# CS5541 - Computer Systems
# Assignment Name: Bitslab
# Spring 2018
# Due: Wednesday, February 07, 2018

Jason Eric Johnson

01/22/2018

## 1 Introduction

The purpose of this assignment is to become more familiar with bit-level representations of integers and floating point numbers. 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.

## 2 Handout Instructions

> Students can download the assignment files from the Elearning Dropbox for this assignment. The puzzle skeleton file is called `bitslab.zip`. This file contains the files `bitslab.c`, `bitslab.h`, and an example of a test file with a main() function called `test.c`. Note that if you compile and run these files as they are the `bitAnd(15,3)` call will return 2. This is because that function hasn't been implemented yet (that's your task) and all function skeletons return 2 right now.

The `bitslab.c` file contains a skeleton for each of the 14 programming puzzles. Your assignment is to complete each function skeleton using only *straightline* code for the integer puzzles (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:

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

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 `bitslab.c` for detailed rules and a discussion of the desired coding style.

# 3 The Puzzles

This section describes the puzzles that you will be solving in `bitslab.c`.

## 3.1 Bit Manipulations

Table 1 describes a set of functions that manipulate and test sets of bits. The "Points" field gives 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. See the comments in `bitslab.c` for more details on the desired behavior of the functions.

| Name | Description | Points | Max Ops |
|---|---|---|---|
| `bitAnd(x,y)` | `x & y` using only `\|` and `~` | 2 | 8 |
| `getByte(x,n)` | Get byte `n` from `x`. | 2 | 6 |
| `logicalShift(x,n)` | Shift right logical. | 3 | 20 |
| `bitCount(x)` | Count the number of 1's in `x`. | 4 | 40 |
| `bang(x)` | Compute `!x` without using `!` operator. | 4 | 12 |

Table 1: Bit-Level Manipulation Functions.

## 3.2 Two's Complement Arithmetic

Table 2 describes a set of functions that make use of the two's complement representation of integers. Again, refer to the comments in `bitslab.c` for more information.

| Name | Description | Points | Max Ops |
|---|---|---|---|
| `tmin()` | Most negative two's complement integer | 1 | 4 |
| `fitsBits(x,n)` | Does `x` fit in `n` bits? | 2 | 15 |
| `divpwr2(x,n)` | Compute $x/2^n$ | 2 | 15 |
| `negate(x)` | $-x$ without negation | 2 | 5 |
| `isPositive(x)` | `x > 0`? | 3 | 8 |
| `isLessOrEqual(x,y)` | `x <= y`? | 3 | 24 |

Table 2: Arithmetic Functions

### 3.3 Floating-Point Operations

For this part of the assignment, you will implement some common single-precision floating-point operations. In this section, you are allowed to use standard control structures (conditionals, loops), and you may use both `int` and `unsigned` data types, including arbitrary unsigned and integer constants. You may not use any unions, structs, or arrays. Most significantly, you may not use any floating point data types, operations, or constants. Instead, any floating-point operand will be passed to the function as having type `unsigned`, and any returned floating-point value will be of type `unsigned`. Your code should perform the bit manipulations that implement the specified floating point operations.

Table 3 describes a set of functions that operate on the bit-level representations of floating-point numbers. Refer to the comments in `bits.c` for more information.

| Name | Description | Points | Max Ops |
|---|---|---|---|
| `float_neg(uf)` | Compute `-f` | 4 | 10 |
| `float_i2f(x)` | Compute `(float) x` | 4 | 30 |
| `float_twice(uf)` | Computer `2*f` | 4 | 30 |

Table 3: Floating-Point Functions

## 4 Evaluation

Your score will be computed out of a maximum of 100 points based on the following distribution:

**40** Correctness points.

**28** Performance points.

**32** Assignment report points.

*Correctness points.* The 14 puzzles you must solve have been given a difficulty rating between 1 and 4, such that their weighted sum totals to 40. You will get full credit for a puzzle if it performs correctly and no credit otherwise.

*Performance points.* The main concern at this point in the course is that you can get the right answer. However, I 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 I want you to be more clever. Thus, for each function I'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 correct function that satisfies the operator limit.

*Assignment report points.* A PDF file explaining (briefly) how you solved each puzzle. This description can be as simple as "I added the two inputs." or a paragraph or two going into the details of the more complex puzzles. There is no minimum or maximum length for the report. Just write what you need to clearly explain your solution to each puzzle. That means to be specific in your description. "I used the code to solve the

problem." will not get you points. You don't have to write a huge amount, but make sure you explain HOW the code you wrote solved the problem. You will also meet with me to explain in detail any puzzle or puzzles I ask you about. Be prepared to speak in-depth about your process and reasoning behind how you arrived at your solutions.

# 5 Handin Instructions

Submit Bitslab using Elearning. Follow all the instructions on the "Homework Guidelines" page carefully.

You will submit your assignment as a single `.zip` file, named as specified in the "Homework Guidelines". It should include your report `PDF` file, which I would like you to name:

```
Bitslab-Report-[YourName].pdf
```

Also include your `bitslab.c` file and a `README.txt` file, which will contain authorship information and any references. These are the only three files you need to include in the `.zip` file.

So, since my name is Jason Johnson, my submission would be a single `.zip` archive called:

```
CS5541-JasonJohnson-Bitslab.zip
```

It would contain the following files:

```
Bitslab-Report-JasonJohnson.pdf
bitslab.c
README.txt
```