# UMass Boston CS 240
# Homework 4
# Due 3/21/2019 17:00

Make a subdirectory hw4 in your home directory for this assignment. Copy all four files from `/home/ming/240/hw4` to your hw4. The assignment is to use bitwise operations to add two numbers.

## 1 Half Adder and Full Adder

In digital circuits, the half adder adds two input bits, P and Q, and produces two output bits, sum S and carry C. See the left part of Figure 1. The truth table of the half adder is as follows.

| input | | output | |
|---|---|---|---|
| P | Q | C | S |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Ideally, we should write one function that computes both S and C, but we have not discussed how to return multiple values from a function. Therefore, we write two functions that compute S and C separately, as follows.

```
enum bits {ZERO, ONE};

enum bits halfAdderSum(enum bits P, enum bits Q)
```

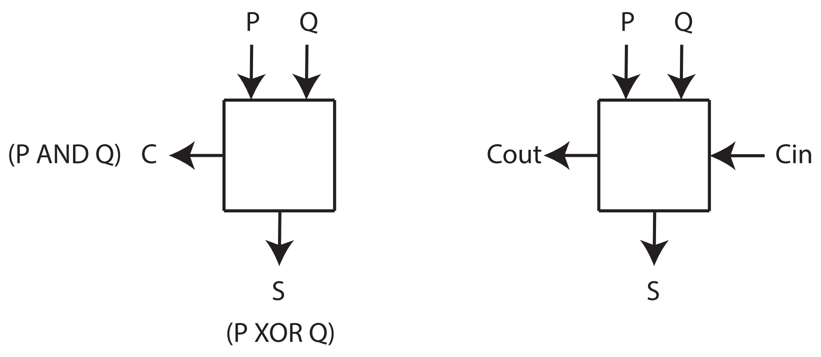

Figure 1: Left: the half adder. Right: the full adder

```
{
  return P ^ Q;
}

enum bits halfAdderCarry(enum bits P, enum bits Q)
{
  return P & Q;
}
```

The half adder adds only two bits. It becomes inadequate when we want to add more than two bits. Let us consider adding two 8-bit integers, $P_7P_6P_5P_4P_3P_2P_1P_0$ and $Q_7Q_6Q_5Q_4Q_3Q_2Q_1Q_0$. After we add $P_0$ and $Q_0$, the carry bit may be 1. We must incorporate it when adding $P_1$ and $Q_1$. Thus we need the full adder that takes three inputs: P, Q, and the carry-in Cin. The output bits are sum S and carry-out Cout. See the right part of Figure 1. The truth table of the full adder is as follows.

| input | | | output | |
|---|---|---|---|---|
| P | Q | Cin | Cout | S |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Your task in this part of the homework is to implement the two functions for the full adder in the file `adder.c`.

```
enum bits {ZERO, ONE};

enum bits fullAdderSum(enum bits P, enum bits Q, enum bits Cin)
{
}

enum bits fullAdderCarry(enum bits P, enum bits Q, enum bits Cin)
{
}
```

You should start by working out the Boolean expressions for S and Cout, and try to use as few Boolean operators as possible. The essence of C programming is brevity and efficiency.

## 2 Adding Two Numbers

Now you are ready to add two 32-bit integers bit by bit. Using a cascade of full adders, you can add a pair of bits and the carry bit from the previous position, save the sum bit, and send the carry bit to the next position. The first Cin should be set to zero. If we assume the sum of the two numbers does not overflow the 32-bit storage, we can safely discard the last Cout.

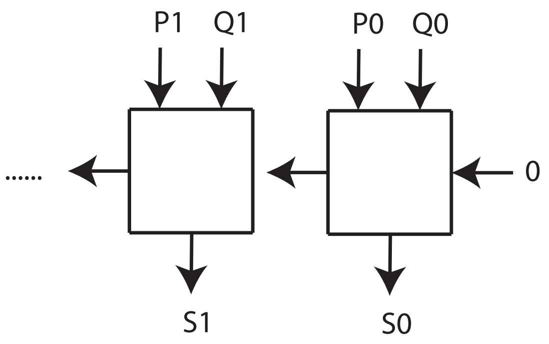

Figure 2: Using a cascade of full adders to add two numbers.

Your task in this part of the homework is to implement `myAdd()` in the file `myAdd.c`. The algorithm works as follows.

1. Initialize Cin to zero
2. For $i = 0, 1, \ldots, 31$
    (a) Extract the $i$-th bits from P and Q
    (b) Use `fullAdderSum()` and `fullAdderCarry()` to calculate the sum bit S and the carry bit Cout
    (c) Write the sum bit to the $i$-bit of `mySum`
    (d) Move Cout to Cin for the next iteration

There are many ways to extract a bit or write a bit in a 32-bit storage. See the lecture notes 7 and 8 for different methods that operate on bits.

## 3 The Driver

The driver is in the file `main.c`. You should not change anything in the main program, but you should write some comments at the top of the file.

Make sure your C code follows the style guidelines. In the `readMe.txt` file, you should write pseudo code and discuss what you found difficult about this assignment, how you planned your approach to it, and what you learned completing it.

## 4 Grading Rubric

1. (10 points)
    (a) Existence of the directory "/home/user??/hw4"
    (b) Inside hw4: adder.h, adder.c, main.c, myAdd.c, and readMe.txt
    (c) Other files are allowed
2. (20 points) `adder.c`

   (a) Adherence to the style guidelines
   (b) Sensible variable names, comments, etc.
   (c) `icc -Wall -c adder.c` gives no warnings or errors

3. (10 points) `main.c`
   (a) Comments at top of file
   (b) `icc -Wall -c main.c` gives no warnings or errors

4. (40 points) `myAdd.c`
   (a) Adherence to the style guidelines
   (b) Sensible variable names, comments, etc.
   (c) `icc -Wall adder.c main.c myAdd.c -o adder` gives no warnings or errors
   (d) `./adder` produces correct output

5. (20 points) `readMe.txt`
   (a) Pseudo code
   (b) Discussion of difficulty, approach, and lessons