Name: Sean Bartholomew
Date: 20160520
Current Module: Assembly
Project Name: Fibonacci

Project Goals:

The goal of the project is to create a program entirely in assembly language. The program will calculate the 0-300th numbers in the Fibonacci sequence.

Considerations:

How to assembly.
How to calculate the nth term of the Fibonacci sequence. (recursive vs non)
How to handle invalid data.
How to get user input.
How to store the numbers.
Which registers to use.

Initial Design:

In the initial design I attempted to implement a recursive solution to the problem. It did not work correctly. The second design of the program utilized multiple registers (4 per number) to store two numbers in the sequence. The program accepts one command line argument that must be an integer. The argument is the nth term of the sequence to calculate. The program will output the calculated number in hexadecimal.

Data Flow:

Execution begins with passing the command line argument into strtol to convert it to an integer. If strtol indicates that an error has occurred, the program exits. Otherwise the number is stored in rdi to be utilized as the count variable. There are two numbers (a and b) that are stored in the program. Number a is stored in registers r8-r11 and number b is stored in r12-r15. A is initialized to 0 and b is initialized to 1. The count variable is stored in rdi. The program calculated the nth term of the sequence in a while loop. The loop adds a and b together, and stores the result in a. It then swaps a and b. and loops until the count variable is equal to zero. The result is always in stored in a. After the loop exits the data in r8-r11 is moved into the appropriate registers to call printf, and the result is printed to the screen.

Communication Protocol:

CMD line options:

No options, one mandatory integer variable.

Signals handled:

None.

Potential Pitfalls:

Liam: I'm sure he will find a way to break it. Correctly handling user input. Correctly calculating the nth term of the sequence.

Test Plan:

User Test / Test Cases:

Input data larger than buffer
Non ASKI characters
Input a string instead of an integer.
Negative values.
Multiple command line arguments.

Expected Result:

I expect the behavior outlined in the initial design to be appropriately implemented. The program will not seg-fault.

Conclusion:

The program works as intended. It does calculate one additional term of the Fibonacci sequence. This is done to avoid writing special case scenarios for the first two numbers in the sequence. If I had more time I would test to see what is more efficient: utilizing jumps for the first two numbers or simply doing the extra calculation. In an event doing the extra calculation make the code easier to follow as every number is handled in exactly the same way.