

CSCI 210: Computer Architecture

Lecture 25: Datapath

Stephen Checkoway

Oberlin College

Dec. 6, 2021

Slides from Cynthia Taylor

Announcements

- Problem Set 8 due Friday
- Lab 7 due Sunday (up tonight)
- Office Hours tomorrow 13:30 –14:30

The Processor: Datapath & Control

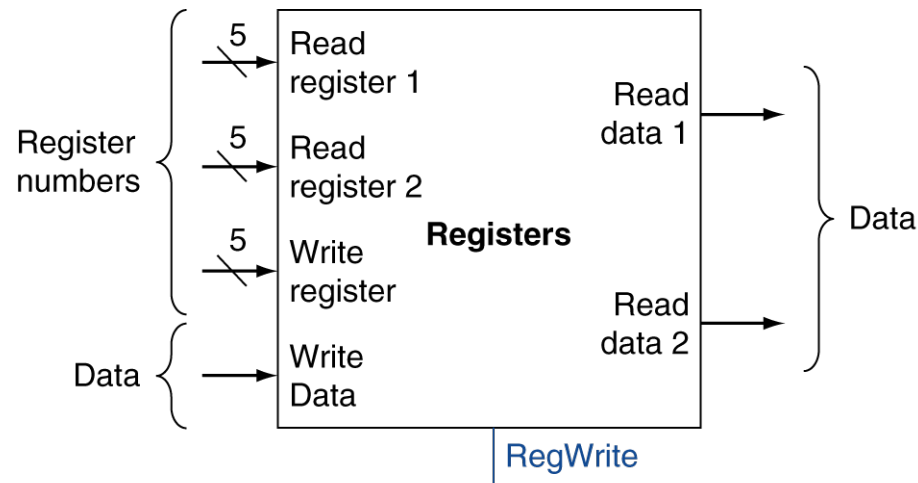
- We're ready to look at an implementation of MIPS simplified to contain only:
 - memory-reference instructions: `lw, sw`
 - arithmetic-logical instructions: `add, sub, and, or, slt`
 - control flow instructions: `beq`

Generic implementation

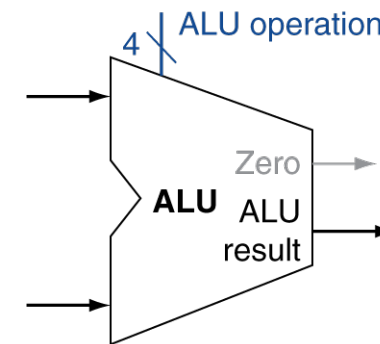
- **Fetch**
 - Use the program counter (PC) to supply instruction address
 - Get the instruction from memory
 - Update the program counter to the next instruction
- **Decode instruction**
 - Read registers
 - Use the instruction to decide exactly what to do
- **Execute**
 - Perform necessary data manipulation
 - Write to registers

R-Format Instructions

- Read two register operands
- Perform arithmetic/logical operation
- Write register result



a. Registers



b. ALU

Which of these describes our interface for memory?

What do we need for

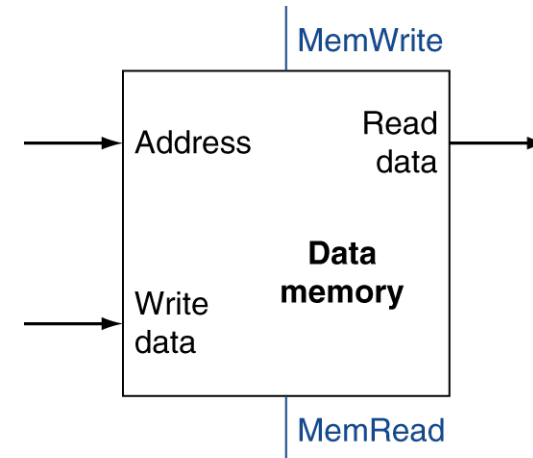
`lw $t2, 0($t3)` or `sw $t4, 4($t5)`

- A. One 32-bit data output, one 5-bit select input, one 32-bit data input, two 1-bit control inputs
- B. One 32-bit data output, two 5-bit select inputs, two 1-bit control inputs
- C. One 32-bit data output, one 32-bit select input, one 32-bit data input, two 1-bit control inputs
- D. One 32-bit data output, one 32-bit select input, two 1-bit control inputs
- E. None of the above

Data Memory

- 32-bit address input
- 32-bit data to write input
- 32-bit data output

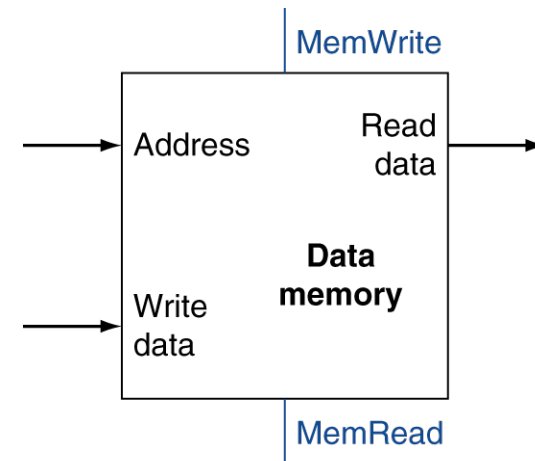
- 1-bit MemWrite control
- 1-bit MemRead control



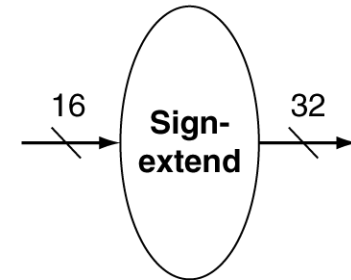
a. Data memory unit

Load/Store Instructions

- Read register operands
- Calculate address using 16-bit offset
 - Use ALU, but sign-extend offset
- Load: Read memory and update register
- Store: Write register value to memory

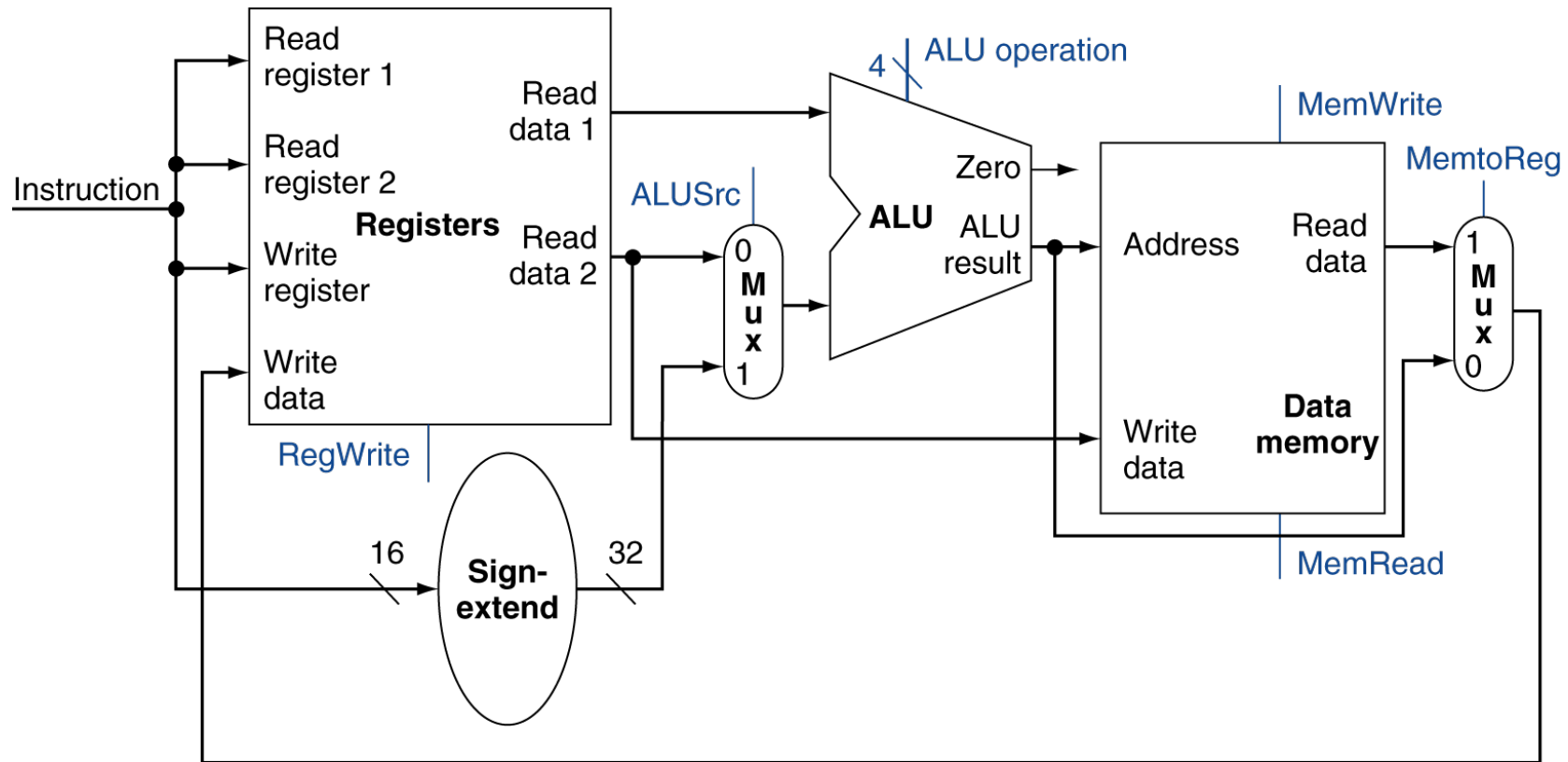


a. Data memory unit



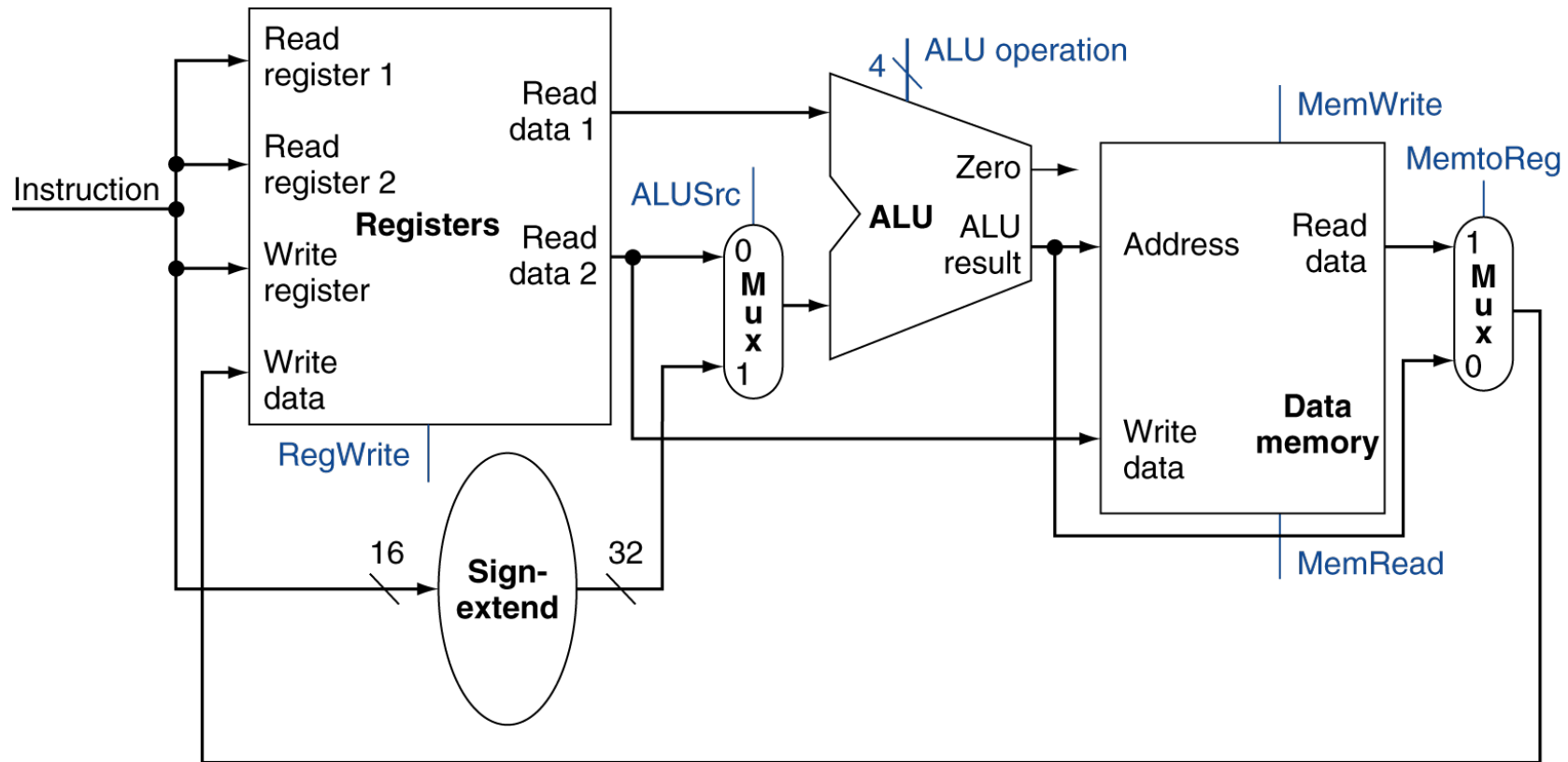
b. Sign extension unit

R-Type/Load/Store Datapath



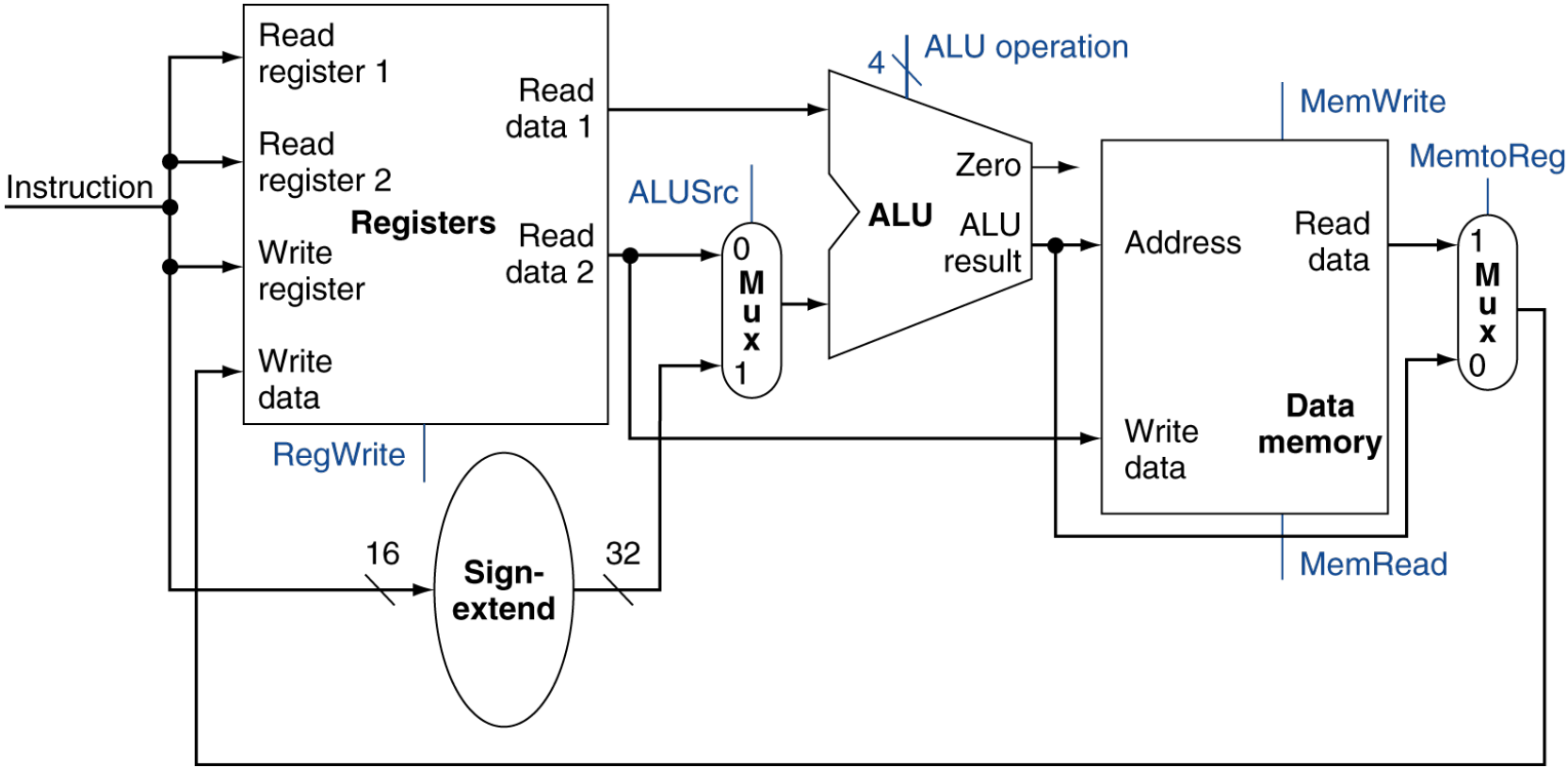
Add \$t0, \$t0, \$t1

\$t0 = 5
\$t1 = 6



lw \$t1, 4(\$t0)

\$t0 = 0x07AB8110
0x07AB8114 holds 12



Which is true about the ALU and the register file in MIPS?

- A. The ALU *always* performs an operation before accessing the register file
- B. The ALU *sometimes* performs an operation before accessing the register file
- C. The register file is *always* accessed before performing an ALU operation
- D. The register file is *sometimes* accessed before performing an ALU operation
- E. None of the above.

Conditional Branch Instructions Require

A. ALU

```
beq $t2, $t3, 0x4F35
```

B. Registers and an ALU

C. Registers, ALU and Memory

D. Registers, an ALU and an Adder

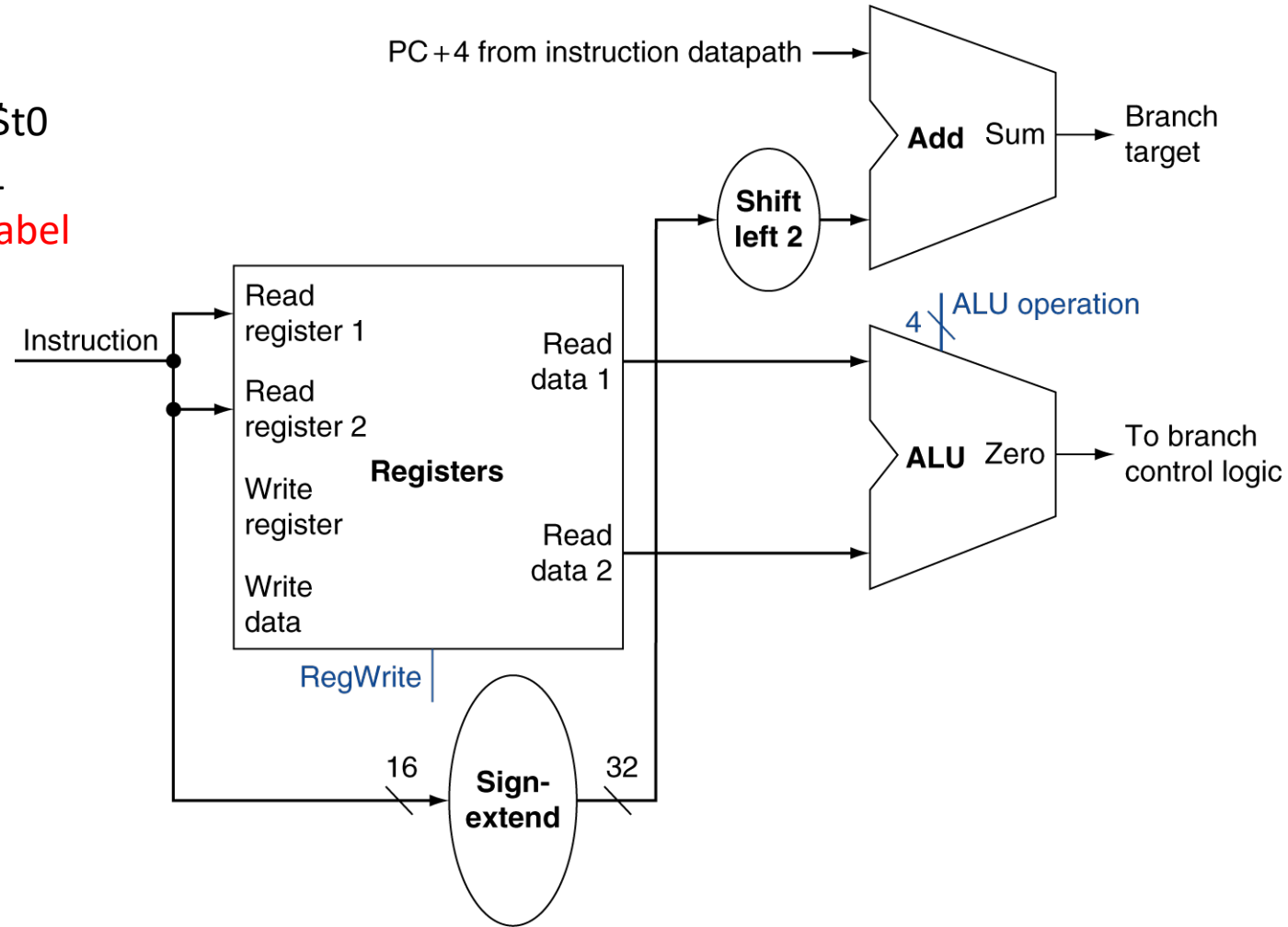
Branch Instructions

- Read register operands
- Compare operands
 - Use ALU, subtract and check Zero output
- Calculate target address
 - Sign-extend offset
 - Shift left 2 bits (word offset)
 - Add to PC + 4
 - Already calculated during instruction fetch

Branch Instructions

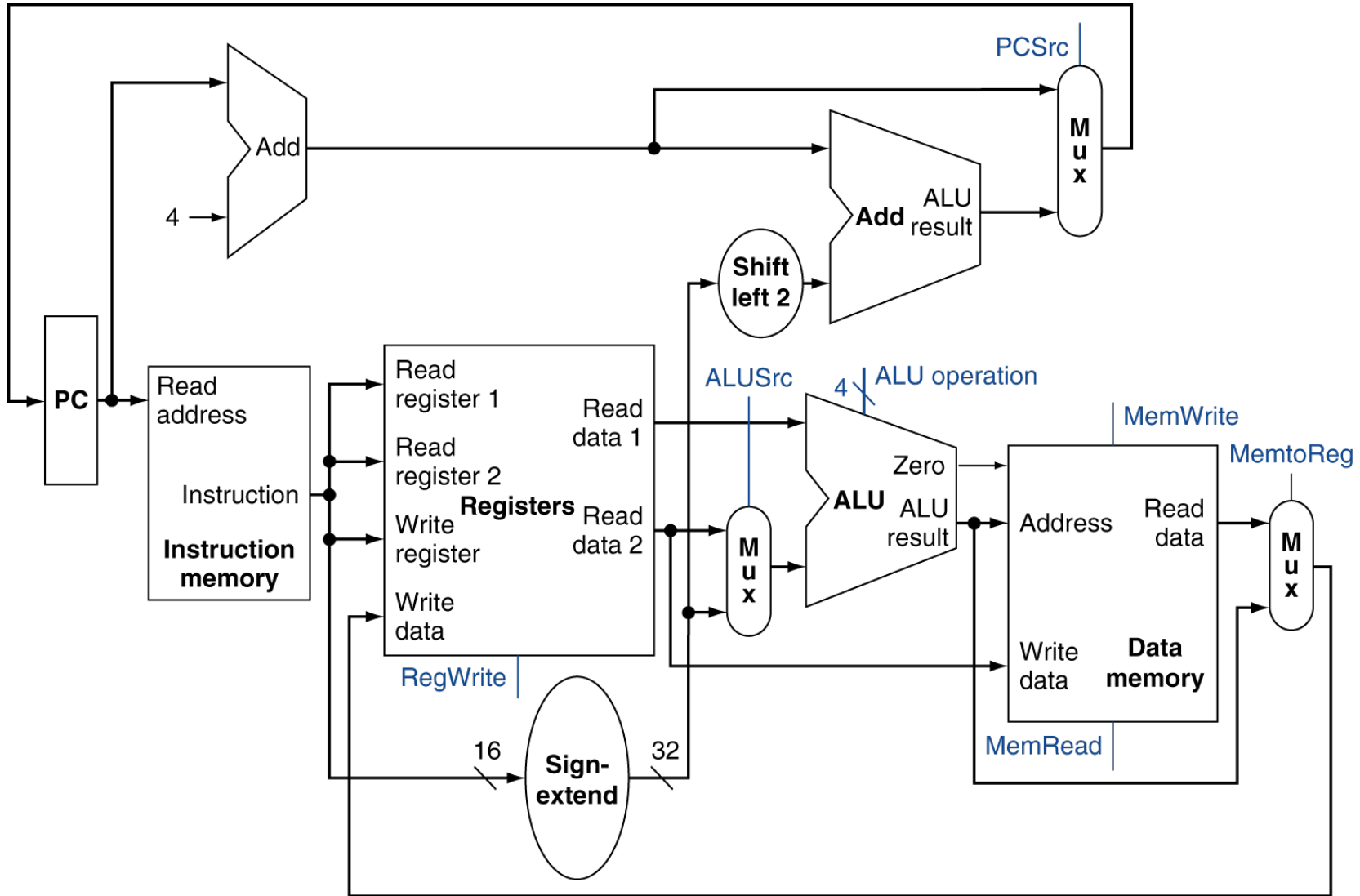
label:

add \$s0, \$s0, \$t0
 addi \$t0, \$t0, 1
 beq \$t0, \$t1, label



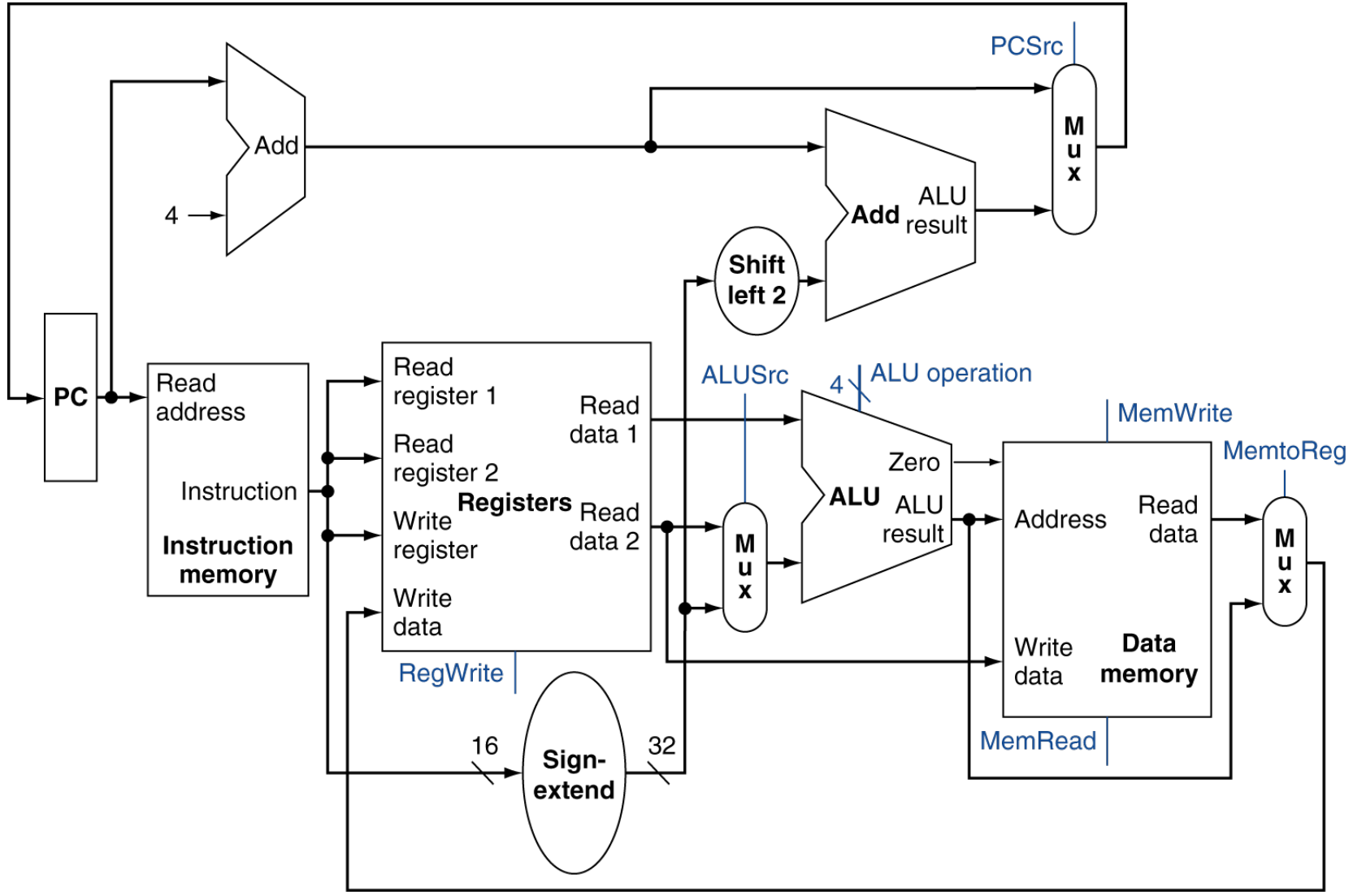
op = 0x04	rs = 8	rt = 9	imm = 0xFFFFD
-----------	--------	--------	---------------

Datapath (still simplified a bit)

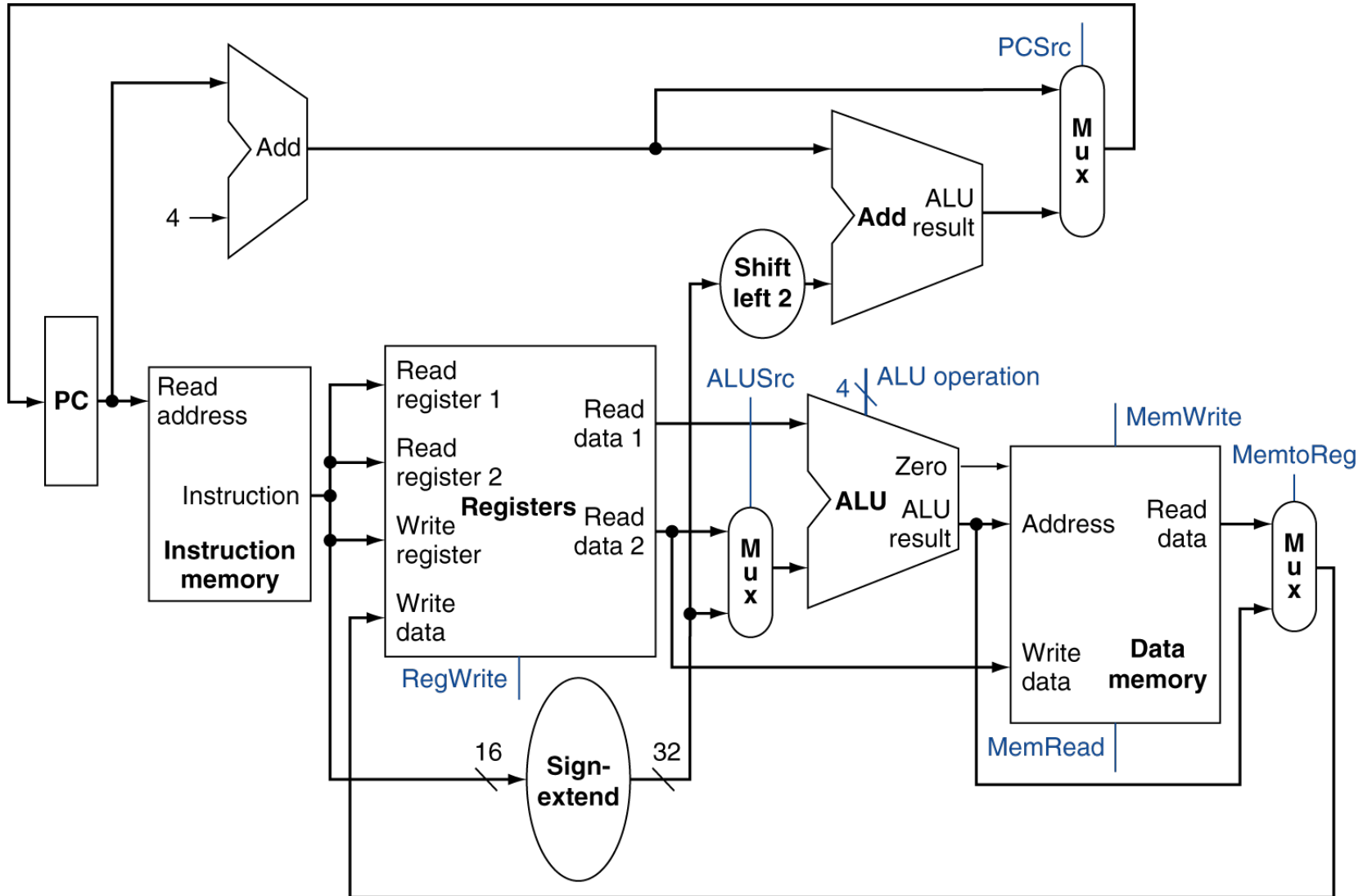


addi \$t1, \$t0, -1

\$t0 = 10

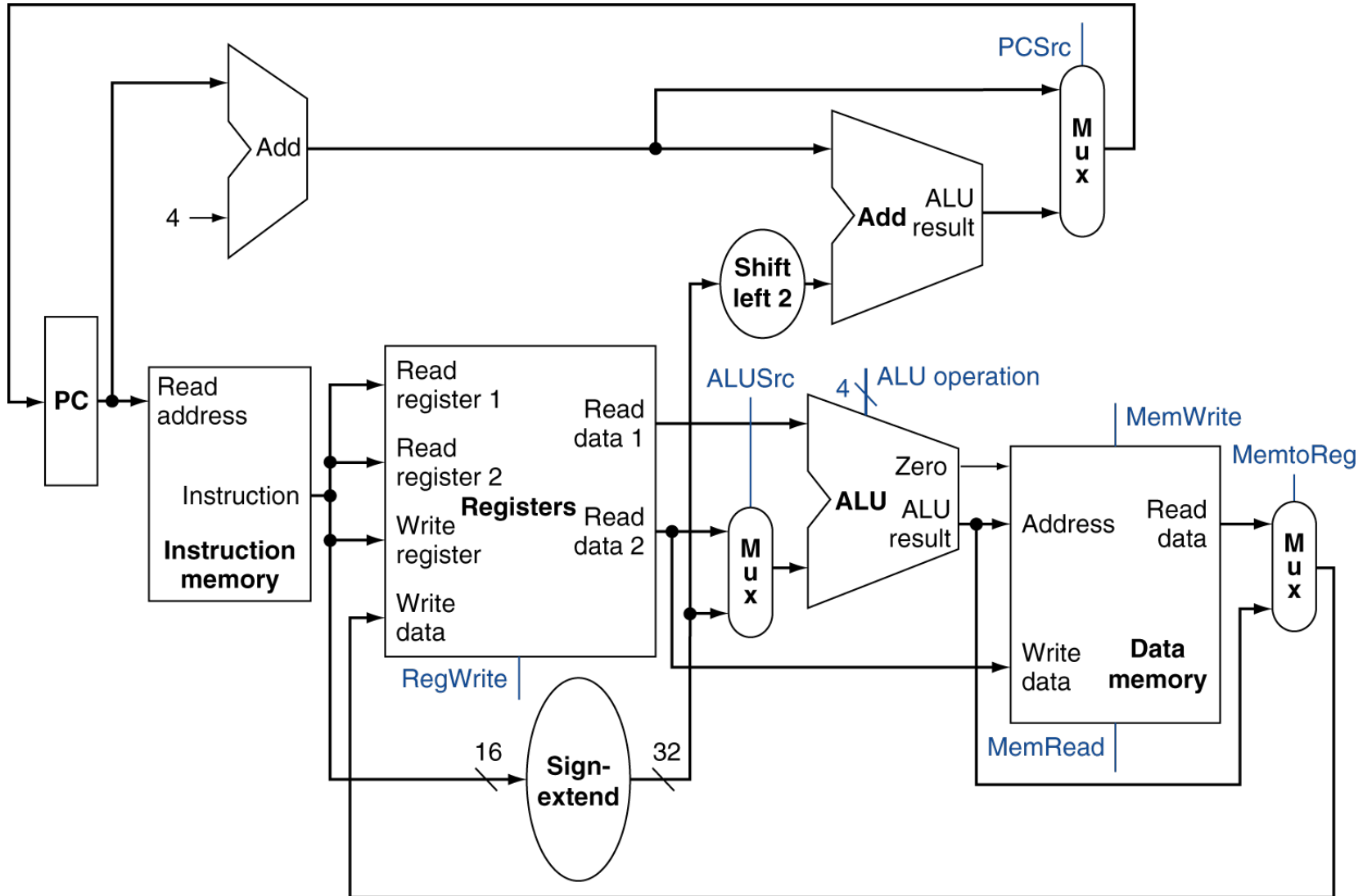


What do we need to add to support ori?



sw \$t1, 8(\$t0)

\$t0 = 0x07AB8110
\$t1 = 5



Composing the Elements

- First-cut data path does an instruction in one clock cycle
 - Each datapath element can only do one function at a time
 - Hence, we need separate instruction and data memories
- Use multiplexers where alternative data sources are used for different instructions

Key Points

- CPU is just a collection of state and combinational logic
- We just designed a very rich processor, at least in terms of functionality
- $ET = IC * CPI * \text{Cycle Time}$

Reading

- Next lecture: Control Path
 - Section 5.4
- Problem Set 8 due Friday
- Lab 7 due Sunday