P2 – Adding and Subtracting

3.1 Implementing Single Bit Adder

```
int main() {
    bool a, b, c, sum, carry;

    cout << "First Bit: ";
    a = readBit();

    cout << "Second Bit: ";
    b = readBit();

    cout << "Carry Bit: ";
    c = readBit();


    fullAdder(a, b, c, sum, carry);

    cout << "Sum: ";
    printBin(sum);

    cout << "Carry: ";
    printBin(carry);

    return 1;
}
```

```
void halfAdder(bool a, bool b, bool &s, bool &c) {
    s = XOR(a, b);
    c = AND(a, b);
}

void fullAdder(bool a, bool b, bool Cin, bool &s, bool &c) {
    bool c1, c2, s1;

    halfAdder(a, b, s1, c1);
    halfAdder(Cin, s1, s, c2);
    c = OR(c1, c2);
}
```

```
D:\2020+21\ELEC1204 - Advanced Programming\P2\src\OneBitAdder>main.exe
First Bit: 0
Second Bit: 1
Carry Bit: 1
Sum: 0
Carry: 1

D:\2020+21\ELEC1204 - Advanced Programming\P2\src\OneBitAdder>main.exe
First Bit: 1
Second Bit: 1
Carry Bit: 1
Sum: 1
Carry: 1

D:\2020+21\ELEC1204 - Advanced Programming\P2\src\OneBitAdder>main.exe
First Bit: 1
Second Bit: 0
Carry Bit: 0
Sum: 1
Carry: 0
```

3.2 Implementing an 8-Bit Adder

```cpp
int main() {
    bool a[N];
    bool b[N];
    bool c[N + 1] = {false, };

    bool sum[N];
    bool carry;



    cout << "Input number 1: ";
    readNum(a);
    cout << "Number 1: ";
    printByte(a);
    cout << endl;


    cout << "Input number 2: ";
    readNum(b);
    cout << "Number 2: ";
    printByte(b);
    cout << endl;

    cout << "Input Carry: ";
    c[0] = readBit();


    for(int i=0; i<N; i++) {
        fullAdder(a[i], b[i], c[i], sum[i], c[i+1]);
    }

    cout << "\n\nSum: ";
    printByte(sum);
    cout << endl;

    cout << "Carry: ";
    printBin(c[N]);
    cout << endl;

    return 1;
}
```

```
D:\2020+21\ELEC1204 - Advanced Programming\P2\src\nBitAdder>main.exe
Input number 1: 15
Number 1: 00001111
Input number 2: 1
Number 2: 00000001
Input Carry: 0
Sum: 00010000
Carry: 0

D:\2020+21\ELEC1204 - Advanced Programming\P2\src\nBitAdder>make mai
g++     main.cpp   -o main

D:\2020+21\ELEC1204 - Advanced Programming\P2\src\nBitAdder>main.exe
Input number 1: 15
Number 1: 00001111
Input number 2: 15
Number 2: 00001111
Input Carry: 0


Sum: 00011110
Carry: 0

D:\2020+21\ELEC1204 - Advanced Programming\P2\src\nBitAdder>main.exe
Input number 1: 255
Number 1: 11111111
Input number 2: 255
Number 2: 11111111
Input Carry: 0


Sum: 11111110
Carry: 1

D:\2020+21\ELEC1204 - Advanced Programming\P2\src\nBitAdder>main.exe
Input number 1: 128
Number 1: 10000000
Input number 2: 45
Number 2: 00101101
Input Carry: 0


Sum: 10101101
Carry: 0
```

3.3 Subtraction

```
//modified for 2s compliment
void readNum(bool *bin) {
    int num;

    cin >> num;

    if (num < 0) {
        bin[N-1] = true;

        num = num * -1;
        num--;

        for(int i=0; i<N-1; i++) {
            bin[i] = !(num % 2);
            num = num / 2;
        }
    }
    else {
        bin[N-1] = false;

        for(int i=0; i<N-1; i++) {
            bin[i] = num % 2;
            num = num / 2;
        }
    }

}
```

```
void convertTwosComplement(bool *bin) {
    bool cr[N+1];
    bool n1[N] = {1, 0, 0, 0, 0, 0, 0, 0};
    bool temp[N];

    //invert all bits
    for(int i=0; i<N; i++) {
        bin[i] = !bin[i];
    }

    //add 1 to complement
    for(int i=0; i<N; i++) {
        fullAdder(bin[i], n1[i], cr[i], temp[i], cr[i+1]);
        bin[i] = temp[i];
    }
}
```

```
cout << "Input number 1: ";
readNum(a);
cout << "Number 1: ";
printByte(a);
cout << endl;


cout << "Input number 2: ";
readNum(b);
cout << "Number 2: ";
printByte(b);
cout << endl;

cout << "Input Carry: ";
c[0] = readBit();

cout << "Input Sign: ";
if(readSymbol()) {
    convertTwosComplement(b);
}

cout << "Inverted B: ";
printByte(b);


for(int i=0; i<N; i++) {
    fullAdder(a[i], b[i], c[i], sum[i], c[i+1]);
}

cout << "\n\nSum: ";
printByte(sum);
cout << endl;

cout << "Carry: ";
printBin(c[N]);
cout << endl;
```

```
D:\2020+21\ELEC1204 - Advanced Prc
Input number 1: 5
Number 1: 00000101
Input number 2: -3
Number 2: 11111101
Input Carry: 0
Input Sign: -
Inverted B: 00000011

Sum: 00001000
Carry: 0
```

4 Additional Work

```
cout << "Input number 1: ";
readNum(a);
cout << "Number 1: ";
printByte(a);
cout << endl;


cout << "Input number 2: ";
readNum(b);
cout << "Number 2: ";
printByte(b);
cout << endl;


//initially load the accumulation
sum[0] = AND(b[0], a[0]);
for(int i=0; i<N-1; i++) {
    acc[i] = AND(a[i+1], b[0]);
}
acc[N-1] = false;


for(int i=1; i<N; i++) {


    for(int j=0; j<N; j++) {
        fullAdder(acc[j], AND(a[j], b[i]), false, acc[j], carry);
    }

    sum[i] = acc[0];
    acc[0] = acc[1];
    acc[1] = acc[2];
    acc[2] = acc[3];
    acc[3] = carry;
}

sum[4] = acc[0];
sum[5] = acc[1];
sum[6] = acc[2];
acc[7] = acc[3];

cout << "Multiplication: ";
for(int i=1; i<=8; i++) {
    printBin(sum[N - i]);
}
cout << endl;

cout << "Carry: ";
printBin(carry);
```

This code only works for certain values, I was unable to debug it in time before the end of the lab.

```
D:\2020+21\ELEC1204 - Advanced Programming\P2\src\adderCascade>main.exe
Input number 1: 5
Number 1: 0101
Input number 2: 5
Number 2: 0101
Multiplication: 00010111
Carry: 0
D:\2020+21\ELEC1204 - Advanced Programming\P2\src\adderCascade>main.exe
Input number 1: 3
Number 1: 0011
Input number 2: 4
Number 2: 0100
Multiplication: 11000111
Carry: 0
D:\2020+21\ELEC1204 - Advanced Programming\P2\src\adderCascade>
```

Full 3.3 Code

```
#include <iostream>
#include <math.h>

using namespace std;



#define N 8


bool AND(bool a, bool b);
bool OR(bool a, bool b);
bool XOR(bool a, bool b);

void printState(bool n);
void printBin(bool n);
void printByte(bool *n);


bool readBit(void);
void readNum(bool *bin);
bool readSymbol(void);

void halfAdder(bool a, bool b, bool &s, bool &c);
void fullAdder(bool a, bool b, bool Cin, bool &s, bool &c);


void convertTwosComplement(bool *bin);


int main() {
    bool a[N];
    bool b[N];
    bool c[N + 1] = {false, };

    bool sum[N];
    bool carry;



    cout << "Input number 1: ";
    readNum(a);
    cout << "Number 1: ";
    printByte(a);
    cout << endl;


    cout << "Input number 2: ";
```

```
    readNum(b);
    cout << "Number 2: ";
    printByte(b);
    cout << endl;

    cout << "Input Carry: ";
    c[0] = readBit();

    cout << "Input Sign: ";
    if(readSymbol()) {
        convertTwosComplement(b);
    }

    cout << "Inverted B: ";
    printByte(b);


    for(int i=0; i<N; i++) {
        fullAdder(a[i], b[i], c[i], sum[i], c[i+1]);
    }

    cout << "\n\nSum: ";
    printByte(sum);
    cout << endl;

    cout << "Carry: ";
    printBin(c[N]);
    cout << endl;

    return 1;
}


void printState(bool n) {
    cout << ((n) ? "True" : "False") << endl;
}

void printBin(bool n) {
    cout << ((n) ? "1" : "0");
}

void printByte(bool *n) {

    for(int i=1; i<=N; i++) {
        printBin(n[N - i]);
    }
}

bool readBit() {
```

```
    char buf;
    cin >> buf;

    if(buf == '1') {
        return true;
    }
    else {
        return false;
    }
}

//modified for 2s complement
void readNum(bool *bin) {
    int num;

    cin >> num;

    if (num < 0) {
        bin[N-1] = true;

        num = num * -1;
        num--;

        for(int i=0; i<N-1; i++) {
            bin[i] = !(num % 2);
            num = num / 2;
        }
    }
    else {
        bin[N-1] = false;

        for(int i=0; i<N-1; i++) {
            bin[i] = num % 2;
            num = num / 2;
        }
    }
}

bool readSymbol() {
    char sym;

    cin >> sym;

    if(sym == '-') {
        return true;
    }
    else {
        return false;
```

```
    }
}


void convertTwosComplement(bool *bin) {
    bool cr[N+1];
    bool n1[N] = {1, 0, 0, 0, 0, 0, 0, 0};
    bool temp[N];

    //invert all bits
    for(int i=0; i<N; i++) {
        bin[i] = !bin[i];
    }

    //add 1 to complement
    for(int i=0; i<N; i++) {
        fullAdder(bin[i], n1[i], cr[i], temp[i], cr[i+1]);
        bin[i] = temp[i];
    }
}



bool AND(bool a, bool b) {
    return a && b;
}

bool OR(bool a, bool b) {
    return a || b;
}

bool XOR(bool a, bool b) {
    return (a != b);
}


void halfAdder(bool a, bool b, bool &s, bool &c) {
    s = XOR(a, b);
    c = AND(a, b);
}

void fullAdder(bool a, bool b, bool Cin, bool &s, bool &c) {
    bool c1, c2, s1;

    halfAdder(a, b, s1, c1);
    halfAdder(Cin, s1, s, c2);
    c = OR(c1, c2);
}
```