

# Lab 5 preview

Hung-Wei Tseng

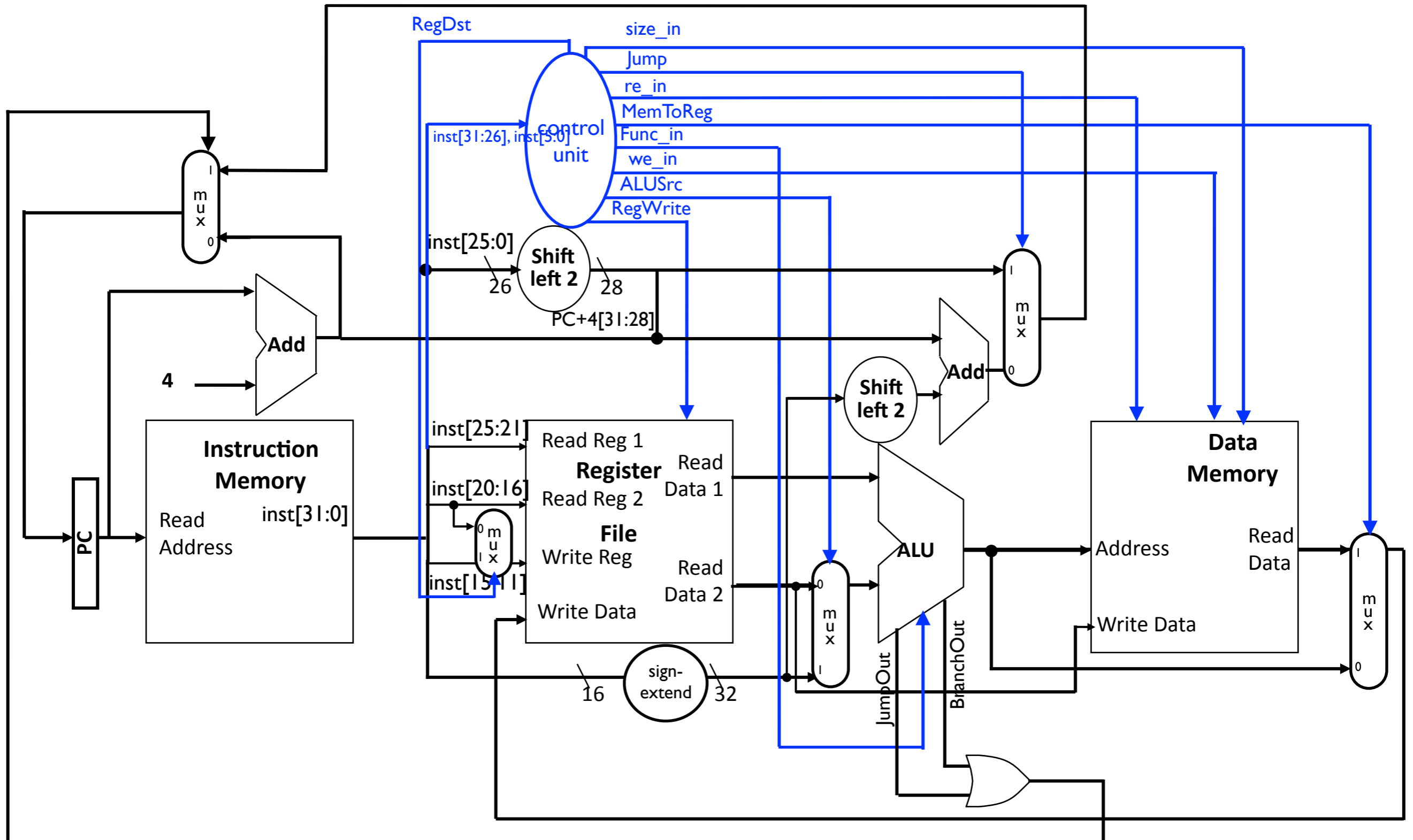
# Announcement

- Lab 4 due Thursday before 5:30pm!
  - Interview with any of us
- Lab 5 & 6 due next Thursday
  - No extension
- Come and have some pizza together next Friday @ 6p!

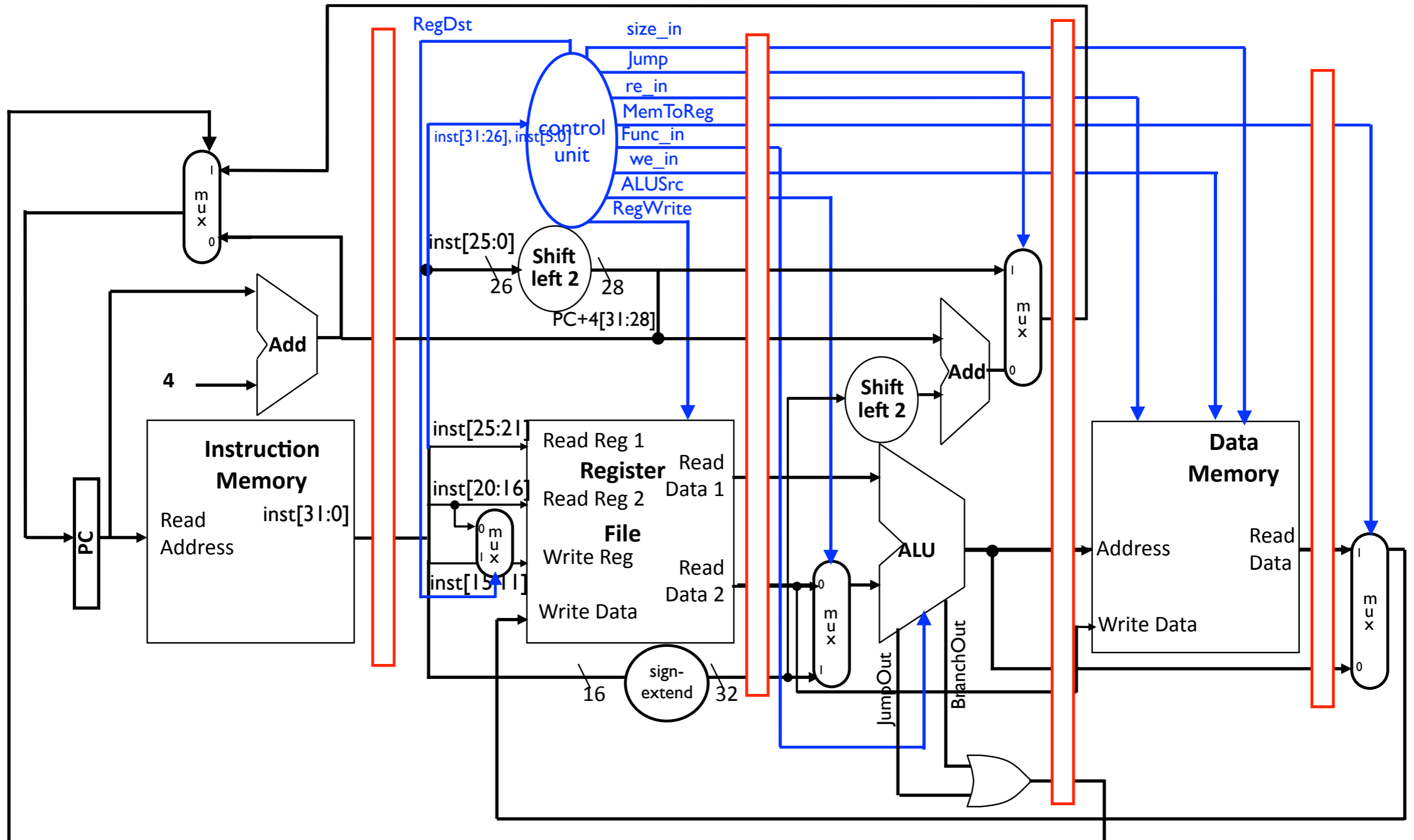
# In Lab 5...

- Pipeline your processor
  - Teach it “walk”, and teach it “fly”
  - A working pipeline processor is better than a crappy 5-stage
    - Your processor does not have to be 5-stage
  - Please complete the lab as soon as possible

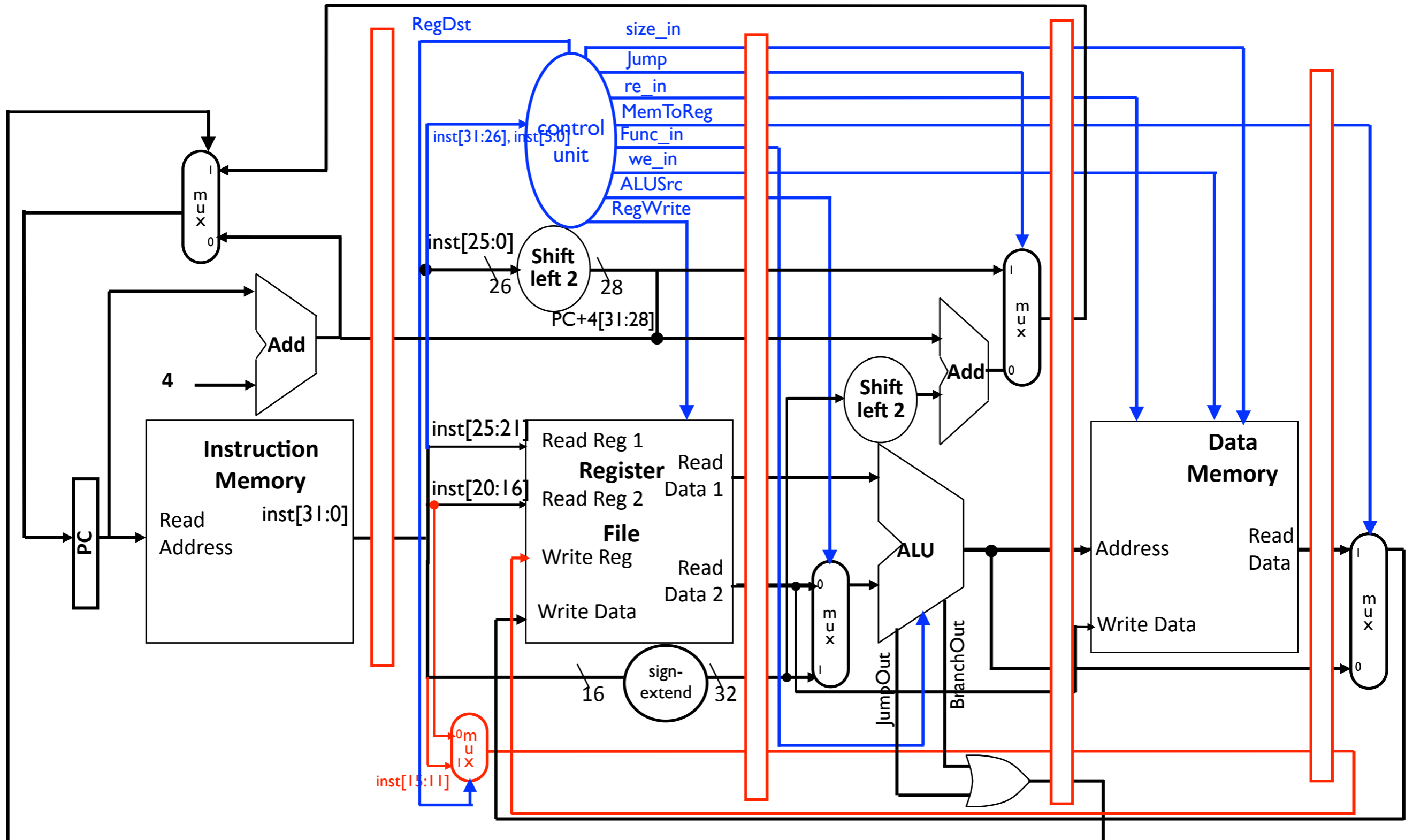
# In lab4, you already have...



# In lab5, we are going to pipeline it!

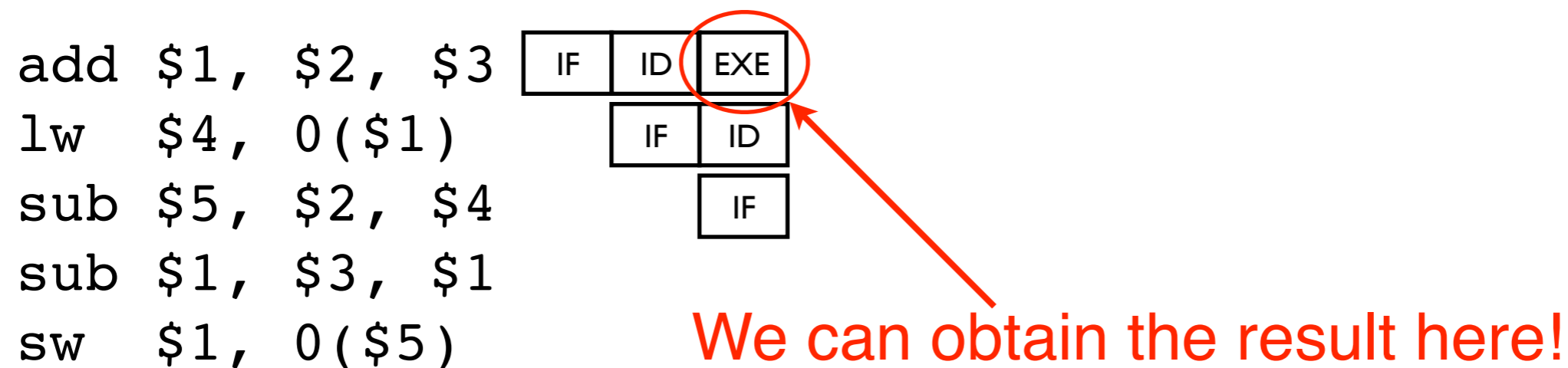


# It's not just adding pipeline registers!



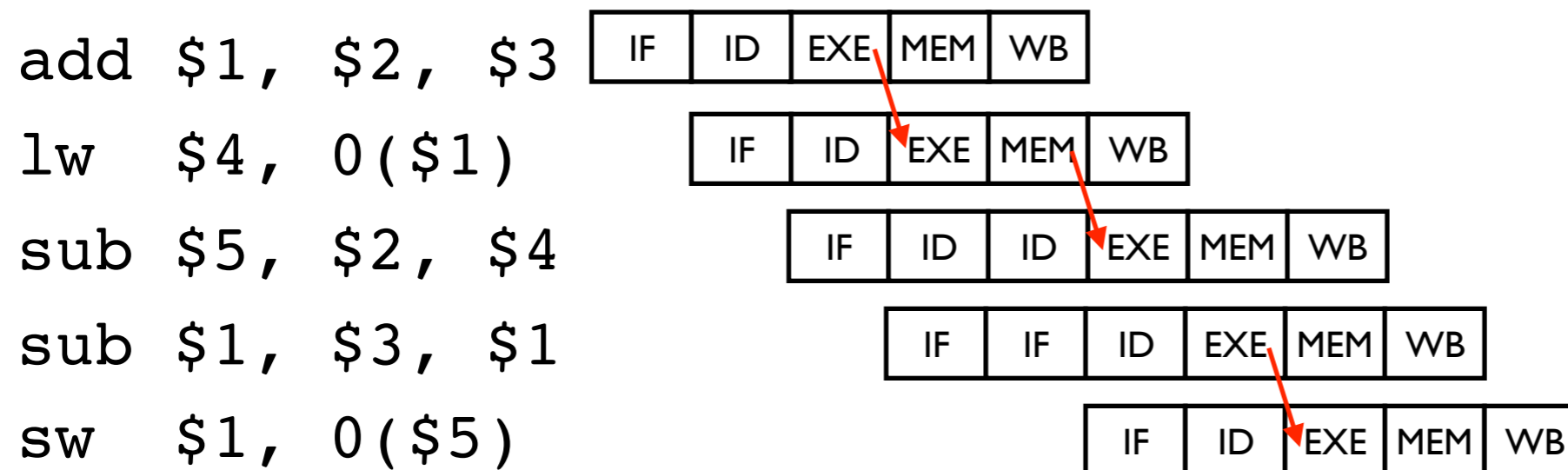
# Sol. of data hazard II: Forwarding

- The result is available after EXE and MEM stage, but publicized in WB!
- The data is already there, we should use it right away!
- Also called bypassing



# Sol. of data hazard II: Forwarding

- Take the values, where ever they are!



10 cycles! CPI == 2 (Not optimal, but much better!)



# Design a forwarding unit

- How many of the following inputs are required for forwarding the result from the previous instruction (Ins#1) to the EXE stage of the current instruction (Ins#2)?
  - Rd of Ins#2
  - Rs of Ins#2
  - Rt of Ins#2
  - ReadData 2 of Ins #2
  - Rd of Ins#1
  - Rs of Ins#1
  - Rt of Ins#1
  - ReadData 2 of Ins #1
  - Control signals of Ins #1

A. 5

B. 6

C. 7

D. 8

E. 9



We need to know the following:

1. If the ins#1 update a register (RegWrite)

2. If the destination register of ins #1 (rt, rd) is a source of ins #2

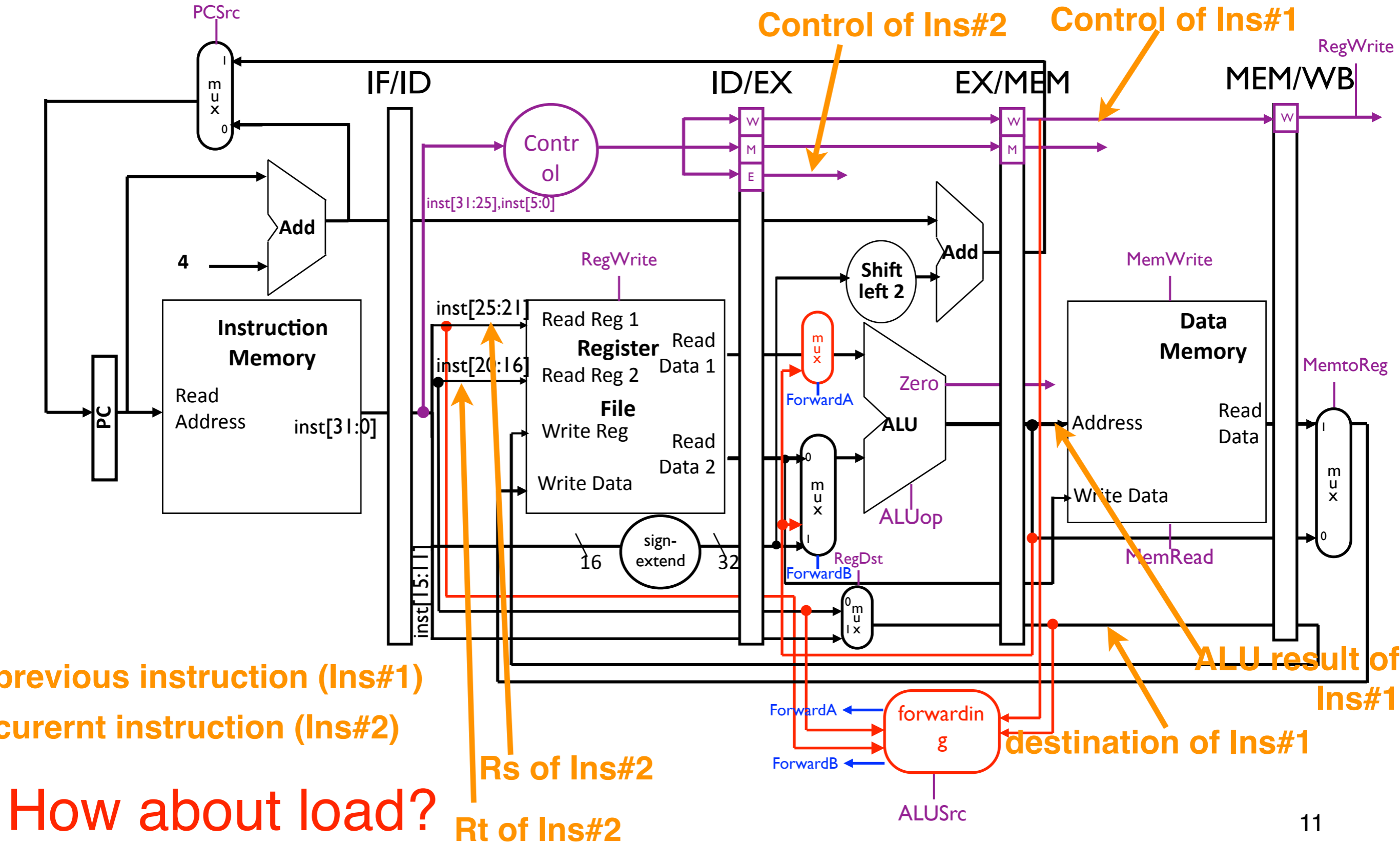
If ins #1 is R-type: rs, rt of ins #2 == rd of ins #1

If ins #1 is I-type: rs, rt of ins #2 == rt of ins #1

# When can/should we forward data?

- If the instruction entering the EXE stage consumes a result from a previous instruction that is entering MEM stage or WB stage
  - A source of the instruction entering EXE stage is the destination of an instruction entering MEM/WB stage
  - The previous instruction must be an instruction that updates register file

# Forwarding in hardware



previous instruction (Ins#1)

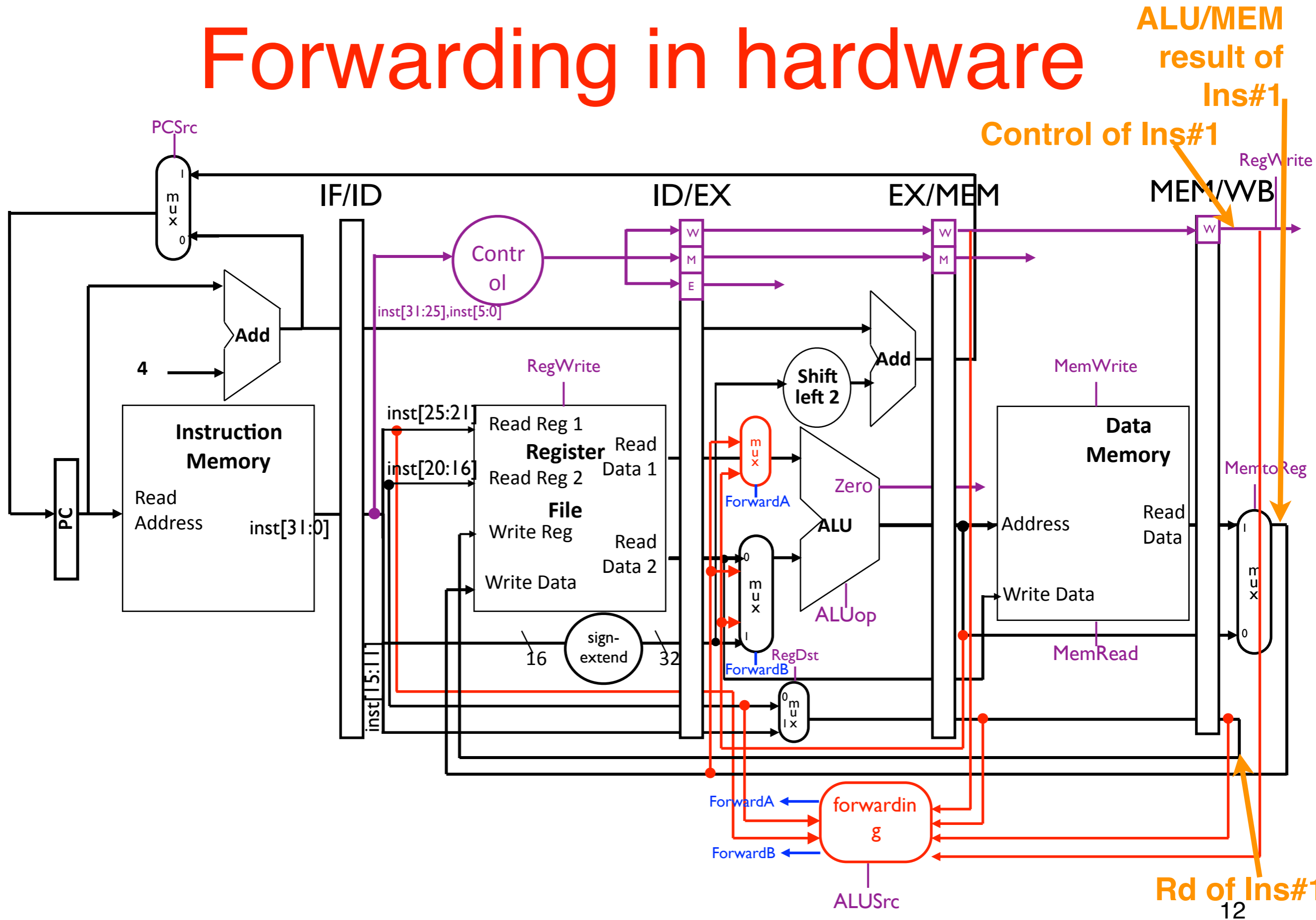
current instruction (Ins#2)

How about load?

Rs of Ins#2

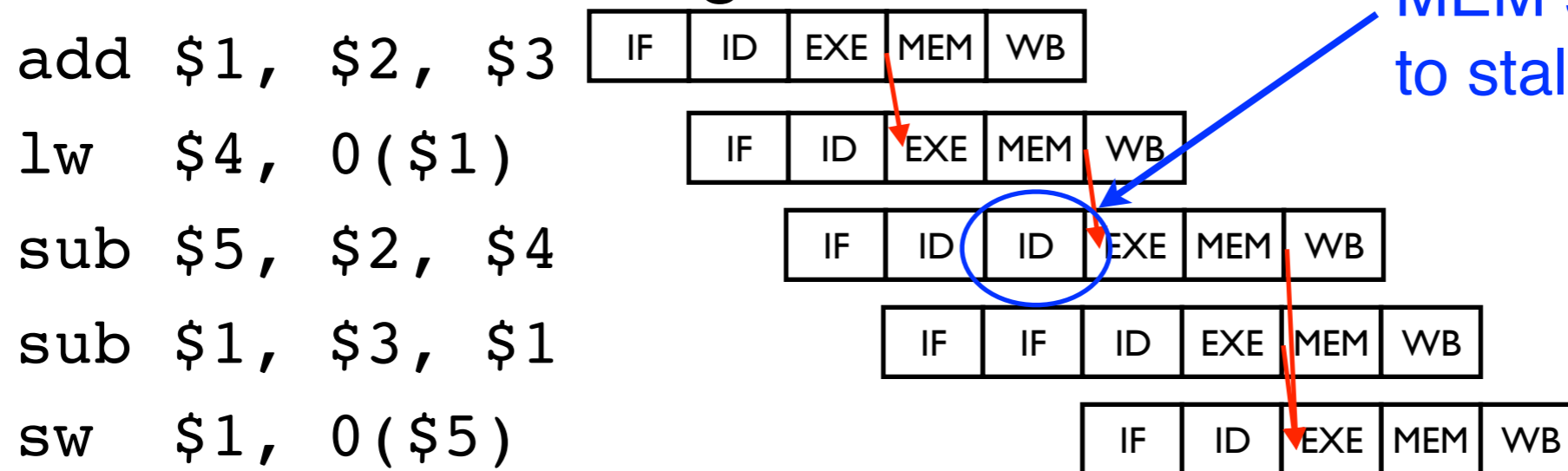
Rt of Ins#2

# Forwarding in hardware



# There is still a case that we have to stall...

- Revisit the following code:



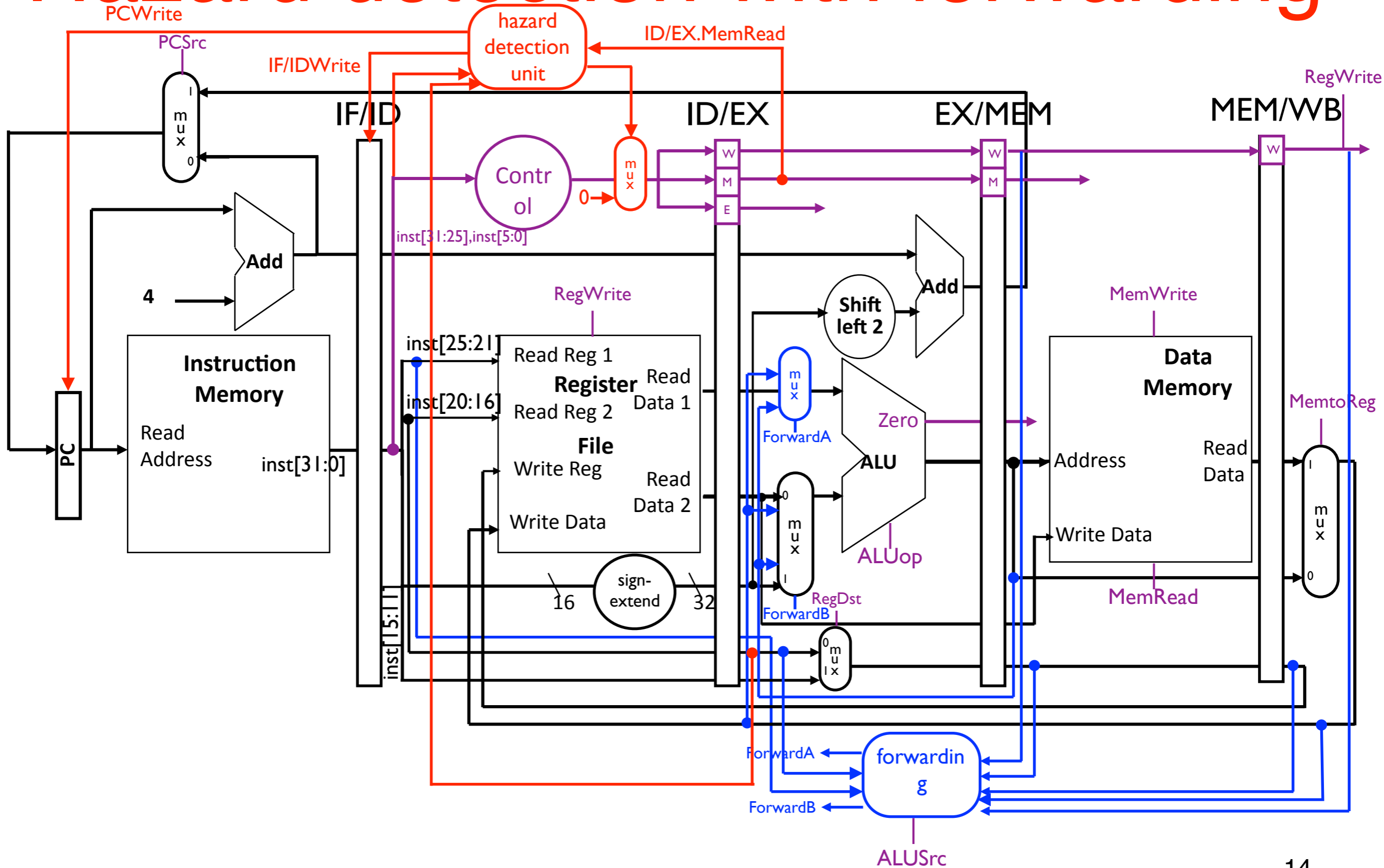
lw generates result at MEM stage, we have to stall

- If the instruction entering EXE stage depends on a load instruction that does not finish its MEM stage yet, we have to stall!
- We call this hazard detection

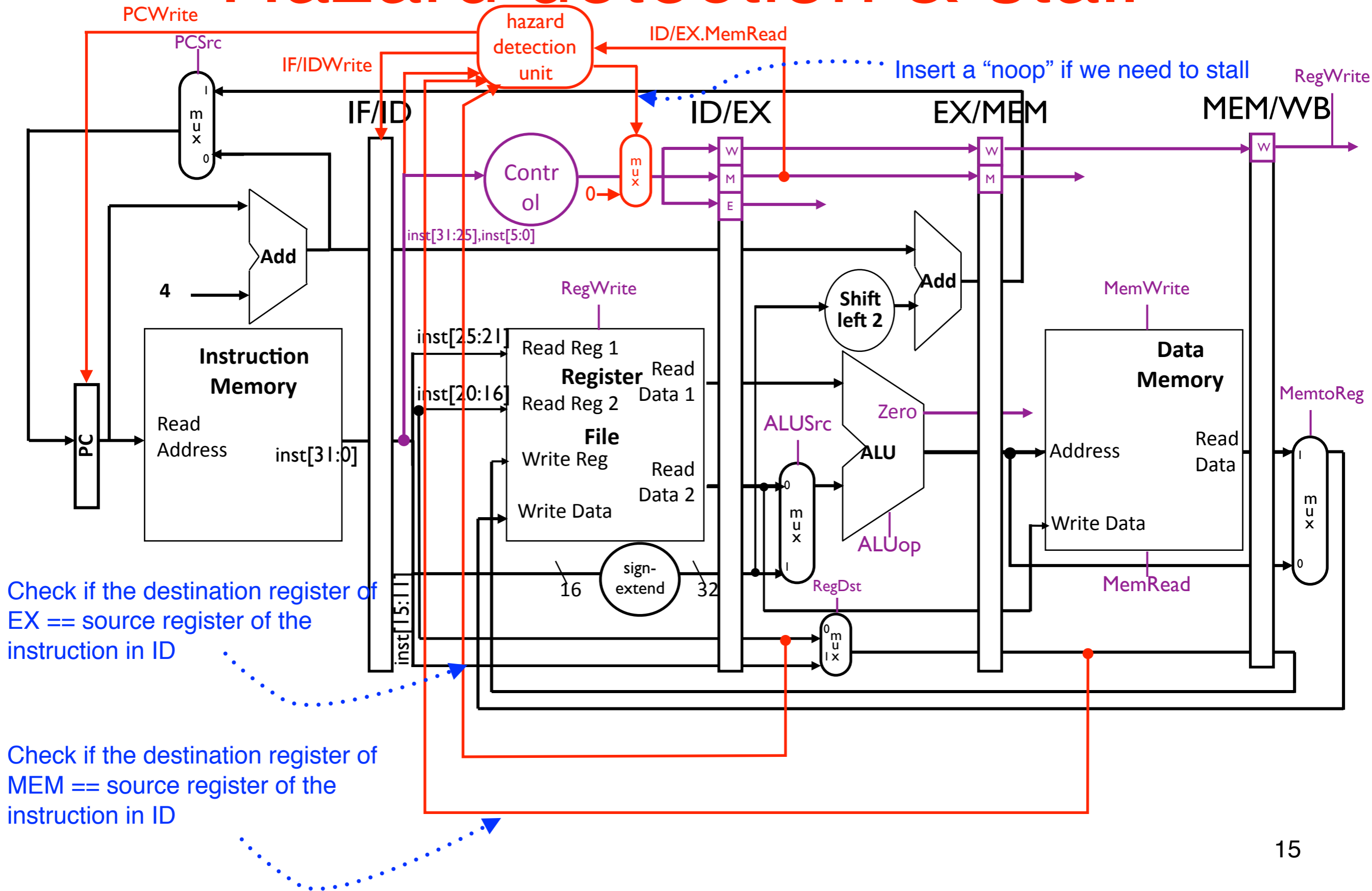
We need to know the following:

- If an instruction in EX/MEM updates a register (RegWrite)
- If an instruction in EX/MEM reads memory (MemRead)
- If the destination register of EX/MEM is a source of ID/EX (rs, rt of ID/EX == rt of EX/MEM #1)

# Hazard detection with forwarding



# Hazard detection & stall



# Dealing with hazards

- In standard 5-stage MIPS pipeline, you will meet
  - Data hazard
    - Stall
    - Data forwarding (bonus)
  - Control hazard
    - Stall
    - You may move branch resolution to the ID stage
    - Branch prediction (bonus)



# Instead of 5-stage

- 4-stage
  - For example, IF, ID, EX+MEM, WB
  - Fewer stall conditions
  - Longer cycle time
- 3-stage
- 2-stage
  - IF+ID, the rest
- IF alone cannot be a pipeline stage

# Benchmarks

- In this lab, we will use the following programs
  - No branch hello world
  - Hello world with branch
  - Fibonacci number
  - Start with PC 0x400000
- A set of testing scripts

# Interview questions

- Show the schematics
- Show the waveforms of three benchmarks until the end
- Measure the IC, total cycles, CPI
- Report the Fmax
  - We can calculate the performance of your processor!

# Lab 6: Optimization!

- Data forwarding
- Branch prediction
- Cache
- Superscalar
- Special bonus for fastest processor

Don't forget to come  
by 8/2 6pm @ WLH  
2111