CS 110
Computer Architecture
Pipeline II

Instructors:

Chundong Wang, Siting Liu & Yuan Xiao

Course website: https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2025/index.html

School of Information Science and Technology (SIST)

ShanghaiTech University

2025/4/8

Administratives

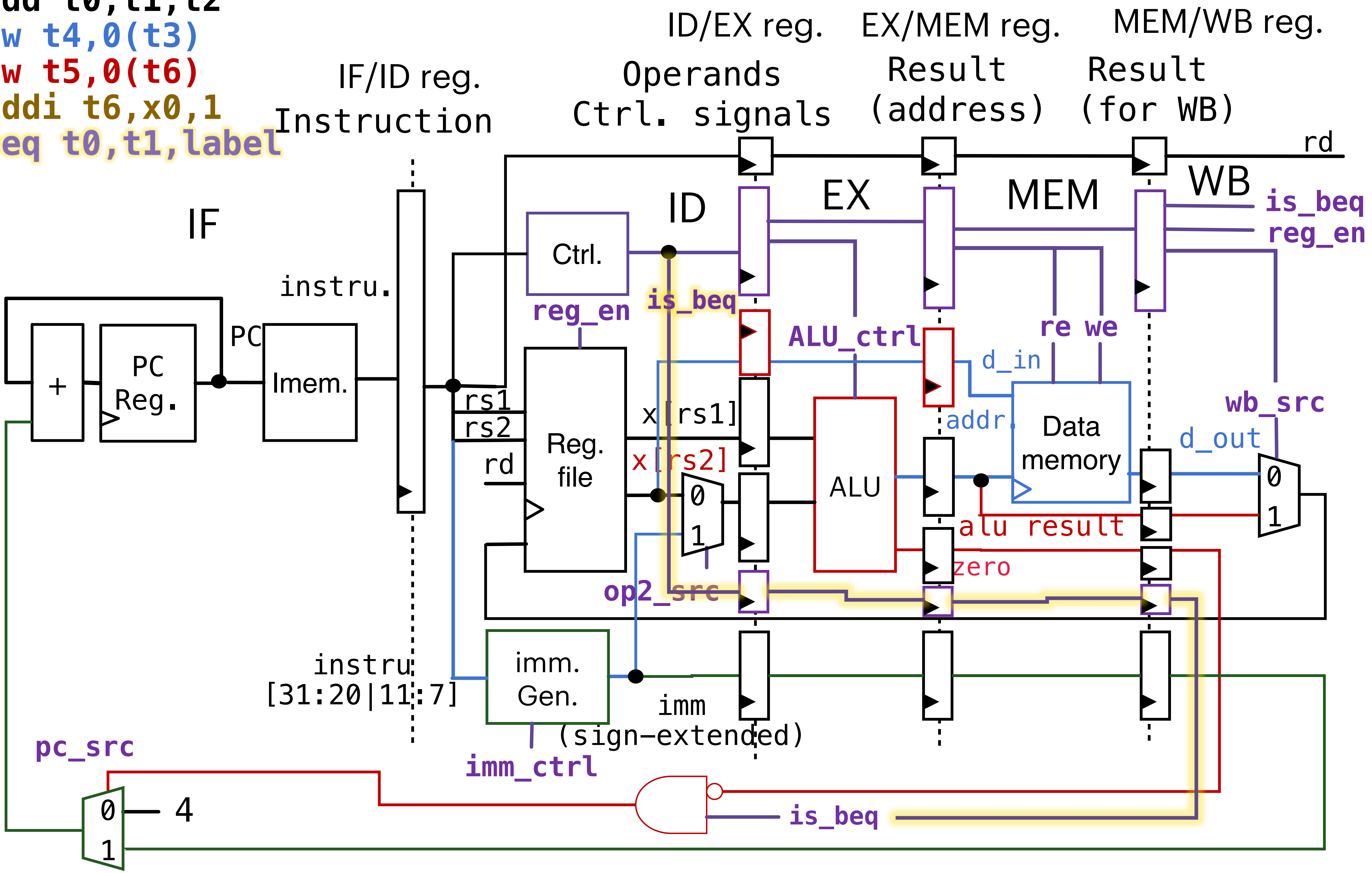
- HW 4 ddl April 15th
- Proj. 1.2 ddl April 17th
- Friday discussion (teaching center 301) on datapath .
- This week to check project 1.1 in lab sessions. TA will ask questions about project 1.1. Lab 8 will be released today, to check next week.
- Mid-term I on this Thursday, 8am-10am, in the teaching center 301/303, ARRIVE EARLY to find your seats and get prepared! Remember to bring your student ID cards. They will be checked during the exam.

Outline

- Starting this lecture, we will improve the performance of our CPU
- Performance evaluation
- Pipeline
- Hazards
    - Structural hazards
    - Data hazards
    - Control hazards

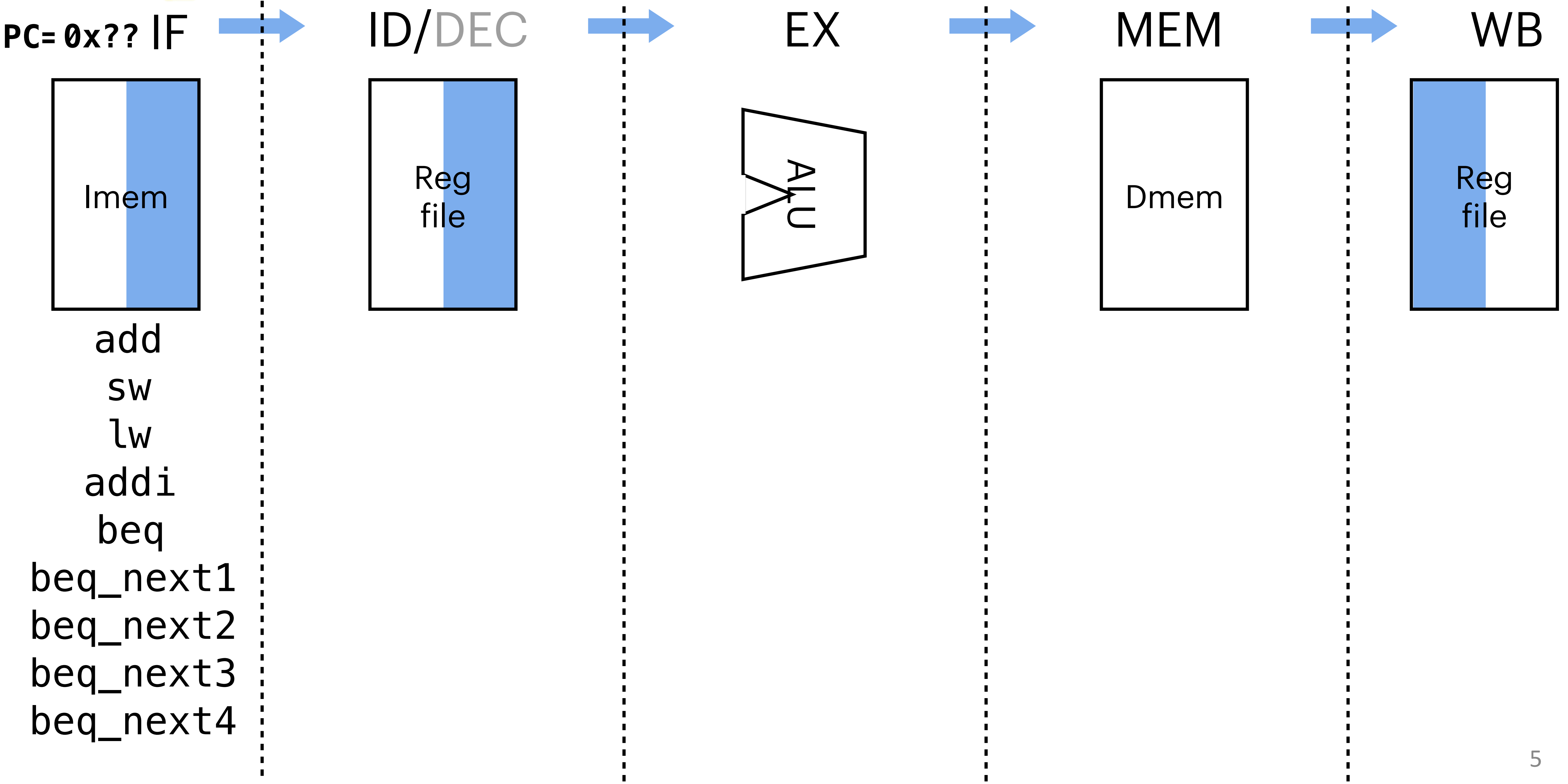
Detailed considerations

```
add t0,t1,t2
sw t4,0(t3)
lw t5,0(t6)
addi t6,x0,1
beq t0,t1,label
```



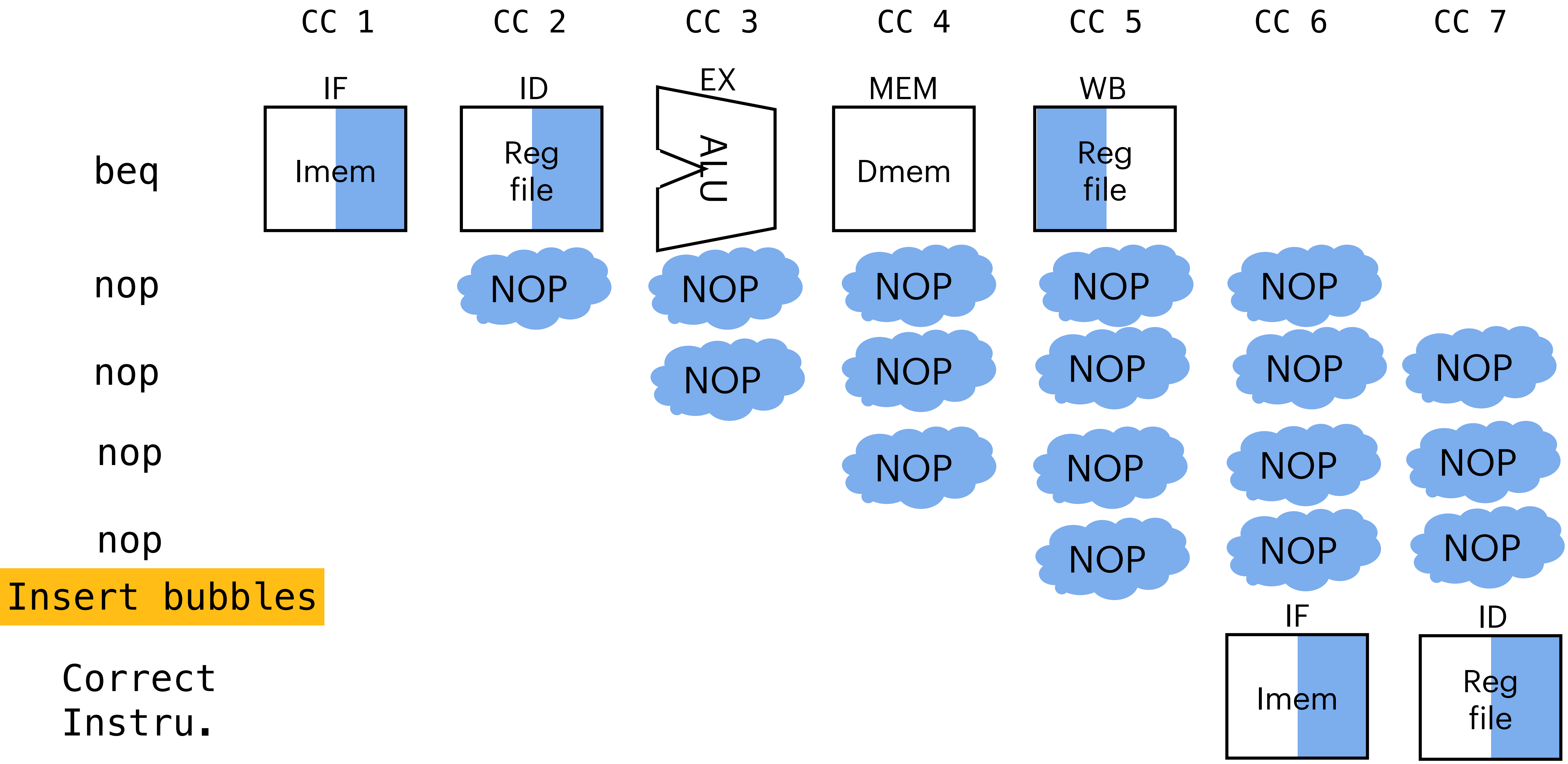
```
0x0:add t0,t1,t2
0x4:sw t4,0(t3)
0x8:lw t5,0(t6)
0xc:addi t6,x0,1
0x10:beq t0,t1,label
0x14:beq_next1
0x18:beq_next2
0x1c:beq_next3
0x20:beq_next4
0x24:beq_next5
```

Detailed considerations



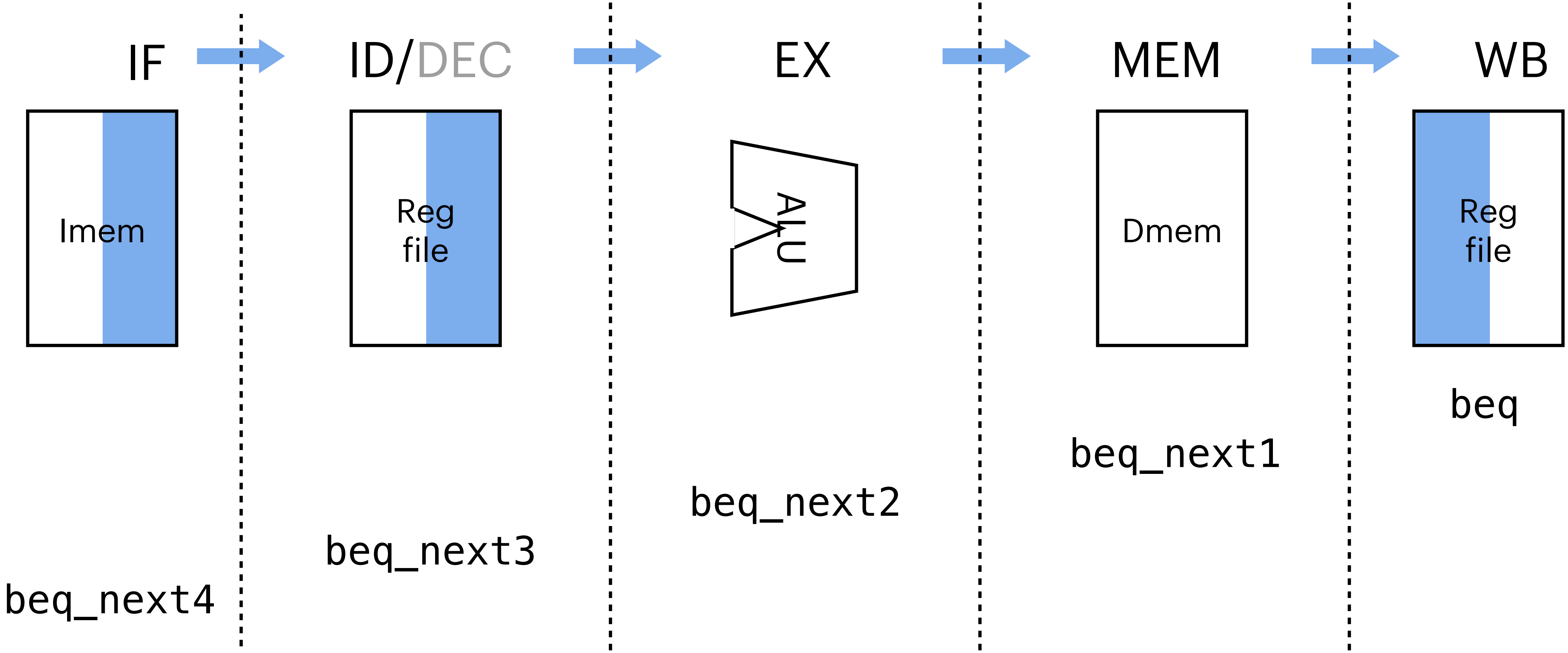
Control hazards--solution 1

We can wait ...



Control hazards -- solution 2

- Assume branch not taken (static)
- Extra control logics to deal with the cases that the branches are taken
  - Flush the pipeline and restore the states



Speculation

Control hazards -- solution 2

- Assume branch not taken (static)
- Not optimal in some cases

```
int A[20];
int sum = 0;
for (int i=0; i < 20; i++)
    sum +=  A[i];

# Assume x8 holds pointer to A
# Assign x10=sum
add  x10, x0, x0 # sum=0
add  x11, x8, x0 # ptr = A
addi x12,x11, 80 # end = A + 80
Loop:
    lw   x13,0(x11)   # x13 = *ptr
    add  x10,x10, x13 # sum += x13
    addi x11,x11, 4   # ptr++
blt x11, x12, Loop  # ptr < end
```

Wrong speculations except the last branch

Control hazards -- solution 2

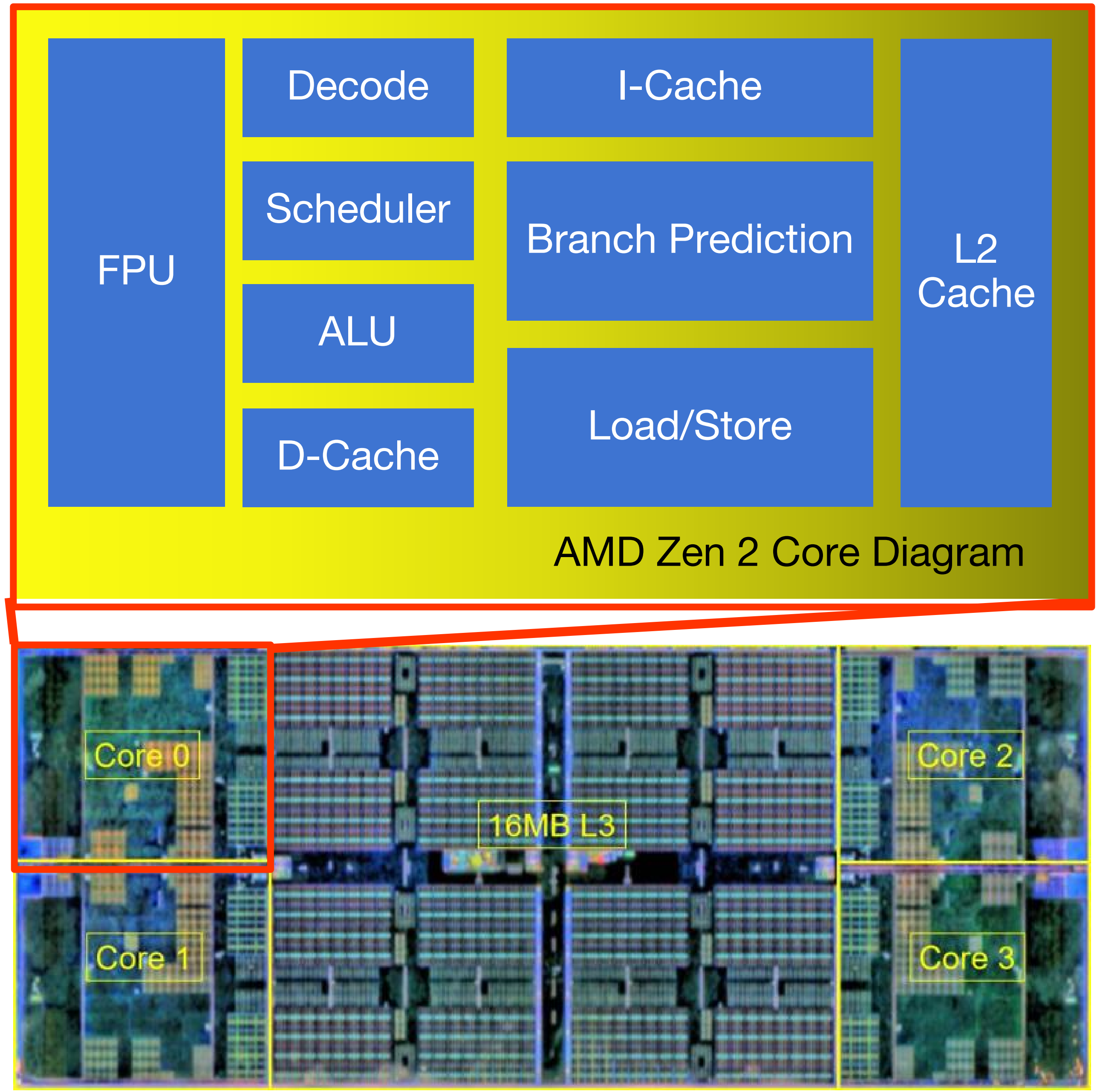
- Alternatively, dynamic branch prediction (when the program is running)

```
int A[20];
int sum = 0;
for (int i=0; i < 20; i++)
    sum +=  A[i];

# Assume x8 holds pointer to A
# Assign x10=sum
add  x10, x0, x0 # sum=0
add  x11, x8, x0 # ptr = A
addi x12,x11, 80 # end = A + 80
Loop:
    lw   x13,0(x11)   # x13 = *ptr
    add  x10,x10, x13 # sum += x13
    addi x11,x11, 4   # ptr++
blt x11, x12, Loop  # ptr < end
```

- Record the position of branch
- Record if the branch is taken for this branch
- Predict if the branch will be taken based on the current record
- Can be modeled as an FSM
- Use one or more bits to represent “(strong) taken” or “(strong not taken)”

Real stuff



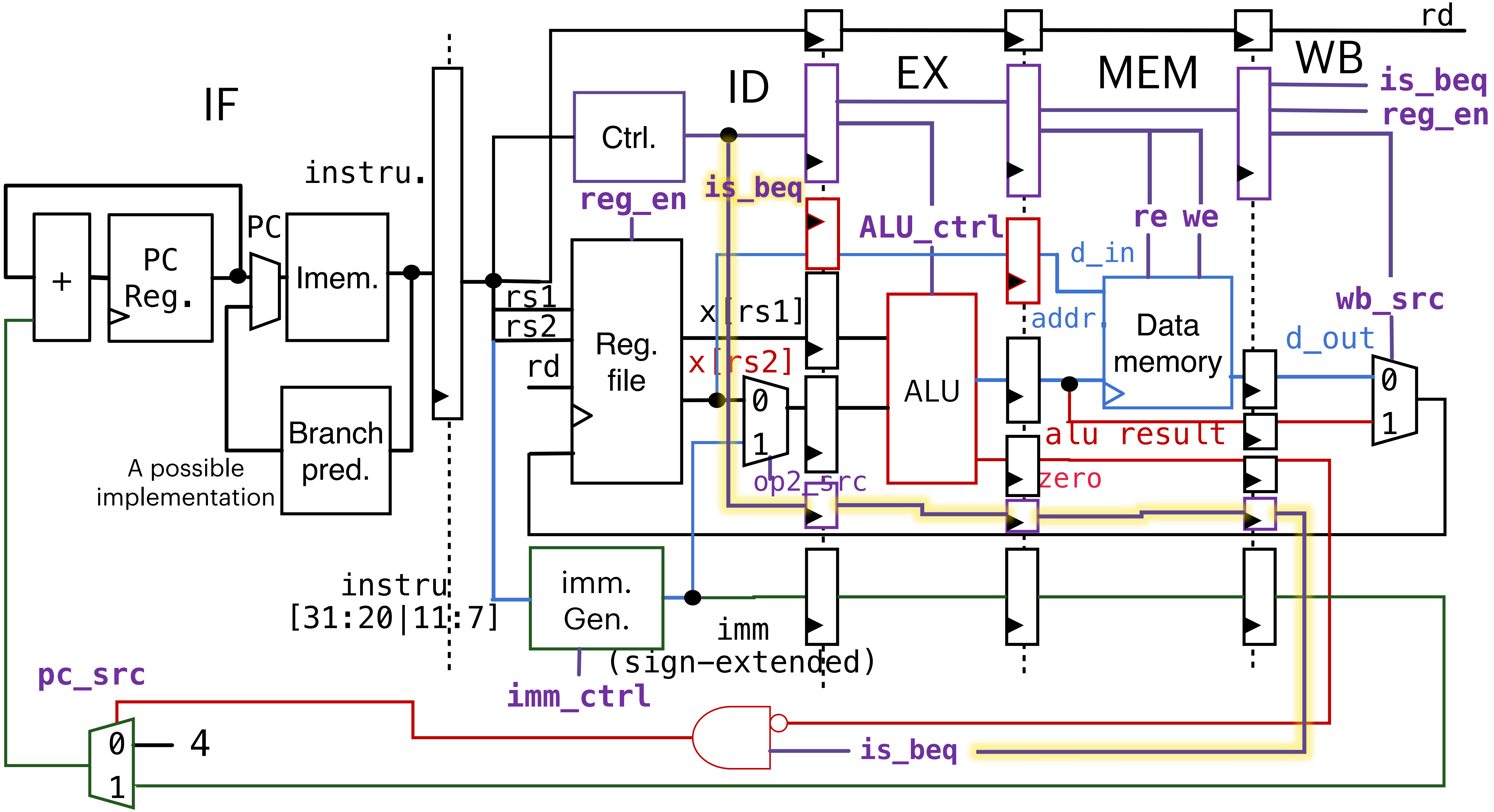
Unit	Zen	Zen 2
Floating Point	128b	256b
L0 Branch Target Buffer	8 entries	16 entries
L1 Branch Target Buffer	256 entries	512 entries
L2 Branch Target Buffer	4K entries	7K entries
Op Cache	2K ops	4K ops
Integer Physical Register File	168 entries	180 entries
Integer Scheduler	84 entries	92 entries
AGEN	2	3
ROB	192 entries	224 entries
L2DTLB	1.5K	2K
L3 Cache Size	8MB	16MB

7.83 mm^2 per core

[1] T. Singh et al., "2.1 Zen 2: The AMD 7nm Energy-Efficient High-Performance x86-64 Microprocessor Core," IEEE International Solid- State Circuits Conference - (ISSCC), 2020, pp. 42-44.

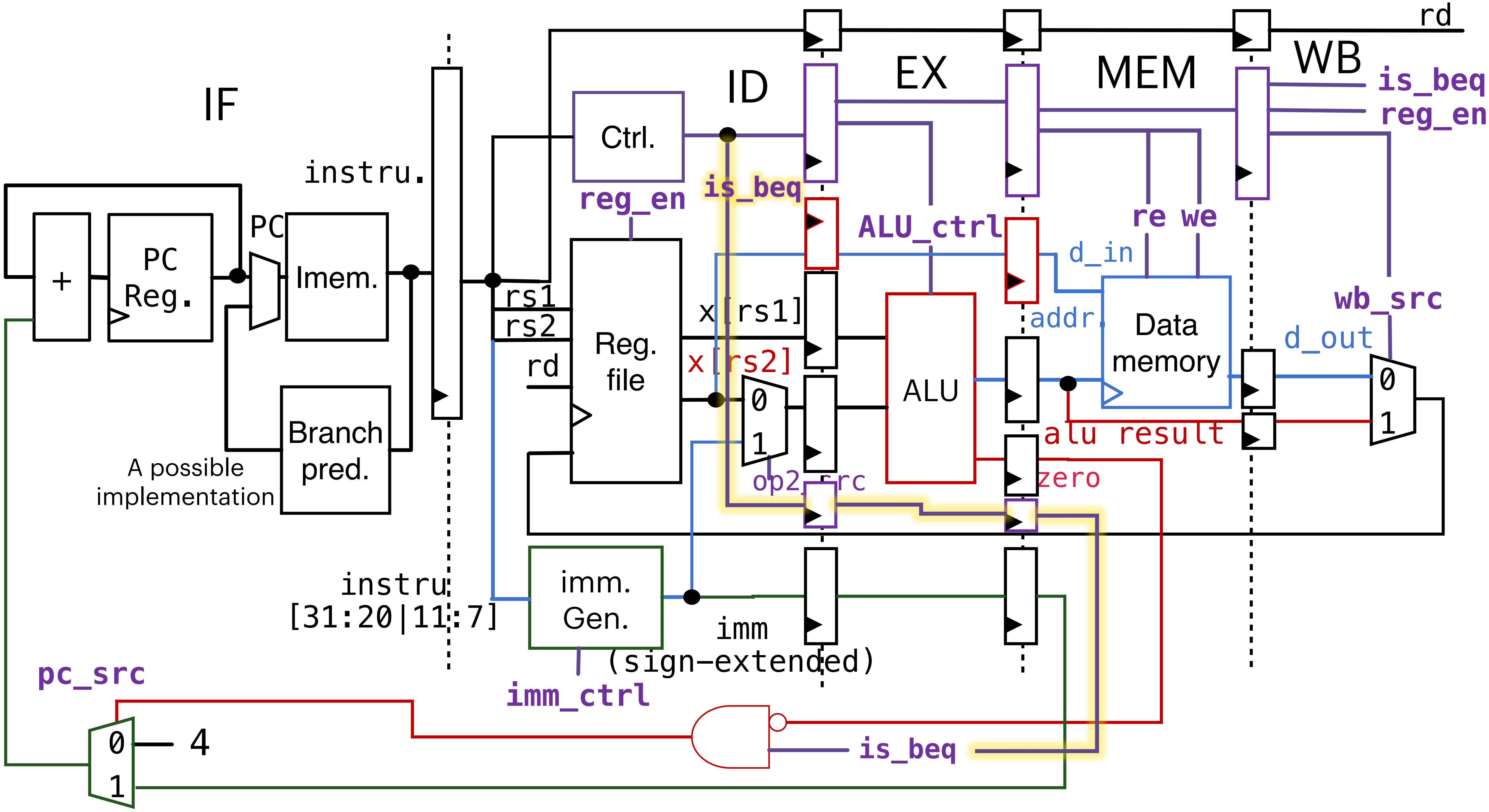
Control hazards -- solution 3

- Use idea similar to forwarding to reduce the delay of branches



Control hazards -- solution 3

- Use idea similar to forwarding to reduce the delay of branches



Summary on control hazards

- The delay in determining the proper instruction to fetch is called a control hazard or branch hazard

CS 110
Computer Architecture
Multi-issue

Instructors:

Chundong Wang, Siting Liu & Yuan Xiao

Course website: https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2025/index.html

School of Information Science and Technology (SIST)

ShanghaiTech University

2025/4/8

Outline

- Overview on parallelism
- Multi-issue
  - Static multi-issue (VLIW)
  - Dynamic multi-issue (superscalar)
  - Design cases in modern computer systems

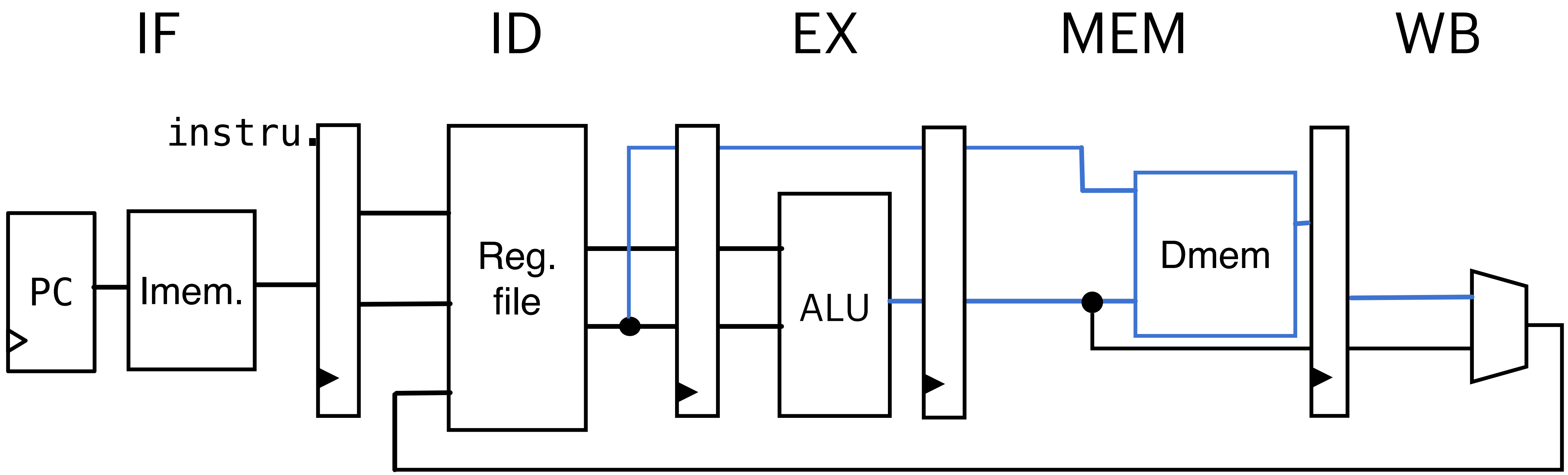
Review

- Overview on parallelism
  - Instruction-level parallelism (ILP)
    - Pipeline, deeper for faster clock, but potentially more hazards
    - Today’s lecture (Multi-issue)
  - Data-level parallelism (DLP)
    - SIMD
  - Thread-level parallelism (TLP)
    - Multi-threading/Hardware hyper-threading

$$\frac{Time}{Program} = \frac{Instructions}{Program} \cdot \frac{Cycles}{Instruction} \cdot \frac{Time}{Cycle}$$

Single-issue

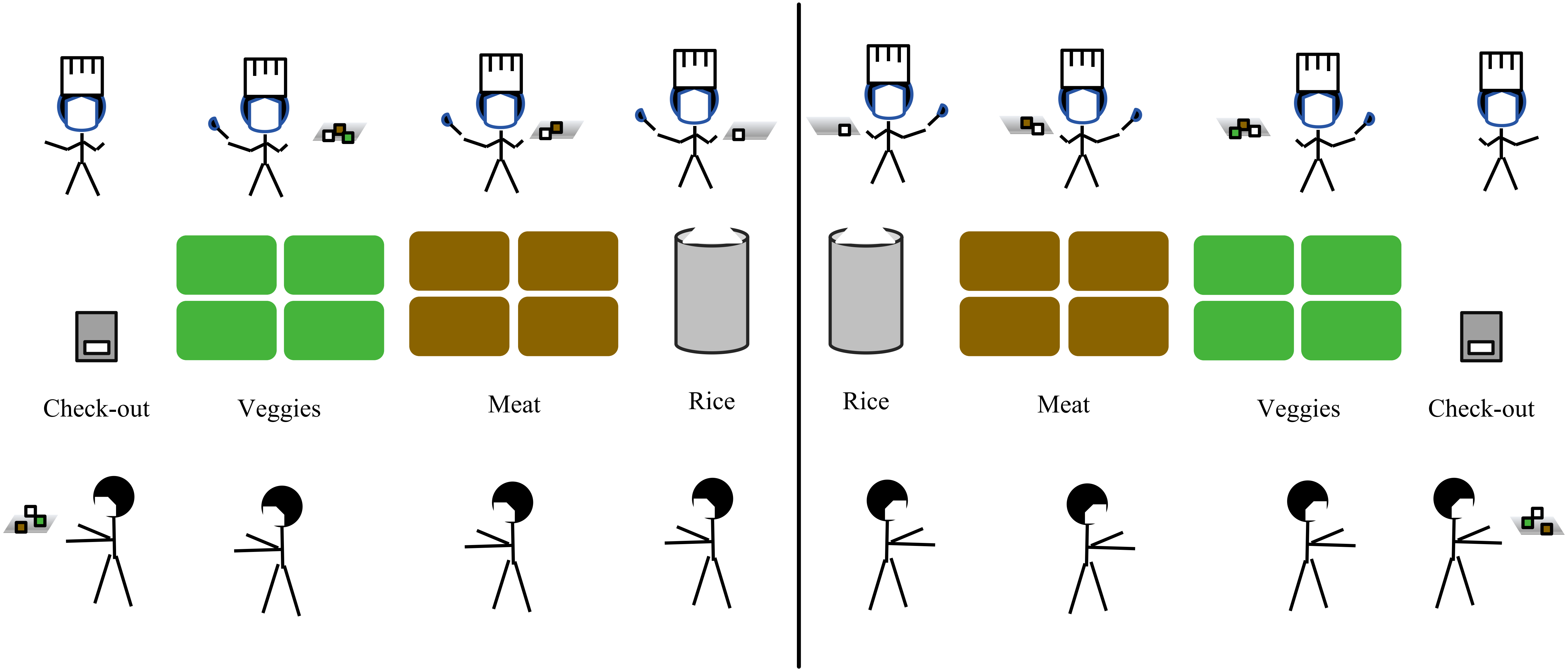
- Simplified 5-stage pipelined single-issue CPU datapath



- At most 1 instruction is “issued” to the datapath at 1 clock cycle

average CPI ≥ 1

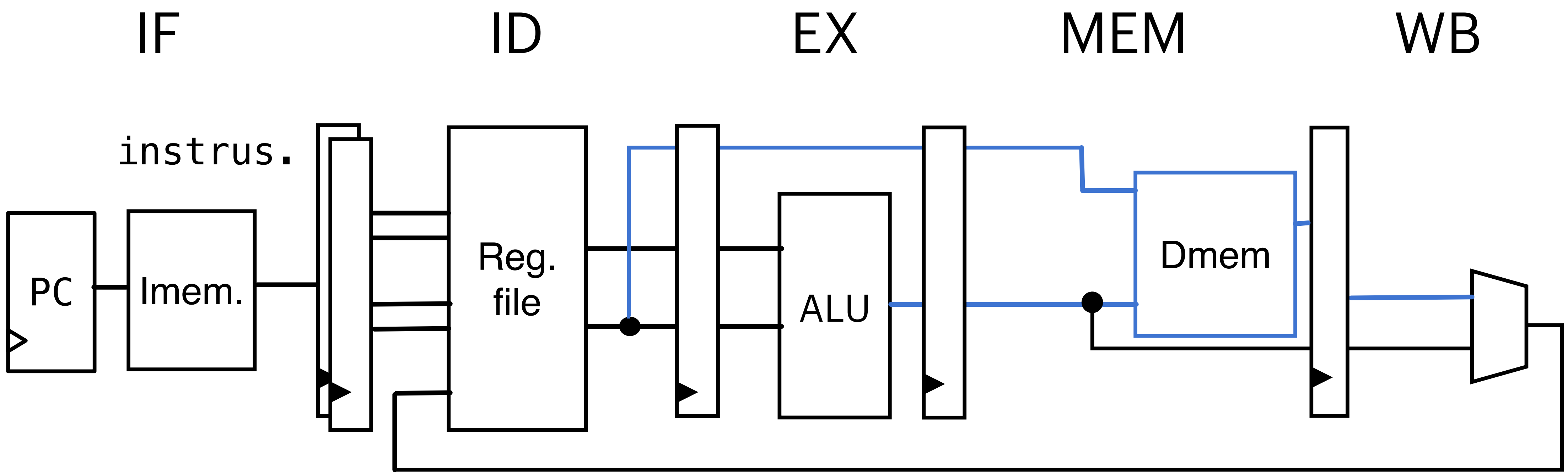
Multi-issue



Canteen analogy
(combined with pipelining)

Multi-issue (Hardware)

- Multi-issue CPU datapath

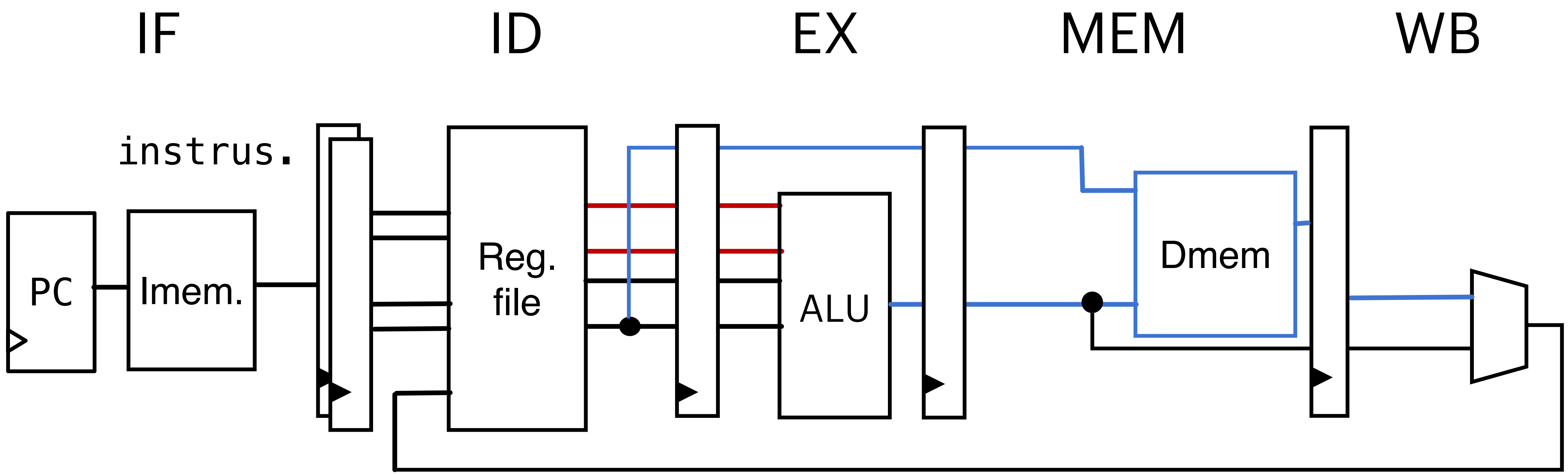


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI can be smaller than 1.

Multi-issue (Hardware)

- Multi-issue CPU datapath

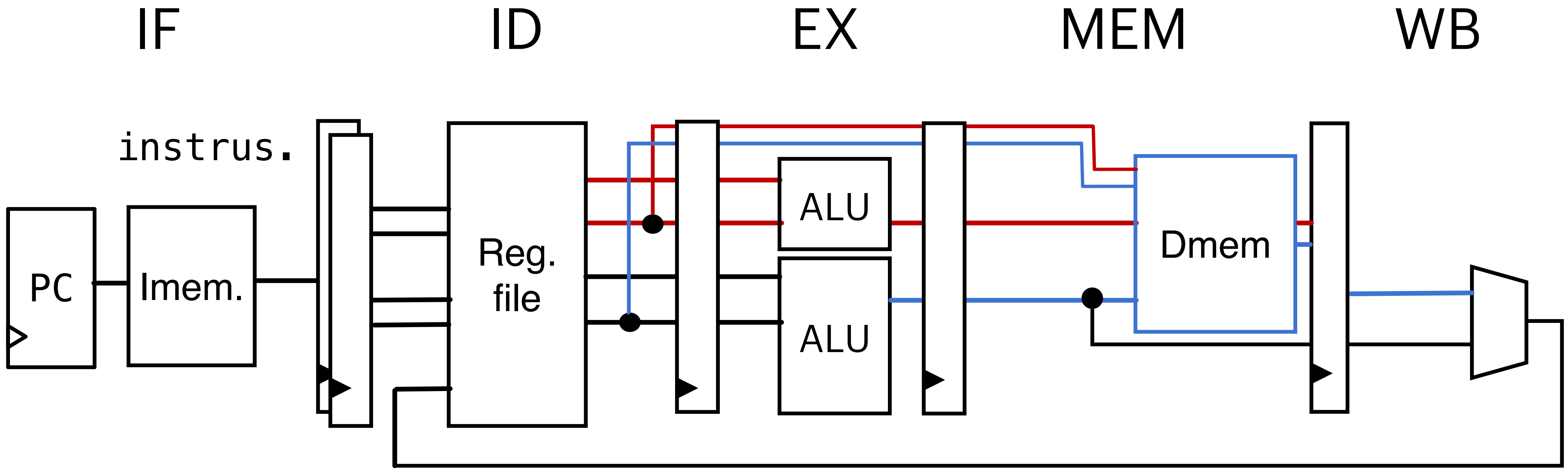


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI may be smaller than 1.

Multi-issue (Hardware)

- Multi-issue CPU datapath

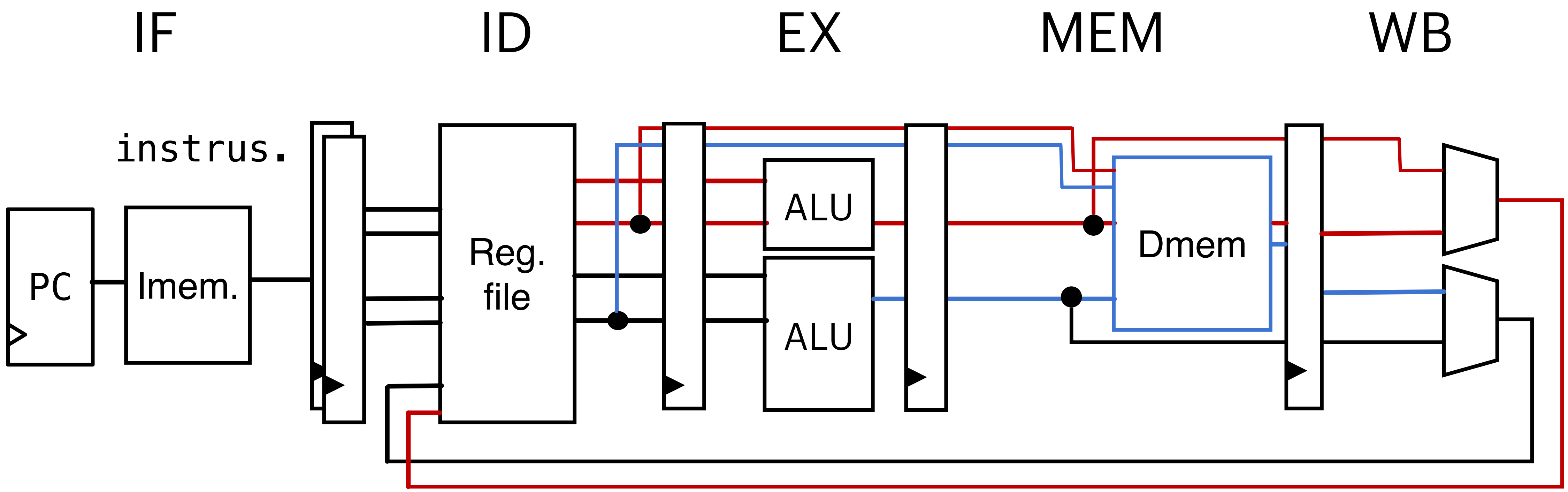


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI may be smaller than 1.

Multi-issue (Hardware)

- Multi-issue CPU datapath

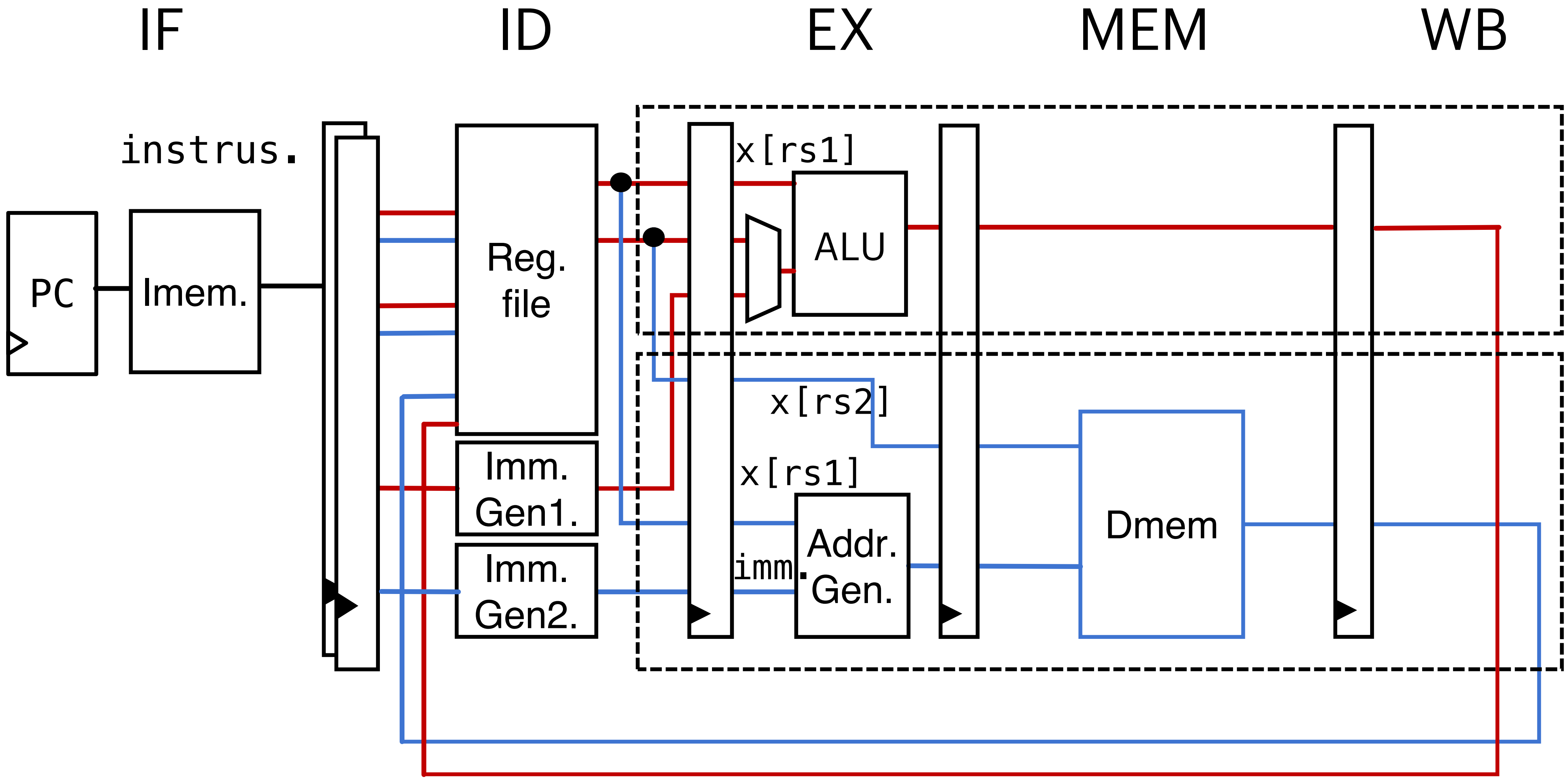


Hardware has to be adapted to avoid structural hazards

- Issue multiple instructions to the datapath in 1 clock cycle, average CPI may be smaller than 1.

Multi-issue (Hardware)

- In practice, we build different datapaths for different types of instructions
- E.g.

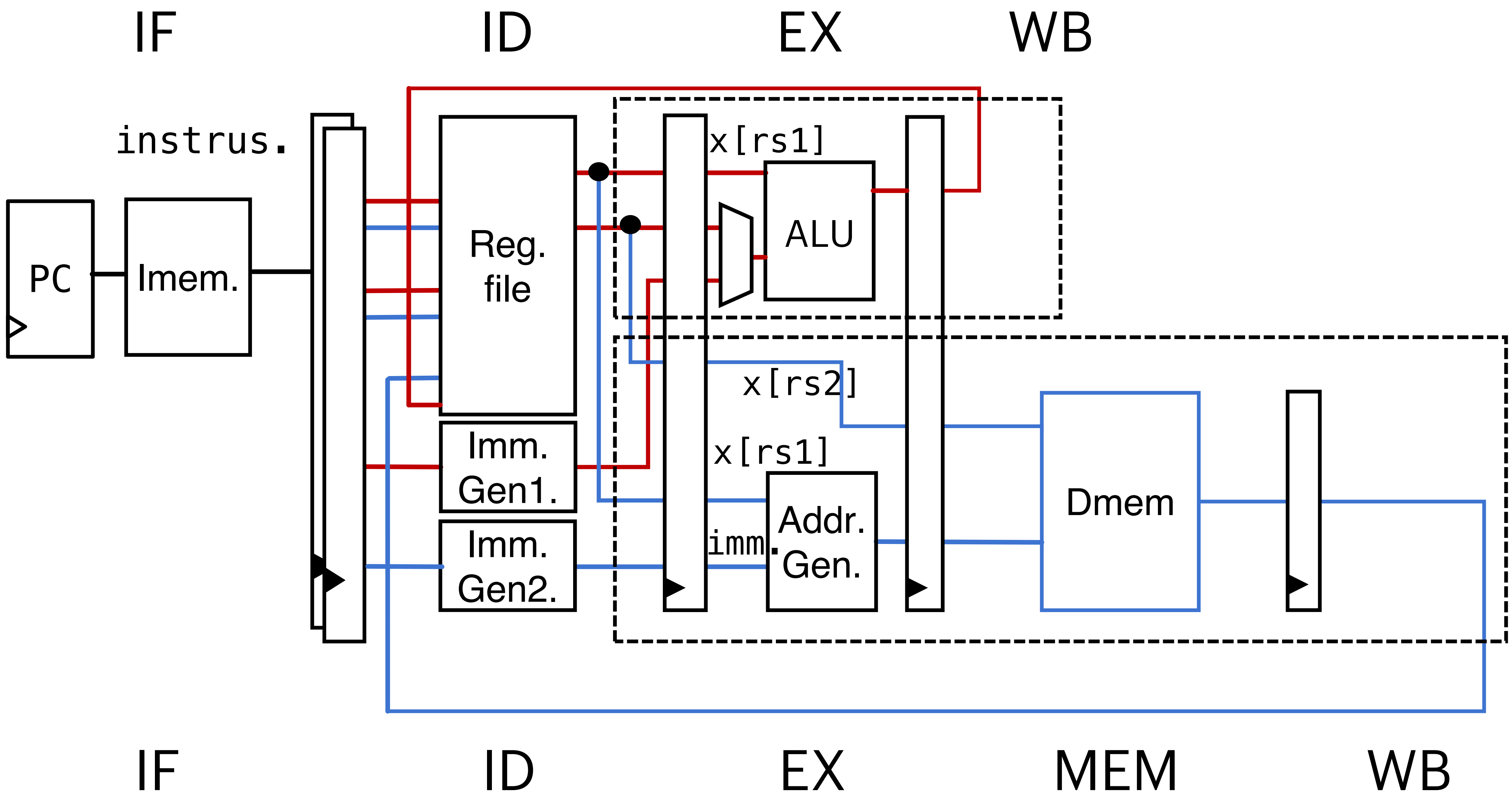


🟥 Arithmetic, logic and beq path

🟦 Memory access (load & store) path
A.K.A., load-store unit (LSU)

Multi-issue (Hardware)

- In practice, we build different datapaths for different types of instructions
- E.g.



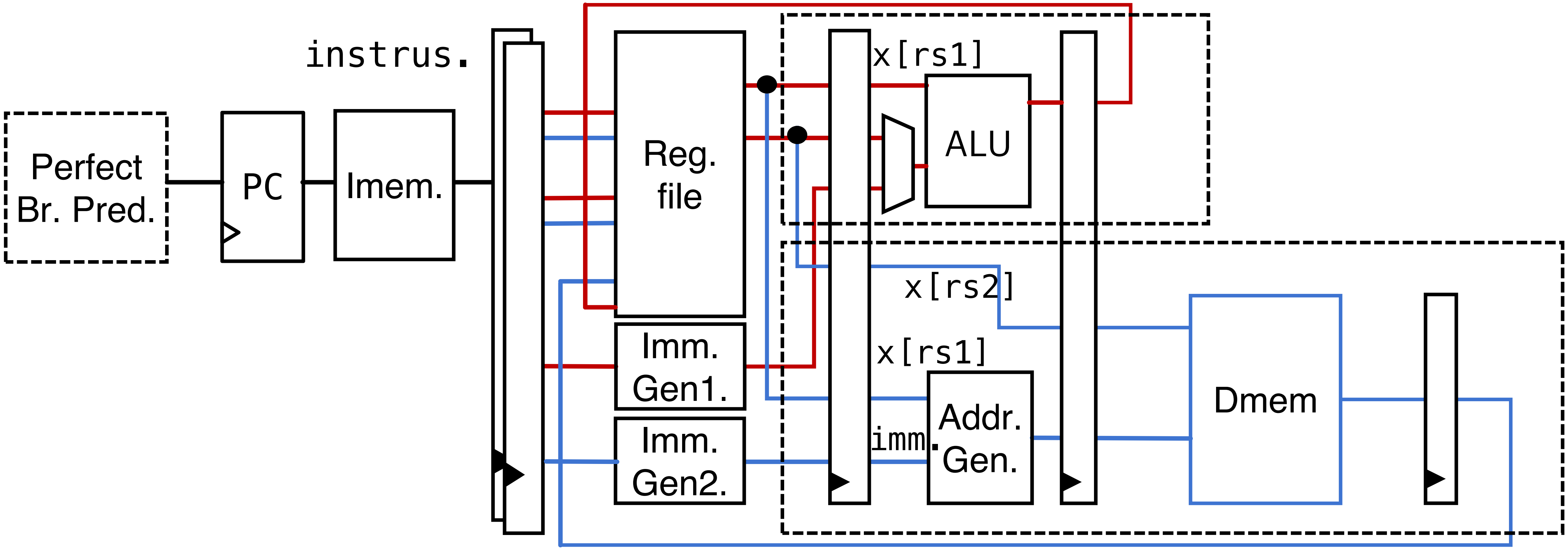
— Arithmetic, logic and beq path

— Memory access (load & store) path
A.K.A., load-store unit (LSU)

Multi-issue

- Ideally, issue two instructions with different type to ALU/mem datapaths

Instruciton type	cc1	cc2	cc3	cc4	cc5	cc6	cc7
ALU or branch	IF	ID	EX	WB			
Load or store	IF	ID	EX	MEM	WB		
ALU or branch		IF	ID	EX	WB		
Load or store		IF	ID	EX	MEM	WB	
ALU or branch			IF	ID	EX	WB	
Load or store			IF	ID	EX	MEM	WB



Multi-issue

```
for (int i=1000; i>0; i=i-1)
     x[i] = x[i] + s;
```

```
1。      addi s0,x0,s      #initialize s0
2.Loop:lw   t3,0(t1)      #load array element
3.      add  t3,t3,s0     #add s to $t3
4.      sw   t3,0(t1)     #store result
5.      addi t1,t1,-4     #t1 =t1-4
6.      bne  t1,t2,Loop   #repeat loop if t1!=t2
```

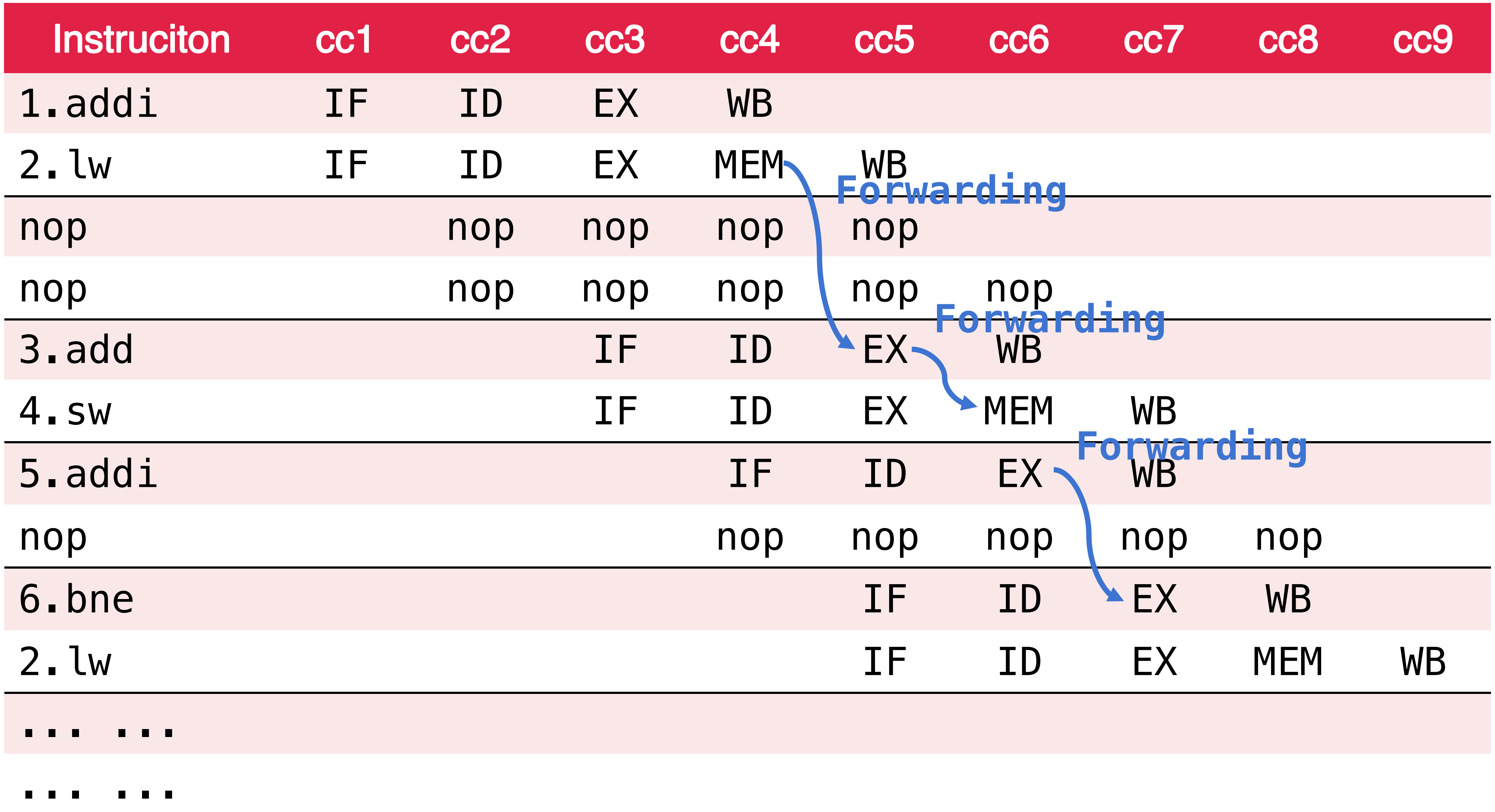
Instruciton	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9
1.addi	IF	ID	EX	WB					
2.lw	IF	ID	EX	MEM	WB				
3.add		IF	ID	EX	WB				
4.sw		IF	ID	EX	MEM	WB			
5.addi			IF	ID	EX	WB			
nop			nop	nop	nop	nop	nop		
6.bne			IF	ID	EX	WB			
2.lw			IF	ID	EX	MEM	WB		
3.add				IF	ID	EX	WB		
4.sw				IF	ID	EX	MEM	WB	
5.addi					IF	ID	EX	WB	
nop					nop	nop	nop	nop	nop

Hazards!

Multi-issue

```
for (int i=1000; i>0; i=i-1)
     x[i] = x[i] + s;
```

```
1.       addi s0,x0,1       #initialize s0
2.Loop:lw   t3,0(t1)       #load array element
3.       add  t3,t3,s0      #add s to $t3
4.       sw   t3,0(t1)      #store result
5.       addi t1,t1,-4      #t1 =t1-4
6.       bne  t1,t2,Loop    #repeat loop if t1!=t2
```



Instruciton	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9
1.addi	IF	ID	EX	WB					
2.lw	IF	ID	EX	MEM	WB				
nop		nop	nop	nop	nop				
nop		nop	nop	nop	nop	nop			
3.add			IF	ID	EX	WB			
4.sw			IF	ID	EX	MEM	WB		
5.addi				IF	ID	EX	WB		
nop				nop	nop	nop	nop	nop	
6.bne					IF	ID	EX	WB	
2.lw					IF	ID	EX	MEM	WB
... ...									
... ...									

Multi-issue

- Loop unrolling & register renaming to optimize (also will be used in SIMD)

```
1.   addi    s0,x0,s
2.L:lw       t3,0(t1)
3.   lw      t4,-4(t1)
4.   lw      t5,-8(t1)
5.   lw      t6,-12(t1)
6.   add     t3,t3,s0
7.   add     t4,t4,s0
8.   add     t5,t5,s0
9.   add     t6,t6,s0
10.  sw      t3,0(t1)
11.  sw      t4,-4(t1)
12.  sw      t5,-8(t1)
13.  sw      t6,-12(t1)
14.  addi    t1,t1,-16
15.  bne     t1,t2,L
```

Instruciton	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9
1.addi	IF	ID	EX	WB					
2.lw t3	IF	ID	EX	MEM	WB				
nop		nop	nop	nop	nop				
3.lw t4		IF	ID	EX	MEM	WB			
6.add t3			IF	ID	EX	WB			
4.lw t5			IF	ID	EX	MEM	WB		
7.add t4				IF	ID	EX	WB		
5.lw t6				IF	ID	EX	MEM	WB	
8.add t5					IF	ID	EX	WB	
10.sw t3					IF	ID	EX	MEM	WB
9.add t6						IF	ID	EX	WB
11.sw t4						IF	ID	EX	MEM
14.addi							IF	ID	EX
12.sw t5							IF	ID	EX
15.bne								IF	ID
13.sw t6								IF	ID

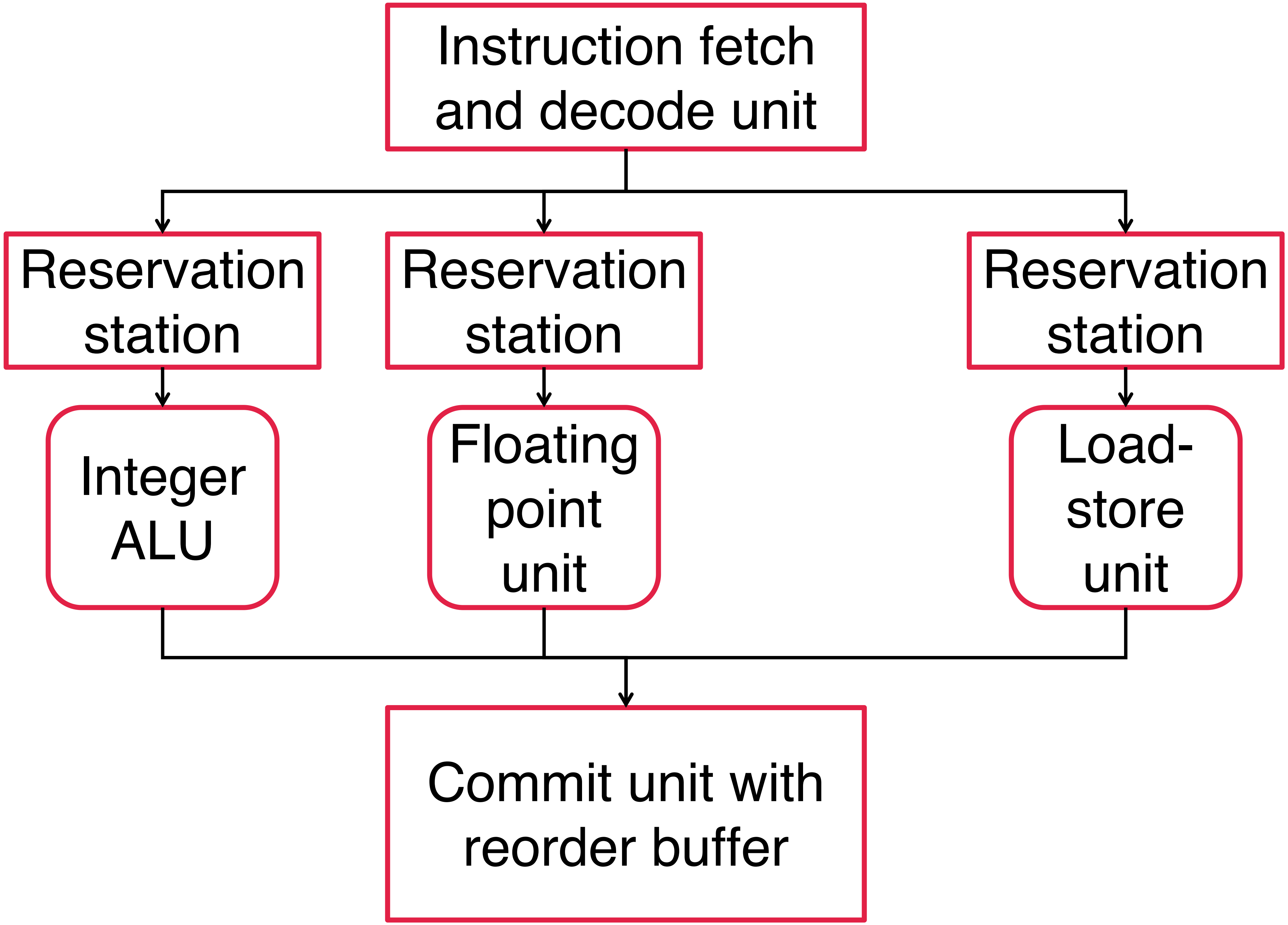
Static vs. Dynamic multi-issue

- Static multi-issue
  - Package instructions into issue slots and detect hazards statically (at compile time mostly)
  - Hardware may also detect/resolve hazards
  - Also called VLIW (very long instruction word)

- Dynamic multi-issue
  - Package instructions into issue slots and detect hazards dynamically (during execution by hardware mostly)
  - Compiler may also help avoiding hazards
  - Also called superscalar

Instruciton type	cc1	cc2	cc3	cc4	cc5	cc6	cc7
`ALU or branch`	IF	ID	EX	WB			
`Load or store`	IF	ID	EX	MEM	WB		
`ALU or branch`		IF	ID	EX	WB		
`Load or store`		IF	ID	EX	MEM	WB	
`ALU or branch`			IF	ID	EX	WB	
`Load or store`			IF	ID	EX	MEM	WB

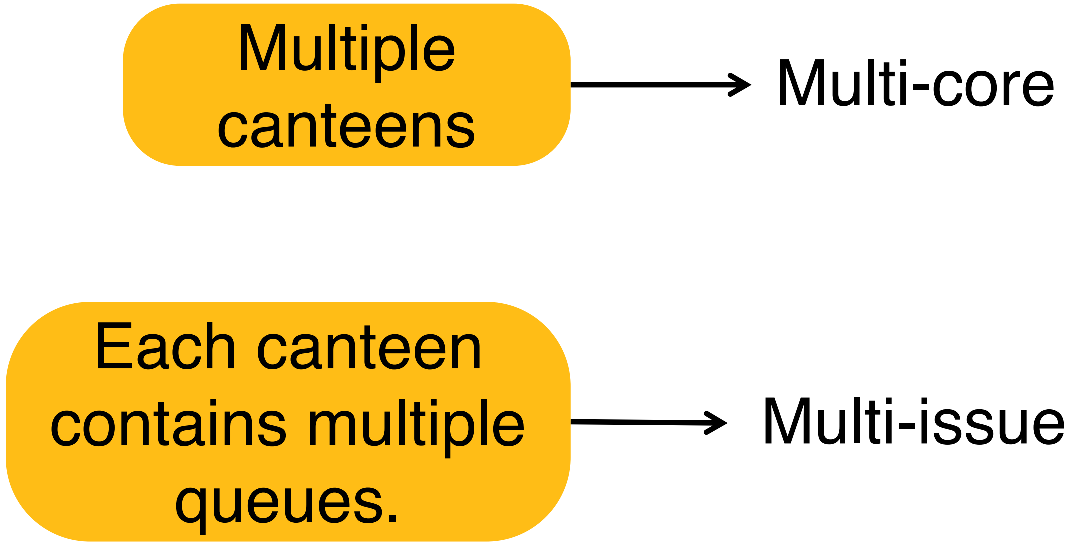
Hardware implementation of superscalar



More in CS211 CA II

Multi-issue Pitfalls

- Multi-issue is not multi-core;
- Multi-issue is not SIMD;
- Multi-issue can be combined with pipelining, SIMD, multi/hyper-threading, etc. to improve the performance of the processor;



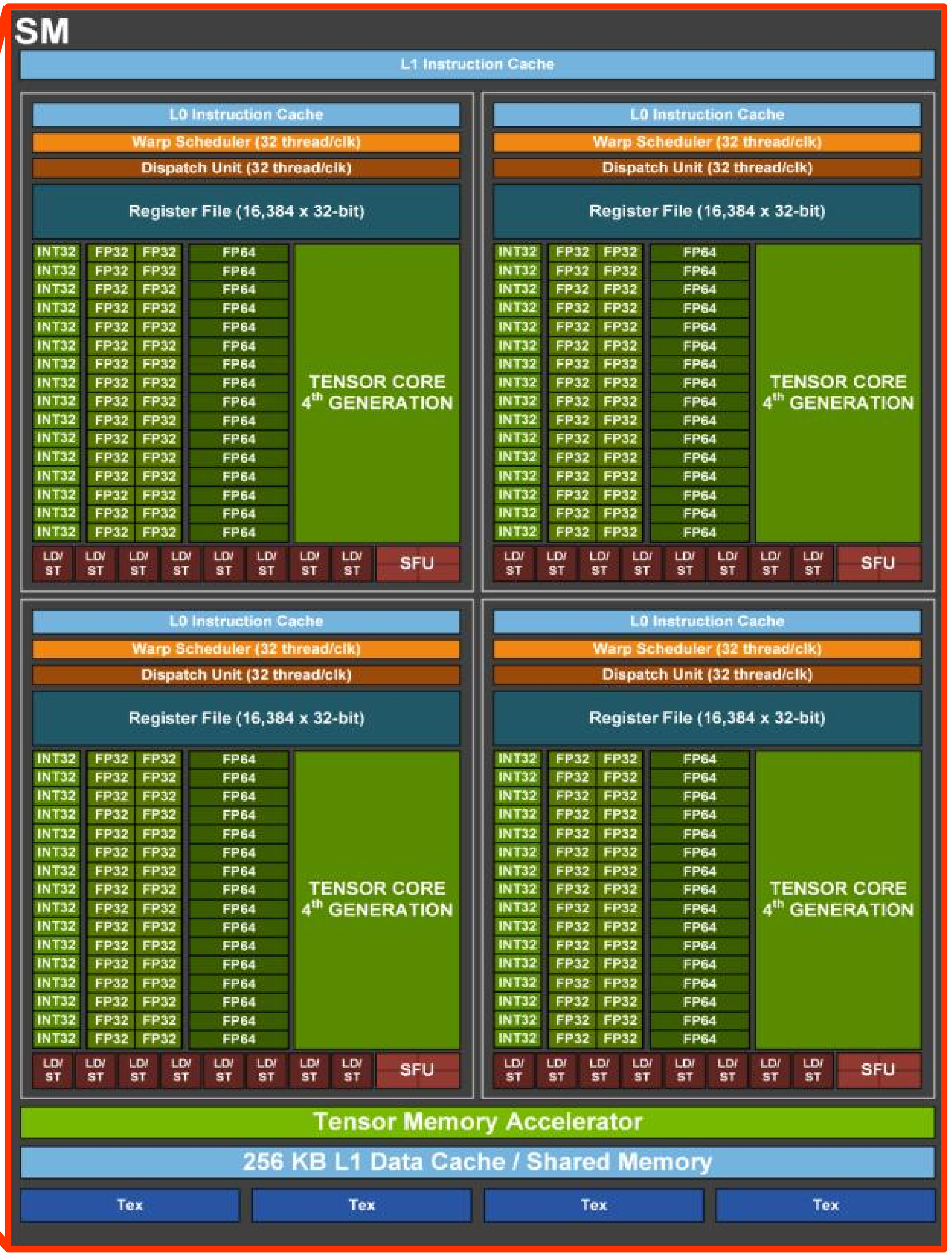
Multiple canteens → Multi-core

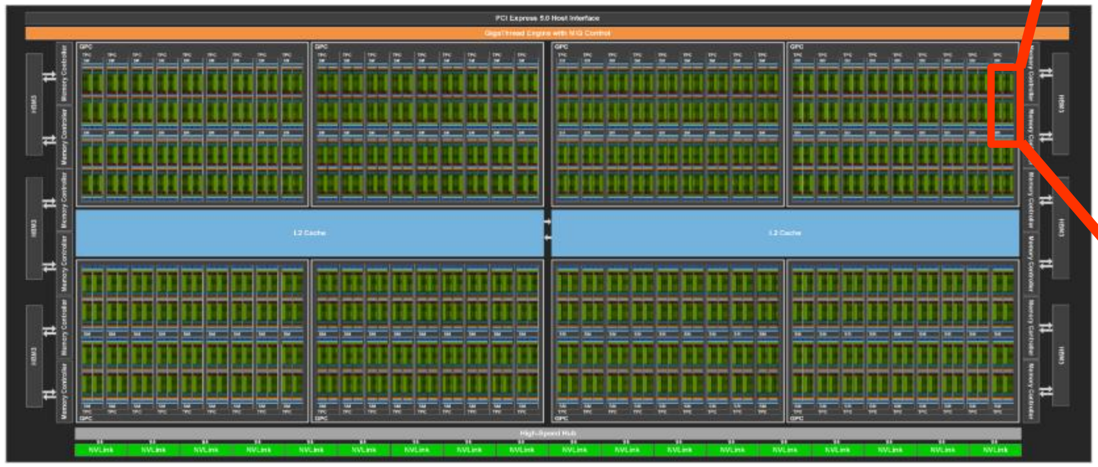
Each canteen contains multiple queues. → Multi-issue

Multi-issue Advancements-GPU

- Multi-issue can be combined with SIMD;
- In Fermi and later NVIDIA GPU architectures, the scheduler issues
  - 2 INT instructions or 2 single-point FP instructions or
  - 1 mixed INT or FPU or load or store or SFU instructions

One streaming multi-processor inside an H100 GPU


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

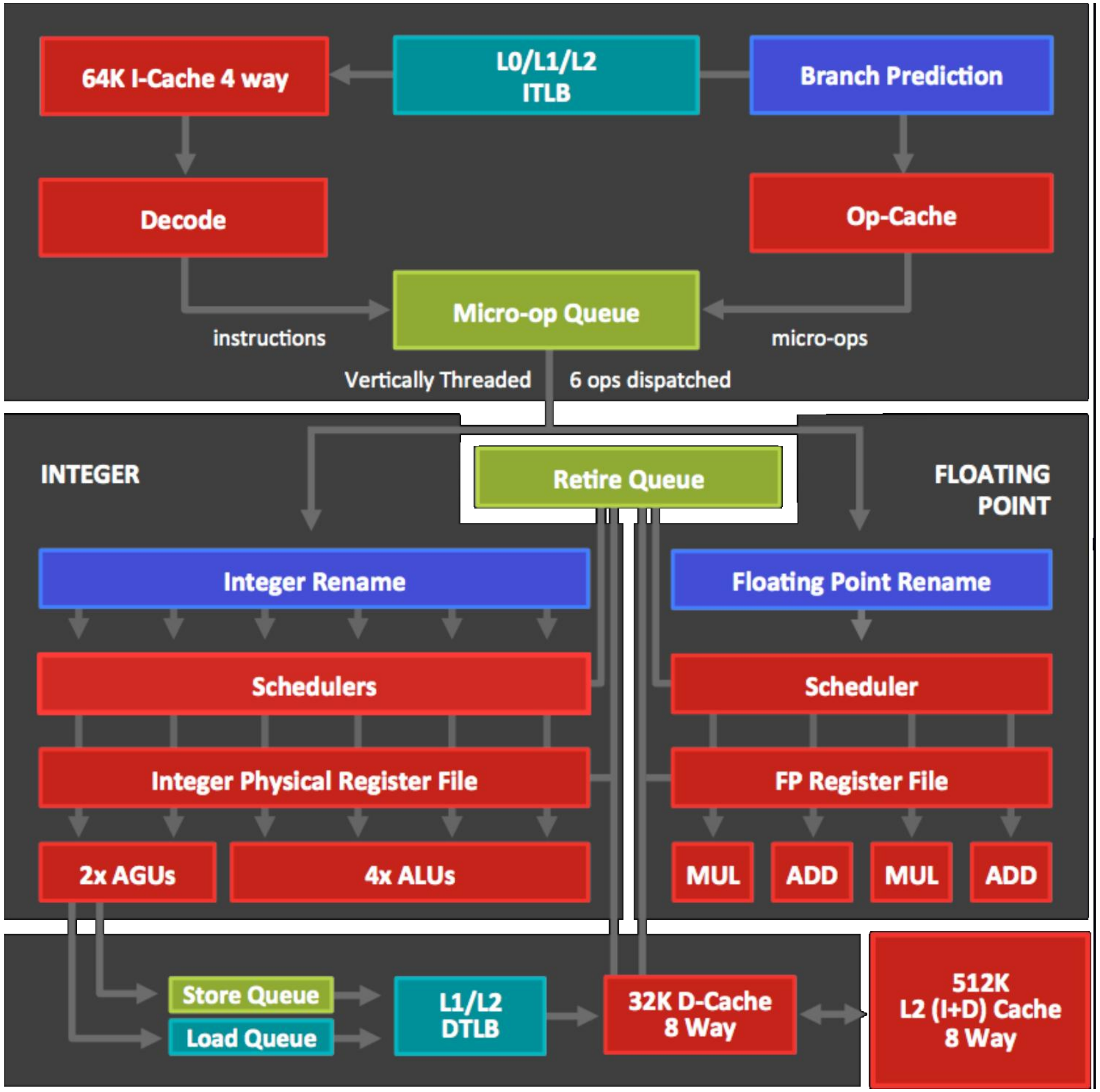


More in EE219 AI computing systems

Credit to Nvidia

Multi-issue Advancements-CPU

- Multi-issue can be combined with multi/hyper-threading;
- Thread-level parallelism (TLP)
  - To tolerate latency (e.g., cache miss)
  - To further improve throughput
  - To reduce context switch penalty
- Types of Multithreading
  - Fine-grained: threads scheduled cycle by cycle
  - Corase-grained: threads scheduled on events (e.g., cache misses)/time quantum
  - SMT: simultaneous multi-threading



Credit to AMD

AMD CPU (Zen architecture) that supports simultaneous multithreading

More in CS211 CA II

Summary

- **Instruction-level parallelism**
  - Pipeline
    - Insert pipeline registers to execute the instructions stage by stage;
    - Multiple instructions co-exist in the pipeline to realize parallelism;
    - Induce (structural/data/control) hazards;
    - Strategies to deal with the hazards (insertion of bubbles, forwarding, hardware re-design, code scheduling, branch prediction, etc.);
  - Multi-issue
    - Multiple datapaths to execute multiple instructions in parallel;
    - Need to consider hazards as well;
    - Static vs. Dyanmic multi-issue