Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Fall 2019

Sequential Processor

Chap. 4.1, 4.3, 4.4

Introduction

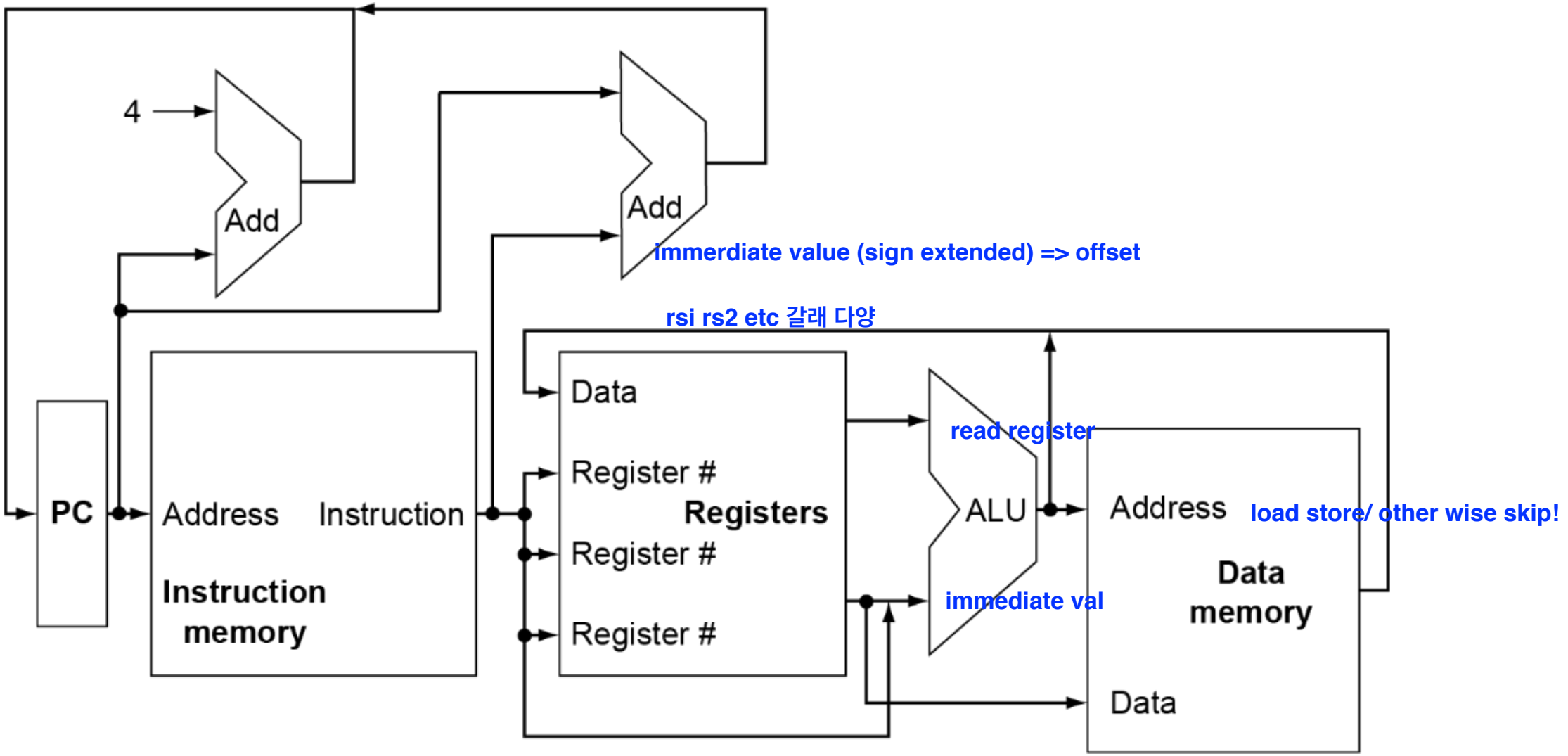
- CPU performance factors
  - Instruction count – Determined by ISA and compiler + algorithm
  - CPI and cycle time – Determined by CPU hardware
- We will examine two RISC-V implementations
  - A simplified (sequential) version
  - A more realistic pipelined version
- Simple subset, shows most aspects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

Instruction Execution

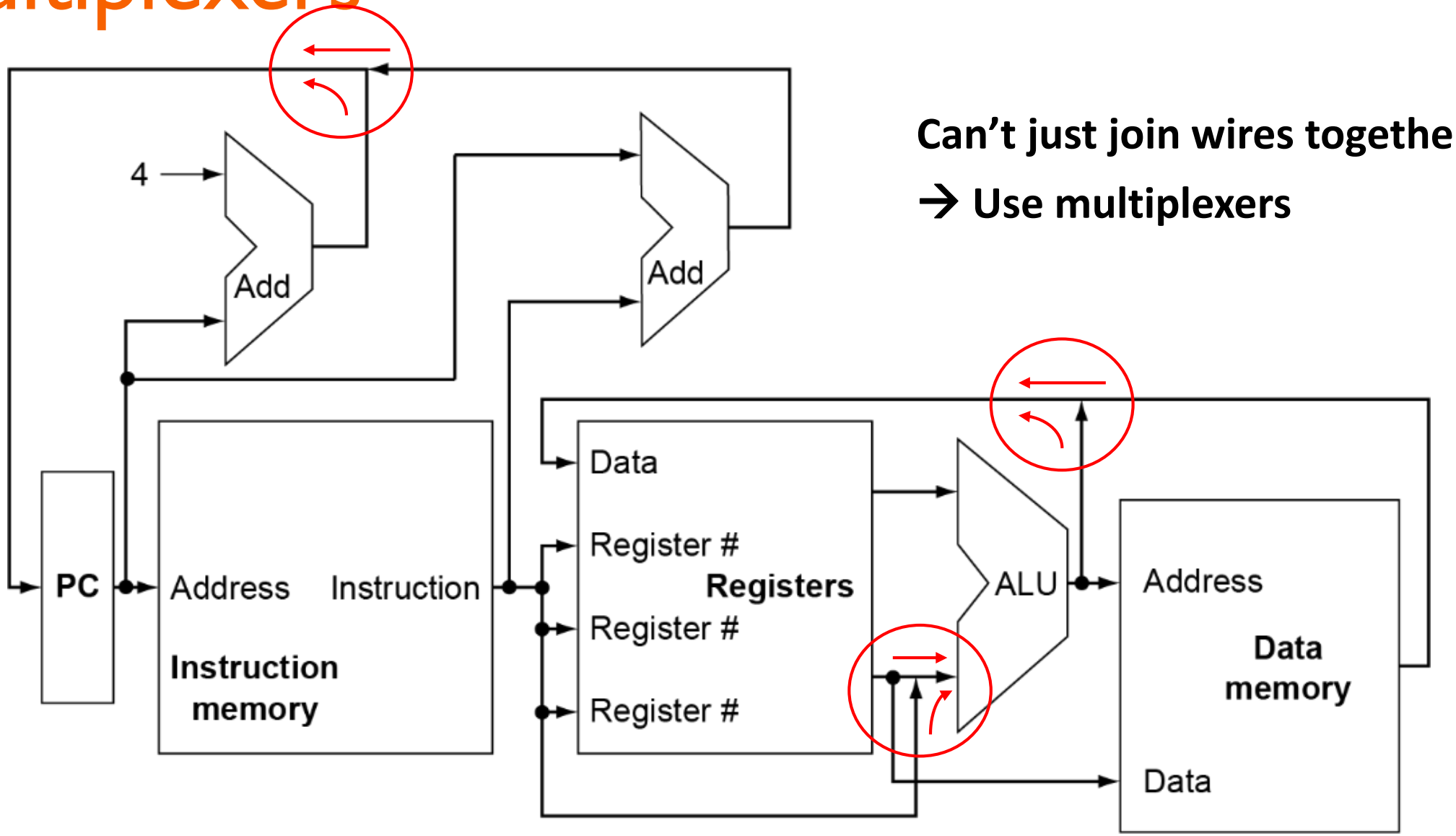
- PC → instruction memory, fetch instruction
- Register numbers → register file, read registers
- Depending on instruction class
  - Use ALU to calculate: Arithmetic result
Memory address for load/store
Branch comparison
  - Access data memory for load/store
  - PC ← target address or PC + 4

CPU: Datapath + Control

CPU Overview



Multiplexers

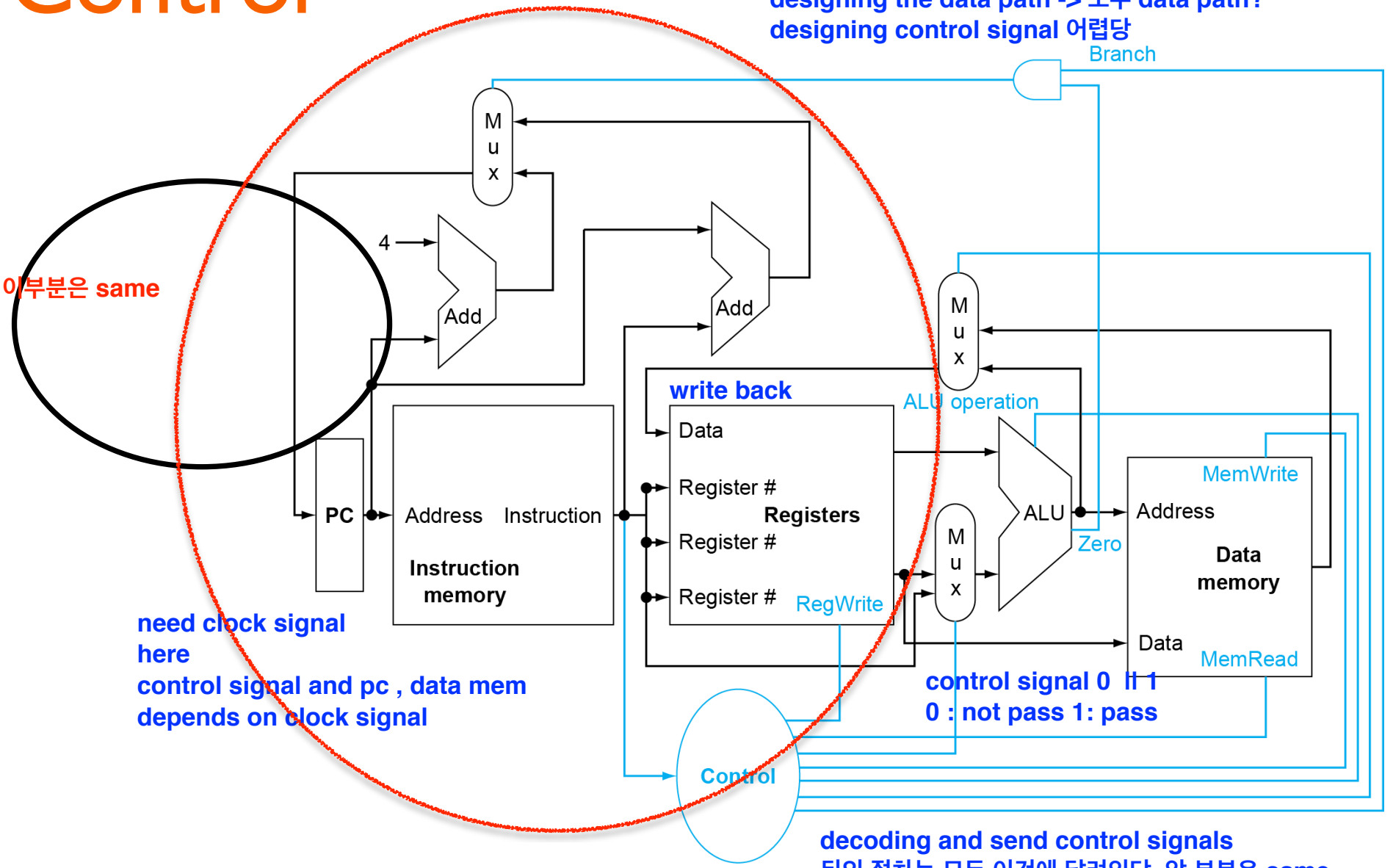


Can't just join wires together

→ Use multiplexers

Control

components : mux adder instruction memory registers data memoty ALU etc
designing the data path -> 모두 data path?
designing control signal 어렵당



0부분은 same

need clock signal
here
control signal and pc , data mem
depends on clock signal

control signal 0 or 1
0 : not pass 1: pass

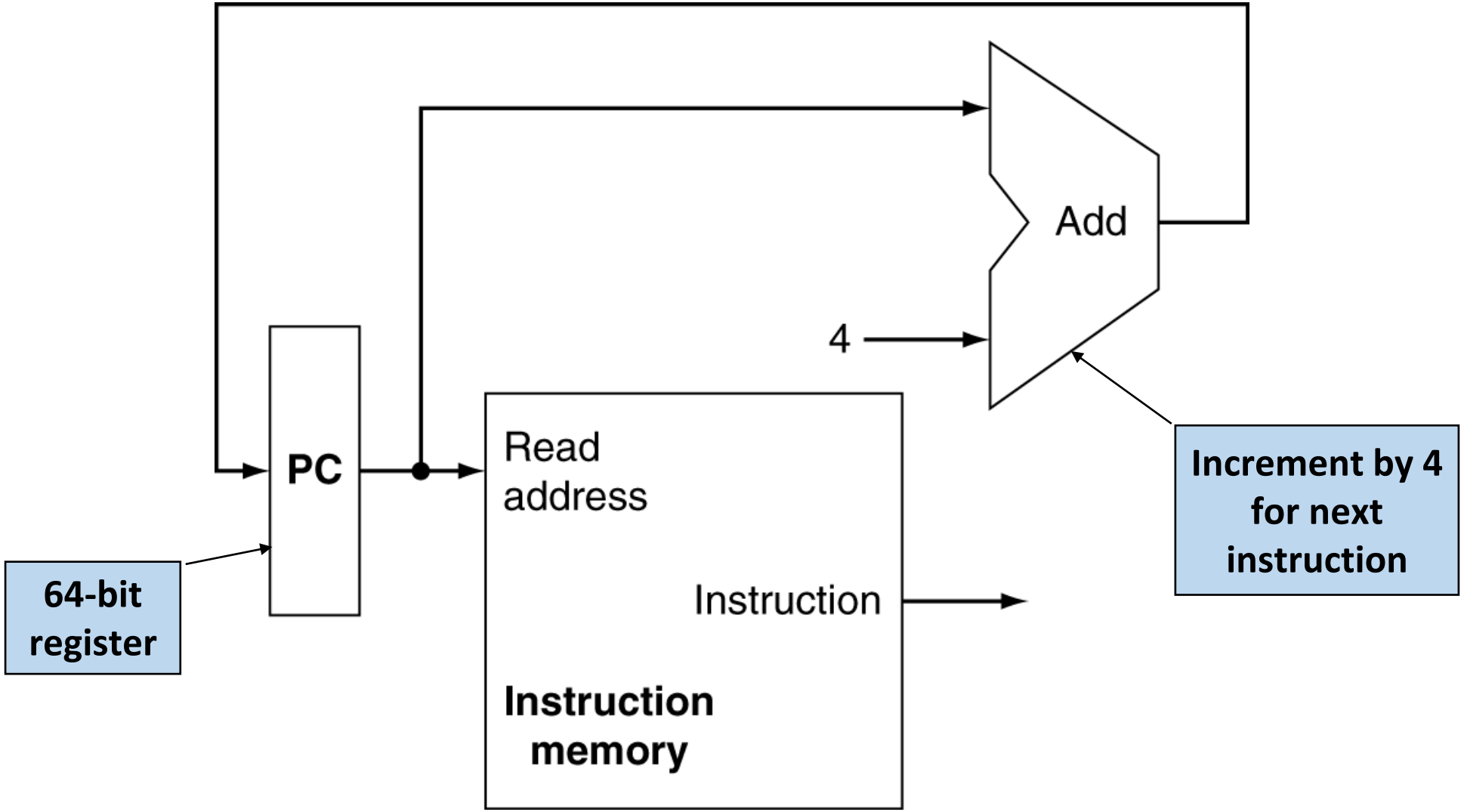
decoding and send control signals
뒤의 절차는 모두 이것에 달려있당. 앞 부분은 same

Building a Datapath

- Datapath
  - Elements that process data and addresses in the CPU
  - Registers, ALUs, MUX's, Memories, …

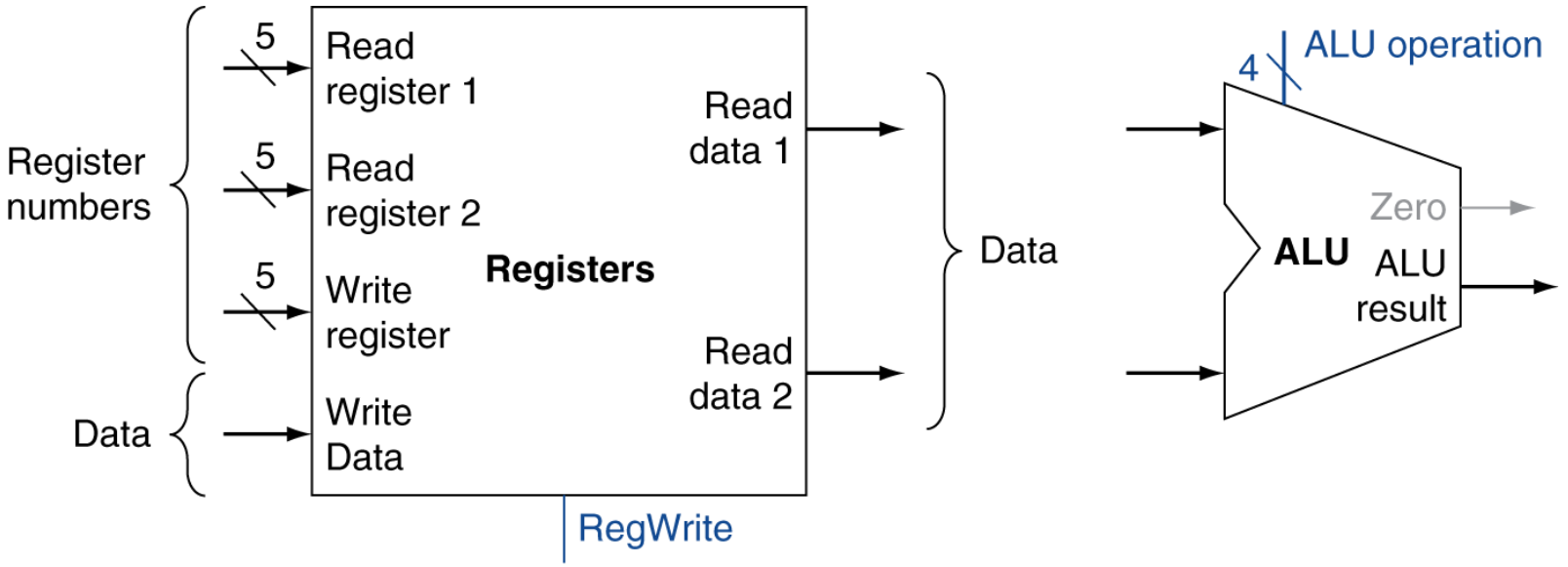
- We will build a RISC-V datapath incrementally
  - Refining the overview design

Instruction Fetch



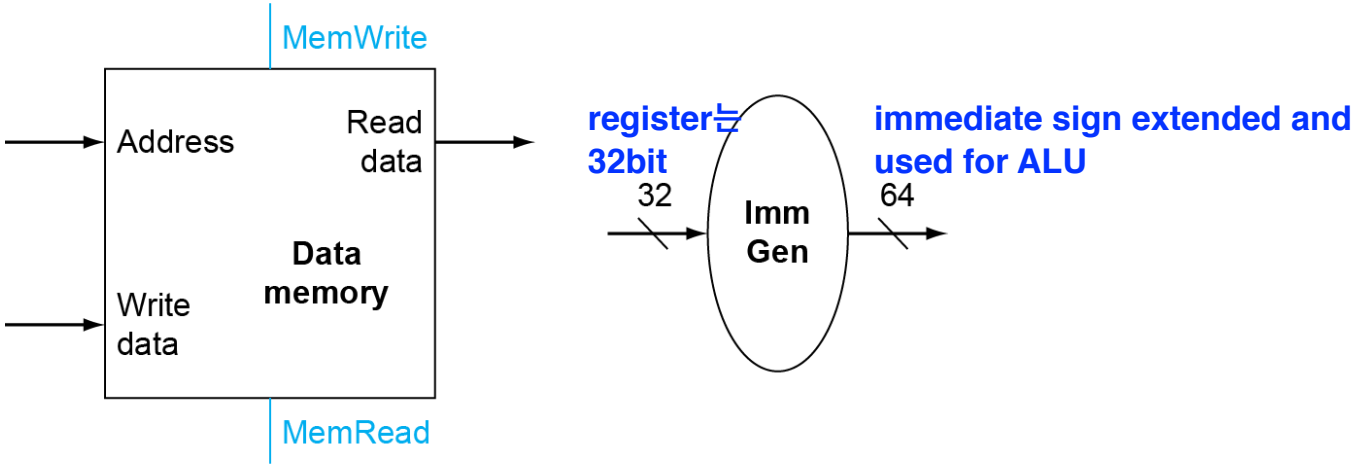
R-Format Instructions

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



Load/Store Instructions

- Read register operand
- 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

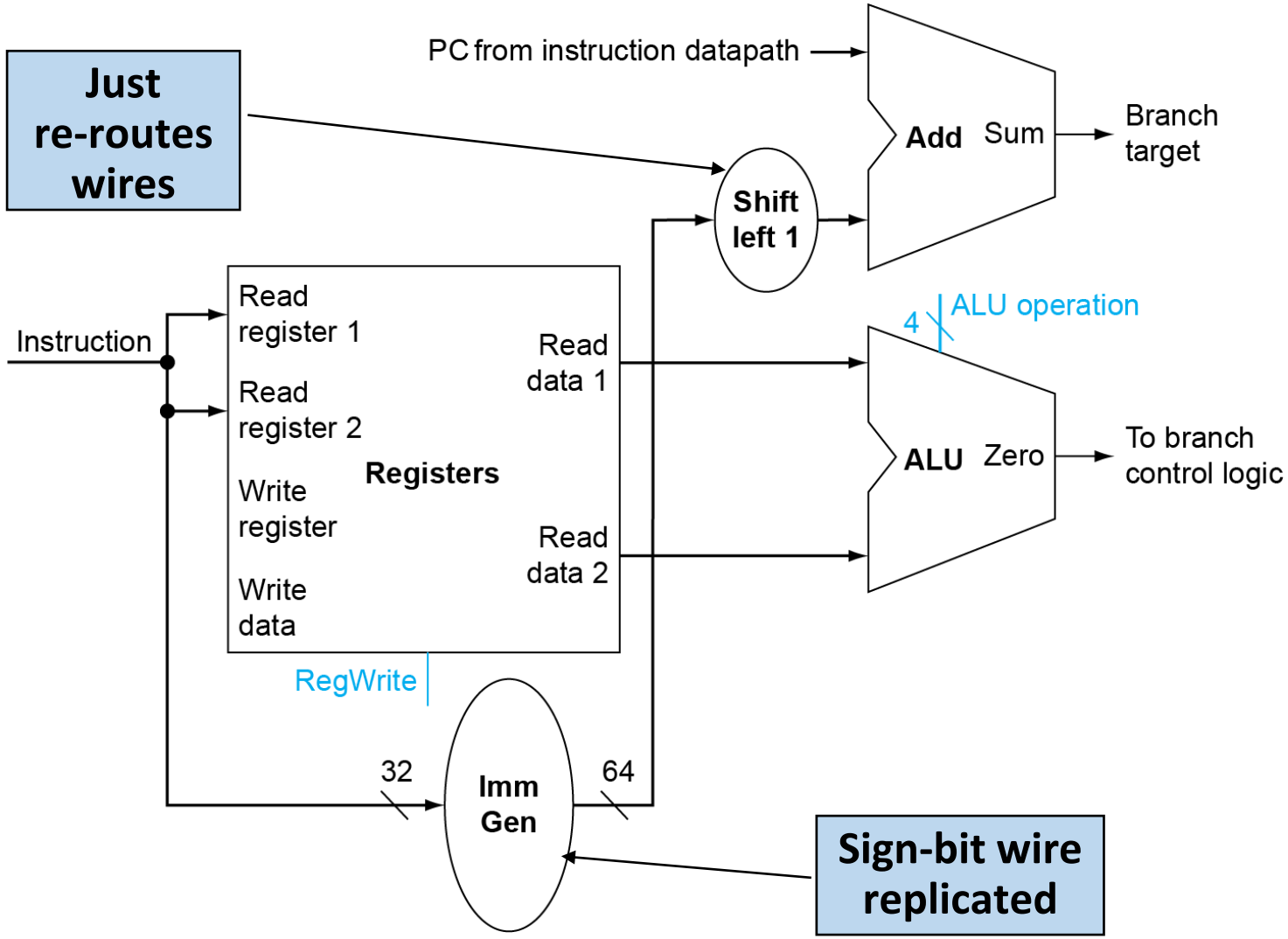
b. Immediate generation unit

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 targer address

Branch Instructions (cont'd)

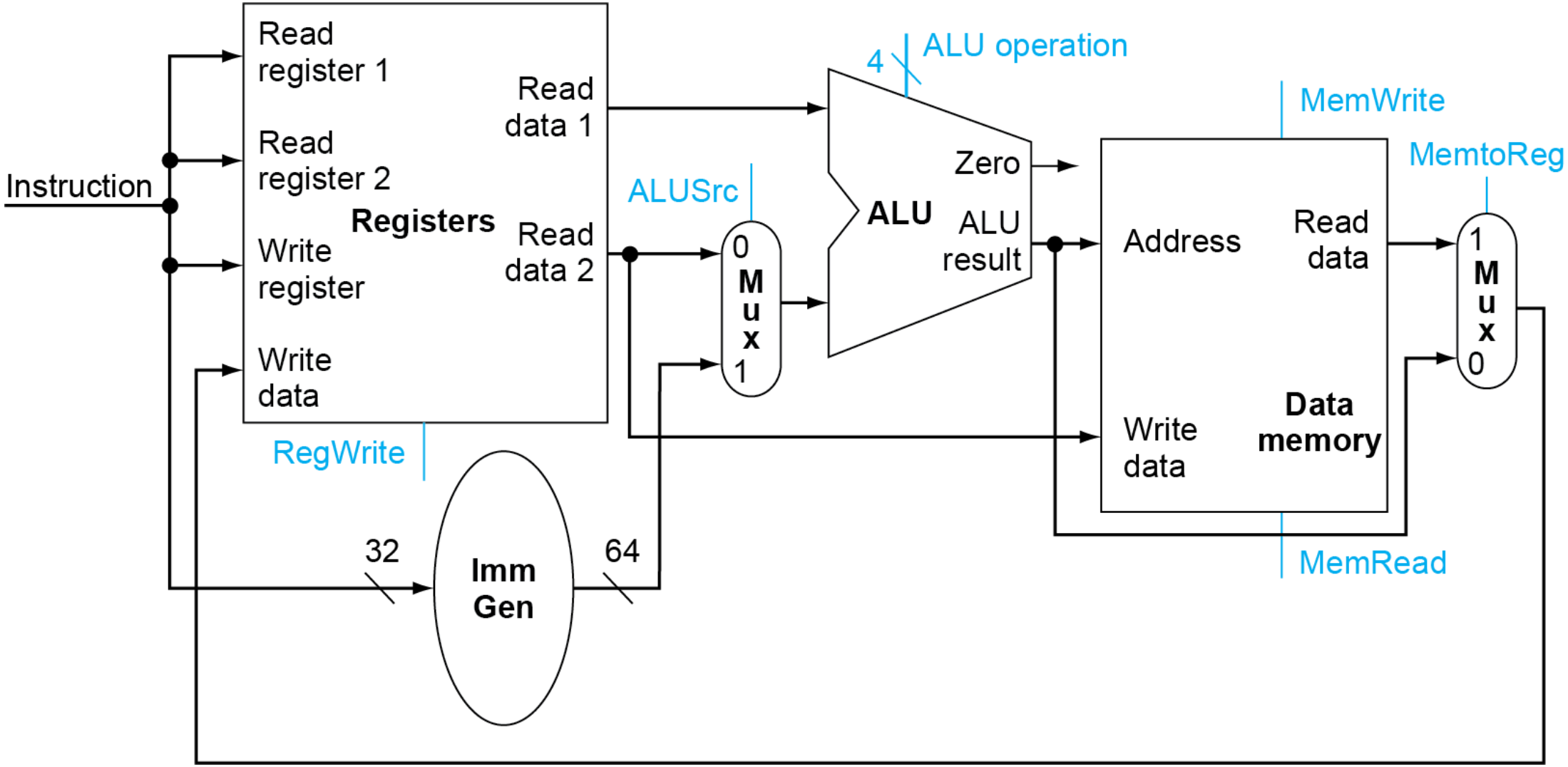


Composing the Elements

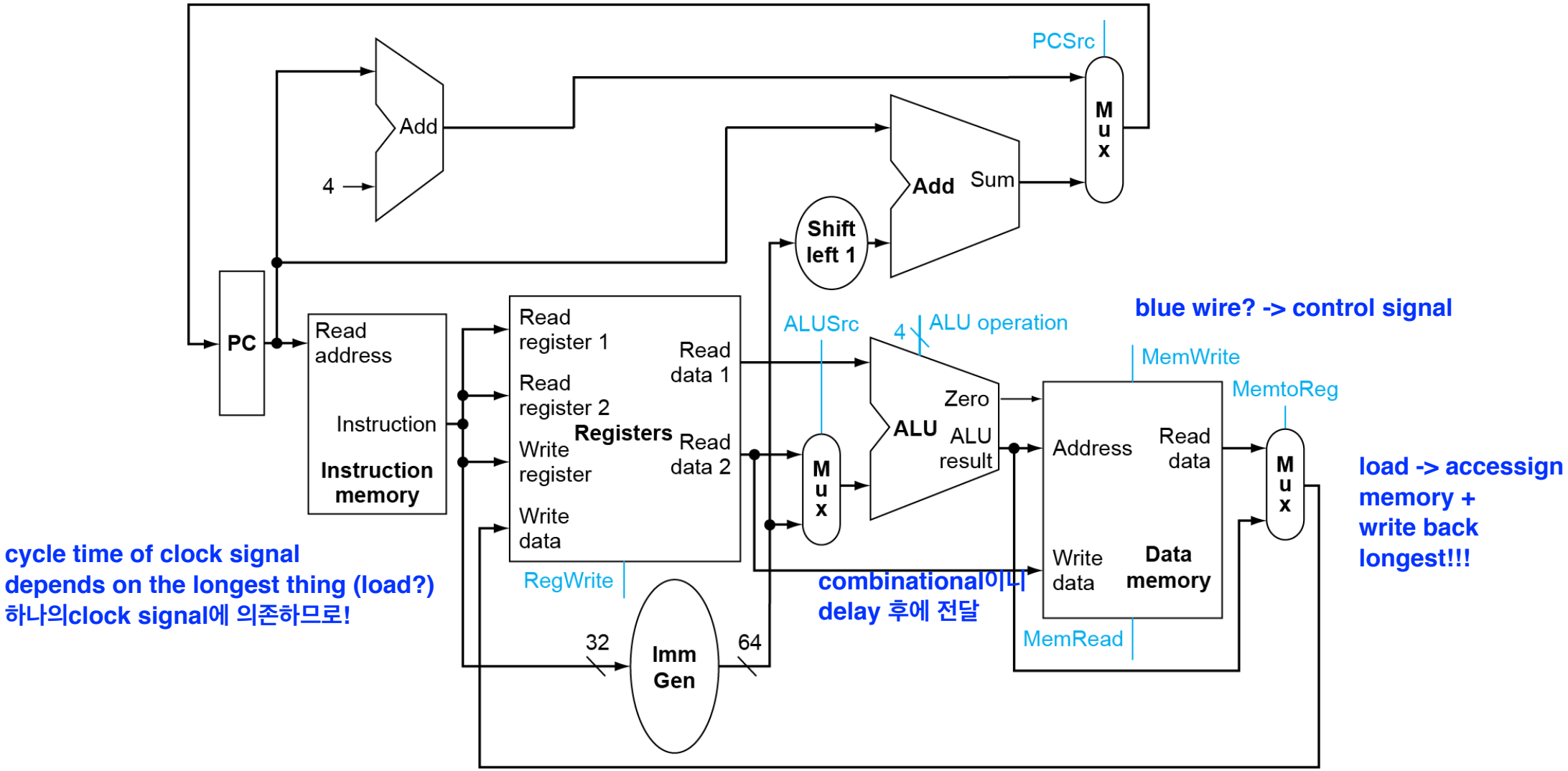
- First-cut datapath does an instruction in one clock cycle
  - Each datapath element can only do one function at a time
  - Hence, we need separate instruction and data memories

- Use multiplexers where alternate data sources are used for different instructions

R-Type/Load/Store Datapath



Full Datapath



ALU Control

- ALU used for
  - Load/Store: F = add
  - Branch: F = subtract check 0 ll 1
  - R-type: F depends on opcode

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract

ALU Control (cont'd)

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

opcode	ALUOp	Operation	funct7	funct3	ALU function	ALU control
ld	00	load register	XXXXXXX	XXX	add	0010
sd	00	store register	XXXXXXX	XXX	add	0010
beq	01	branch on equal	XXXXXXX	XXX	subtract	0110
R-type	10	add	0000000	000	add	0010
		subtract	0100000	000	subtract	0110
		AND	0000000	111	AND	0000
		OR	0000000	110	OR	0001

The Main Control Unit

- Control signals derived from instruction

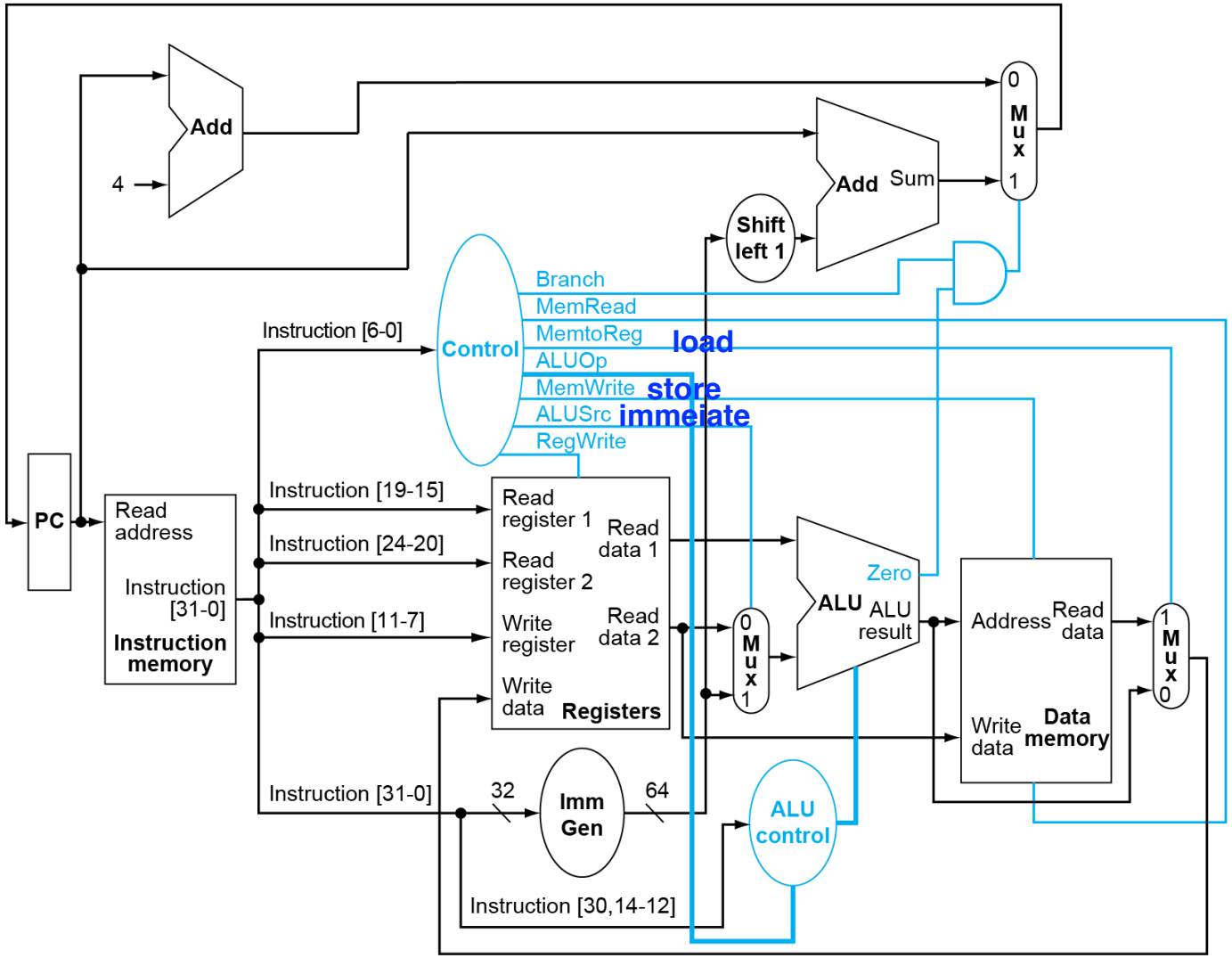
imme 필요하지 않더라도 일단 rs로 해석하고 register 로 보내고 나중에 필요 없으면 버린다.

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

left most bit! (always) -> wenn need sign extend replicationg sign bit needed!!!

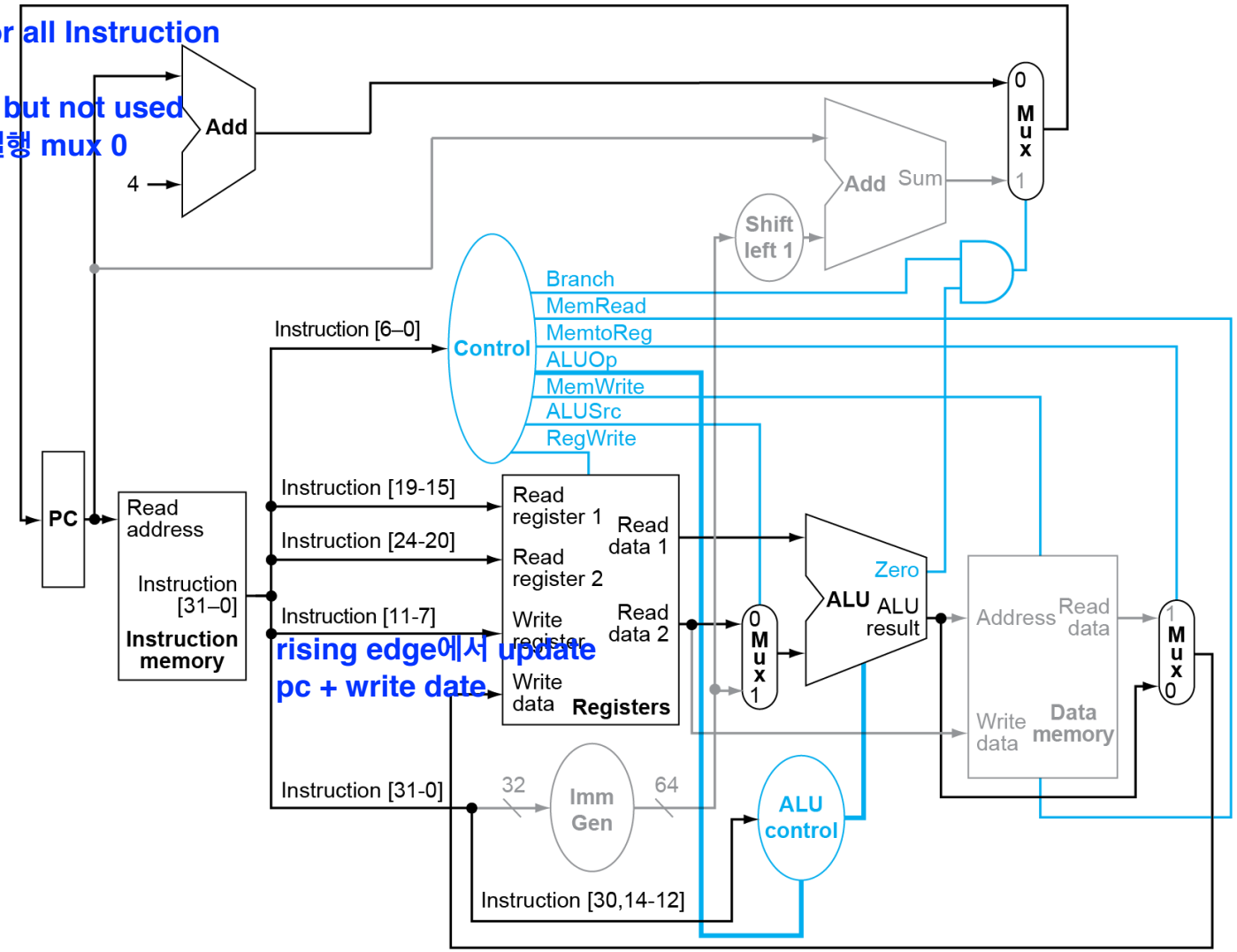
ALUOp		Funct7 field							Funct3 field			
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operation
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

Datapath with Control

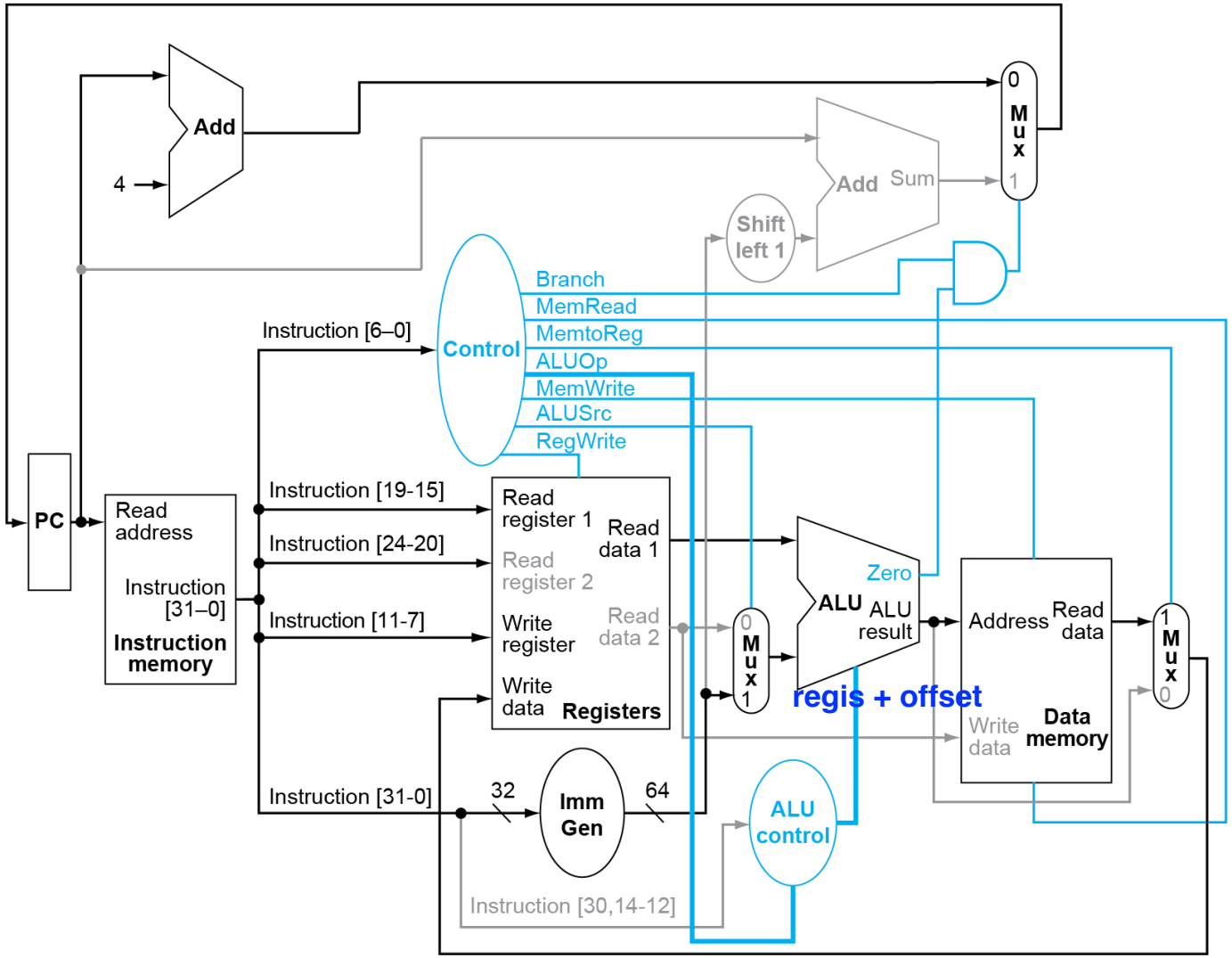


R-Type Instruction

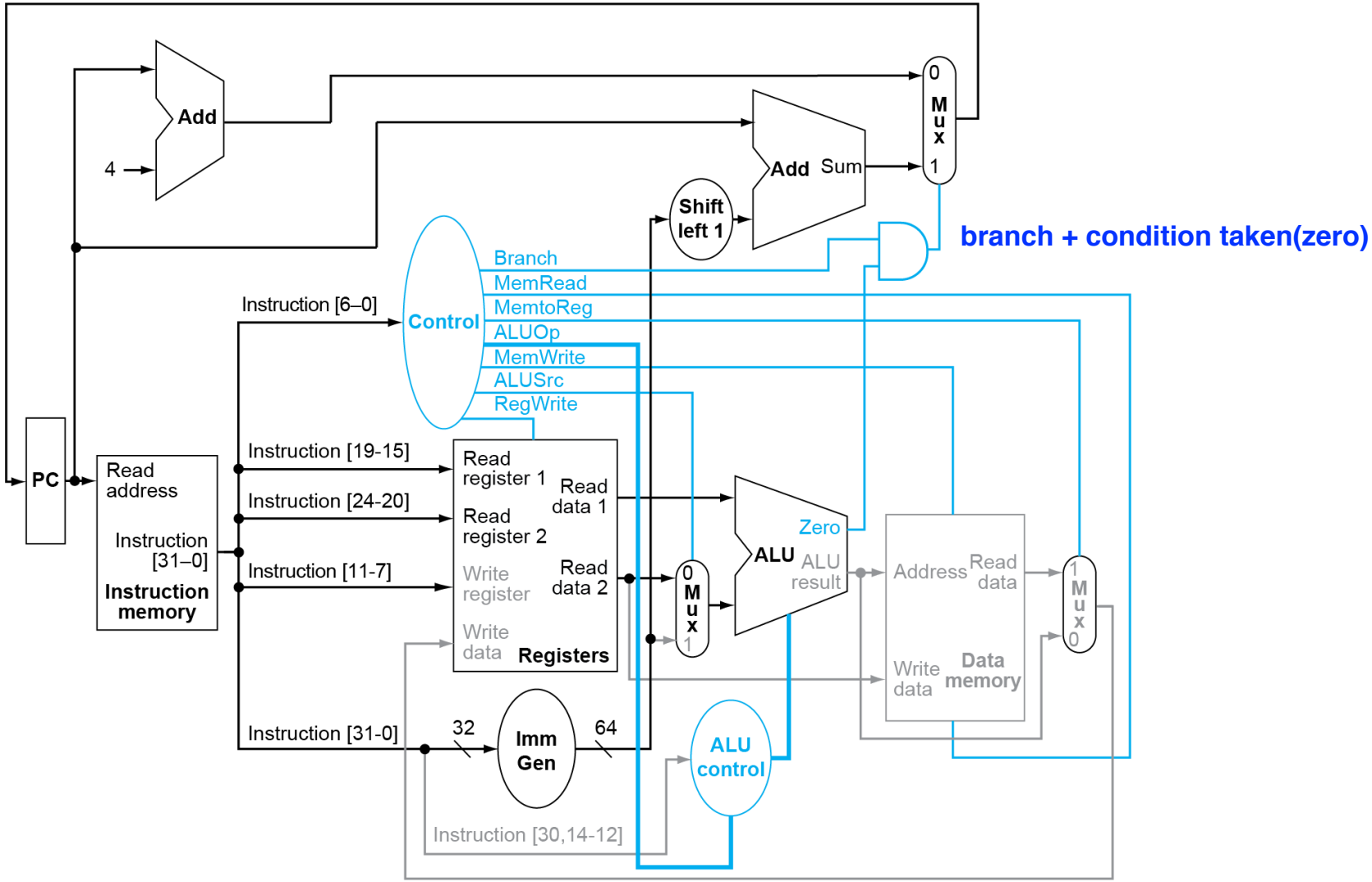
ALU
accessing은 same for all Instruction
decoding peparing
immediate prepared but not used
rs1 rs2 passed alu 실행 mux 0



Load Instruction



BEQ Instruction



the only change => pc

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