Team Pied Piper Checkpoint 2 Progress Report and Roadmap

Progress Report:

Who worked on each part of the design:
Similar to checkpoint 1, each member of the team worked on each feature of this checkpoint. From creating the hardware paper design of checkpoint 2, to the actual programming of the hazard detection unit, forwarding unit, and arbiter and caches, to debugging and verification, all members were present during every meeting and provided equal contribution to the work needed to be done. As stated before, we did this rather than split the different features between group members as we all want to ensure we fully understand our design, and so debugging would be easier in the future as all members know all aspects of the code and design.

Functionalities we implemented:
For checkpoint 2, the functionalities that we implemented include, but are not limited to: arbiter, instruction cache, data cache, hazard detection unit, and forwarding unit. The arbiter and both caches were completed first as we wanted to ensure we were able to properly handle cache hits and misses, especially when we have both an instruction cache miss and data cache miss at the same cycle, before moving onto the more difficult features. The hazard detection unit came next and was implemented with the ability to detect control and data hazards. The hazard detection unit also implements a static branch-not-taken prediction which means we assume that the branch is not taken, however, if we find out later that it was supposed to be taken, the unit flushes the entire pipeline and loads the correct PC value into the PC register. The forwarding unit detects if there will be a write back to the register file and if the following instructions are dependent on that value. If so, we directly pass the output of the ALU or the data read from memory, if the instruction is a load, to the following instruction to bypass the writeback stage to speed up the process and avoid data dependencies.

Testing Strategy:
Regarding testing, we ran multiple tests. First, we ran the mp4-cp1.s testcode provided to us to ensure that we did not break any functionality that we implemented previously. Once this was verified, we began testing with mp4-cp2.s testcode. We ran this testcode on our working processor design from MP2 and went through the signals between the working simulation and our new pipelined processor side-by-side and compared how our signals differed from the working design. When we saw a value that was different, we would look at the PC value and instruction corresponding to the area of the error and begin debugging the corresponding stage or unit associated with that instruction. We repeated this process with our testcode, including ones we wrote specifically for this checkpoint, until our results matched the working design's results.

Roadmap:

Who is going to implement and verify each feature and functionality:
Like the previous checkpoints, each team member has agreed to meet up together to work on all features and functionalities together. Again, this is because we all are curious about learning how the processor works and want to take part in the implementation of all features. Furthermore, debugging is easier when all members are aware of how every feature works. Our current plan is to work over fall break to complete the advanced features and any necessary paperwork required for CP3.

What are those features or functionalities:
Features & Functionalities to be implemented for checkpoint 3:

- RISC-V M Extension (Basic Add-Shift) - 3 Points
- L2 Cache + 4 Way-Set Associativity - 4 points
- Basic Hardware Prefetching - 4 points
- Eviction Write Buffer (maybe upgraded to victim buffer if time permits) - 4 points
- Parametrized Cache (# of ways and # of sets) - 3 points
- Progress Report & Roadmap for CP3