Features of the proposed solution

It is important to have a strong initial idea of what the final product will contain, with a list of all the desired features. Below are the lists of features for each of the three programs in my project, showing both the main, necessary features and also some extra, ideal features.

# Compiler

To be reasonably complete, the compiler has to have all of the following features:

* Take an input in the C language (or a subset of it) and output assembly code
* Recognise major C constructs such as variables, arithmetic operations, arrays, if statements and loops
* Show the user what their code looks like after initial preprocessing, with comments on the changes
* Show the user the tokenised form of their code, i.e. the code split up as a list of tokens
* Display a syntax tree of the code to the user
* Show the process of working up the tree, compiling each expression and statement and showing records of variables, etc
* Present output in the form of assembly code
* Present both input and output in text boxes

Ideally, I would also like it to be able to do the following:

* Recognise more complex C constructs, like pointers, structs and functions
* Take input from and send output to files directly
* Run the compilation at full speed usefully, without showing the graphics

# Assembler

To be reasonably complete, the assembler has to have all of the following features:

* Take an input in the custom assembly language used and output bytecode
* Break up the code into lines and tokens, displaying this to the user (e.g. for each line, show the label, command, operands etc)
* Show the numeric forms of the commands and opcodes
* Show the records of where the labels point to
* Show how the lines are packed into the correct number of bytes and put together into the final program

Ideally, I would also like the following items to be a part of the program:

* For each line, offer an English version of the command to show the program understands it
* Make the GUI interactive
* Allow the assembler to run at full speed without a GUI

# Interpreter

* Run the bytecode given to it successfully
* Display a graphical view of the parts of the CPU doing their work for each instruction
* Animate the movement of numbers around the CPU
* Show a section of memory in the GUI and show accessing and editing of memory locations

If I am able, I also want to add the following features:

* Run the code at full speed without showing a GUI