**CIS 21JA Assignment 5**

**Overview**  
Write a program that adds 2 unsigned, 16-bit integers and prints the sum as a numeric text string.  
  
**Details**  
The program does the following 4 steps:

1. Read the 2 input

1. Prompt the user for one unsigned, 16 bit integer at a time.
2. Check that each input number is within the range of a *un*signed 16-bit integer.

Otherwise print an error message and keep re-prompting until you get a valid number. The error message should tell the user the range of valid values so they know what to enter. See sample output.

1. If the user enters 1 valid number and 1 invalid number, only print the error message and re-prompt for the invalid number.

Starting at this point, your code *must use 16-bit registers only*, unless you're calling Irvine IO routines.

2 pts of the lab is for using 16 bit registers for all calculation.

2. Add the 2 numbers

1. Add the 2 numbers and then check whether the sum is valid. Remember that you're only allowed to use 16-bit registers, and just looking at a 16-bit value won't tell you if the number is valid or not. How would you check for invalid data?
2. If the sum is invalid, print an error message and go to step 4.

3. If the result is valid:

1. Convert the numeric result into a string of characters. You need to define a text string in the .data section, and then store each digit of sum as a text character in the text string. What size text string should you define?
2. Call writeString and print the text string to screen. When printing the result, print a text explanation first, such as "Sum = ", then print the resulting string. See sample output.

This step takes more effort than just an add instruction of step 2. You will need to have your wits about you as you figure out:

* how to turn the number 217 in AX into an array of '2', '1', '7' characters to print. The ASCII table might come in handy.
* how to access the right memory location within the array of characters in order to store the characters and print them out.

Reminder and hint: Accessing an array in assembly is very similar to HLL.

Assuming you have an array Arr with 4 elements: 10, 20, 30, 40

In C++: using the array name using pointer notation

if i is 1: Arr[i] is 20 if ptr = Arr then [ptr + i] is 20

In assembly: using the array name: using pointer notation

if ebx is 1 Arr[ebx] is 20 if edx = OFFSET Arr then [edx + ebx] is 20

(Assuming Arr is an array of bytes)

4. After printing either the resulting string or the "invalid result" error message:

1. Ask the user whether to continue again
2. Accept 'y' or ' Y' as the answer to loop back to step 1.
3. If the user enters anything else, end the program.

**Additional requirements**

* Document your program to get full credit. Don't forget your name and lab description. Explain your loop and if else implementation.
* Don't use any memory variables during calculation, except for strings that are used for IO.   
  Store all numeric data in registers.
* Use 16-bit registers only, unless you're calling an Irvine IO procedure.
* You must use writeString to print the result. Using writeInt means an automatic 5 point deduction.
* Don't use bit-wise instructions (shift, and, or...) for this lab.
* Don't use the decision directives of MASM. Implement loops and if statements with assembly instructions.
* Keep your logic flow as simple as you can. Use "fall through" logic as shown in class notes or the book.

**Testing**

Test your result adequately: with valid and invalid input, with valid and invalid sum (result of step 2).

Sample program output (the upper range of valid data has been replaced with --- so you can fill it in your code)  
  
Enter first 16 bit unsigned integer: 29

Enter second 16 bit unsigned integer: -3

The number must be between 0 and ---

Enter second 16 bit unsigned integer: 39

Sum = 68

Continue? y/n: y

Enter first 16 bit unsigned integer: 60000

Enter second 16 bit unsigned integer: 60000

The sum is larger than 16 bits.

Continue? y/n: y

Enter first 16 bit unsigned integer: 0

Enter second 16 bit unsigned integer: 0

Sum = 0

Continue? y/n: n

Press any key to continue . . .

**Final note:** Why am I making life difficult for you by requiring that you use 16 bit registers?

Answer: In the "real world" sometime your calculation result can exceed your data storage size. If you're working on a 32-bit machine, your calculation result can exceed 32 bits. If you work on a 64-bit machine, your calculation result can exceed 64 bits. It doesn't matter what size data you have. How would you detect this problem when it occurs? You can use the same mechanism that you use in this assignment.

The reason I limit the data size to 16 bits is to make it easier to test your code. 16-bit data is shorter to type in than 32 bit data.