Computer Architecture
CS 429, Fall 2012
Unique Numbers: 52873, 52875, 52880, 52885

Lab Assignment 0: Introduction to C
Assigned: Wednesday, September 5, 2012
Due: Wednesday, September 19, 2012, 2:00 pm

Introduction

The purpose of this assignment is to become more familiar with the C programming language. This lab asks you to write C code that solves various prgramming problems using a subset of C.

Logistics

You are to work alone on this project, but you may discuss programing tricks and bit-manipulation code with your peers. The entire "hand-in" will be electronic. Any clarifications and revisions to the assignment will be posted on the course Web page.

Hand Out Instructions

> You will find the file `casm-handout.tar` referenced on the homework page of the class website. You will need to download this file so you can use its contents.

Start by copying `casm-handout.tar` to an empty, protected directory in which you plan to do your work. Make sure that your directory has only permissions to allow you to access your work. For example, you can create an empty directory by issuing the command `mkdir -m 700 lab0`, which will create a new directory immediately in the directory where you are "standing." Then, give the command: `tar xvf casm-handout.tar`. This will cause several files to be unpacked in the directory. The only file you will be modifying and turning in is `problems.c`.

Looking at the file `problems.c` you'll notice a place to include your name and UTCS UserID into which you should insert the requested identifying information about yourself. Do this right away so you don't forget.

The `problems.c` file also contains a skeleton for your code. Your assignment is to implement the function descriptions by adding code to the file `problems.c`. In most cases, you are to use only *straight-line* code (i.e., no loops, no IF statements, are OK) and a limited number of C arithmetic and logical operators. Specifically, you are *only* allowed to use the following nine operators:

```
! ~ & ^ | + << >> <=
```

For this laboratory, a "while" statement may be included where you may modify the "while" test argument as well as the body of the "while" statement. You are not allowed to use any constants longer than 8 bits. See the comments at the beginning of the file `problems.c` for detailed rules and a discussion of the desired coding style. You should compile the updated version of the file `problems.c` with the command `gcc -O2 -m32 problems.c`.

Coding Challenge

You have been given a number of templates for a variety of integer coding problems. In the `main` procedure, you may wish to implement a specification for your code. We will certainly check your code on a very large (likely exhaustive) number of tests. Also, for some of the problems, you are expected to include the number of X86 instructions produced by the "gcc" compiler for each routine where there is a comment: "Number of X86 instructions:". To get the number of instructions, you need to compile `problems.c` with the `-S` flag; that is, issue the `gcc -S -O2 -m32 problems.c` command and inspect the assembler produced.

Please do not write lots of code! Pay attention to the maximum number of statements allowed. The main thing you need to do is to think carefully about what is asked and then code it in a dense manner. These problems should be fairly straight forward, but they will cause you to think carefully about using C to answer simple questions. You will only submit the updated `problems.c` file – insert your code there.

Evaluation

Your code will be compiled with GCC and run and tested on one of the public Linux machines. Your score will be computed out of a maximum of 100 points based on the following distribution:

90 Correctness of code running on one of the public Linux machines.

10 Style points, based on your instructor's subjective evaluation of the quality of your solutions and your comments.

Regarding performance, our main concern at this point in the course is that you can get the right answer. However, we want to instill in you a sense of keeping things as short and simple as you can.

We have reserved 10 points for a subjective evaluation of the style of your solutions and your commenting. Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they need not be extensive.

Advice

You are welcome to do your code development using any system or compiler you choose. Just make sure that the version you turn in compiles and runs correctly on our UT CS public Linux machines. If it doesn't compile, we can't grade it.

Hand In Instructions

Please follow the instructions below for turning in your work.

- Make sure you have included your identifying information in your file `problems.c`.
- Remove any extraneous print statements and get rid of any compiler warnings.
- To handin your `problems.c` file, type:

  ```
  turnin --submit beltagy lab0 problems.c
  ```

  where `problems.c` is the file containing your solutions.
- After turning something in, if you discover a mistake and want to submit a revised copy, type

  ```
  turnin --submit beltagy lab0 problems.c
  ```

  which will replace your original copy with a new one.