Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Fall Term 2012

SYSTEMS PROGRAMMING AND COMPUTER ARCHITECTURE
Assignment 1: Manipulating Bits

Assigned on: **27th Sep 2012**
Due by: **4th Oct 2012**

Introduction

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

Logistics

This is an individual project. Hand-in for this assignment is going to be using Subversion (SVN). Please see the *Hand In Instructions* section for more information on how to submit your solution file. Any clarifications and revisions to the assignment will be posted on the course Web page.

Hand Out Instructions

Download the datalab handout tar ball (`datalab-handout.tar`) from the course Web page.

Start by copying `datalab-handout.tar` to a directory in which you plan to do your work. Then give the command: `tar xvf datalab-handout.tar`. This will cause a number of files to be unpacked in the 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`. Section *Advice* gives some more information on compiler requirements.

Looking at the file `bits.c` you'll notice a C structure `student` into which you should insert the requested identifying information about you. Do this right away so you do not forget. Nickname field in this structure will be used to display your results on the course Web site.

The `bits.c` file also contains a skeleton for each of the 15 programming puzzles. Your assignment is to complete each function skeleton using only *straightline* code (i.e., no loops or conditionals)

Name	Description	Rating	Max Ops
`bitNor(x,y)`	∼`(x|y)` using only `&` and ∼	1	8
`bitXor(x,y)`	`^` using only `&` and ∼	2	14
`isNotEqual(x,y)`	`x != y?`	2	6
`getByte(x,n)`	Extract byte `n` from `x`	2	6
`copyLSB(x)`	Set all bits to LSB of `x`	2	5
`reverseBytes(x)`	Reverse the bytes of `x`	3	25
`bitCount(x)`	Count number of 1's in `x`	4	40
`bang(x)`	Compute `!x` without using `!` operator	4	12
`leastBitPos(x)`	Mark least significant 1 bit	4	6

Table 1: Bit-Level Manipulation Functions.

and a limited number of C arithmetic and logical operators. Specifically, you are *only* allowed to use the following eight operators: ∼ `! & ^ | + << >>`

A few of the functions further restrict this list. 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.

Part I: Bit manipulations

Table 1 describes a set of functions that manipulate and test sets of bits. The "Rating" field gives the difficulty rating (the number of points) for the puzzle, and the "Max ops" field gives the maximum number of operators you are allowed to use to implement each function.

Function `bitNor` computes the NOR function. That is, when applied to arguments `x` and `y`, it returns ∼`(x|y)`. You may only use the operators `&` and ∼. Function `bitXor` should duplicate the behavior of the bit operation `^`, using only the operations `&` and ∼.

Function `isNotEqual` compares `x` to `y` for inequality. As with all *predicate* operations, it should return `1` if the tested condition holds and `0` otherwise.

Function `getByte` extracts a byte from a word. The bytes within a word are ordered from 0 (least significant) to 3 (most significant). Function `copyLSB` replicates a copy of the least significant bit in all 32 bits of the result. Function `reverseBytes` reverses the bytes of `x` .

Function `bitCount` returns a count of the number of 1's in the argument. Function `bang` computes logical negation without using the `!` operator. Function `leastBitPos` generates a mask consisting of a single bit marking the position of the least significant one bit in the argument. If the argument equals 0, it returns 0.

Part II: Two's Complement Arithmetic

Table 2 describes a set of functions that make use of the two's complement representation of integers.

Function `tmax` returns the largest integer.

Function `isNonNegative` determines whether `x` is greater than or equal to 0.

Function `isLessOrEqual` determines whether `x` is less than or equal to `y`.

Function `divpwr2` divides its first argument by 2^n, where n is the second argument. You may

Name	Description	Rating	Max Ops
`tmax(void)`	largest two's complement integer	1	4
`isNonNegative(x)`	`x >= 0?`	3	6
`isLessOrEqual(x,y)`	`x <= y?`	3	24
`divpwr2(x,n)`	`x/(1<<n)`	2	15
`abs(x)`	absolute value	4	10
`addOK(x,y)`	Does `x+y` overflow?	3	20

Table 2: Arithmetic Functions

assume that $0 \leq n \leq 30$. It must round toward zero.

Function `abs` is equivalent to the expression `x<0?-x:x`, giving the absolute value of `x` for all values other than *TMin*.

Function `addOK` determines whether its two arguments can be added together without overflow.

Note

If you want your solution auto-graded and ranked on the "Beat the Prof" website, then please follow the instructions in the *Hand In Instructions* section and also see the *Beat the Prof* section. Your code will be compiled with GCC.

The 15 puzzles you must solve have been given a difficulty rating between 1 and 4, such that their weighted sum totals to 40.

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 more clever. 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.

Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they do not need to 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 with the `dlc` and `btest` programs. If it does not compile, we can not grade it. All required tools to compile the programs should be installed on the lab machines. If you want to build it on your own machine, make sure to install gcc and gcc-multilib (e.g. on Ubuntu: `sudo apt-get install build-essential gcc-multilib`)

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 isPositive`.

Hand In Instructions

- Make sure you have included your identifying information in your file `bits.c`.
- Remove any extraneous print statements.
- Name your bits.c file as *your_nethz_username-bits.c*
- You are going to use SVN (`http://subversion.apache.org`) to handin your assignments for this course. Make sure you have it installed on your computer. On Ubuntu, you can install it by typing 'sudo apt-get install subversion'.
- We have set up a directory for each student in the course and the directories are named according to NETHZ User IDs. Clone your SVN folder to your computer using the following command (Replace the NETHZ_USERNAME with your own NETHZ User ID and enter the whole command in a single line).

  ```
  svn co --username NETHZ_USERNAME
  https://svn.inf.ethz.ch/svn/systems/casphs12_students/trunk/NETHZ_USERNAME
  ```

- Create a new directory called `assignment1` inside your directory.
- Copy your solution file into the newly created assignment folder and then type:

  ```
  svn add assignment1
  ```

  This will add your assignment folder and its contents into to your working copy and schedule them for addition to the SVN repository. They will be uploaded and added to the repository on your next commit when you type:

  ```
  svn commit -m "checking in assignment 1"
  ```

  Note that the string you provide after the -m switch does not have to be the same as the one provided here. It is intended to be a message for keeping a history on how the document evolves.

- After the handin, if you discover a mistake and want to submit a revised copy, place your new file into the assignment folder and type:

  ```
  svn commit -m "checking in revised assignment 1"
  ```

- For more information about how to use SVN, start with typing `svn --help`

Beat the Prof

You can check-in solutions to your SVN folder as often as you like. Solutions are going to be graded periodically and rankings are going to be listed on the course page using the 'nickname' fields you enter into the student structure of bits.c. In this assignment, you are going to compete against each other and the professor. The goal is to solve each data puzzle using the fewest number of operators. Entries are sorted by score, defined as (total instructor operations - total student operations). Students who match or beat the professor's operator count for each puzzle will be winners. You can follow the rankings via the link that will be provided in the course Web page's 'Assignments' section shortly after the release of the assignment.