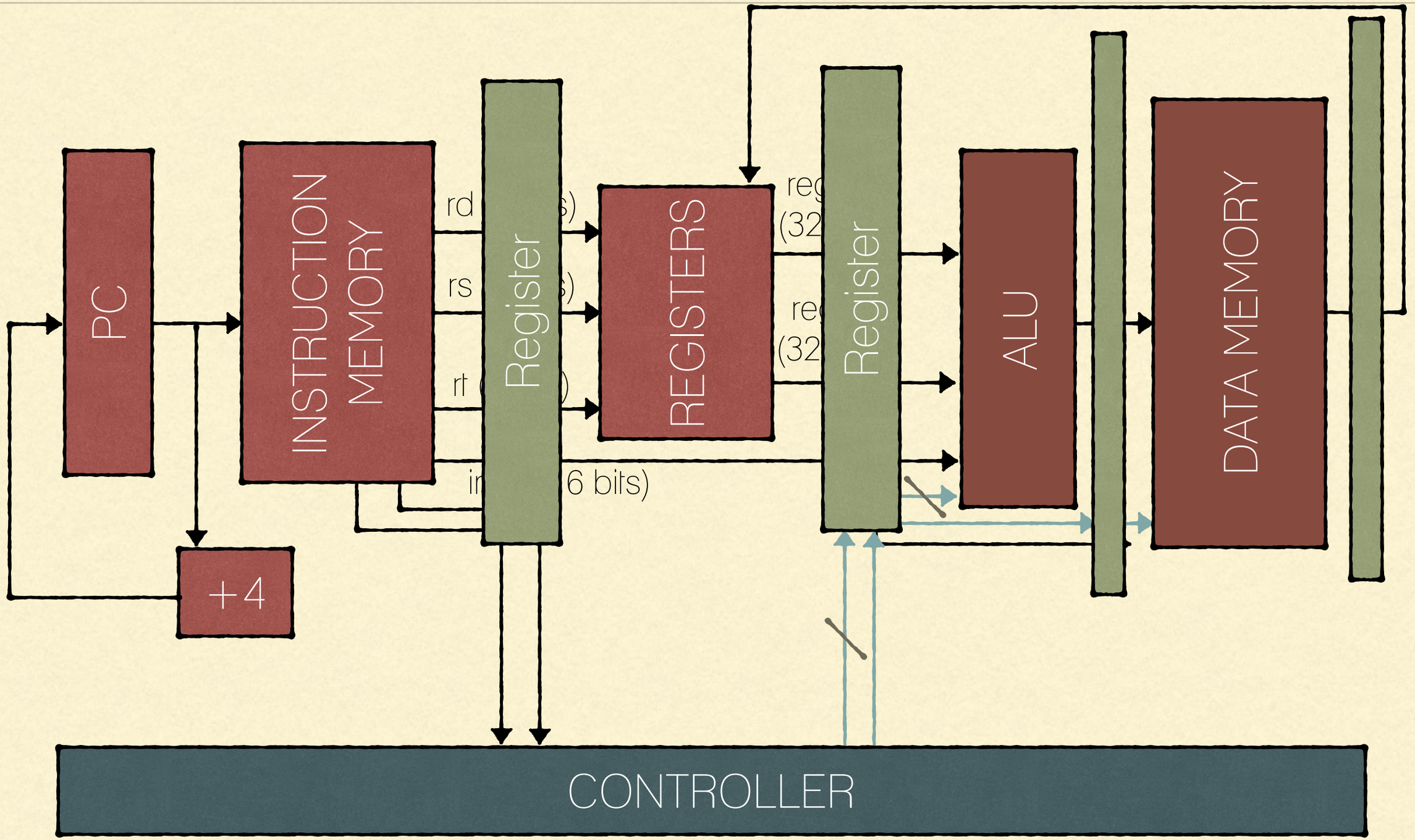
PIPELINING HAZARDS PART I

Ayman Hajja, PhD

STAGES OF EXECUTION ON PIPELINED DATAPATH



PIPELINING HAZARDS

- Pipelining Hazards: A hazard is a situation that prevents starting the next instruction in the next clock cycle
- Structural hazard
    - Required resource is busy (e.g. needed in multiple stages)
- Data hazard
    - Data dependency between instruction
    - Need to wait for previous instruction to complete its data read/ write
- Control hazard
    - Flow of execution depends on previous instruction

STRUCTURAL HAZARD

- Structural Hazards
  - Conflict for use of a resource
  - MIPS pipeline with a single memory?
    - Load/Store requires memory access for data
    - Instruction fetch would have to stall for that cycle
- Hence, pipelined data paths require separate instruction/data memories — more about this when we talk about memory caches

EXAMPLE OF STRUCTURAL HAZARD

	Stages								
Instruction 1	IF	ID	EX	MEM	WB				
Instruction 2		IF	ID	EX	MEM	WB			
Instruction 3			IF	ID	EX	MEM	WB		
Instruction 4				IF	ID	EX	MEM	WB	
Instruction 5					IF	ID	EX	MEM	WB

- Instruction 1 is loading/storing to the memory, and Instruction 4 is loading an instruction from the memory
    - However, since we have two separate caches, one for the data, and one for the instructions, then a conflict won't occur

EXAMPLE OF STRUCTURAL HAZARD

	Stages								
Instruction 1	IF	ID	EX	MEM	WB				
Instruction 2		IF	ID	EX	MEM	WB			
Instruction 3			IF	ID	EX	MEM	WB		
Instruction 4				IF	ID	EX	MEM	WB	
Instruction 5					IF	ID	EX	MEM	WB

Can we read and write at the same time?

- Solution to this conflict presented in next slide.

STRUCTURAL HAZARD

- Solution:

    1. Build RegFile (register file) with independent read and write ports

    2. Split RegFile access in two: Write during 1st half and read during 2nd half of each clock cycle

        - Possible because RegFile access is VERY fast (takes less than half the time of ALU stage)

DATA HAZARD: WHERE DO WE HAVE A PROBLEM HERE?

- Consider the following sequence of instructions:
  - add $t0, $t1, $t2
  - sub $t4, $t0, $t3
  - and $t5, $t0, $t6
  - or $t7, $t0, $t8
  - xor $t9, $t0, $t10

DATA HAZARD

- Consider the following sequence of instructions:
  - add $t0, $t1, $t2
  - sub $t4, $t0, $t3
  - and $t5, $t0, $t6
  - or $t7, $t0, $t8
  - xor $t9, $t0, $t10

EXAMPLE OF STRUCTURAL HAZARD

	Stages								
add $t0, $t1, $t2	IF	ID	EX	MEM	WB				
sub $t4, $t0, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB

EXAMPLE OF DATA HAZARD

Here, we're calculating the new value for $t0

	Stages								
add $t0, $t1, $t2	IF	ID	EX	MEM	WB				
sub $t4, $t0, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB

EXAMPLE OF DATA HAZARD

Here, we have the correct value for $t0

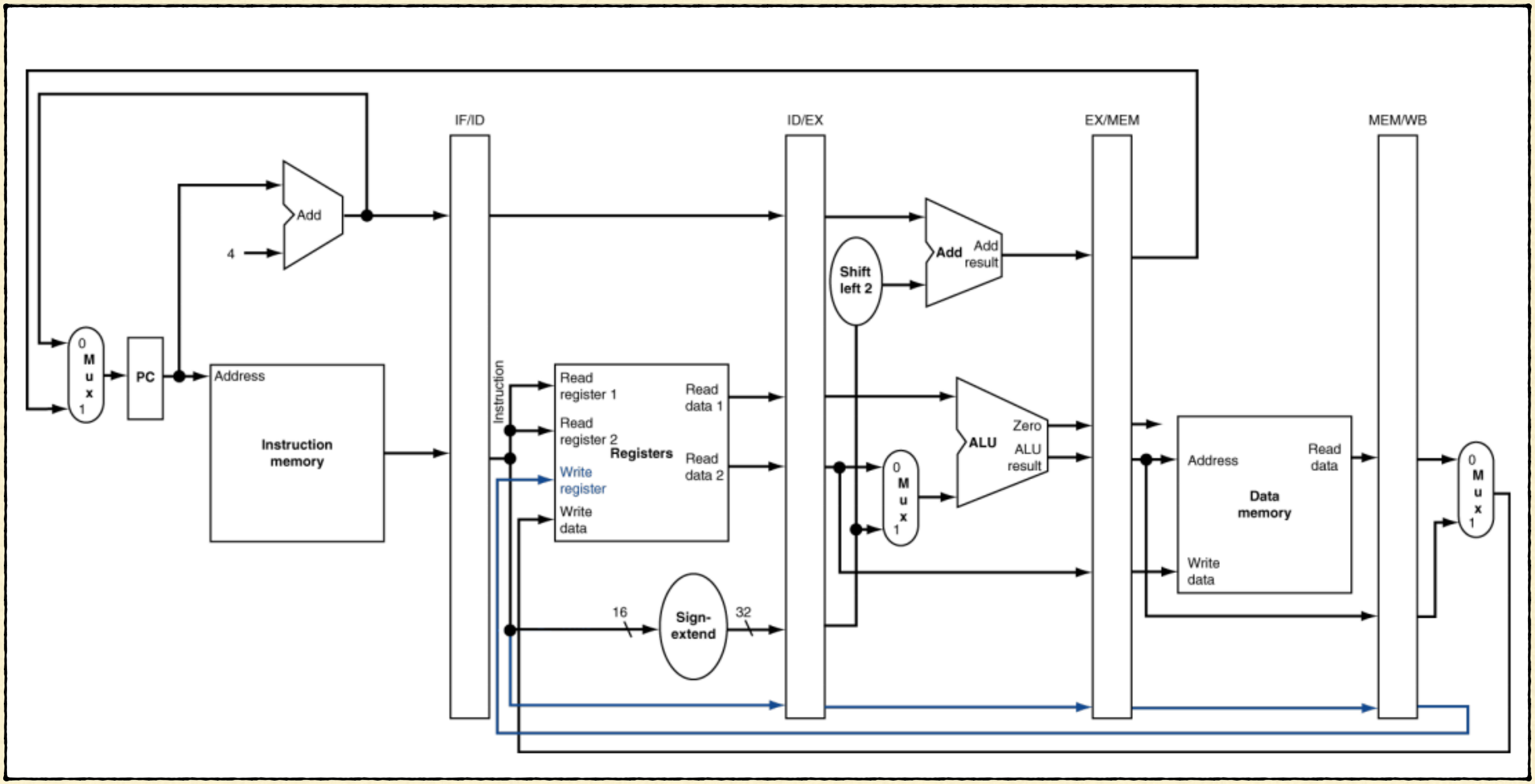
	Stages								
add $t0, $t1, $t2	IF	ID	EX	MEM	WB				
sub $t4, $t0, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB

EXAMPLE OF DATA HAZARD

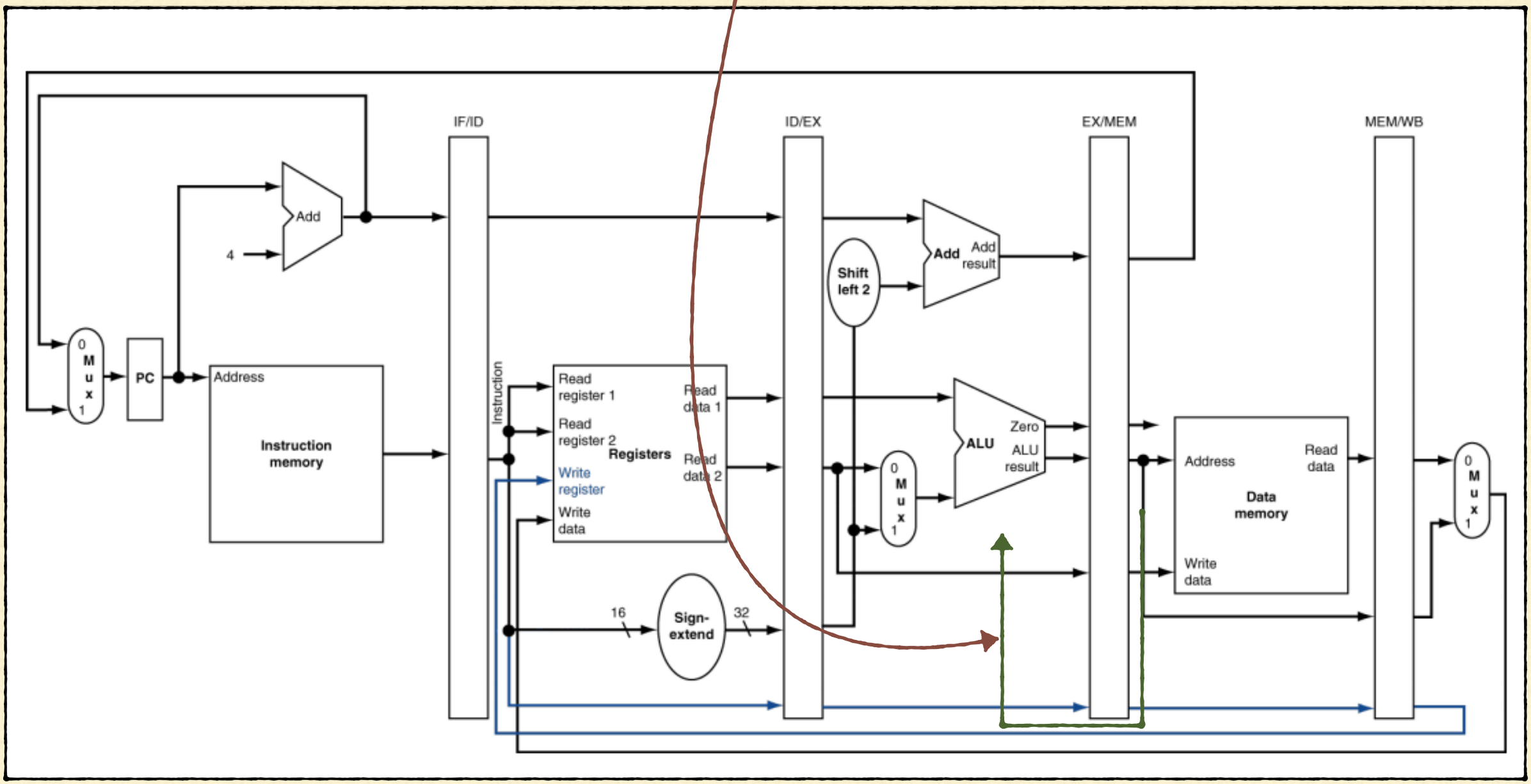
Here we need the correct value of $t0

	Stages								
add $t0, $t1, $t2	IF	ID	EX	MEM	WB				
sub $t4, $t0, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB

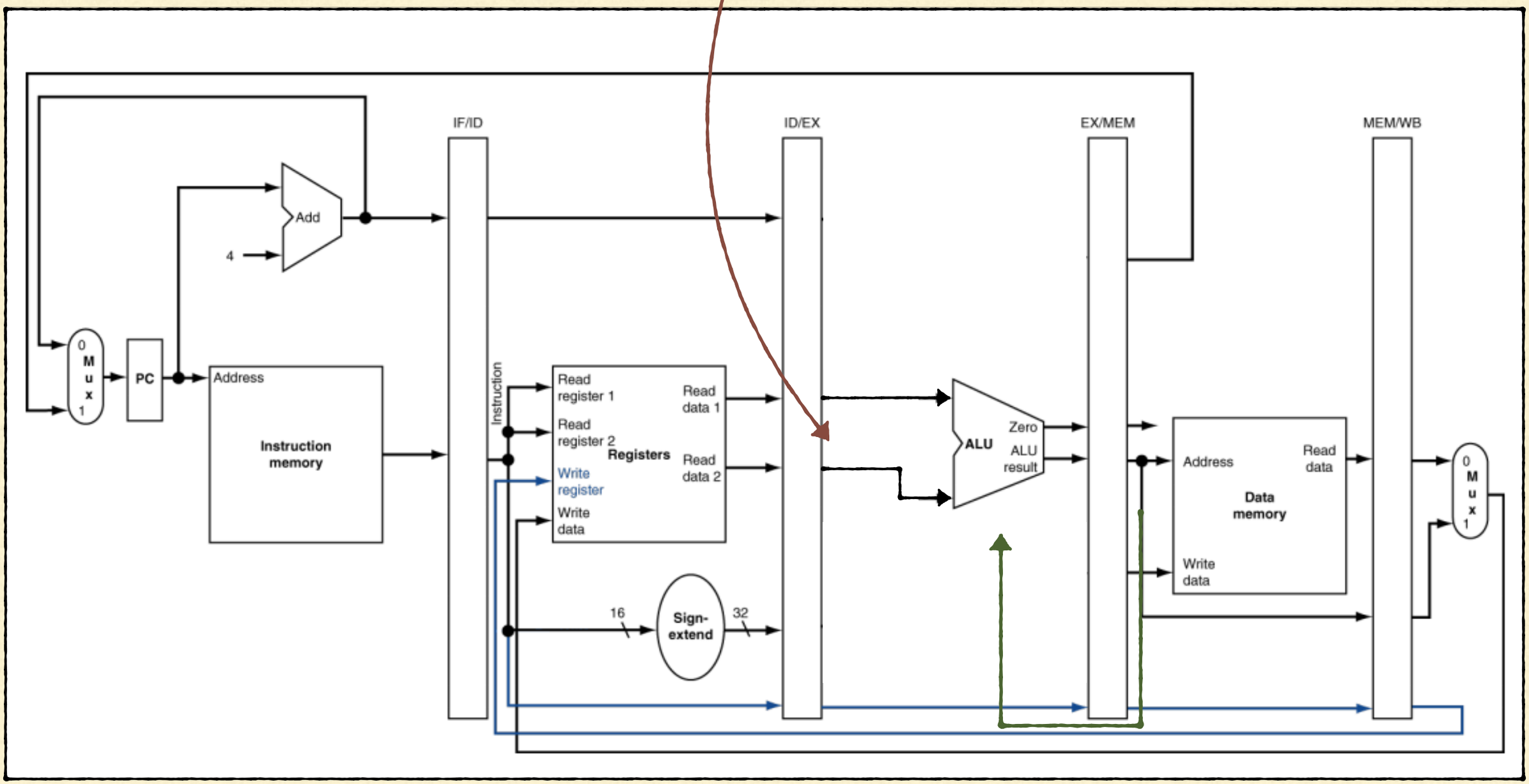
WHAT CHANGES NEED TO BE MADE?



We need to send the result back to the previous (ALU) stage for the next instruction to use!



Let's assume for now that we're dealing only with R-format instructions (no immediate), just to simplify our diagram.



DATA HAZARD

- We'll resume explaining how forwarding works after few slides; now let's consider the following sequence of instructions:
  - lw $t0, 16($t2)
  - sub $t4, $t1, $t3
  - and $t5, $t0, $t6
  - or $t7, $t0, $t8
  - xor $t9, $t0, $t10
- Do we have a problem here?

EXAMPLE OF DATA HAZARD

	Stages								
lw $t0, 16($t2)	IF	ID	EX	MEM	WB				
sub $t4, $t1, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB

EXAMPLE OF DATA HAZARD

Here, we have the correct value for $t0

	Stages								
lw $t0, 16($t2)	IF	ID	EX	MEM	WB				
sub $t4, $t1, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB

EXAMPLE OF DATA HAZARD

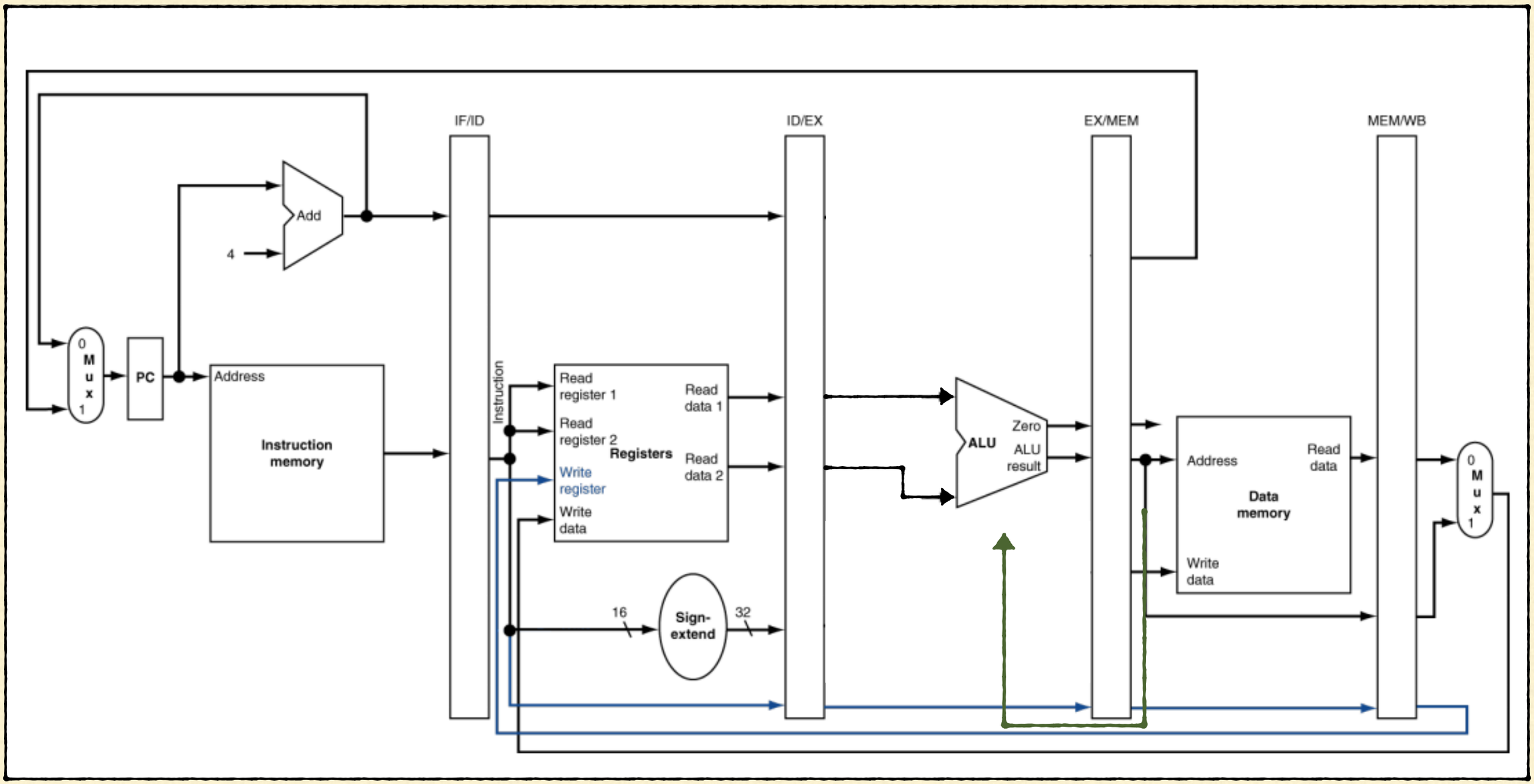
Here we need the correct value of $t0

	Stages								
lw $t0, 16($t2)	IF	ID	EX	MEM	WB				
sub $t4, $t1, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB

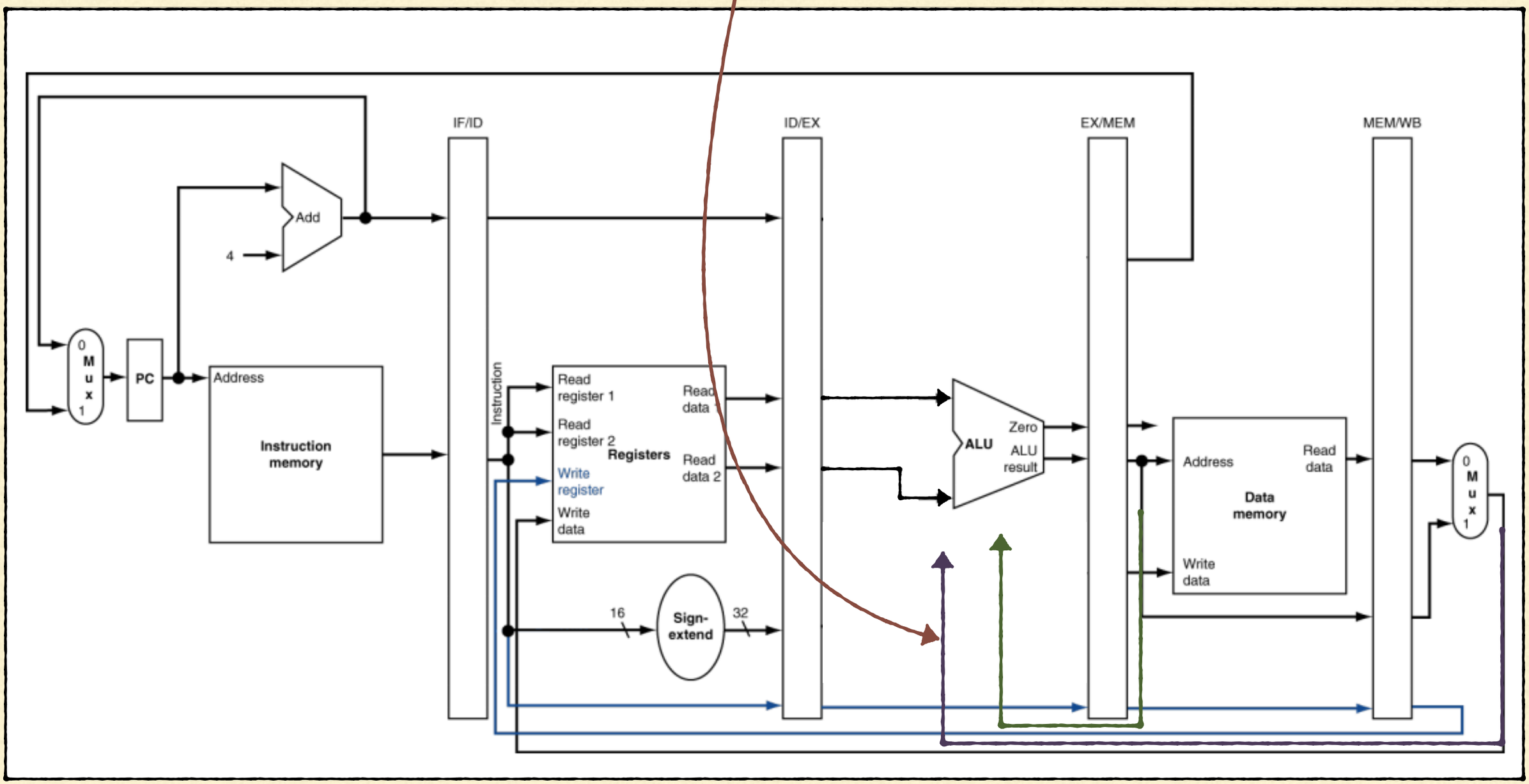
EXAMPLE OF DATA HAZARD

One solution is to forward the value of $t0 from WB to ALU

	Stages								
lw $t0, 16($t2)	IF	ID	EX	MEM	WB				
sub $t4, $t1, $t3		IF	ID	EX	MEM	WB			
and $t5, $t0, $t6			IF	ID	EX	MEM	WB		
or $t7, $t0, $t8				IF	ID	EX	MEM	WB	
xor $t9, $t0, $t10					IF	ID	EX	MEM	WB


IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
M
u
x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign-extend
32
ALU
Zero
ALU result
Address
Data memory
Read data
Write data
M
u
x
0
1

We need to send the result back to the previous (ALU) stage!

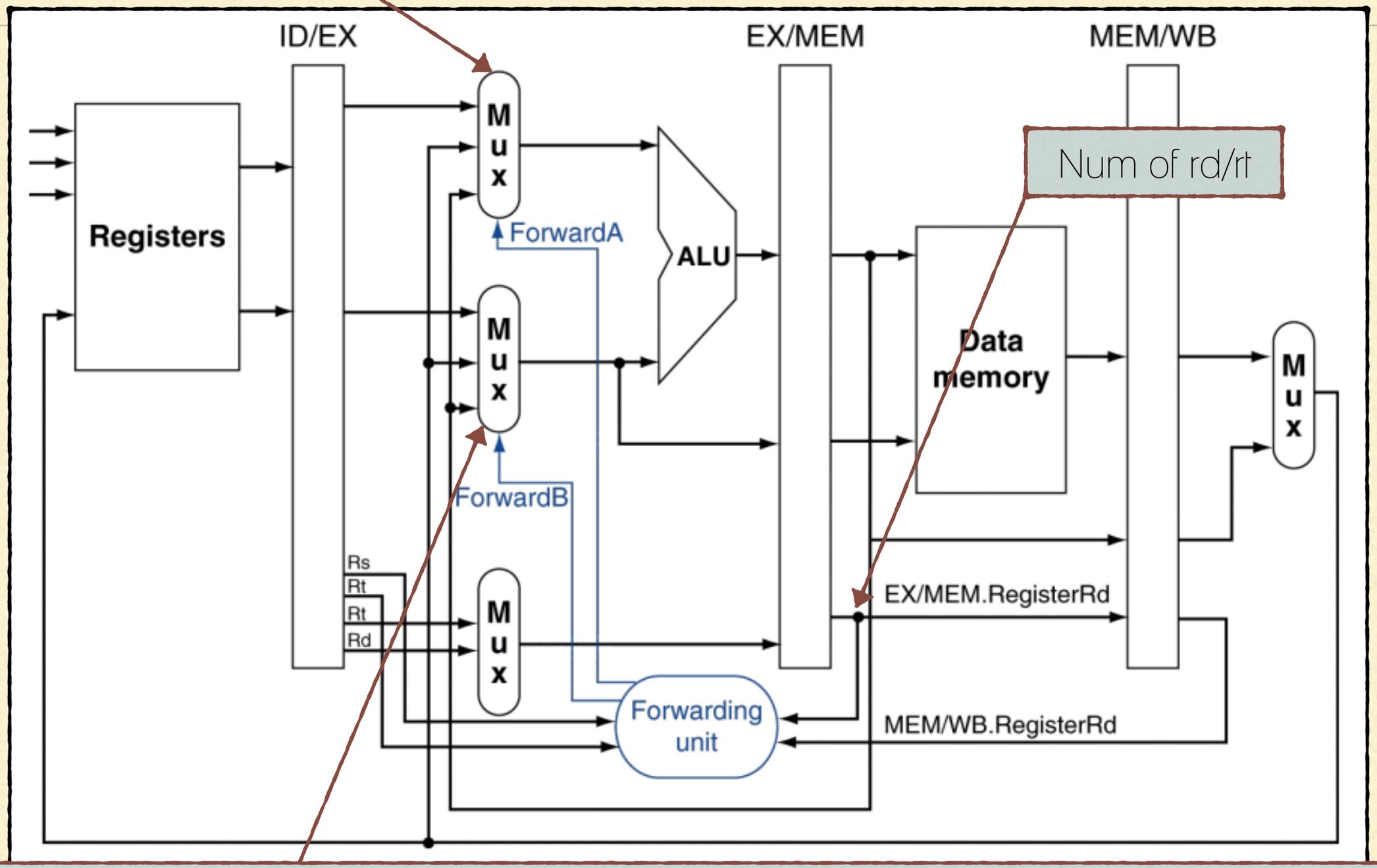


DATA HAZARD

- Now the big question is: How would the CPU know whether to use the values forwarded back from other instructions or the values being provided by register ID/EX?
- In this example, the CPU should NOT use the value forwarded back:
    - add $t0, $t1, $t2
    - sub $t4, $t1, $t3
- However, in this example, the CPU SHOULD use the values forwarded back:
    - add $t0, $t1, $t2
    - sub $t4, $t0, $t3

INTRODUCING FORWARDING UNIT

Three different input possibilities: rs, output of ALU, & output of accessing memory



Three different input possibilities: rt, output of ALU, & output of accessing memory