a) What did you learn from this project?

We chose to build a 5-stage pipeline, which presented challenges that are not present with the single-cycle or multi-cycle. We learned valuable lessons in debugging complex issues while dealing with the control and data hazards that occurred with our system as we continued our development. Our lectures in class taught us what we needed to solidify our nebulous ideas into concrete solutions to our pipelines. One example of this being the discovery of the our data hazard between instructions wanting to write then read a register immediately. The very next lecture presented us with the concept of bubbling and the forwarding unit.

b) What would you do differently next time?

First, we would have developed a standardized format for instructions, which would reduce the number of multiplexers required for the decode stage. Second, we would have implemented our control logic as a micro-programmable ROM array. This would have simplified the many alterations to our control we had to make as we added hardware solutions to the pipeline's hazards. Third, we would have implemented more complex branch prediction logic, as our current solution is to assume that the branch is always taken, flushing the pipeline is if it was not supposed to be taken. This performs extremely well in loops, but fails ~50% of the time for condition statements. Otherwise we would have actually done the branch comparison and decision in the decode stage. Fourth, we would like to attempt development of a second parallel pipeline, for the same reason we chose the pipeline at the start of the semester, the opportunity to learn more and the challenge of the problem.

c) What is your advice to someone who is going to work on a similar project?

First, keep a copy of each version of your project of from each stage. Second, try a lot of weird series of instructions, some of them break your project in ways you would not predict. Third, use a micro-programmable control unit, its will save you a lot of time debugging compared to the complex expressions that you could otherwise use. Fourth, specifically for this class, use the Quartus II software and the Modelsim simulator from the beginning, as you have to use them for the final additions to the project, and they have slightly different rules for syntax in VHDL, so transferring over at a late stage can be a headache.