing any data or corrupting data.

GRADING

- Main() takes in a user input for the value n
  - 5 points
- The fibonacci subroutine has a properly implemented general case (F(n) = F(n-1) + F(n-2))
  - 5 points
- The fibonacci subroutine has properly implemented base cases (F(0) = 0 and F(1) = 1)
  - 5 points
- The first fibonacci subroutine call returns the correct output to main() from n = 0 to n = 9
  - 5 points
- The runtime stack is properly utilized and the total result of F(n) is stored in mains return value location at x5013
  - 5 points
- Your assembly program outputs the Fibonacci sequence number n correctly as formatted below:

  ```
  Please enter a number n: 9

  F(14) = 34
  ```

  - 5 points
- Total
  - 30 points