Lab 2

Introduction

This laboratory procedure introduces the use of subroutines and the stack. You will write a recursive subroutine that computes a *Fibonacci number*, using the stack for temporary data storage. You will also use some simple techniques to determine the amount of stack memory used and the execution time of your code.

Background

The Fibonacci numbers are a number sequence that has many applications in the field of mathematics and also appears in nature. The sequence consists of the values $0, 1, 1, 2, 3, 5, 8, 13, \cdots$. After the first two numbers, each successive Fibonacci number $F(N)$ is the sum of the previous two Fibonacci numbers. Another way of describing this sequence is

$$\begin{aligned} F(0) &= 0 \\ F(1) &= 1 \\ F(N) &= F(N-1) + F(N-2) \qquad \text{for } N > 1 \end{aligned}$$

With this definition we see that we can compute $F(N)$ *recursively*. Suppose we have a subroutine that computes $F(N)$. If the input argument to the subroutine is 0 or 1 then the subroutine simply returns the same value. If the input argument is greater than 1, then the subroutine *calls itself* twice with arguments $N-1$ and $N-2$. The subroutine keeps calling itself with smaller and smaller arguments, until the arguments become 0 or 1 and the subroutine can directly return a value.

All of the variables needed for the subroutine must go on the stack, so that every time the subroutine is called it gets a new set of memory locations that it can use.

Procedure

Your task for this lab is to write an assembly-language subroutine that calculates the Nth Fibonacci number, given that N is a non-negative integer. Use an immediate value into a register as your N value. To test your output, you should change this number to demo to your instructor. The output should be the binary number on the leds.

Extra Credit

1. Calculate the time your algorithm takes using an oscope.

2. Use the push button as an input for N. Give the user an amount of time to push the button, and this could represent your N.