CISC340 Project 3 - Pipelined Processor Overview

Tim Callies, Mugdha Danda

November 14, 2018

Description

This project is an extension of our earlier processor simulators, except this one runs with a pipelined design. It is split into 5 stages, and each stage runs a part of an instruction during each cycle. Before a cycle runs, the state of the processor is printed to stdio, and at the end of the program a number of other statistics are printed as well.

Each stage performs its task using old data as a reference, and the results are put onto a copy of the old data which overwrites data after each clock cycle. This ensures that different stages of the pipeline are not colliding with each other during execution. We also must account for possible hazards in our design, and these are all handled appropriately.

We also found the use of pointers in this project to be challenging, particularly the difference between $->$ and . following a variable. However, we were able to sort this out by looking through the rest of the code.

Challenges

The most difficult part of this project was accounting for the data hazards by implementing forwarding. We had to write a complicated series of conditional statements to ensure that we are getting the correct data from the other registers in the pipeline. Conceptualizing the logic behind this was difficult, and it required multiple iterations of pseudo-code before we had an answer that we were confident in.

The other part we struggled with was getting the statistical data to show at the end. We had a hard time determining what the 'retired' and 'fetched' instruction statistics were referring to. We had to consult the notes several times before we had a solution that made sense to us.

Remaining Issues

One issue that remains in our project is that we are using a series of 'if' statements to check for data hazards, rather than 'else if' statements. While the code runs correctly, if we wanted to get better performance we would want to use more else statements so that the machine does not have to calculate every conditional when running through the code.

We did come up with a possible solution for this, however we felt that the version of our

code that uses more 'if' statements was much more legible. Having clear and readable code was a bigger priority for us in this project.