Edwin Louie, Ryan Metzger, Sohil Pokharna
“Three Don’t Cares”

ECE 411 CP 1 Updates

What we have done

This week we implemented the basic pipelined datapath which includes the following and the division of tasks:
FETCH – Edwin/Sohil
DECODE- Edwin/Sohil
EXECUTE- Edwin/Sohil
MEMORY- Edwin/Sohil
WRITEBACK- Edwin/Sohil
Control Word Generation -Ryan

We followed our datapath drawing from the design in the design checkpoint with the additions and suggestions from our TA.

Testing:
After fixing all syntax and complier errors, we tested the datapath through writing our own testcode and systematically comparing the provided mp4_cp1.s code with the CPU we developed in MP2. We ran out of time, but would further add in the randomizer to create random instructions and check with the monitor to make sure we are getting the values we intend to get.

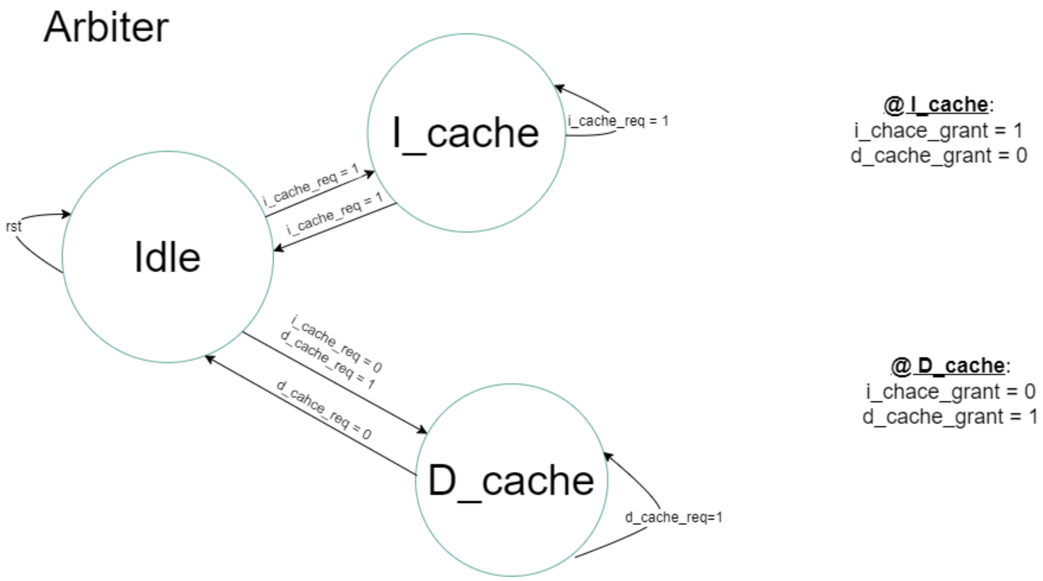
Plans for next Checkpoint

Note: These assignments are subject to change also if you design a unit you are also responsible for creating the verification code in general.

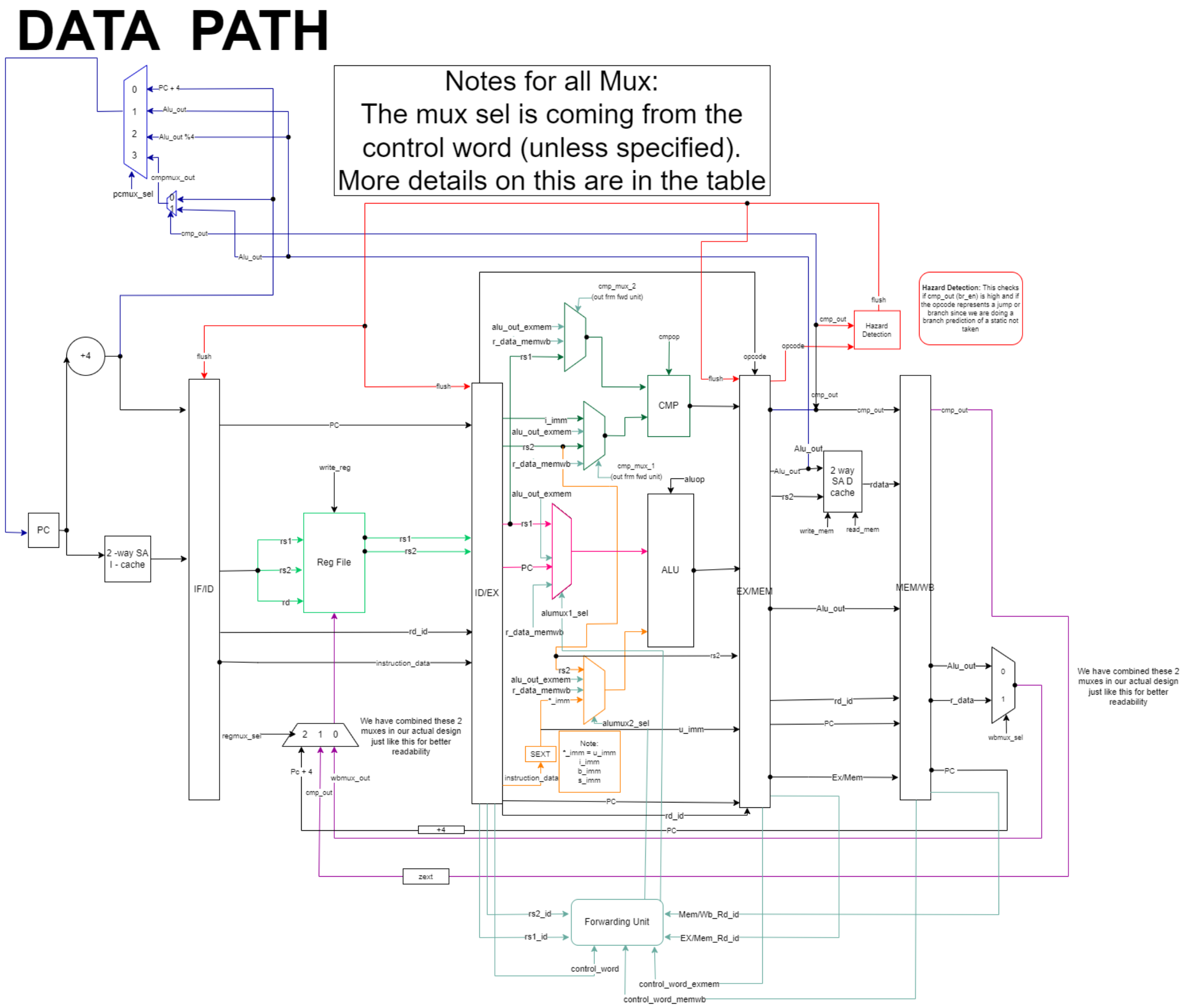
Arbiter - Edwin
Forwarding -Sohil, Ryan
Stalling – Edwin, Sohil
Hazard Detection -Sohil
I-Cache – Ryan, Sohil
D-Cache – Ryan, Sohil

We will simply insert the I-cache and D-cache into our datapaths by replacing where the magic memory was located. Following the insertion, we will add the arbiter such that we can grant each cache the ability to read/write from the main memory. We will also have forwarding unit and hazard detection units based on the design we have drawn below. We currently are thinking about stalling, though based on our implementation of a static not-taken branch predictor, we may not need stalling at all, but rather flushing of the instructions after the branch from the other pipeline registers/stages.

Arbiter State Machine



Updated Datapath with forwarding



Notes for all Mux:
The mux sel is coming from the
control word (unless specified).
More details on this are in the table

Hazard Detection: This checks if cmp_out (br_en) is high and if the opcode represents a jump or branch since we are doing a branch prediction of a static not taken

We have combined these 2 muxes in our actual design just like this for better readability

We have combined these 2 muxes in our actual design just like this for better readability

Fetch Unit

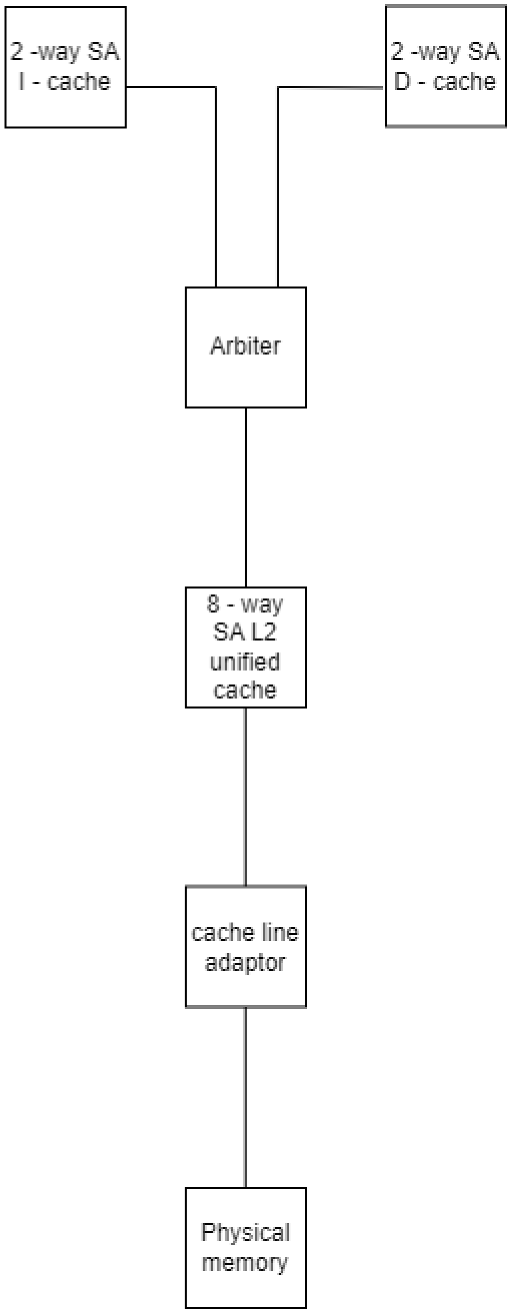
- EX hazard
  - if (EX/MEM.RegWrite
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 10
  - if (EX/MEM.RegWrite
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 10

- MEM hazard

  -
    ```
    if (MEM/WB.RegWrite
      and not (EX/MEM.RegWrite
            and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
      and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
     ForwardA = 01
    ```

  -
    ```
    if (MEM/WB.RegWrite
      and not (EX/MEM.RegWrite
            and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
      and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
     ForwardB = 01
    ```

Memory Block unit



Plans for Advanced Designs:

- 8-way L2 set associative cache
- Eviction Write Buffer
- Tournament Branch Predictor
- Return Address Stack
- 4-way set associative BTB
- RISC-V M Extension