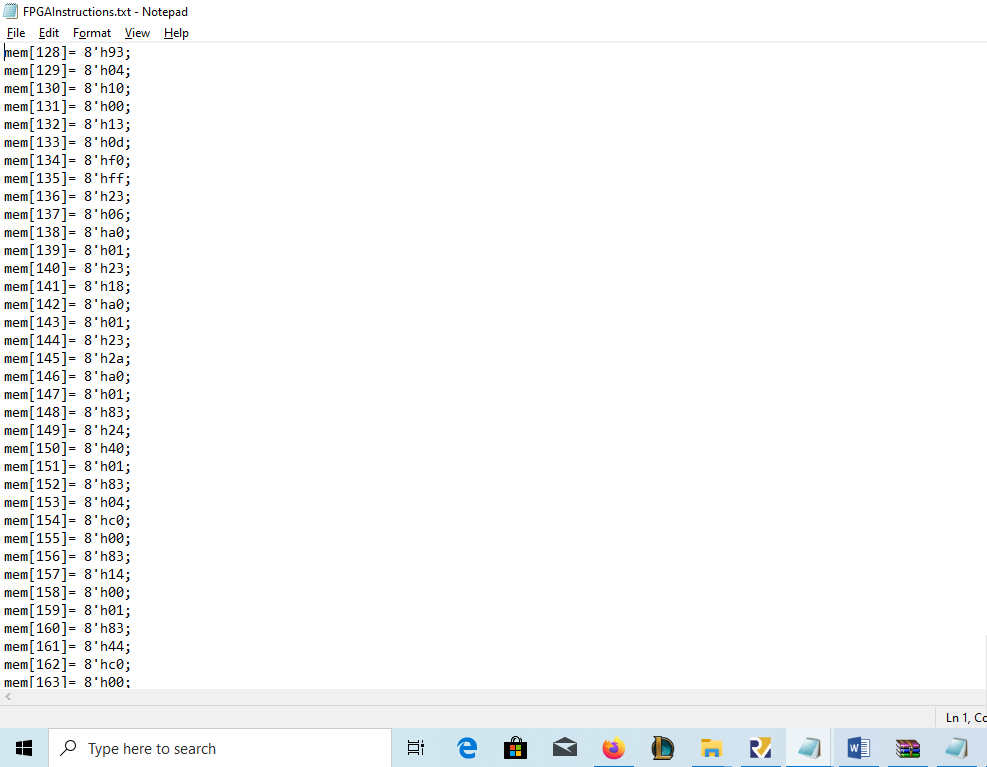
Architecture Project 1 – Milestone 4 - Full Pipelined Implementation with bonuses

Issues:

During our development process, we faced a few problems:

1. The first problem we faced was that the synthesis of the memory module took so long to synthesize. The reason for that was that the address that was used to access the memory was 32 bits. In order to solve this, we created an inner wire which only took the first 9 bits of the address and this inner wire was used to access the memory. This significantly reduced the LUTs used by the memory module and allowed the datapath to get synthesized and work on the FPGA successfully.
2. Another issue was testing the compressed instructions as there was no available environment that allows us to write compressed instructions. What we did was manually write 8 compressed instructions and incorporated them inside our testbench and they worked successfully.
3. Initially, we put the decoding module of the compressed instructions in the decoding stage. Even though this works, we moved it to the fetching stage for better efficiency (since the decoding stage has a relatively long process).
4. The final issue was converting the testbench into a synthesizable version which can be loaded onto the FPGA. To do that, we wrote a C++ Parser that takes the instructions in HEX and converts them into statements in the manner displayed in the following screenshot. Fortunately, it worked just fine.



Assumptions:

We are assuming that any unidentified instruction is a NOP instruction.

What Works:

According to our testbench, we believe that all of the instructions work as expected with valid hazard handling. Moreover, the compressed instructions are also functional. It is worth mentioning that for testing purposes on the FPGA, we are only displaying positive numbers to have the ability to display 4-digit numbers. Consequently, we will not be able to display negative numbers (BCD will treat them as unsigned numbers).

What does not work:

N/A