

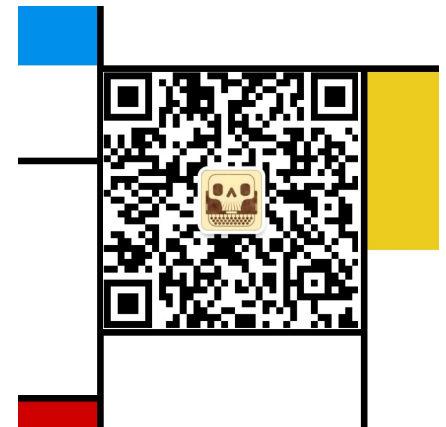
ECE411: Project Presentation

Huang, Jinghan & Ren, Yuhang

Office: 浙江大学国际校区一号书院3B19 & 2B17

Email address: jinghanhuang@zju.edu.cn

HP: 18161332576 & 13319259198



MP2: Overview

one-level, unified, 2-way set-associative cache

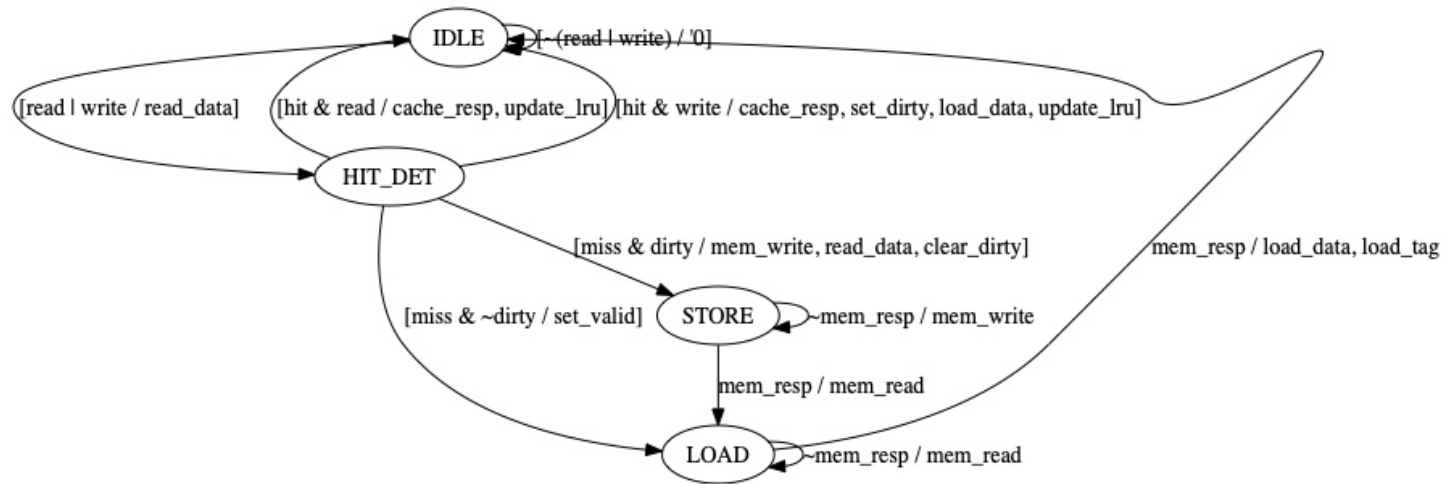
Specifications

- i. 8 lines per way with 8 words (32 bytes) per line
- ii. Write-back policy
- iii. LRU replacement policy
- iv. Read/Write hits take two clock cycles to complete
- v. 4 control states in cache controller

Components

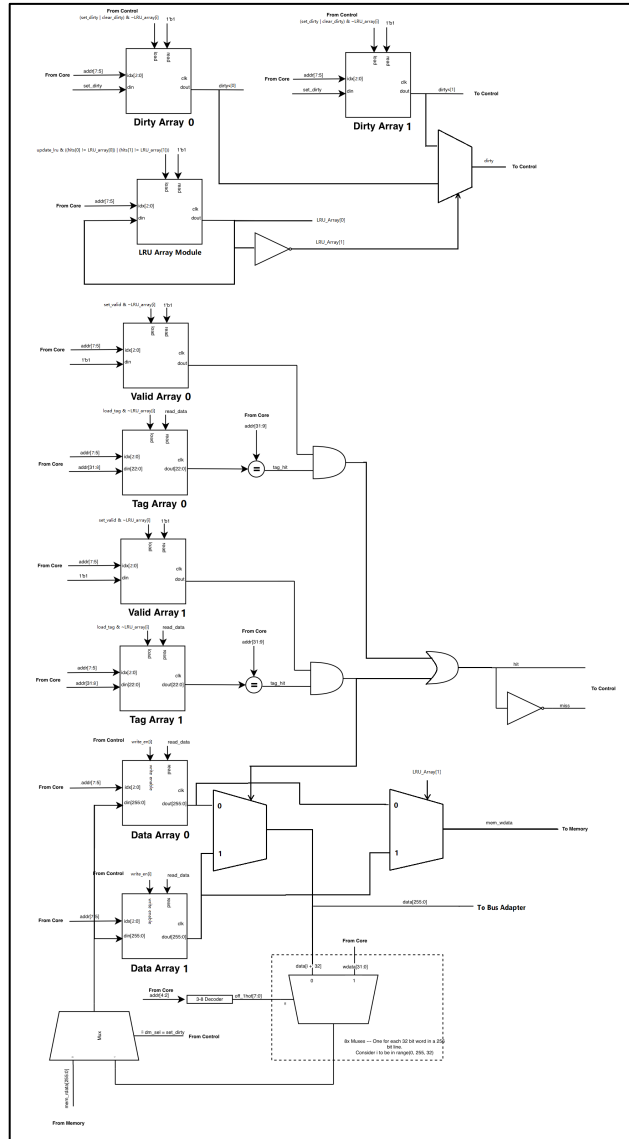
- i. Control unit, which implements the state machine for cache.
- ii. 4B to 32B bus adapter, which selects 32-bit memory from 256 bits according to byte-enable signals.
- iii. Cache datapath, which consists of the following parts:
 - 1. Dirty array, valid array, tag array and data array.
 - 2. LRU array module
 - 3. MUXes

MP2: Cache Control – Mealy State Machine

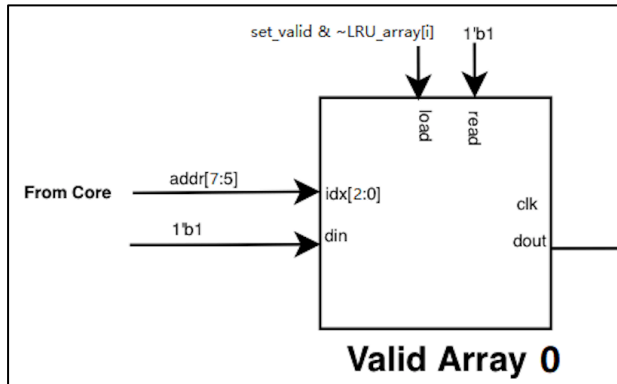


State	Input	Next state	Signal
IDLE	read write	HIT_DET	read_data = 1'b1;
HIT_DET	hit && read	IDLE	cache_resp = 1'b1; update_lru = 1'b1;
	hit && write	IDLE	cache_resp = 1'b1; set_dirty = 1'b1; load_data = 1'b1; update_lru = 1'b1;
	miss && dirty	STORE	clear_dirty = 1'b1; mem_write = 1'b1; read_data = 1'b1;
	miss && ~dirty	LOAD	set_valid = 1'b1;
STORE	mem_resp	LOAD	mem_read = 1'b1;
	~mem_resp	STORE	mem_write = 1'b1;
LOAD	mem_resp	IDLE	load_data = 1'b1; load_tag = 1'b1;
	~mem_resp	LOAD	mem_read = 1'b1;

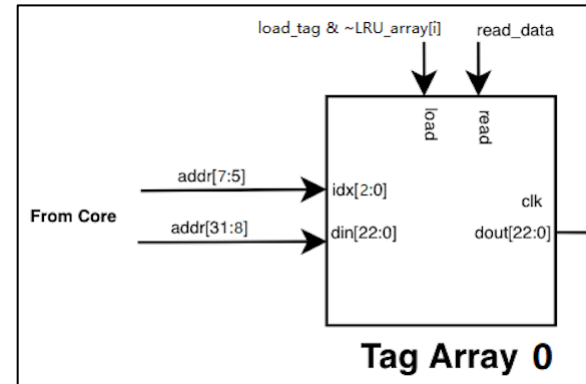
MP2: Cache Datapath



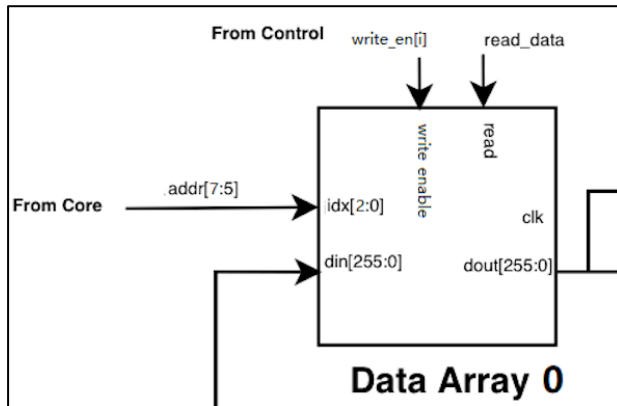
MP2: Cache Datapath



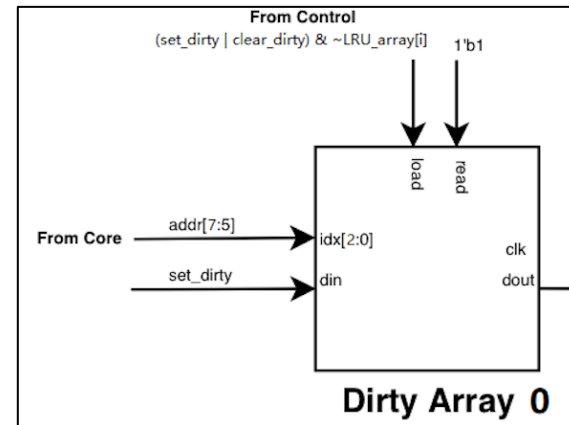
Valid Arrays



Tag Arrays



Data Arrays

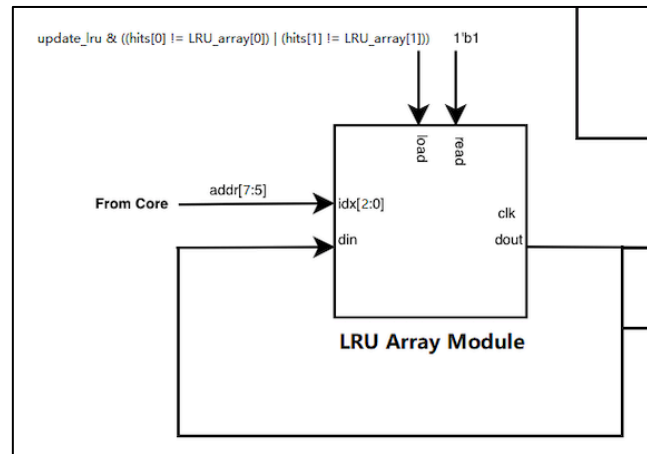


Dirty Arrays

MP2: Cache Datapath

```
// Arrays
genvar i;
generate begin: arrays
  for (i = 0; i < num_ways; i++)
  begin: array_loop
    ... assign tag_hit[i] = (tag == tags[i]);
    ... assign hits[i] = tag_hit[i] & valids[i];
    ...
    ... array #(s_index, 1) dirty_array
    ... (.clk,
    ...   .read(1'b1),
    ...   .load((set_dirty | clear_dirty) & ~LRU_array[i]),
    ...   .index,
    ...   .datain(set_dirty),
    ...   .dataout(dirtys[i])
    ... );
  end
end
```

MP2: Cache Datapath



LRU Array

```
if(LRU_array[0] == 1'b0)
begin
    dirty = dirtys[0];
    mem_wdata = datas[0];
    pmem_waddress = {tags[0], index, 5'b00000};
end
else
begin
    dirty = dirtys[1];
    mem_wdata = datas[1];
    pmem_waddress = {tags[1], index, 5'b00000};
end
```

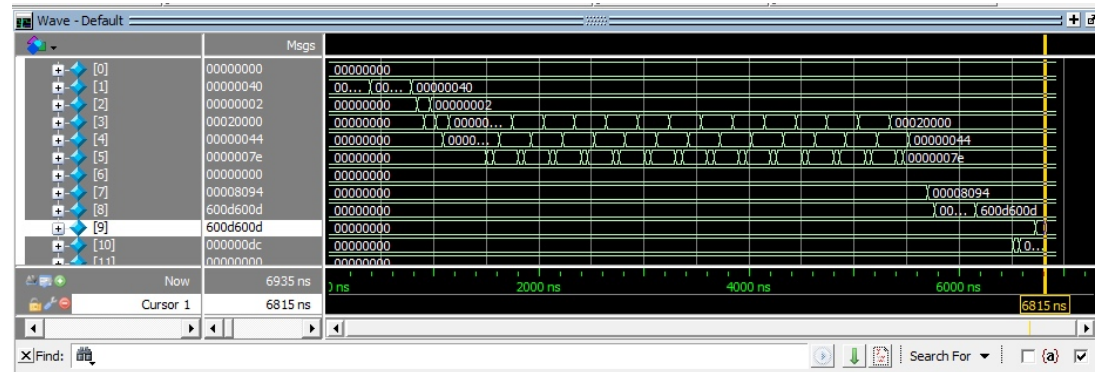
```
if(load_data) begin
    case(set_dirty)
    1: begin
        mux_data = mem_wdata256;
        write_en[0] = mem_byte_enable256 & {32{hits[0]}};
        write_en[1] = mem_byte_enable256 & {32{hits[1]}};
    end
    0: begin
        mux_data = mem_rdata;
        write_en[0] = {32{1'b1}} & {32{~LRU_array[0]}};
        write_en[1] = {32{1'b1}} & {32{~LRU_array[1]}};
    end
    endcase
end
```

MP2: Test Bench & Simulation Results

riscv_mp1test.s

```
riscv_mp1test.s      cache.sv

1  riscv_mp0test.s:~
2  .align 4~
3  .section .text~
4  .globl _start~
5  ~# Refer to the RISC-V ISA Spec for the functi~
6  ~# the instructions in this test program.~
7  _start:~
8  ~# Note that the comments in this file should~
9  ~# an example of good commenting style!! They~
10 ~# in an effort to help you understand the ass~
11 ~
12 ~# Note that one/two/eight are data labels~
13 ~lw x1, threshold # X1 <= 0x80~
14 ~lui x2, 2      # X2 <= 2~
15 ~lui x3, 8      # X3 <= 8~
16 ~srli x2, x2, 12~
17 ~srli x3, x3, 12~
18 ~
19 ~addi x4, x3, 4  # X4 <= X3 + 2~
20 ~
21 loop1:~
22 ~slli x3, x3, 1  # X3 <= X3 << 1~
23 ~xori x5, x2, 127 # X5 <= XOR (X2, 7b'11111111~
24 ~addi x5, x5, 1  # X5 <= X5 + 1~
25 ~addi x4, x4, 4  # X4 <= X4 + 8~
26 ~
27 ~bleu x4, x1, loop1 # Branch if last result~
28 ~
29 ~andi x6, x3, 64 # X6 <= X3 + 64~
30 ~
31 ~auipc x7, 8      # X7 <= PC + 8~
32 ~lw x8, good      # X8 <= 0x600d600d~
```



List			
File Edit View Add Tools Bookmarks Window Help			
List - Default			
ps	/mp2_tb/dut/cpu/datapath/regfile/data[8]		
delta	/mp2_tb/dut/cpu/datapath/regfile/data[9]		
0 +0		00000000	00000000
5765000 +1		00000098	00000000
6155000 +1		600d600d	00000000
6725000 +1		600d600d	000000ac
6815000 +1		600d600d	600d600d

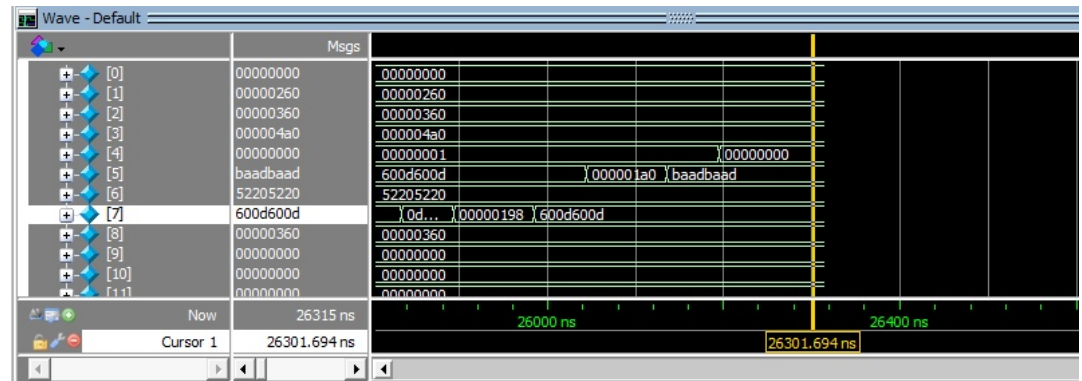
MP2: Test Bench & Simulation Results

mp2_final.s

```

mp2_final.s
1 lw_sw_all:
2 .align 4
3 .section .text
4 .globl _start
5 _start:
6
7 # Get some base addresses
8 la x1, G60 # R0
9 la x2, S60 # R1
10 la x3, B60 # R2
11
12 lw x8, B74
13 lw x8, S76
14 lw x7, B70
15 lw x8, G7C
16 lw x7, G7C
17 lw x7, S7E
18 lw x7, B78
19 lw x7, B78
20 lw x7, B78
21 lw x6, B62
22
23 lw x5, S66
24 sw x5, 16(x2) # S68 <= x5
25
26 lw x6, B66
27 sw x2, 20(x2) # S6A <= x2
28
29 sw x2, 8(x2) # S64 <= x2
30 sw x3, 0(x1) # G60 <= x2
31
32 addi x5, x2, 1

```




ps	delta	/mp2_tb/dut/cpu/datapath/regfile/data[5]	/mp2_tb/dut/cpu/datapath/regfile/data[6]	/mp2_tb/dut/cpu/datapath/regfile/data[7]
23945000	+1	600d600d	52205220	600d600d
24005000	+1	600d600d	52205220	00000190
24095000	+1	600d600d	52205220	0d900d8f
24155000	+1	600d600d	52205220	00000198
24245000	+1	600d600d	52205220	600d600d
24305000	+1	000001a0	52205220	600d600d
24395000	+1	baadbaad	52205220	600d600d
24575000	+1	baadbaad	00000180	600d600d
24665000	+1	baadbaad	52205220	600d600d
24725000	+1	00000188	52205220	600d600d
24815000	+1	600d600d	52205220	600d600d
24875000	+1	600d600d	52205220	00000190
24965000	+1	600d600d	52205220	0d900d8f
25025000	+1	600d600d	52205220	00000198
25115000	+1	600d600d	52205220	600d600d
25175000	+1	000001a0	52205220	600d600d
25265000	+1	baadbaad	52205220	600d600d
25445000	+1	baadbaad	00000180	600d600d
25535000	+1	baadbaad	52205220	600d600d
25595000	+1	00000188	52205220	600d600d
25685000	+1	600d600d	52205220	600d600d
25745000	+1	600d600d	52205220	00000190
25835000	+1	600d600d	52205220	0d900d8f
25895000	+1	600d600d	52205220	00000198
25985000	+1	600d600d	52205220	600d600d
26045000	+1	000001a0	52205220	600d600d
26135000	+1	baadbaad	52205220	600d600d

MP2: Timing Analysis


Stratix IV EP4SGX530NF45C4ES

Slow 950mV 85C	83.56MHz
Slow 950mV 0C	86.33MHz

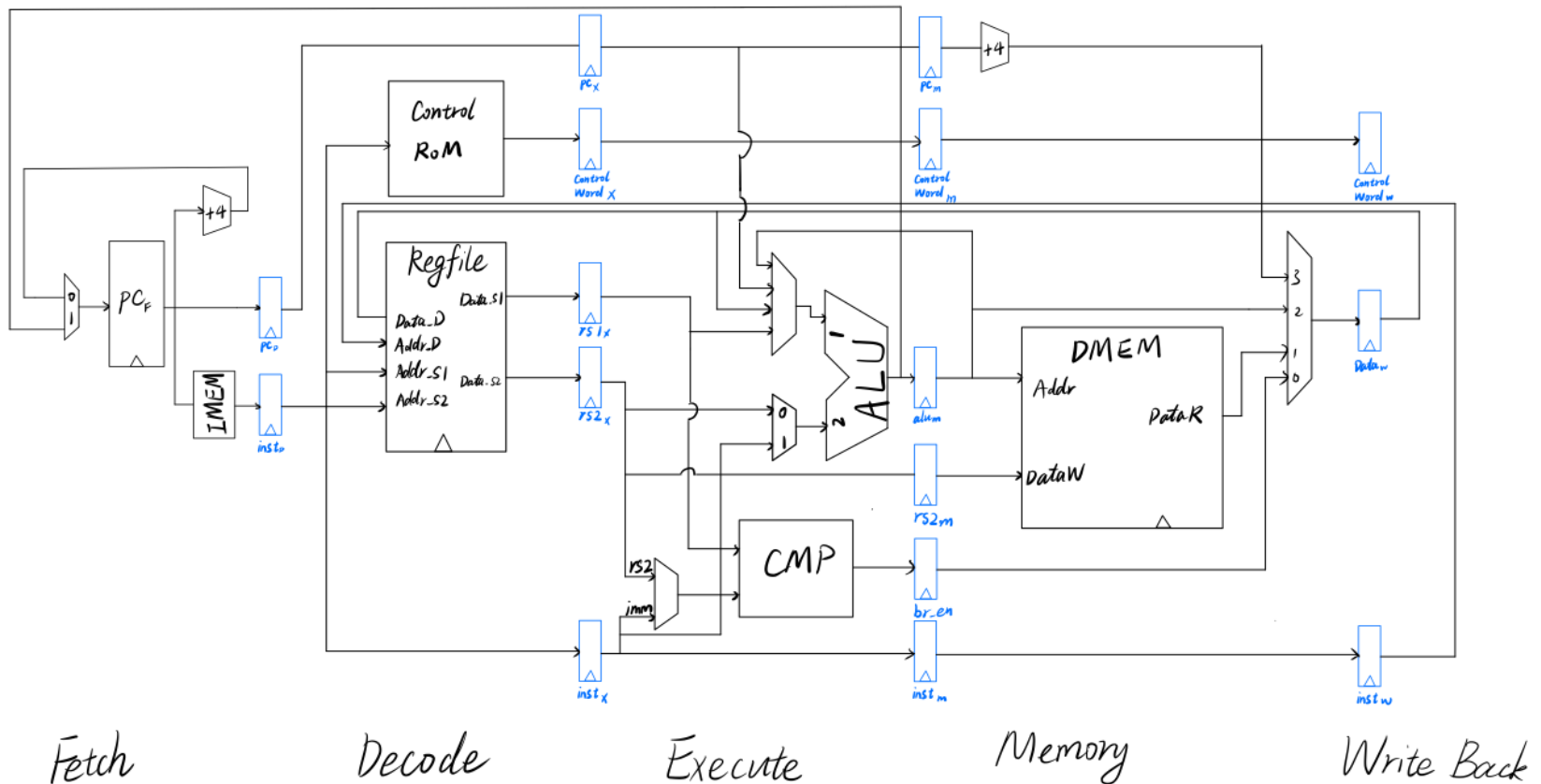
Slow 950mV 85C Model Fmax Summary

 <<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	83.56 MHz	83.56 MHz	clk	

Slow 950mV 0C Model Fmax Summary

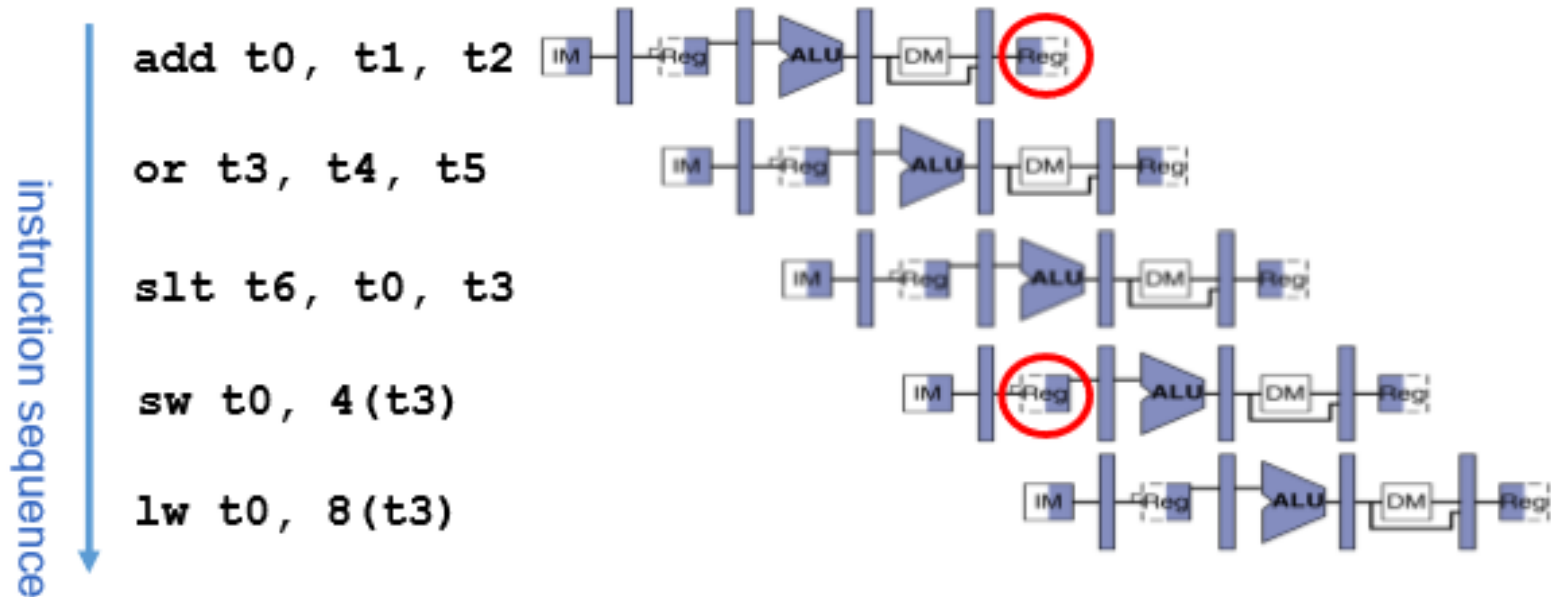
 <<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	86.33 MHz	86.33 MHz	clk	

MP3: Pipeline Control



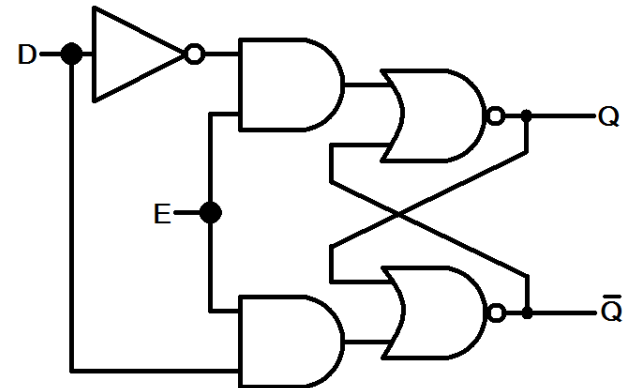
Five stage pipeline, passing instruction and pc etc. stage by stage.
Generate control word in Decode Stage, and pass on.

MP3 Conflict Handling: Structural Hazard

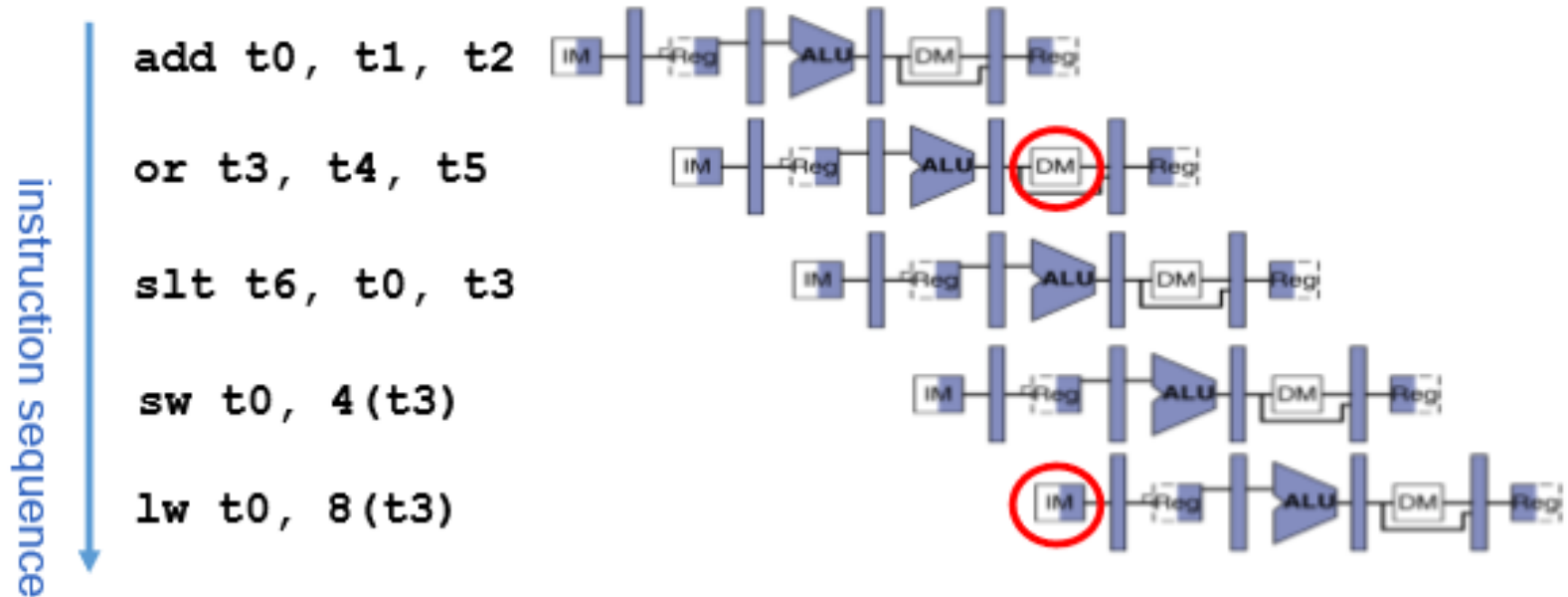


Transparent register file: An instruction writes into a register in the first half cycle and another instruction can access the same register in the next half cycle.

One possible solution: Latch Registers

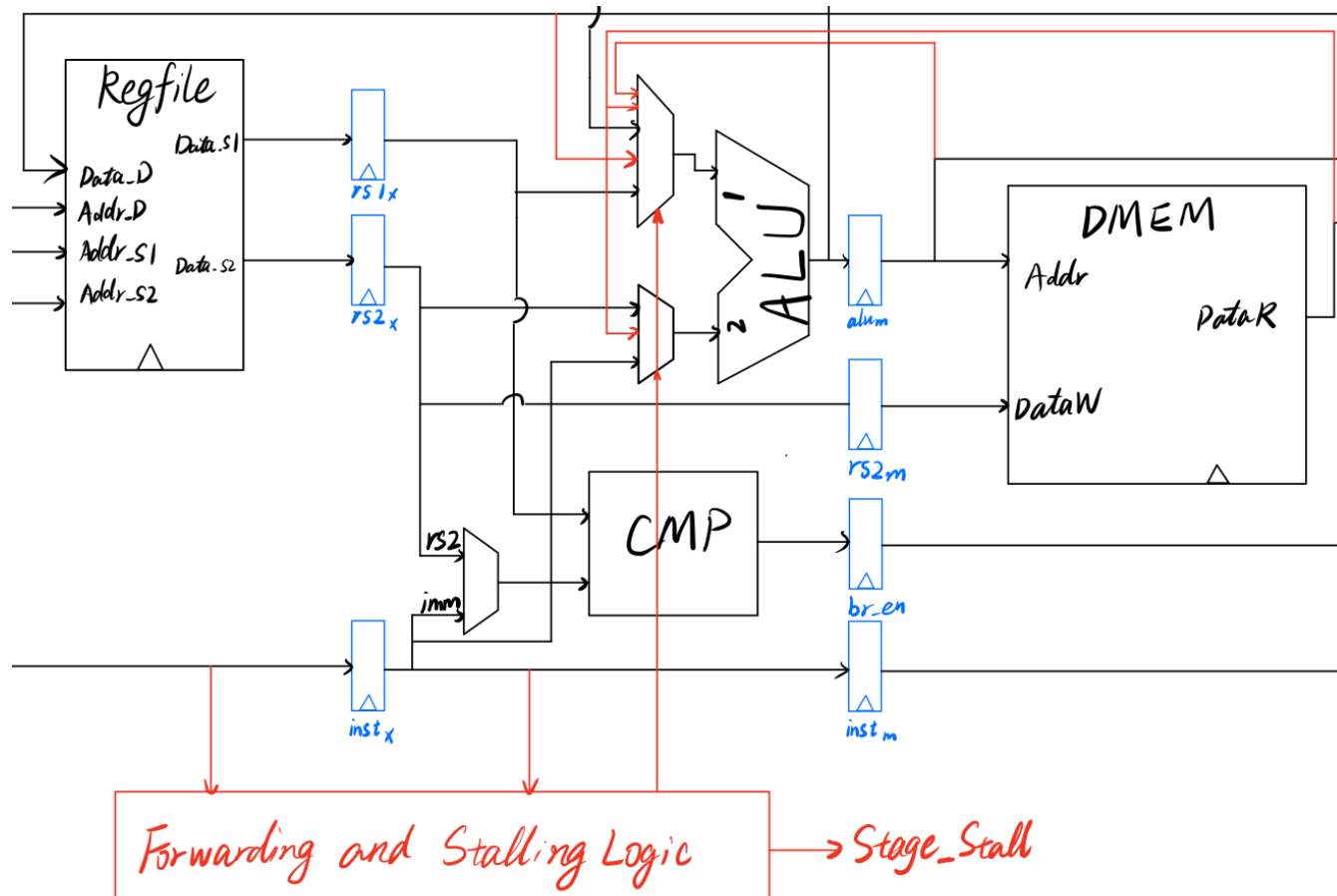


MP3 Conflict Handling: Structural Hazard

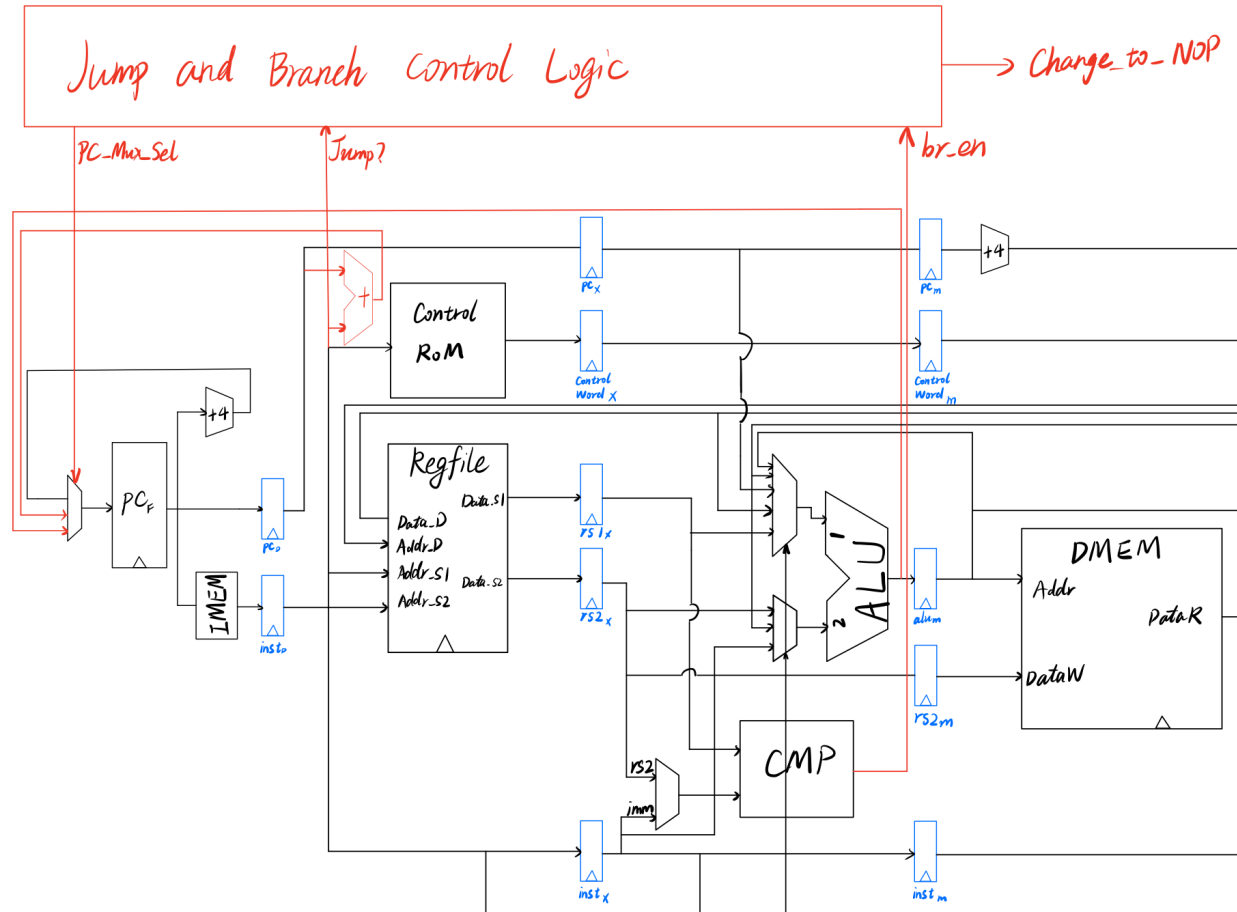


Use of two separate memories:
IMEM and DMEM are implemented as two caches
(More on this later)

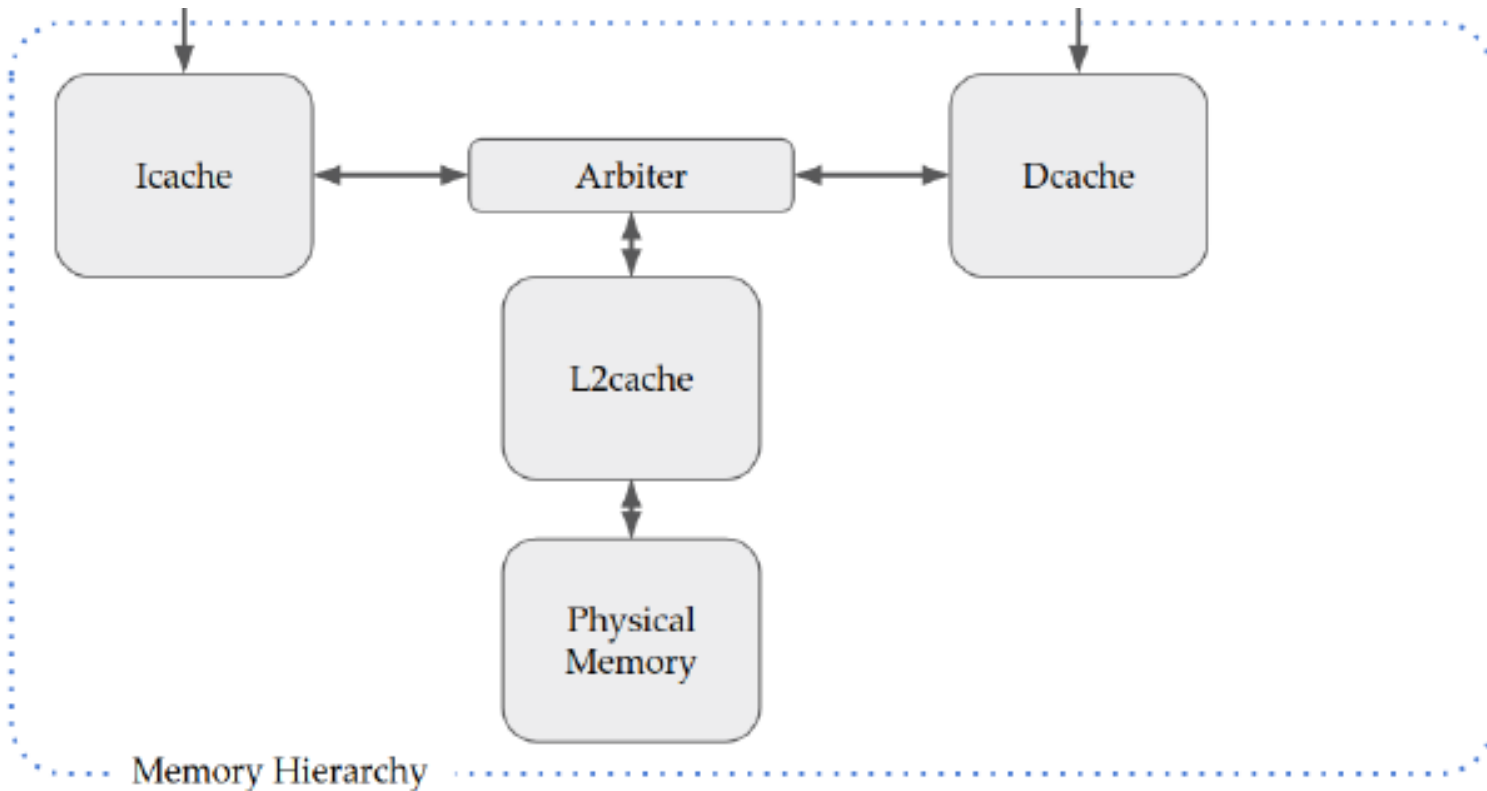
MP3 Conflict Handling: Data Hazard



MP3 Conflict Handling: Control Hazard

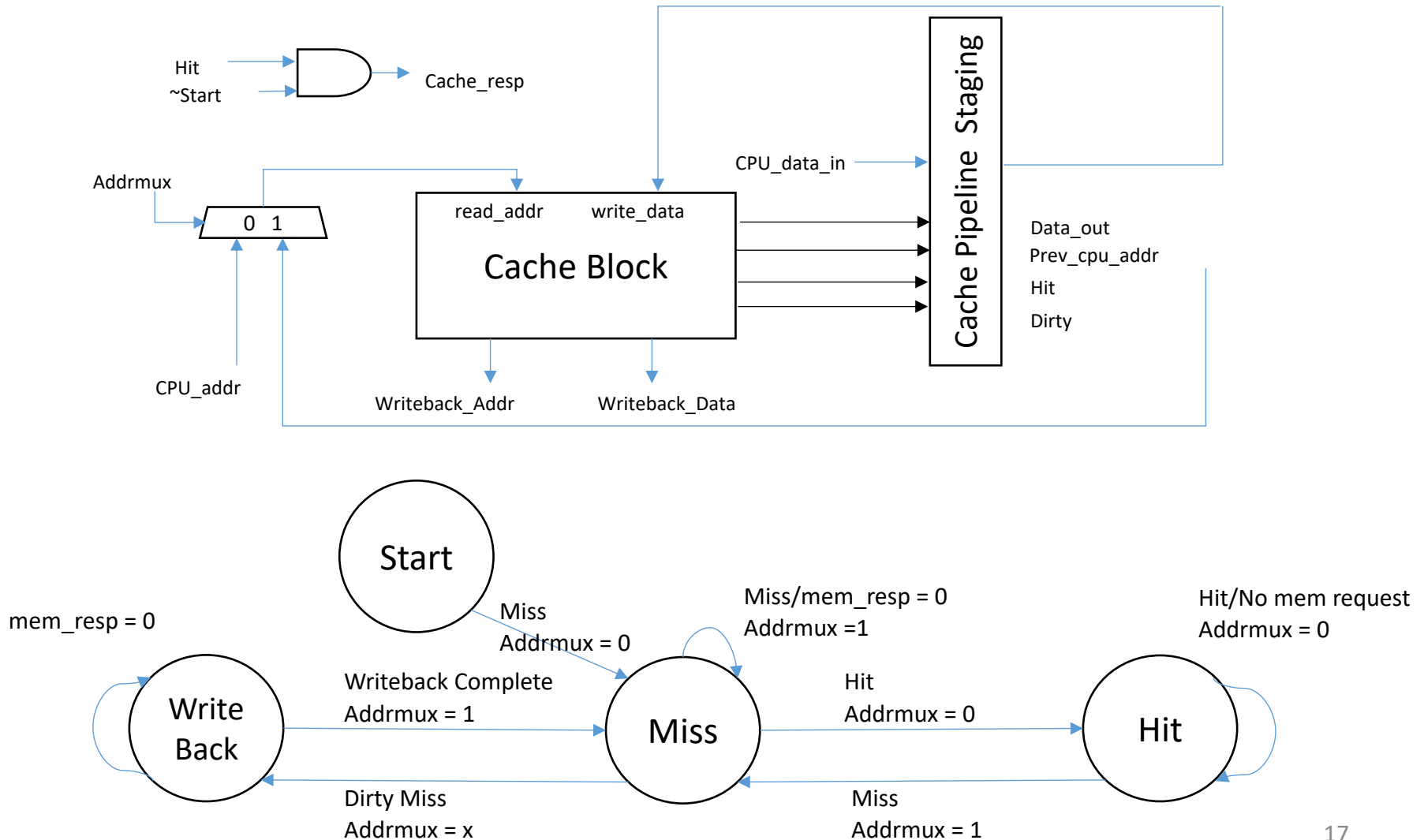


MP3 Memory Hierarchy

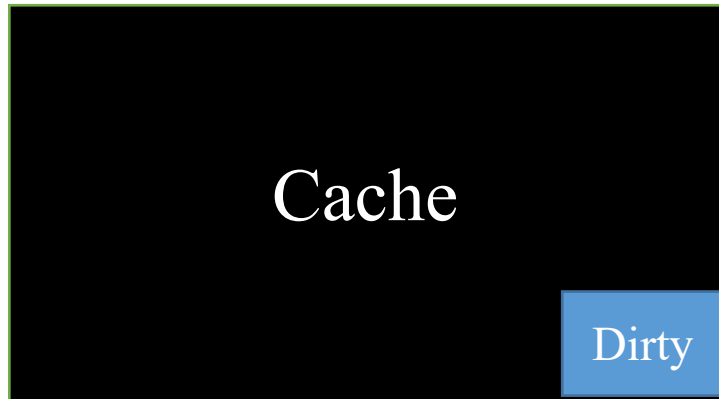


- Split L1 Cache
- Arbiter
- Unified L2 Cache

MP3: Memory: D-cache and I-cache



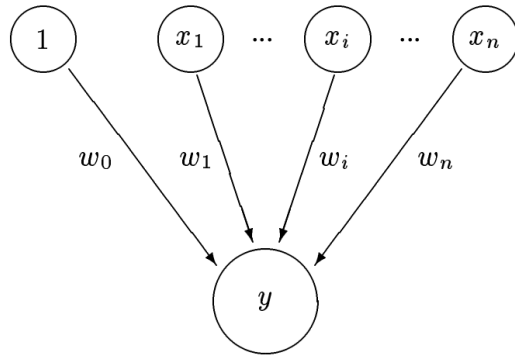
MP3 Optimization: Eviction Write Buffer



MP3 Optimization: Eviction Write Buffer



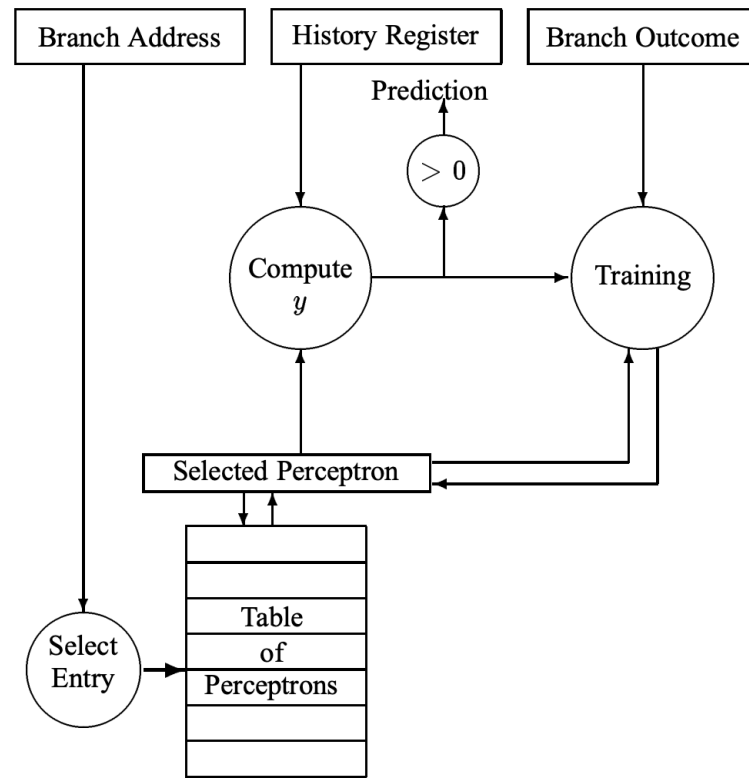
MP3 Optimization: Neural Branch Prediction



Perceptron Model

```
if  $\text{sign}(y_{out}) \neq t$  or  $|y_{out}| \leq \theta$  then
  for  $i := 0$  to  $n$  do
     $w_i := w_i + tx_i$ 
  end for
end if
```

Training Algorithm



Perceptron Predictor Block Diagram

MP3 Optimization: Neural Branch Prediction

- ***History Length***

- Long history lengths can yield more accurate predictions. The best history lengths ranged from 12 to 62, depending on the hardware budget.

- ***Representation of weights.***

- The weights for the perceptron predictor are signed integers.

- ***Threshold***

- The threshold is a parameter used to decide whether the predictor needs more training. The best threshold for a given history length h is always exactly $\theta = \lfloor 1.93h + 14 \rfloor$

MP3 Optimization: Hardware Prefetching - Stream Buffer

