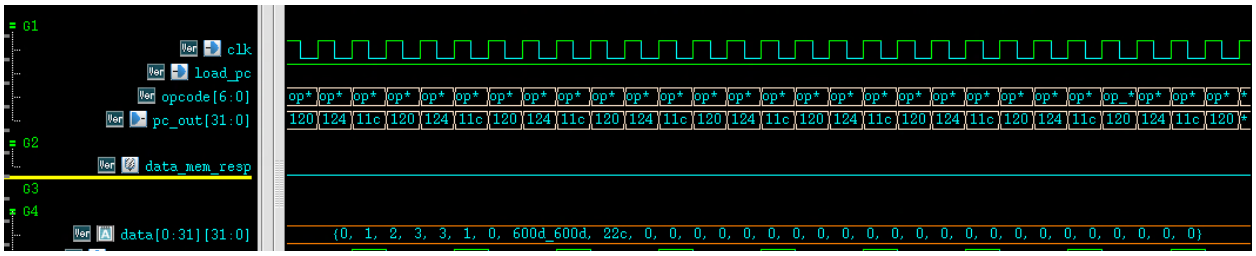
Progress Report:

For checkpoint 2, we had to have hazard detection, forwarding, including static-not-taken branch prediction for all control hazards, and an arbiter implemented and integrated, such that both split caches (I-Cache and D-Cache) connect to the arbiter, which interfaces with memory.

Below is the work distribution:

- Kelsey: Hazard detection & forwarding
- Murat: Arbiter, L1 Caches
- Xinpei: Static branch predictor

Testing Strategy: We received the correct resulting register values from our mentor TA and compared them to our results from the provided test code. Here are the results:



```
opcode[6:0]    op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op* op_* op* op* op* *
pc_out[31:0]   120 124 11c 120 124 11c 120 124 11c 120 124 11c 120 124 11c 120 124 11c 120 124 11c 120 124 11c 120 124 11c 120 *
data[0:31][31:0]  {0, 1, 2, 3, 3, 1, 0, 600d_600d, 22c, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

We also compared the flags of forwarding and changes in PC values to the test code. These comparisons allowed us to test the branch predictor and forwarding individually.

Time and Energy:

No information was provided about time and energy analysis.

Roadmap:

Checkpoint 3 task is advanced design. Our advanced design will include the following:

- L2+ cache system
- L-TAGE branch predictor
- Advanced hardware prefetching
- RISC-V M Extension

Here are the projected CP3 assignments:

- Kelsey: Advanced hardware prefetching
- Murat: L2+ cache system, RISC-V M Extension
- Xinpei: L-TAGE branch predictor

Advanced Feature Proposal and Designs:

L2+ cache system:

As a group, we decided that adding an advanced cache would be harder to test since an issue in it would break the whole pipeline. So, we decided to implement simple L2+ caches. These caches would take more than one cycle to hit and speed up our memory accesses. We will first implement an L2 cache and add L3 if we have time after completing all the other features.

L-TAGE branch predictor:

Taking references from Seznec (https://www.irisa.fr/caps/people/seznec/L-TAGE.pdf), we will implement an advanced L-TAGE branch predictor. Here is a description of it:

“TAGE relies on several predictor tables indexed through independent functions of the global branch/path history and the branch address. The TAGE predictor uses (partially) tagged components as the PPM-like predictor. It relies on (partial) match as the prediction computation function. TAGE also uses GEometric history length… This allows to efficiently capture correlation on recent branch outcomes as well as on very old branches.” (Seznec, André. “A 256 Kbits L-TAGE branch predictor.” .)

Advanced hardware prefetching:

We will attempt to implement PC based strided prefetching based on the source provided to us in the MP4 ReadMe (Baer and Chen, "An effective on-chip preloading scheme to reduce data access penalty," SC 1991.).

If we do not manage to implement it, we will use the One block lookahead (OBL) prefetch as a backup.

RISC-V M Extension:

We will add a multiplier and a divider to our design and extend our instructions to include them based on the “M” Standard Extension (The RISC-V Instruction Set Manual, p.47). We will attempt to design a Dadda multiplier, as it is faster and requires fewer gates than Wallace trees. We will determine the exact details of our hardware design during the checkpoint.