Discussion

- Explain what is the meant by the design principal “make the common case fast.” Considering MIPS processor, identify a design choice that uses this principal. Identify a special case (an uncommon case) and explain how MIPS handle it.
- Floating point number representation involves splitting the data unit (e.g., word) into multiple fields. What are these fields? How the stored value is calculated? How would changing the size of these fields affect the number precision and range?
- The principal of “performance by prediction” is used in computer design. Identify one case for which this principal is used to improve the performance. Explain how this principal is used to improve the performance.
- It is well-known that the design of computer has a cost-performance tradeoff. Identify two scenarios that confirm this tradeoff and explain the tradeoff aspects.

The Processor

Dr. Ahmed Zahran

WGB 182

a.zahran@cs.ucc.ie

How a processor is designed?

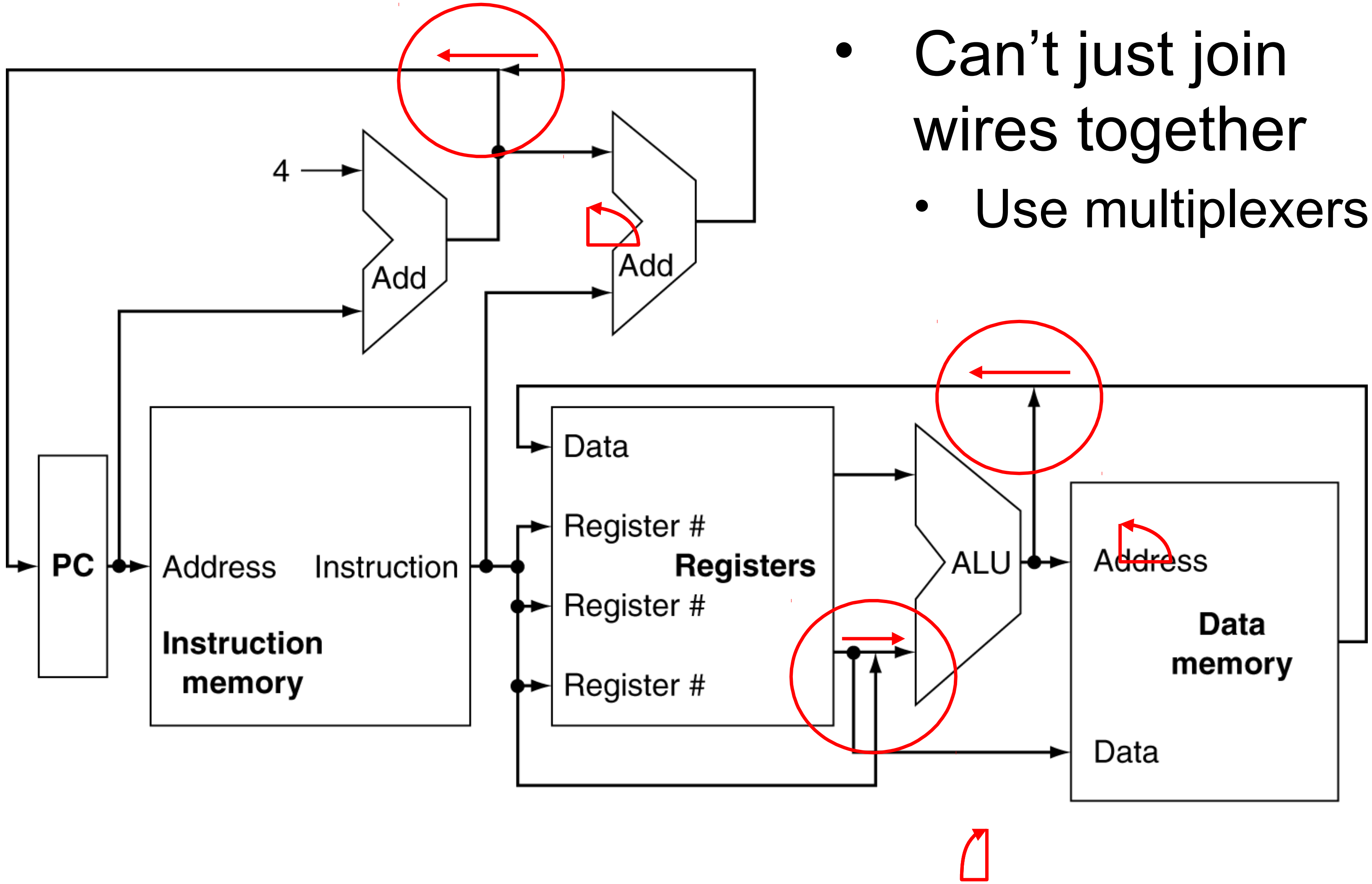
- Processors are designed to execute binary (machine) instructions
- Simple subset, shows most aspects
    - Memory reference: lw, sw
    - Arithmetic/logical: add, sub, and, or, slt
    - Control transfer: beq, j
- We will examine two MIPS implementations
    - A simplified version
    - A pipelined version

Instruction Execution

- Two basic steps for all instructions

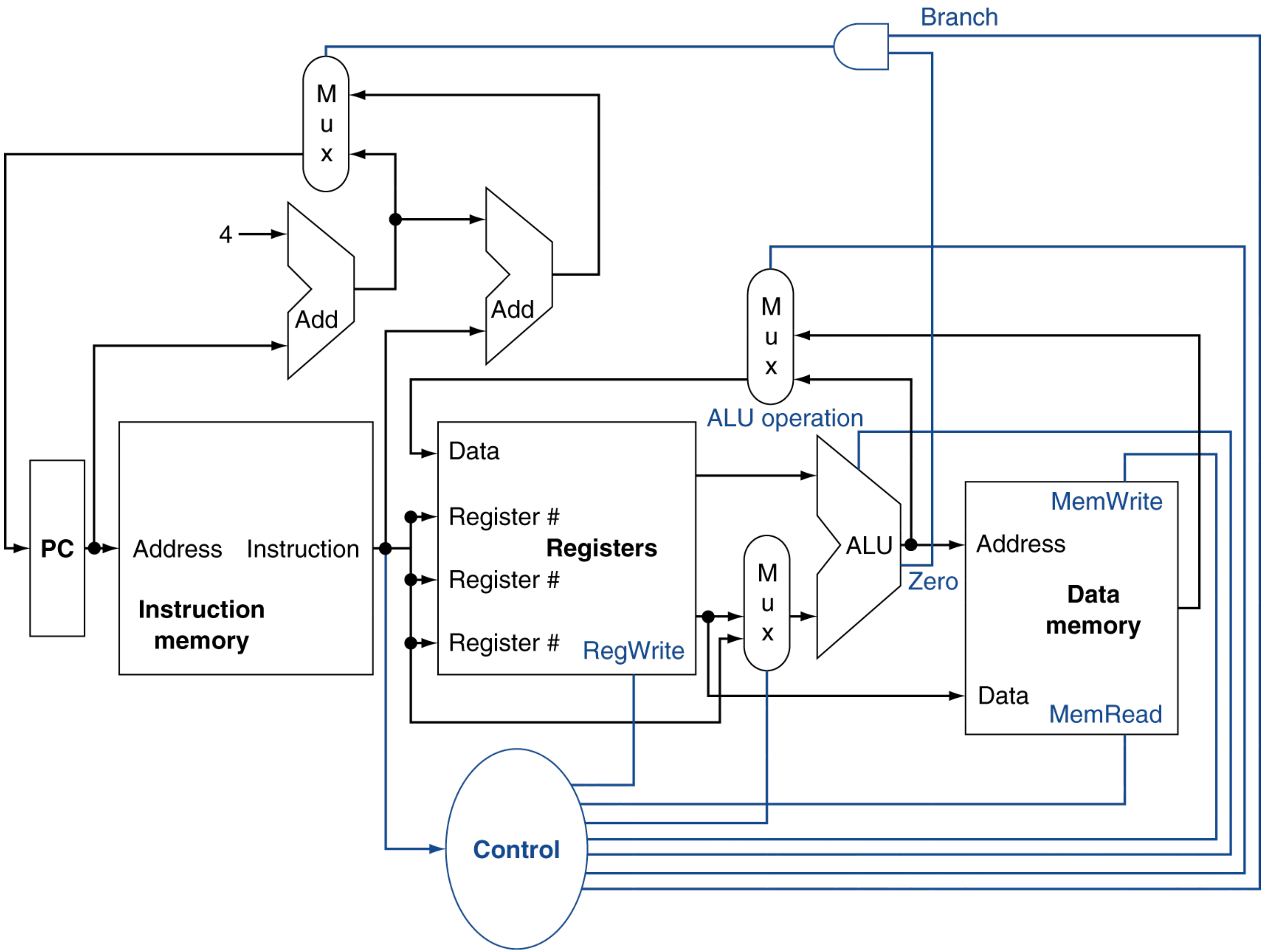
1. Instruction fetch
    - PC → instruction memory,
    - PC ← target address or PC + 4
2. Execution
    - Read and/or write registers
        - Register numbers → register file,
    - Use ALU to calculate
        - Arithmetic result
        - Memory address for load/store
        - Branch target address

CPU Overview



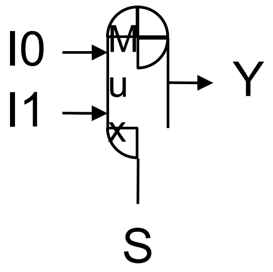
- Can’t just join wires together
  - Use multiplexers

Control

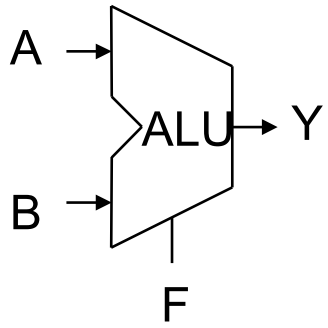

Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
Data
Register #
Registers
Register #
Register #
RegWrite
PC
Address
Instruction
Instruction
memory
M
u
x
ALU
Zero
MemWrite
Address
Data
memory
Data
MemRead
Control

Processor Components

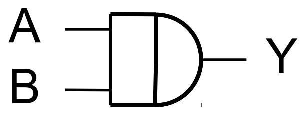
- Multiplexer
  - Y = S ? I1 : I0



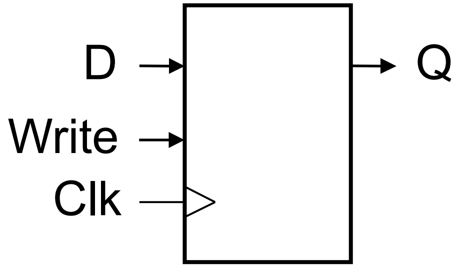
- Arithmetic/Logic Unit
  - Y = F(A, B)

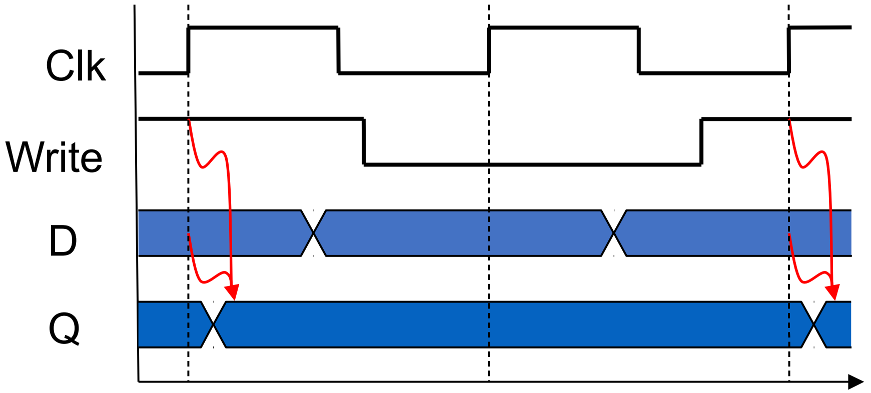


- Logic-gate
  - Y = A & B



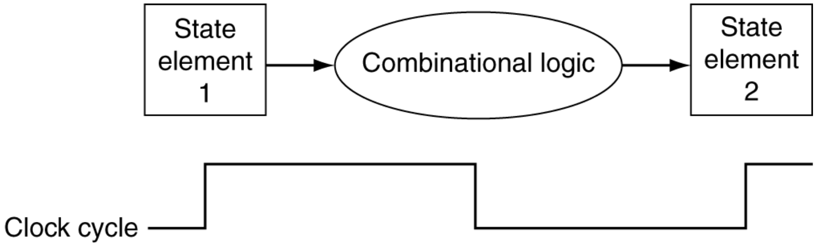
- Sequential elements

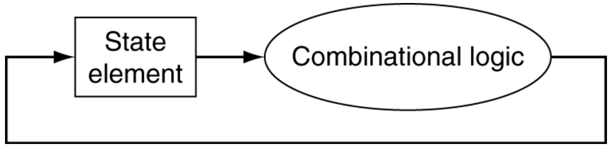




Clocking Methodology

- Combinational logic transforms data during clock cycles
    - Between clock edges
    - Input from state elements, output to state element
    - Longest delay determines clock period



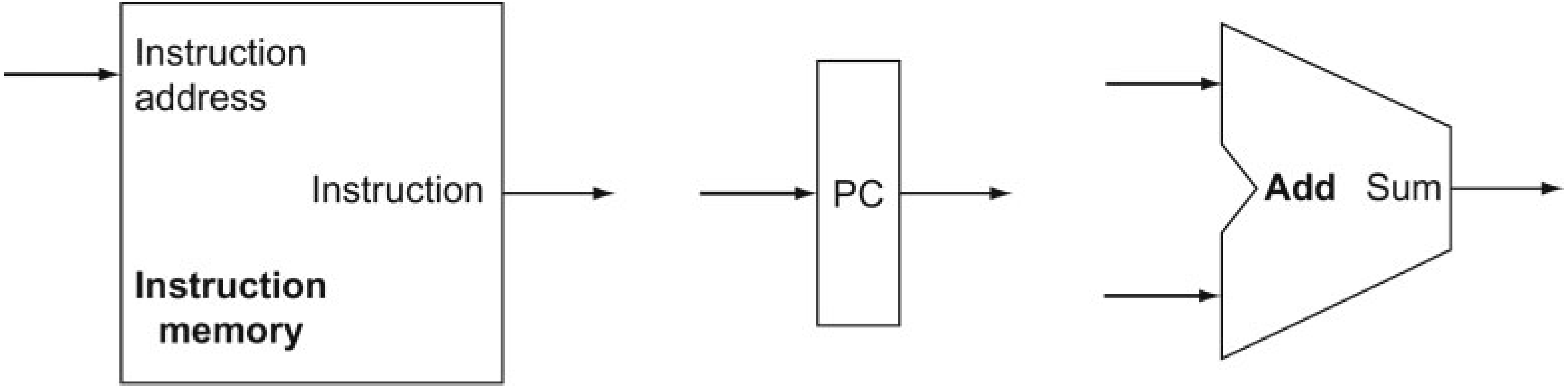


MIPS Datapath

Building a Datapath

- Datapath
  - Components that process data and addresses in the CPU
    - Registers, ALUs, mux’s, memories, …
- We will build a MIPS datapath incrementally
  - Identify and connect logic components required by different processor functions
    - Instruction fetch
    - Executing instructions

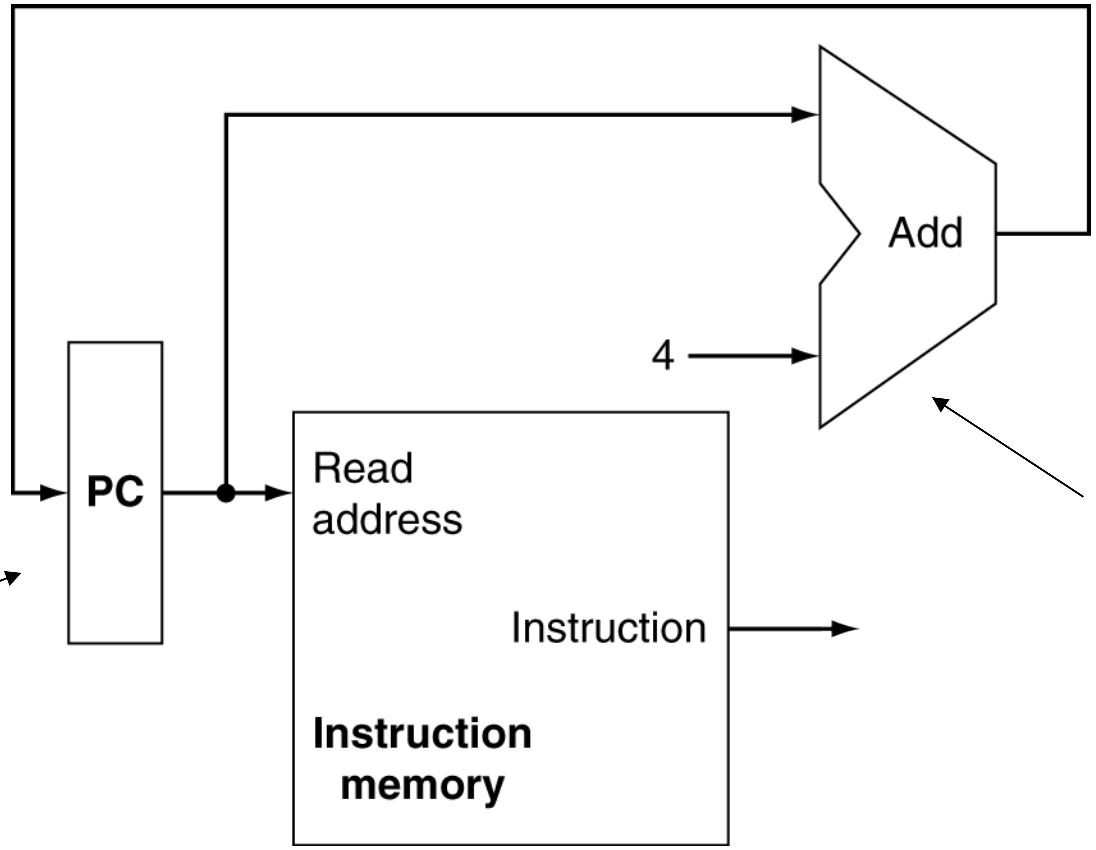
Instruction Fetch



a. Instruction memory b. Program counter c. Adder

Requires two state elements
Instruction memory and PC

Instruction Fetch



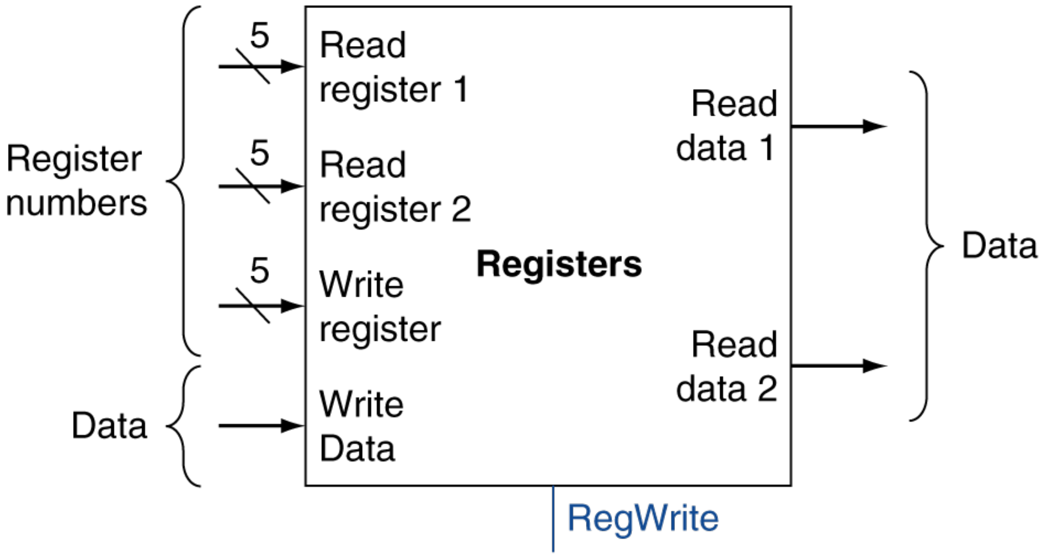
32-bit register

Increment by 4 for next instruction

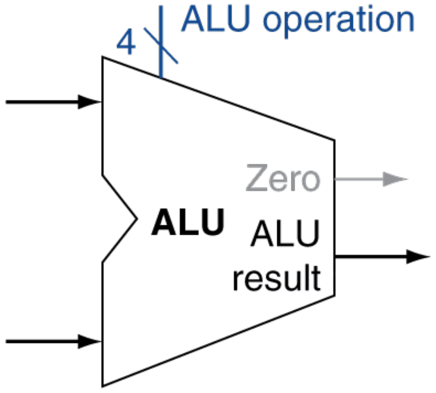
R-Format Instructions

0	rs	rt	rd	shamt	funct
31:26	25:21	20:16	15:11	10:6	5:0

- Read two register operands
- Perform arithmetic/logical operation



a. Registers

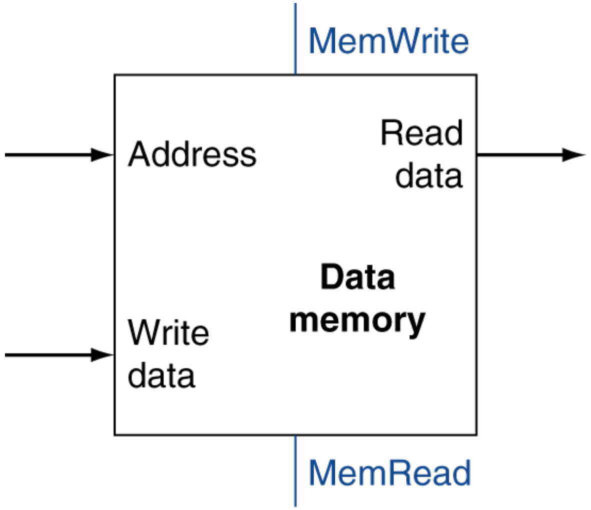


b. ALU

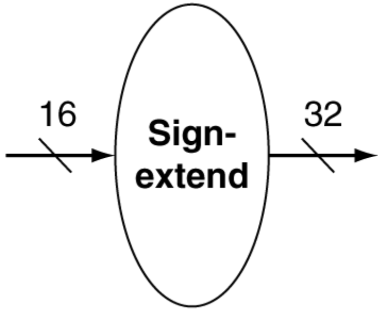
Load/Store Instructions

35 or 43	rs	rt	address
31:26	25:21	20:16	15:0

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



a. Data memory unit

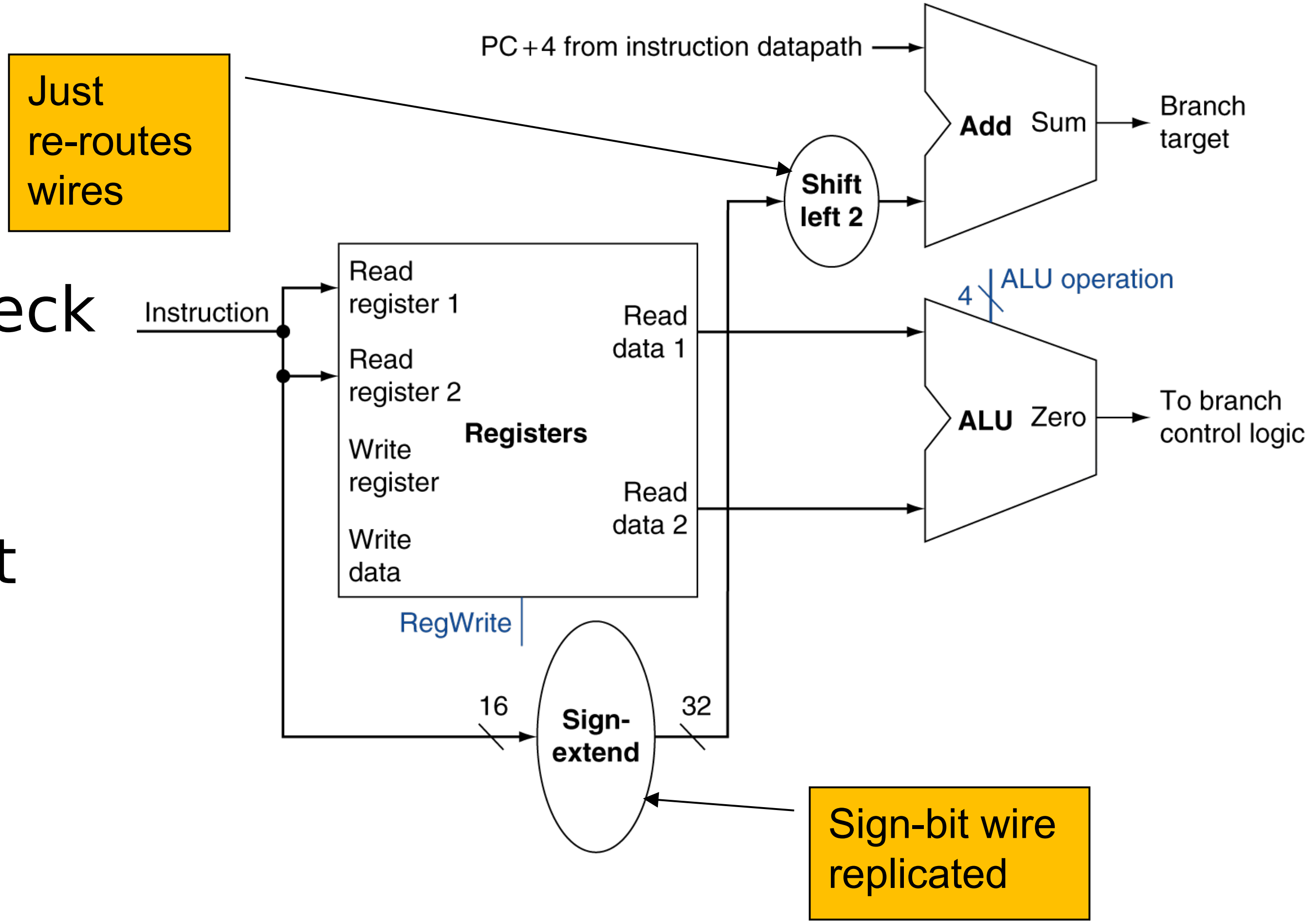


b. Sign extension unit

Branch Instructions

35 or 43	rs	rt	address
31:26	25:21	20:16	15:0

- 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


Just re-routes wires

Sign-bit wire replicated

Summary