# CprE 381: Computer Organization and Assembly-Level Programming

# Project Part 3 Report

Team Members: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Alek Norris\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Drew Kearns\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Project Teams Group #:\_\_\_\_\_\_\_\_\_Term Proj1\_2s\_07\_\_\_\_\_\_\_

1. **Introduction -** *Write a one paragraph summary/introduction of your term project*

For our term project in CPR E 381, we were tasked with building a basic processor from scratch following the MIPS ISA. The project was divided into five parts. In Part One, we created basic designs in VHDL, such as multi-bit adders and gates, honed our debugging skills, and learned to track and resolve bit-level issues. Part Two escalated our challenge, requiring us to develop a simple adder-subtractor ALU and expand a single D-Flip Flop into multiple D-Flip Flops to store up to 32 bits, forming the foundation of a single register. Additionally, we constructed a register file capable of holding 32 registers, each containing 32 bits. In Part Three, we implemented a single-cycle processor compatible with the MIPS ISA, enabling us to run programs. Parts Four and Five were combined but completed individually; we first transformed our basic single-cycle processor into a four-stage processor and modified our initial programs to include software data and control hazard avoidance techniques. Lastly, we tackled hardware hazard avoidance by integrating a hazard detection and forwarding unit, which could detect instructional hazards and either forward the data to mitigate the hazard or stall the processor and wait for the hazard to resolve that way.

1. **Benchmarking -** *Please generate a table for each of your final single-cycle, software-scheduled pipeline, and hardware-schedule pipeline designs.*

**A table with numbers and instructions

Description automatically generatedUPDATE IF WE FINISH FORWARDING**

1. **Performance Analysis -** *Explain in your own words why the performance was better on one processor versus another or why some applications may have had a smaller difference in performance between processors versus other applications.*
2. **Software Optimization -** *Describe one software optimization (i.e., assembly level software refactoring) that would improve the performance of software on the software scheduled pipeline relative to the others. Provide an estimate of the performance benefit this change could have given your specific benchmarks.*
3. **Hardware Optimization -** *Describe at least one different hardware optimization for each design that would improve its performance. Optimization cannot be turning it into one of the other designs. Certain optimizations can be beneficial to more than one design. Choose one design on which you would apply the optimization. Briefly list the specific set of changes you would have to make to your design to accommodate each optimization (a figure would be helpful). Provide an estimate of the performance benefit each optimization could have given your specific benchmarks.*
4. **It Depends -** *Describe your approach to building these programs. If one of these cases is impossible given your designs, argue quantitatively why that is the case.*
5. **Challenges -** *In at least three detailed paragraphs, describe the three most critical challenges your group faced, how you resolved them, and how you could avoid them in the future.*
6. **Demo** - *Each member of the project group will be required to be present for the demo, which will take place during regular lab hours. During this time, you will describe the various design tradeoffs of your project parts, describe how they compare to each other, demonstrate simulations of your benchmarked applications, and discuss potential optimizations.*