CS 271 Computer Architecture and Assembly Language
Programming Assignment #3

Objectives: more practice with

1. Implementing data validation
2. Implementing an accumulator
3. Integer arithmetic
4. Defining variables (integer and string)
5. Using library procedures for I/O
6. Implementing control structures (decision, loop, procedure)

Description:

Write and test a MASM program to perform the following tasks:

1) Display the program title and programmer's name.
2) Get the user's name, and greet the user.
3) Display instructions for the user.
4) Repeatedly prompt the user to enter a number.
   a) Validate the user input to be in [-88, -55] or [-40, -1] (inclusive).
   b) Count and accumulate the valid user numbers until a non-negative number is entered. Detect this using the SIGN flag.
   (The non-negative number and any numbers not in the specified ranges are discarded.)
   c) Notify the user of any invalid numbers (negative, but not in the ranges specified)
5) Calculate the (rounded integer) average of the valid numbers.
6) Display:
   a) the number of validated numbers entered (Note: if no valid numbers were entered, display a special message and skip to *f.*)
   b) the sum of negative numbers entered
   c) the maximum (closest to 0) valid user value entered
   d) the minimum (farthest from 0) valid user value entered
   e) the average, rounded to the nearest integer (e.g. -20.5 rounds to -20, -20.51 rounds to -21)
   f) a parting message (with the user's name)

Requirements:

1. The *main* procedure must be modularized into commented logical sections (procedures are not required this time)
2. The four value limits must be defined as constants.
3. The user input loop should terminate depending on the value of the SIGN flag.
4. The program must be fully documented. This includes a complete header block for identification, description, etc., and a comment outline to explain each section of code.
5. The usual requirements regarding documentation, readability, user-friendliness, etc., apply.
6. Submit your text code file (*.asm*) to Canvas by the due date.

Notes:

1. There are no new concepts in this programming assignment. It is given for extra practice, to keep MASM fresh in your mind while we study internal/external data representation.
2. This is an integer program. Even though it may make more sense to use floating-point computations, you are **required** to do this one with integers.

Example (see next page)

Example (user input in ***italics***):

```
Welcome to the Integer Accumulator by General Kenobi
What is your name? Grievous
Hello there, Grievous

Please enter numbers in [-88, -55] or [-40, -1].
Enter a non-negative number when you are finished to see results.
Enter number: -15
Enter number: -100
Number Invalid!
Enter number: -36
Enter number: -10
Enter number: 0
You entered 3 valid numbers.
The maximum valid number is -10
The minimum valid number is -36
The sum of your valid numbers is -61
The rounded average is -20
We have to stop meeting like this. Farewell, Grievous
```

Extra-credit options (original definition must be fulfilled):

1. Number the lines during user input. Increment the line number only for valid number entries.
2. Calculate and display the average as a floating-point number using FPU operations, rounded to the nearest .001. This calculation would need to be done in addition to the normal program functionality, with the result printing after the rounded integer average.

To ensure you receive credit for any extra credit options you did, you must add one print statement to your program output PER EXTRA CREDIT which describes the extra credit you chose to work on. You will not receive extra credit points unless you do this. The statement must be formatted as follows...

```
--Program Intro--
**EC: DESCRIPTION

--Program prompts, etc—
```