**Imperial Login: mjn18**

**Group Name: Lelolulalil**

1. **Group Working**

Understanding that pair-work projects could spell overlaps in responsibilities when handled poorly, we agreed to start from opposite ends and meet halfway. I created the simulator’s structural logic. Once I was done with that, Wei Loon would start on the instruction and memory logic, while I moved on to work on the testbench. When Wei Loon was done with his part, he would then help to generate test cases for the load and store instructions. This method proved to be really effective as there were minimal overlaps. The only waiting time was for me to complete the simulator’s structural logic.

1. **Time Management**

Despite this project being six weeks long, the Vector Calculus test between the formative assessment and the final submission that created headaches for Wei Loon and I in terms of time management. Nonetheless, we still managed our time decently. Being a person who does not enjoy leaving things to the last minute, I led the overall pace of the project. I ensured we had ample time left to upgrade our project before the respective deadlines instead of rushing to finish till the last second. This, together with Wei Loon’s efficient workstyle, made us complete the project with sufficient time remaining.

1. **Software Skills (e.g. debugging, design, coding)**

I designed the simulator’s structural logic (reading, decoding and linking the binary to the corresponding function). Despite just learning object-oriented programming, I used OOP to categorize functions better. In retrospect, this choice was right. Though the OOP was basic, Wei Loon had little confusion working on the instruction and memory logic, therefore saving us time.

I was responsible for creating the testbench and over two hundred tests. I initially had difficulty implementing the bash script as its syntax was confusing. After much patience and determination, I successfully iterated through binary files and produced output files to match the answer files.

1. **Tools and Infrastructure**

Initially, I used visual studio code for the simulator. However, there were constant problems when Wei Loon tried executing it on his MacBook. Hence, I switched to Ubuntu app on Windows to prevent further issues. If we had done that earlier, we would have saved much more time.

I worked on the generation of the testbench and test codes because of my Excel VBA proficiency. Instead of using third-party software, I decided to create my own automation process with VBA. This enabled swift translation from instructions into binaries, which saved loads of time compared to manually creating hundreds of testbenches.

1. **Understanding of Computer Architecture**

By using OOP to simulate each hardware block in the MIPS architecture like the registers, memory and ALU, I learnt about the interactions between these hardware blocks when instructions are called.

By creating my own testbench automation process with Excel VBA, I learnt in-depth about the bit distributions and bit positions of registers codes, op-codes, addresses, immediate and function codes for r-type, i-type and j-type instructions.

Furthermore, with the creation of the tests, I learnt about the importance of delay slots and how they would generate completely different answers for branch and jump tests if not correctly taken into consideration.