CS 4400: Computer Systems
Fall 2010

Lab Assignment 1: Manipulating Bits

Part 1: Integer Puzzles

Due Date: Wednesday, September 8 at 11:59p

Introduction

The purpose of this assignment is to become more familiar with bit-level representations and manipulations. You'll do this by solving a series of programming "puzzles." Many of these puzzles are quite artificial, but you'll find yourself thinking much more about bits in working your way through them.

Logistics

You will work on this lab individually. The only hand-in will be electronic through the CADE `handin` tool. Type `man handin` to the shell on any of the CADE machines for more information. Any clarifications and revisions to the assignment will be sent out via the class mailing list. Questions may be emailed to `teach-cs4400@eng.utah.edu`, or asked in person (during class, instructor office hours, or TA consulting hours). *BEWARE*: We will not tell you how to solve these puzzles since the whole point is to get you to think about how to solve these kinds of problems. However, we will help clear up confusions and attempt to guide your thinking into more productive directions.

Getting The Files

Go to the class web page for Lab 1: Part 1 and save the file `lab1_part1.tar.gz`.

Copy the `tar.gz` file to a protected directory in which you plan to do your work. A protected directory is one that cannot be read by users other than yourself. You can create such a directory like this:

```
mkdir my_dir
chmod go-rwx my_dir
```

Type `man chmod` to get more details about changing UNIX permissions.

Give the command:

```
tar -zxvf lab1_part1.tar.gz
```

This will cause a number of files to be unpacked in your protected directory. The **ONLY** file you will be modifying and turning in is `bits.c`.

The file `btest.c` allows you to evaluate the functional correctness of your code. The file `README` contains additional documentation about `btest`. Use the command `make btest` to generate the

test code and run it with the command `./btest`. The file `dlc` is a compiler binary that you can use to check your solutions for compliance with the coding rules. The remaining files are used to build `btest`.

Looking at the file `bits.c` you'll notice a C structure `student` into which you should insert the requested identifying information—namely your name and CADE login. *Do this right away so you don't forget.*

The `bits.c` file also contains a skeleton for each of the 10 programming puzzles. Your assignment is to complete each function skeleton using only *straightline* code (i.e., no loops or conditionals) and a limited number of C arithmetic and logical operators. Specifically, you are *only* allowed to use the following eight operators:

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

Also, you are not allowed to use any constants longer than 8 bits. See the comments in `bits.c` for detailed rules and a discussion of the desired coding style.

Although the lab1-* machines are 64 bit, do NOT use any 64 bit types. In other words, use only the `int` type, not the `long` type.

Evaluation

Your code will be compiled with GCC and run and tested on one of the class machines. Your score will be computed out of a maximum of 45 points based on the following distribution:

25 Correctness of code running on one of the class machines.

20 Performance of code, based on number of operators used in each function.

The 10 puzzles you must solve have been given a difficulty rating between 1 and 4, such that their weighted sum totals to 25. We will evaluate your functions using the test arguments in `btest.c`. You will get full credit for a puzzle if it passes all of the tests performed by `btest.c`, half credit if it fails one test, and no credit otherwise.

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. Furthermore, some of the puzzles can be solved by brute force, but we want you to be cleverer. Thus, for each function we've established a maximum number of operators that you are allowed to use for each function. This limit is very generous and is designed only to catch egregiously inefficient solutions. You will receive two points for each function that satisfies the operator limit.

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 class machines (`lab1`). If it doesn't compile, we can't grade it.

The `dlc` program, a modified version of an ANSI C compiler, will be used to check your programs for compliance with the coding style rules. The typical usage is

```
./dlc bits.c
```

Type `./dlc -help` for a list of command line options. The README file is also helpful. Some notes on `dlc`:

- The `dlc` program runs silently unless it detects a problem.
- Don't include `<stdio.h>` in your `bits.c` file, as it confuses `dlc` and results in some non-intuitive error messages.

Check the file `README` for documentation on running the `btest` program. You'll find it helpful to work through the functions one at a time, testing each one as you go. You can use the `-f` flag to instruct `btest` to test only a single function, e.g., `./btest -f is_equal`.

Submission Instructions

When you are ready to hand in your code you simply type:

```
handin cs4400 lab1_part1 bits.c
```

If you subsequently find a better solution prior to the deadline then just resubmit and the old version will be overwritten. The file dates will be used to determine whether you handed it in on time or not. Submissions will close at 11:59p three days after the due date, and at that point no more submissions will be accepted.

Hint: It is a very bad idea to get behind in this class. The pace is rather continuous. Late penalties will have a significant detrimental effect on your grade, if previous years are any indication.

- Make sure you have included your identifying information in your file `bits.c`.
- Remove any extraneous print statements.
- Make sure that `dlc` does not complain about your code in any way and that you have followed all the rules in this handout. *If* `dlc` *does not run silently, your assignment may not be graded.*
- Make sure you have not included `<stdio.h>` as it will confuse `dlc`.

"Beat the Prof" Contest (Optional)

For fun, we are running an online contest where you can try to match or beat the number of operators required by the instructor's solution.

To submit your entry, copy the `submit.pl` script (located in `/home/cs4400/contest/`) to the directory that contains your `bit.c` solution. To run the script, simply type:

```
./submit.pl
```

If you want to be anonymous, modify the identifying information in the `student` struct of your `bits.c` file before submitting your entry.

To check the status of the contest, go to `http://www.eng.utah.edu/~bomb/contest.html`.

Participating in the contest is optional and does not affect your Lab 1 grade. You must use `handin` to submit your solution for grading, regardless of whether you submit your solution to the contest.