Keystone (Group 6)
dlofran2, mwdanie2, gentil2
Kenny Umenthum
ECE 411 – MP3.2
Fall 2017

Progress Report

The goal for the second checkpoint was to finish the implementation of the remaining LC-3b instructions (i.e. JMP, JSR, JSRR, LDB, LDI, STB, STI, RET, SHF and TRAP) and implement a fully functioning arbiter connected to two L1 caches (I-Cache and D-Cache, see paper diagram). Although, for the purpose of checkpoint 2, the arbiter may be connected to physical memory instead of the L2 cache, we tried to start the implementation of the L2 4-way set-associative cache. Although parts of this checkpoint went relatively well, implementing and integrating the L1 caches and some instructions were a bit of a challenge.

While JMP, JSR/JSRR, RET, SHF, TRAP, LDB and LDI were not too big of a trouble to create and get to work, both STORE instructions gave us more work. In fact, when performing an STB operation, we noticed that although the high byte of a word could be written to either the lower or the upper 8 bits of the word out, it was not the case for the lower byte. Indeed, it came to our attention that two hexadecimal digits were off while the rest of the word was right. The issue was that it was set such that the lower byte is always written to the lower part of a word, consequently, when we had an odd address, our implementation would not behave as expected. Besides the problem with STB, we encountered an issue with the behavior of STI. It appeared to be a problem with the state machine and some signals that had to be explicitly reset: we had to inhibit mem_write_out for the first state of STI in order for the operation to be executed properly.

Completing the pipelined process was also a bit of a challenge when debugging. When we were unit testing, everything worked properly, that is the pipeline and the L1 caches. However, when integrating everything together and testing the cache code in the pipelined processor, we would get half-cycle mem_resp on a hit, which, ultimately, would cause the stall logic to not work correctly.

Another issue we encountered was with missing some branches. In fact, br_enable would go high while stalling, and thus would not be processed when the stall stopped. To fix that we had to hard-code a mux selector to 1'b1 when both br_enable and imem_stall go high. The same type of problem happened for JSR and was fixed in a pretty similar way.

Once more, we noticed a comparable problem with the operation TRAP. The issue was more about the way we implement some stages. We moved TRAPZEXT from IF stage in order to pass it in the MAR MUX in the MEM stage.

Finally, the arbiter did not cause us too much trouble as only two signals had to be modified to fix it from breaking after 200ns.

Once again, we tried to split the work between the three members, but debugged together. Dominic and Matt modified different files at once during the debugging phase, all three of us updated the paper diagram as we noticed additions and modifications in the implementation, while I took notes while debugging in order to maintain a bug log file and we are planning on keeping the same pattern for the next checkpoints.