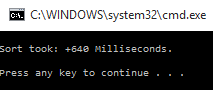
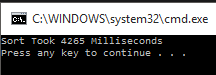
Philip Geramian

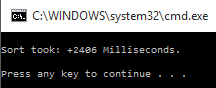
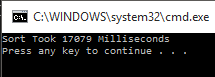
CSE-671

Lab 3\_2 (Bonus)

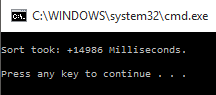
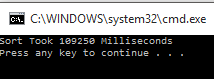
For this lab we had to implement the Bubble sort algorithm using x86 assembly code, as well as in C++. The purpose of doing this is to see how that when a compiler takes your C++ code and converts it to assembly it may not be doing so in the most optimized way. This in turn will result in a slower execution time. For this lab we made use of the Irvine32 toolkit for x86 assembly as it provides us with some more complex procedures that can be used to write this program. For example this can be seen with the Random32 call, as this is not an instruction in the x86 language. Another advantage in using the Irvine32 toolkit is that it comes with a preconfigured Visual Studio project that is ready to write our assembly code in and assemble it. The Bubble sort algorithm has a time complexity of O(n2), which results in poor execution time on large sets of data, so in order to better see the differences in runtimes between both of our programs it is best to run the sort on a large array of random numbers. To do this we simply generate a large enough array and fill it with randomly generated values. To further expand my data set I ran this comparison on arrays of 20,000, 40,000, and 100,000 values. The execution times can be seen below.

20,000 ASM 20,000 C++

40,000 ASM 40,000 C++

100,000 ASM 100,000 C++

As we can see in the graph below, there is a noticeable difference between the assembly code runtime and the C++ runtime:

This graph clearly shows the improvements gained by using the assembly bubble sort, as we can see the exponential increase that comes with the algorithm, and thus we can see the time saved by using more optimized code.

From this we can learn that the compiler used by Visual Studio is not making the most optimized code when going from C++ to assembly, while when we write the assembly code ourselves we can see how we can make any optimizations that we want, thus decreasing our code runtime.