Class: ECE 411 Spring 2018
Assignment: MP3 CP4 Report
Due: April 15, 2018
Members: Harsh Modhera (hmodhe2), Ben Li (cli91), Forrest Hornitschek (hornits2)

Checkpoint 4: Progress

This checkpoint gave us three distinct tasks: implementing static branch prediction, an eviction write buffer, and a performance counter. For the branch prediction, we discovered this was a feature we had already included since we designed the original pipeline back in checkpoint 1. The processor uses the “not taken” approach, in which the fetch stage assumes, following a branch, that it will not be taken and continues to fetch the next instructions in sequence to fill the delay slots that come with waiting until the execute stage to determine the actual result of the branch. In the case the branch is taken, the preceding stage registers are flushed and pass through the pipeline as NOP instructions.

We decided on an eviction write buffer between the L2 cache and physical memory. The purpose of this buffer is to rearrange the order of the read and write in the write back process, so that the processor can retrieve its data in a single access to memory and continue its execution. Otherwise, it spends a memory access writing back the evicted data to memory, followed by another access for the read. Inserting an additional module between L2 and memory required modification of the wishbone signals that are sent to memory. Since multiple instances attempt to utilize physical memory, the datapath needs extra logic to select the address it should read/write from and when it should begin with a request. Because L2 waits for memory on a read and the EWB on a write, it needs to MUX the ack’s provided by both to know when to continue and service the arbiter’s request(s).

We added performance counters for the total branches, the number of mispredicts, and the number of hits and misses for each of the three caches. We created counters, which are modified registers, that resets on seeing an STI in the memory stage with an address that’s within a pre-defined range. On LDI, a mux that determines the data to pass to the write back stage is flipped to use the counter values instead of the output from the data cache. Lastly, an increment signal for each of the counters increases the value in them by one each time it is raised high on a positive edge clock. For this input, the logic to increment each counter differs accordingly.

Checkpoint 4: Bugs

While working on the write buffer, it was challenging to time the states appropriately with the L2’s FSM such that it could smoothly transition between reads and writes. With a buffer now taking care of writes, the processor is free to execute the next instructions while the write to memory occurs. So, it is now possible for execution to return to L2 in that time period and make a new request to memory. But once the EWB has completed its write, the STB continues to be

raised by the L2, meaning there is no one cycle wait that is required for making consecutive memory accesses. Our solution was to include an additional wait state in the write buffer that holds for one cycle after receiving an ACK from memory, so that it would ignore L2's request until it next becomes available.

Initially, the cache hit and miss counters updated two cycles past the point they hit/miss respectively. This could cause an outdated output on counter read requests. The was also a logic error in the reset signal to the counters that cost many of the counters to reset prematurely and unexpectedly. The increment signals also suffered from an edge case bug that would cause extraneous increments when the CPU was stalled.

Checkpoint 4: Testing

As mentioned earlier, we already thoroughly tested the static branch prediction with the not taken schema as this was implemented in checkpoint 1.

For testing the eviction write buffer, we used Harsh's mp2-ctest.asm file, which originally was written to test the cache lines and evictions. With this test code, we were able to realize some of the bugs mentioned above. More importantly though, this test file allow us to see the process of eviction of a cache miss in the L1 data cache, which evicted to the L2, which then evicted to the EWB. Seeing the entire process happen allowed us to verify the correctness of our EWB state machine as well as the functionality of the STR and LDR operations that caused an interaction with the EWB.

For performance counter testing, we first ran various previous test codes to ensure that our pipeline functions normally when not dealing with any counters, as it should. After that, we specifically wrote a small program that forces a predetermined number of branches, hits, misses, etc. We test flushing and reading the counter addresses to ensure that the counters work. After verifying that the counter values are what we expect, we augment normal program code with counter code and check the register values for proximity to what we'd expect.

Checkpoint 4: Member Contribution

For this checkpoint, static branch prediction was already done. Forrest and Harsh worked on implementing and getting the eviction write buffer to work.. Ben implemented and worked on getting the performance counters to work. Each task involved writing test code and using it to thoroughly debug the respective portion. We all met up after to go over all the code and merge the two portions into one functional mp3 checkpoint 4.

Checkpoint 5: Roadmap

For the upcoming checkpoint, we are tasked with implementing even more advanced functionality for our pipelined processor. The rubric requires a minimum of 20 points worth of

advanced functionality. We plan to aim for the following within the timeframe of checkpoint 5: implement a higher-associative L2 cache for 5 points, a victim cache for 8 points, a global 2 level branch history table for 7 points, and a branch target buffer for 5 points. After learning about the different schemes in lecture and doing hw2, we feel we have a good understanding of dynamic branch prediction and the branch target buffer. We hope that increasing the cache complexity will not be too difficult as we all have a good understanding of the topic from working with them throughout this semester. Lastly, during checkpoint 4, we originally tried to implement a victim cache without realizing it when trying to design the eviction write buffer. Hopefully, that head start will help in the coming checkpoint.

Checkpoint 5: Proposed Member Contribution

Like most of the previous checkpoints, we will meet up to figure out how we plan to spend the week and how we can break up/designate the work. Each task will involve writing test cases in order to thoroughly test and verify the advanced feature.

For now, Forrest will work on increasing the cache as his mp2 cache was used. Ben will work on dynamic branch prediction. Harsh will work on victim cache. If all these get done in time, we will all work on integrating the branch target buffer to work with the global branch history table.