CP3 Writeup

Progress Report

John: For this checkpoint John spent most of the time debugging why the checkpoint 3 code would not run using the cache/physical memory. He also implemented a backwards taken/forwards not taken branch predictor which is used in the tournament branch predictor which achieves a 80% accuracy on all the checkpoint codes. He also implemented tfac gating that saves on power and has verified via waveforms that the clock signal is not provided when applicable.

Gally: Gally also spent most of the time debugging why the checkpoint 3 code would not run using the cache/physical memory. He was also able to add the risc-v M extension with the basic multiplier and divider. He also implemented an advanced hardware prefetching mechanism based on strides.

.

Arjun: Arjun spent some time debugging why the checkpoint 3 code would not run using the cache/physical memory. He also focused on making an L2 cache which has a parameterized number of sets, 8 way fully set associative with a write eviction buffer. He verified via waveforms that data was being read correctly into the hierarchy as well as being evicted and written back correctly.

Roadmap

John: John will see if he can optimize our branch predictor and see if we can achieve better predict accuracy. He has already added counters to determine the accuracy rate of the current prediction scheme and will experiment and see if other schemes proved better results.

Gally: Gally plans on making the prefetching more efficient as well as seeing if he can make a more sophisticated divider than the current shift subtract implementation.

Arjun: Arjun will explore the different configurations of the L1 and L2 caches to see what configuration is optimal for running the competition code.

Collectively as a group we will work on the presentation and final report that summarizes our project. We also need to implement more counters that help measure performance and will have that complete for the final presentation and report.

Status of Optimizations

Risc-v M Extension - Has been implemented and that we have verified. It uses a combinational multiplier and modified subtract shift algorithm to divide.

L2 cache - We have implemented the hierarchical L2 cache system. It is an 8-way fully set associative cache where the number of sets is parameterized. This L2 cache also uses the alternate replacement policy LIFO which acts like a stack where the most recently used way is evicted. On top of that we implemented a write eviction buffer and can see that evicted data is first written to a buffer and the data request is first served and then when no requests are being served, the data is written back.
Hardware prefetching - Has been implemented**.** Depending on the state of the counter, a memory address is loaded into a register. Also depending on the state of the counter, we output the difference between the addresses known as the stride and the cpu fetches both of these addresses.

Eviction Cache Buffer - Has been implemented as described above with the L2 cache.

Clock Gating - We implemented clock gating so that when a register-register instruction is in the mem stage, we do not provide a clock signal to the d_cache/memory, therefore saving on power, however minimally.

Branch Predictor - We decided to implement a tournament branch predictor that chooses between other branch predictors we have already implemented. These branch predictors are: static not taken, backwards taken/forwards not taken, and local history table. This predictor uses the counter method to make its selection between the branch predictors.

Counters

The following are our counters for branch prediction and jump prediction on cp3 code:

total branch instruction = 7305
total branch mispredict = 3217
static not taken mispred = 3217
static backwards taken forwards not taken mispred = 1697
dynamic local branch history table mispred = 216
tournament mispred = 208
total jal instruction = 3074
total jal mispredict = 3074
total jalr instruction = 3079
total jalr mispredict = 3079

Performance Improvement

The following improvement is shown on the cp2 code:

Baseline - 5305000 ps
With Optimization - 4845000 ps