**Hannselthill Camacho**

**Jose Delgado**

**ECE176**

**3/25/20**

**RISC Processor**

The motivation for designing a RISC processor comes from the year of learning about several RISC architectures and using them for labs. They are very efficient at performing a single task or a few tasks with one goal in mind. This would allow us to gain a deeper understanding of the innerworkings of a processor.

A Reduced Instruction Set Computer Processor contains a very limited amount of instructions. This reduces the amount of overhead and space involved when compared to a CISC (Complex Instruction Set Computer). Which makes a RISC processor suitable for resource limited applications. The instructions would follow the NIOS II instruction set and 32-bit data width. The NIOS II ISA would be used as a reference to implement the processor. Another reference I would follow is chapter 9 in Digital System Design using Verilog. Depending on complexity I might implement a 16-bit implementation instead of 32-bit. This would allow for easier debugging when dealing with a smaller data portion to perform operations on.

The control portion of my code would include a state machine which I would implement methods from 106 to optimize. This would send signals to the Data portion on what type of instruction to perform and what to do with the operands involved. The first instructions I would test are ADD and SUB to check if the processor is operating as intended. The test performed would involve how the processor handles an overflow, underflow, and normal conditions.

An alternative and possibly an extension of my project if I get the processor working would be to implement an Instruction Set Architecture also following the previously provided references. Instructions at the very minimum would be ADD, SUB, DIV, MUL, STORE, LOAD. Immediate versions of the same instructions would also be implemented.