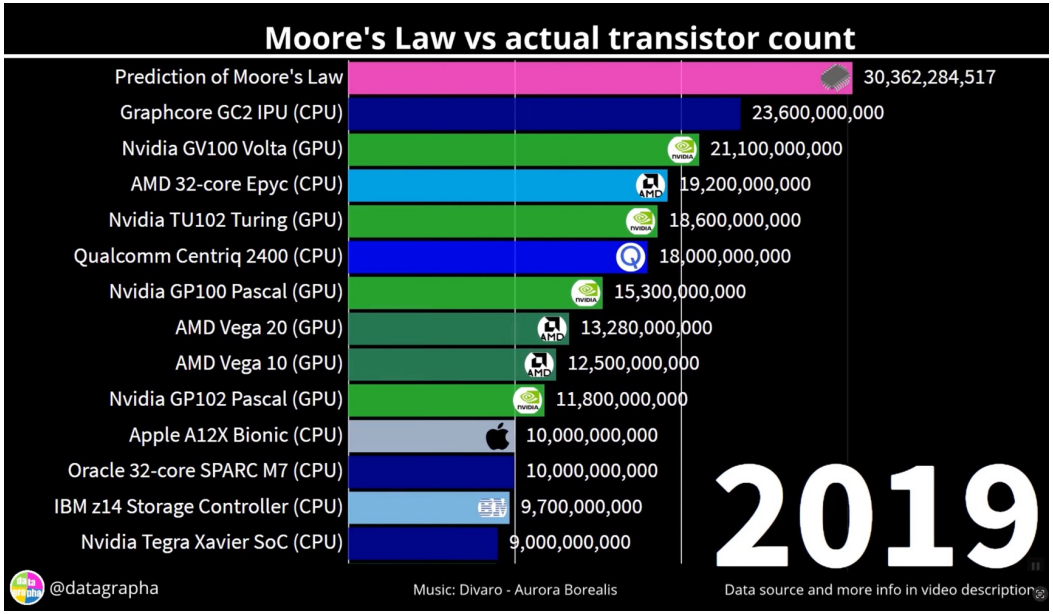
Jae W. Lee (jaewlee@snu.ac.kr), SNU Computer Science and Engineering

***Slide credits**: [CS:APP3e] slides from CMU; [COD:RV2e] slides from Elsevier Inc.*

The Processor (1)

Lecture 4
September 27th, 2023



CompArch Today: [Animation: Visualizing Moore's Law in Action (1971-2019)]
https://www.visualcapitalist.com/visualizing-moores-law-in-action-1971-2019/

This video clip compares the predictions of Moore's Law with data from actual computer chip innovations occurring between 1971 (Intel 4004) to 2019. The most recent NVIDIA H100 (Hopper) GPU reportedly has 80 billion transistors in TSMC 4N process (2022).

Introduction

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

Outline

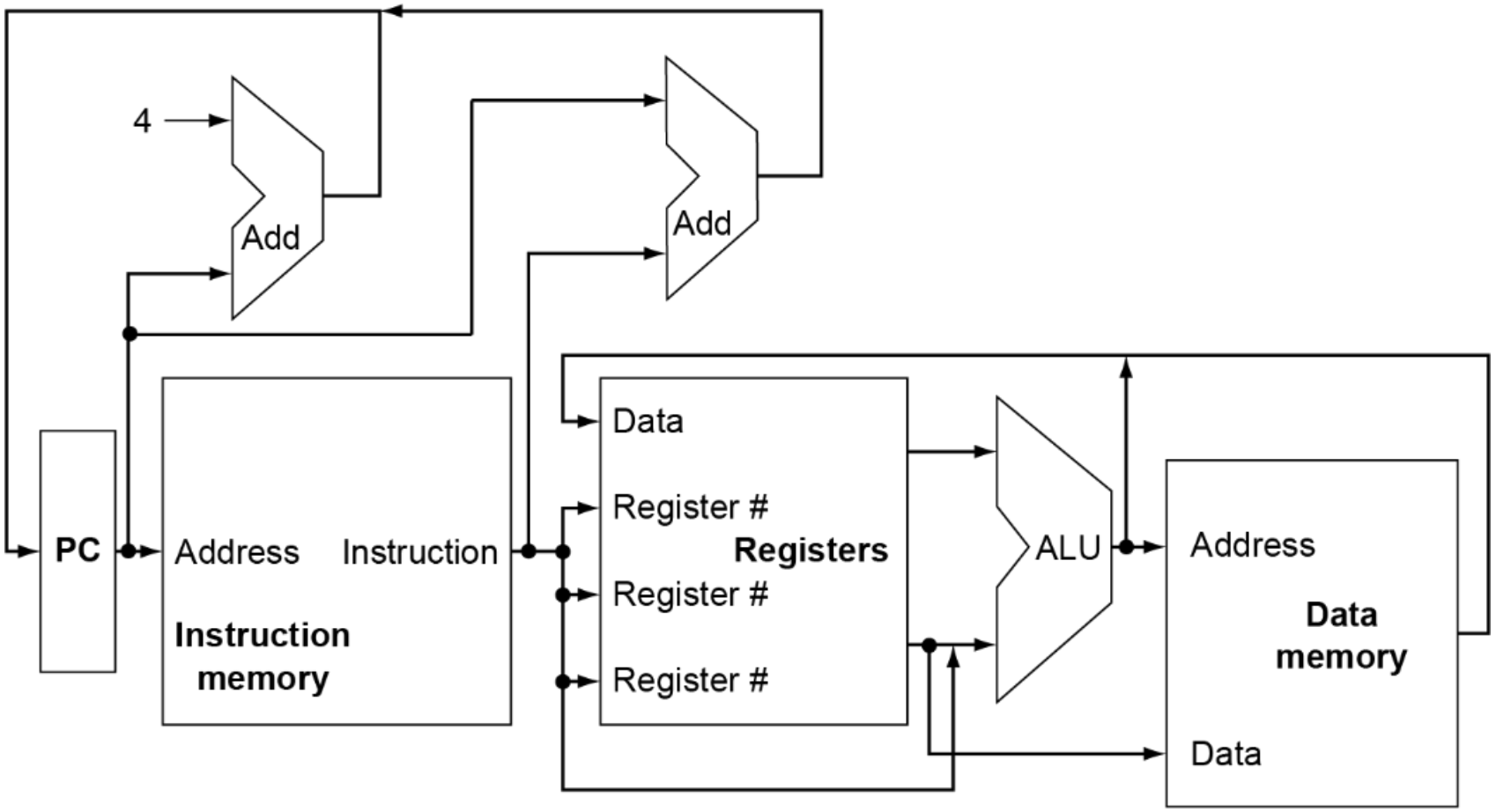
Textbook: [COD:RV2e] 4.1-4.4

- **Logic Design Basics & Implementation Overview**
- **Building a Datapath**
- **Control Logic Design**

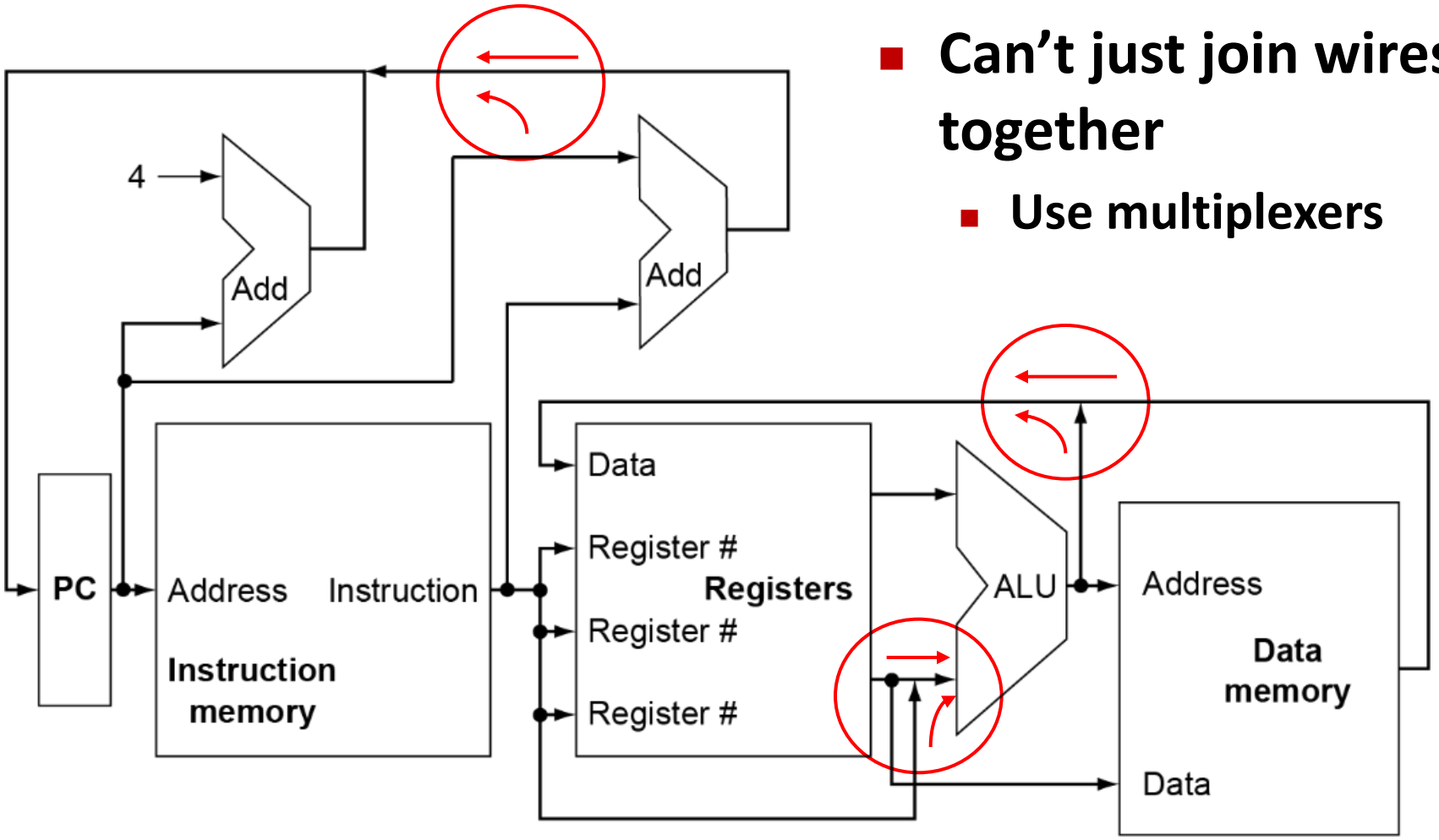
Instruction Execution

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

CPU Overview

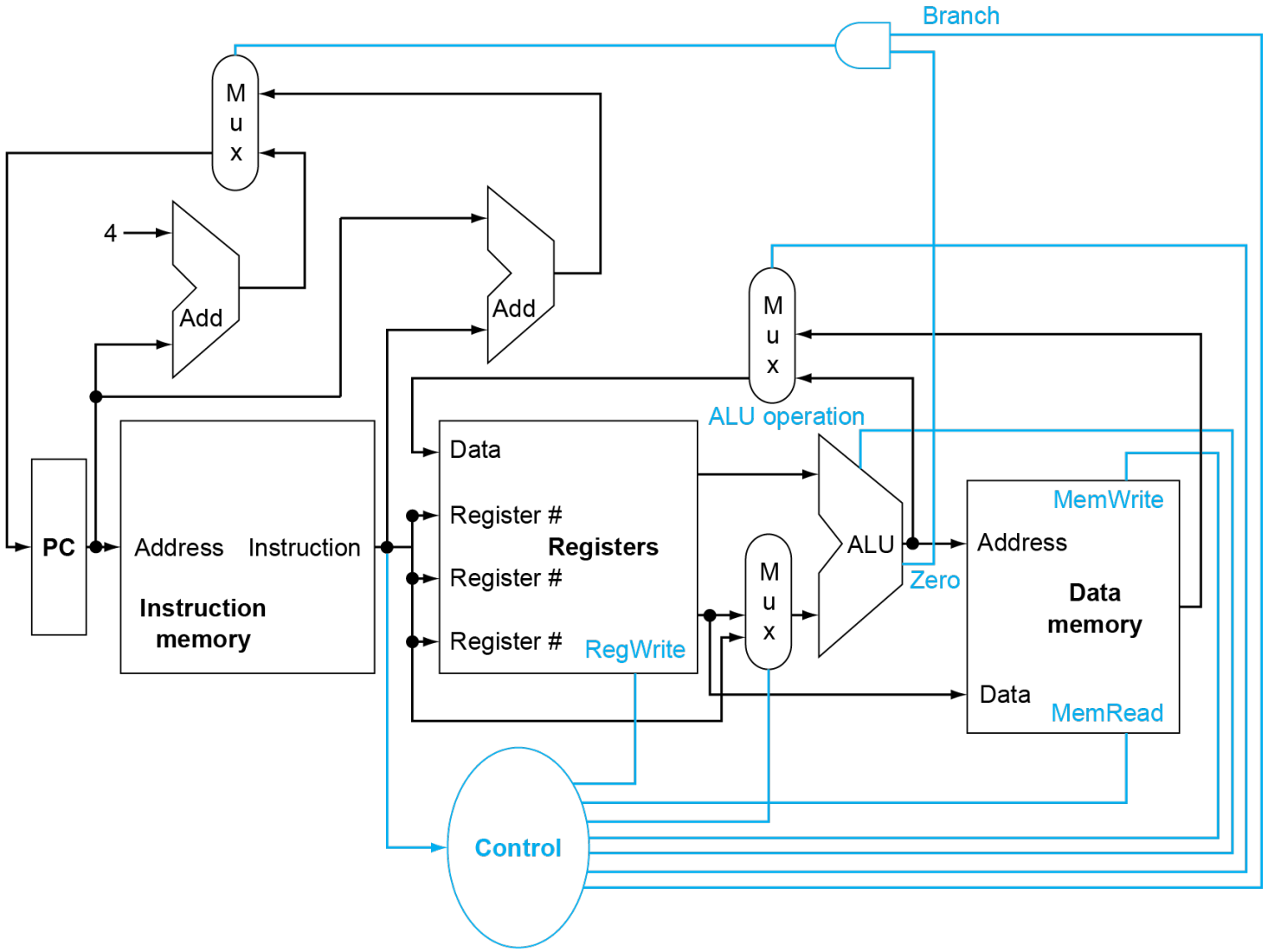


Multiplexers



- Can't just join wires together
  - Use multiplexers

Control

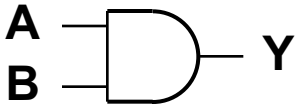

Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
PC
Address
Instruction
Instruction
memory
Data
Register #
Registers
Register #
Register #
RegWrite
M
u
x
ALU
Zero
MemWrite
Address
Data
memory
Data
MemRead
Control

Logic Design Basics

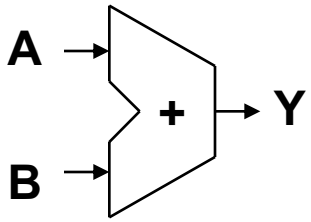
- **Information encoded in binary**
  - Low voltage = 0, High voltage = 1
  - One wire per bit
  - Multi-bit data encoded on multi-wire buses
- **Combinational element**
  - Operate on data
  - Output is a function of input
- **State (sequential) elements**
  - Store information

Combinational Elements

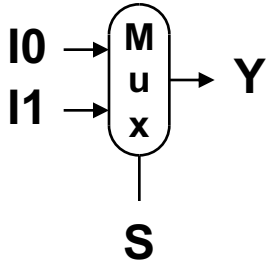
- **AND-gate**
  - Y = A & B



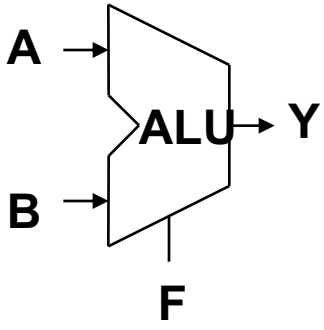
- **Adder**
  - **Y = A + B**



- **Multiplexer**
  - **Y = S ? I1 : I0**

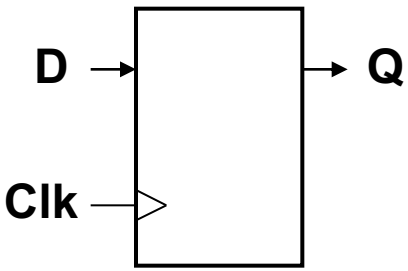


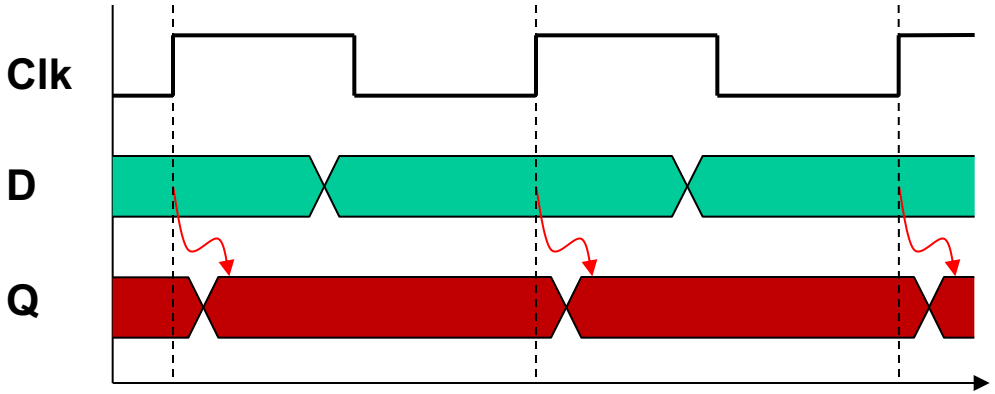
- **Arithmetic/Logic Unit**
  - **Y = F(A, B)**



Sequential Elements

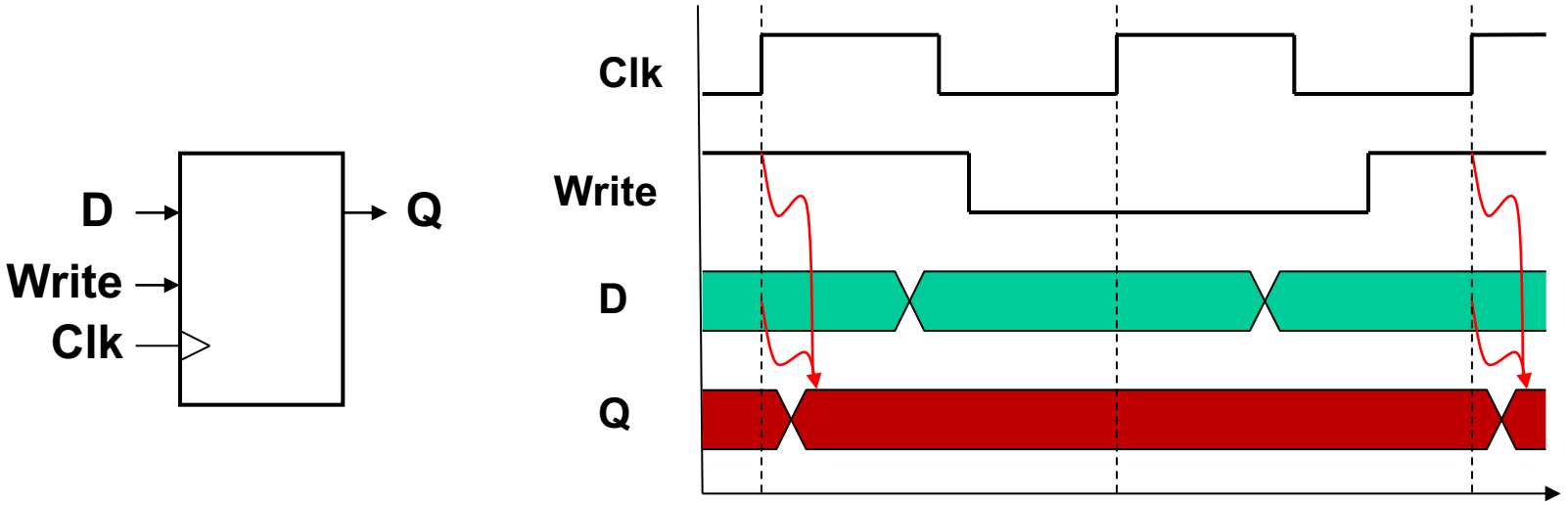
- **Register: stores data in a circuit**
  - Uses a clock signal to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1





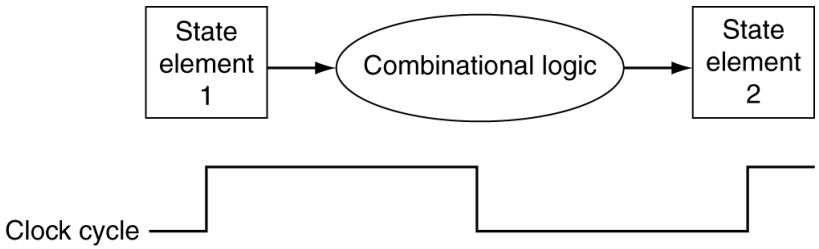
Sequential Elements

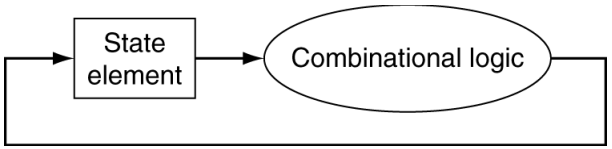
- **Register with write control**
  - Only updates on clock edge when write control input is 1
  - Used when stored value is required later



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





Outline

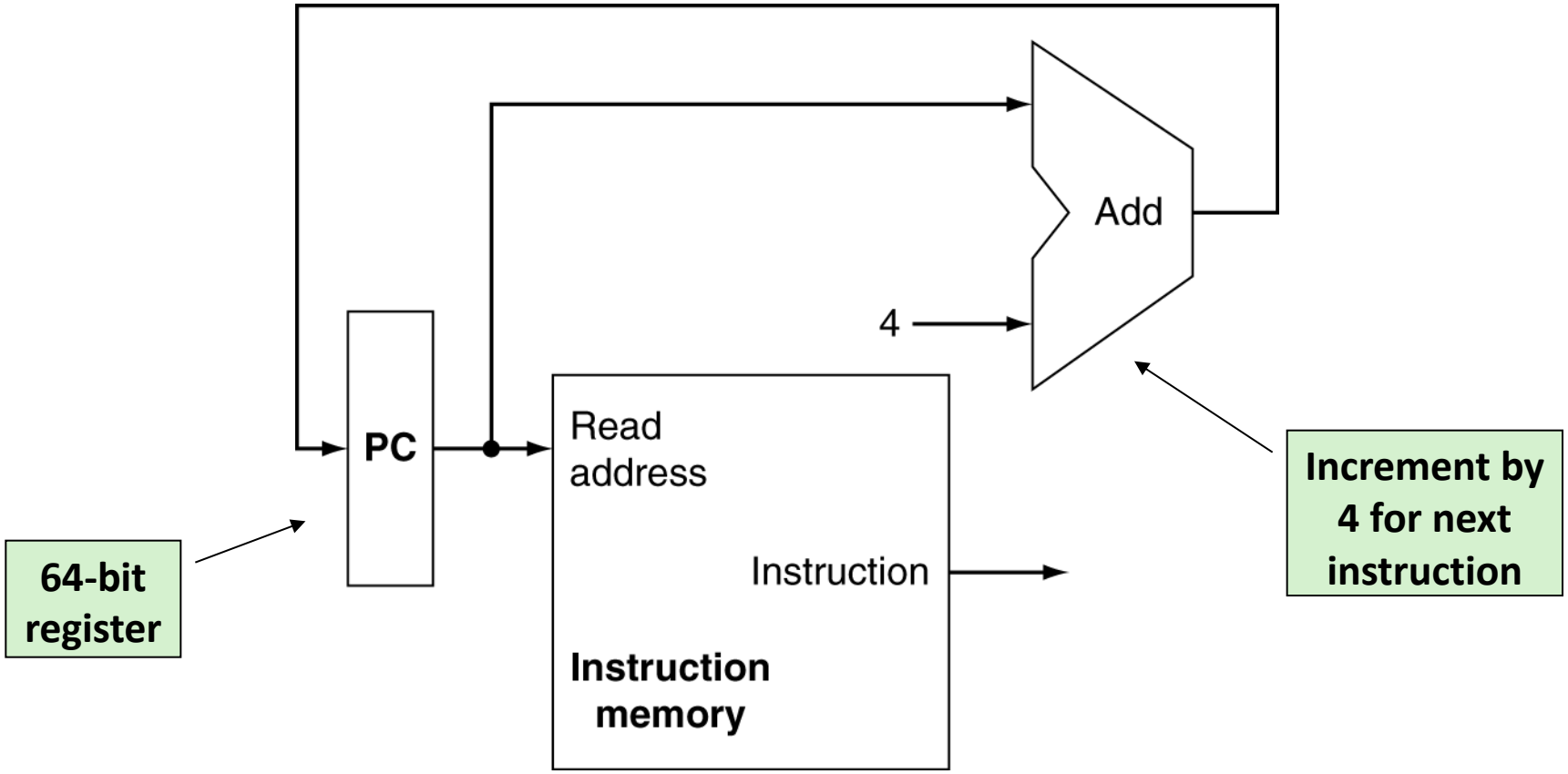
Textbook: [COD:RV2e] 4.1-4.4

- Logic Design Basics & Implementation Overview
- **Building a Datapath**
- Control Logic Design

Building a Datapath

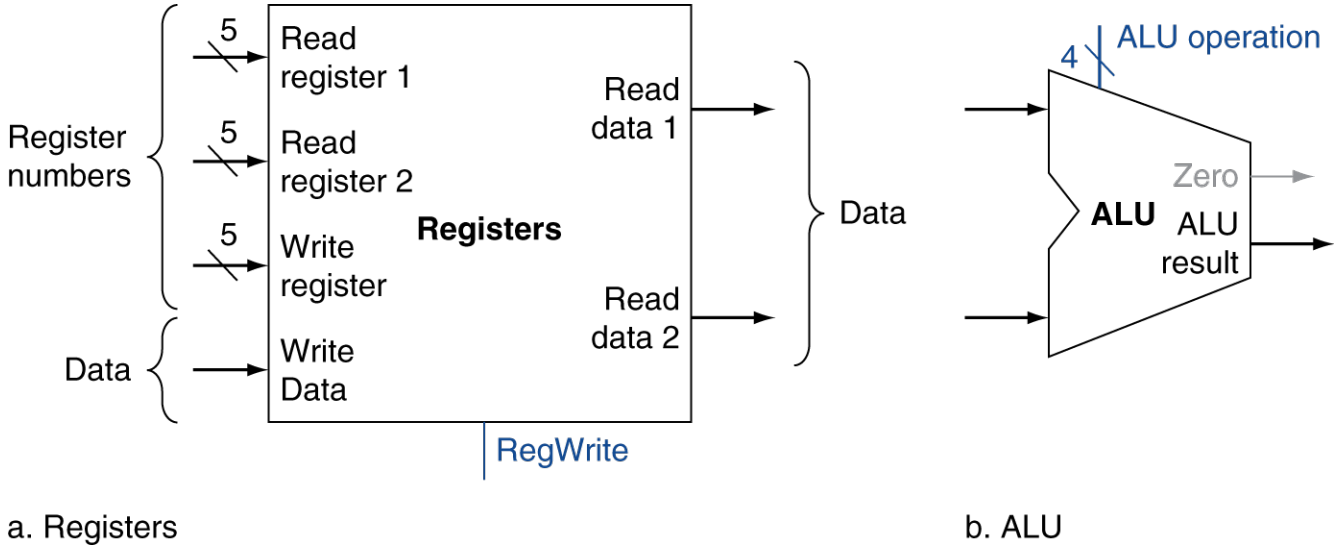
- **Datapath**
  - Elements that process data and addresses in the CPU
    - Registers, ALUs, mux's, memories, …
- **We will build a RISC-V datapath incrementally**
  - Refining the overview design

Instruction Fetch



R-Format Instructions

- Read two register operands (e.g., add)
- Perform arithmetic/logical operation
- Write register result

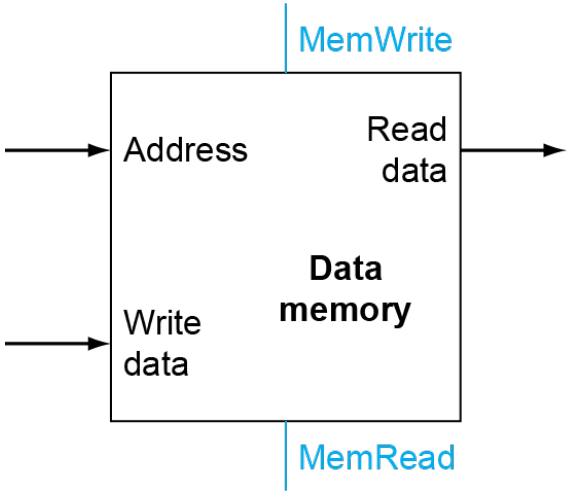


a. Registers

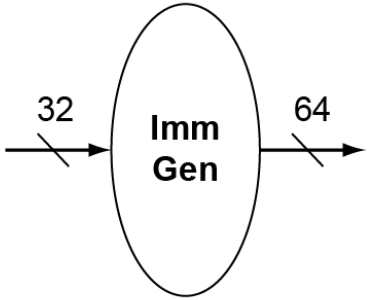
b. ALU

Load/Store Instructions

- **Read register operands**
- **Calculate address using 12-bit offset**
  - Use ALU, but sign-extend offset
- **Load: Read memory and update register**
- **Store: Write register value to memory**



a. Data memory unit

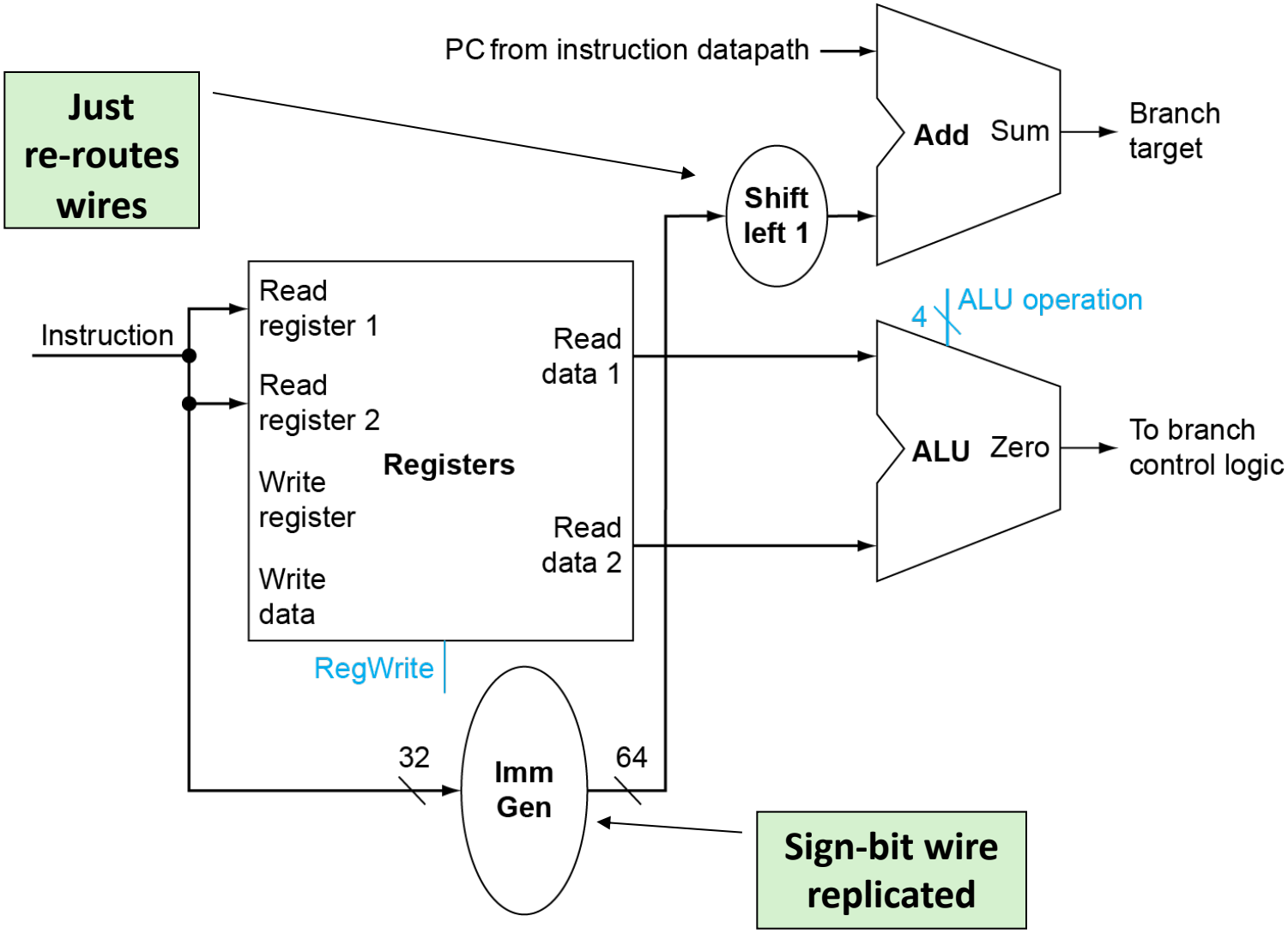


b. Immediate generation unit

Branch Instructions

- **Read register operands**
- **Compare operands**
  - Use ALU, subtract and check Zero output
- **Calculate target address**
  - Sign-extend displacement
  - Shift left 1 place (halfword displacement)
  - Add to PC value

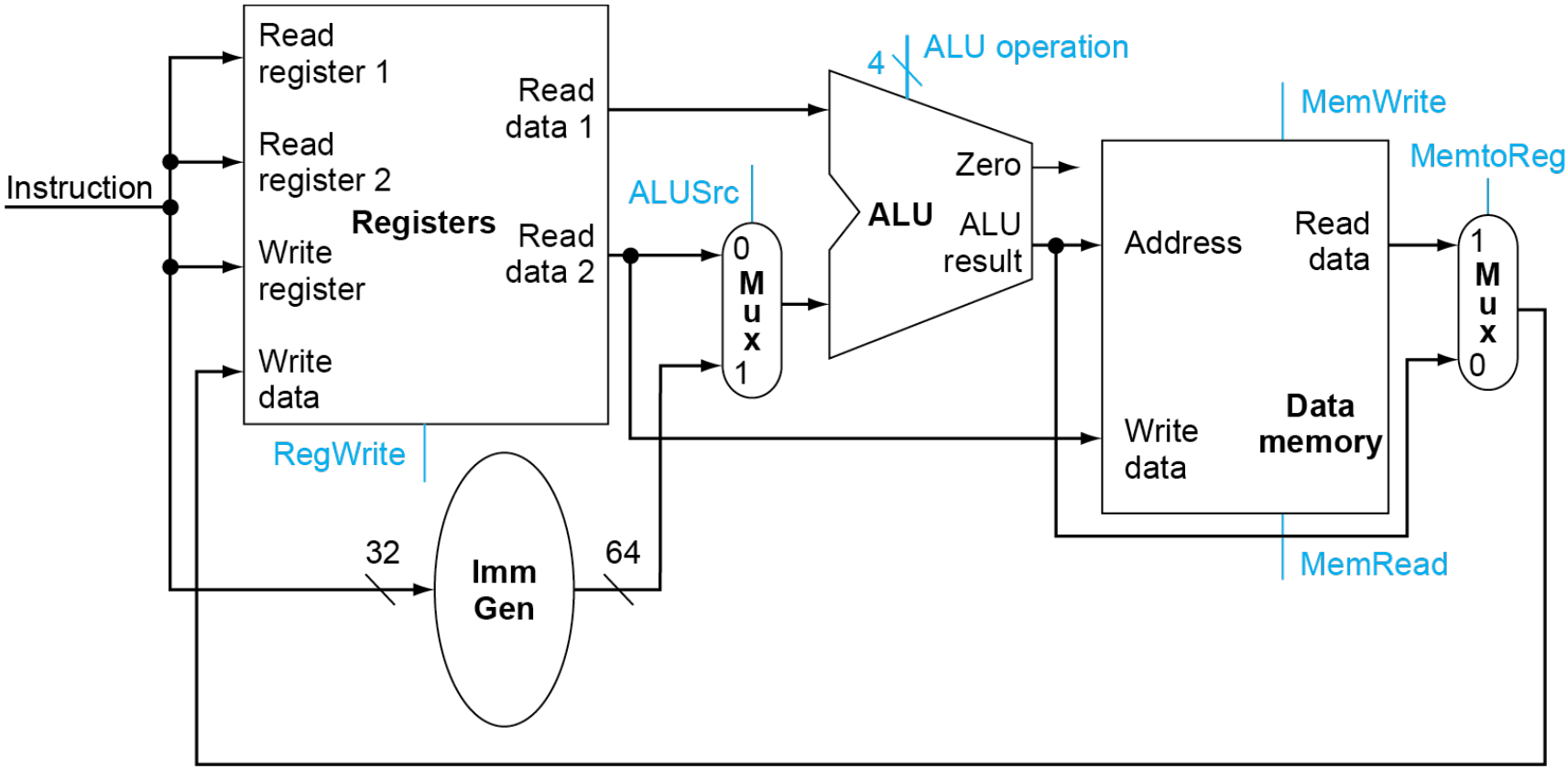
Branch Instructions



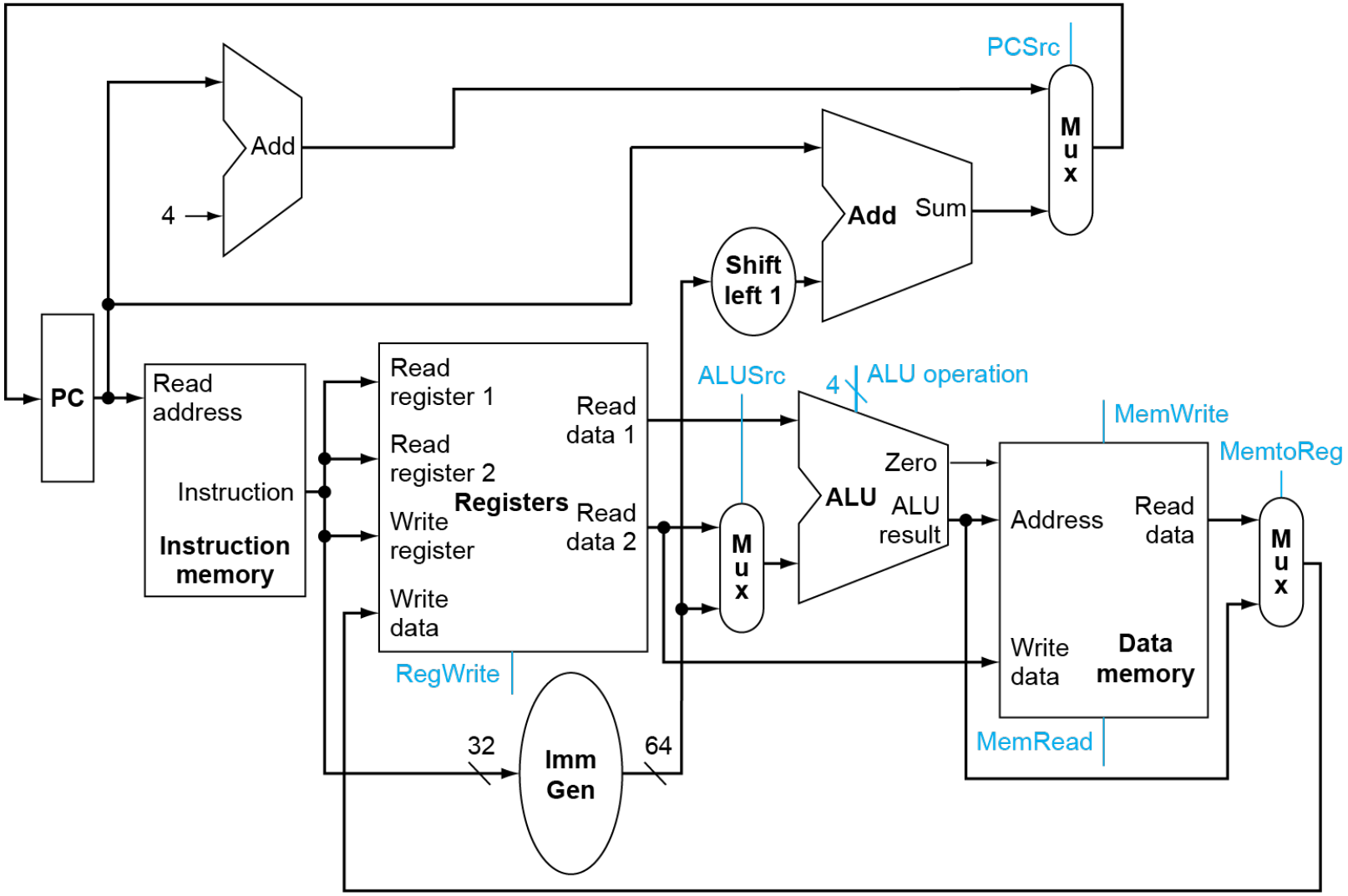
Composing the Elements

- **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



Full Datapath



Outline

Textbook: [COD:RV2e] 4.1-4.4

- Logic Design Basics & Implementation Overview
- Building a Datapath
- **Control Logic Design**

ALU Control

- **ALU used for**
  - Load/Store: F = add
  - Branch: F = subtract
  - R-type: F depends on opcode

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract

ALU Control

- **Assume 2-bit ALUOp derived from opcode**
  - Combinational logic derives ALU control

opcode	ALUOp	Operation	Opcode field	ALU function	ALU control
ld	00	load register	XXXXXXXXXXX	add	0010
sd	00	store register	XXXXXXXXXXX	add	0010
beq	01	branch on equal	XXXXXXXXXXX	subtract	0110
R-type	10	add	000 0000000	add	0010
		subtract	000 0100000	subtract	0110
		AND	111 0000000	AND	0000
		OR	110 0000000	OR	0001

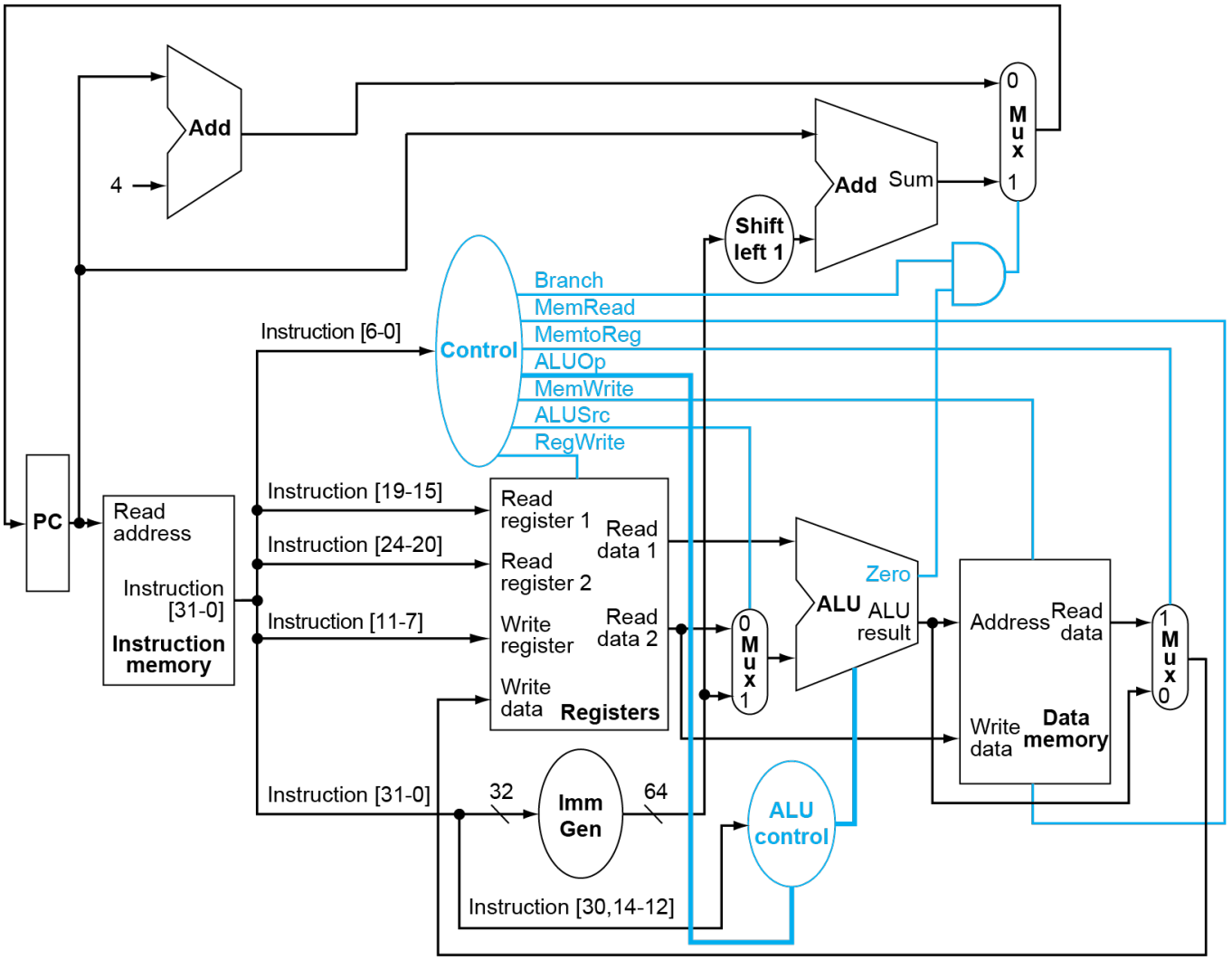
The Main Control Unit

- Control signals derived from instruction

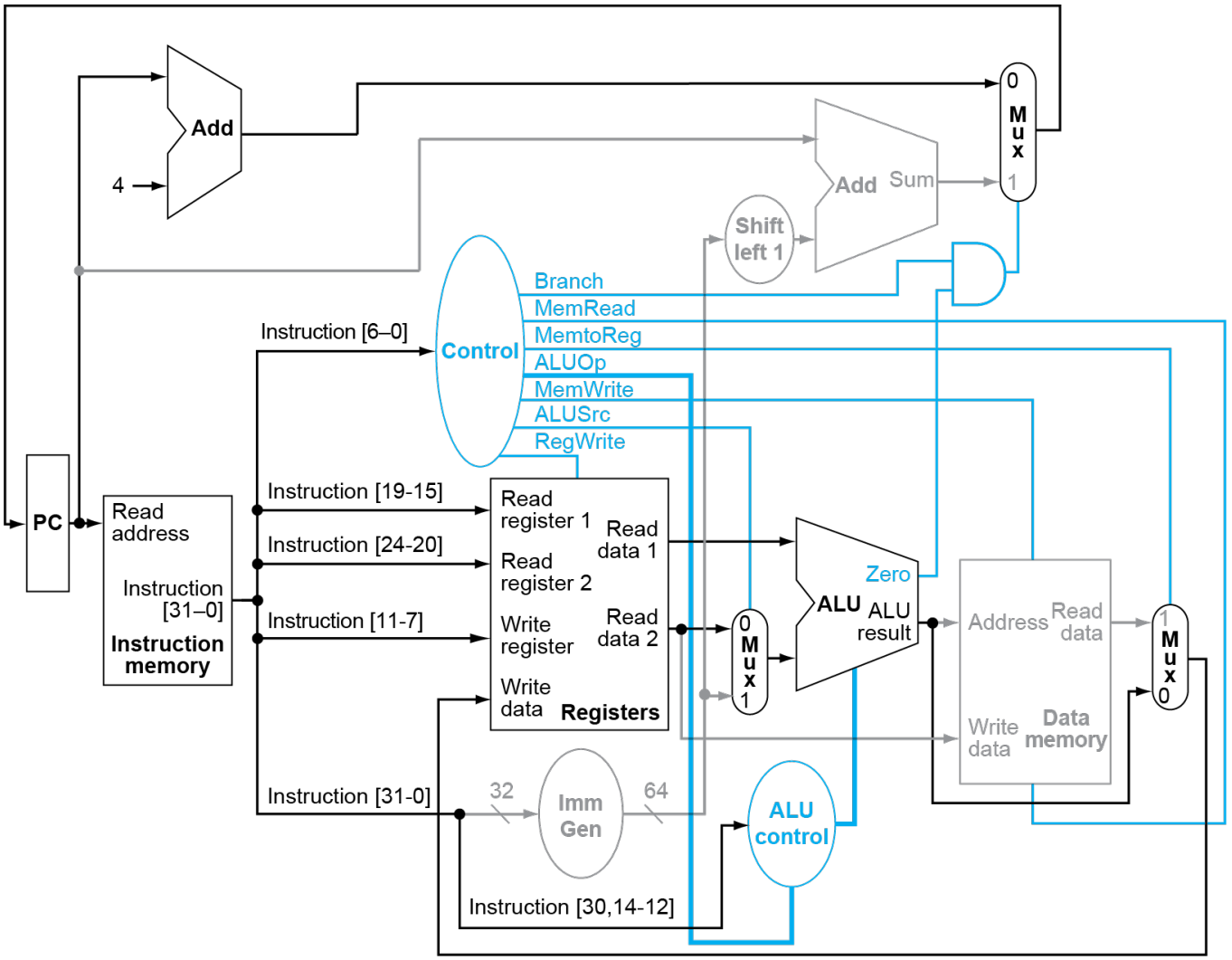
Name (Bit position)	Fields 31:25	24:20	19:15	14:12	11:7	6:0
(a) R-type	funct7	rs2	rs1	funct3	rd	opcode
(b) I-type	immediate[11:0]		rs1	funct3	rd	opcode
(c) S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode
(d) SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode

ALUOp		Funct7 field							Funct3 field			
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operation
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

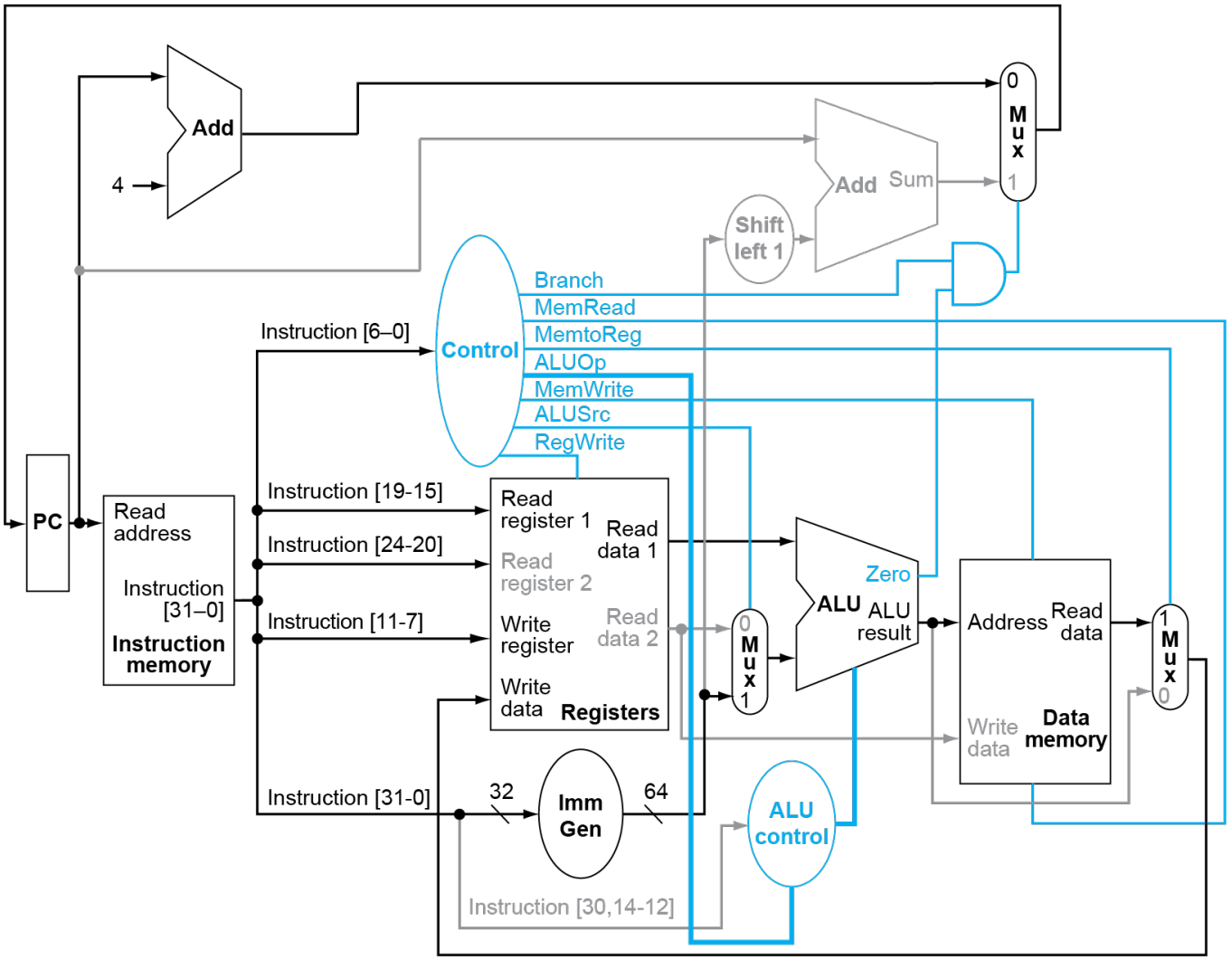
Datapath With Control



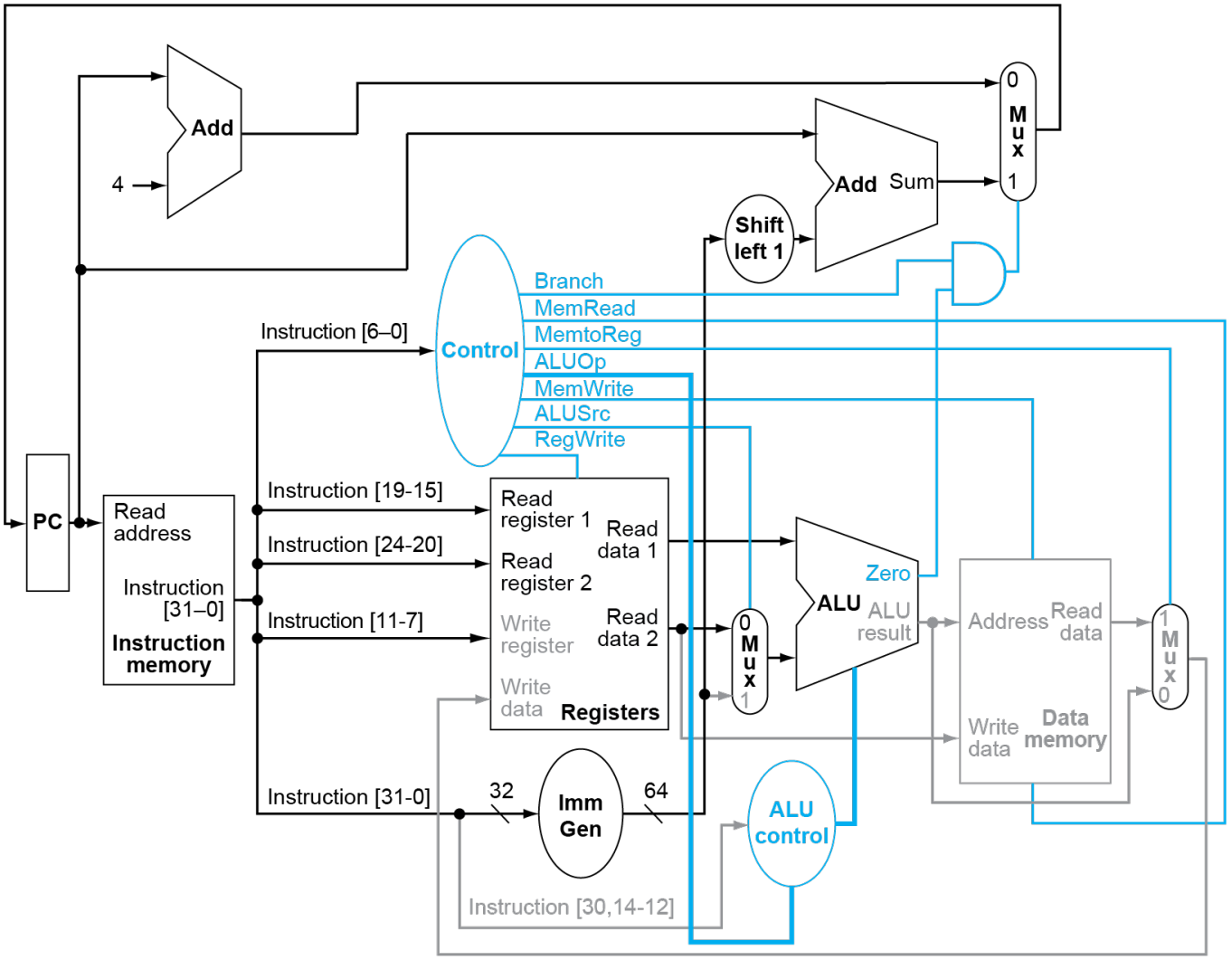
R-Type Instruction



Load Instruction



BEQ Instruction



Performance Issues

- **Longest delay determines clock period**
  - Critical path: load instruction
  - Instruction memory → register file → ALU → data memory → register file
- **Not feasible to vary period for different instructions**
- **Violates design principle**
  - Making the common case fast
- **We will improve performance by pipelining**