The design of our simulator is aimed to mirror the design process of the CPU itself. The registers, being without function, and just storage of data, live within our simulator. We had functionality wrapped around these registers, both in the simulator to display data, as well as in the CPU to decode and work with the instruction data we find in memory. We created a separate module for memory, as a separate housing of data, outside of the simulator, but also to mirror the CPU model, such that memory is a concept that sits outside of the processor, and is just merely there to be interacted with to load in and store from registers. Each of the registers is designed to display the number of bits that reflect the maximum storage capacity for the registers. So a register that can support 16 bits displays 16 zeros within the simulator, a register that can only support 12 only displays 12 zeros, and so forth. We need the interface of the simulator, to be relatively straightforward, to make the input obvious to the user at the top, to make all the registers labelled, and then to put the extra buttons, larger and obvious to the user to load the program, run it, or step As this project progresses, we also included thought towards how we improve upon this design. By creating a separate processing file, in a separate memory module, we should be able to expand to cache-based memory, expand the number of processes that we can support in the instructions that we can execute, as well as being able to increase the complexity of programs, jump codes, and so on. Although we discussed design that was not based around the GUI., this first part of the project was a GUI, focused, effort, and priority was aimed towards functionality over scalability