

# MP3

**Team: Cache Money Records (Young Moolah Baby)**

## Checkpoint 1 Report

### **Progress**

This is the first checkpoint of MP3, so it's relatively straightforward. We used two wishbone modules to connect between our CPU datapath (Fetch and Memory stage respectively) and magic memory. For datapath, it's similar to MP1 design except that we added pipeline registers after each stage for pipelining purpose. We created the components that needed for all instructions except for STI/LDI. Another change is that instead of having a state machine to control the control signals in each state, we used control words for different opcodes. In this checkpoint, we didn't consider data/control hazards situations.

Functionalities implemented and tested: ADD, AND, NOT, STR, LDR, BR.

We wrote test programs to test each instruction by itself, then we tested them together using the provided cp1 test code.

### **Contributions**

We all coded the checkpoint together and debugged together

### **Roadmap**

Ryan: Implement all the remaining instructions

Subhash: Cache design/data forwarding paper design

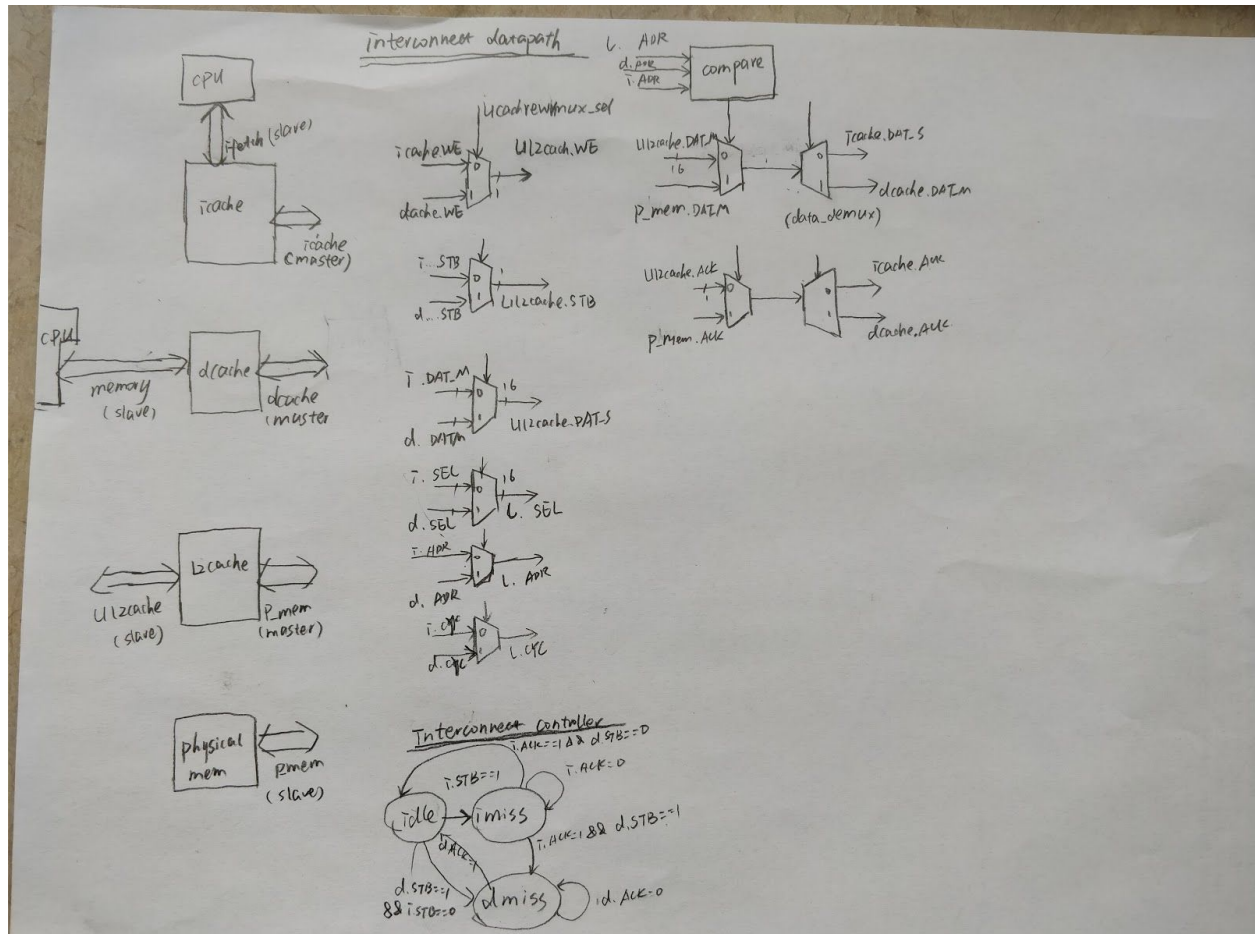
Chaohua: Cache interconnect

## Cache design

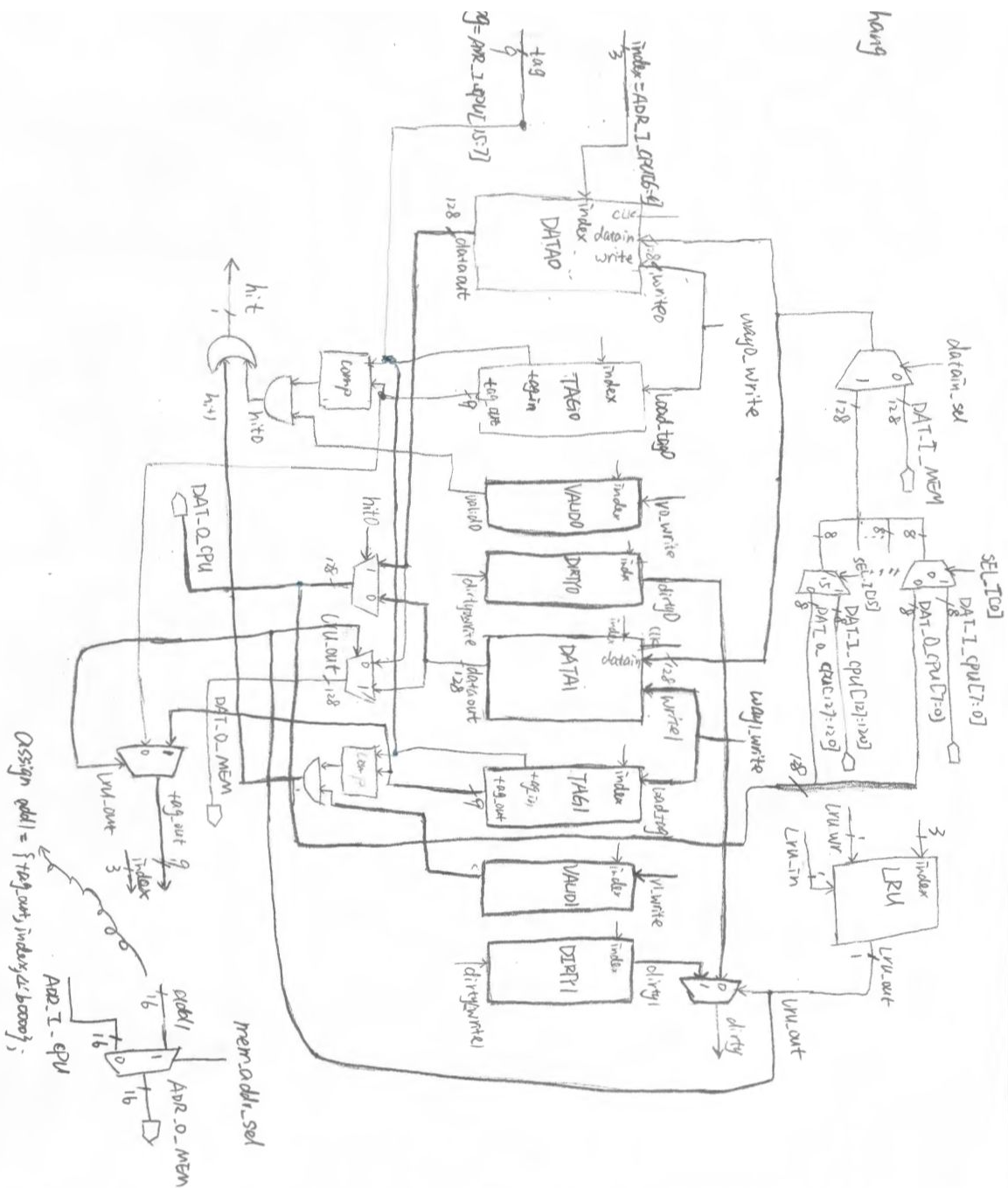
ICache and DCache have wishbone master port and they both need to connect to L2 cache, sharing one slave port. So we use mux/demux to select data coming from or send to icache and dcache according to control signals.

If it's an instruction miss (icache read/write), L2 communicates with icache, L1L2 cache wishbone connecting to icache wishbone. If it's a data miss (dcache read/write), L1L2 cache wishbone connects to dcache. If address coming from icache and dcache is same with L1L2 cache's address, then physical memory also communicates with icache and dcache which means if it's a icache/dcache miss and also L2 cache miss, icache/dcache can get data from physical memory directly.

Below is the paper design for interconnect and cache. For icache, dcache and L2 cache, We used the same cache design from mp2.



having



## **Checkpoint 2 Report**

### **Progress:**

For this checkpoint, we implemented all the remaining LC-3b instructions left from last checkpoint. We also implemented wishbone interconnect, connecting split L1 cache, L2 cache and physical memory. Compared with L1 cache, we increased the number of sets for L2 cache. We tested our processor with cp2 a/b test code and passed test-o-matic check.

### **Contribution:**

Ryan: Implemented all the remaining instructions

Subhash: Cache design/data forwarding paper design

Chaohua: Cache interconnect

We coded and debugged together.

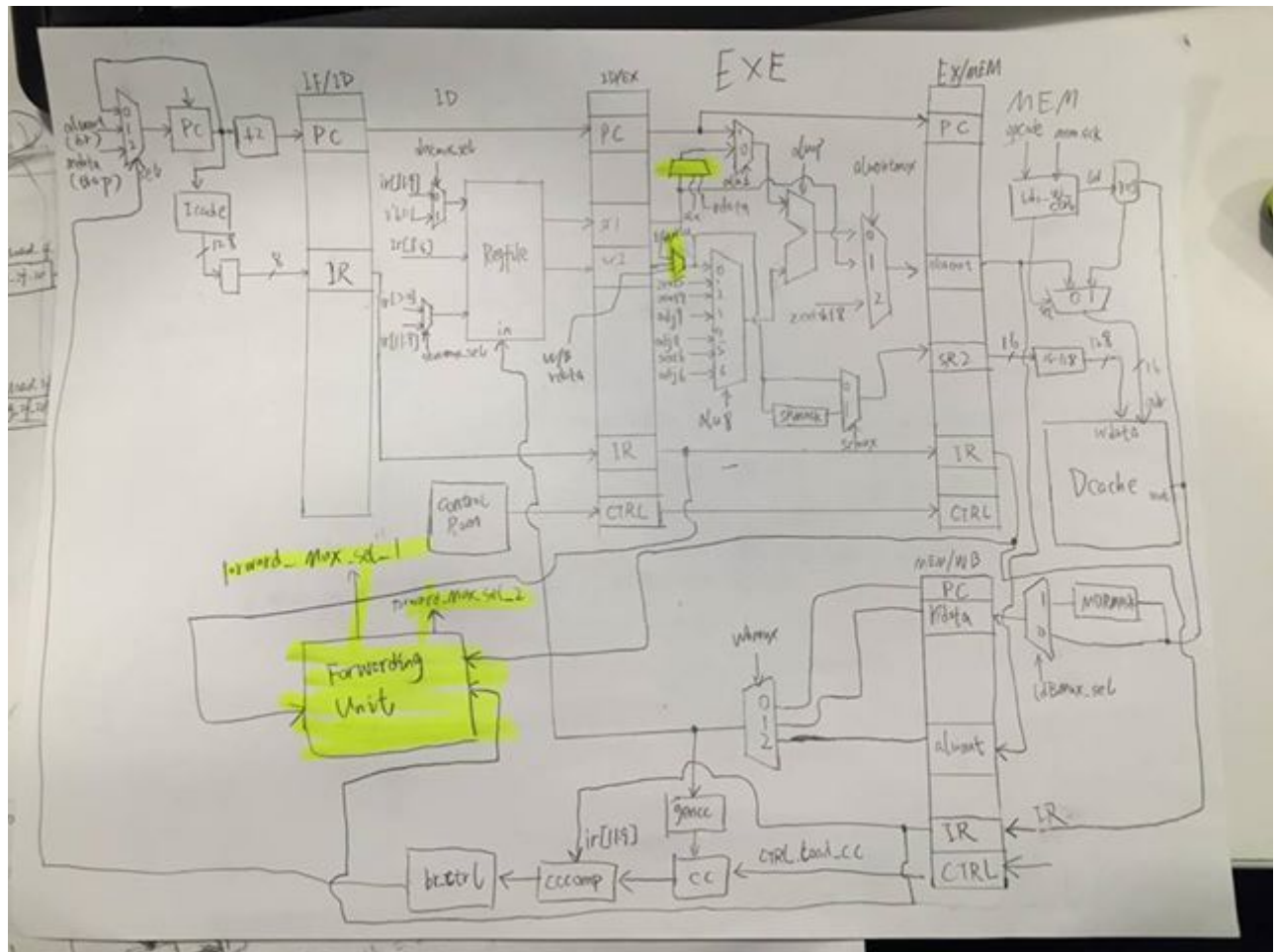
### **Roadmap:**

Ryan: Hazard detection and data forwarding

Subhash: Branch prediction, eviction write buffer, performance counters design

Chaohua: Improve cache and finish unified L2 cache.

### **Data forwarding design:**



## **Checkpoint 3 Report**

### **Progress:**

For this checkpoint, we detected the situation that has data and control hazard and implemented data forwarding. For data forwarding, what we did was we have a control logic that takes in instructions from various pipeline registers and detects the dependencies between those instructions and forward data accordingly. Then we found out there's a potential data hazard situation where a register value is used before it loads the value from the previous instruction. Our solution to this is to insert control bubble after load instruction. We also implemented control hazard detection and static branch prediction.

### **Contribution:**

Chaohua: Data forwarding, static branch prediction

Ryan: static branch prediction

We did paper design together.

### **Roadmap:**

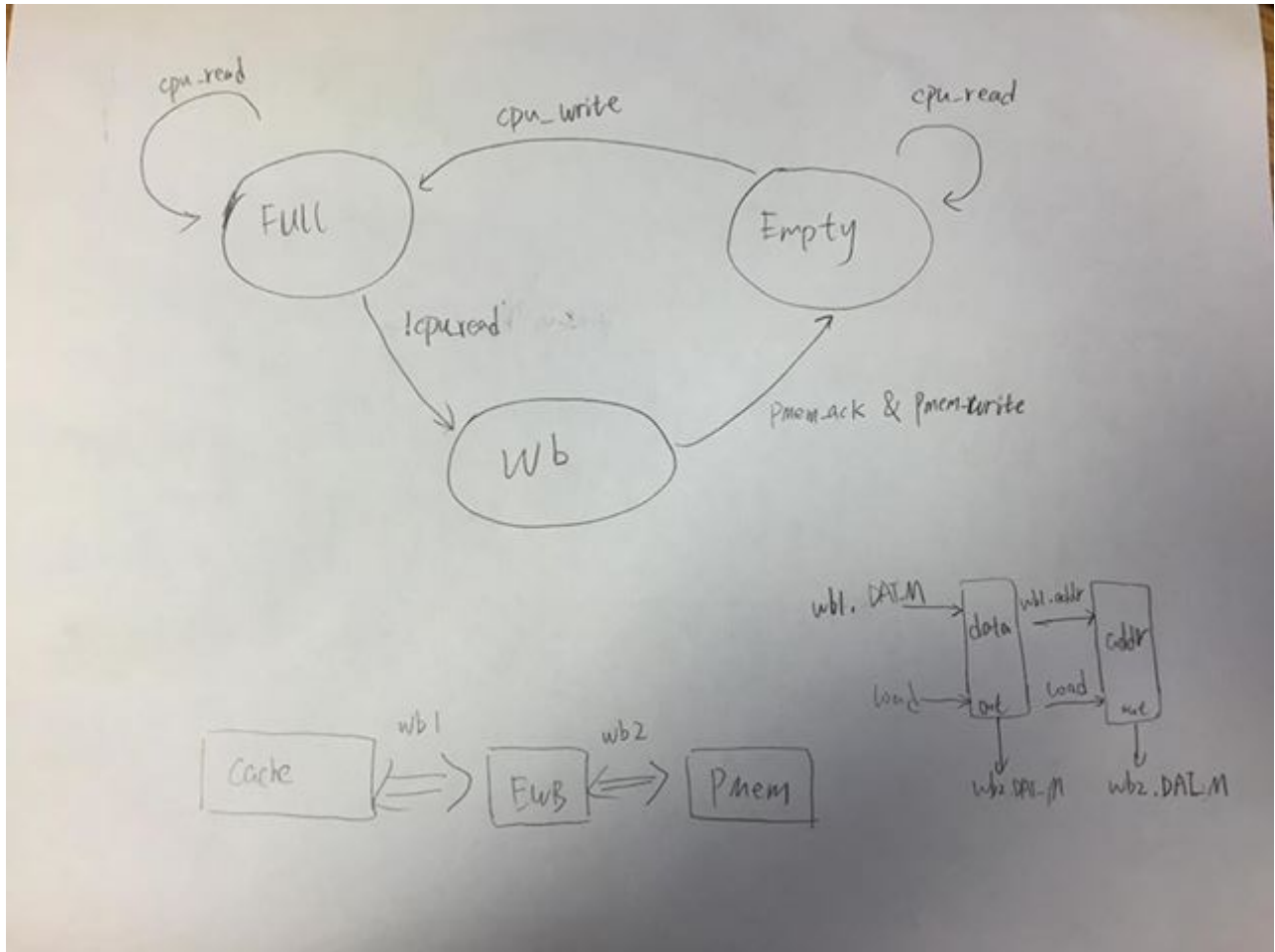
Chaohua: Static branch prediction

Subhash: Performance counter

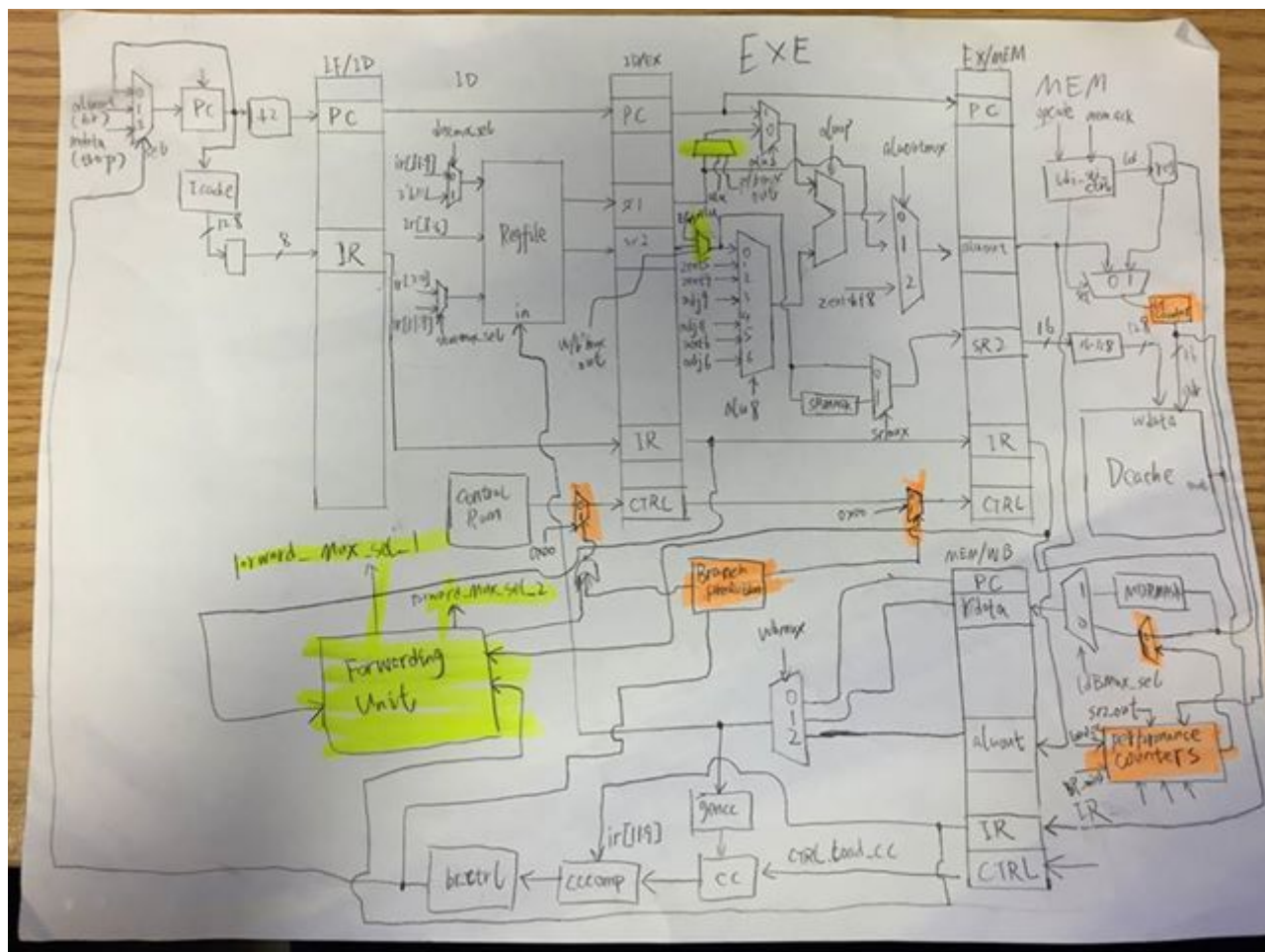
Ryan: Write Eviction Buffer

### **Paper design:**

Eviction write buffer:



Performance counter and static branch prediction:





## Checkpoint 4 Report

### Progress:

For this checkpoint, We checked the control hazard detection and static branch prediction implemented last checkpoint. We also finished the one-entry evict write buffer between L2 cache and physical memory.

Program Counter:

### Contribution:

Chaohua: evict write buffer

Ryan: evict write buffer, performance counters

We debugged together.

### Roadmap:

Chaohua:

- 4-way set-associative or higher L2 cache with LRU Policy (Pseudo or True)

- Global 2-level branch history table

Subhash: Victim cache

Ryan: Branch target buffer