Computer Arithmetic: Subword Parallelism
The Processor

Dr. Vincent C. Emeakaroha

13-02-2017

vc.emeakaroha@cs.ucc.ie

Subword Parallellism

- Designed to address multimedia applications
- Graphics and audio applications can take advantage of performing simultaneous operations on short vectors using parallelism
  - Example: 128-bit adder:
    - Sixteen 8-bit adds
    - Eight 16-bit adds
    - Four 32-bit adds
- Also called data-level parallelism, vector parallelism, or Single Instruction, Multiple Data (SIMD)

x86 FP Architecture

- Originally based on 8087 FP coprocessor
  - Used as a push-down stack
  - Registers indexed from TOS: ST(0), ST(1), …
- Intel provided 128-bit extension register for FP operations
  - Four single precision
  - Two double precision
  - Support simultaneous arithmetic operations
- In 2011 Intel doubles to 254-bit register width
  - Single operation can support
    - Eight 32-bit
    - Four 64-bit

Right Shift and Division Pitfalls

- Left shift by *i* places multiplies an integer by 2^i
- Right shift divides by 2^i?
  - Only for unsigned integers
- For signed integers
  - Arithmetic right shift: replicate the sign bit
  - e.g., –5 / 4
    - $1111\ 1011_2$ >> 2 = $1111\ 1110_2$ = –2
    - Instead of -1

Associativity Pitfall

- Parallel programs may interleave operations in unexpected orders
  - Assumptions of associativity may fail

		(x+y)+z	x+(y+z)
x	-1.50E+38		-1.50E+38
y	1.50E+38	0.00E+00	
z	1.0	1.0	1.50E+38
		1.00E+00	0.00E+00

- Need to validate parallel programs under varying degrees of parallelism
- Integer operations should be handled differently as floating-point

Who Cares About FP Accuracy?

- Important for scientific code
  - But for everyday consumer use?
    - "My bank balance is out by 0.0002¢!" ☹
- The Intel Pentium FDIV bug in 1994
  - The market expects accuracy
  - Costed $500 Million to correct
  - No Christmas bonus for Intel Engineers that year.

The Processor

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`

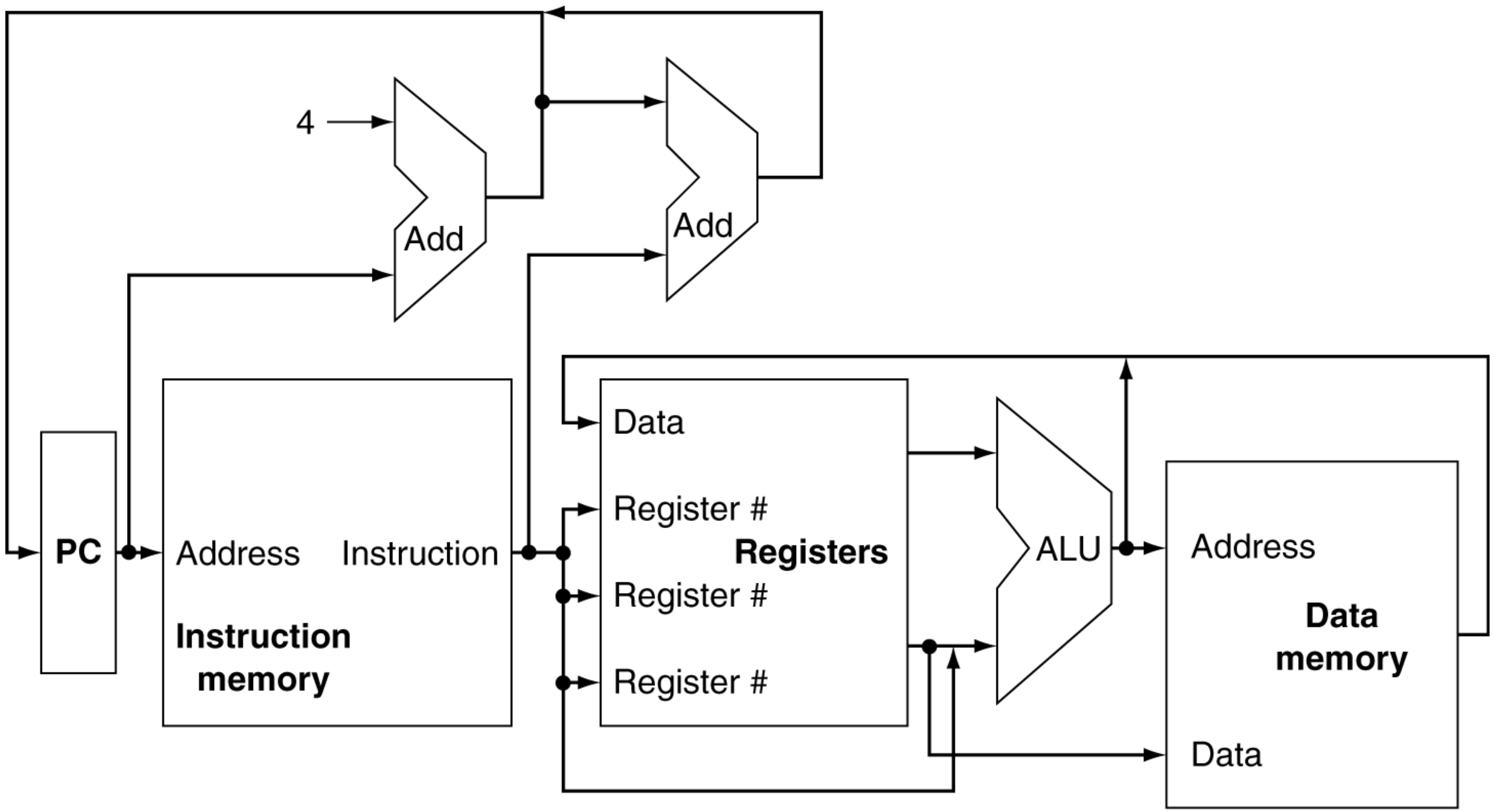
Instruction Execution

- Two basic steps for all instructions

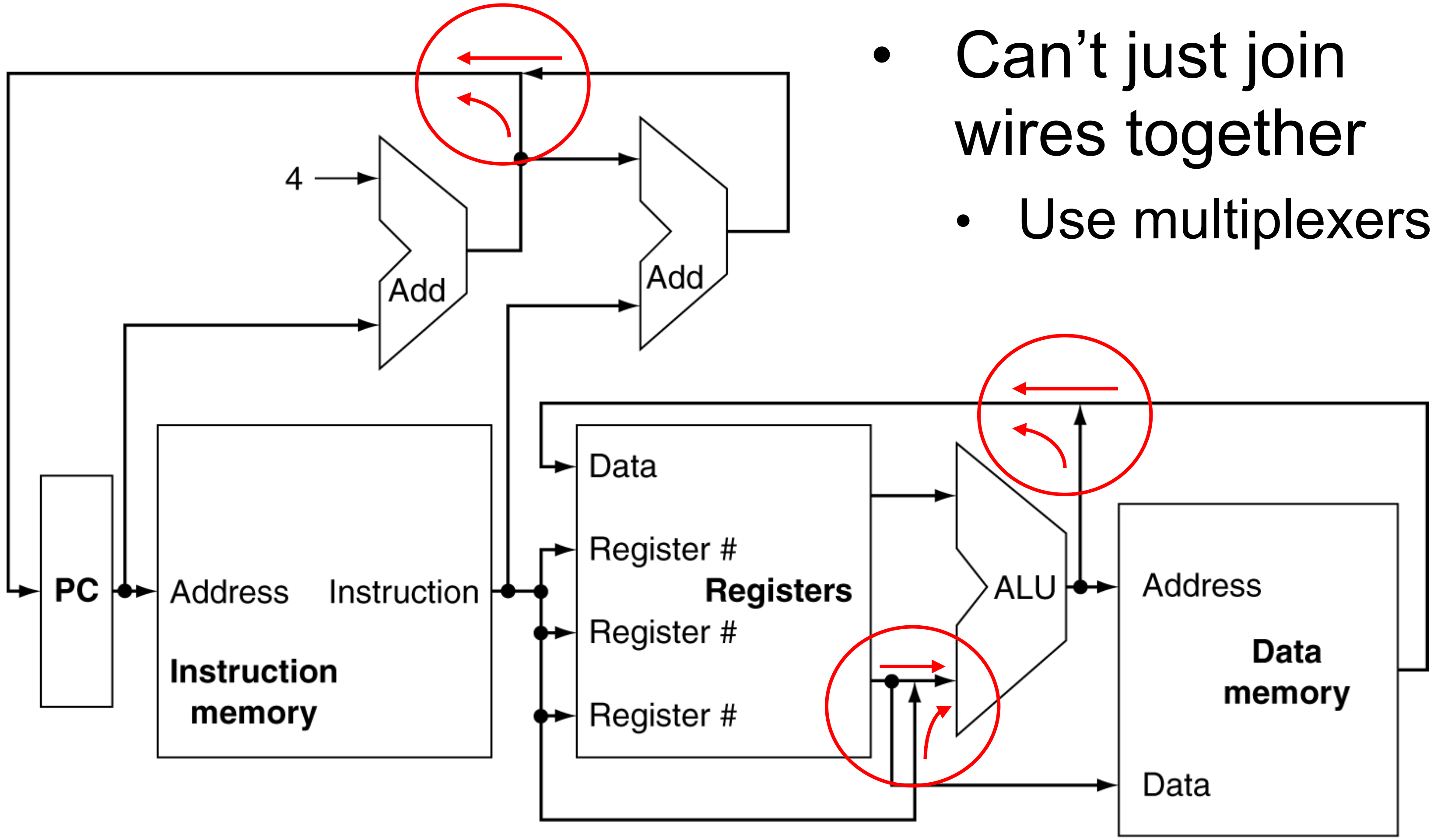
1. PC $\rightarrow$ instruction memory,
   - fetch instruction
2. Register numbers $\rightarrow$ register file,
   - read or write registers

- 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

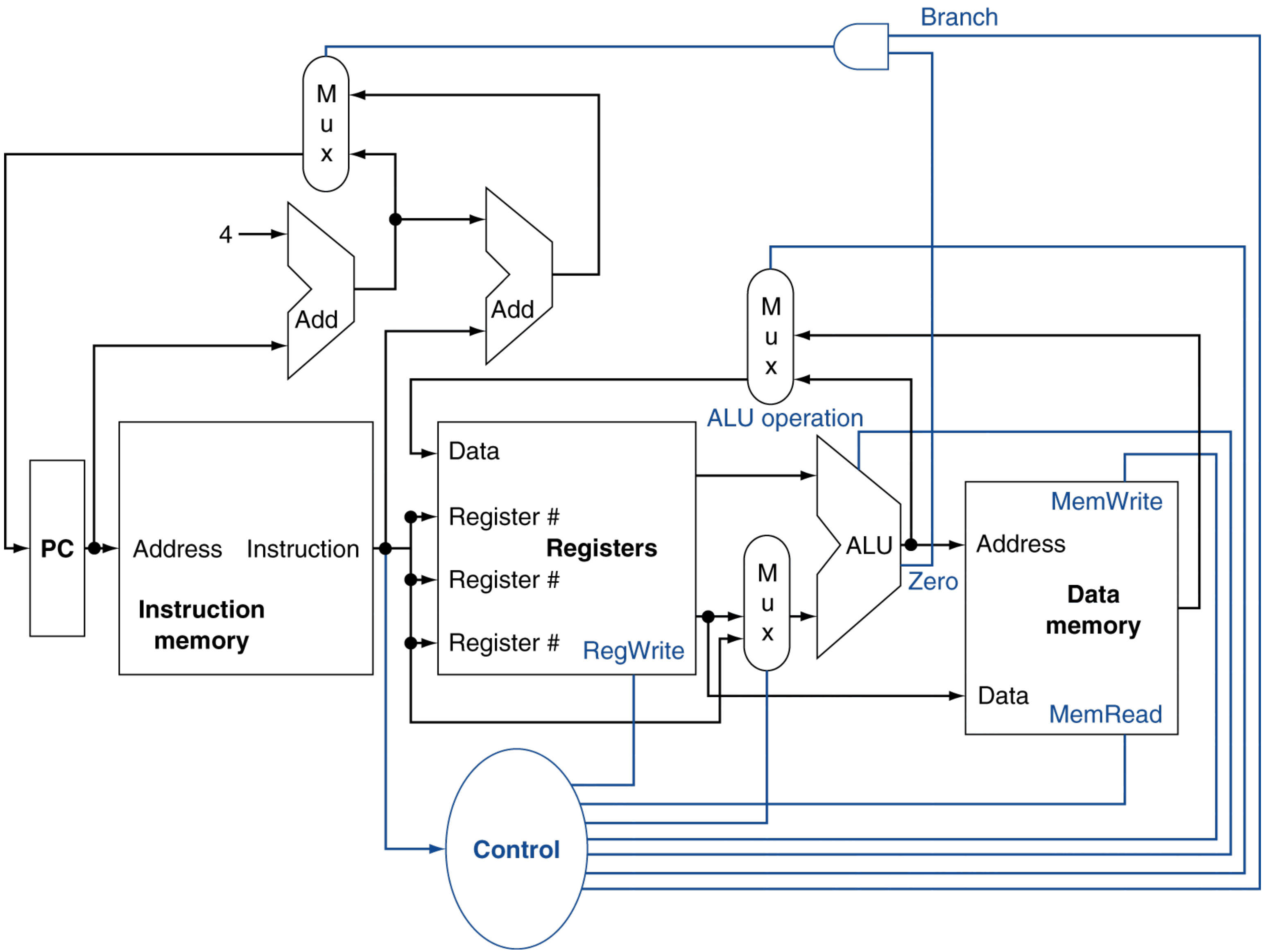


Multiplexers



- Can't just join wires together
  - Use multiplexers

Control



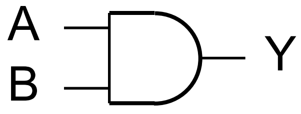
Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
Data
Register #
Registers
Register #
Register #
RegWrite
PC
Address
Instruction
Instruction
memory
M
u
x
ALU
Zero
MemWrite
Address
Data
memory
Data
MemRead
Control

Logic Design Basics

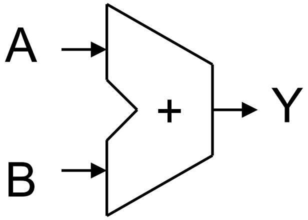
- Information encoded in binary
  - Low voltage = 0, High voltage = 1
  - One wire per bit
  - Multi-bit data encoded on multi-wire buses
- Combinational element
  - Operate on data
  - Output is a function of input
- State (sequential) elements
  - Store information

Combinational Elements

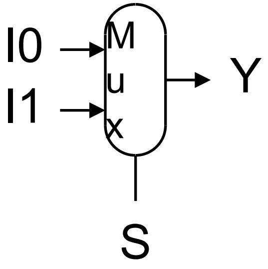
- AND-gate
  - Y = A & B



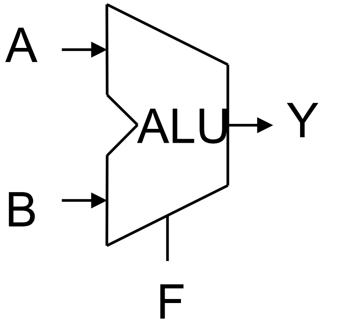
- Adder
  - Y = A + B



- Multiplexer
  - Y = S ? I1 : I0

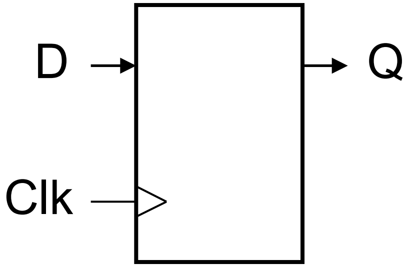


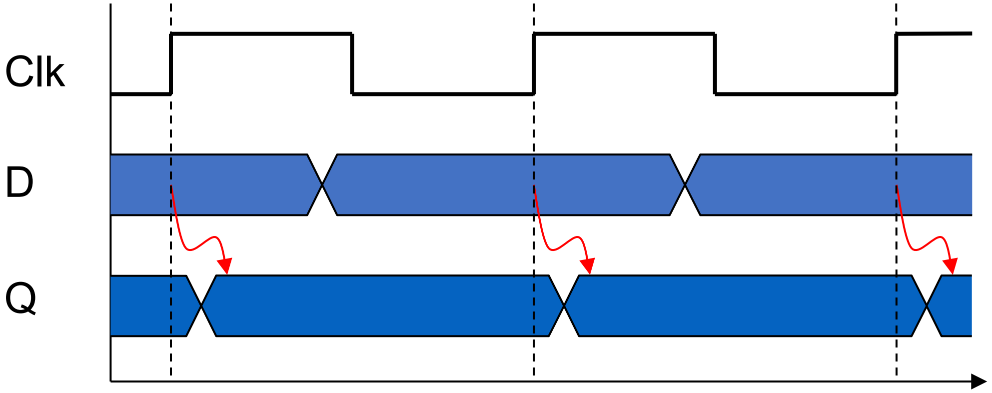
- Arithmetic/Logic Unit
  - Y = F(A, B)



Sequential Elements

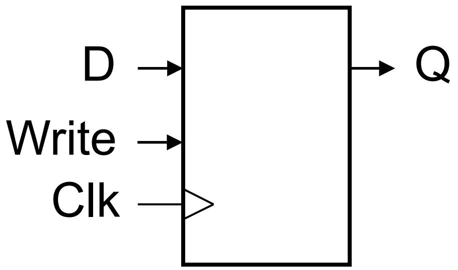
- Register: stores data in a circuit
  - Uses a clock signal to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1

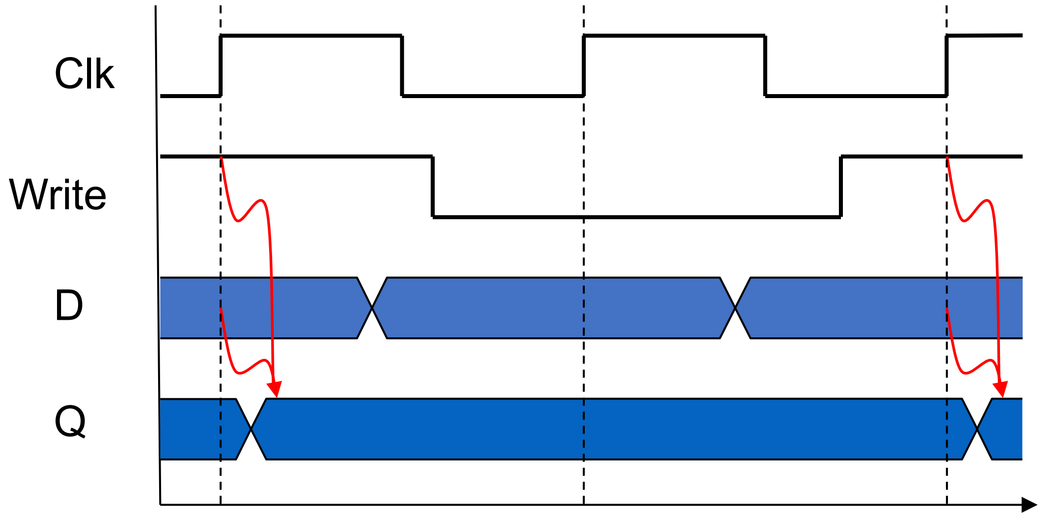




Sequential Elements

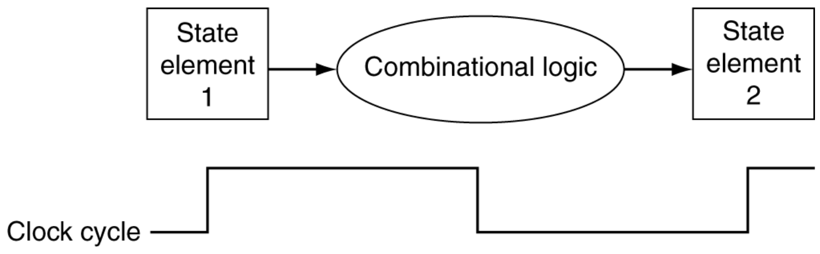
- Register with write control
  - Only updates on clock edge when write control input is 1
  - Used when stored value is required later

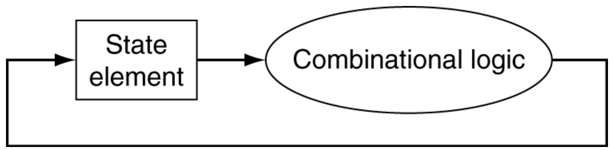




Clocking Methodology

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

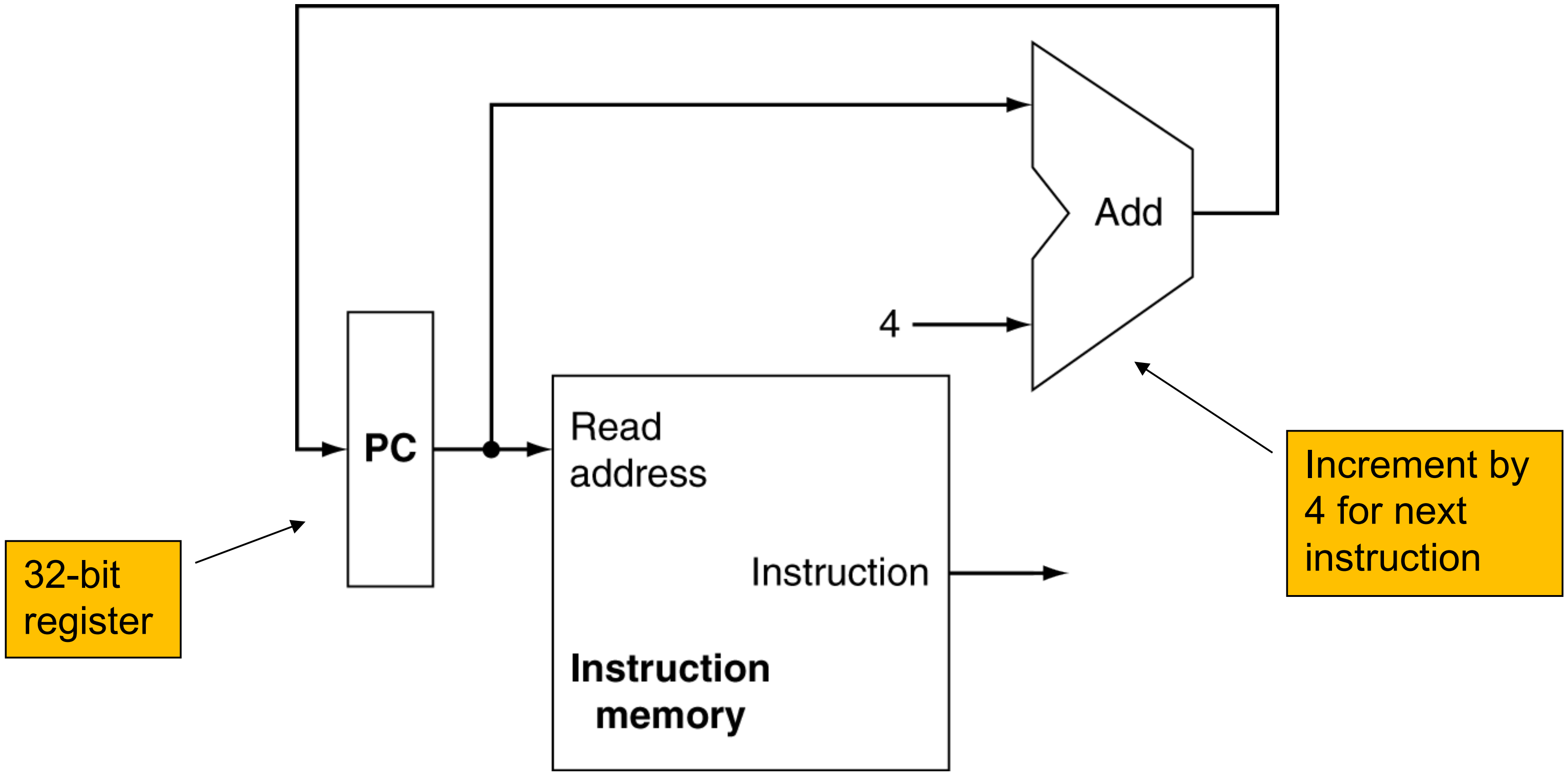




Building a Datapath

- Datapath
    - Elements that process data and addresses in the CPU
        - Registers, ALUs, mux’s, memories, …
- We will build a MIPS datapath incrementally
    - Consisting of 3-elements
    - Implementing instructions

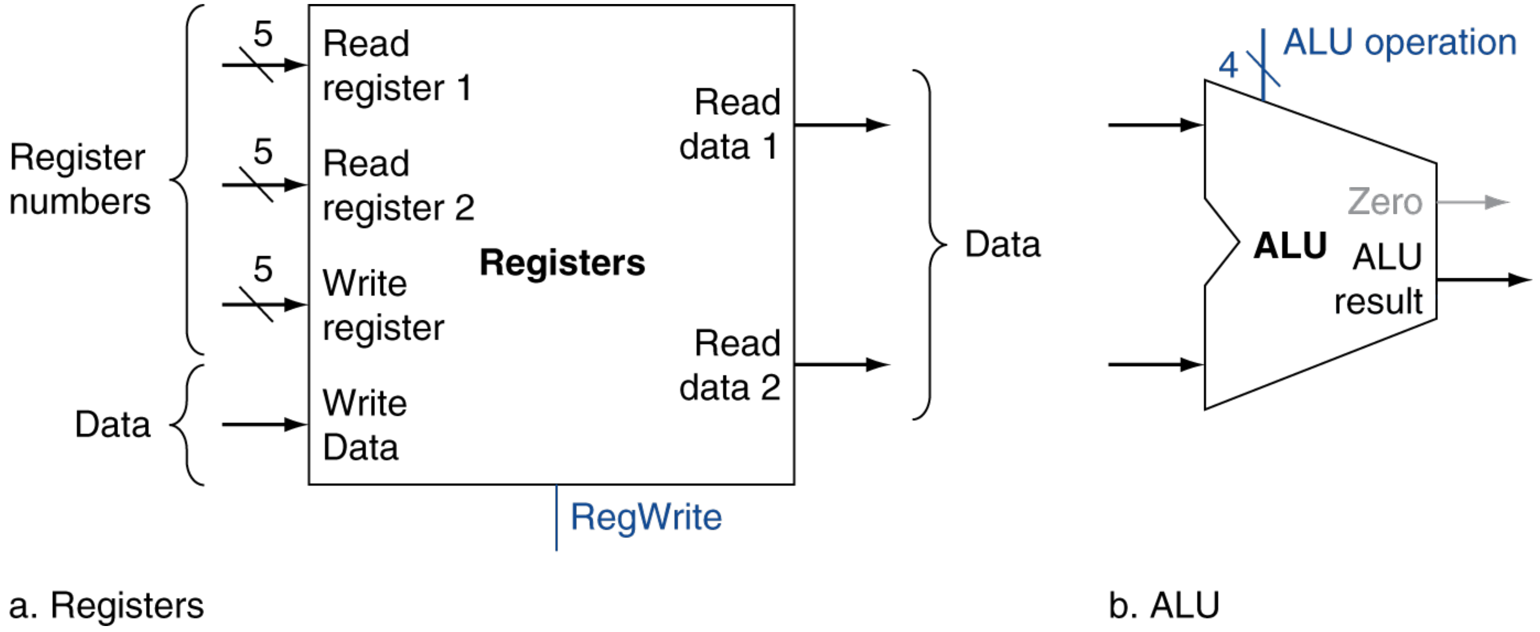
Instruction Fetch



Requires two state elements
Instruction memory and PC

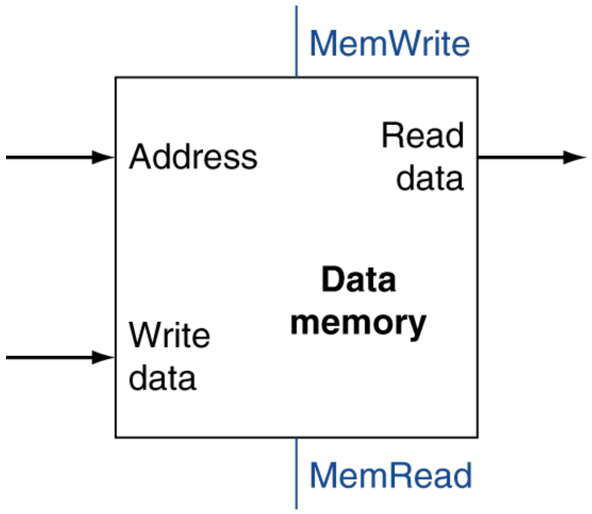
R-Format Instructions

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

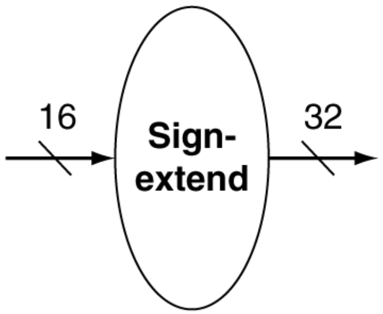


Load/Store Instructions

- 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