Anthony Nguyen - alnguyn2
Hemang Nehra - hnehra2

CP3: Progress Report

Functionalities

For this checkpoint, we implemented some of our proposed advanced features. The arbiter was modified to allow for prefetching the next instruction. We found that this was the easiest solution since the arbiter is able to control what data is sent from each cache and this allows an easier way to interface with shadow memory. We are also trying to implement a pipelined L1 cache. Ideally, we'd also implement a more advanced branch predictor if time allows. We've run into bugs that are causing us to fail the CP3 test code and debugging took longer than expected.

Contributions

Anthony worked on debugging the CPU to ensure that CP3 test code can be passed. He also worked on the basic prefetching implementation in the arbiter. Hemang worked on pipelining the L1 caches. Both worked on debugging and integrating everything into the CPU. Performance metrics were gathered from each person and their respective implementations. The overall performance and performance metrics were taken after each advanced feature was implemented together.

Testing Strategies

Since the advanced features require modifying different parts of the CPU, we tried to add things incrementally. When prefetching was implemented, we tested to see if it ran properly without the other advanced features in the CPU. This allows us to ensure that one advanced feature is working independently of the other features. We've also run some performance counters to compare our basic pipelined CPU when adding each advanced feature to see how each feature affects the performance.

Anthony Nguyen - alnguyn2
Hemang Nehra - hnehra2

Advanced Features

1. Basic Prefetch

We implemented a one block lookahead (OBL) prefetch. The arbiter was modified to fetch the next instruction from memory on a cache hit. The prefetcher was only implemented for the Icache because we believed that it would be more beneficial since instructions are more likely to be executed sequentially and used before eviction compared to the Dcache. We expected the results to be faster than the baseline, however, our implementation of the basic prefetching resulted in a slower performance.

Prefetching Performance with CP3 TestCode

Metrics:	Number of Occurrences
Mispredictions	3,751
Stalls	19,373
Branch Instructions	6,618
Icache Hits	251,124
Icache Misses	498,423
Dcache Hits	39,372
Dcache Misses	186,485
Execution time	7.5ms

Baseline Performance

Metrics:	Number of Occurrences
Mispredictions	12,903
Stalls	19,373
Branch Instructions	6,618
Icache Hits	198,603
Icache Misses	322,872
Dcache Hits	29,429
Dcache Misses	136,247
Execution time	5.2ms

Some values are not exact since there are times where signals that trigger these counters randomly spike during a clock edge.

Anthony Nguyen - alnguyn2
Hemang Nehra - hnehra2

2. Pipelined L1 Cache

We wanted to design a faster cache since the provided cache has poor performance. Initially, using our MP3 cache design would result in much better performance however, neither of us were able to get our designs working. The idea was to improve on the cache to make cache misses have less penalty however, we were unable to complete and debug the design in time.