Progress Report

1. What did you group implement for this checkpoint? What major problems did your team encounter and how your group solved them?

We implemented a partial RV32I pipelined datapath for this checkpoint. It handles all the required instructions, without taking into account potential hazards that might occur. The only memory interface it has an interaction with is the given magic memory model. It is not a cache, just a highly idealized memory model.

A few major challenges that our team encountered was definitely the fact that we are not able to meet up physically, due to pandemic situations — this took from us the required interactions to teach and learn from each other and also come up with better plans to carry out the project. Other challenges included figuring out what the stage registers actually do, because initially we could not grasp the idea of just passing a control word from one stage to the next and then disassemble that word into the various signals so as to use them in the stage itself. Furthermore, we also struggled with documentation about connecting our datapath to the magic memory because we were not entirely sure how memory interfaces worked in the given files. We also struggled with the correctness of data as well as we compared our code to the results from the auto-grader. We solved most of these issues either by discussing on our group chat or by extensive reading of various sources on the internet. When all else failed, we also resorted to trial and error by going through signals in ModelSim. ModelSim also gave us a lot of trouble as we came across errors that waves could not be added to the simulation. Furthermore, we could not figure out how to make the system halt — we finally figured out all of these after talking with our mentor TA and also going through trial and error. We think that more documentation should be provided about top.sv in the future, to avoid these random problems.

2. A Roadmap that list your team's detailed schedule for the upcoming two weeks.

For the next two weeks, we intend to work more towards the cache while also analyzing the current datapath for potential hazards considering that it will be the crux of our problems coming up. While the cache design itself will be difficult, we want to have one of our teammates just scrutinizing the current design for potential hazards and ways to edit the pipeline design to implement forwarding and resolve said hazards. We feel like we have come up with a strategy of having two team members working on the current checkpoint and writing code and debugging, while the third looks towards the future and the next checkpoint, making a paper design and even the roadmap so that the team always has a direction to go towards and is never stranded without a plan. The third teammate also works on the reports and paper designs since they are almost worth the same amount of points as the implementation itself.

3. A cache and arbiter design containing datapath drawing, signal table (Name, width, connection, short description for usage), state machine and its description (if your design contains state machine). (State actions, transition conditions, etc.)

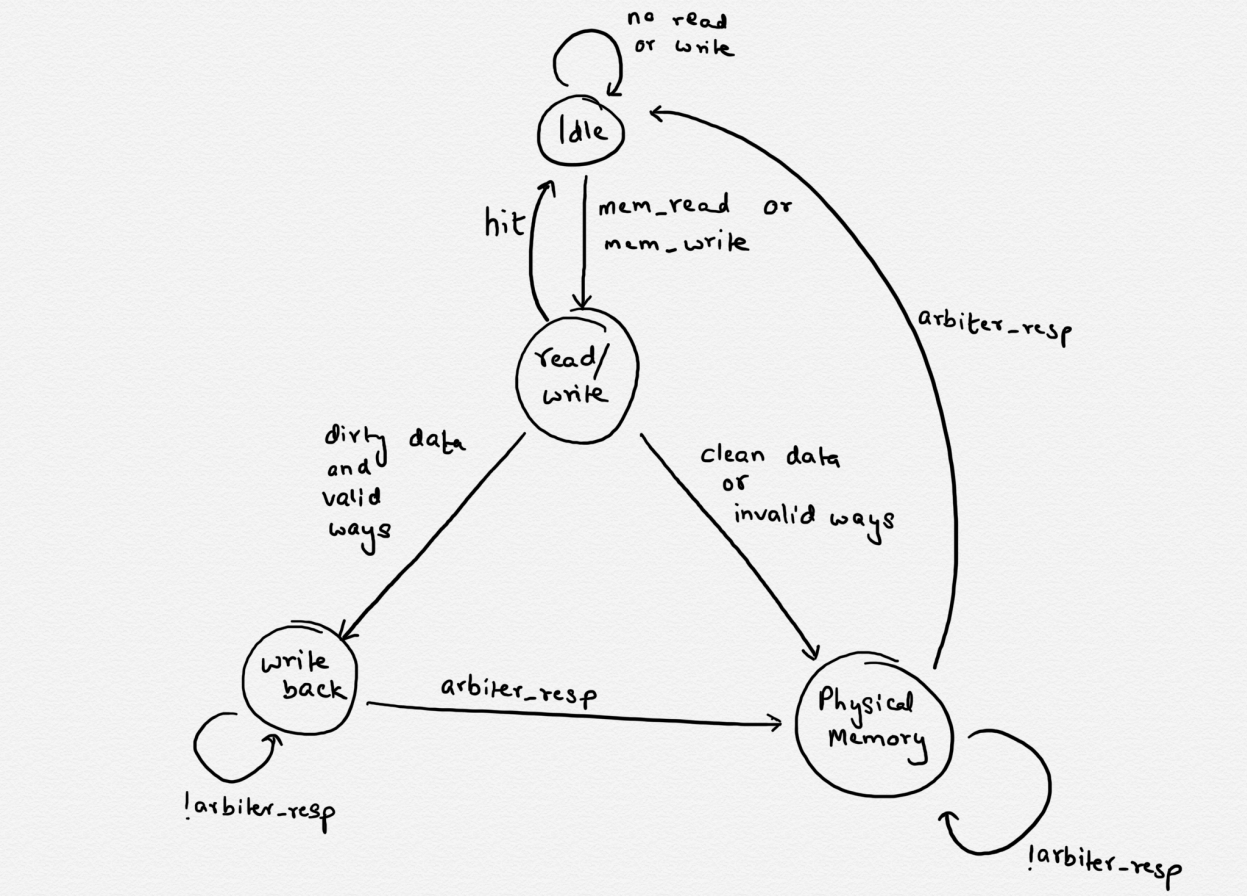

Figure 1: Cache State Machine

```
read_i =     1'b0; // pmem read
write_i =    1'b0; // pmem write

mem_resp =      1'b0;

load_tag_0 =    1'b0;
read_tag_0 =    1'b1; // default read
load_valid_0 =  1'b0;
read_valid_0 =  1'b1; // default read
load_dirty_0 =  1'b0;
read_dirty_0 =  1'b1; // default read
read_data_0 =   1'b1; // default read
load_data_0 =   1'b0;

load_tag_1 =    1'b0;
read_tag_1 =    1'b1; // default read
load_valid_1 =  1'b0;
read_valid_1 =  1'b1; // default read
load_dirty_1 =  1'b0;
read_dirty_1 =  1'b1; // default read
read_data_1 =   1'b1; // default read
load_data_1 =   1'b0;

valid_in =      1'b0;
dirty_in =      1'b0;
way_select =    1'b0;
load_LRU =      1'b0;
read_LRU =      1'b1; // default read
lru_in =        1'b0; // maybe, could be hit0??
pmem_mux_sel =  2'b00;// default to {mem_address[31:5] , 5'd0}
data_select =   1'b0;
```

Figure 2: Default signal values in Cache Idle State

I-Cache & D-Cache Specs:
2-way
8 sets
32 bits/set for quick access

Arbiter

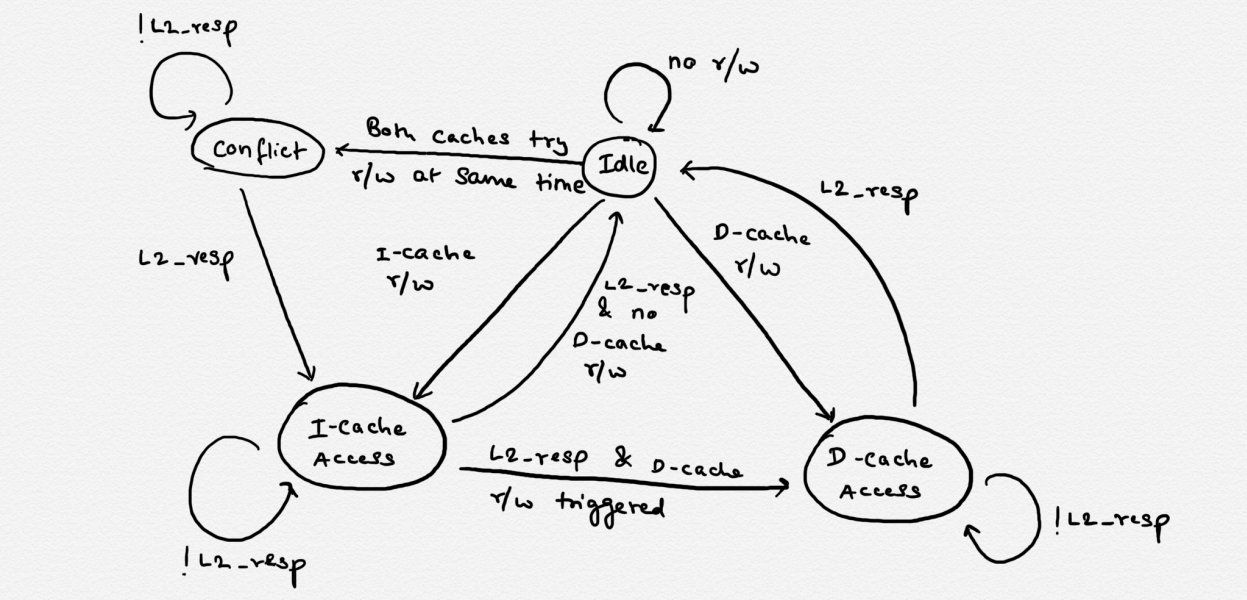

Figure 3: Arbiter State Diagram

Signal Name	Value in Idle/Conflict/ I-Cache	D-Cache Value
read_mux_sel	1'b0	1'b1
write_mux_sel	1'b0	1'b1
memaddr_mux_sel	1'b0	1'b1
memwdata_mux_sel	1'b0	1'b1
memrdata_mux_sel	1'b0	1'b1
memresp_mux_sel	1'b0	1'b1
mbe_mux_sel	1'b0	1'b1

Explanation: Value 1'b0 indicates choosing I-Cache, it is given priority. While the value 1'b1 indicates allowing the D-Cache access to the Arbiter and hence L2-Cache. The arbiter datapath will consist of only muxes that choose data from the data caches and allow each of them access. As the state diagram and the default value "1'b0" shows, we have chosen to give the I-Cache priority.

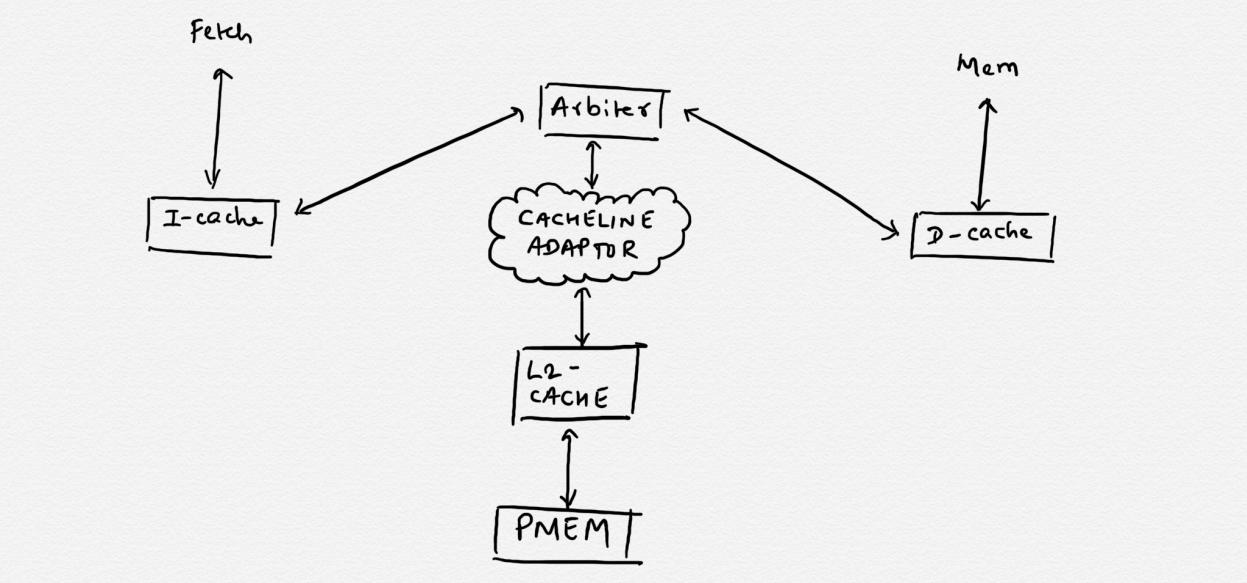

Figure 4: Cache and Memory Model Hierarchy