We implemented Tomasulo's Algorithm, including the memory operations, control operations, and we integrated our instruction and data caches. Our processor is now a fully working implementation of Tomasulo's Algorithm, and is capable of running coremark.

Progress report:

Contributions:
Yuqi: Cache integration and fixing bugs.
Yunxuan: Memory instructions and load/store buffer.
Nathan: Control instructions/buffer and progress report.

Functionalities:
Capable of holding instructions in the queue until reservation stations are ready.
Capable of holding instructions in reservation stations until they are ready.
Capable of doing alu operations and multiplication instructions.
Can send several instructions to the ROB out of order and the ROB will commit them in order.
Capable of doing branch and jump operations and flushing the mispredicted instructions.
Capable of using an instruction and data cache.
Capable of doing load and store functions using a load buffer.

Testing strategies: Verdi and assembly test cases. We made assembly test cases to test the branch and jump functions separately to ensure they were jumping to the correct pc value and were flushing mispredicted values. We also used assembly test cases to test the load and store instructions, and compared that with what the spike log should be from mp_pipeline. We ran coremark on our processor once we had finished every part of cp3.

Timing and energy analysis:
- data required time: 1.916443
- data arrival time: 1.916139
- fmax: 521.8 MHz

Roadmap:

Features and functionalities:

Branch predictor: need some form of branch predicting other than static-not-taken.

Superscalar: runs multiple instructions concurrently (up to 8).

Early branch recovery: save clock cycles by resolving a branch earlier.

Future contributions: We are unsure of who will actually end up doing which advanced features, but our current plan is: superscalar - (Yuqi), early branch recovery - (Yunxuan), branch predictor - (Nathan).