CS 213, Fall 2000
Homework Assignment H1, Part A
Assigned: Sept. 12, Due: Mon., Sept. 18, 11:59PM

Randy Bryant (`Randy.Bryant@cs.cmu.edu`) is the lead person for this assignment.

The purpose of this assignment is to learn IA32/Linux assembly language and to become familiar with how C code is translated into assembly. You will do this by looking at a series of assembly language functions and decompiling them to find the C source code that produced them. Reverse engineering this code will improve your understanding of both the C constructs and the assembly code.

Introduction

In this assignment you will be given the assembly code file for a set of functions. Your task is to derive C source code that compiles into code that is functionally equivalent. In fact, you should strive to make your code produce assembly code that is identical to the supplied assembly code. None of the functions requires more than a few lines of code.

When you are trying to figure out what a given functions does, try creating a small example to see what code the compiler emits. If you can create a series of small functions that produce part of the answer, you can then piece them together to create a solution.

Logistics

You must work alone on this assignment. The only "hand-in" will be electronic. Any clarifications and revisions to the assignment will be posted on Web page `assigns.html` in the class WWW directory.

All files for this assignment are in the directory:

```
/afs/cs.cmu.edu/academic/class/15213-f00/H1a
```

You will want to do your work on one of the class "fish" machines to be sure that you are using the correct version of the GCC compiler. See the class WWW pages for more information on these machines.

First create a (protected) directory to work in, and copy our template code using the command:

```
tar -xvf /afs/cs.cmu.edu/academic/class/15213-f00/H1a/H1a.tar
```

This will create a variety of files, including one named `probs.c`. In this assignment you will only modify this file.

Your first step should be to fill in your name and Andrew ID in the structure at the beginning of this file.

Your task is to fill in the bodies of six functions in `probs.c`, named `arith`, `mem`, `optarith`, `ifcode`, `whilecode`, and `forcode`, respectively. Your goal is to make them have the same functionality as the corresponding functions in the file `probs-solve.s`.

The original C code for these functions has the following characteristics:

`arith`**:** Some simple arithmetic operations.

`mem`**:** Some memory referencing through pointers.

`optarith`**:** Some arithmetic operations that get optimized by the compiler.

`ifcode`**:** A conditional expression.

`whilecode`**:** A while loop.

`forcode`**:** A for loop.

Although you could brute force your solution by writing C code that uses goto's and things like that, try to write the cleanest and most abstract C code you can.

Evaluation

You will only get credit for problems in which you are able to exactly match the functionality of the assembly code versions. The six problems count one point apiece. Use the command `make atest` to generate a program that will run your functions against the original functions on some test data. See the file `README` for documentation on this program.

Hand In

Hand in your solution using the command

```
make handin NAME=yourname VERSION=XX
```

where `yourname` is your Andrew ID, and `XX` is the version number, i.e., 2, 3, ... Only your highest numbered version submitted before the deadline will be graded.