CPT_S 260 Intro to Computer Architecture
Lecture 27

Single Cycle MIPS
March 23, 2022

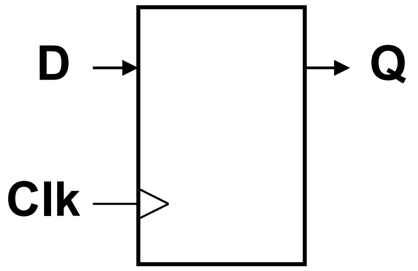
Ganapati Bhat
School of Electrical Engineering and Computer Science
Washington State University

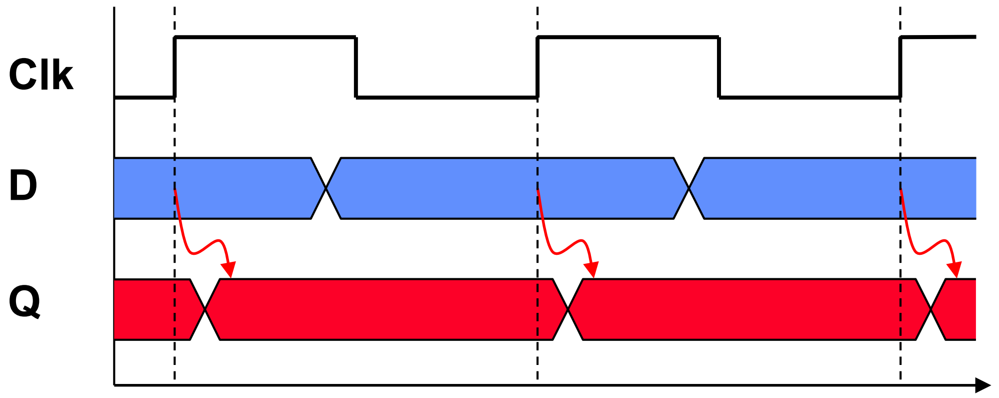
Announcements

- **Mid term exam 2**
  - Friday, April 1, 2022
  - Format will be similar to exam 1
  - Tentative topics: MIPS, digital logic, single cycle MIPS implementation
- **Final exam**
  - May 4, 2022
  - 7:30 pm to 9:30 pm
  - If you’re taking Calculus 3, please reach out to me

Recap: Sequential Elements

- **Flip flops: stores data in a circuit**
  - Uses a clock signal to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1





3

Single Cycle MIPS Processor Design

- **Topics:**
  - Introduction
  - Instruction Execution
  - Use Digital Blocks to Build the Data Path!
    - » Arithmetic/Logic
    - » Load/Store
    - » Branch/Jump
  - Control Unit
    - » ALU Control
    - » Datapath Control
  - Complete Hardware

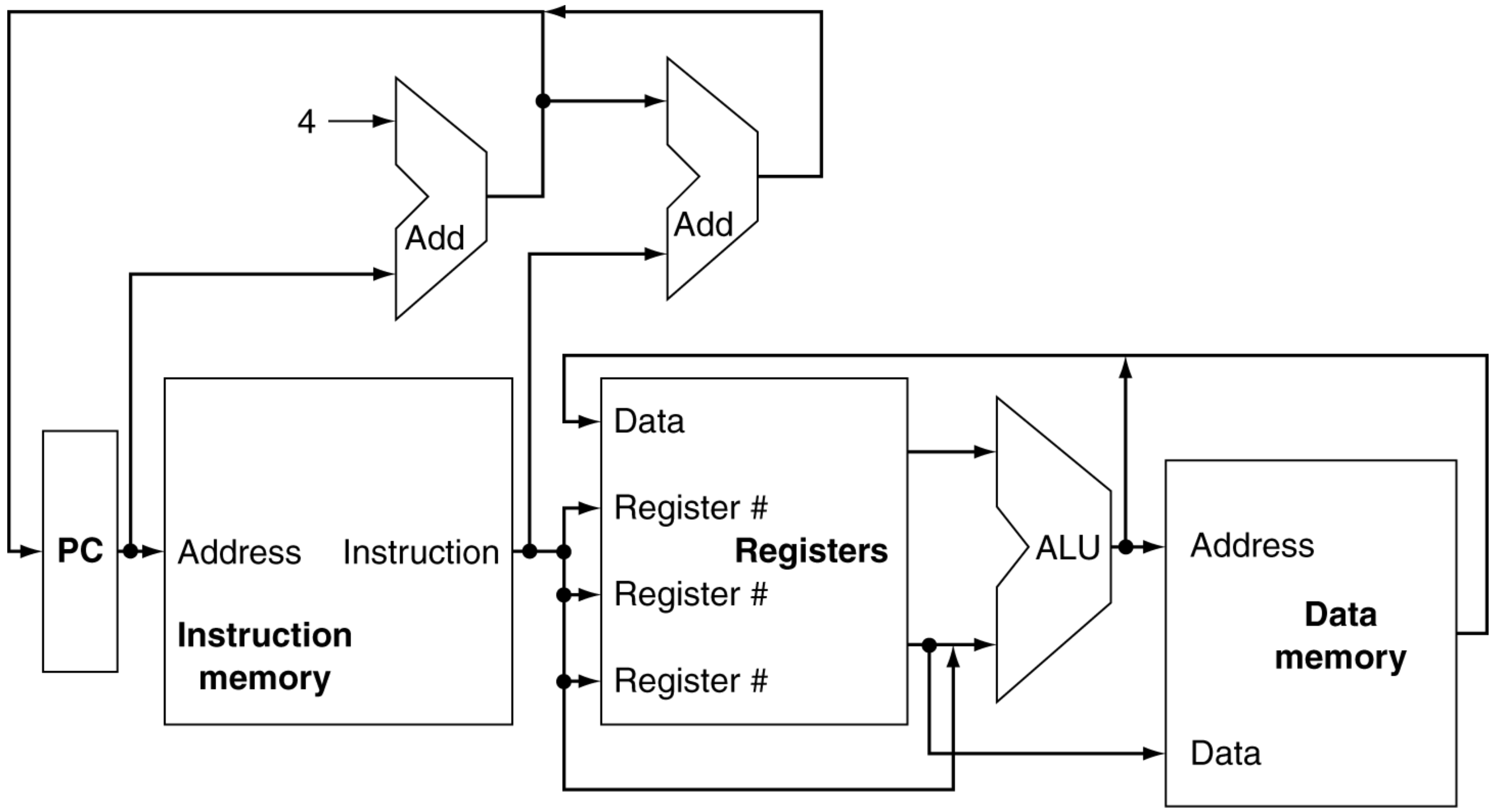
Introduction

- **CPU performance factors**
  - Instruction count
    - Determined by ISA and compiler
  - CPI and Cycle time
    - Determined by CPU hardware
- **We will examine two MIPS implementations**
  - A simplified version
  - A more realistic pipelined version
- **Simple subset, shows most aspects**
  - Memory reference: `lw`, `sw`
  - Arithmetic/logical: `add`, `sub`, `and`, `or`, `slt`
  - Control transfer: `beq`, `j`

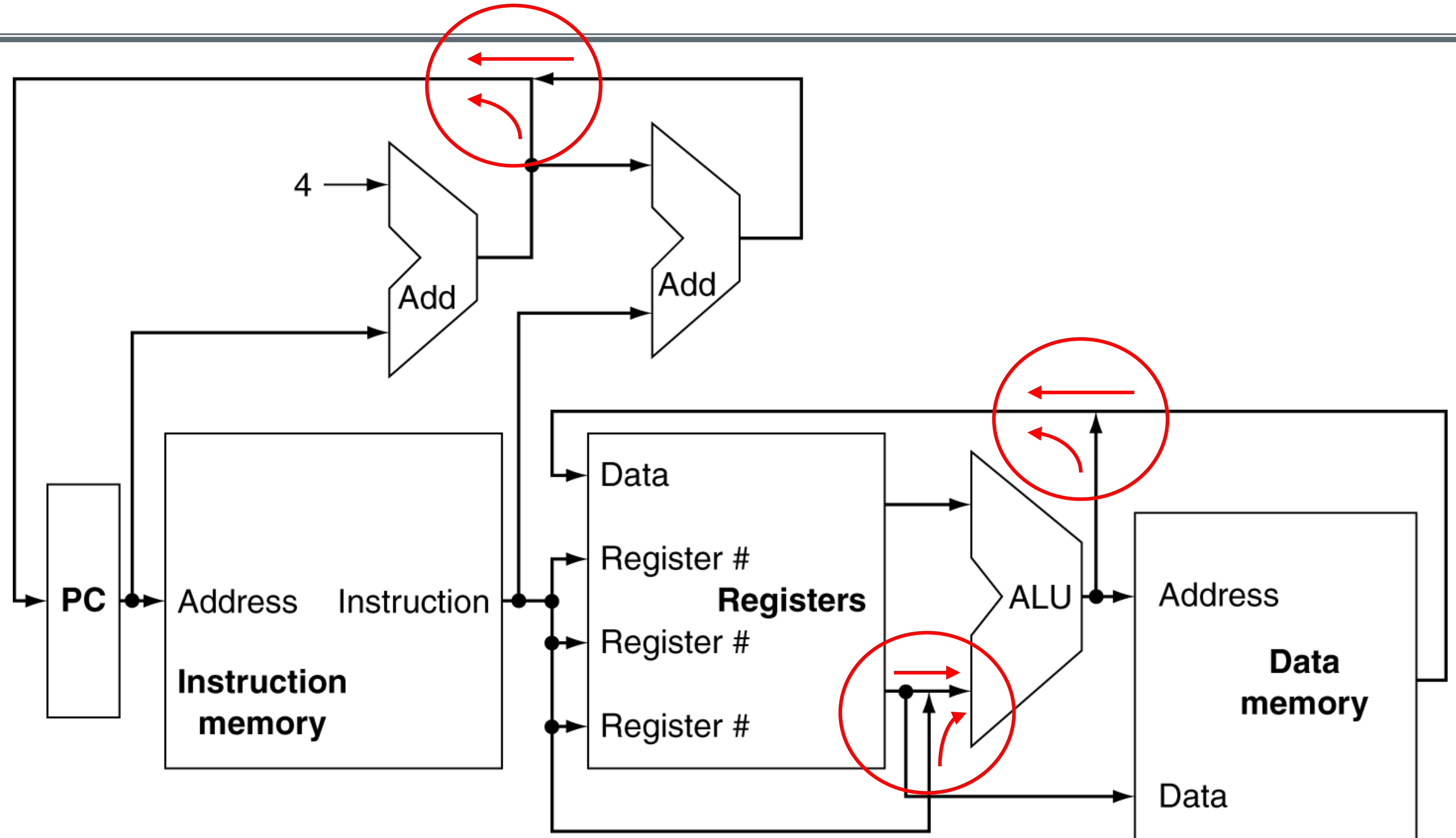
Instruction Execution

- **PC $\rightarrow$ instruction memory, fetch instruction**
- **Register numbers $\rightarrow$ register file, read registers**
- **Depending on instruction class**
  - Use ALU to calculate
    - » Arithmetic result
    - » Memory address for load/store
    - » Branch target address
  - Access data memory for load/store
  - PC $\leftarrow$ target address or PC + 4

CPU Overview

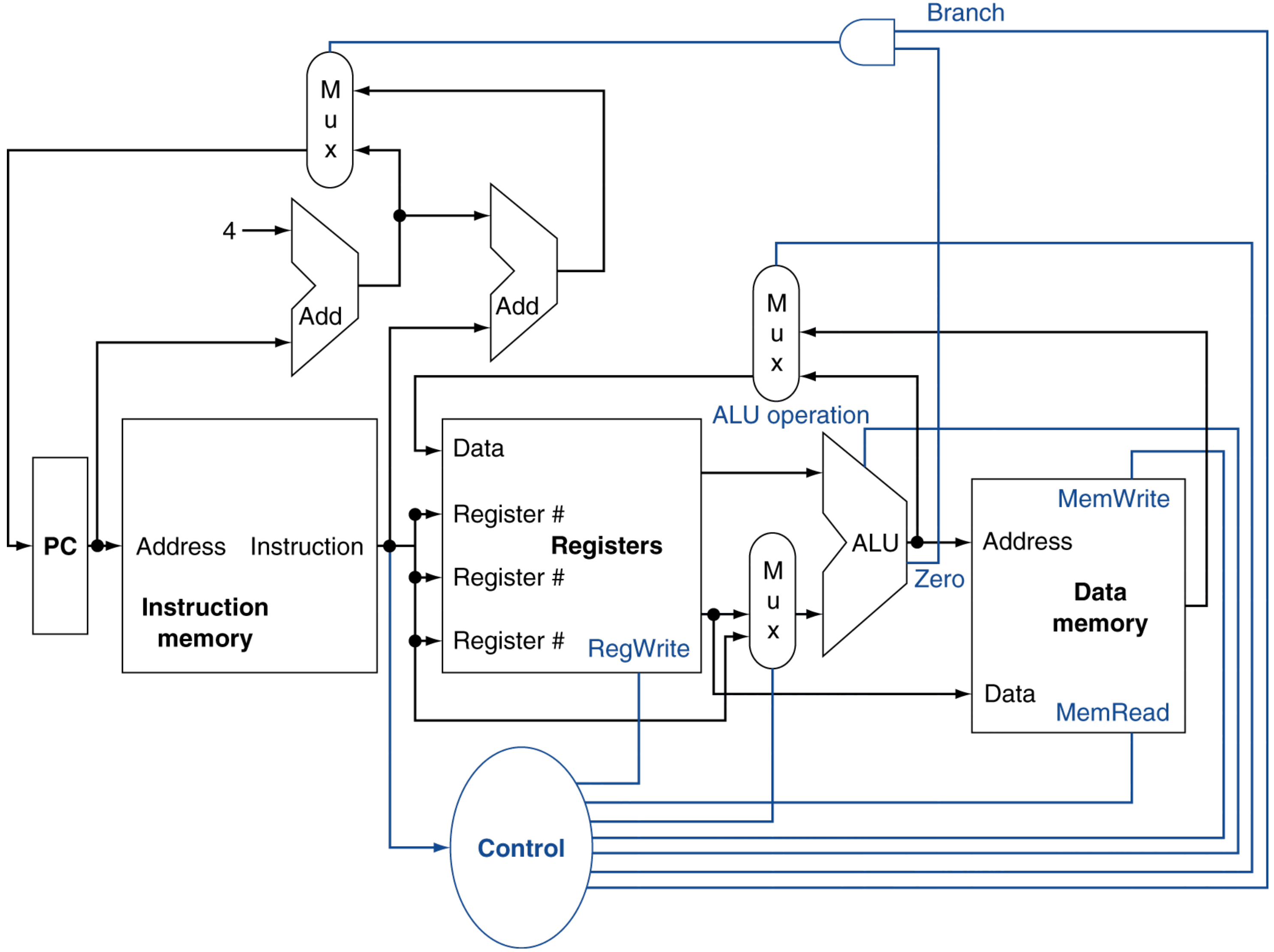


Multiplexers



- Can’t just join wires together
  - Use multiplexers

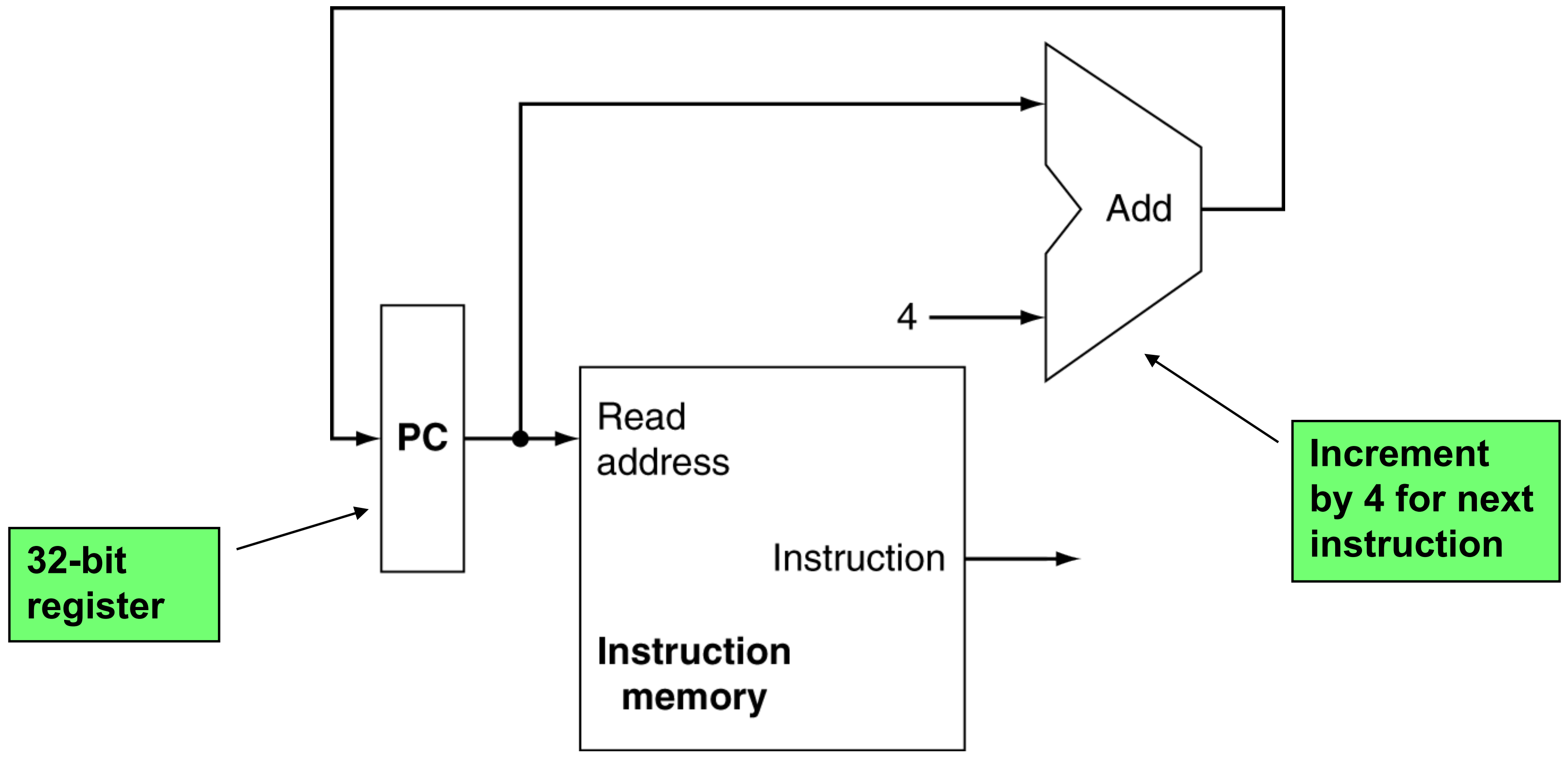
Control


Branch
M u x
4
Add
Add
M u x
ALU operation
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
RegWrite
M u x
ALU
Zero
MemWrite
Address
Data memory
Data
MemRead
Control

Building a Datapath

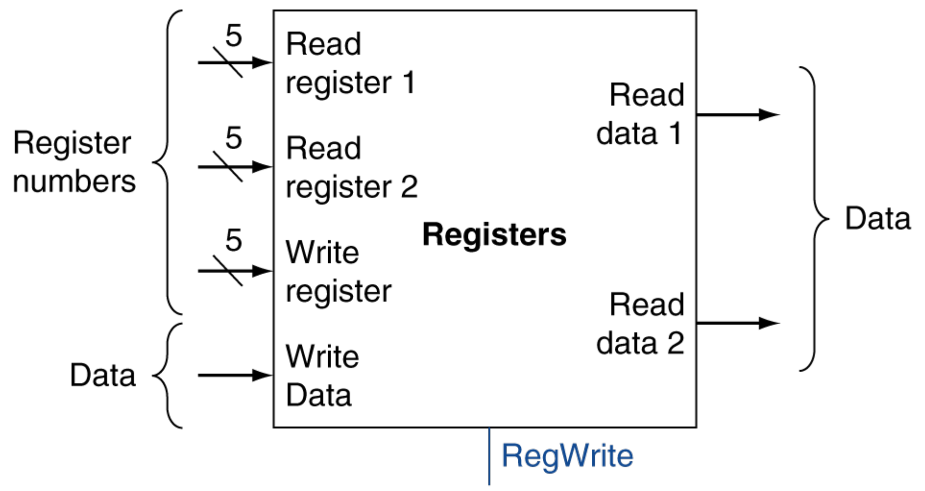
- **Datapath**
  - Elements that process data and addresses in the CPU
    - » Registers, ALUs, mux’s, *memories*, …
- **We will build a MIPS datapath incrementally**
  - Refining the overview design

Instruction Fetch

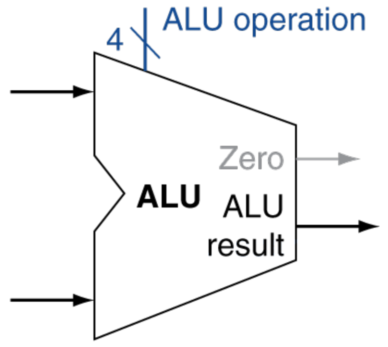


R-Format Instructions

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



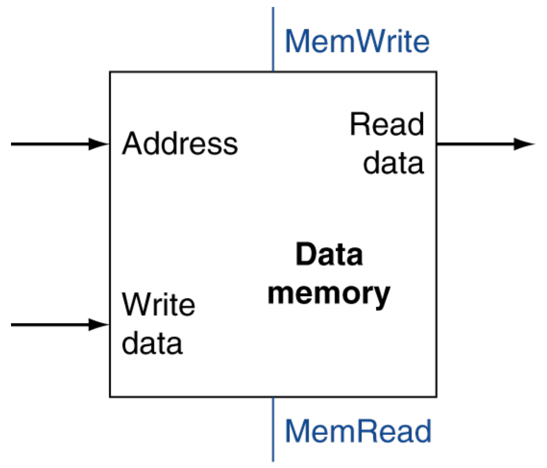
a. Registers



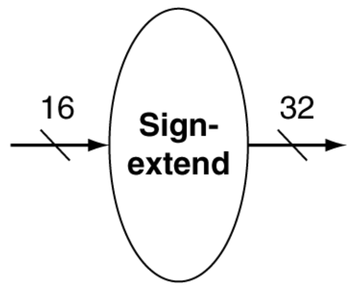
b. ALU

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

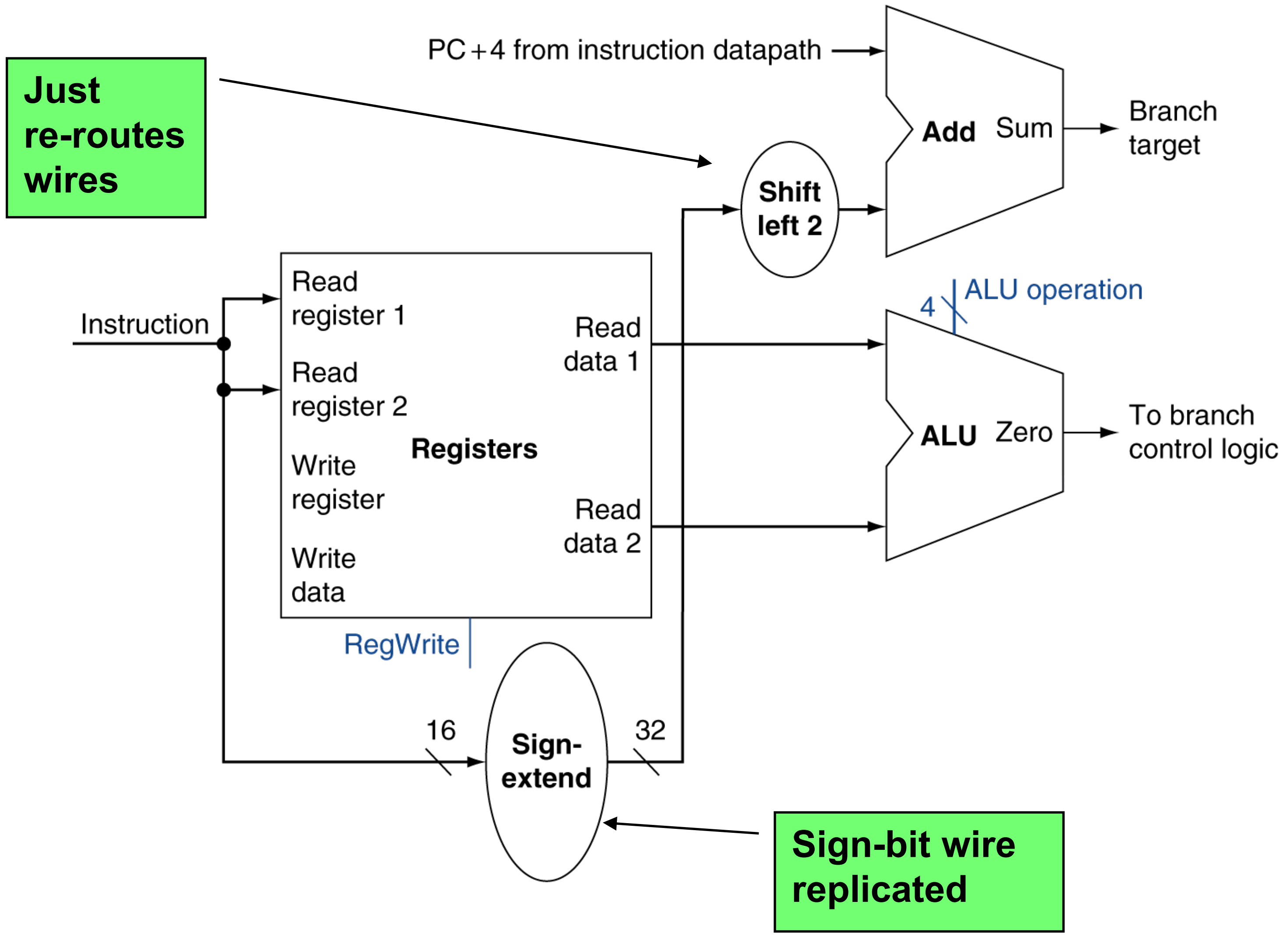


b. Sign extension unit

Branch Instructions

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

Branch Instructions



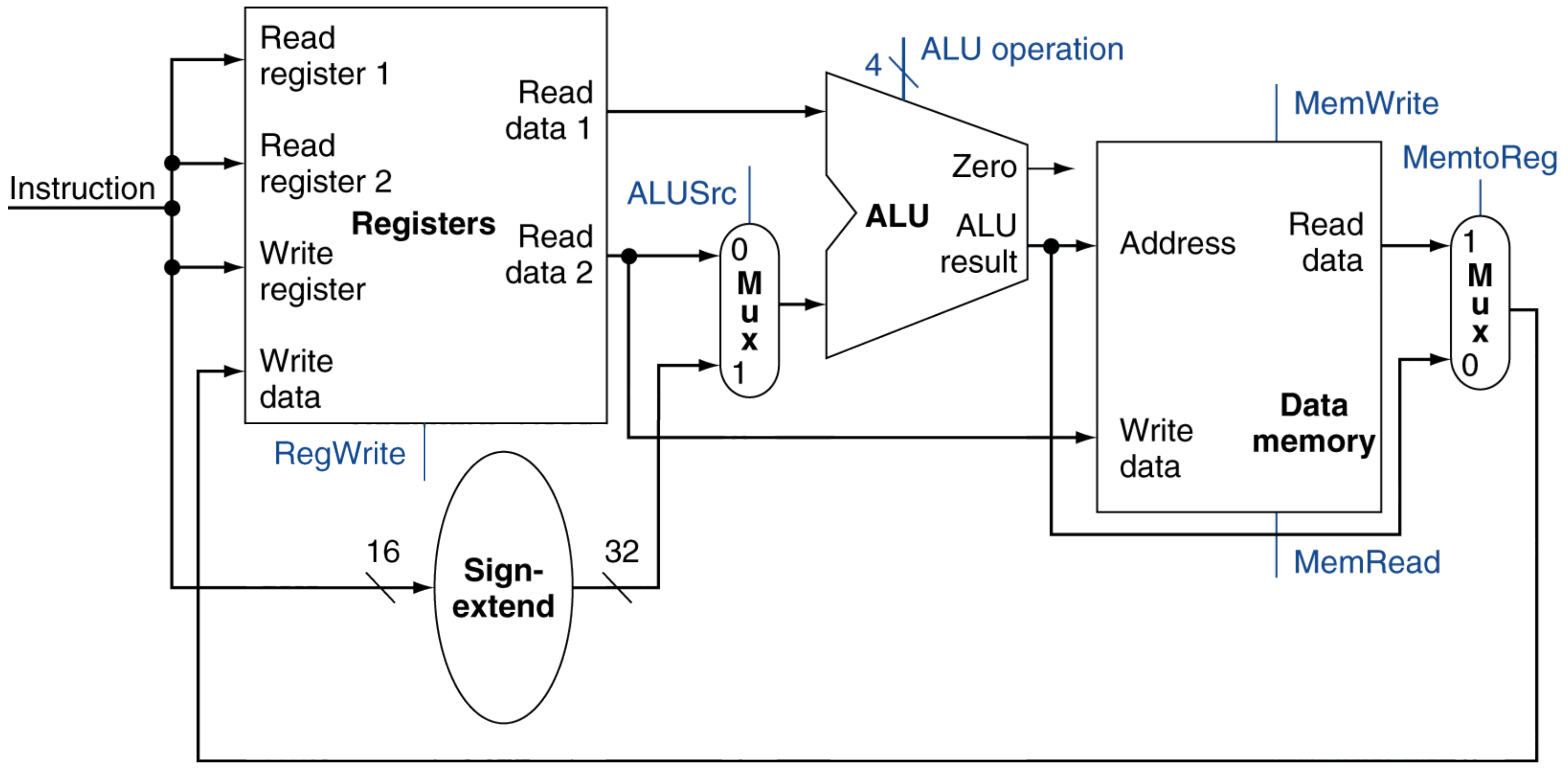
Composing the Elements

- **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 alternate data sources are used for different instructions**

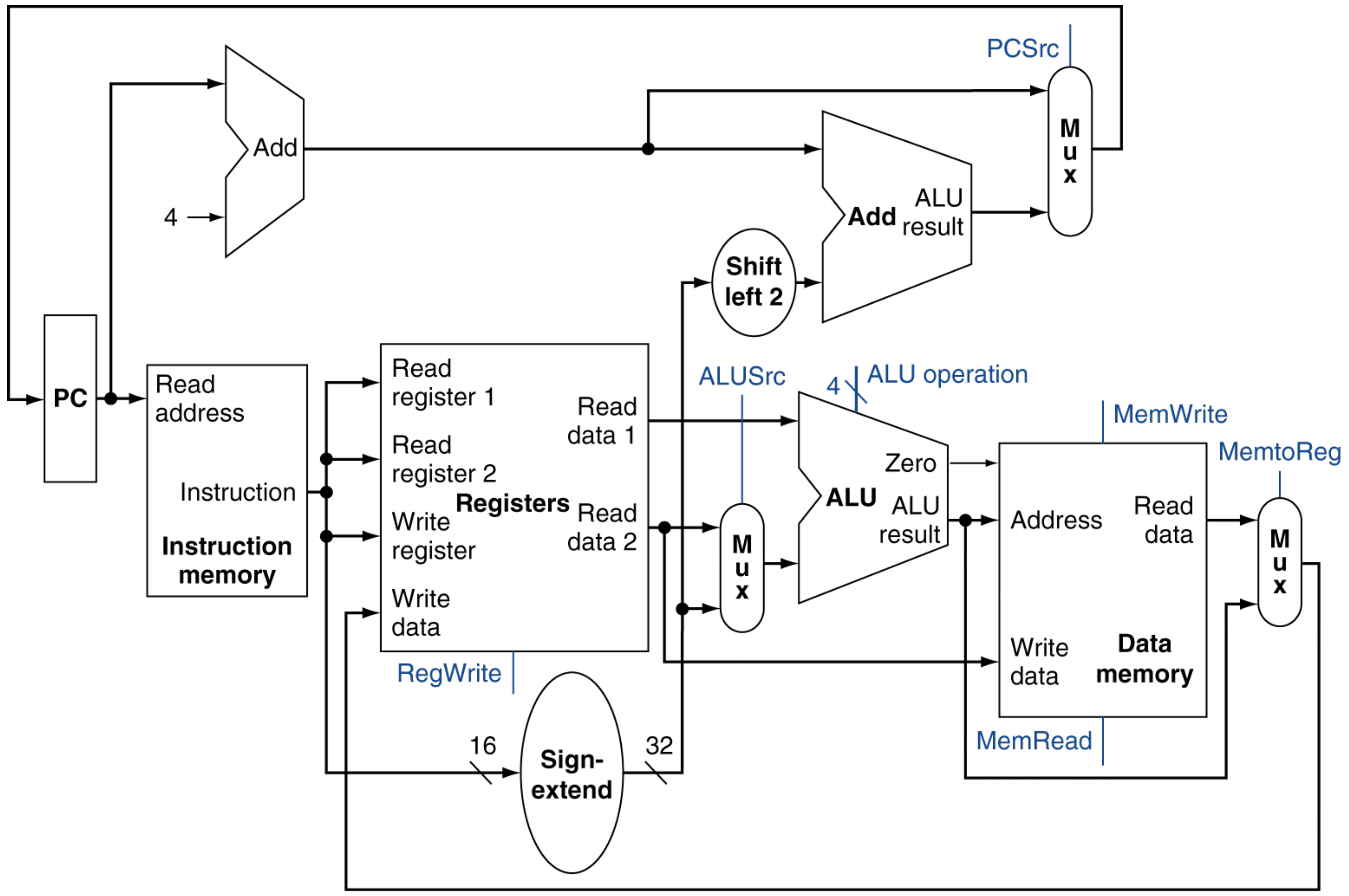
R-type/ Load-Store Datapath

- **Operations of arithmetic-logical (or R-type) instructions and the memory instructions are quite similar**
- **The arithmetic-logical instructions use the ALU, with the inputs coming from the two registers**
- **The memory instructions can also use the ALU to do the address calculation**
  - Second input is the sign-extended 16-bit off set field from the instruction
- **The value stored into a destination register comes from the ALU (for an R-type instruction) or the memory (for a load).**

R-Type/Load/Store Datapath



Full Datapath


PCSrc
Add
4
M u x
Add
ALU result
Shift left 2
PC
Read address
Instruction
Instruction memory
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
RegWrite
16
Sign-extend
32
ALUSrc
M u x
4
ALU operation
Zero
ALU
ALU result
MemWrite
MemtoReg
Address
Read data
Write data
Data memory
MemRead
M u x