My contribution to this project primarily consisted of backend data processing. I had two major tasks: implementing the C code compilation process, and parsing the individual assembly instructions to determine the appropriate outcome of each one. This first task involved setting up the basic API endpoint on the server to accept a request from the client, and then proceeding through the process of compiling the C code and parsing out the hex data. This involved the use of various Linux commands to compile the code, determine what functions were present in the code, and then extract the necessary binary code from the compiled object file, before sending this data back to the client. The second task involved using a case structure and the Iced API to determine the numbers and operations involved with each, and cause the necessary effect. This also involved working carefully with the different kinds of data types involved.

This project built heavily on my capabilities from CS 2011 of basic computer assembly architecture. This understanding of how assembly code, and computer registers and the stack work was essential to understanding how to implement the effects of each assembly instruction and visualize them. I also utilized some of the skills I had built from my last co-op with CAS when it came to client/server interactions, to create the API endpoint on the server to compile the code. Many of the challenges surrounding these tasks involved having to deal with different possible data types with 32-bit and 64-bit numbers, and understanding how to utilize the Iced API to get necessary data about each instruction. Ultimately, this did reduce the scope of our final product, as this took a while to work out, but we were successful in implementing a reduced feature product.