## Goal

Get familiar with bit-level representations of integers. You’ll do this by solving a series of programming “puzzles.”

### Implementation Details

1. Change the current working directory to HW5. You will find a number of files in this directory. The only file you will be modifying is bits.c.
2. $ cd HW5
3. $ ls
4. > bits.c btest.c CodingRules.txt dlc Driverlib.pm fshow.c Makefile tests.c
5. > bits.h btest.h decl.c Driverhdrs.pm driver.pl ishow.c README
6. Complete each function skeleton by following the rules below. Refer CodingRules.txt for more details on rules.
   * Use **only** straightline code for the integer puzzles (i.e., no loops or conditionals; no for, while, if or else)
   * You are **only** allowed to use the following eight operators:
   * ! ̃ & ˆ | + << >>
   * No constants longer than 8 bits.
7. **You will get full credit for a puzzle if it passes all of the tests performed by btest, and no credit otherwise**

### The Puzzles

See the comments in bits.c for more details on the desired behavior of the functions.

You may also refer to the test functions in tests.c. These are used as reference functions to express the correct behavior of your functions, although they don’t satisfy the coding rules for your functions.

| **Name** | **Description** | **Points** |
| --- | --- | --- |
| bitAnd(x,y) | x & y using only | and ̃ | 5 |
| getByte(x,n) | Get byte n from x | 10 |
| byteSwap(x,n,m) | Swap the nth and mth byte from x | 10 |
| logicalShift(x,n) | Shift right logical | 15 |
| bitCount(x) | Count the number of 1’s in x | 20 |
| bitParity(x) | Return 1 if x contains an odd number of 0’s | 20 |
| bang(x) | Compute !x without using ! operator | 20 |

### Evaluation

**You will get full credit for a puzzle if it passes all of the tests performed by btest, and no credit otherwise.**

### Important

**Note:**

* Don’t include the <stdio.h> header file in your bits.c file, as it confuses dlc and results in some non-intuitive error messages. You will still be able to use printf in your bits.c file for debugging without including the <stdio.h> header, although gcc will print a warning that you can ignore.
* The dlc program enforces a stricter form of C declarations than is the case for C++ or that is enforced by gcc. In particular, any declaration must appear in a block (what you enclose in curly braces) before any statement that is not a declaration. For example, it will complain about the following code:

int foo(int x) {

int a = x;

a \*= 3; /\* Statement that is not a declaration\*/

int b = a; /\* ERROR: Declaration not allowed here \*/

}

### Autograding your work

We have included some autograding tools in the handout directory —btest,dlc, and driver.pl— to help you check the correctness of your work.

After every single change you make to your bits.c file follow the steps below.

$ make clean

$ make

$ ./driver.pl

#### Tools

1. **btest:** This program checks the functional correctness of the functions in bits.c. To build and use it, type the following two commands:

$ make

$ ./btest

You must rebuild btest each time you modify your bits.c file.  
You can use the -f flag to instruct btest to test only a single function:

$ ./btest -f logicalShift

If your implementation fails, this will show you the input arguments for which it failed. For ex:

$ ./btest -f logicalShift

> Score Rating Errors Function

> ERROR: Test logicalShift(-2147483648[0x80000000],1[0x1]) failed...

> ...Gives -1073741824[0xc0000000]. Should be 1073741824[0x40000000]

> Total points: 0/15

You can feed it specific function arguments using the option flags -1, -2, and -3:

$ ./btest -f bitAnd -1 7 -2 0xf

Check the file README for documentation on running the btest program.

1. **dlc:** This is a modified version of an ANSI C compiler from the MIT CILK group that you can use to check for compliance with the coding rules for each puzzle. The typical usage is:

$ ./dlc bits.c

1. **driver.pl:** This is a driver program that uses btest and dlc to compute the correctness and performance points for your solution. It takes no arguments:

$ ./driver.pl