Donald J. Lauer IV

Programming Assignment 1--CSCI 560 (Spring 2023)

Assigned: January 19, 2023

Due: March 5, 2023

This is the write-up for Project 1 for Computer Architecture CSCI 560; for this project, I was required to create a cycle-accurate pipeline simulator complete with data forwarding and branch prediction. The simulator had to be written in either C or C++, and I chose to go with C++ because that is the programming language I am most familiar with. The first few lines are the “#include” statements. The #include statements are used for any function involving memory allocation, process, conversions, definitions of multiple functions, macro definitions, and built-in functions to handle characters. After the include statements, the rest of the code up until the int main () function is the pipeline registers that define the values of the different functions and some of the code fragments that our instructor provided, such as int field (), void printState (), void printInstruction ().

Inside the main function, the first few lines of code help keep track of the cycles and let users know if the incorrect file was inputted when starting the program. At the start of the main function, we initialize the pc, set all registers to zero, and initialize the instruction field, so the pipeline registers to the noop. Then I modify the main while loop that we need to run so that the program runs effectively. After the while loop, some code pieces allow the machine code file to be opened and loaded into the simulator and then store the information from the machine code file. Once the simulator has the necessary machine code file, the program starts running through each pipeline simulator stage. The simulator will each stage starting with fetch, decode, execute, memory, and writeback.

Once the simulator has been through all the stages, we can begin to account for any data hazards and how to resolve them. The hazard resolver analyses the previous state for any data hazard before any operations are complete. The three main phases of the hazard resolver are “Detect and Stall, Forward, and Detect and Squash. The detect and stall is necessary so the processor can stall any operations that cannot be handled. The forwarding function checks the pipeline registers for data hazards to increase recency so the most up-to-date values can be forwarded to the IDEX register. Finally, the detect and squash function detects if a branch should be taken and updates the state accordingly. Various notes are inside the program to help better understand how the simulator is run.

Throughout the process of this project, I came across many different challenges, such as how to get the branch prediction and data forwarding to work correctly. This project tested my coding knowledge because I had never had much experience working with computer architecture or pipeline simulators. However, the machine code file was the biggest issue I had to deal with during the project. When I began the test phase of my project, I did not know how to use the assembler or create the machine code file. After figuring out how to create the machine code, I started testing the simulator and had to change the puts of the program to make it run more effectively.

When executing the program with the provided machine code, the system would run the pipeline simulator infinity without halting the program. So, I decided to start a couple of machine cod test files, and the simulator would run properly without infinity running. I am unsure if this was an issue with the machine code that was provided for us or the assembler. I will provide the other test file so everyone can see what I am discussing.