Usability features

The usability features of my program are an important consideration. These are the considerations made when designing the User Interface to make sure the program is easy to follow for users, and made as accessible as possible. In this section, I have made an approximate mockup of the design of each program, and annotated each of the design decisions involved. All of these will be part of the same program. The program will be designed with standard Windows interfaces, notably the tabs in the top left, which will across all of the sections consist simply of File and Help. The menu will be arranged like:

* File
  + Open C file – Opens up the compiler view with the code in its display area
  + Open assembly file – Opens up the assembler view with the code in its display area
  + Open bytecode file – Opens up the assembler with the instructions loaded into the memory addresses
* Help
  + About – Displays a popup window with a basic description, the name of the creator and a link to the documentation
  + Documentation – Loads up the program documentation

# Compiler

The compiler screen, displayed when the user chooses to open a C file, looks like this:

File

Help

Code display area

Working area

Assembly

Tables/memory

The parts labelled are:

* **Code display area** – Contains the original C code
* **Working area** – Used to illustrate and explain the current thing that is happening. For example, a line of code being broken up into its tokens.
* **Tables/memory** – Records information the compiler has stored, such as the table of variable names.
* **Assembly** – The assembly code as it currently stands

# Interpreter

The interpreter is built with a similar design, for the most part. It has two possible views. The first is opened when the user chooses to open an assembly file, and continues being used until the process of converting the instruction list to binary begins. It looks like this:

File

Help

Code display area

Instruction list

Working area

Variable table

Label table

The boxes in this case have the following purposes:

* **Code display area** – Shows the assembly code
* **Working area** – Displays the current action taking place, likely showing the line of code, possible colouring certain parts, and a description of what is happening
* **Variable table** – A view of the variable table as it currently stands
* **Label table** – A view of the label table as it currently stands

The label and variable table sections merge and turn into a memory table view when the program reaches that point.

The second screen is simpler. It shows the instruction list on the left third of the screen, what byte value each part of it has in the middle third, and a hexadecimal view of the current bytecode in the right third.

# Interpreter

While still happening in the same program, the interpreter view is quite different. When the user chooses to load up a binary file, it loads up and prepares. The design is roughly based on the LMC. It looks like this:

File

Help

Memory view

General register table

MAR

MDR

CIR

Active instruction description

ALU

Output

Input

PC

Commentary box

This contains far more parts, and should also have illustrations of buses connecting different items. The parts are explained below:

* **General register table** – Split up to show the values of the 8 main registers, or if some of those have been broken up to use the smaller registers within, then to show those instead
* **Output** – Shows a scrollable list of all the things that have been designated for output from the program
* **Input** – A text box where the user types when the program asks for input
* **Commentary box** – A description, in an English sentence, of the process that is currently happening
* **ALU** – An illustration of where values go when maths is done on them
* **PC** – Program counter; displays the memory address of the next instruction
* **MAR** – Memory address register; when a request is made that involves accessing memory, the relevant address is stored here
* **MDR** – Memory Data Register; a buffer to store information going to or coming from the memory
* **CIR** – Current instruction register; the register that stores the current instruction
* **Active instruction description** – An English description of the command in the CIR
* **Memory view** – A scrollable table showing the values stored in memory