**CS 4341 Team Project Report**

1. Team Member Names and Roles:

**(Fill this section individually)**

Satsuki Ueno-

Lucan Hagel-

Alexander Pabst-

Tyler Hagen-

Jason Rummel-

1. Project Description

Our project is a Verilog implementation of a MIPS single cycle processor. The entire project has been designed to match the basic specifications of a 5 stage MIPS processor. Each MIPS module has been separated out into their respective modules in Verilog code, with the identical inputs and outputs you would expect. To see fine details of the implementation, open the Documentation.docx file.

1. Project Implementation

Our project was implemented by a safe, testable system design process. Our first step is to design each module by their respective specification before implementing them in code. This was easily done since the MIPS processor specifications and design were easily modelled due to our previous knowledge from our computer Architecture class, as well as the MIPS processor design slides.

Our next step was to design the specifications in code. Matching the appropriate inputs and outputs to their design counterparts, we created each module in separate Verilog modules. This makes it easy and simple to have interaction between our many modules, just like how different physical components would do in real life.

Our last step was to verify our implementation, which we did by including various Test modules with every single module in our code. This way, we can guarantee a good output for every single input we would receive. This also complies with real life validation, where test units are created to ensure that the modules would work correctly.

1. Process

We worked on our project in a way that people collaborate on code in real life. We used Git as a version control system to make group-wide changes that everyone can contribute to. This minimizes conflicts between group members that would overwrite each other’s code, and makes it simple to always have the latest version. This is entirely done through the internet as well, which means we can maximize our efficiency if we are unable to meet in person. Every change can be found on our repo, so if we have to rollback some code to a previous state, we can easily do so without keeping old copies.

1. Challenges

Although most of the development process went smoothly, the biggest challenges we faces were ensuring correct functionality and debugging. With such a low level language like Verilog, it’s difficult to actually write code and see that your code it doing what it should. Input and output is limited to the waveform and a very limited console. Making the Test modules definitely helped with this issue, but even then it took a significant amount of time to debug code. Just setting up the simulation takes a long time, where you have to force values for the inputs, apply a clock signal, and validate the output signals. This process must be repeated for every recompilation of the Verilog code as well, meaning each changes requires another simulation.

1. What I Learned

**(Fill this section individually)**