**Assignment 2 Project Report**

**Nicholas Petosa**

**Design Options & Approaches**

The assembler was implemented in Python and can convert assembly into a MIF file, which is loaded onto the board. We built a multicycle processor with Verilog which has an ALU (implemented with a case statement), a register file (implemented with an array of 32bit registers), a dmem and imem (each implemented with an array of 2048 32-bit registers), a PC, a MAR and IR (all 32-bit register). The processor is also responsible for multiplying the JAL and BRANCH immediates by 4. The ShOff does this by left-shifting the immediate by 2 and putting it on the bus. An interesting difference between our architecture and the standard LC-2200 architecture is the use of opcode to select an ALU function. We can forgo ALUFunc because almost every instruction uses the ALU for one type of operation. For the exceptions (Branches, which need to check equality and then add), there is a special altFunc signal which indicates to the ALU to use the secondary function instead of the first. To implement memory mapped IO, we check the MemEnable wire to see if an IO immediate is present. If that is the case, for LEDR and HEX, we set the desired pattern at the end of the file. For KEY, we modify DrMem to populate the bus with the current KEY state rather than a memory location.

**Challenges**

When creating the processor, one of the most significant challenges was debugging. Compilation was very time consuming, so we wanted to make sure we got the most information we could between changes. To do this, I implemented a timer similar to that in assignment 1 in order to slow down the clock pulse to 1 tick per second. On each tick, we made the HEX display show the current value of the bus and made the LEDR show the current microstate we were on in binary. We then assembled custom ASM files to debug certain instructions. This enabled us to build the compiler piece by piece and helped tremendously when tracking down where things went wrong.

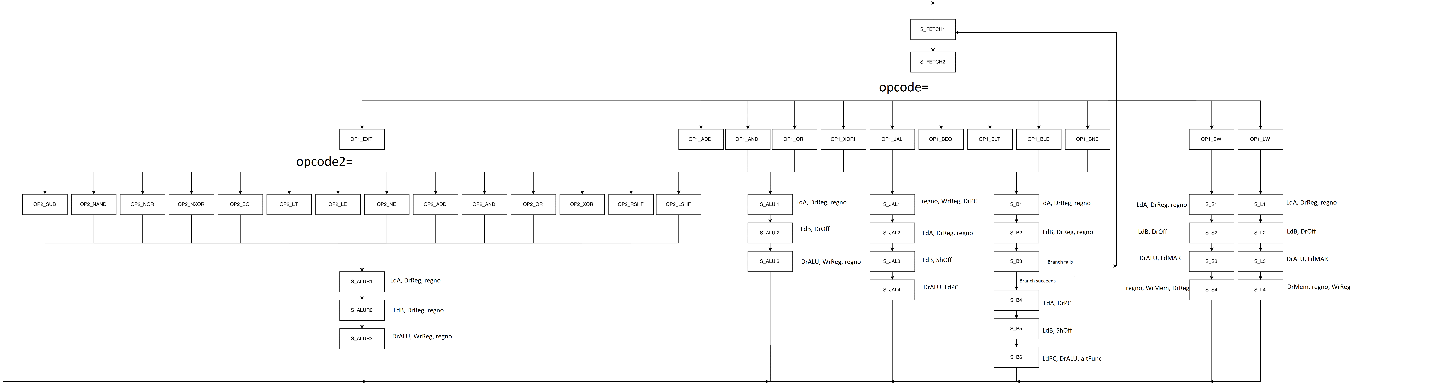
A more specific issue we had was figuring out exactly how addressing worked for our ISA and what that meant for implementation. Specifically, we did a lot of trial and error until we realized the purpose of ShOff was to multiply BR and JAL offsets by 4. At one point the processor was successfully completing the median program but was failing the test program. We ultimately narrowed this down to an assembler issue.

**Contribution: 50%**

I created the processor in Verilog and worked to debug it with my partner.

**State Diagram**

This is much too small to read in this PDF, so we included the full-sized PNG as part of the submission (statediagram.png).



**Data path**