CS270 Programming Assignment 2
“Arithmetic/Logic Operations in C”

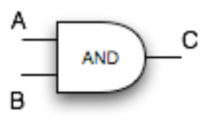
Programming Assignment is due Monday, February 27 (via checkin by 5:00pm)
Programming assignments are to be done individually.

Goals

For this programming assignment you will write a C program to simulate bit-wise operations, an adder and a floating point adder. You will

- Learn how to simulate arithmetic and logic operations with C functions.
- Increase your understanding of circuits which are fundamental to computer hardware.
- Understand union types and casting in C

For example, you will be simulating a “fat” AND gate as shown below:



The Assignment

Make a subdirectory called PA2 for the programming assignment, all files should reside in this subdirectory. Copy conversion.c, conversion.h, logic.h, logic.c, main.c, and the Makefile to PA2. For this assignment we are using a 16 bit unsigned integer type (`uint32_t`) to represent an LC-3 word.

The `conversion.c` file already contain declarations, as well as implementations of the following functions:

```
LC3_Word floatToLC3_Word(float i);
LC3_Word static floatToLC3_WordI(uint32_t x);
float LC3_WordToFloat(LC3_Word y);
```

You need to implement the following function:

```
uint32_t static LC3_WordToFloatI(LC3_Word y);
```

Instructions for implementing this function will be discussed in class. There are (at least) two ways to implement it, either implementation is acceptable. For the data types and union types please use C library for help.

The file `logic.c` contains declarations of the following functions:

```
LC3_Word NOT(LC3_Word A);
LC3_Word AND(LC3_Word A, LC3_Word B);
LC3_Word OR(LC3_Word A, LC3_Word B);
LC3_Word XOR(LC3_Word A, LC3_Word B);
LC3_Word ADD(LC3_Word A, LC3_Word B);
LC3_Word FLADD(LC3_Word A, LC3_Word B);
void floatToHEX(float A);
```

The implementation for the function `void floatToHEX(float A)` is given. You have to implement all the other functions. Use this function to debug your code.

Special instructions for bitwise addition:
As we are representing a 16 bit LC3 word, you do not have to implement the overflow (17th bit) but assume as the LC-3 hardware does, that the answer is always legal.

Special instructions for Floating point addition:

You need to implement this function for only the cases when both arguments have the same sign. However, the exponents may be different and you must correctly account for this by shifting the number with the smaller exponent before adding the fractional parts as discussed in class. You do not have to implement for the special cases when the exponent is 0 or 31 (all 1s). You may also assume that the answer will never be in this form.

The main program contains a single test for each of the following functions:

```
LC3_Word NOT(LC3_Word A);
LC3_Word AND(LC3_Word A, LC3_Word B);
LC3_Word OR(LC3_Word A, LC3_Word B);
LC3_Word XOR(LC3_Word A, LC3_Word B);
LC3_Word ADD(LC3_Word A, LC3_Word B);
LC3_Word FLADD(LC3_Word A, LC3_Word B);
```

To compile all files into a program called pa2, type the following command:

```
$ make
```

To run the compiled program, type the following command:

```
$ ./pa2
```

Part of this assignment is to make sure that your test your programs as thoroughly as possible.

Submission Instructions

All programming assignments will be submitted via “checkin”, same as the previous assignment. When you are done, your directory should have the conversion.c, conversion.h, logic.h, logic.c, main.c files. To package the files into a single file (pa2.tar) for submission, type the following command from your home directory:

```
$ make pack
```

Submit the file pa2.tar produced by the command shown above.

Reminders:

Make sure that your tar file is named `pa2.tar`, all lowercase.
Make sure that your tar file unpacks to a directory names PA2.
Comments headers are required for each file and function.
Function return values, names, and parameters must exactly match, including case.

Grading Criteria

To grade the assignment, we will verify that your logic functions work correctly (100 points). In addition points will be deducted for improper coding style and not following assignment directions (10 points). The grading factors we consider for coding style include having clear and concise comments, consistent indentation, and the minimal amount of code to solve the problem.

We will also award 10 points for thorough testing and submission of test cases.

Distribution of points for the assignment:

10 points each for the following functions:

```
LC3_Word NOT(LC3_Word A);
LC3_Word AND(LC3_Word A, LC3_Word B);
LC3_Word OR(LC3_Word A, LC3_Word B);
LC3_Word XOR(LC3_Word A, LC3_Word B);
LC3_Word ADD(LC3_Word A, LC3_Word B);
```

20 points each for the following functions:

```
LC3_Word FLADD(LC3_Word A, LC3_Word B);
uint32_t static LC3_WordToFloatI(LC3_Word y);
```

Late Policy

Late assignments will be accepted up to 48 hours past the due date and time for a deduction of 10% (24 hours late), 20% (24-48 hours late) and will not be accepted past this period. If you have submission problems, DO NOT EMAIL THE ASSIGNMENT TO US. You must use checkin to submit assignments.