## 1. Project Summary

**In 1 paragraph describe what you accomplished in the Class Project. This should be written as if you were going to give it to a job interviewer, and should NOT be a Phase-by-Phase description of the project.**

In the class project, I designed a RISC-V processor. I designed block diagrams to effectively visualize the interaction between the signals, registers, pipelines, control blocks, multiplexors, etc. in the processor. Furthermore, I learned how to read and write RISC-V instructions and I designed test cases for testing the functionality of my processor. I learned in-depth about how the machine code for each instruction is decoded, and I designed a decoder that sets the appropriate destination register, source registers, immediates, and other registers/signals according to the corresponding bits in the 8-byte word associated with each instruction. I created a working 5-stage pipeline, and I learned about which steps are completed in which pipeline stage and how information is communicated between stages. I also learned about the different kinds of hazards present with pipelining and implemented hazard detection and resolution for each. I also learned in depth about program control and how a processor evaluates branching, jumping, and how the data hazards that these can present are resolved. Another important aspect I practiced in the project was good coding practices for simulating my processor in C, using enums, default cases, and appropriate registers/signals for each pipeline stage.

## 2. Challenges

**Describe at least three elements of the project which were challenging, why they were difficult and what you did to overcome the challenges.**

1. One part of the project that was challenging was debugging. It was easy to know that the code was not working, but very difficult to figure out why. It was hard to keep track of which value was in which stage of the pipeline and where the source of error was coming from. With so many different signals and registers being pipelined and set by control blocks, multiplexors, and the ALU, it was hard to follow where the code was going wrong. To overcome this challenge, I became familiar with what goes on in each pipeline stage, the order in which each pipeline stage is executed, when registers/signals are updated, and I learned how to use breakpoints to track the state of the code at different points in execution.
2. Another difficult aspect of the project was using Codasip. The main challenge was having to generate a new SDK every time I changed any code since this took around thirty seconds each time and greatly increased the amount of time I spent working on the project. There were also a lot of other obstacles with Codasip such as constant glitches, often showing incorrect/old values for registers and often not even showing the register

section at all. Codasip would go offline a lot and make it extremely easy to lose work. To overcome these obstacles, I was vigilant about saving my work and making sure that Codasip was online when I was working, I made sure I followed the instructions very closely, and, most of all, I was patient.

3. It was hard to estimate how much time it would take to complete each phase. I could breeze through most of the project but then spend an inordinate amount of time trying to debug a cryptic and obscure problem. For example, in phase 5, I spent as much time trying to fix one bug (that I eventually found out was because I declared a variable as a signal instead of a register) as I did on the entire rest of the phase. The way I overcame this obstacle was I regularly went to office hours if I got stuck and I allocated more time to do the phase than I thought I would need.

## 3. Takeaways

**Describe the three (or more) most significant things you learned in this project.**

1. One significant thing I learned in this project was the importance of investing time into creating a good block diagram. At first, I did not find the diagram very important and I rushed through it so I could get to the coding portion of the phase. However, I eventually found out that spending a lot of time making an accurate and readable block diagram saved me more time down the road with coding and debugging.
2. Another significant thing I learned in this project was the value of good coding practices. At first, I thought it was frustrating and time-consuming to create enums, default cases, define different names for equal sizes, etc. However, I realized that doing a good job on these parts was also an investment because doing so made my code so much more readable and made debugging far faster and easier. I also got into the habit of writing comments for the same reason.
3. An important thing I learned from this project was about the RISC-V instructions themselves. I took Computer Systems last semester and really did not understand the x86 assembly instructions and how they work, but now I feel like I can read a series of assembly instructions and actually figure out what the program does, even with branching and memory operations.
4. An additional important lesson for me from this project was the importance of following the phase instructions. I found that if I read every line in the instruction write up a few times and made a conscious effort to understand why I was doing what I was doing, I gained a much stronger understanding of the material. I also ended up needing to correct fewer bugs, and when I did debug, it took less time, since I had a greater understanding of the cause-and-effect of different registers/signals/pipelines in my code.

## 4. Suggested Improvements(Optional)

**Describe any suggestions you have which would make the Class Project a better learning experience.**

I wish the instructions for each phase were clear about when it was telling us how to do something versus if we had to figure it out on our own. Often I could not distinguish between the two. It would have been nice if the instructions told us that we had to do something but not how we do it because oftentimes I found myself scratching my head and then finally implementing something that the instructions did not even mention but expected me to do anyway. Also, I think something that would make this a better learning experience is if Codasip was less glitchy and unreliable. This is obvious, but I think it made it significantly more frustrating and harder to learn. I also think that this class could benefit from having more TAs/Office hours because most of the few office hours offered were during my other classes and I often had to skip class to go to office hours. I rarely found Kieth's youtube videos useful; I found that they covered the parts of the phases that were either the easiest to understand in the first place or just not very useful. It just seemed like it was redundant information that was usually already covered in-depth in the phase instructions, and did not warrant a video. I am not complaining that there were extra resources to help us, but rather I am commenting on how much more effectively I could have learned if these videos ever addressed a harder aspect of the phase or a part I was stuck on.