CPEN 3710 **Lab Exercise 6**

Conditional Processing; Using Irvine's Procedures for Input and Output

Required Materials:

- Your textbook, *Assembly Language for Intel-Based Computers* (6th edition)
- Removable or network device (Flash drive, memory card, MyMocsNet account mapped to a drive letter, etc.) for storage of your programs
- These instructions
- Intel-compatible, Windows-based personal computer (like the ones in EMCS 306) with text editor, MASM 6.15, and Microsoft Visual Studio or CodeView (if needed for debugging)

Preparation for Laboratory:

Read the material on conditional processing in sections 6.1-6.5, pages 180-211 of your textbook. Also review the coverage of Irvine's pre-written procedures in section 5.3, pages 134-157. Read the following instructions and design an Intel x86 assembly language program to perform the required task.

Instructions:

Write an assembly language program that repeatedly prompts the user to enter signed decimal integer numbers. The program should be run from the command prompt, output a text prompt to the screen, and then wait for the user to type in a number followed by the Enter key. (The legitimate range of user input values is any signed integer that can be represented in 32 bits.) After each number is entered, the program should determine and display the following information about the number: whether it is positive or negative; whether or not it is evenly divisible by 16; and whether or not it can be correctly expressed in 5 decimal digits (in other words, whether it falls in the range $-99999 \le \boldsymbol{n} \le +99999$). For example, if the user entered the number +5, the output generated by the program should look similar to this:

```
+5 is a positive number.
+5 is not evenly divisible by 16.
+5 can be correctly represented in five decimal digits.
```

On the other hand, if the user entered -328000, the output would look like this:

```
-328000 is a negative number.
-328000 is evenly divisible by 16.
-328000 cannot be correctly represented in five decimal digits.
```

After determining and displaying the above information, the program should prompt the user to enter another number. This process should continue until the user enters the value 0 (which is neither positive nor negative). At that point, execution should terminate and return to the command prompt.

Assemble, link, and test your program. Make sure to **test it for each of the *eight* possible input cases** (permutations of positive vs. negative, evenly divisible by 16 vs. not, fits in 5 decimal digits vs. too positive or negative to fit in 5 digits), **as well as the special case of 0 which exits the program**. When you are sure it is working, run it from the command prompt and **capture a screen shot(s) of a test run that illustrates <u>all nine</u> possibilities and the corresponding outputs.**

To Hand In: *(due no later than 3:00 p.m. Tuesday, October 14)*

Turn in a printed copy of the **program listing** (.LST file) for the program you wrote. **(Be sure to follow the documentation instructions given in the "programming style and grading guidelines" handout.)** Also print the **output you captured from the screen** when the program was run. *Staple the program listings and output together neatly* and submit them by the date and time indicated above.