信息科学与技术学院
School of Information Science and Technology

CS 110
Computer Architecture
Pipeline II

Instructors:
Siting Liu & Chundong Wang
Course website: https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2024/index.html
School of Information Science and Technology (SIST)
ShanghaiTech University

2024/4/18

Administratives

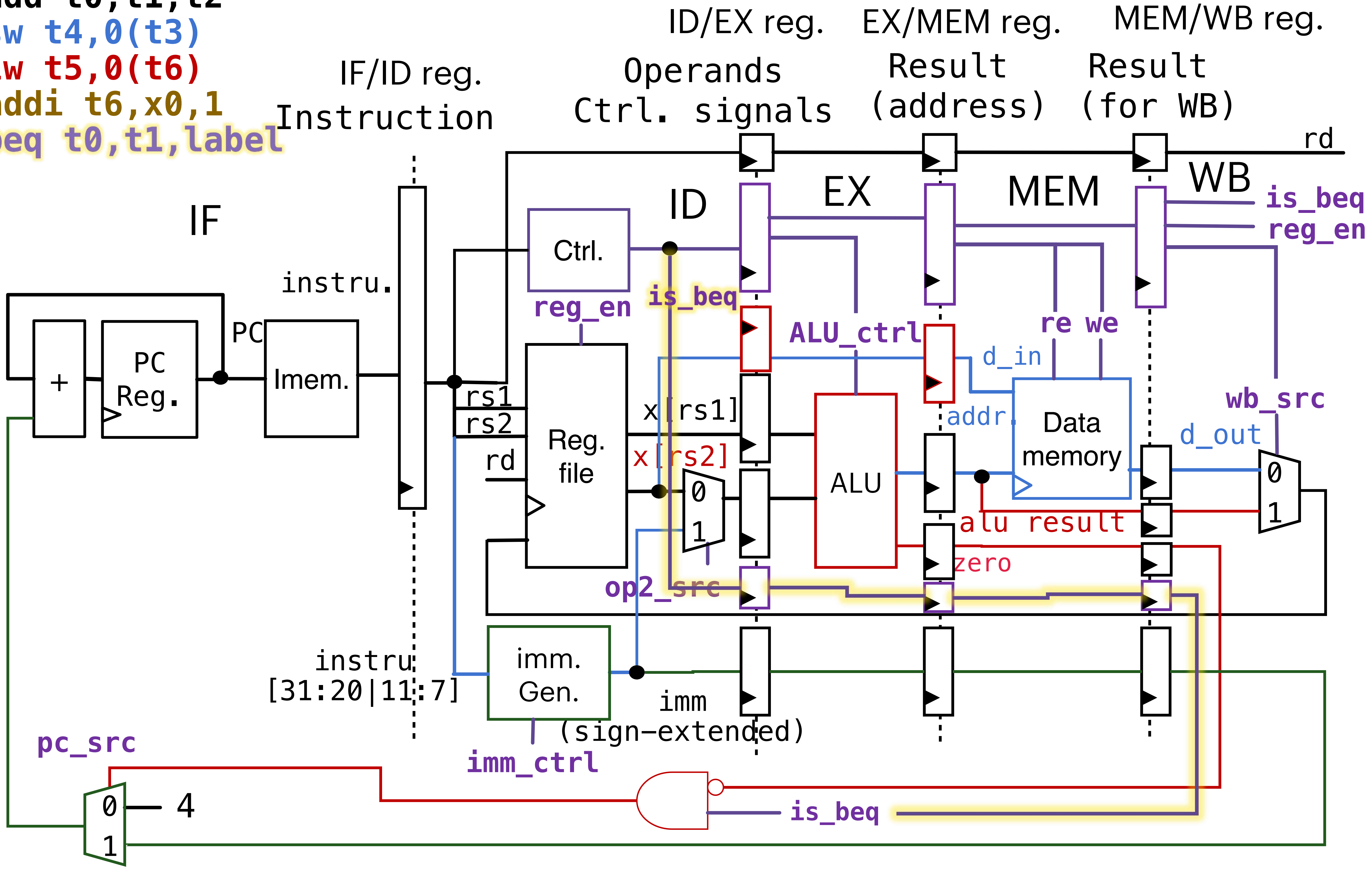
- No Lab this week, instead, we check Project 1.1 this week at the lab sessions. Lab 8 will be released. It is about pipeline.
- HW 4 ddl April 29th
- Proj 1.2 ddl April 25th
- Proj 2.1 ddl May 7th (near the second mid-term, tentatively May 9th 8am-10am, so start early)
- Discussion (teaching center 301) schedule
  - Next discussion is about pipeline
  - The same content for Friday and the next Monday.

Outline

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

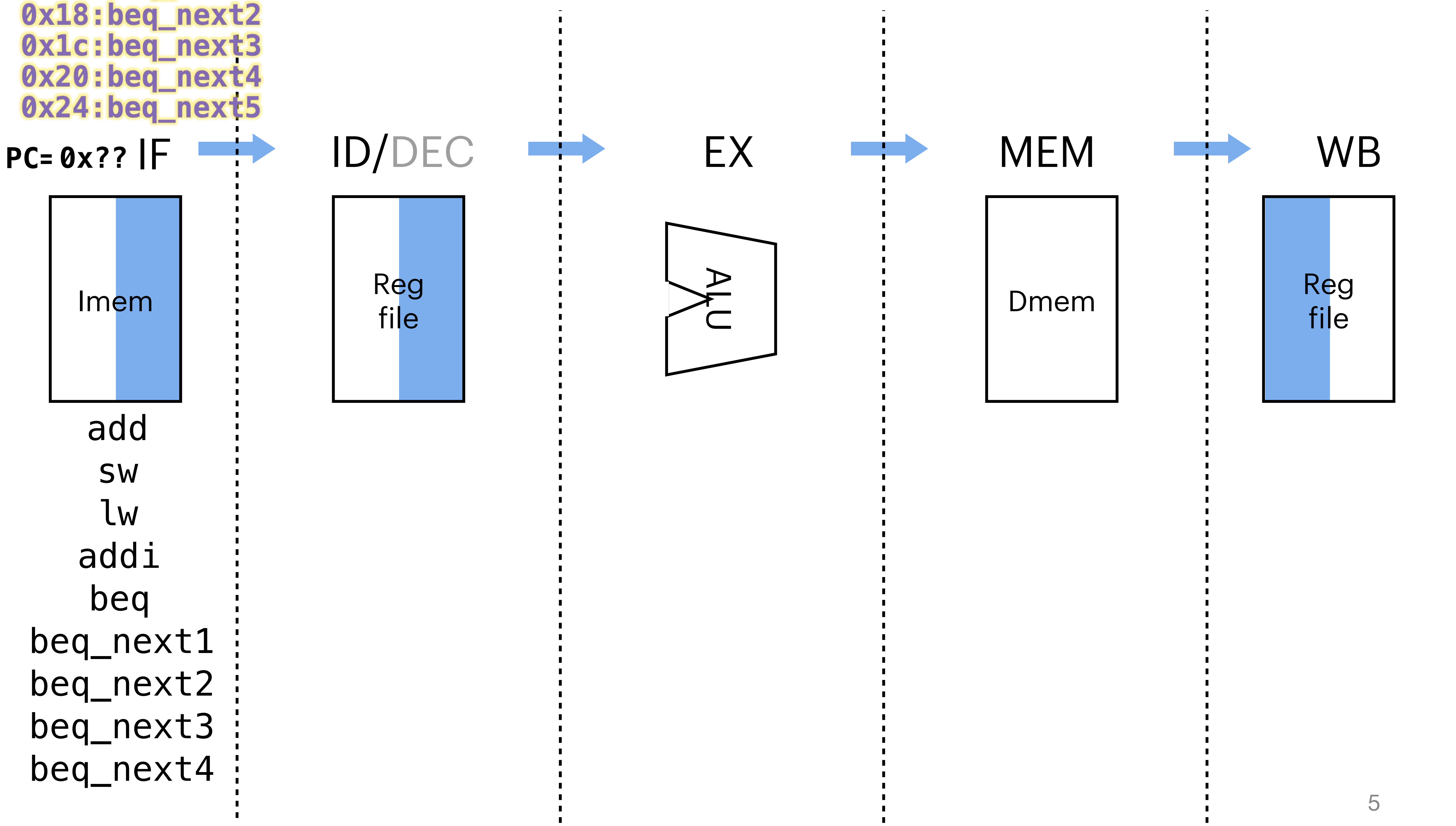
Detailed considerations

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



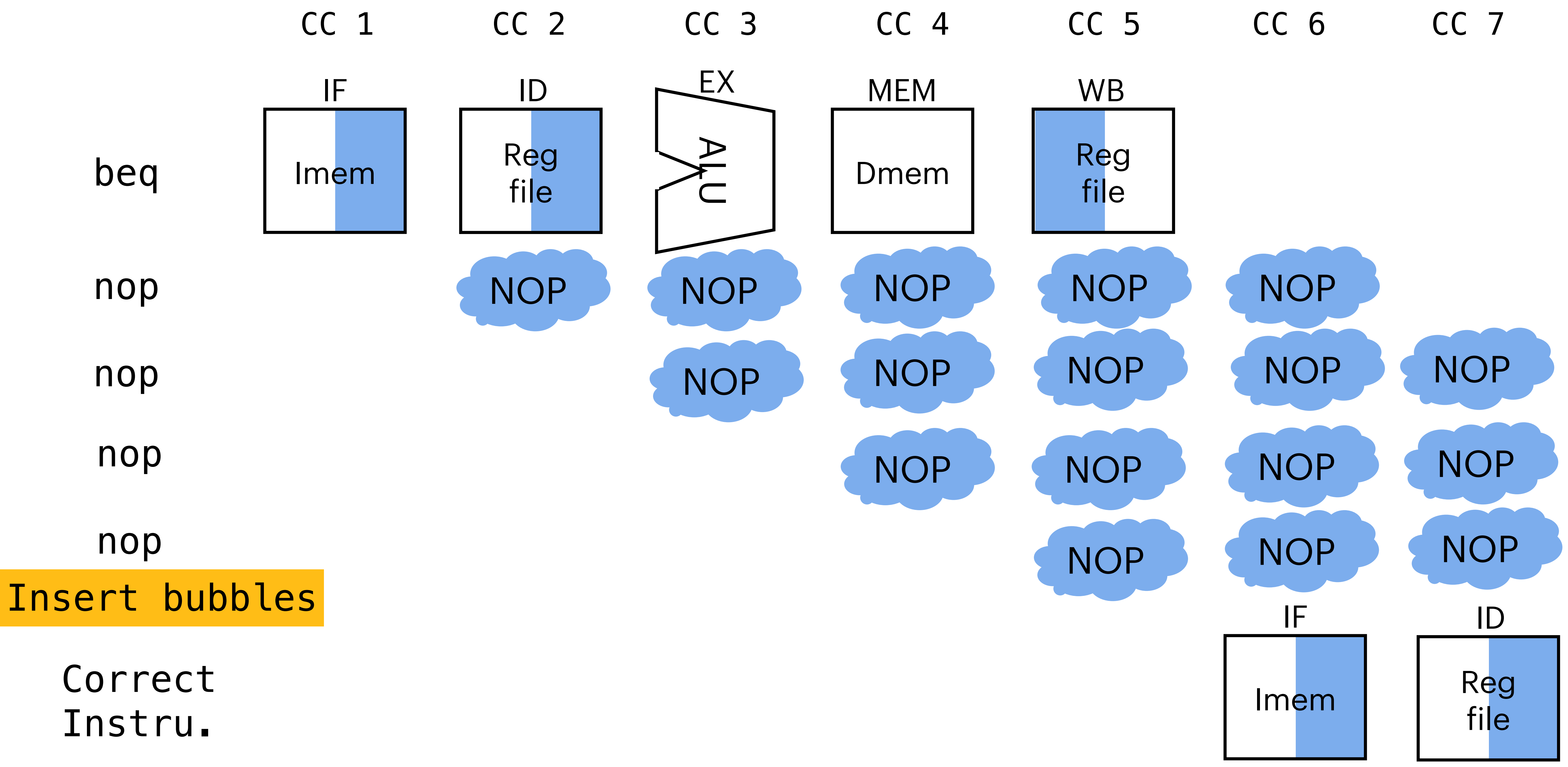
Detailed considerations

```
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
```



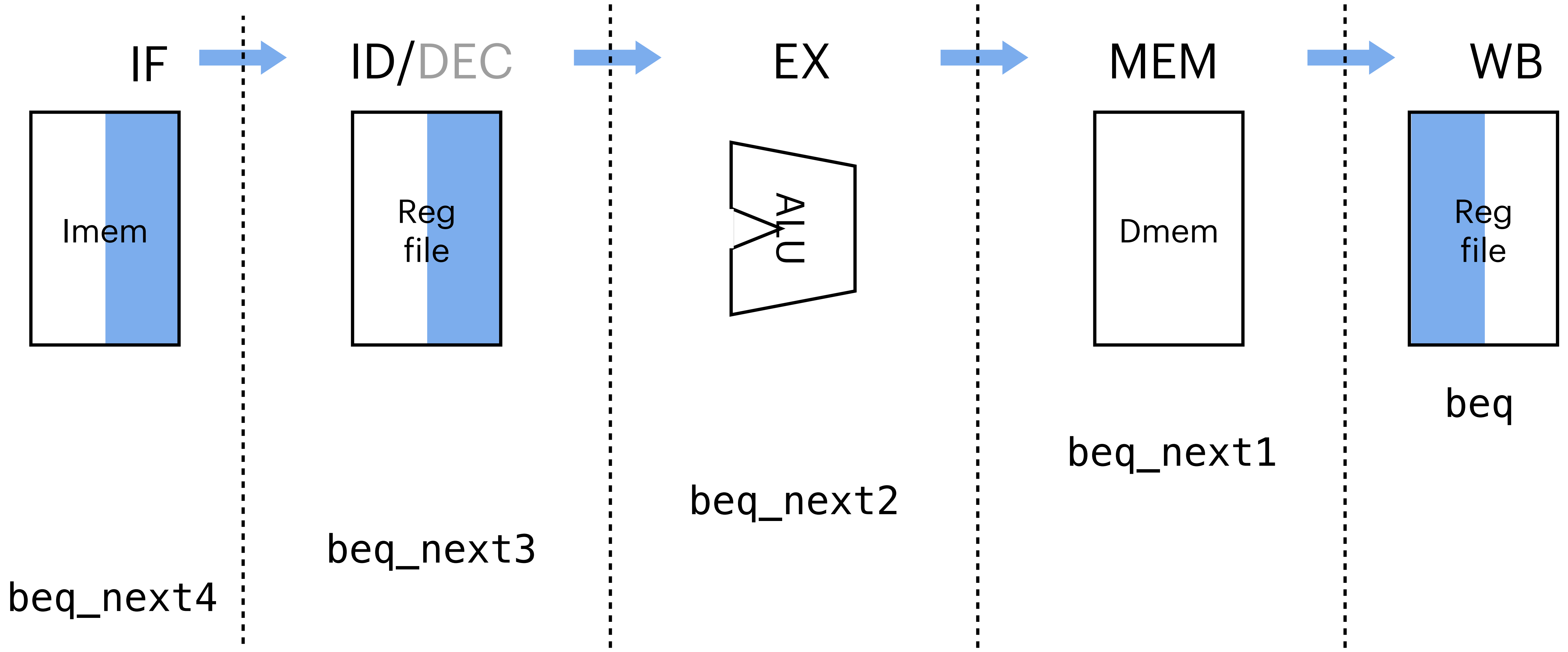
Control hazards--solution 1

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

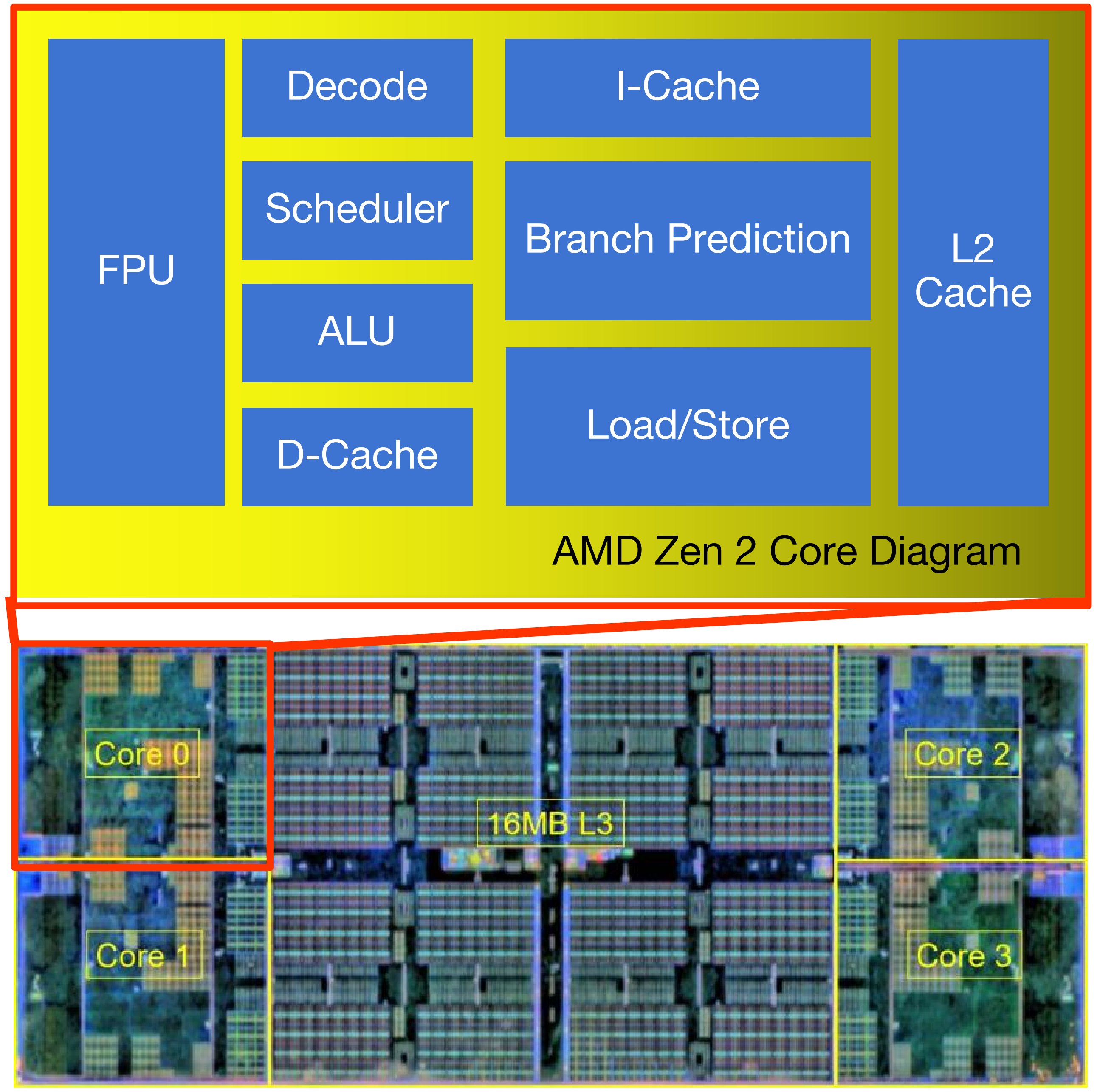
- Alternatively, dynamic branch prediction (when the program is running)

```
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



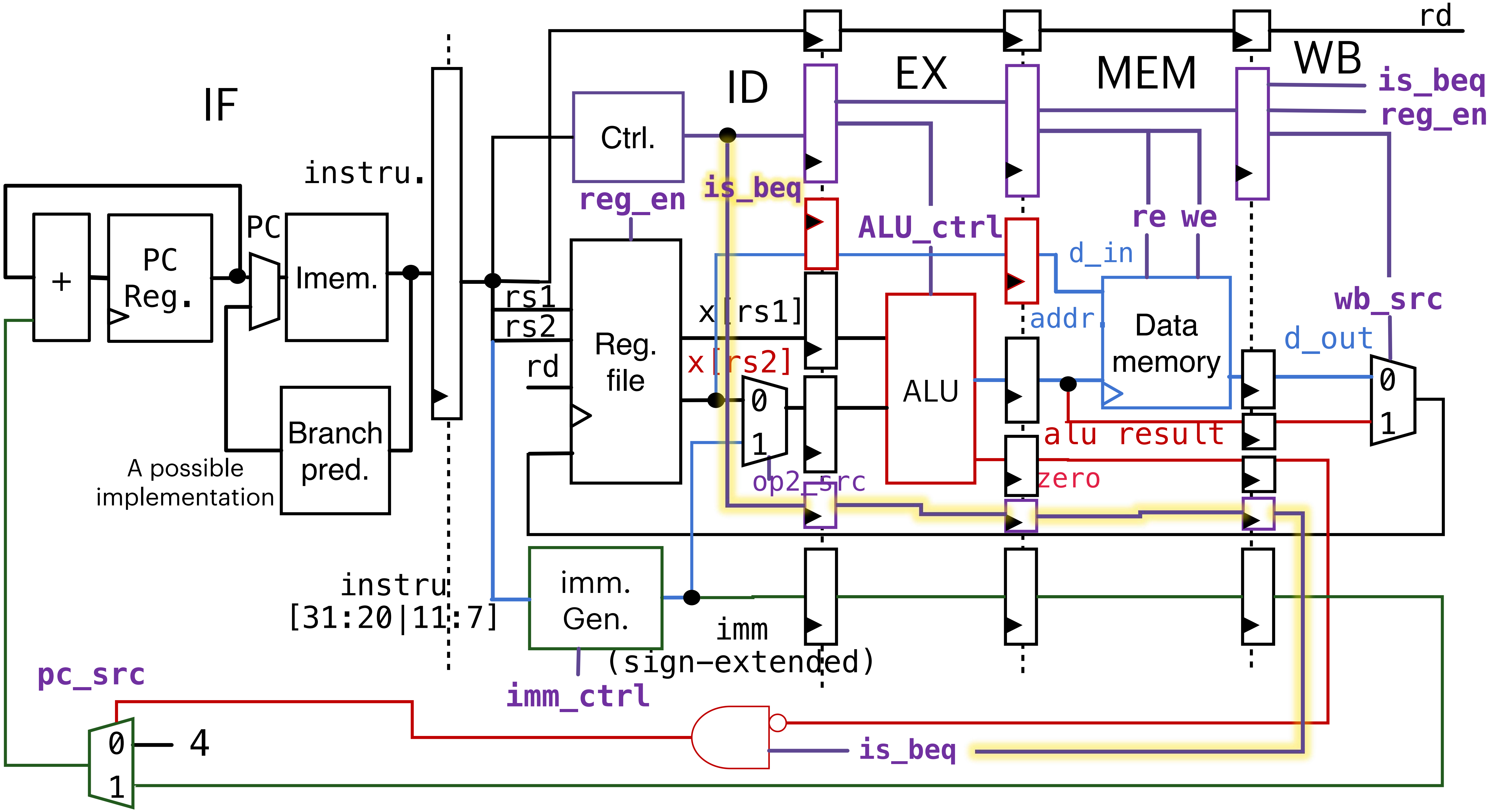
Unit	Zen	Zen 2
Floating Point	128b	256b
L0 Branch Target Buffer	8 entries	16 entries
L1 Branch Target Buffer	256 entries	512 entries
L2 Branch Target Buffer	4K entries	7K entries
Op Cache	2K ops	4K ops
Integer Physical Register File	168 entries	180 entries
Integer Scheduler	84 entries	92 entries
AGEN	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.

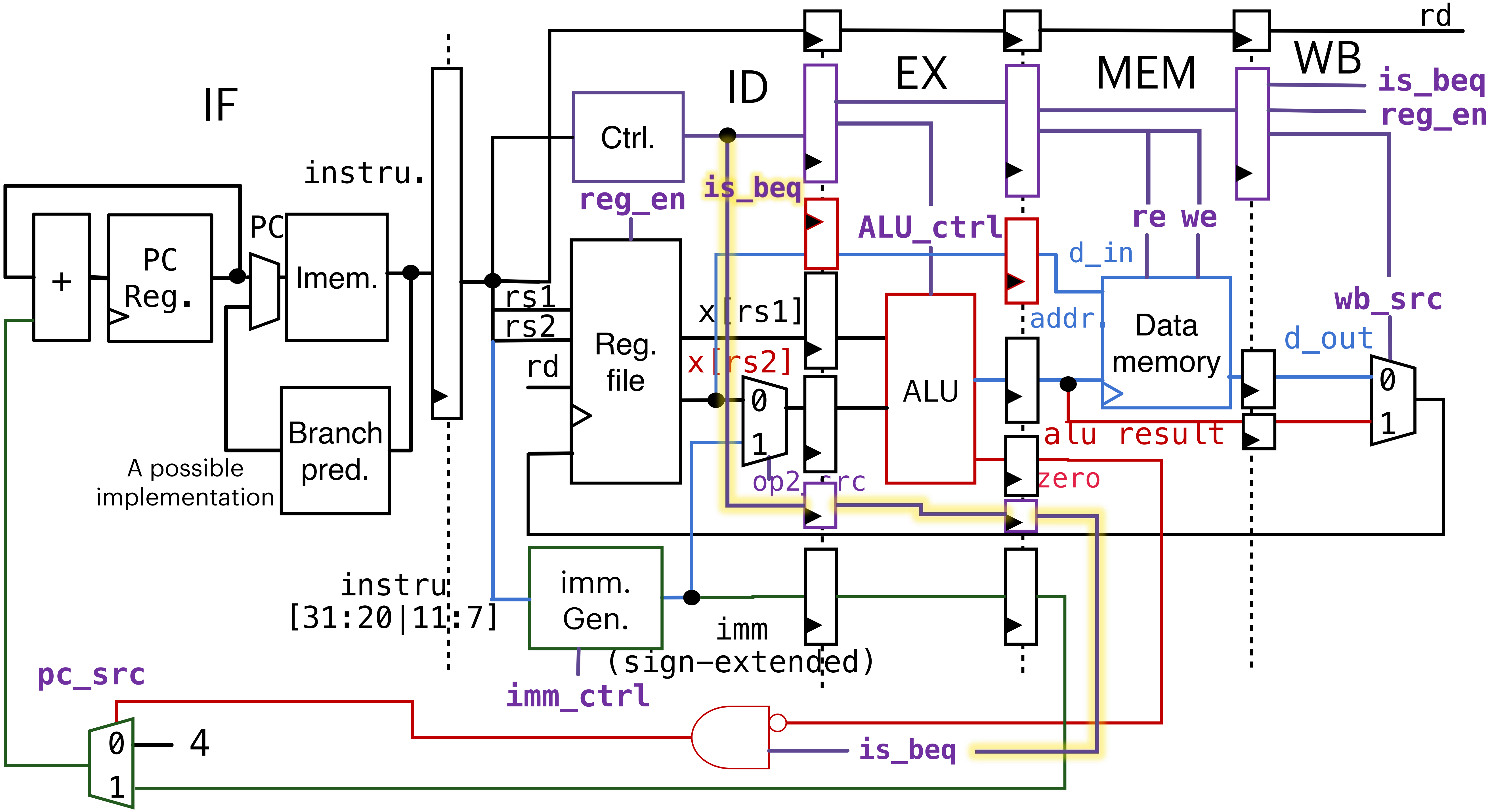
Control hazards -- solution 3

- 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

More Parallelism

- Instruction-level parallelism
  - Pipeline: multiple instructions co-exist in the pipeline
  - Static multi-issue (during compile): VLIW
  - Dynamic multi-issue (during execution): superscalar