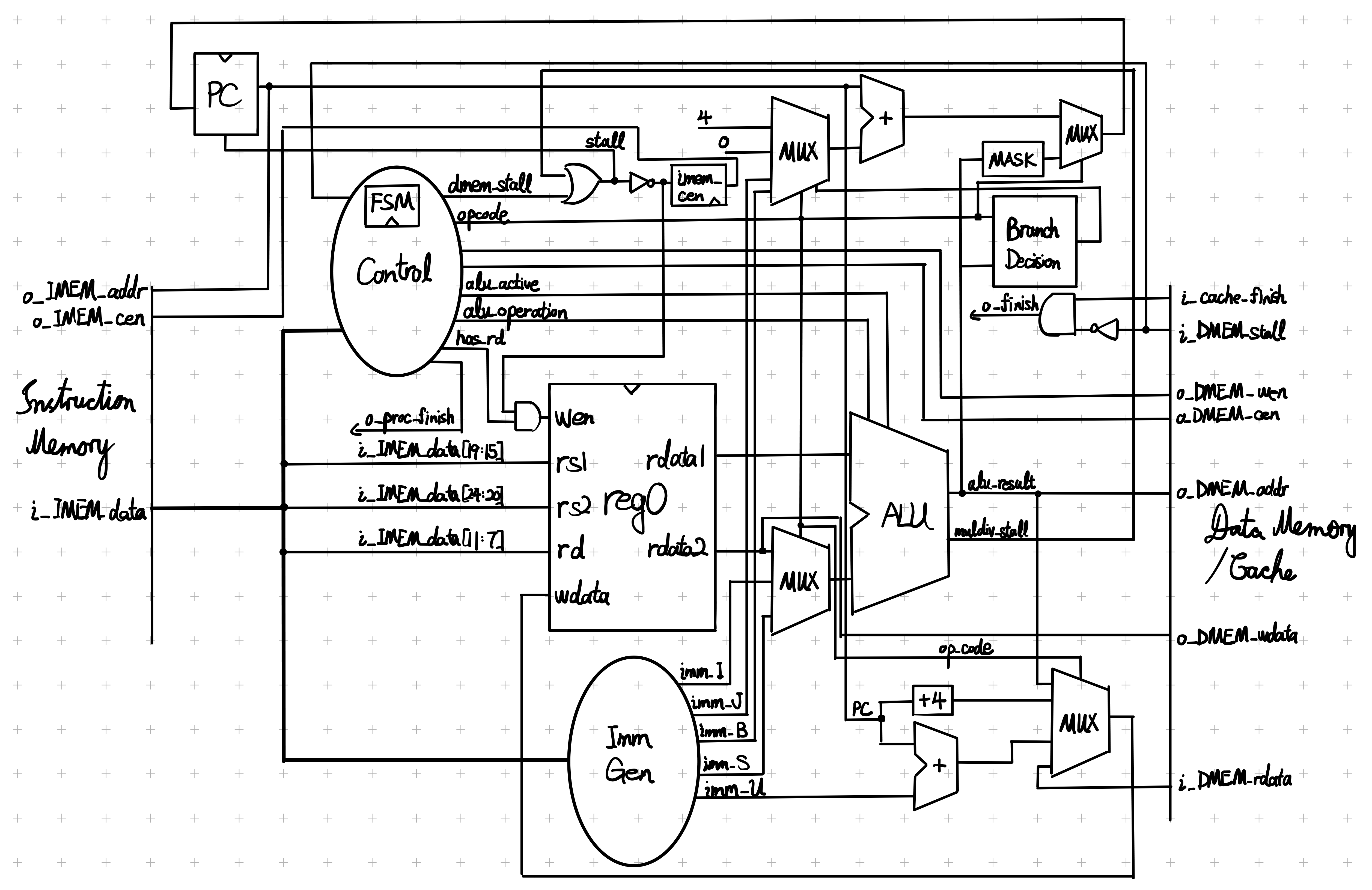
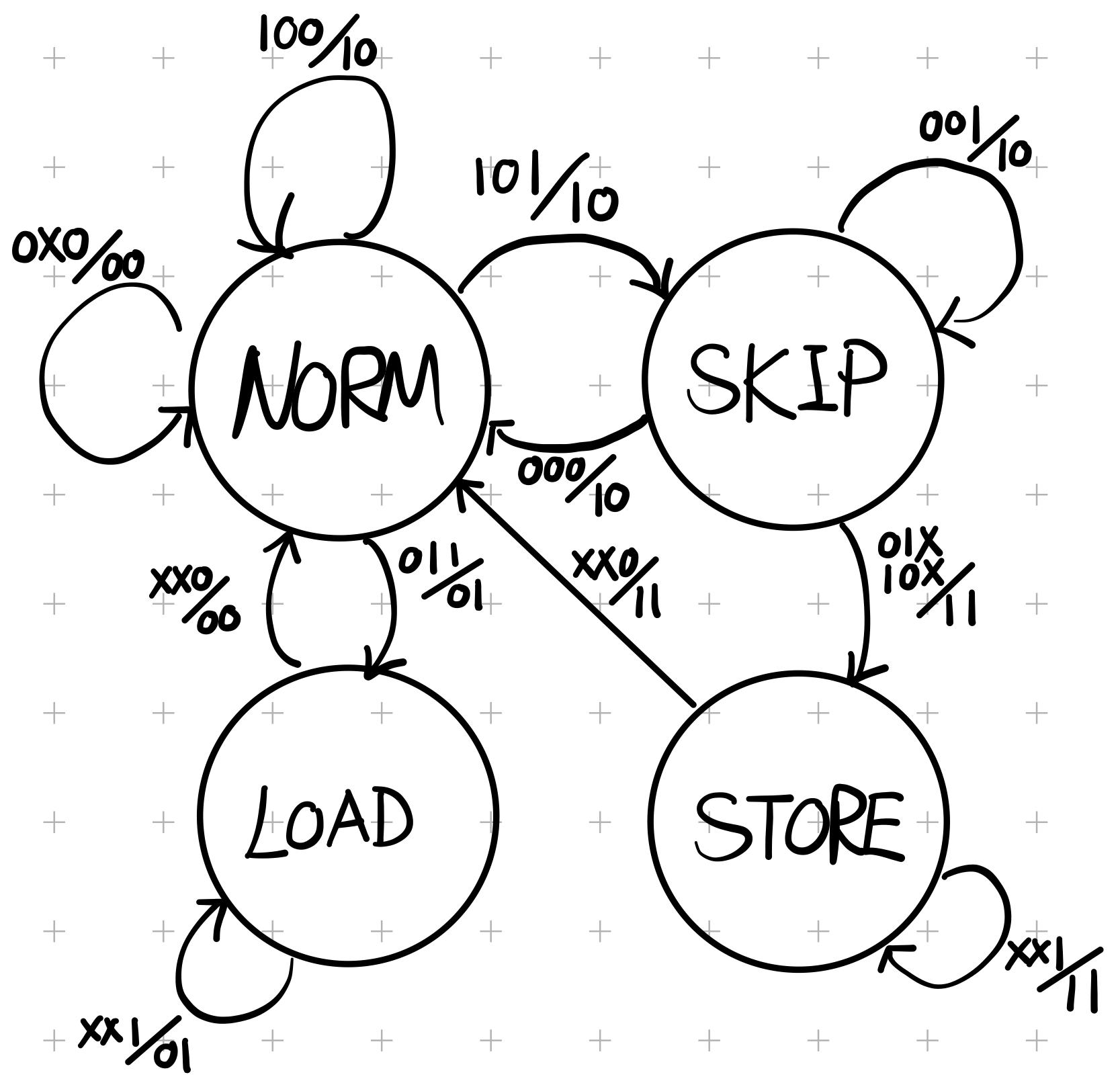
# Computer Architecture Project

**CPU Design**

Block diagram:



FSM:   
arrow format:



Data paths not in the lecture slides:

For JAL, write PC+4 to R[rd], and PC increment by imm\_J.

For JALR, write PC+4 to R[rd] while ALU sums R[rs1] and imm\_I. Then the result has its LSB masked and stored to PC.

For AUIPC, write PC+imm\_U to R[rd].

For ECALL, set o\_proc\_finish, stop PC incrementation, and set o\_finish to !i\_DMEM\_stall && i\_cache\_finish.

For multicycle instruction, we stall PC, register file and instruction memory if needed, until the stalling signal turns low.

Since no write-back is needed when storing data to memory, we may skip waiting for !i\_DMEM\_stall until memory access is needed for the current instruction. This is implemented by the above Mealy machine.

It is observed that the data memory requires the input write enable signal to be kept high in order to produce an accurate stall signal, so o\_DMEM\_wen is also produced by the FSM.

**Cache design**

**Cache 1**

**Specification:**

2-way set associative cache

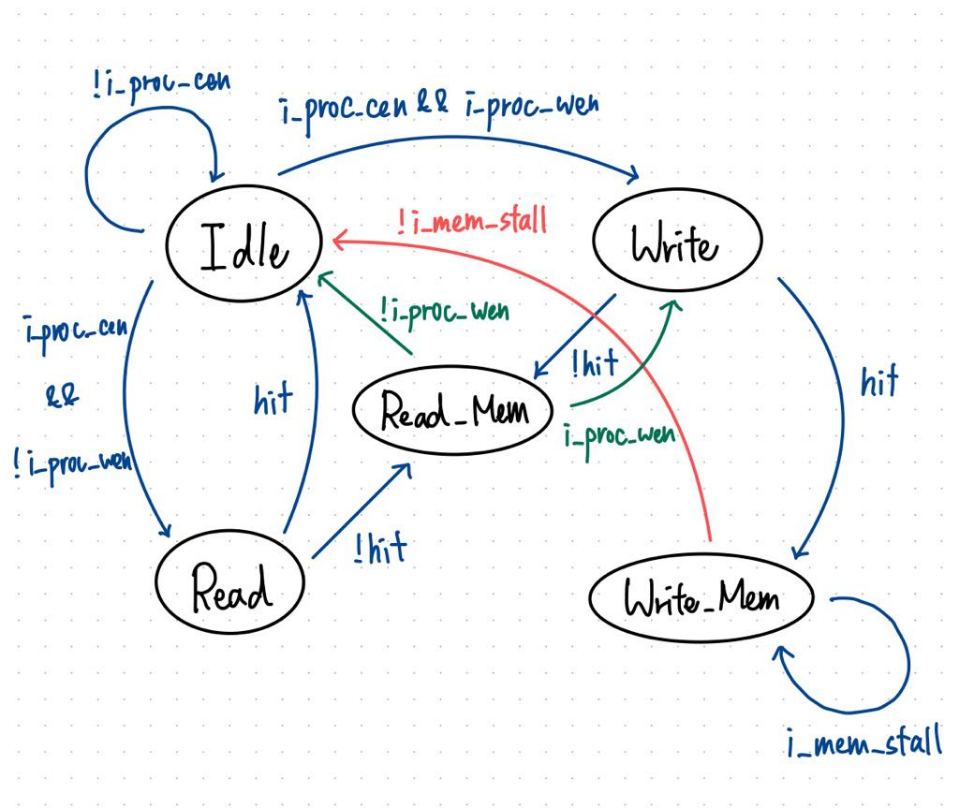
Number of Sets: 8

Block Size: 16 bytes

Write Policy: Write-through

Replacement Policy: Random

**Finite State Machine:**



**Cache 2**

**Specification**

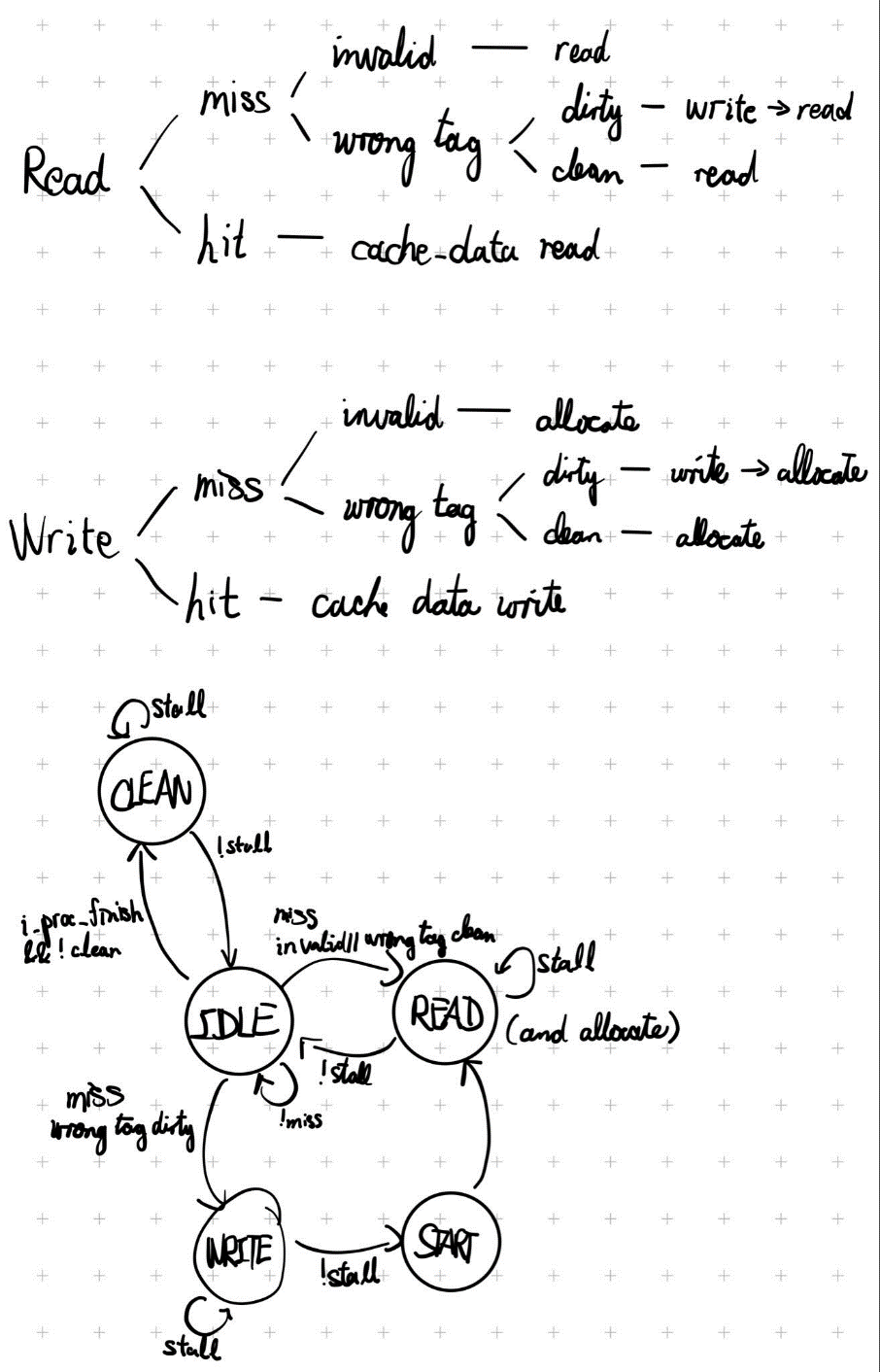
Direct mapped

Number of Blocks: 16

Block Size: 16 bytes

Write policy: Write-back

**Finite State Machine:**



**Time Performance Improvements:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction Set | I0 | I1 | I2 | I3 | Write Policy |
| No Cache | 77 | 290 | 202 | 1324 |  |
| Store skip | 74 | 261 | 183 | 1228 |  |
| 2-way associative | 63 | 266 | 198 | 742 | Through |
| Improvement | 1.22 | 1.09 | 1.02 | 1.78 |  |
| Store skip | 60 | 232 | 179 | 633 | Through |
| Improvement | 1.23 | 1.13 | 1.02 | 1.94 |  |
| Direct mapped | 55 | 180 | 142 | 426 | Back |
| Improvement | 1.40 | 1.61 | 1.42 | 3.11 |  |
| Store skip | 54 | 165 | 134 | 415 | Back |
| Improvement | 1.37 | 1.58 | 1.37 | 1.96 |  |

**Conclusion:**

Write policy influences more on clock cycle than cache structure. We may implement a better cache by using write-back in associative caches.