Nathan Beauchamp, Rishi Thakkar, Melissa Jin
Team name: Silver Pipelining
ECE 411 MP3, Checkpoint 1

Progress Report:

For Checkpoint 1, our group implemented a basic pipelined processor supporting six instructions in the LC-3b ISA (ADD, AND, NOT, LDR, STR, and BR). To do this, our first step was to modify our datapath paper design according to feedback from our mentor TA. Next, we implemented this datapath design in SystemVerilog, creating the appropriate registers and combinational logic for each pipeline stage. Third, we designed the control ROM by tracing the execution of instructions through the datapath and determining what the control signal values needed to be at each stage. In SystemVerilog, our control ROM module was implemented using a simple case statement on the input opcode. Finally, we created a top-level module to interface with our testbench and connect our control ROM and datapath instances.

So far, work has been divided fairly evenly between the members of our group- Rishi and Melissa implemented the datapath, while Nathan implemented the control ROM. The three of us worked together to integrate the components and debug the design using the provided test code. Throughout the entire process, we communicated with one another to establish common naming conventions and SystemVerilog coding structures. This minimized the time required to successfully integrate the separate components into a cohesive design. Following a systematic design approach also enabled us to reduce our debugging time.

Roadmap:

In future checkpoints, we will continue to add functionality to our basic pipelined processor design. Currently, our design does not account for control or data hazards- i.e., instructions whose operands are determined as a result of a previous instruction that has not yet left the pipeline. We will need to design and implement a hazard detection method that can account for these cases. Furthermore, our design currently uses a dual-port magic memory, which is impossible to implement for a real processor. In future checkpoints, we will implement a two-level cache with separate data and instruction storage. This is necessary to minimize pipeline stalls upon a cache hit caused by a load or store instruction- without a separate I-cache and D-cache, we wouldn't be able to load the next instruction and a data value in the same clock cycle. The two L1 caches will interface with the L2 cache by means of an arbiter, which will decide which L1 cache should interface with the L2 in a given cycle. This L2 cache will then interface with a more realistic physical memory similar to the one used in MP2. Finally, we will need to implement the remaining instructions in the LC-3b instruction set.

Specifically for Checkpoint 2, we will implement the remaining LC-3b instructions, the split L1 caches, and the cache arbiter. Depending on time constraints, we may implement the L2 cache for this checkpoint as well. If we so do, the cache arbiter will interface with this; else, it will simply connect to the physical memory.