# COE 0147 Lab 6

**Adders and Number Representation**

# Adding or Subtracting two binary numbers

In this part of your assignment, you will be adding (or subtracting) two 16-bit binary numbers using a 1-bit adder. The two 16-bit binary numbers will be given to you in form of a string. 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 form of a 16-character long string. You have to add (or subtract) A and B, and store the binary representation of the result in C. Do not store the overflow bit in C, instead store it in $v0 for printing.

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 on courseweb and fill out the TODOs with your own code. For procedures, the template already defines necessary inputs and outputs.

This template code prompts the user to enter A and B, and then an operation for A and B, either addition (enter 0) or subtraction (enter 1). Finally, the code prints out C along with the overflow value. User input is a 16-bit binary number in form of a 16-character long string.

**Important:**

1. **If you use the stack, you must make sure it’s cleaned up correctly! In other words, the $sp value before you begin and after you end should be the same! There is code to help check this.**
2. **We are enforcing the MIPS conventions for $s registers. That means, you are NOT allowed to trash them. So, if you use them, you must restore them. Don’t be lazy when restoring registers. We test for if you trash the registers by** 
   1. **Writing some random values in the $s registers before we call your functions.**
   2. **Seeing if they are still there when your function returns.**

**We included test code. We will change the test values when checking!**

**Hint:**

(1)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 corresponding character representation of them.

(2) You could consider the binary A and B are in 2’s complement form. If we perform addition, you are able to easily get the result with 1-bit adders. If we want subtraction, the easiest way is to convert B into its 2’s complement and perform addition in the same way as the adding operation. **Subtracting B means adding –B.**

(3) In 2’s complement, the overflow bit does not necessarily mean the overflow happens, so we only want to print the overflow bit. **If you are interested**, you can also try to identify the real overflow with your code.

## Submit lab6part1\_username.asm on courseweb.

1. **Number Representation**

For the following questions, fill out the test on courseweb: Lab 6 Written Part.

**You should probably do this by hand first; you only get 1 submission attempt for this part!**

**For the questions below, if it says it is n-bit, when you put in the answer, make sure you put all n-bits! Even if there are leading zeros!**

* 1. Add the following unsigned binary numbers (show the overflow bit)

1010 0110 1001

+ 1111 1100 0101

----------------

* 1. Convert the following decimal numbers to binary numbers (represent each as a 16-bit number):

1624, 48265, 1010

* 1. Convert the following unsigned binary numbers to decimal numbers:

Number 1: 10000001 01011110

Number 2: 00000110 01010011

* 1. Convert the following decimal numbers into 9-bit binary numbers (with sign-magnitude):

48, 126, 34

* 1. Convert the following 9-bit binary numbers (with sign-magnitude) to decimal numbers:

010011110, 100110111, 110101010

* 1. Convert the following decimal numbers into 9-bit binary numbers in 1’s complement form:

56, 145, 52

* 1. Convert the following 8-bit binary numbers in 1’s complement to decimal numbers:

01010011, 11010010, 11110111

* 1. Convert the following decimal numbers into 9-bit binary numbers in 2’s complement form:

196, 12, 95

* 1. Convert the following 8-bit binary numbers in 2’s complement to decimal numbers:

01010101, 10111101, 11010000