**CSC 3100 LAB 1**

**Lab Instructions:**

Submit assembly code along with a brief explanation of the code’s logic and how it relates to the core concepts covered in the lab objectives, make sure to include register snapshots showing that your code actually works.

**Lab Objectives:**

1. Explore core concepts related to assembly language programming and its role in the broader spectrum of languages and applications.
2. Gain insights into the underlying hardware associated with x86 assembly language, emphasizing its direct communication with the machine.
3. Understand the basic operations inside the processor during the execution of instructions.
4. Learn how programs are loaded and executed by the operating system.
5. Write a program in assembly that calculates the expression A= (A+B) –(C+D)

**Code:**

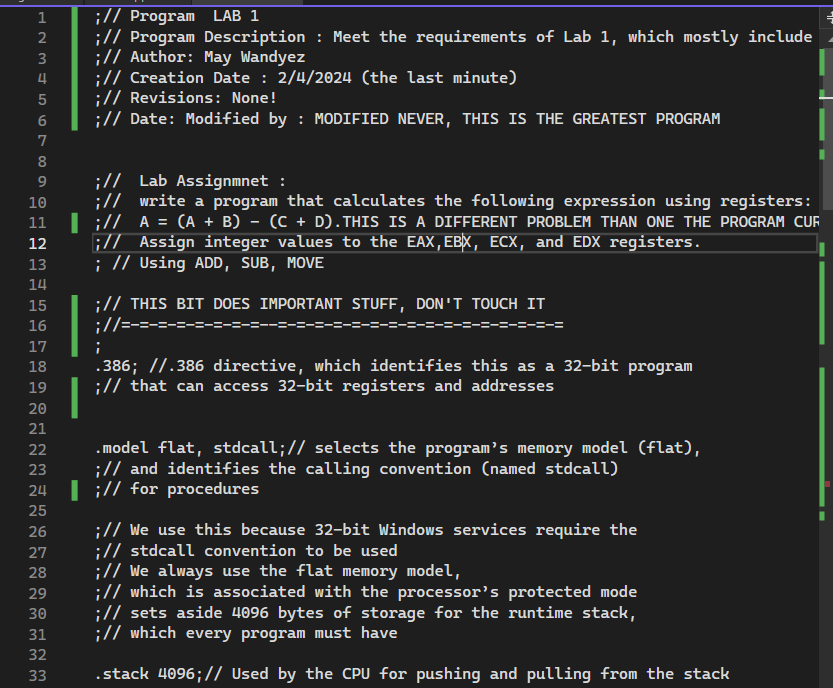


Figure 1 – Code lines 1-33: This section of the code gives insight onto how assembly actually works. Unlike C++ you have to manually set up that the program is a 32-bit program, you have to select the memory mode, and also specify the stack. This is very annoying – and is probably why we all use higher level programming languages that manage this for us.

A screenshot of a computer program

Description automatically generated

Figure 2 – Codes line 35-64: This section includes the description of how to create function in assembly, which includes the required keyword, and a list of the input parameters after the comma. This section also includes the .data section where the actual data is considered – this is where you declare your variables. Apparently C is a keyword, you cannot name your variable C. This caused a lot of problems and I wish I had known that earlier.

A screenshot of a computer program

Description automatically generated

Figure 3 – Code lines 64-86: This section includes adding initial values to variables in assembly, it should be noted that this COULD have been done in the data section with the initial values for the code, however I thought this was causing me an error (it wasn’t) so I tried it an alternate way. This was done by moving values onto the registers where the variables were stored. I later moved the values of the variables onto the register eax-edx

A screenshot of a computer program

Description automatically generated

Figure 4 – Code lines 87-108: This code concerns the actual equation. Unlike in C++ or other languages, in assembly the calculations have to be completed piecemeal on the register (as far as I know) this fulfills the objective of understanding how basic operations are performed inside the processor during the execution of instructions because complicated mathematical formulas have to literally be broken down to their basic operations inside of the registers. It should be noted that the desired formula should be (4+3)-(2+1) = 4, and if we check the registers, this is exactly how they are.

Results:

