CMPS 351: Fall 2013 Program #3
Date Assigned: Thursday, October 17, 2013 Due: Tuesday, October 29, 2013

For this project, you will write an MIPS function to calculate the requested Fibonacci number. Your function should take an unsigned integer index for its parameter and recursively calculate the corresponding Fibonacci number. Fibonacci numbers are defined as $F(0) = 0$, $F(1) = 1$ and $F(x) = F(x-1) + F(x-2)$. An iterative solution is much faster, but will not fulfill the criteria for this project.

Your program will not need to interact with the console. Instead, you will use a test suite that has been provided for you. The test suite will call your function (fibonacci) with the parameter ($a0 contains the index) and will wait for your function to return its results in $v0. The test suite will also tell you whether your function has correctly calculated the results. Finally, the test suite will also test to make sure that you are following assembly language conventions.

You should attach your code to the test suite for testing by running either:
Windows: copy /Y <Fibonacci Function Name>.asm + "Program #3 - Test Suite.asm" <output>.asm
Unix: cat <Fibonacci Function Name>.asm "Program #3 - Test Suite.asm" > <output>.asm
The result of either of these commands is a file that contains both sets of code, so that SPIM can load all of it at once. Your final submission should only include your function and not the test suite.

Your program should include appropriate comments indicating what the code should be doing and what registers are being used for. Please include your name and CLID in the program headers and include your CLID in the file names. Your programs should be turned in through Moodle before class starts on the due date. You should test your programs using the QT-SPIM simulator to ensure their functionality before submitting them.

Expected output:

```
Test #1 passed.
Test #2 passed.
Test #3 passed.
Test #4 passed.
Test #5 passed.
Test #6 passed.
Test #7 passed.
Test #8 passed.
```

Objectives:

1. To introduce and practice building functions in the MIPS assembly language.
2. To introduce and practice using the stack in the MIPS assembly language.