CS 271 Computer Architecture and Assembly Language

Programming Assignment #6 Option A (Choose Option A **or** Option B)

Due Sunday, Dec. 2 (11:59 PM)

Submit at http://engr.oregonstate.edu/teach before midnight. Submit a backup copy to Blackboard.

Objectives:

1) Designing, implementing, and calling low-level I/O procedures
2) Implementing and using a macro

Problem Definition:

- Implement and test your own *ReadVal* and *WriteVal* procedures for unsigned integers.
- Implement macros *getString* and *displayString*. The macros may use Irvine's *ReadString* to get input from the user, and *WriteString* to display output.
  - *getString* should display a prompt, then get the user's keyboard input into a memory location
  - *displayString* should the string stored in a specified memory location.
  - *readVal* should invoke the *getString* macro to get the user's string of digits. It should then convert the digit string to numeric, while validating the user's input.
  - *writeVal* should convert a numeric value to a string of digits, and invoke the *displayString* macro to produce the output.
- Write a small test program that gets 10 valid integers from the user and stores the numeric values in an array. The program then displays the integers, their sum, and their average.

Requirements:

1) User's numeric input must be validated the hard way: Read the user's input as a string, and convert the string to numeric form. If the user enters non-digits or the number is too large for 32-bit registers, an error message should be displayed and the number should be discarded.
2) Conversion routines must appropriately use the **`lodsb`** and/or **`stosb`** operators.
3) All procedure parameters must be passed on the system stack.
4) Addresses of prompts, identifying strings, and other memory locations should be passed by address to the macros.
5) Used registers must be saved and restored by the called procedures and macros.
6) The stack must be "cleaned up" by the called procedure.
7) The usual requirements regarding documentation, readability, user-friendliness, etc., apply.
8) Submit your text code file (*.asm*) by the due date. The submission site can be found at http:// engr.oregonstate.edu/teach/ (also submit a backup copy to Blackboard**)**.

Extra Credit:

1) 1 point: number each line of user input and display a running subtotal of the user's numbers.
2) 2 points: Handle signed integers.
3) 3 points: make your *ReadVal* and *WriteVal* procedures recursive.
4) 4 points: implement procedures *ReadVal* and *WriteVal* for floating point values, using the FPU.
5) 5 points: handle the input and output with interrupts instead of *ReadString* and *WriteString*.