Team Pied Piper Checkpoint 1 Progress Report and Roadmap

Progress Report:

Who worked on each part of the design:
In terms of group work, each team member worked on each part of the design equally. We made sure to meet up and work on all parts of the checkpoint together, rather than splitting up the work, as we all wanted to ensure that we each had a strong fundamental understanding of how our design is implemented and functions. The parts that we worked on included coding the pipeline design, testing and verifying, writing the progress report and roadmap, and last but not least creating the design for data forwarding, hazard detection, and the arbiter.

Functionalities we implemented:
For checkpoint 1, we added all the functionalities required for a basic pipeline that can handle all of the RV32I instructions except for the FENCE*, ECALL, EBREAK, and CSRR instructions. We did so by splitting our code into “stages" and creating intermediate stage-registers that contain the data to be passed between the different stages, so that the pipeline can always run while being able to simultaneously fetch instruction and data, if required. For the memory interface functionality, since this checkpoint does not require the implementation of caches and the arbiter, we utilized the magic memory dual port module to handle simultaneous memory requests as well as the magic memory module to get the values in the same cycle.

Testing Strategy:
Regarding testing, we ran multiple tests. First, we ran the mp4-cp1.s testcode provided to us on our pipelined design as well as our working mp2 design. We compared the results from both of these designs and saw that they were the same, but that our pipeline design was significantly faster. We wrote some extra test cases to test specific instructions, such as branch, load, and store.

Updated Datapath after Finishing CP1 Pipeline Design:

Link:
https://drive.google.com/file/d/1gtmhXKNMq09-PqD5-fn9AaoWFuNZA2HO/view?usp=sharing

Roadmap:

Who is going to implement and verify each feature and functionality:
Regarding the functionality requirements for checkpoint 2, we see that we will have to implement hazard detection, forwarding, static branch prediction, and the arbiter and caches. We foresee each team member working on every feature implementation equally. As stated before, this is because we all want to ensure that we all are aware of how our design is being implemented and grasp a firm understanding of these new concepts. Splitting up the different functionalities might lead to a harder time in the future since only some team members may be aware of how a certain feature was implemented or works which can be troublesome for debugging.

What are those features or functionalities:
Features & Functionalities to be implemented for checkpoint 2:

- Hazard Detection
- Forwarding
- Static Branch Prediction (for all control hazards)
- Arbiter
- Instruction Cache
- Data Cache
- Proposal for Advanced Features