COE 0147 Spring 2013
Lab #6: 1-bit Adders and Number Representation

This lab contains both a programming component and a written component. Each of you should submit your own solution via email according to the instructions at your TA's website. Each person must turn in their own copies of the lab. If you choose to work with a neighbor/partner, put your partner's name on your submitted copy of the lab. Submission timestamps will be checked. Late submissions will not be accepted. Follow instructions so that your program produces output in the correct format. You will have to zip your files into a .zip archive and submit it via email as directed on the lab website. For the written part, you should turn in a hard copy of this assignment either in recitation class or in corresponding TA's mailbox by the due date. Staple multiple pages together and do not forget to put your name on. Please do not present answers out of order.

Programming Part

Adding Two Binary Numbers
Look at Section 3.2 and Figure 3.1 on page 225 of your textbook. This section explains how to use a 1-bit adder to add two 32-bit binary numbers. In this part of your assignment, you will be adding two 16-bit binary numbers using a 1-bit adder. 1-bit adders have also been explained in detail in class.

The two 16-bit binary numbers will be given to you in a string form. For illustration, consider the following data segment.

```
.data
A:    .asciiz        "0111000011000100"
B:    .asciiz        "1100010001110000"
C:    .space         17
```

A and B are two strings, each of which contains an unsigned 16-bit binary number in a 16-character long string form. You have to fill up C with the binary representation of the result of summing up A and B. Do not store the overflow bit in C. You will also set $v0 register to 1 if overflow occurs or 0 otherwise.

A word of caution: To earn full credit on this lab, your implementation must not convert the string representation of the numbers into a register content, add the registers and convert the result value back into binary form. You must do the addition bit-by-bit using 1-bit adder.

Please use the template code from:

http://www.pitt.edu/~dbd12/teaching/files/lab6part1.asm

which asks the user to input A and B and prints out C and the overflow value in $v0. User input will always be a 16-bit binary number in a 16-character long string form (as shown in the example on the previous page).

Regardless of whether you use the sample code, your submission should be named lab6part1.asm

Hint: Hexadecimal values for '0' and '1' are 0x30 and 0x31 respectively. If you subtract 0x30 from the character representation, you will get the decimal value. Similarly, if you add 0x30 to decimal 0 or 1, you will get the character representation of them.

Written Part

1. Add the following unsigned binary numbers (show the carry and overflow bits)

```
  0010 0110 1001
+ 1111 1100 0101
 --------------
```

2. Subtract the following unsigned binary numbers (show the borrow and underflow bits). Do not convert to two's-complement.

```
  0011 0111 0101
- 1110 1000 1110
 --------------
```

3. Convert the following decimal numbers to binary numbers (represent each as a 16-bit number):
   1639, 48265, 1010

4. Convert the following unsigned binary numbers to decimal numbers:
   Number 1: 10000001 01011110
   Number 2: 00000110 01010011

5. Convert the following decimal numbers into 9-bit binary numbers (with sign-magnitude):
   48, -126, -34

6. Convert the following 9-bit binary numbers (with sign-magnitude) to decimal numbers:
   010011110, 100110111, 110101010

7. Convert the following decimal numbers into 9-bit binary numbers in 1's complement form:
   56, -145, -52

8. Convert the following 8-bit binary numbers in 1's complement to decimal numbers:
   01010011, 11010010, 11110111

9. Convert the following decimal numbers into 9-bit binary numbers in 2's complement form:
   196, -17, -95

10. Convert the following 8-bit binary numbers in 2's complement to decimal numbers:
    01010101, 10111101, 11010000