CSCI 463 Assignment 1 – Binary Integers

20 Points – Due Tuesday, September 24, 2019 at 23:59

Abstract

In this assignment, you will write a program that can add and subtract two's compliment and unsigned binary numbers up to 512 bits long.

1 Input

Your program should read an arbitrary number of pairs of input lines. Each input line is a binary number represented as a string of '1's and '0's whose length may be from 2 to 512 digits. You may assume that the program's input contains only binary numbers supplied one-per-line and that each *pair* of numbers is the same length (number of bits). For example:

Example `testdata` Input File

```
0111
0001
10110100101011101100110111010101111011101110101110011111011111111
10001000001010110001001101110101010111011100101110010011011
00100010000010101100010011011101010101101110110010111001001011101
01000100000101011000100110111010101011011101100101110010010110101
10000
10000
00
11
```

2 Output

For each pair of input lines, your program must print the input numbers, their sum and their difference. Each output line will contain four fields:

1. Identifier

   The identifier starts in column one and indicates the value printed on the line. It will be one of: `v1` for input value one (v_1), `v2` for input value two (v_2), `sum` for $v_1 + v_2$, or `diff` for $v_1 - v_2$.

2. Parity

   The parity is printed as the word `even` or `odd`, as appropriate, starting in column six for the binary value printed in field four.

3. Flags

   The flags field starting in column 11 is only present on the `sum` and `diff` lines and contains an `S` when signed overflow occurs, a `U` when unsigned overflow occurs, and a `Z` if the value in field four is zero. Note that zero or more of these flags may be present at the same time.

/home2/winans/projects/niu/NIU/courses/463/2019-fa/assignments/a1/handout.tex

jwinans@niu.edu 2019-09-18 08:10:59 -0500 v2.0-118-g72b8caa

4. Binary Value

   The binary value for the value specified in field one. Each of the four binary values printed must be the same length as v_1 and v_2. (In other words, you will truncate any carries.)

Your output *must* exactly match the example output discussed here as it will be graded using the `diff(1)` command.

For example, the above input data would produce the following output:

Example Output

```
1  v1   odd       0111
2  v2   odd       0001
3  sum  odd  S    1000
4  diff even      0110
5
6  v1   odd       10110100101011101100110111010101111011101110101110011101111111
7  v2   even      10001000001010110001001101110101010110111011000101110010010011011
8  sum  even SU   00111100110110011110000101001011010010101001111010000011101101O
9  diff odd       00101100100000111011101001100000100100110011100010111010010010O
10
11 v1   odd       00100010000010101100010011011101010101101110110010111001001011O1
12 v2   odd       01000100000101011000100110111010101011011101100101110010010110O
13 sum  odd       01100110001000000100111010011000000001001100011000101011100001111
14 diff even  U   11011101111101010011101100100010101010010001001101000110110100111
15
16 v1   odd       10000
17 v2   odd       10000
18 sum  even SUZ  00000
19 diff even   Z  00000
20
21 v1   even      00
22 v2   even      11
23 sum  even      11
24 diff odd   U   01
```

3 File You Must Write

You will write a C++ program suitable for execution on `hopper.cs.niu.edu` (or `turing.cs.niu.edu`.)

Create a directory named `prog1` and place your source files for this assignment within. Implement your solution for this assignment in a single file named `prog1.cc`. If you prefer, you may also create and use a corresponding `prog1.h` file.

Use `std::cin` and `std::cout` to read and print your data. You may assume that the test data is in the correct format.

When we grade your assignment, we will compile and run it (with our own custom testdata file) on hopper using these commands:

```
g++ -ansi -pedantic -Wall -Werror -Wextra -std=c++11 prog1.cc -o prog1
./prog1 < testdata
```

4 How To Hand In Your Program

When you are ready to turn in your assignment, make sure that the only files in your `prog1` directory is/are the `prog1.cc` (and perhaps `prog1.h`) file(s) discussed above. Then, in the parent of your `prog1` directory, use the `mailprog.463` command to send the contents of the files in your `prog1` project directory to your TA like this:

```
mailprog.463 prog1
```

If `mailprog.463` detects and problems, it will inform you that you have not followed the instructions given above and provide some hints how to proceed. If you followed these instructions you will see the following:

```
winans@hopper:~/$ mailprog.463 prog1

**********************************************************************
* WARNING : Do NOT use this program to mail notes to your Instructor *
*           Doing so may result in the loss of your program !!       *
**********************************************************************

Enter program number for your assignment : 1
shar: Saving /tmp/mailprog.11111 (text)
winans@hopper:~/$
```

5 Grading

The grade you receive on this programming assignment will scored according to the syllabus and its ability to compile and execute on the CSCI Department's computer.

It is your responsibility to test your program thoroughly. We will run it against a variety of test numbers of different sizes.

6 Hints

1. `sum`

   Write a full-adder subroutine that takes three one-bit input arguments (the two value bits and a carry-in) and that returns the result as two one-bit output arguments (the sum and the carry-out.) Then use your full-adder function to add the bits together one column at a time.

   Start at the LSB with a carry-in of zero. Add the column and set the result value for that column using your full-adder routine. Repeat for the rest of the columns by propagating the carry-out from each column to the next one.

   Recall that the signed overflow status is set when the carry-out of the MSB is different from the carry-in of the MSB and the unsigned overflow status is set when there is a carry-out of the MSB. (See section 2.2.5 of RVALP in Blackboard for a discussion.)

2. `diff`

   Recall that

   difference = minuend - subtrahend

   is the same as

   difference = minuend + (-subtrahend)

   Therefore, we can perform subtraction by adding the two's compliment of the subtrahend to the minuend. Write function to calculate and return the 1's compliment of a given value and then add it with a carry-in argument set to 1 when calling the full-adder for the LSB column.

   Note that this method of subtraction will for both *signed* and *unsigned* binary numbers... *except* that the unsigned overflow status will be set when there is ***not*** a carry-out of the MSB. (See section 2.2.5 of RVALP in Blackboard for a discussion.)