Lab 04 - x86-64: fibs

Due: October 9, 2016

1 Introduction

Dory seems to be gaining more and more memories every day. Nemo has noticed that each day, Dory gains as many more memories as she gained in the last two days. So following his curious nature, Nemo wanted to know how many memories Dory was going to to gain the next day, so he talked to Hank to figure it out.

Hank created an expression to describe Dory's situation. That expression is 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.

Note - The Oct 5 lecture notes describe a recursive function in IA32, including the need to update stack pointers. This practice is no longer needed in X86-64. In particular, see slides 27 and 28.

2.1 Handout

Run the terminal command *cs033_install lab04* to install the files for this lab to your home directory. 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.

4 Getting Checked Off

Once you have successfully implemented, submit your work using the handin script.

```
33lab_checkoff lab04 .
```

Remember to read the course missive for information about course requirements and policies regarding labs and assignments.