Jack Barkes, Matthew Mizumoto

Written Report

The current issue with our design stems from the use of branching, resulting in infinite loops during tests. We believe the branch code in the ISA is correct, indicating that the problem lies within the machine code and assembly. Our lookup table is accessed by first loading the desired index into register 14. A branch instruction is then called, using this register to locate the value in the lookup table and perform the appropriate jump. The most probable error is what we are comparing when trying to do a branch instruction. Currently when calling a branch instruction the only argument taken in is a register and then we compare that argument register to the accumulator register. The problem may lie in our program implementation which ends the program when we exceed a certain index in data memory, however we likely are having computational errors that are reaching that end goal. Another potential problem is how we are calculating the jumps. In our code, we calculate the distance from the branch instruction to the target instruction and store that difference in the lookup table. This process is prone to human error, as incorrect values may be stored due to typos or miscounting. Additionally, there could be errors in the Verilog code, such as typos or incorrect register usage. Given more time, our plan to fix this would involve focused debugging of the assembly and machine code. We would put in specific breaks in various functions to verify the correct execution of jumps. Once this is confirmed, we would analyze the two loops to ensure they implement the correct exit conditions. During this analysis, we would use display values in Verilog to track the code's behavior and identify the root cause of the problem. We can additionally look at the waveforms in the testbench to check the current values at given times during our program execution. This systematic approach would allow us to isolate potential problem areas and effectively address the issue, ensuring robust and reliable code execution.