Intro (X86) Assembly Language and Computer Architecture

Checkpoints.

Basic Requirements for All programming exercises:

1. You Must submit individual .asm files, one per programming exercise.

2. Due to the deficiency in the Canvas Document Viewer - in relation to some student computers which are configured in foreign languages mode, you Must Also save the .asm files in pdf format and submit them - to ensure that they can be viewed in Canvas and I can write my remarks over your code, when needed.

3. Without Rich comments on it, an assembly source is deemed un-readable. Your assembly code should be **Fully Commented - accounts for 25% of total points.**

4. Proof of your solutions and test results: To show the operation of your program and to verify results from execution, you should use procedures **WriteString, WriteInt, DumpMem, DumpReg and Gotoxy**(for this very assignment), whichever applicable for a specific exercise, to display the results. See Examples in Ch 5 in the text, or the PowerPoint here in Canvas, on how to call these procedures(Hint: you have to find out which register(s) to be loaded with the argument values). And, again, you can find sample codes in the template file given in **Assignment #2** : apply_Irvine_libs_tmpl.asm .

Make sure the library file Irvine32.inc is included on top of your program; this .inc file includes all the procedures mentioned above, and more. **Without integrating the calls to these procedures to output the results onto the console, you Will Not earn full credit. In addition to the .asm file, please submit the screen captures - saved in pdf format - showing the successful Build and execution with correct resultant data.**

Programming Exercise 1 - SIMPLE ADDITION (1) [10 pts]

Write a program that 1) clears the screen, 2) locates the ***cursor*** near the ***middle of the screen***, 3) prompts the user for two integers, 4) adds the integers, and 5) displays their sum; all of the prompts and output line should be displays should be at the middle of the screen, line by line.

Hint: Use the **Gotoxy** procedure from Irvine32.lib to locate the cursor on the screen; see PowerPoint slide #9 of Ch 5, in Module 6, and Section ***5.4*** in the ***text***.

Programming Exercise 2 - RANDOM STRINGS [15 pts]

Create a procedure that generates a random string of length *L*, containing all capital letters. When calling the procedure, pass the value of *L* in EAX, and pass a pointer to an array of byte that will hold the random string. Write a test program that calls your procedure 20 times and displays the strings in the console window.

Programming Exercise 3 - COLOR MATRIX [15 pts]

Write a program that displays a single character in all possible combinations of foreground and background colors (combinations: 16×16 = 256). The colors are numbered from 0 to 15, so you can use a nested loop to generate all possible combinations.