Lab 04 - Fibs

Out: October 1 - 2, 2012

1 Introduction

Princess Fibonacci[1] has a pair of baby rabbits. Each rabbit takes a month to become an adult. Each pair of adult rabbits produces a new pair of baby rabbits every month. The princess wants to know how many pairs of adult rabbits there will be after n months. She notices that the number of adult pairs in a given month is equal to the number of adult pairs for the previous month, plus the number of baby rabbits from the previous month, which is equal to the number of adult pairs from the month before that.

She expresses this relationship as follows:

$f(0) = 0$

$f(1) = 1$

$f(n) = f(n-1) + f(n-2)$, $n \geq 2$

The resulting sequence $f(0)$, $f(1)$, $f(2)$, ... is known as the *Fibonacci Sequence*.

Here are the first few terms of the sequence:

n	0	1	2	3	4	5	6	7	8	9	10
$f(n)$	0	1	1	2	3	5	8	13	21	34	55

In this lab, you will be writing a function to compute the nth term of the sequence.

2 Assignment

You will be implementing the definition of the Fibonacci sequence given above as a recursive function, `fibs_recursive()`. This function takes a single unsigned integer argument, n, and returns the nth Fibonacci number. You do not need to worry about integer overflow.

Pseudocode for this function is as follows:

```
unsigned int fibs_recursive(unsigned int n):
    if n is 0, return 0
    if n is 1, return 1
    otherwise, return fibs_recursive(n-1) + fibs_recursive(n-2)
```

You will have to make sure your function complies with x86 stack discipline. See the x86 cheatsheet (handed out with last week's lab) or the textbook for more information.

[1] Also known as Leonardo of Pisa.

2.1 Handout

The lab handout contains the following files:

fibs.s: The assembly file in which you will be implementing your function. This file contains a skeleton for the function; your task is to implement the function body only.

fibs.h: A header file declaring the functions implemented in *fibs.s*.

main.c: A C file providing testing functionality.

Makefile: A makefile.

lab04.pdf: This document.

You only need to read and edit *fibs.s* to complete the lab.

3 Running Your Code

We have provided a makefile to compile the support code, link it with your code, and produce an executable. To build everything, use the command

```
make
```

This will create an executable, *fibs*, which you can then run with the command

```
./fibs n [...]
```

where n is one or more indices in the sequence. The support code will run your function and print the result. It will also print warning messages if your code does not deal with saved registers correctly.

To remove all compilation products, use the command

```
make clean
```

3.1 Debugging with GDB

The support code "sandboxes" your function by running it in a separate process. To use GDB to debug your code, enter the command `set follow-fork-mode child` before running the test program. You can then set a breakpoint on the function of your choosing and debug as usual.

3.2 Debugging with x86scope

If you are using x86scope, you will need to add command-line arguments to the run configuration. To do this, select the *Build* → *Configure* menu option and add the numbers you wish to use to test your program. The C code will parse these arguments to integers and pass them to your function.

4 Getting Checked Off

Once you have successfully implemented `fibs_recursive()`, call a TA over to have them check your work. If you do not complete the lab during your lab session, you can get credit for it by completing it on your own time and bringing it to TA hours before next weeks lab. Remember to read the course missive for information about course requirements and policies regarding labs and assignments.