

# CSCI 210: Computer Architecture

## Lecture 25: Datapath

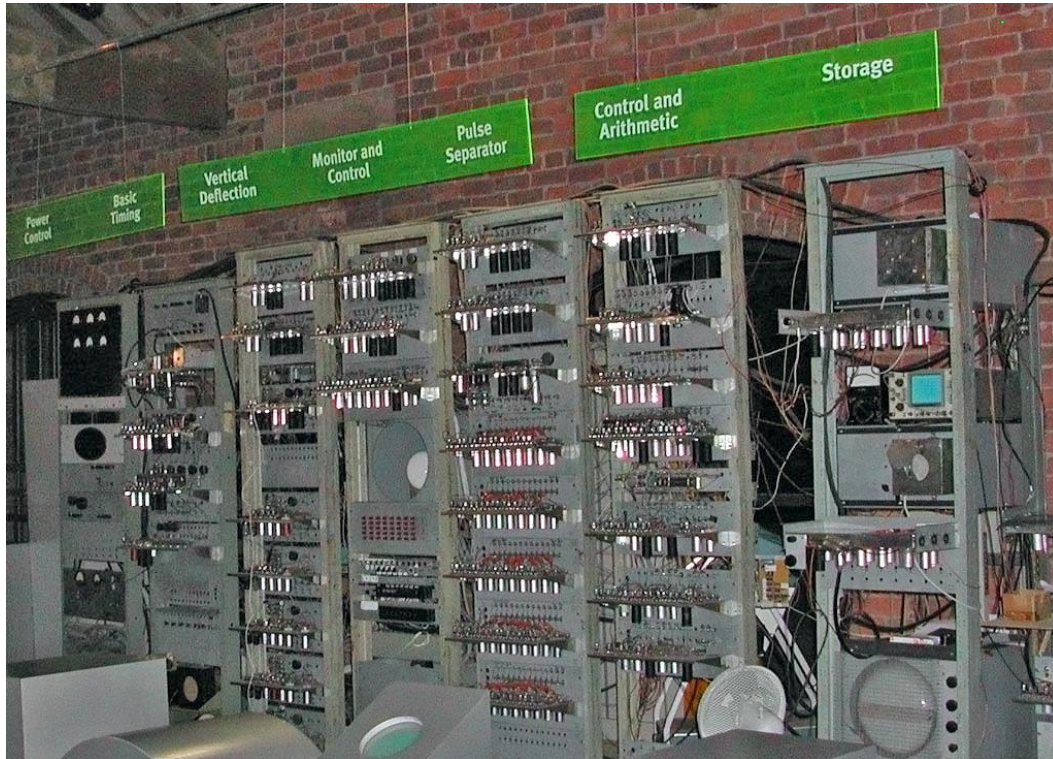
Stephen Checkoway

Slides from Cynthia Taylor

# Today's Class

- The datapath!

# CS History: The Manchester Baby



- First stored-program computer
- Ran its first program on June 21, 1948
- Designed as a testbed for the first random-access memory
- Only arithmetic operations were addition and subtraction
- Its first program calculated the highest proper divisor of  $2^{18}$  (262,144), by testing every integer from  $2^{18}$  downwards
- This program was 17 instructions and took 52 minutes to run

# 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`

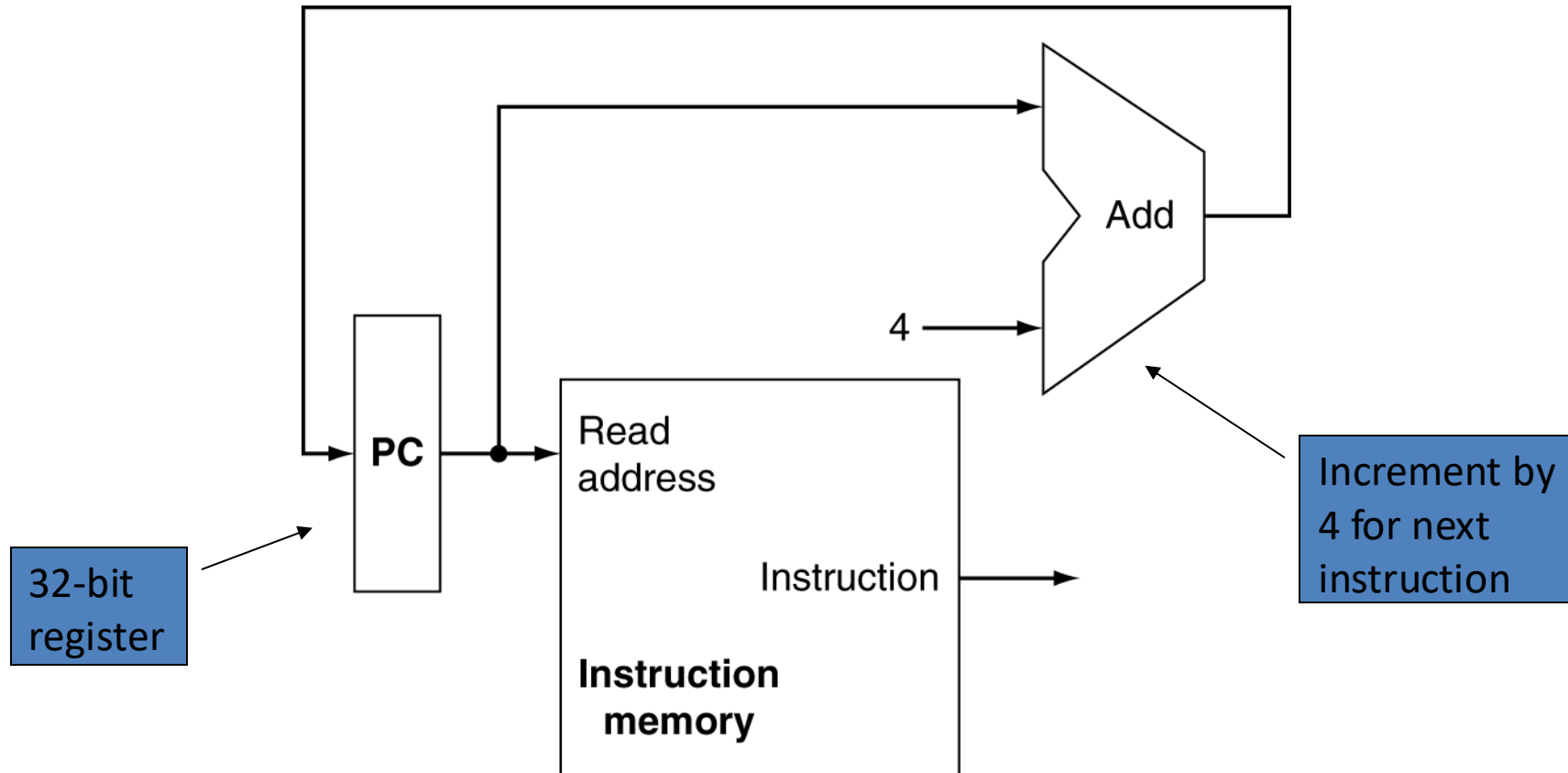
# Generic implementation

- 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
  - Read the instruction to decide how to execute
- Execute
  - Perform necessary data manipulation
  - Write to registers

# To fetch an instruction, what hardware do we need?

- Fetch
    - Use the program counter (PC) to supply instruction address
    - Get the instruction from memory
    - Update the program counter to the next instruction
- A. Register(s), Memory
  - B. Register(s), Adder, Memory
  - C. Register(s), ALU, Memory
  - D. More than this

# Instruction Fetch



# 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
  - Read the instruction to decide how to execute
- Execute
  - Perform necessary data manipulation
  - Write to registers



# Registers for instructions

- `add $t0, $t1, $t2` needs to read the values of registers `$t1` and `$t2` and write to register `$t0`
- `lw $t0, 4($t8)` needs to read one register and write one register
- `sw $t0, -8($s0)` needs to read two registers and write zero registers

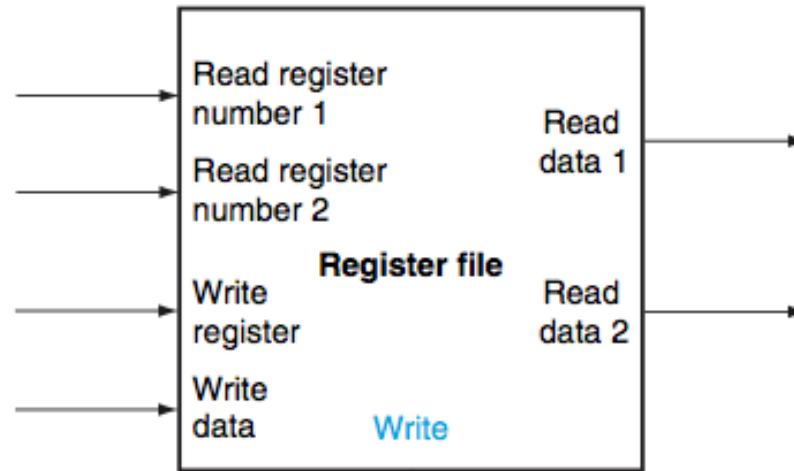
# Interface for the register file

- We need the ability to read from up to 2 registers and write up to 1 register

## Interface:

- Three 5-bit register select inputs (rs, rt, rd)
- Two 32-bit data outputs (data in rs and rt)
- One 32-bit data input (data to write to rd)
- One 1-bit control input (should input data be written to rd or not)

# Register File



Control input that is 1 if the write data should be written to the register specified by write register

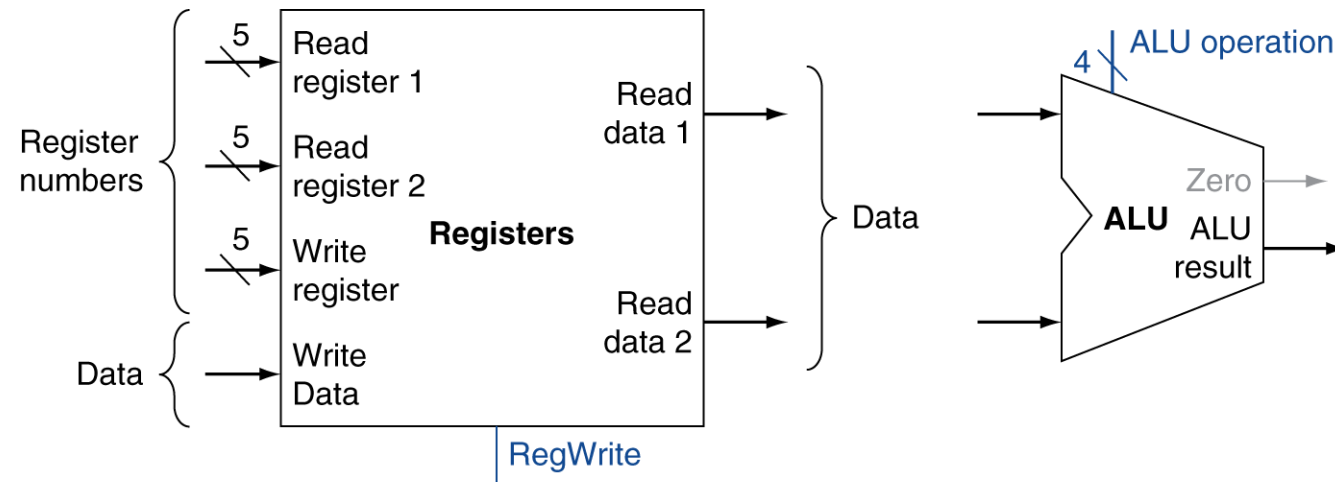


# Generic implementation

- 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
  - Read the instruction to decide how to execute
- 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



# Data memory

- `sub $t0, $t1, $t2` does not read or write memory
- `lw $t0, 0($s0)` reads 32-bits from memory
- `sw $t0, 0($s0)` writes 32-bits to memory

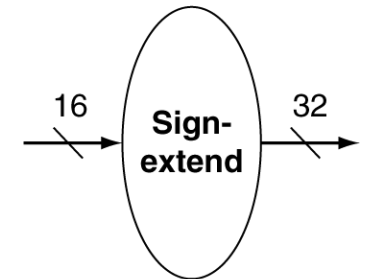
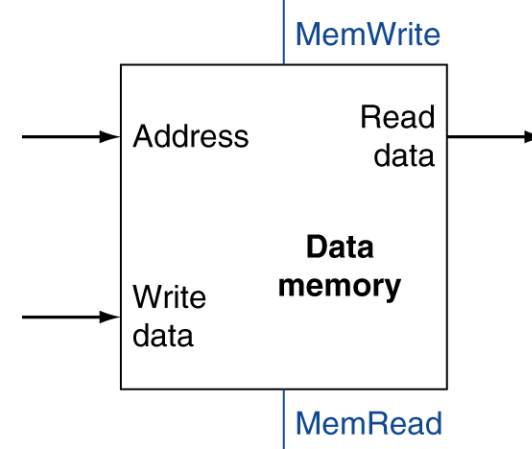
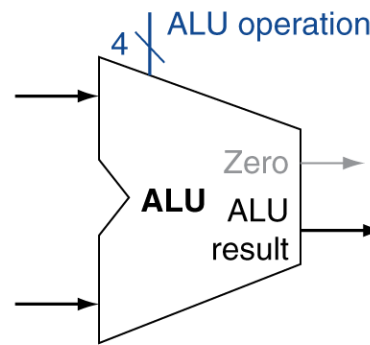
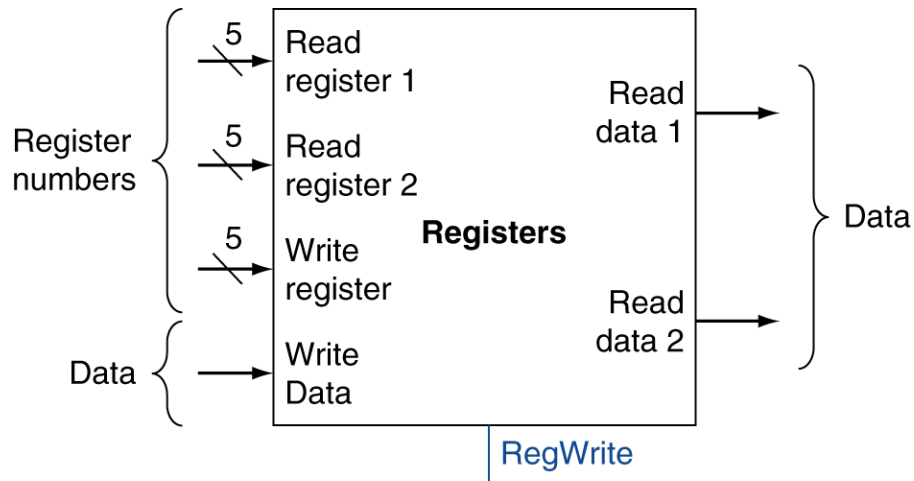
Which of these describes our interface for data memory? What do we need for  
`lw 0($t3), $t2` or `sw 4($t5), $t4`

- A. One 32-bit data output, 1 5-bit select input, 1 32-bit data input, 2 bits of control input
- B. One 32-bit data output, 2 5-bit select inputs, 2 bits of control input
- C. One 32-bit data output, 1 32-bit select input, 1 32-bit data input, 2 bits of control input
- D. One 32-bit data output, 1 32-bit select input, 2 bits of control input
- E. None of the above

Data is what we read/write to memory, select is our address, control is what the unit does

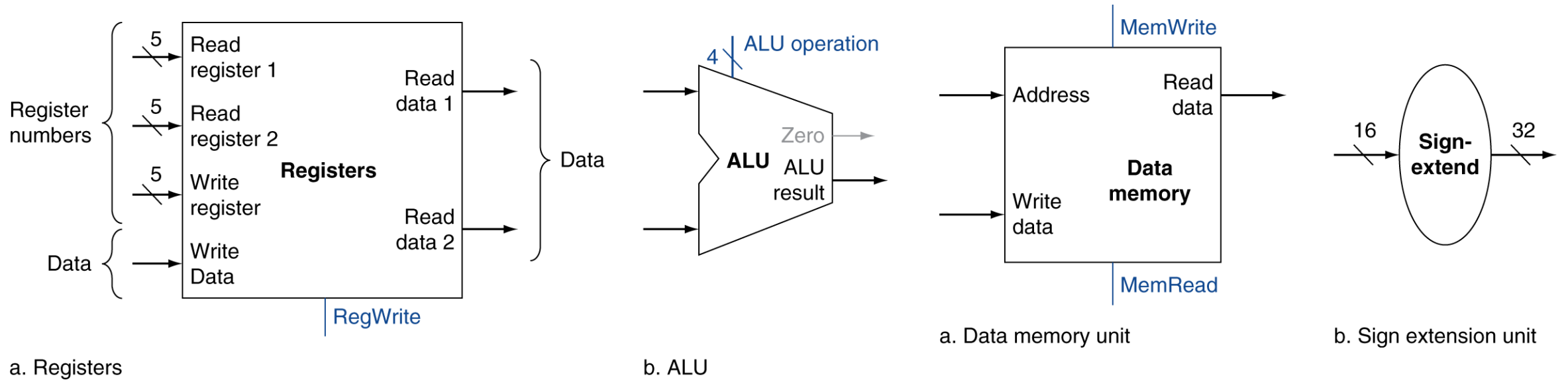
# Load/Store Instructions

- 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





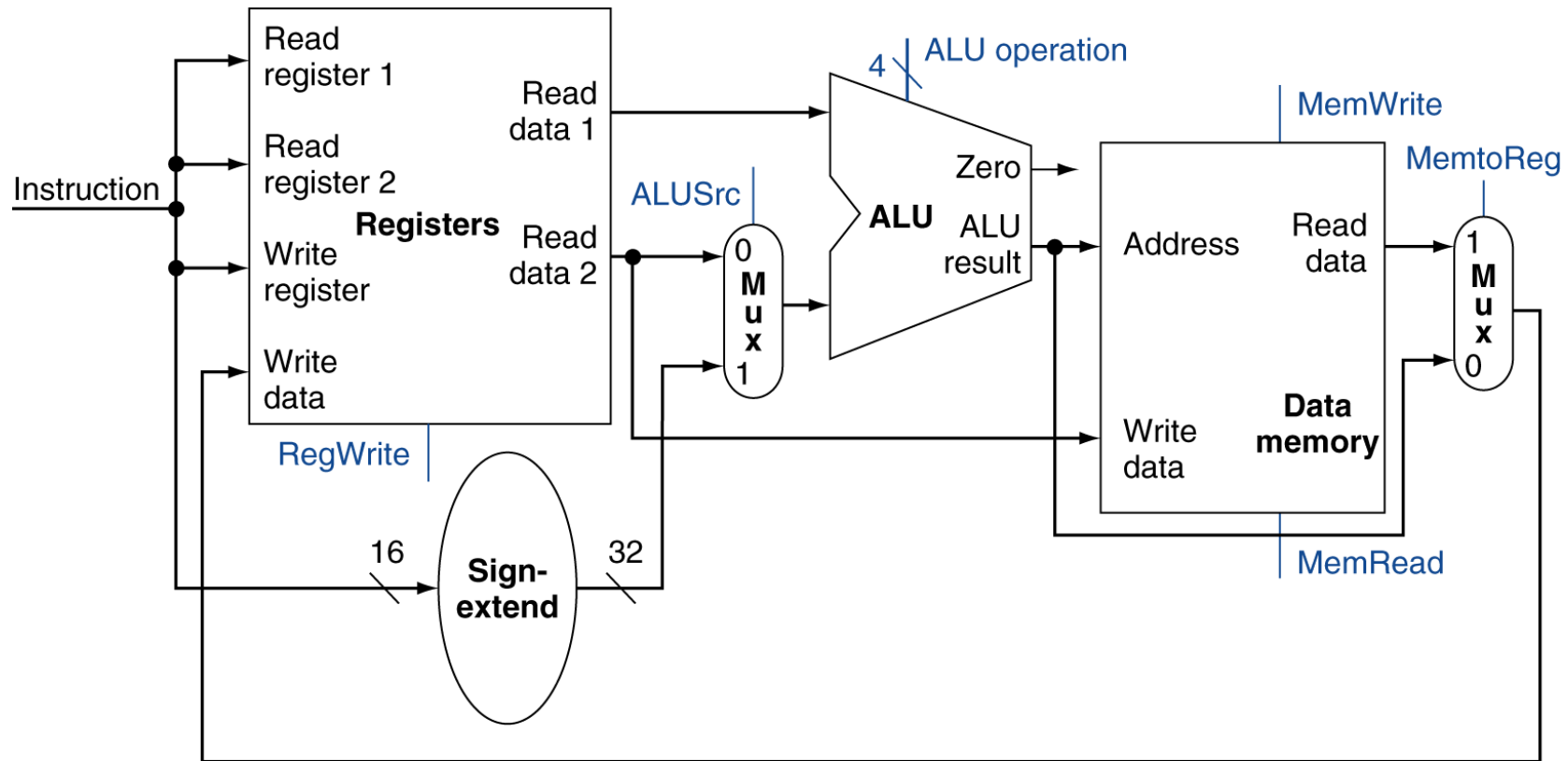
# Memory



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

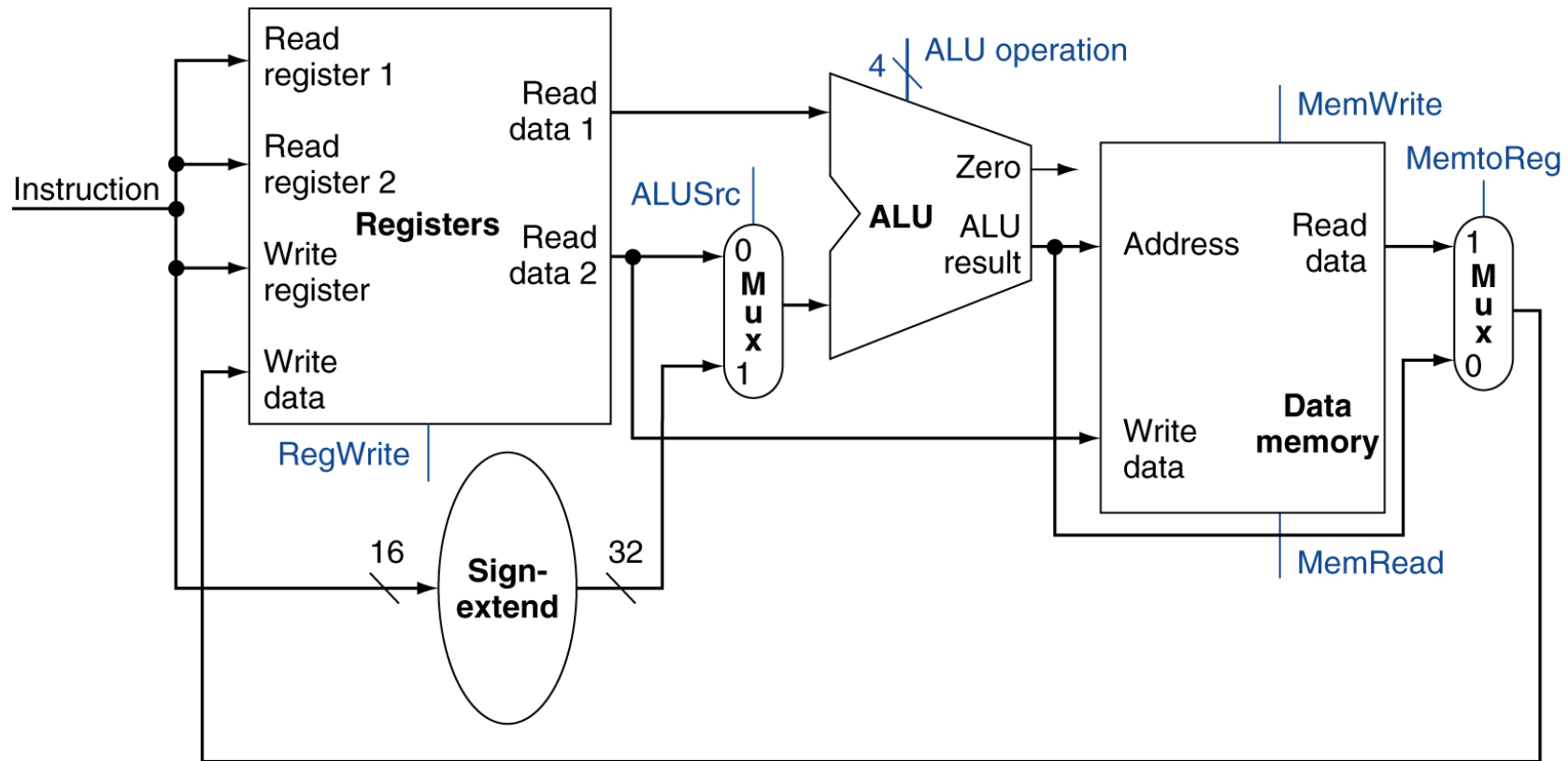
- A. The ALU *always* performs an operation before accessing the register file
- B. The ALU *sometimes* performs an operation before accessing the register file
- C. The register file is *always* accessed before performing an ALU operation
- D. The register file is *sometimes* accessed before performing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

# R-Type/Load/Store Datapath



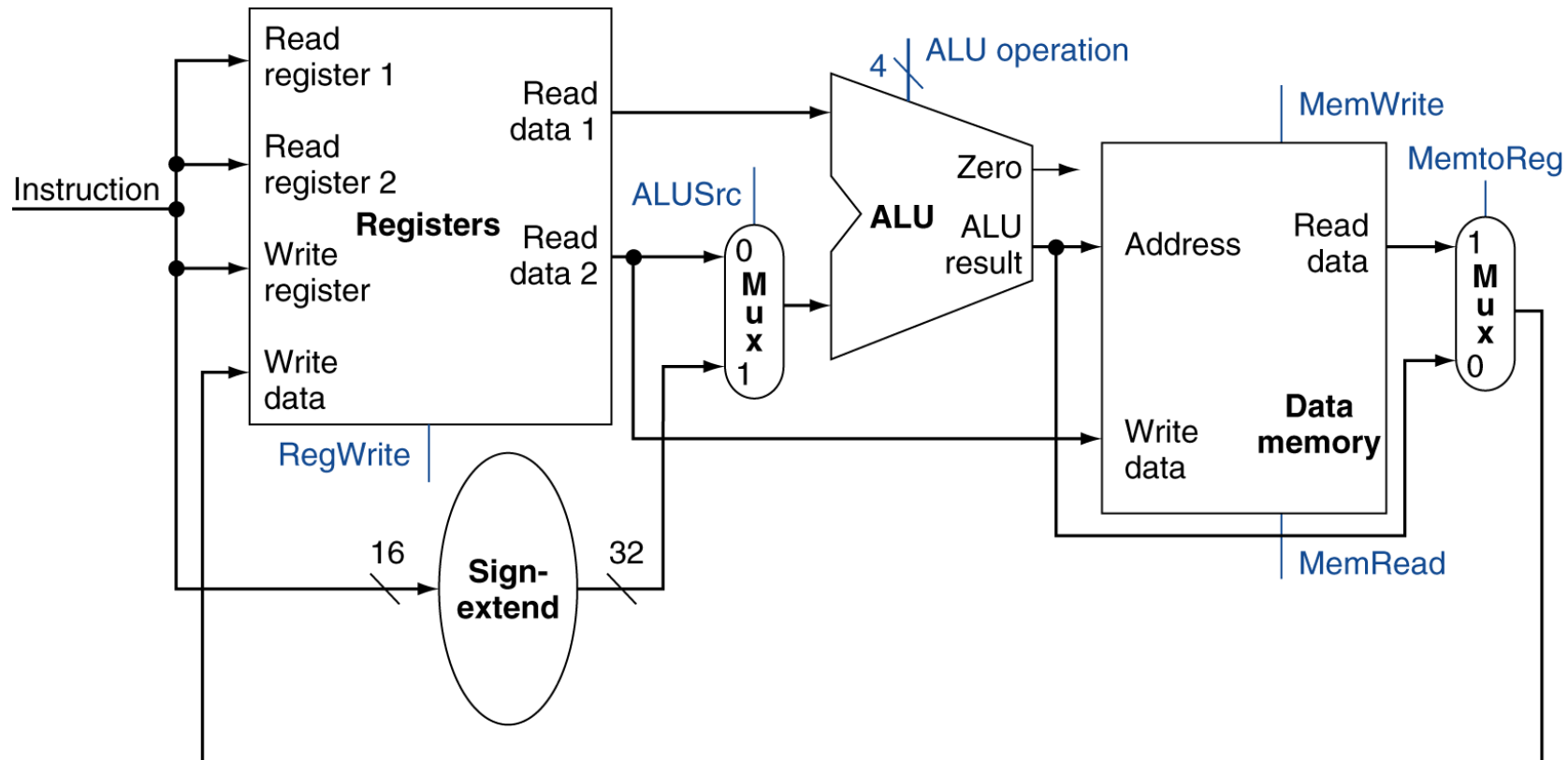
# Add \$t0, \$t0, \$t1

\$t0 is register 8, \$t1 is register 9  
\$t0 holds 5  
\$t1 holds 6



# lw \$t1, 4(\$t0)

\$t0 is register 8, \$t1 is register 9  
\$t0 holds 0x07AB8110  
0x07AB8114 holds 12



# 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

# What do we need to add Conditional Branch Instructions to our datapath?

beq \$t2, \$t3, 0x4F35

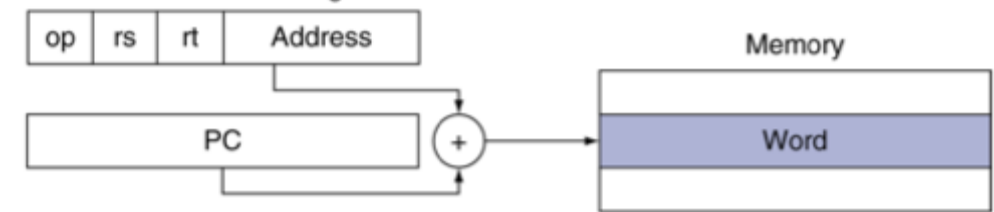
A. ALU

B. Registers and an ALU

C. Registers, ALU and Memory

D. Registers, an ALU and an Adder

4. PC-relative addressing



Read register operands

Compare operands

Use ALU, subtract and check Zero output

Calculate target address

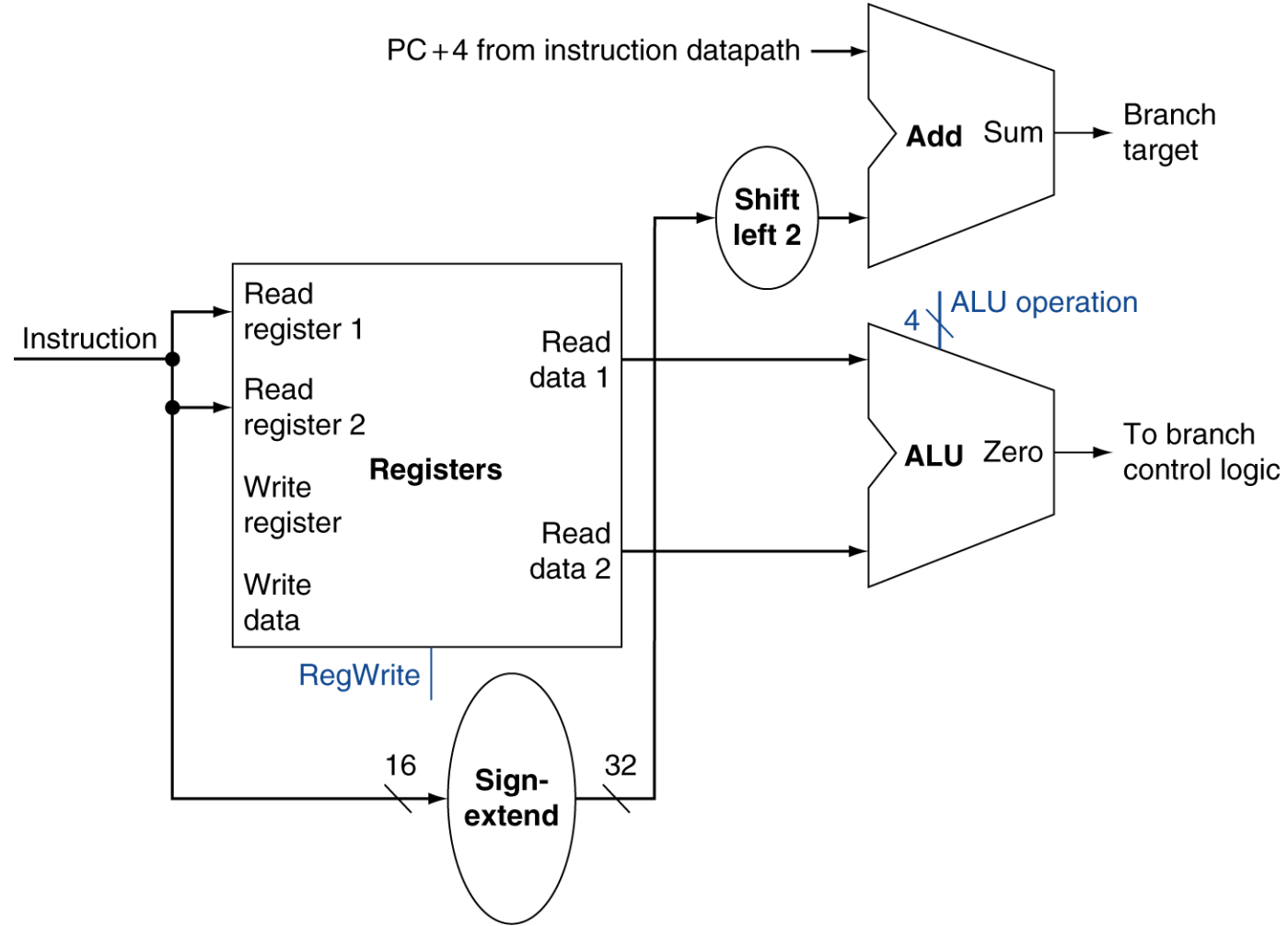
Sign-extend offset

Shift left 2 bits (word offset)

Add to PC + 4

Already calculated during instruction fetch

# Branch Instructions



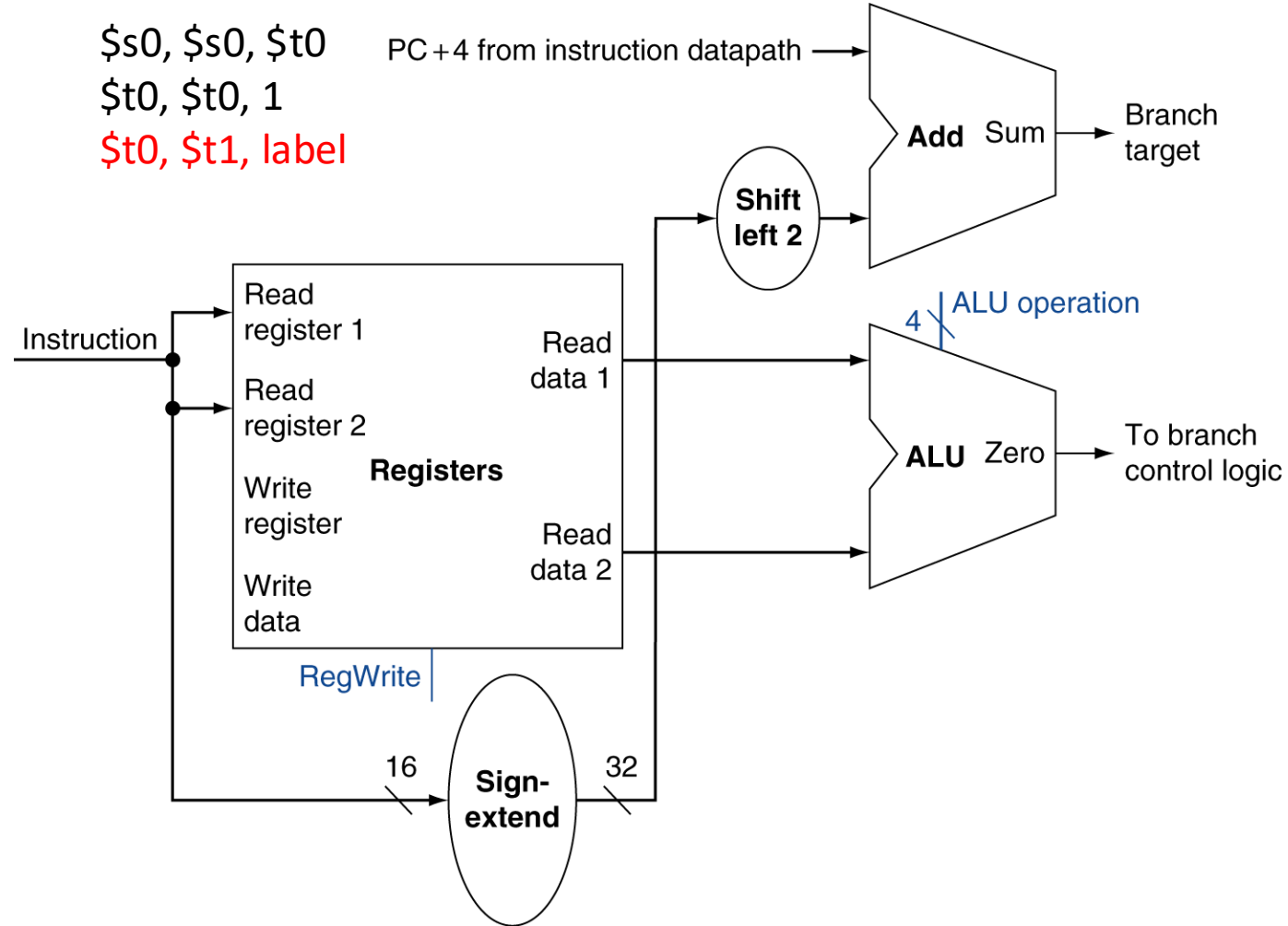


# Branch Instructions

0x4045A130 label: add  
0x4045A134 addi  
0x4045A138 beq

\$s0, \$s0, \$t0  
\$t0, \$t0, 1  
\$t0, \$t1, label

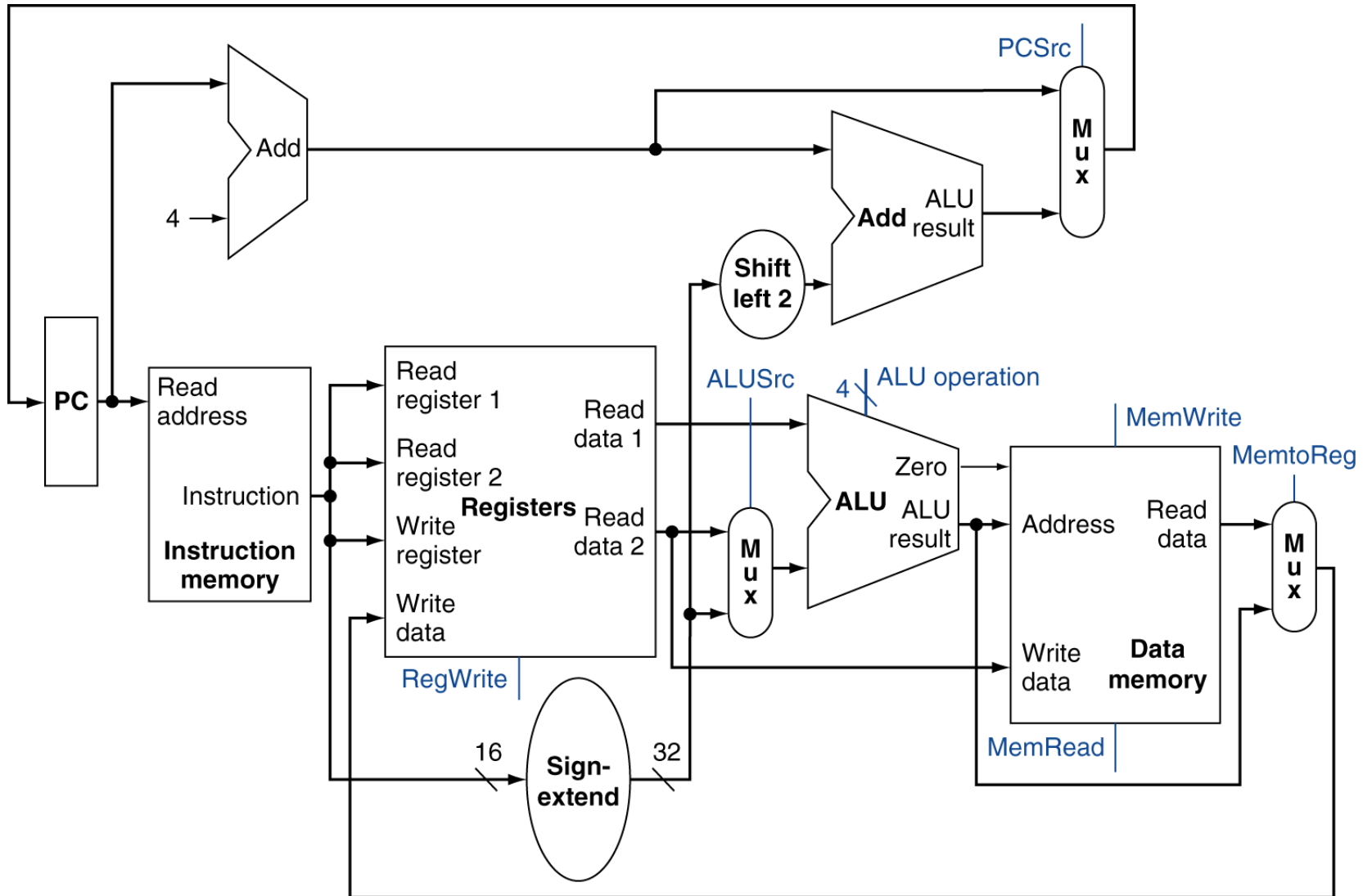
PC + 4 from instruction datapath



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

\$t0 holds 5  
\$t1 holds 5

# Datapath (still simplified a bit)



# Reading

- Next lecture: Data path
  - Section 5.2