Progress Report (CP1):

For checkpoint 1, we all split up the work by working on different units in the pipeline and then connecting them together. We've decided to be working on Microsoft Visual Studio Code Live Share, which lets us all work on the same code at the same time when someone is hosting the session. This allows for us to see what the signals are called in each module and to add/modify signals in our struct. In this checkpoint, we looked through our design checkpoint and created the necessary registers in each pipe. We did this by creating each register for a particular stage, and then passing the outputs into the next stage as necessary. We split this up by each taking a particular stage and writing out the registers for that stage. The four pipeline stages' (ifid, idex, exmem, memwb) registers were created, and then we created the registers in each stage of the pipeline (if, id, ex, mem, wb). We made sure to check each other's work, and also did some individual work by creating the units inside each stage. Yash worked on creating the comparator, ALU, and Control ROM logic, Vishal worked on the JMP logic and the writing to cache logic, while Aayush created some of the internal registers we needed and connected the pieces we all made. The functionality that we have completed at this point are all the riscv32 instructions that were required. We created the skeleton structure for our pipeline and have organized the stages in a way that we can trace back when necessary. In terms of testing and debugging, we used our datapath as a way to see the overall structure, and also referred to our MP2 for the different units as well as what signals we needed to turn on when. We wrote small test programs to unit test the instructions and used ModelSim in order to see if the registers were loaded properly and if the signals were turned high at the correct times. We also used the given test code to make sure each instruction was executed properly, and thankfully we were able to work it out quickly because we checked our design carefully after each unit was made. In terms of the design aspect of this checkpoint, we used diagrams.net so we could all make the designs together. We discussed the stalling and forwarding paths and then wrote out the logic so that we could create the FSM for the Arbiter. We also worked on the Memory Hierarchy by having one person draw the designs (on a screen share) while the other two verify the signals.

Roadmap for CP2:

For this next checkpoint, we need to be able implement hazard detection, forwarding (static-not-taken branch prediction), as well as an arbiter that integrates into our I-Cache and D-Cache. We also have to redesign our cache from MP3 so that it is a one cycle hit. We will work on this checkpoint in the same way we did for checkpoint 1 (VS Code Live Share). Yash and Aayush will work on the hazard detection and forwarding while Vishal works on the Arbiter and the connection to the I-Cache and D-cache. Once again, we will verify each other's designs and because we're coding together in real-time, we can all help each other out and debug together. Working this way also lets us learn all parts of the design, giving us a much better holistic view of the processor. In terms of the advanced features, we already have some in mind that we would like to implement and will keep in our minds while designing this part. We also will work together in changing our MP3 Cache to a 1-cycle hit so better timing is achieved.