Progress Report

What did we implement this checkpoint?

This checkpoint we implemented a one cycle hit, split, 2-way L1 cache into our pipelined datapath design. We created an arbiter to control the signals from each half of the split cache when a miss occurs. We also modified our datapath to stall until instructions or data has been received from memory. Finally, we finished the functionality of the remaining RV32I load and store operations.

What did each team member contribute?

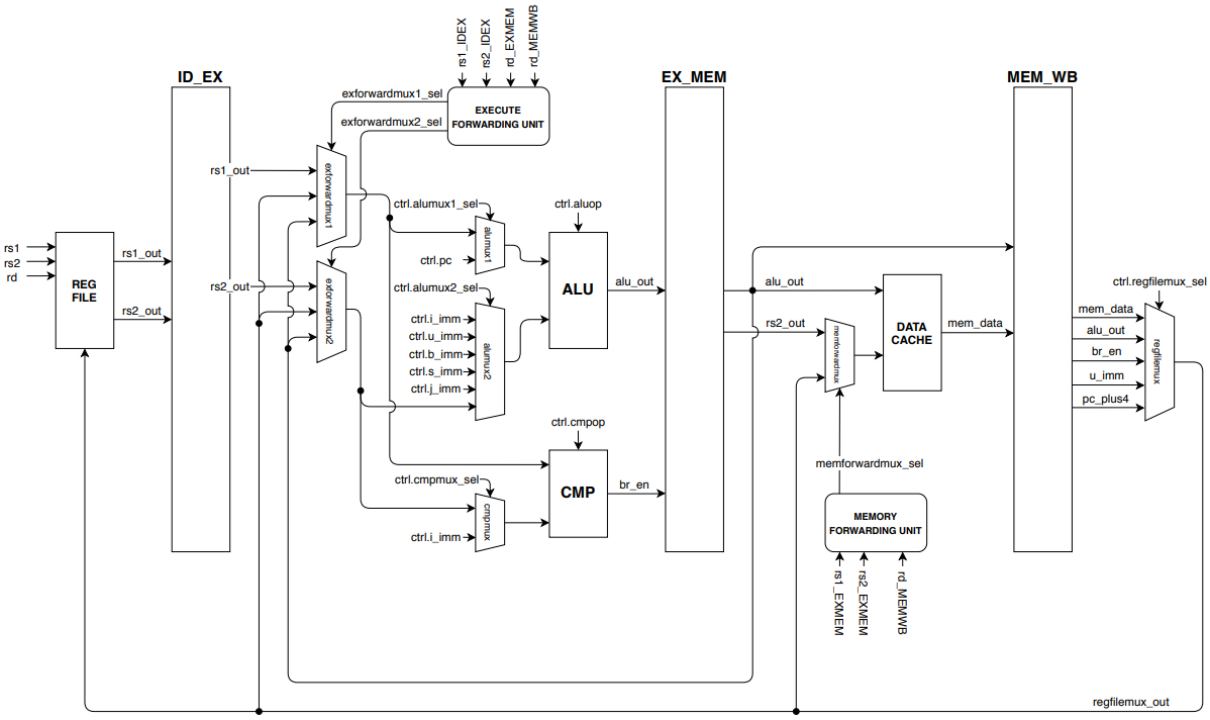
Arkin primarily focused on incorporating the split L1 cache, arbiter, and cacheline adaptor into our design. He went off his own MP2 cache design because we decided his was the best. Yohanes added the needed functionality for the remaining load and store operations, also using his MP2 design as reference. TJ edited the pipelined data path to be cleaner and have less superfluous signals related to the control word. He also added the logic to stall the pipeline when a cache miss occurs.

What major problems did we encounter and how did we solve them?

The first major hurdle for us was converting the MP2 cache design into a one cycle hit cache. We realized that the separate idle state was unnecessary, and we could treat the hit detect state as a pseudo idle state. We ran into additional problems regarding this later since we incorrectly adjusted the next state logic. Initially, all we knew was that the caches were sending response signals at incorrect times. Using Modelsim, we traced the signals from pmem and found no issues. We traced the signals from the cacheline adaptor and found no issues. We traced the signals from the arbiter and found the issue but determined that the arbiter wasn't the source. That left the cache control signals, where we discovered it would progress to the next state even if read and write were low. By adding additional if statements we fixed this issue. The rest of the major issues came near what we thought was the end at the time. Our code was compiling and giving no errors/warnings in Modelsim and our register values looked good throughout execution. However, the design wasn't passing the AG. Being that we lacked any error information in Modelsim, we needed to use the AG log and Piazza to debug. We found that we had overlooked that the AG sees every cycle where a cache response as high as a new instruction or new data. It was our faults for not fully reading the Piazza post, but it was still frustrating to realize the issue technically had nothing to do with our actual design. For the future, AG functionality like this should be in the documentation on GitHub not on a Piazza post.

Roadmap

For the next checkpoint, we will continue the plan structure we used last checkpoint since it worked well. Everyone will contribute to the code in some way, but one of us will focus less on CP3 implementation and more on CP4 design. They will also have the pleasure of doing the progress report and roadmap for the next TA meeting. We have the ability to be fluid and allocate more resources if for some reason we run into an issue that requires all of our attention. Yohanes will be the one thinking about CP4 while TJ and Arkin work on implementing CP3. More specifcally, TJ will implement forwarding and hazard detection as described in the diagram below. Arkin is our cache expert and will integrate the L2 cache.



Planned Advanced Features:

- Array Multiplier
- Basic Prefetching
- Branch Prediction (Will determine with the help of TA which type)
- Pseudo LRU