**Project #0: Assembly Language with IAR Embedded Workbench**

**The Tutorial:**

Our impression of the tutorial was mixed. Though it provided a decent overview of both the board and the functionality of the Workbench environment it was very much lacking in regards to instructions of how to proceed upon the creation of a new project. This was especially true with the lack of information regarding the thumb assembly language’s syntax. Very little information was provided on important matters such as preprocessor directives, access level modifiers, vector tables, operations and their operands, and the overall general boilerplate format for writing a thumb assembly program. The basic introduction to the IDE and the basic assembly language introductions provided within the help files were much more informative, and coupled with the code snippet provided within the lab could be used to come to a general understanding of how everything worked.

**The Assembly Program:**

The assembly program was fairly simple. The hardest part was setting the program up for thumb instructions as well as understanding the basic vector table. We used the stack window to demonstrate the output of the program. The stack trace and the source code for this program are shown in Appendix A, the debug log and the build log are shown in Appendix B. The disassembly of the program is shown in Appendix C.

**The C++ Program:**

The C++ program was also simple. The biggest advantage to the C++ program was the ability to exclude the preamble necessary at the beginning of the assembly file. The source code and the output of this program are shown in Appendix D, the debug log and the build logs are shown in Appendix E, and the disassembly of this program is shown in Appendix F.

**The Comparison:**

Again, with the exception of the preamble in the assembly code, the source code of these two programs is very similar. As shown by Appendix A and D, there are about 8 effective lines of code in each source. The size comparison of these two programs is astounding. The project folder, including the workspace, of the assembly language program is about 136 kb while the C++ program is about 1.23mb. That means the C++ program is nine times the size of the assembly language program. To be made fair however, this comparison of file sizes must come with the caveat that the iostream system library was linked with the C++ source code and the extra instructions used to display the numbers to the console could have contributed to the codes final size as well. Still, after taking this into account the assembly program has a significantly smaller memory footprint in comparison. The C++ program was significantly easier to program solely on the basis that there was no setup involved other than including the aforementioned iostream library. If you compare the disassembled code as shown in Appendix C and F, they are really similar, at least in the main section of the code. If you notice the assembly code in Appendix C, preceding and following the main section of the code is just a bunch of operations that deal with zeroes. This leads us to believe that this code serves no real purpose. Now if you observe the code in Appendix F, the main section is nearly the same, but the preceding and following code actually appears to be functional. This also doesn’t mention the branches that are going off in every direction in the C++ code. This leads us to the conclusion that the assembly code as opposed to the C++ code is much more efficient and executes at a much faster rate.

**The Conclusion:**

If the solution to a programming problem is time sensitive and speed/storage space is not very crucial, then C++ is the way to go. If storage space is a problem and speed is crucial, assembly language is by far the most effective solution.

**The Contributions:**

Zac wrote the C++ program and helped in contributing to the report, Andrew wrote the assembly program and assembled the report, Brandon was a late joiner to the group and got up to speed on IAR programming.