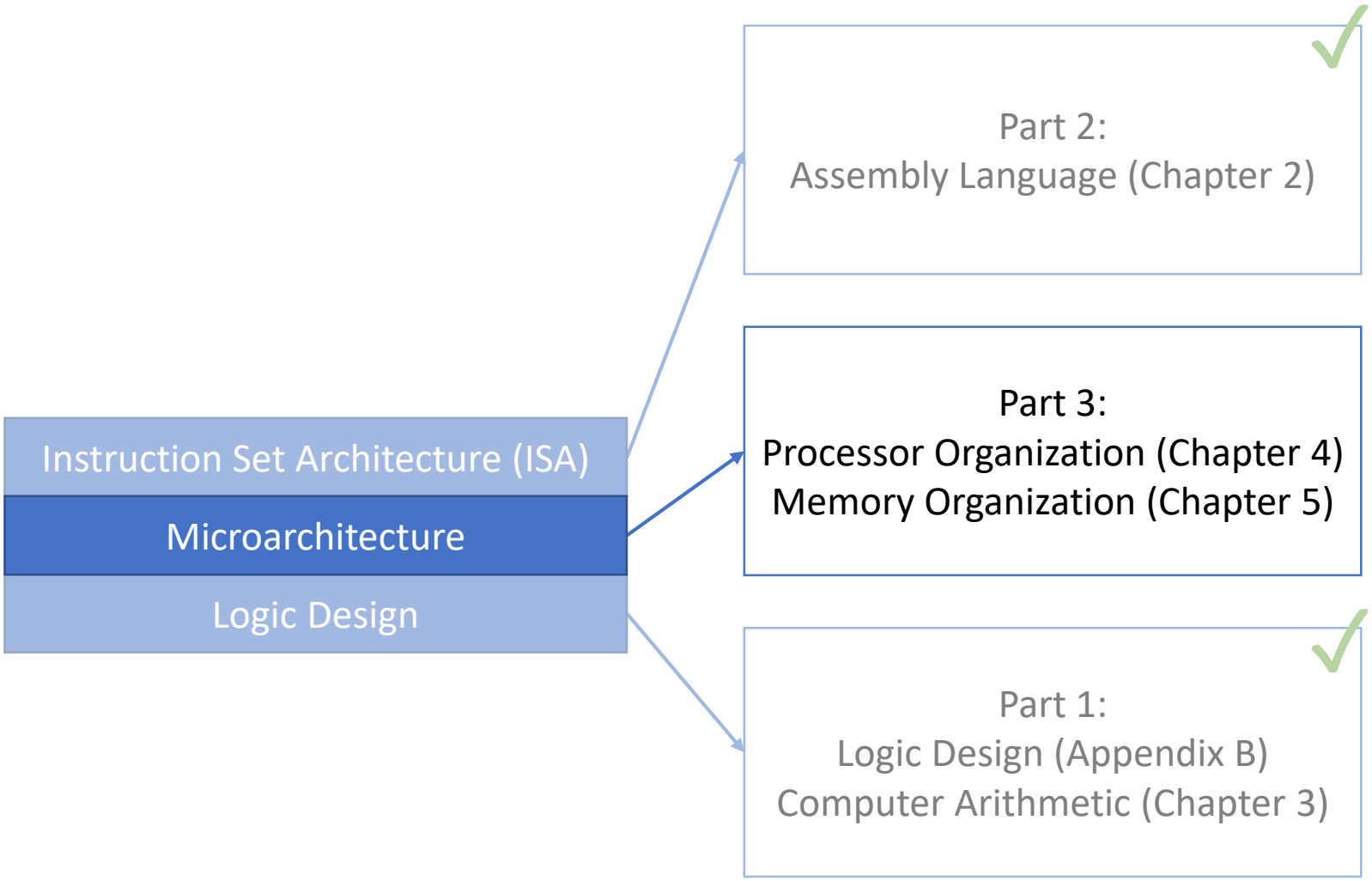
Lecture 22: Datapath Control Signals

CMPS 221 – Computer Organization and Design

Adapted by Izzat El Hajj from textbook slides for Computer Organization and Design, by David Patterson and John Hennessy, Morgan Kaufmann.

Last Time: Building a Datapath



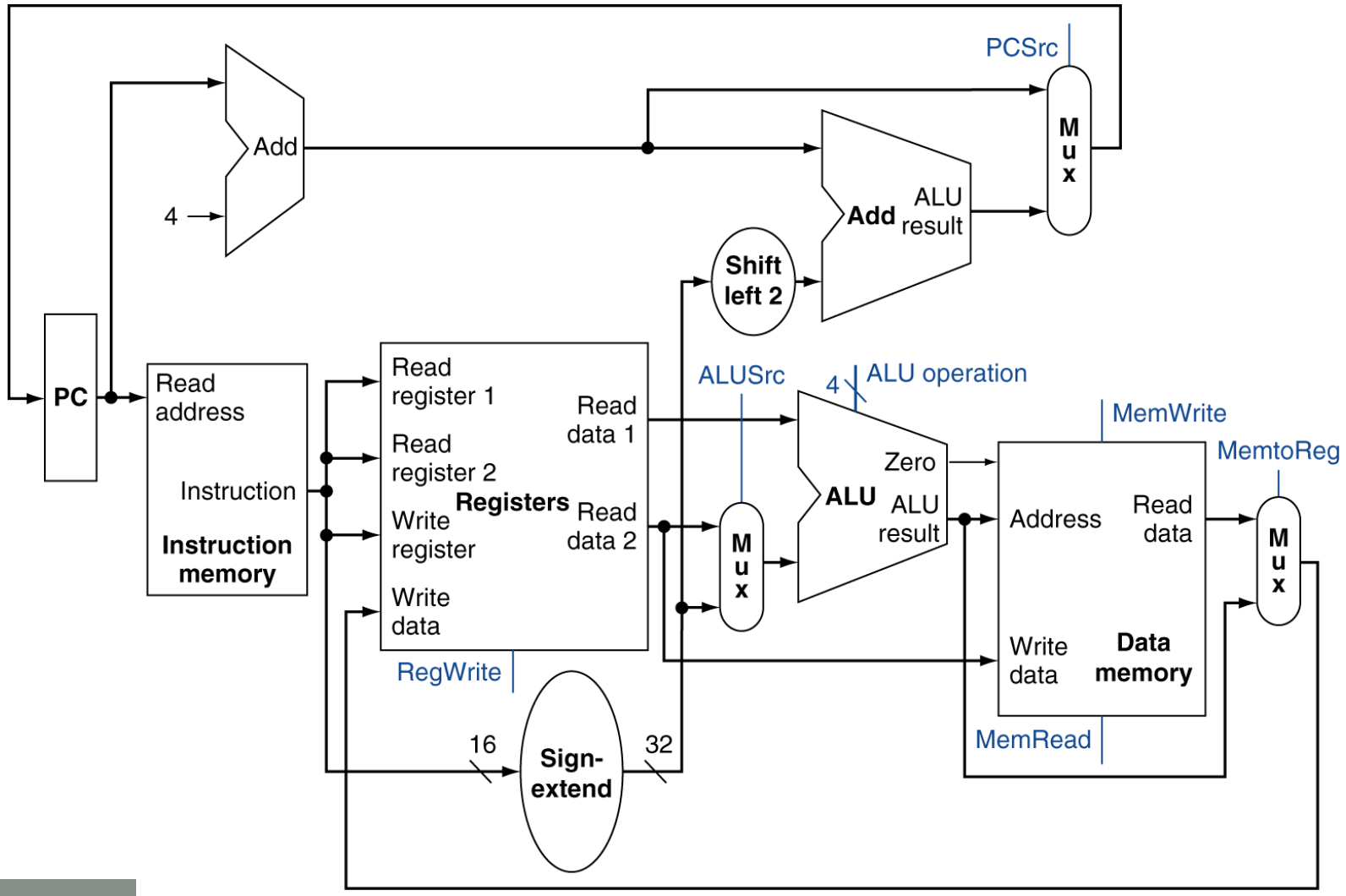
Introduction

- 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
- Focus on a simple subset (shows most aspects)
  - Arithmetic/logical: `add`, `sub`, `addi`, `slt`
  - Memory reference: `lw`, `sw`
  - Control transfer: `beq`, `j`

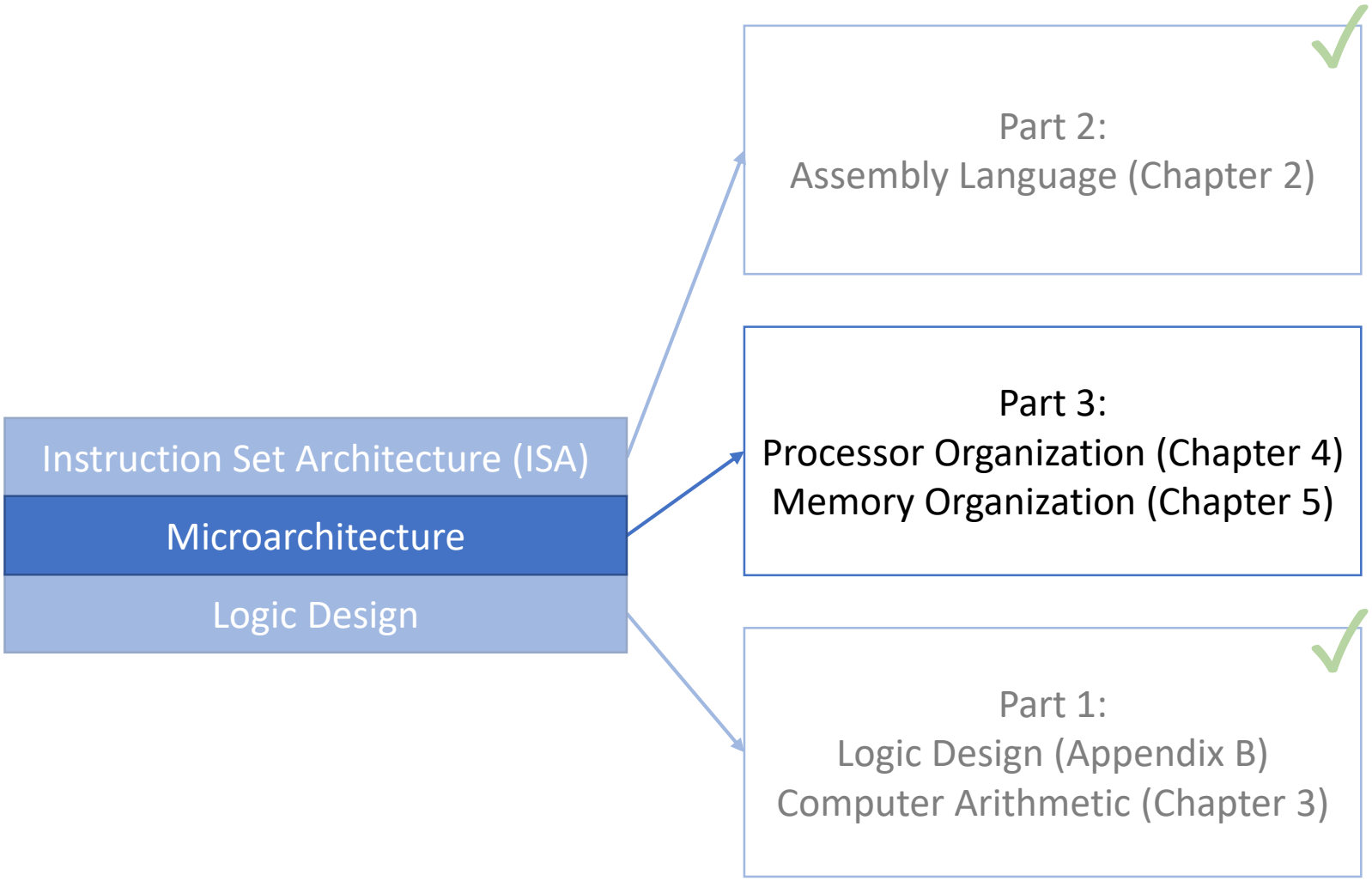
Building a Datapath

- A **datapath** is a set of elements that process data and addresses in the CPU
- Components of a MIPS datapath:
  - PC register, instruction memory (fetch instructions)
  - Register file (read/write registers)
  - ALUs:
    - Arithmetic/logic operations
    - Memory address for load/store
    - Branch target address
  - Data memory (load/store data)

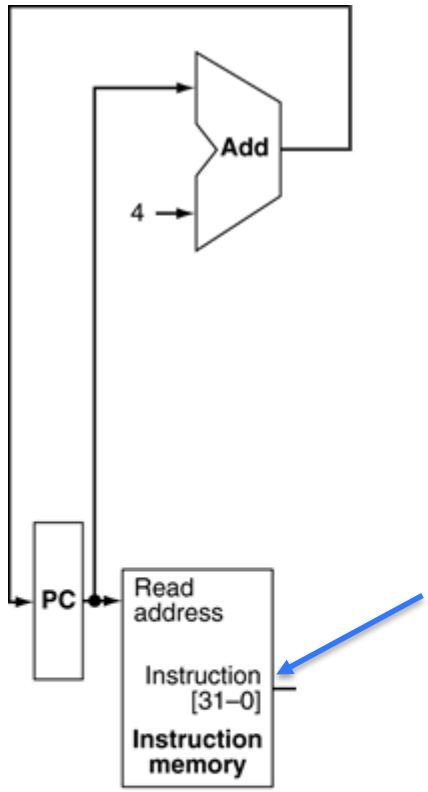
Full Datapath



Today: Datapath Control Signals

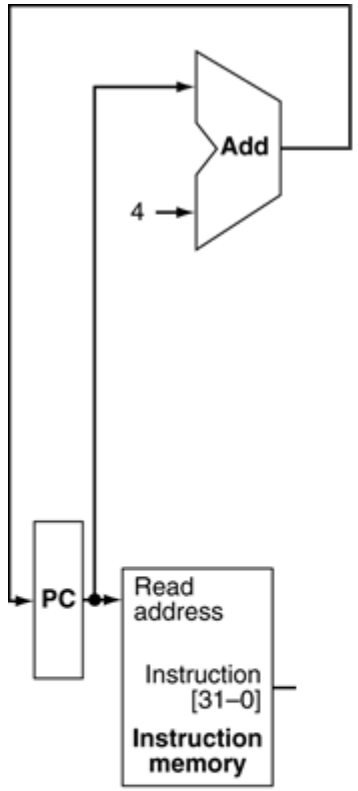


Instruction Fetch



Instruction is 32 bits which will be used to determine the execution

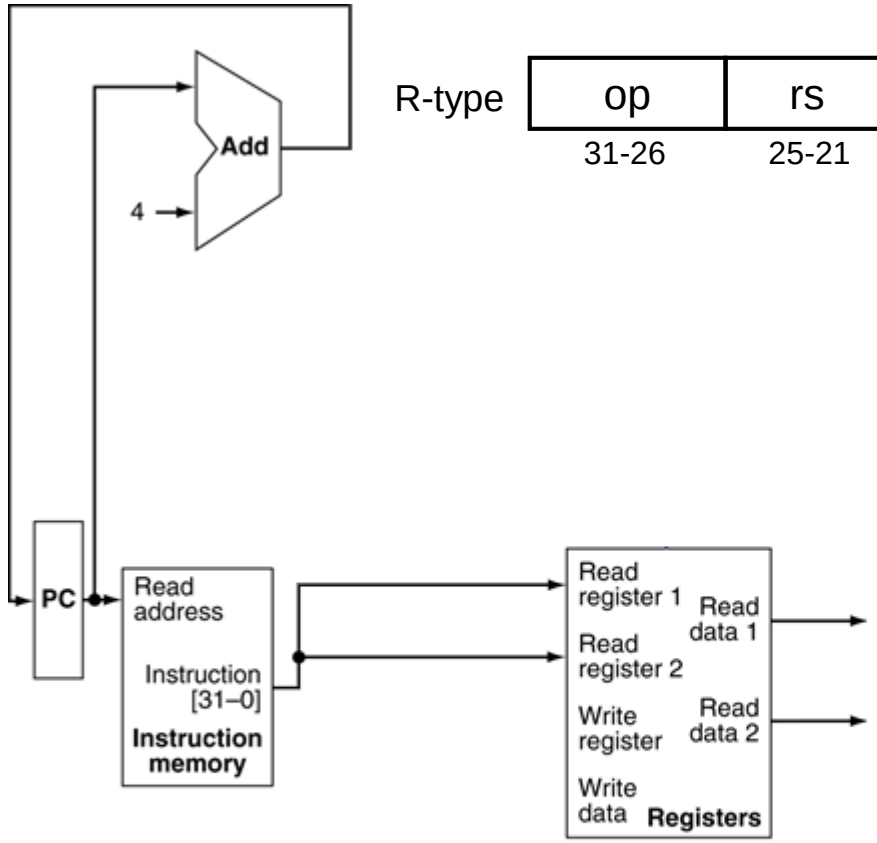
R-type Instructions



R-type

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

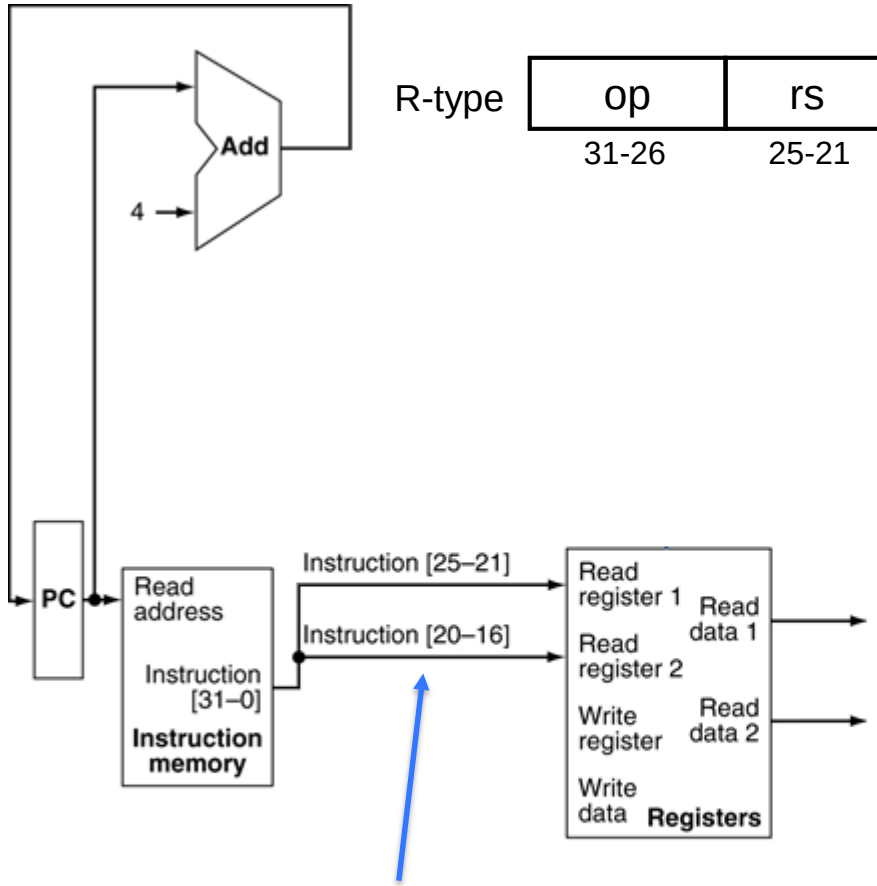
R-type Instructions



R-type

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

R-type Instructions

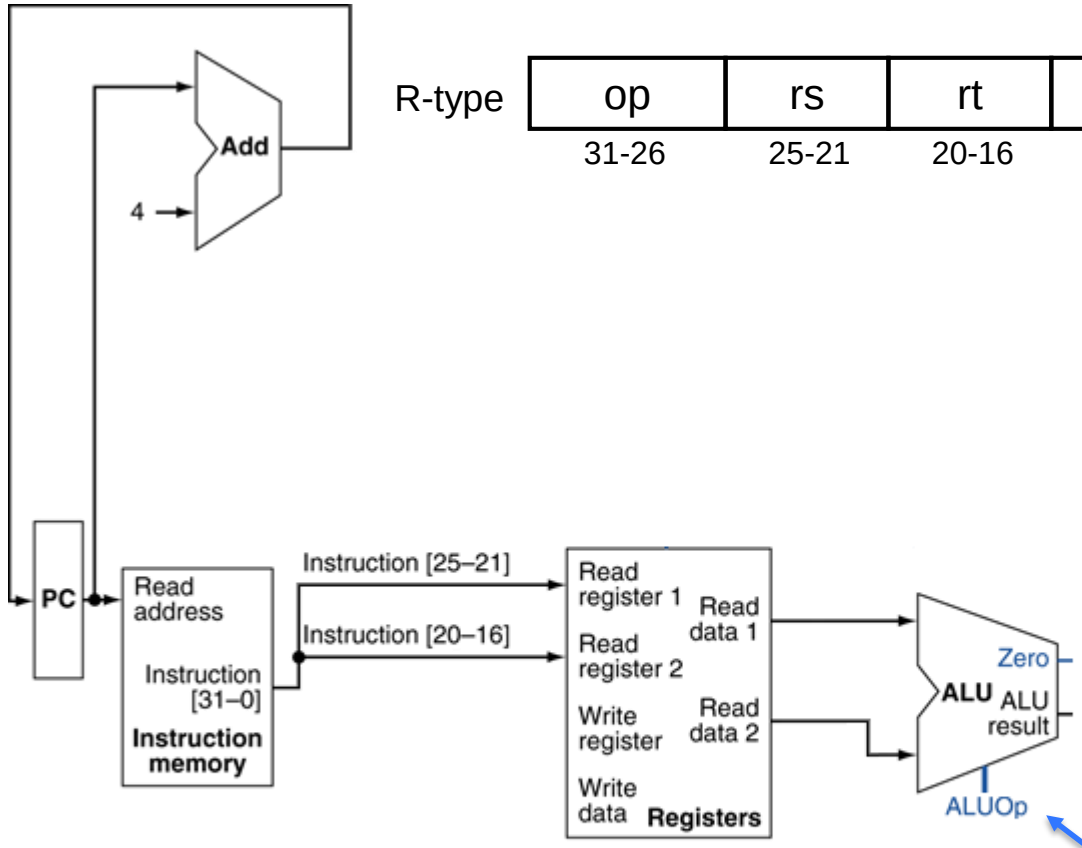


R-type

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

Source register numbers extracted from rs and rt bits

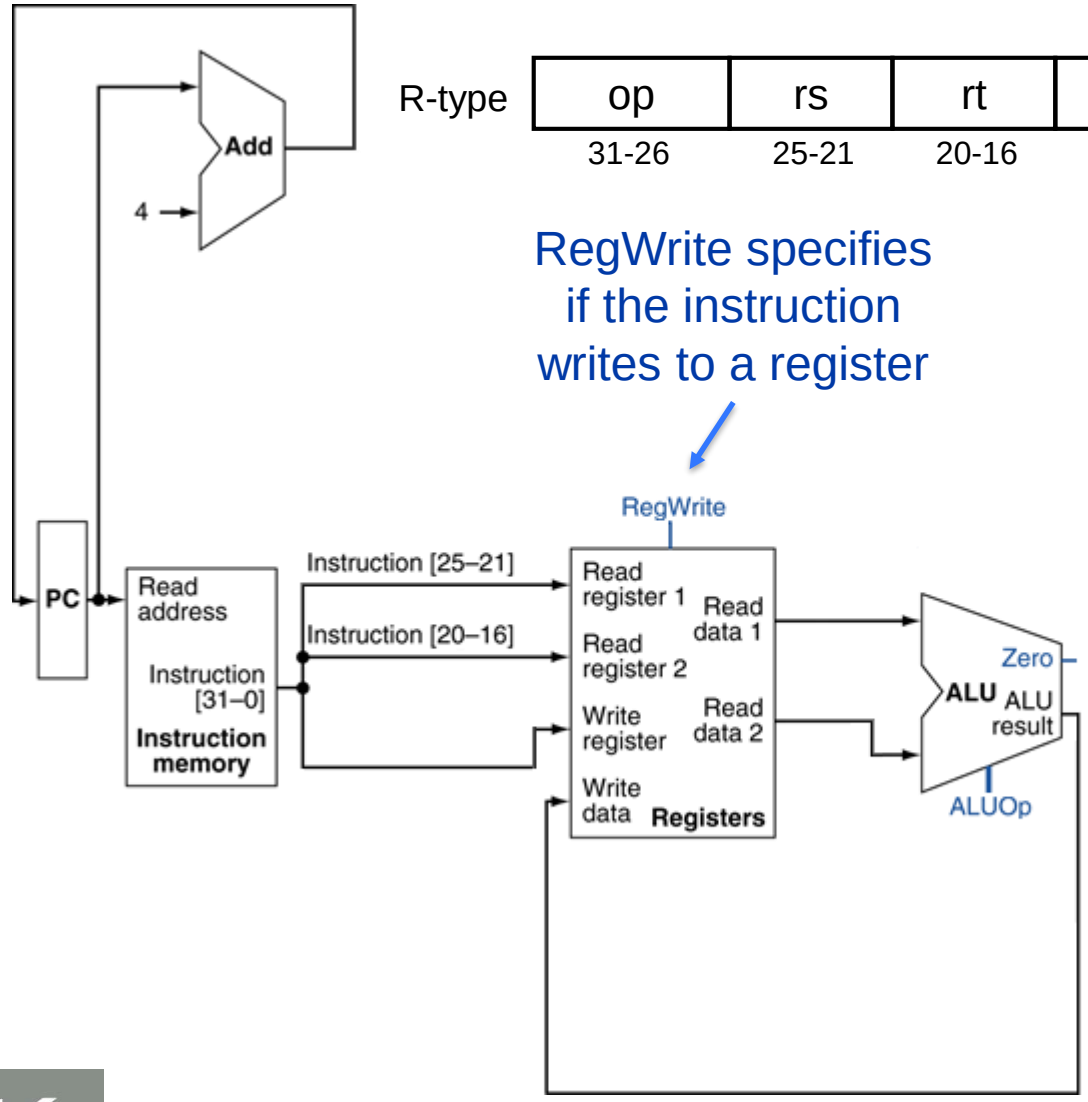
R-type Instructions



R-type	op	rs	rt	rd	shamt	funct
	31-26	25-21	20-16	15-11	10-6	5-0

ALUOp specifies the operation done by the ALU

R-type Instructions

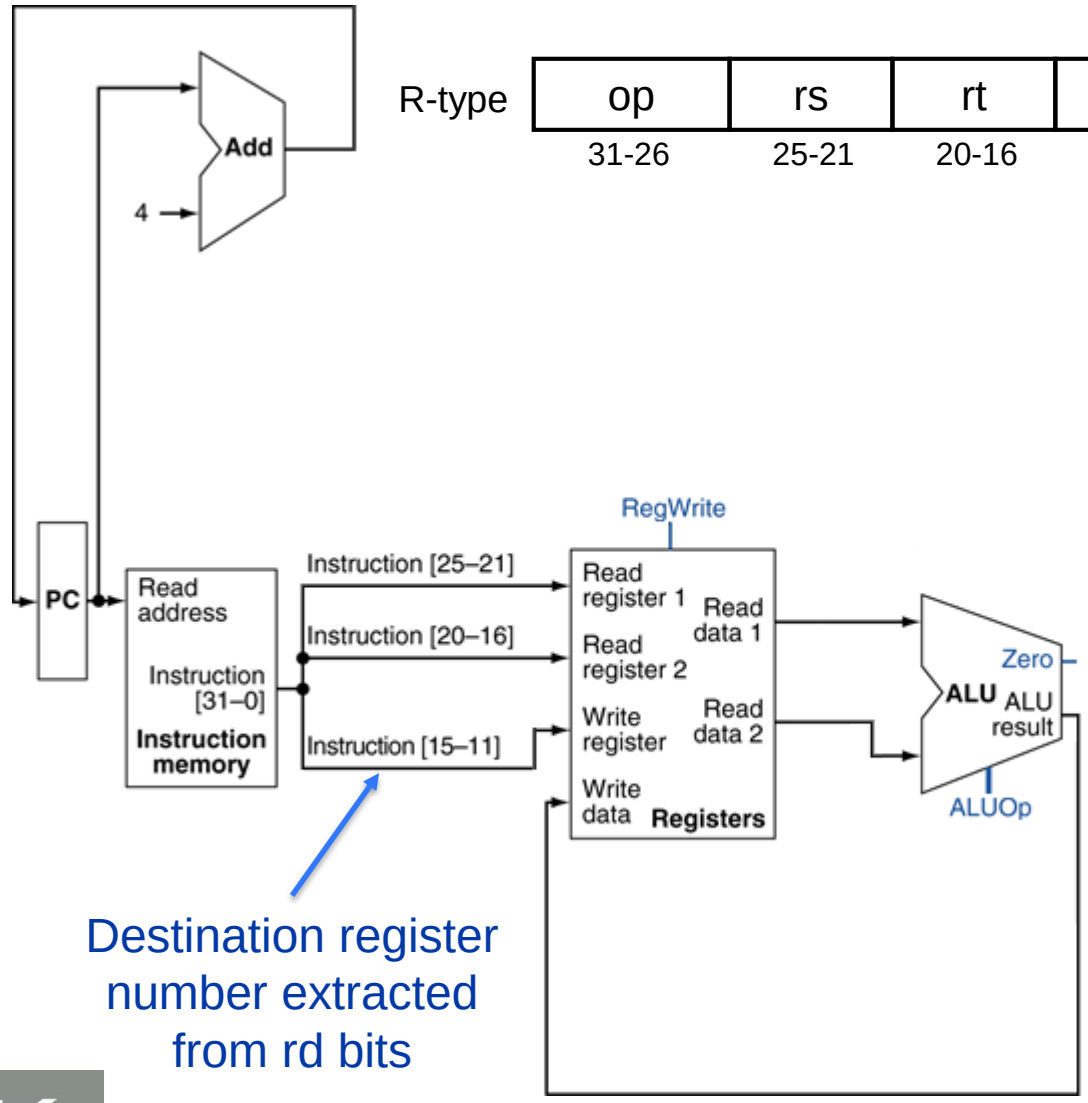


R-type

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

RegWrite specifies if the instruction writes to a register

R-type Instructions

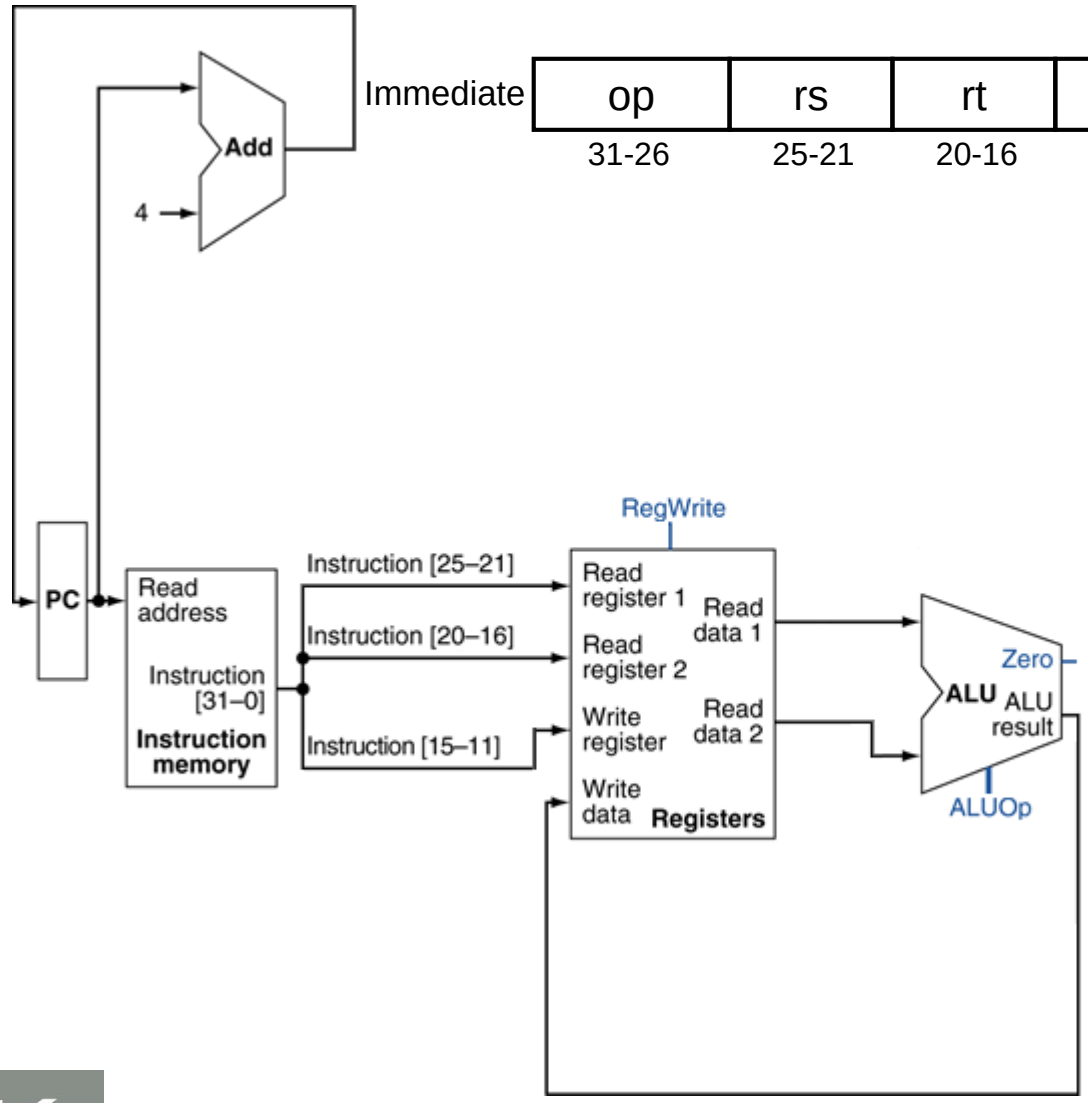


R-type

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

Destination register number extracted from rd bits

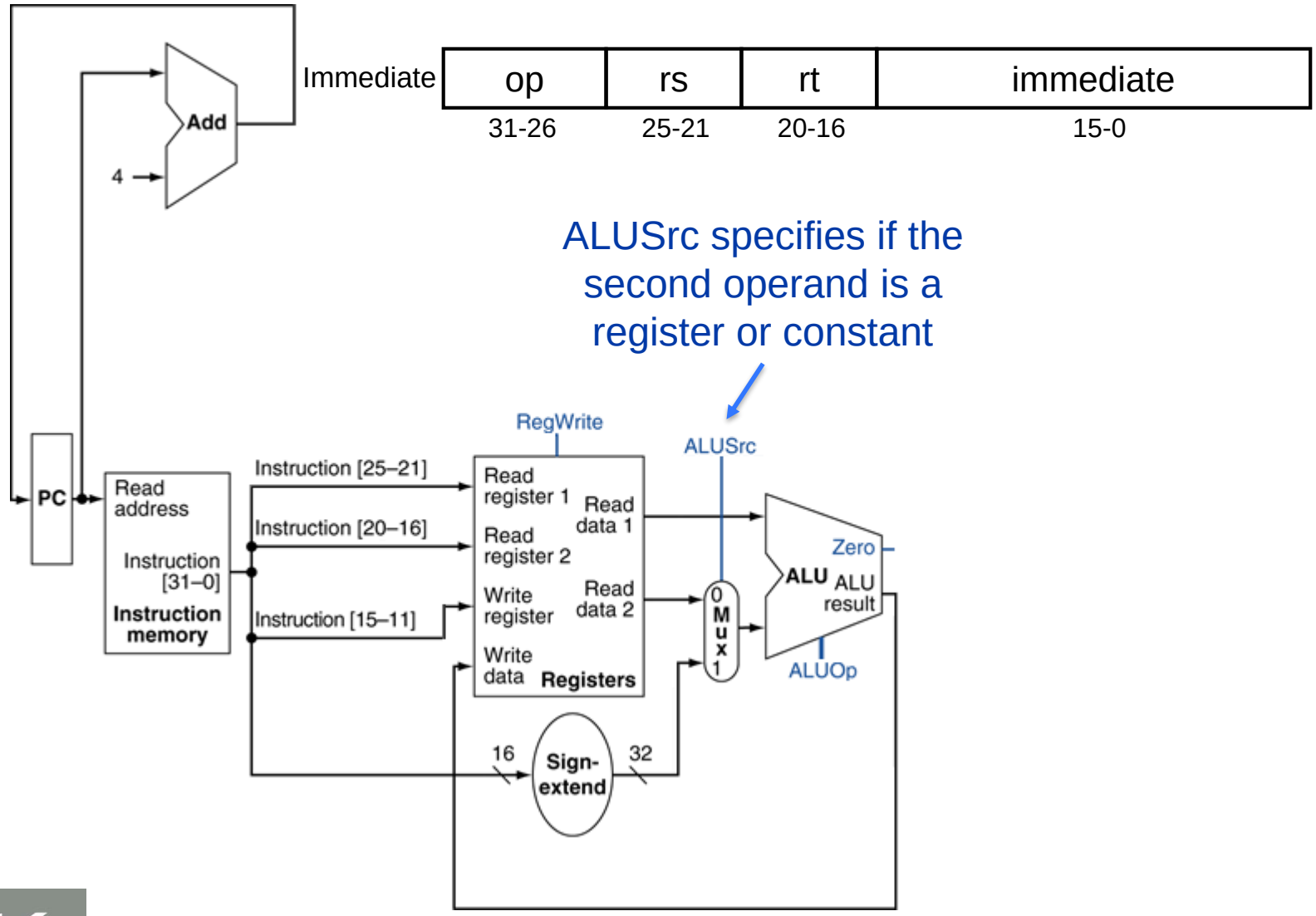
Immediate Instructions



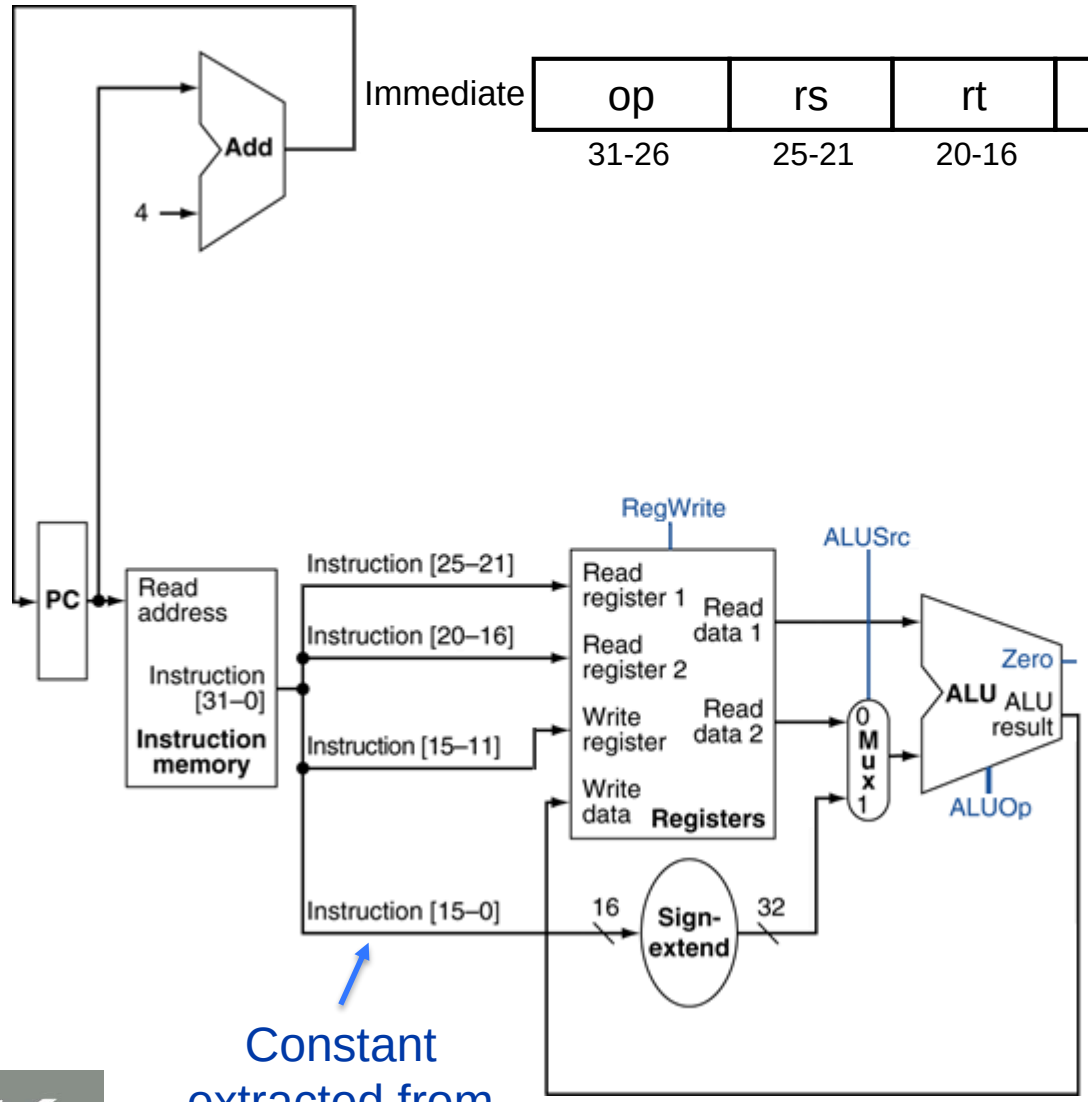
Immediate

op	rs	rt	immediate
31-26	25-21	20-16	15-0

Immediate Instructions



Immediate Instructions

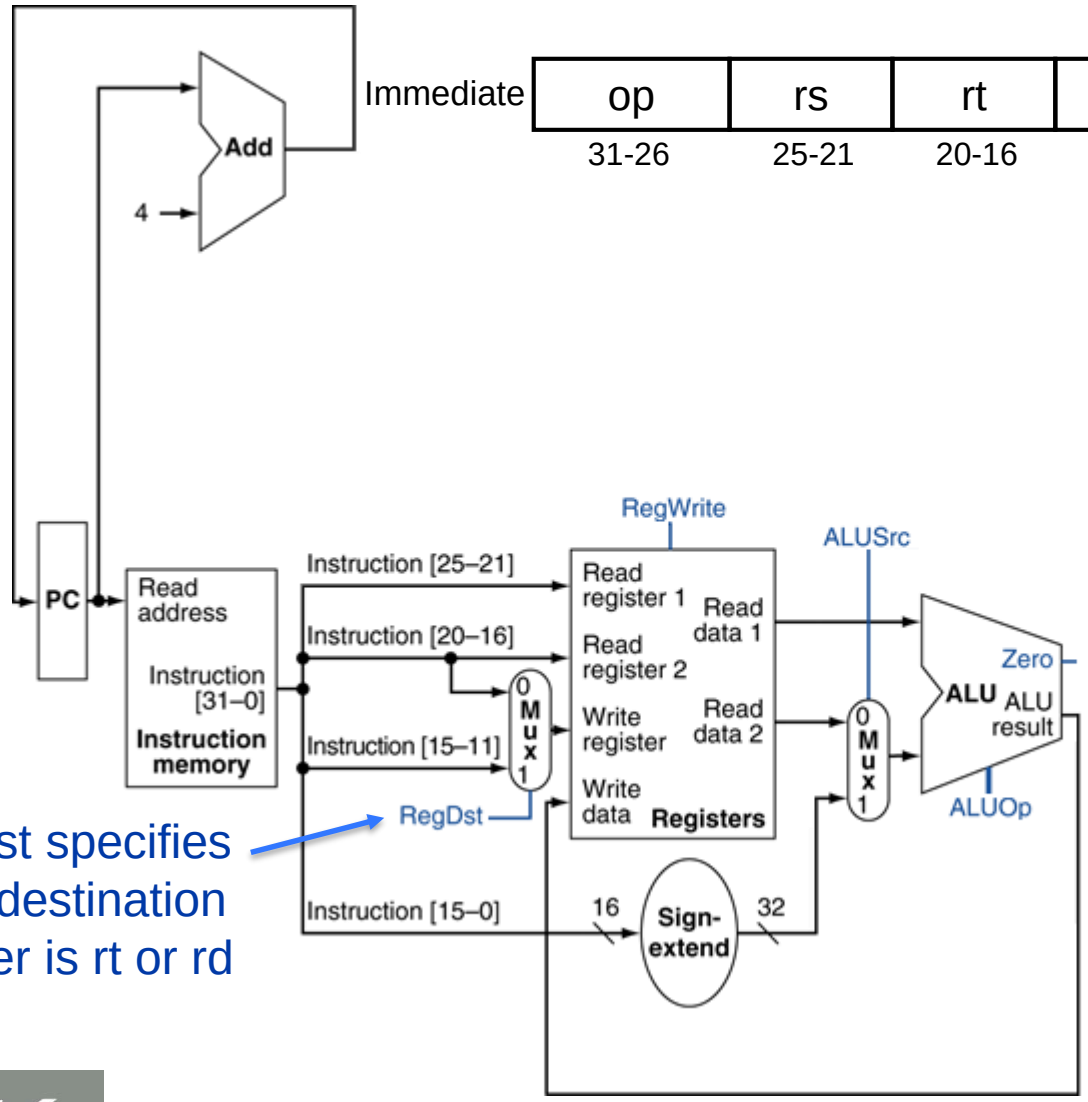


Immediate

op	rs	rt	immediate
31-26	25-21	20-16	15-0

Constant extracted from immediate field

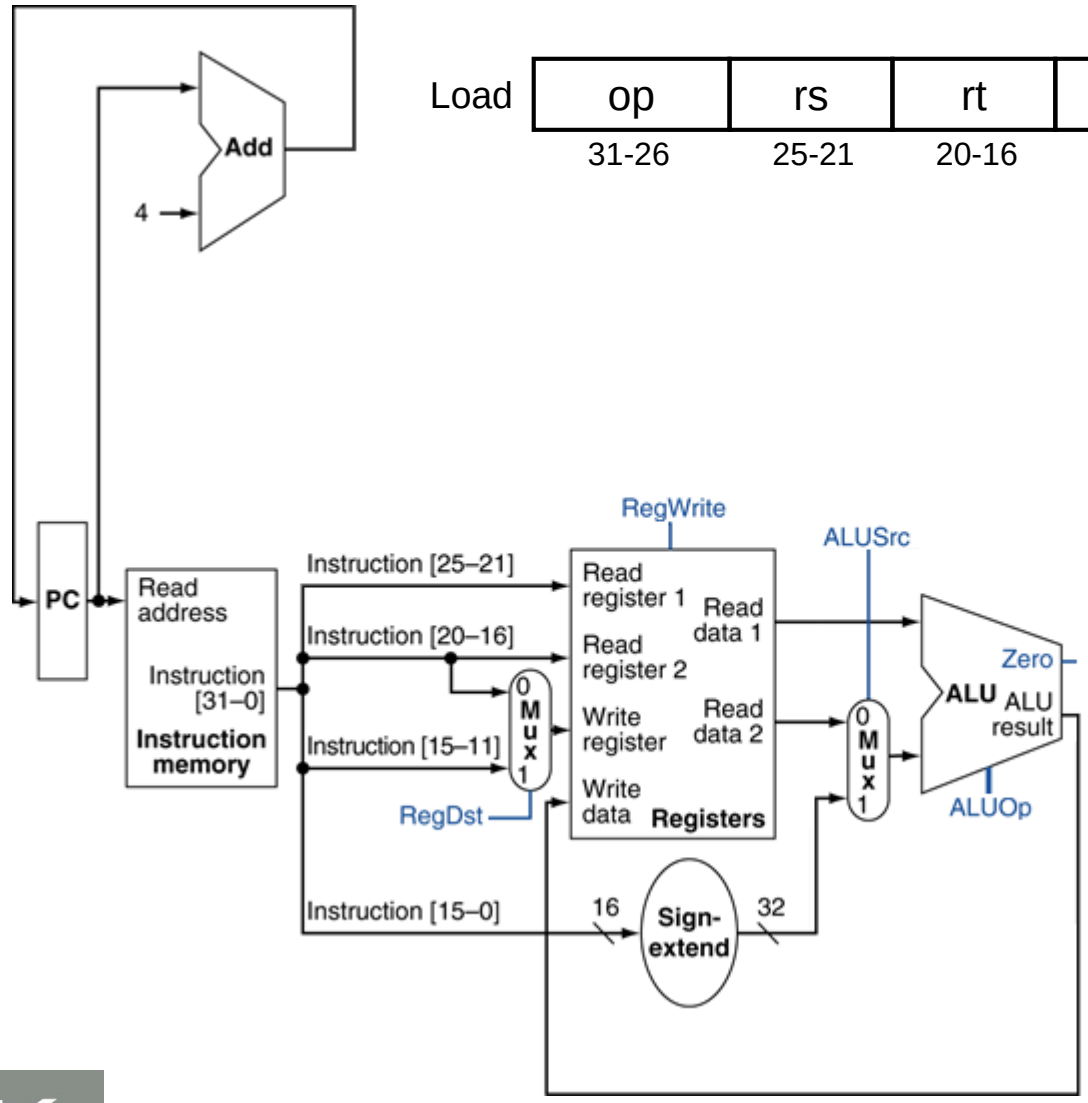
Immediate Instructions



Immediate	op	rs	rt	immediate
	31-26	25-21	20-16	15-0

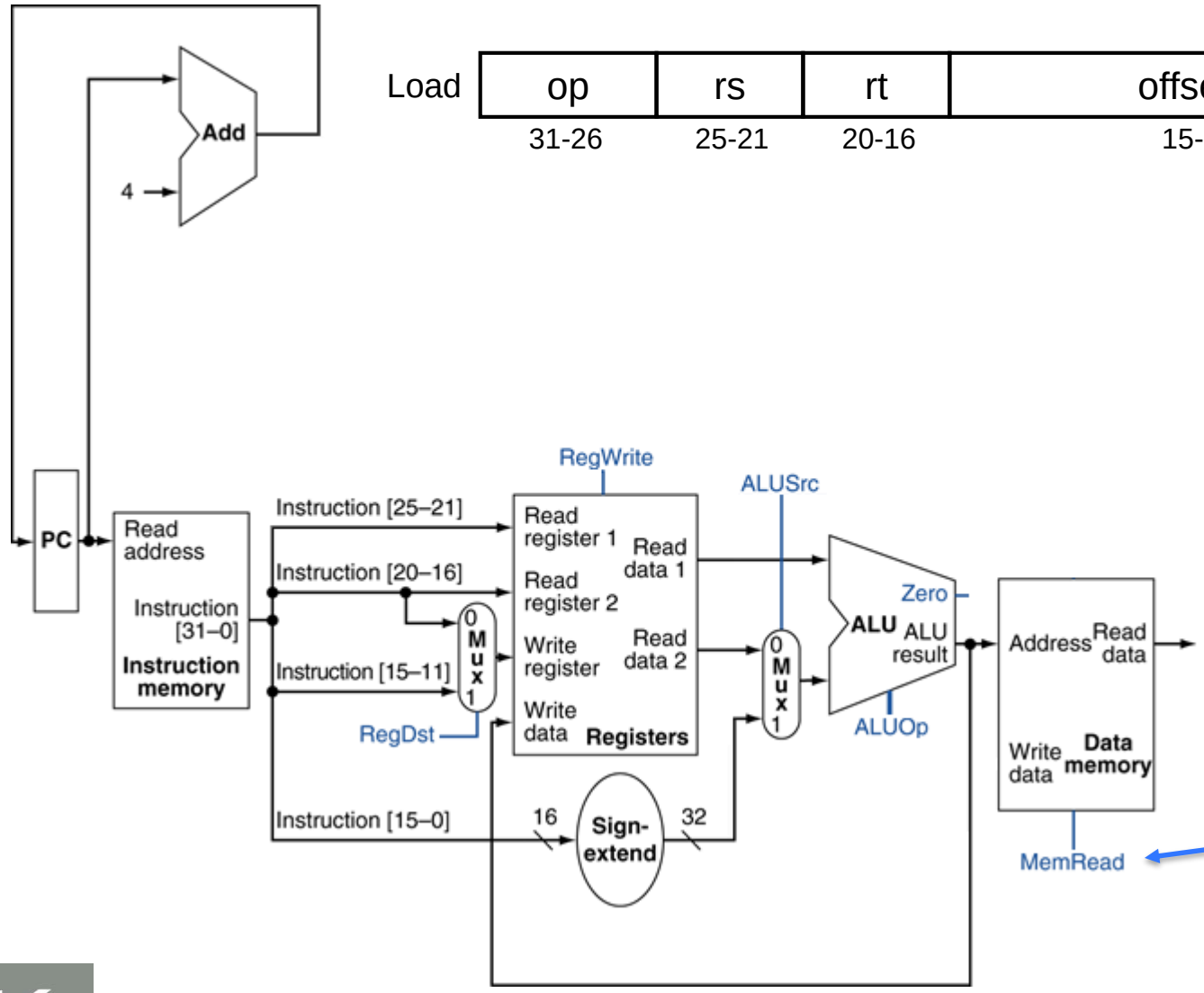
RegDst specifies if the destination register is rt or rd

Load Instructions



Load	op	rs	rt	offset
	31-26	25-21	20-16	15-0

Load Instructions

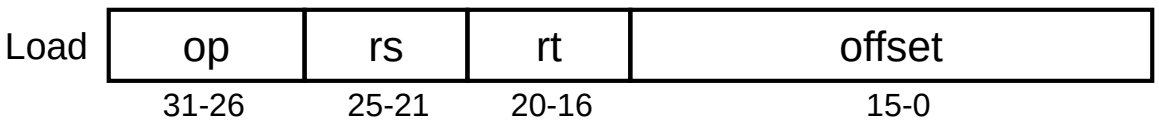


Load

op	rs	rt	offset
31-26	25-21	20-16	15-0

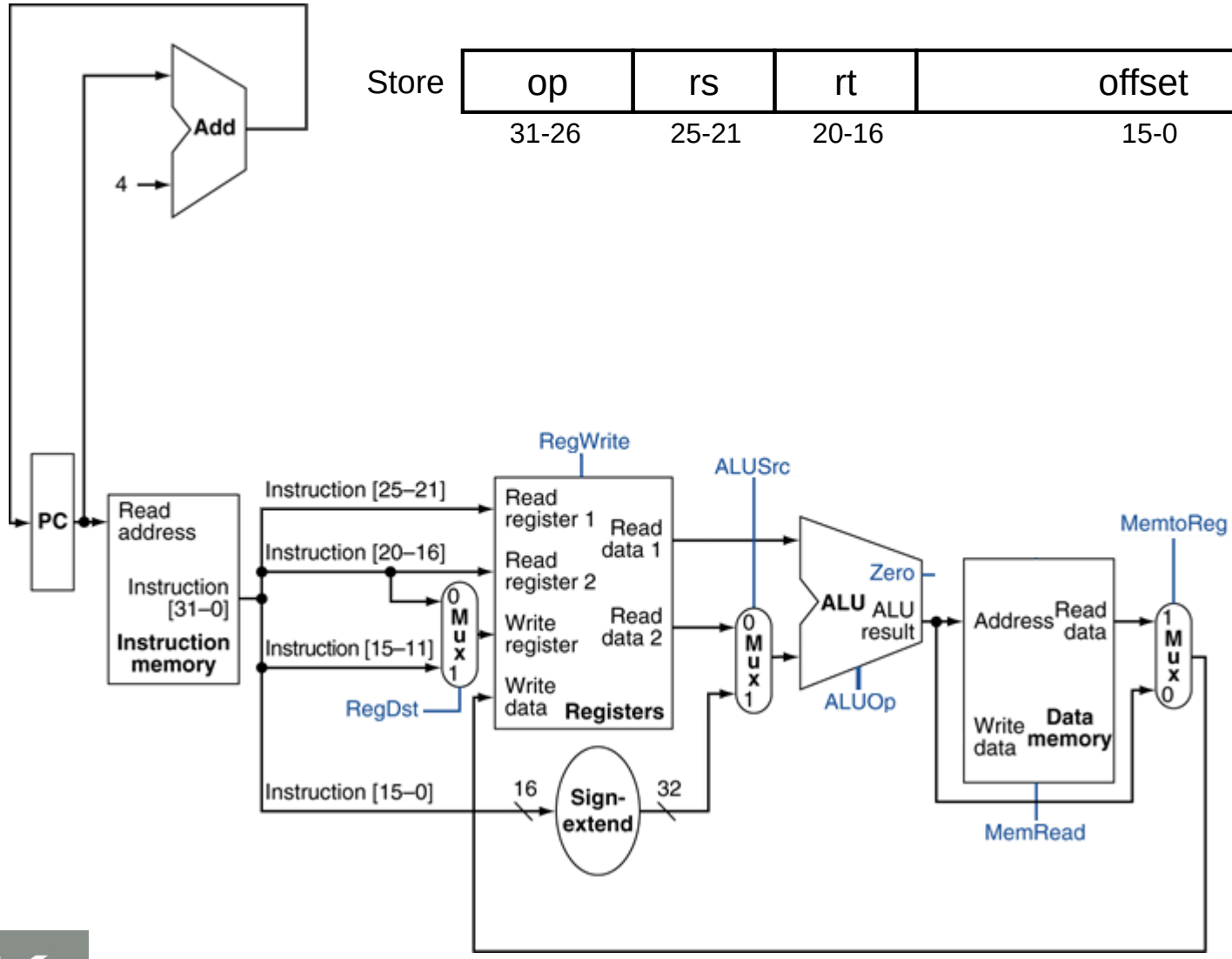
MemRead specifies if memory is read

Load Instructions



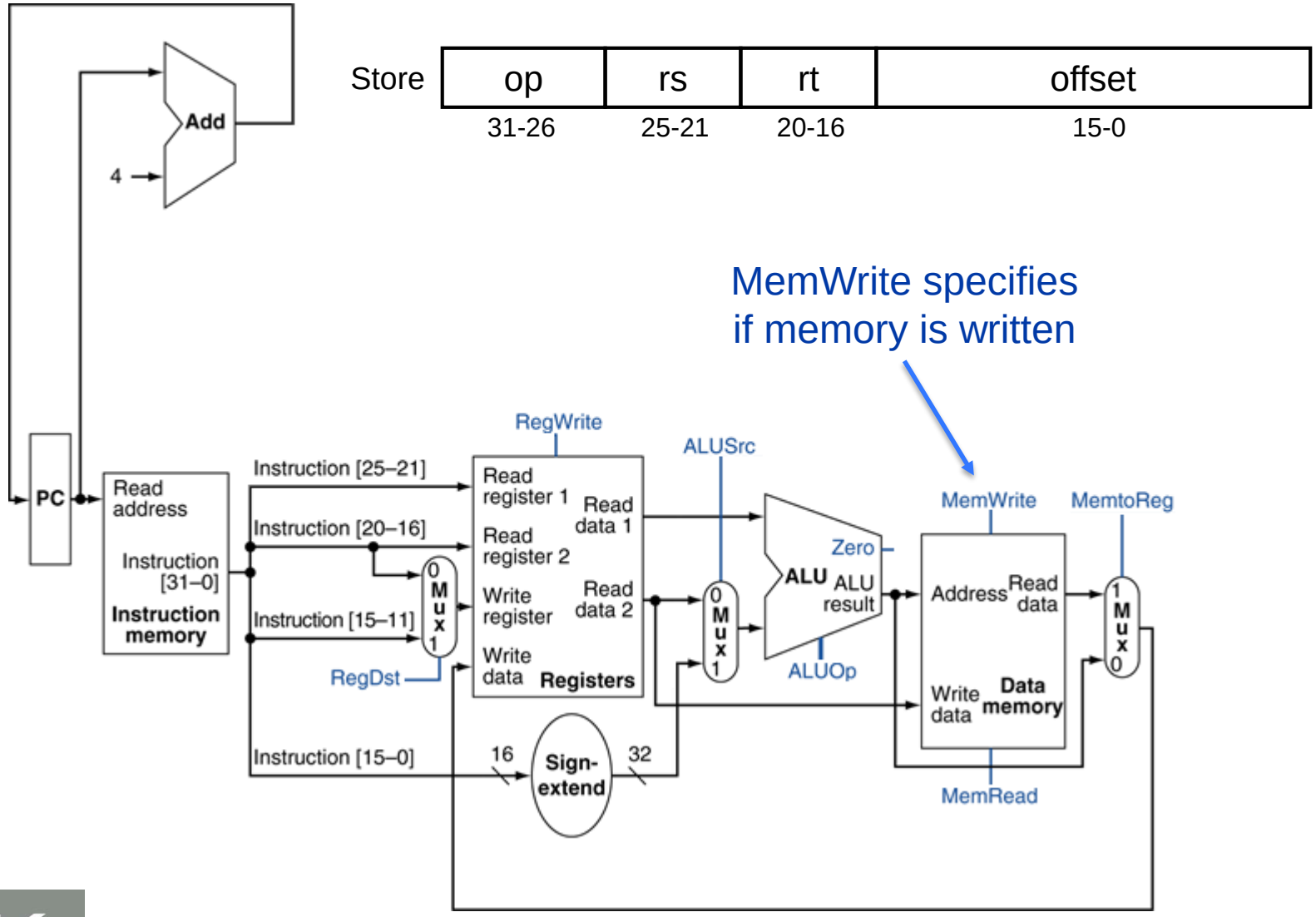
MemtoReg specifies if the memory or ALU output is written back

Store Instructions

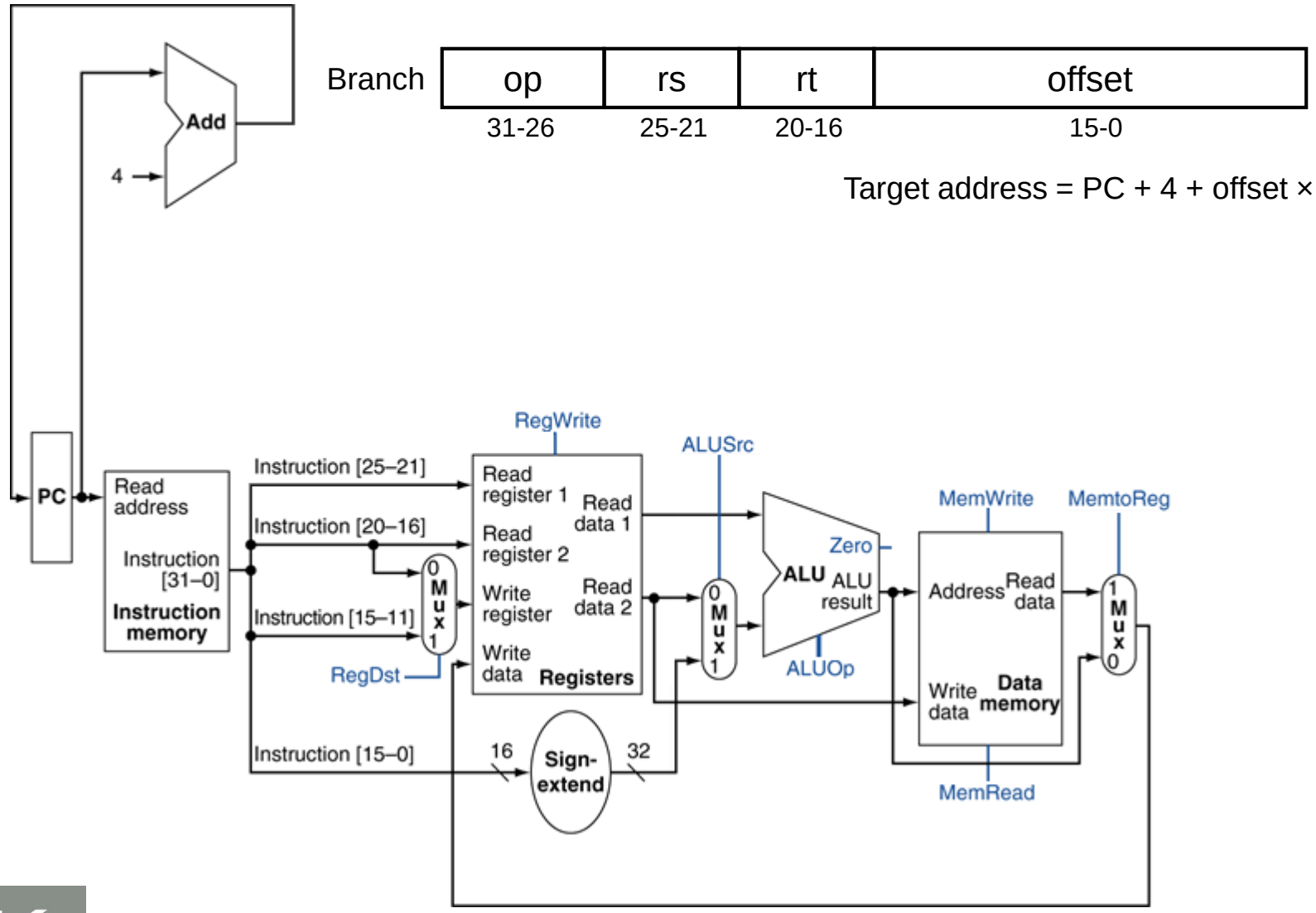


Store	op	rs	rt	offset
	31-26	25-21	20-16	15-0

Store Instructions



Branch Instructions

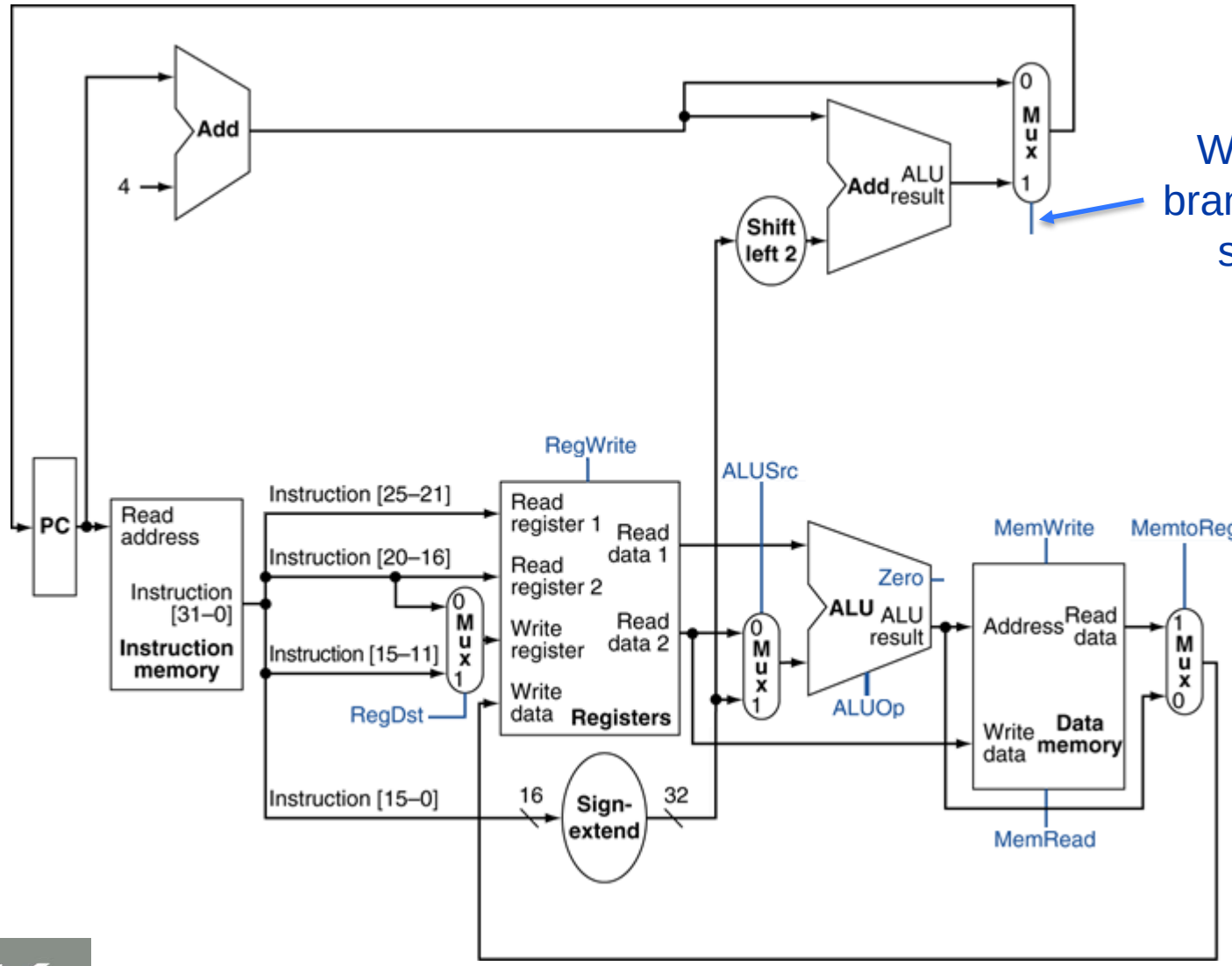


Branch

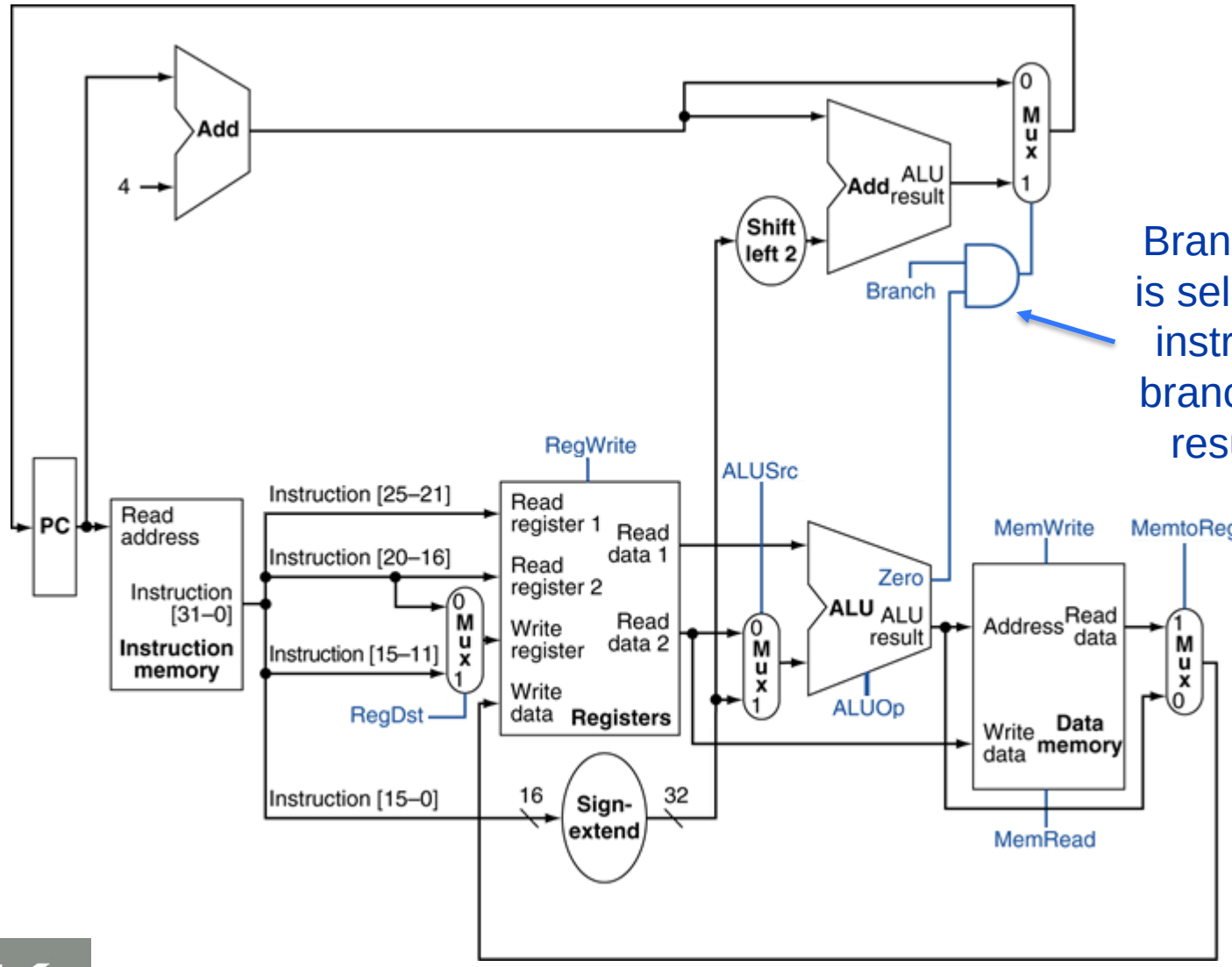
op	rs	rt	offset
31-26	25-21	20-16	15-0

Target address = PC + 4 + offset × 4

Branch Instructions

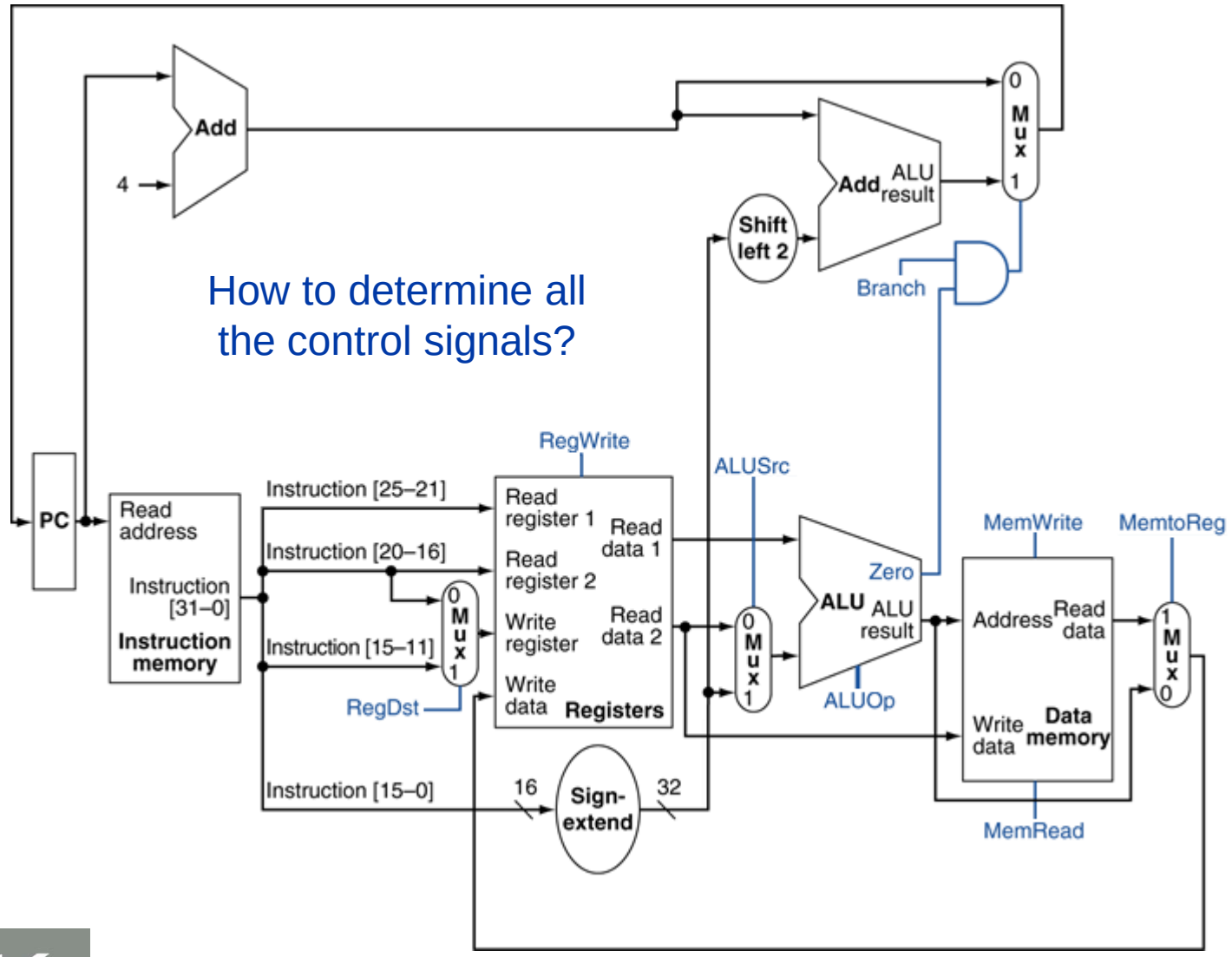


Branch Instructions

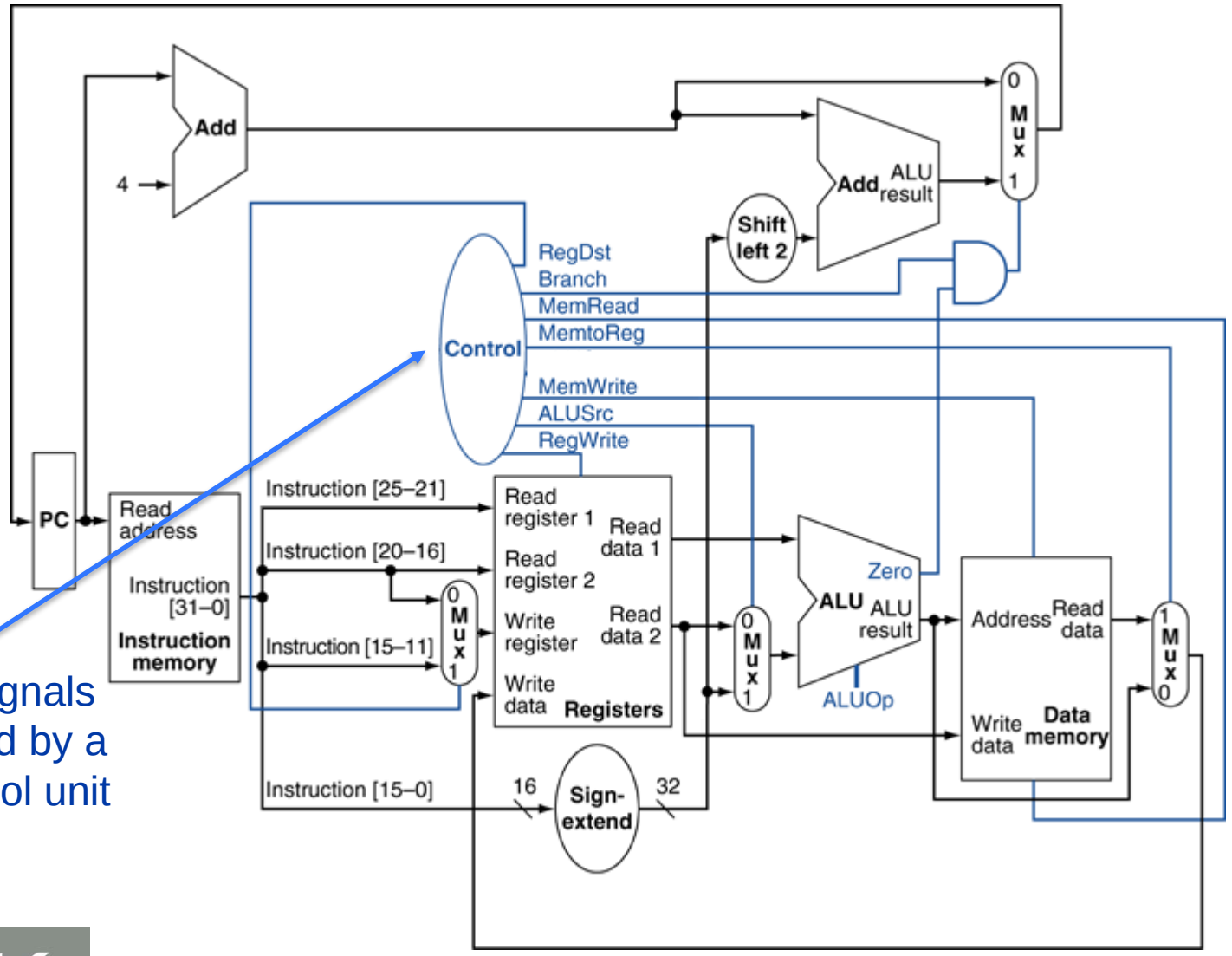


Branch address is selected when instruction is a branch and ALU result is zero

Main Control Unit

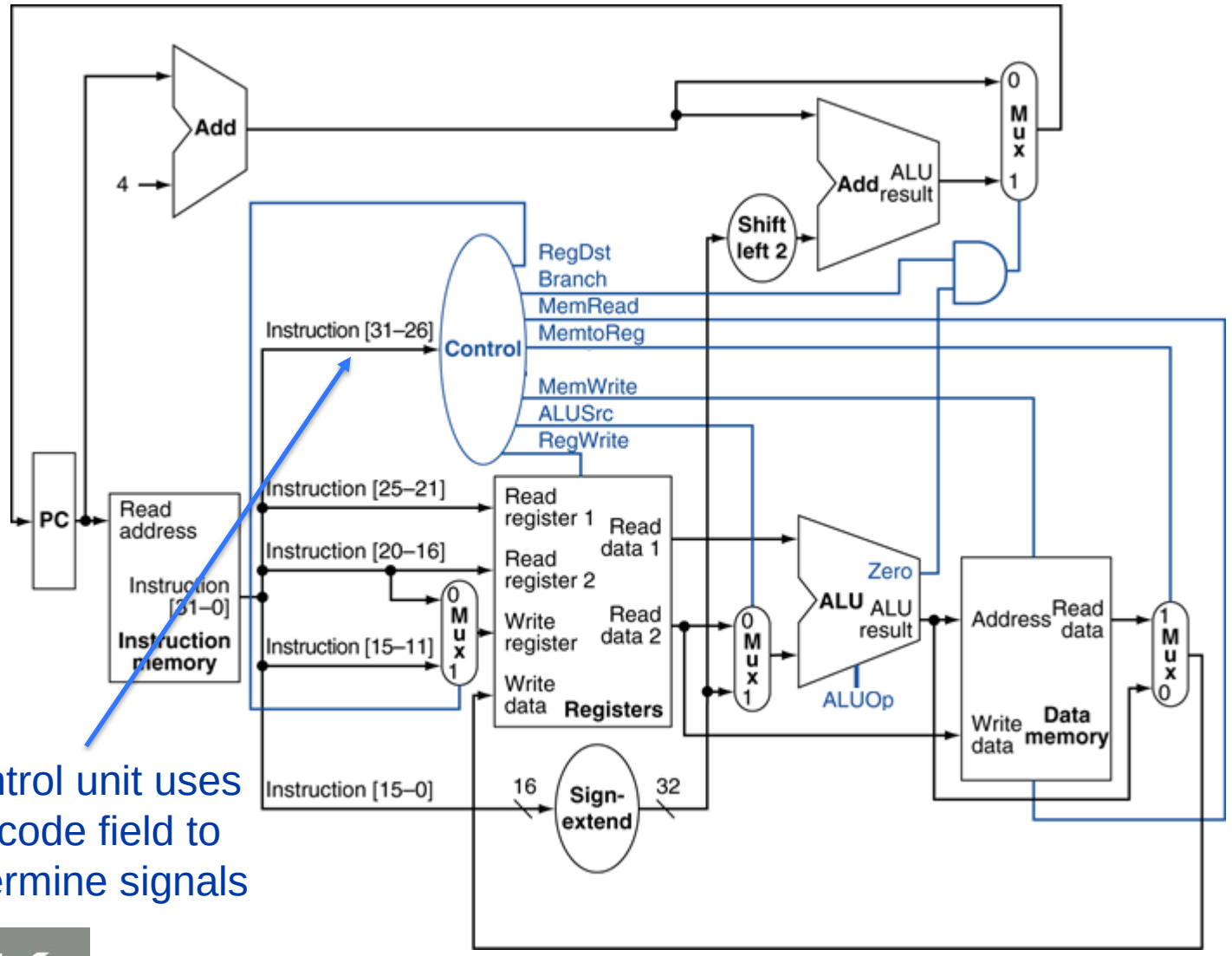


Main Control Unit



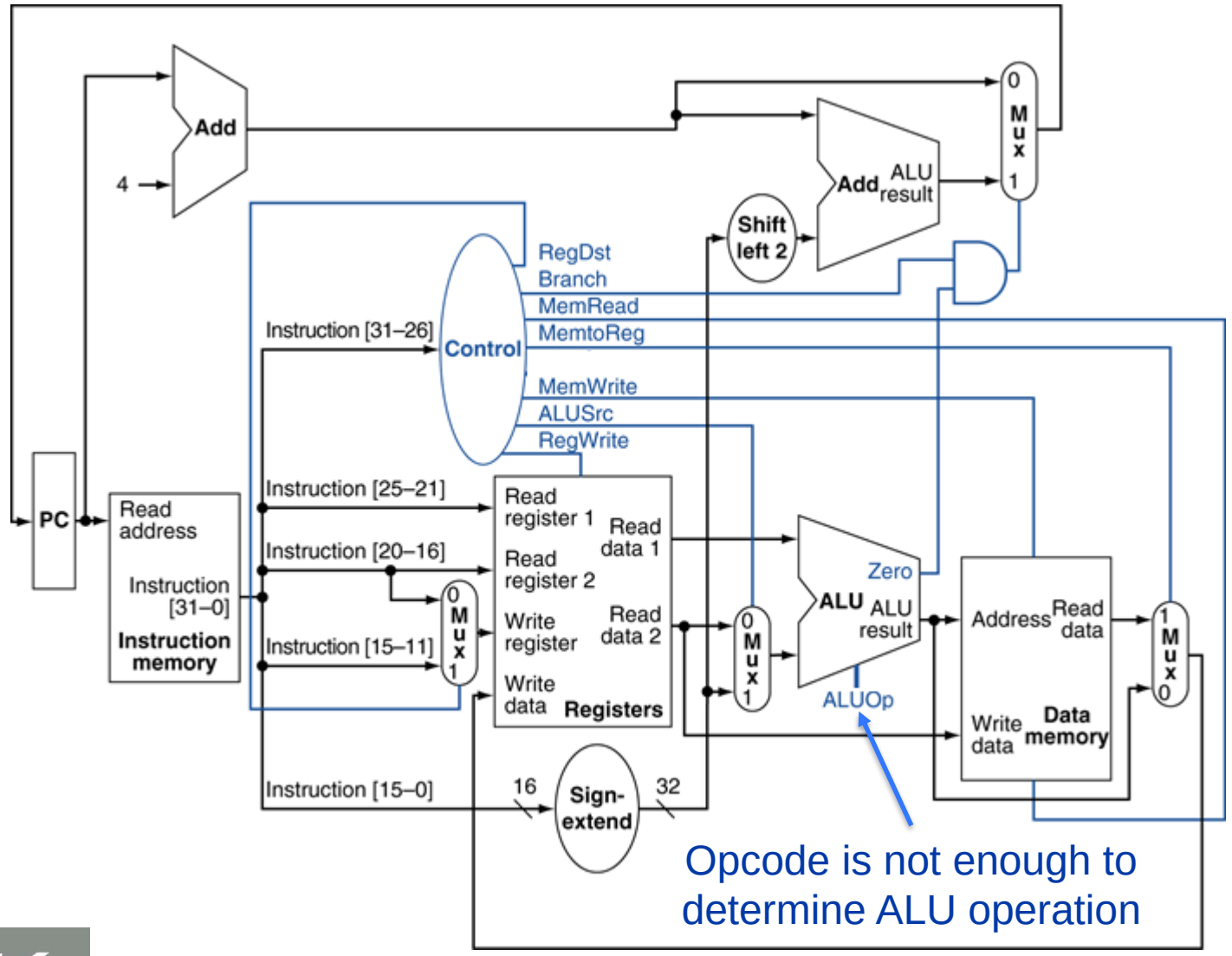
Control signals determined by a main control unit

Main Control Unit

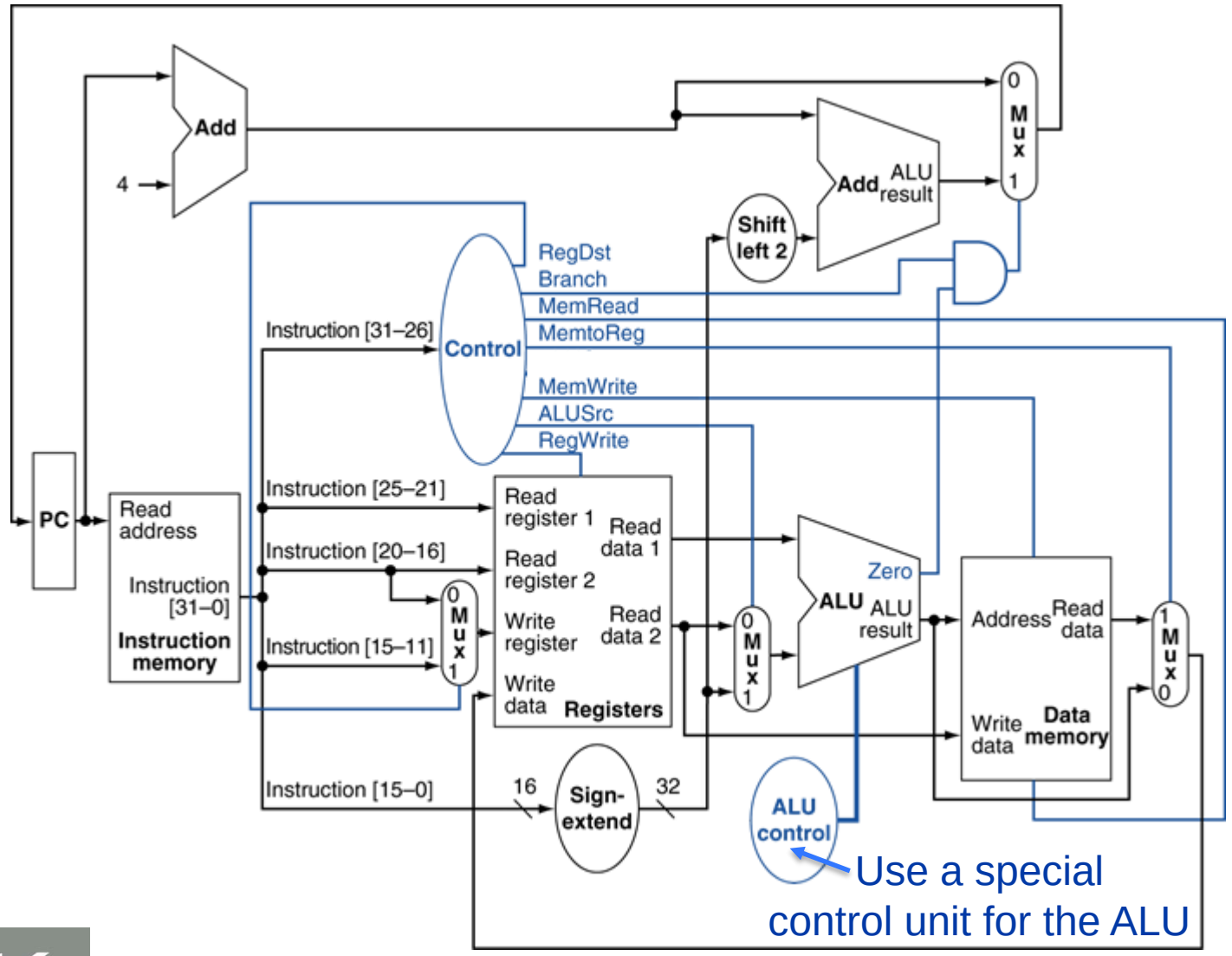


Control unit uses opcode field to determine signals

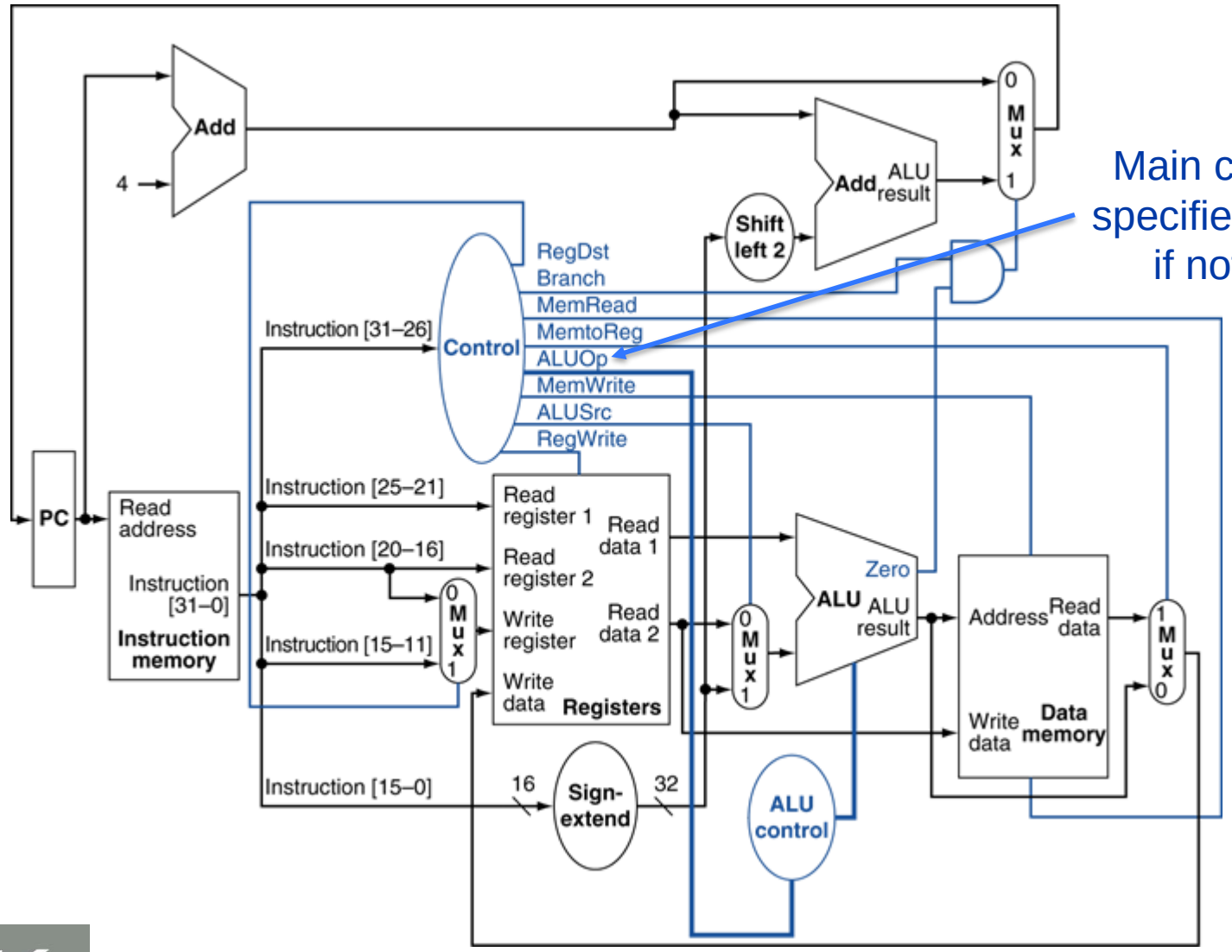
ALU Control Unit



ALU Control Unit

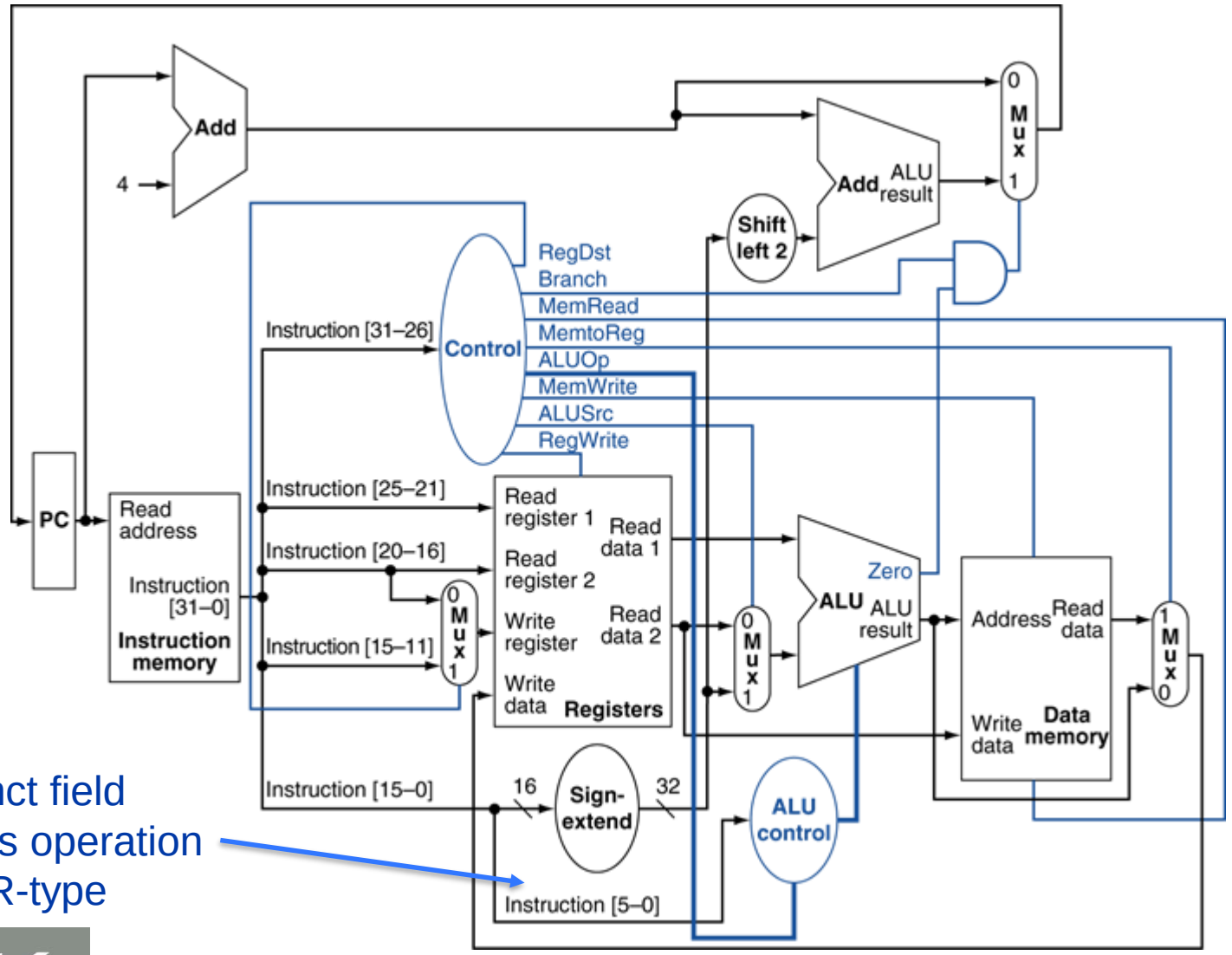


ALU Control Unit



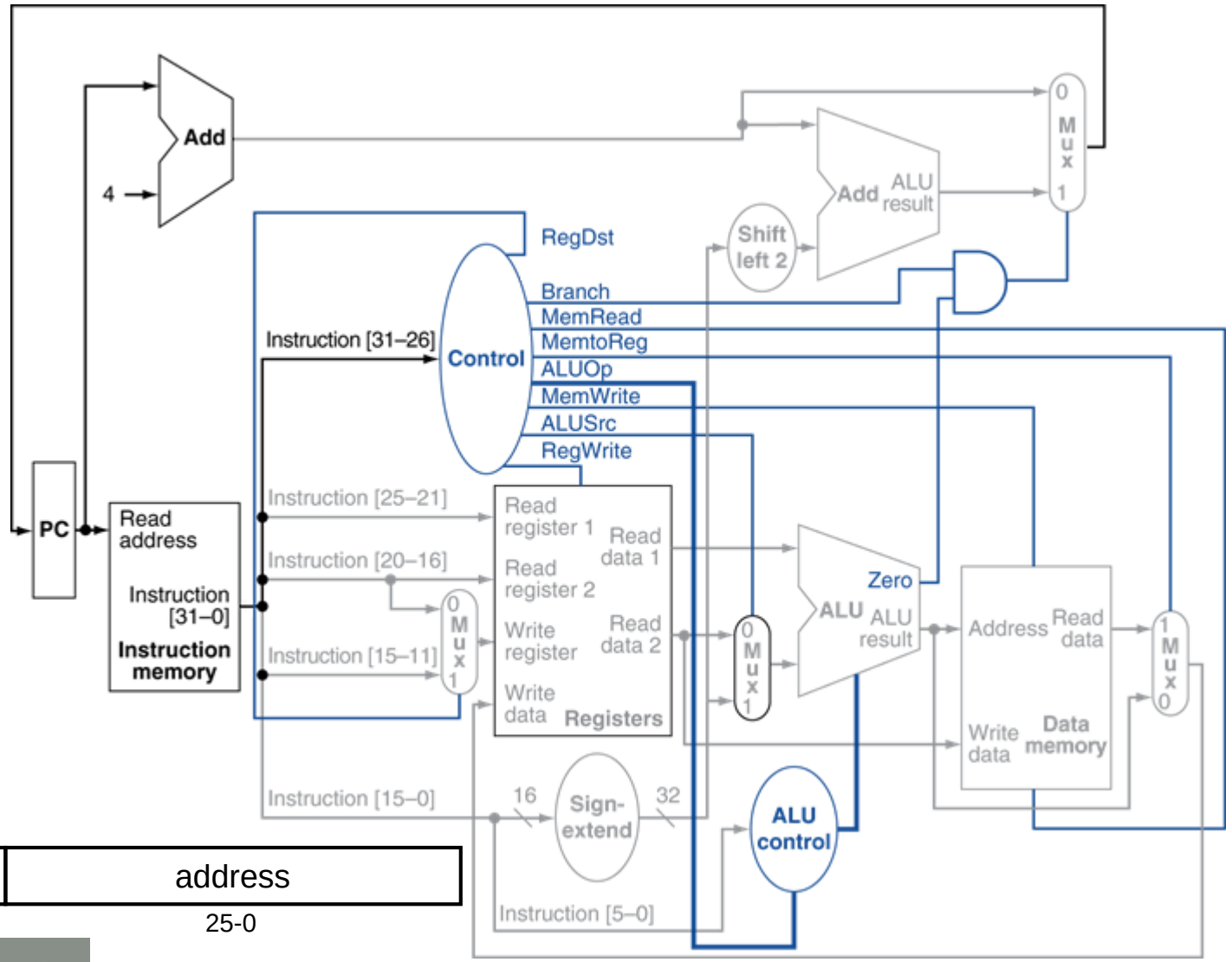
Main control unit specifies operation if not R-type

ALU Control Unit



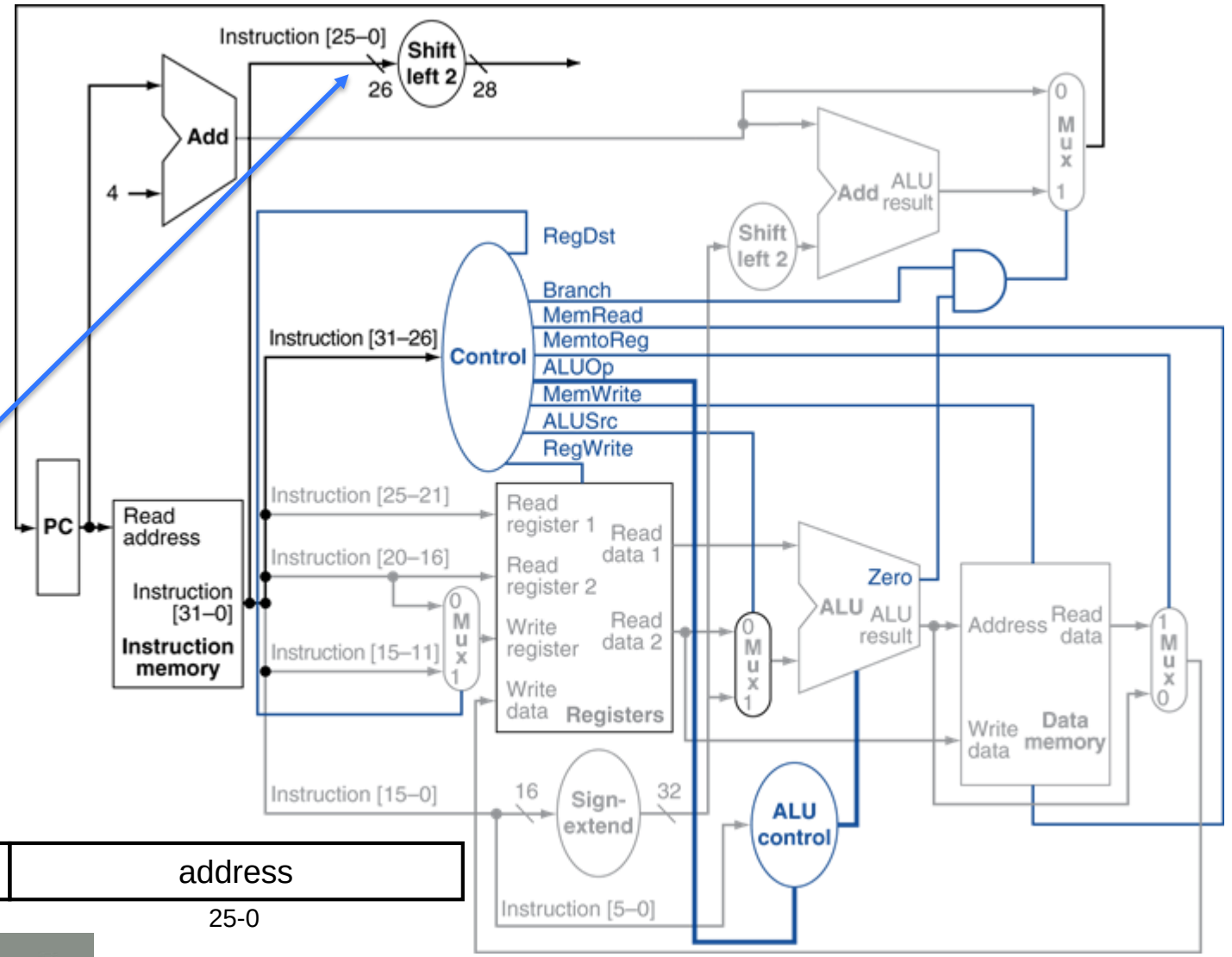
Funct field specifies operation if R-type

Jump Instructions



Jump address = PC+4 [31-28] : address × 4

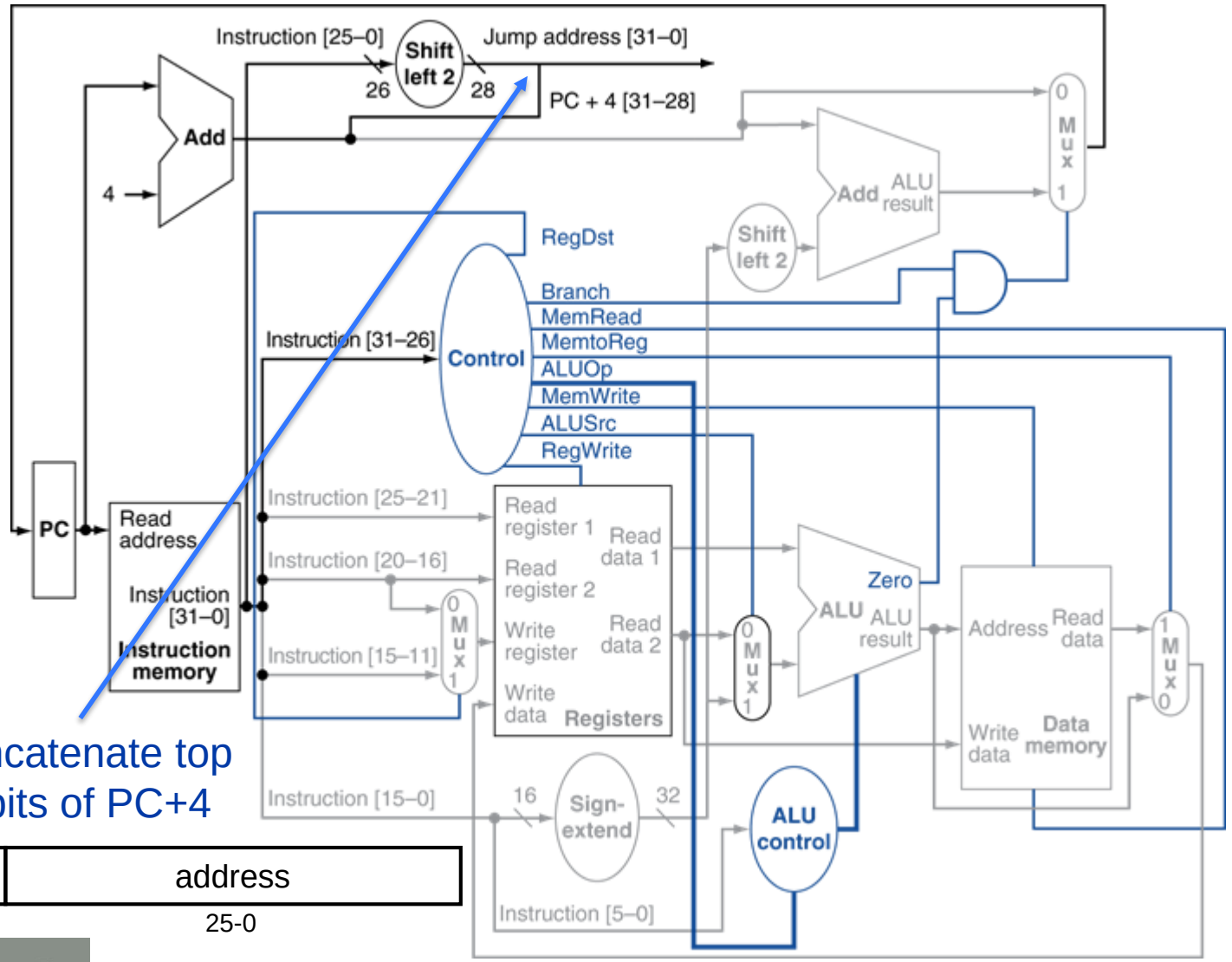
Jump Instructions



Multiply address field by 4

Jump address = PC+4 [31-28] : address × 4

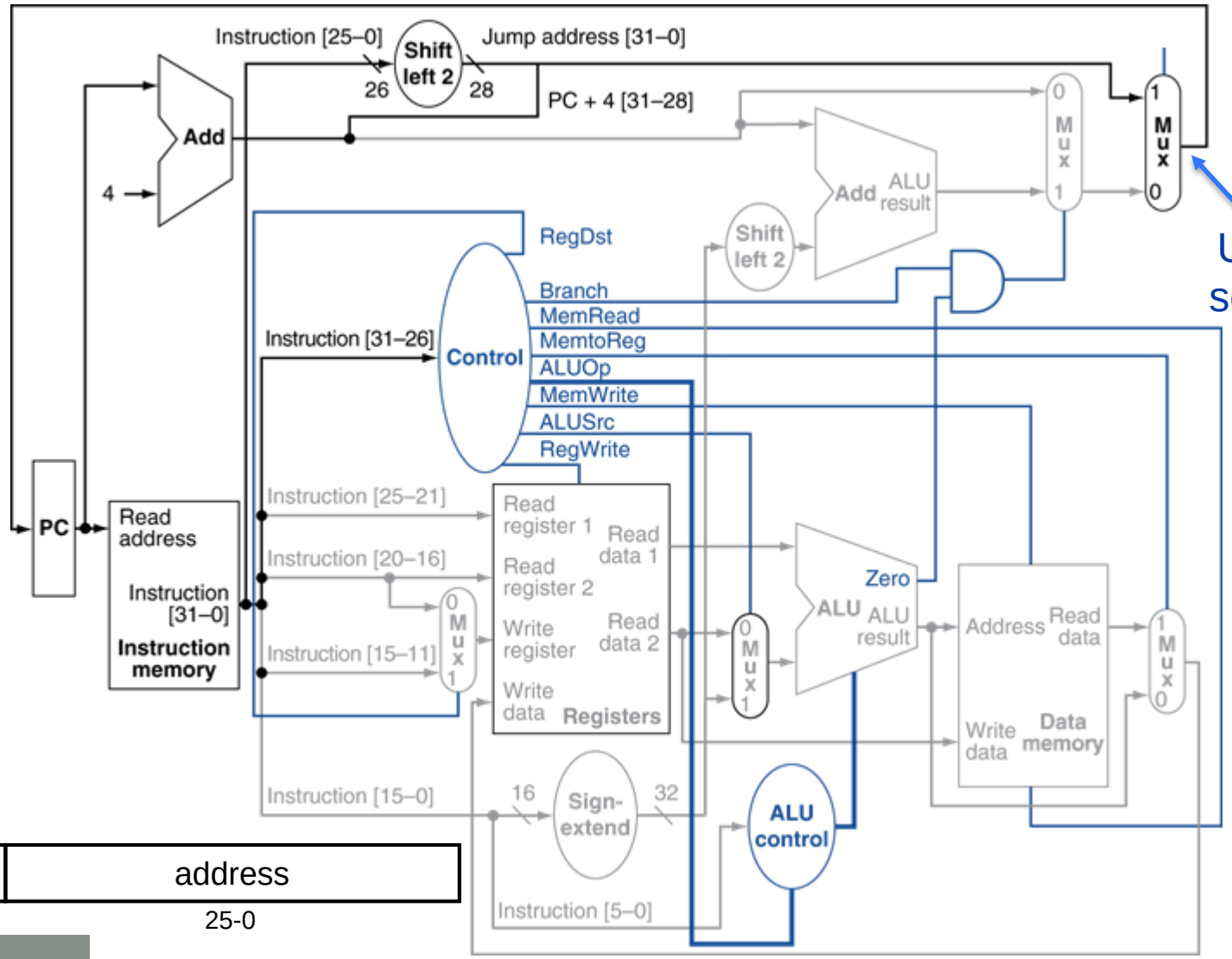
Jump Instructions



Concatenate top 4 bits of PC+4

Jump address = PC+4 [31-28] : address × 4

Jump Instructions

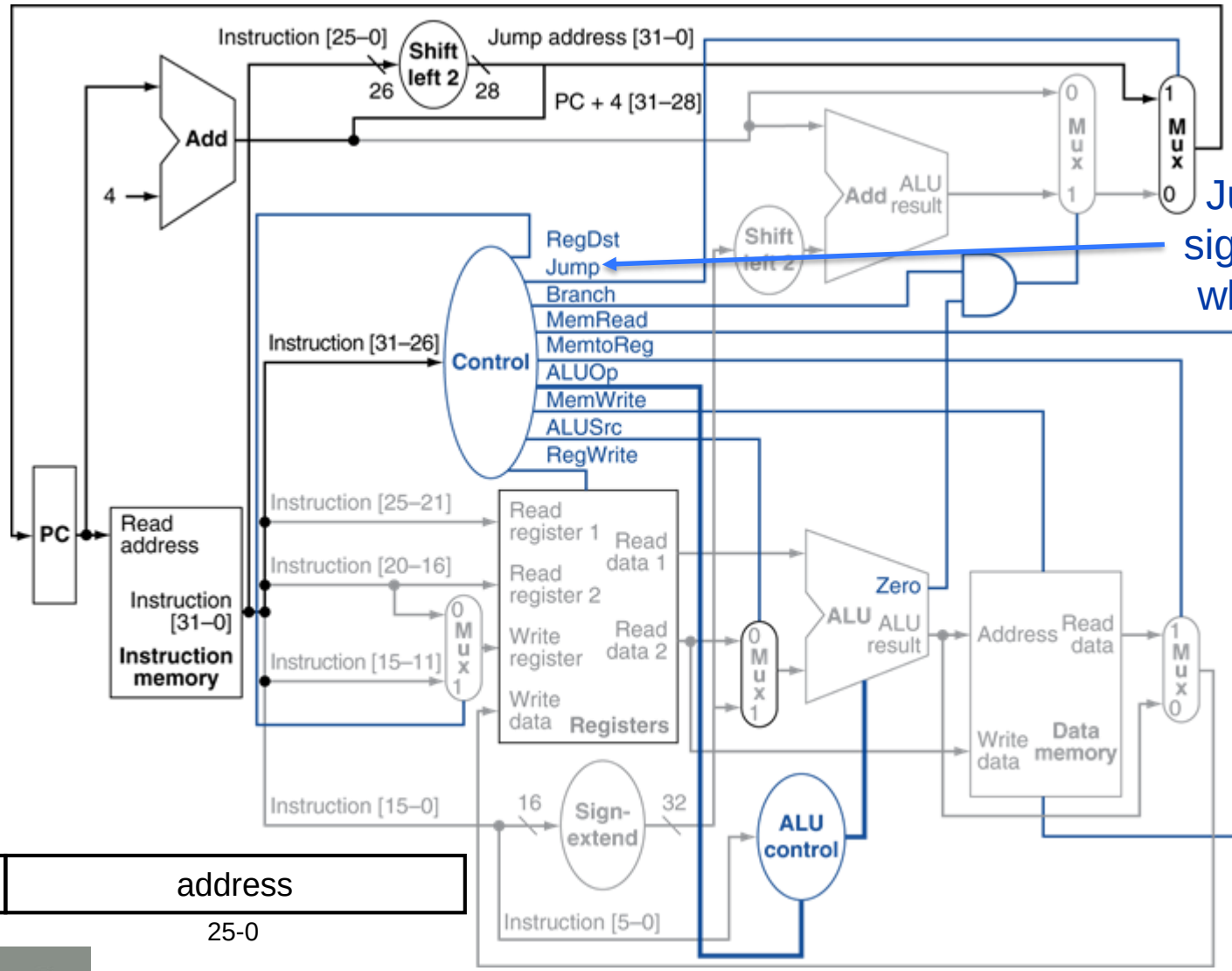


Use mux to select target address

Jump	op	address
	31-26	25-0

Jump address = PC+4 [31-28] : address × 4

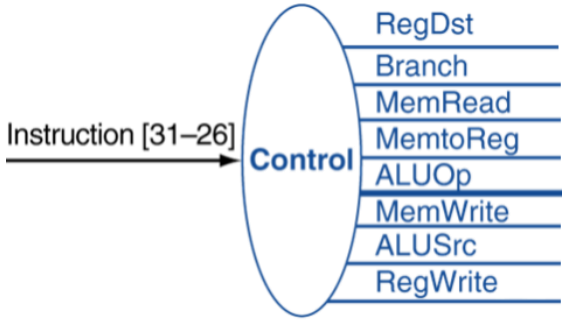
Jump Instructions



Jump control signal specifies what to select

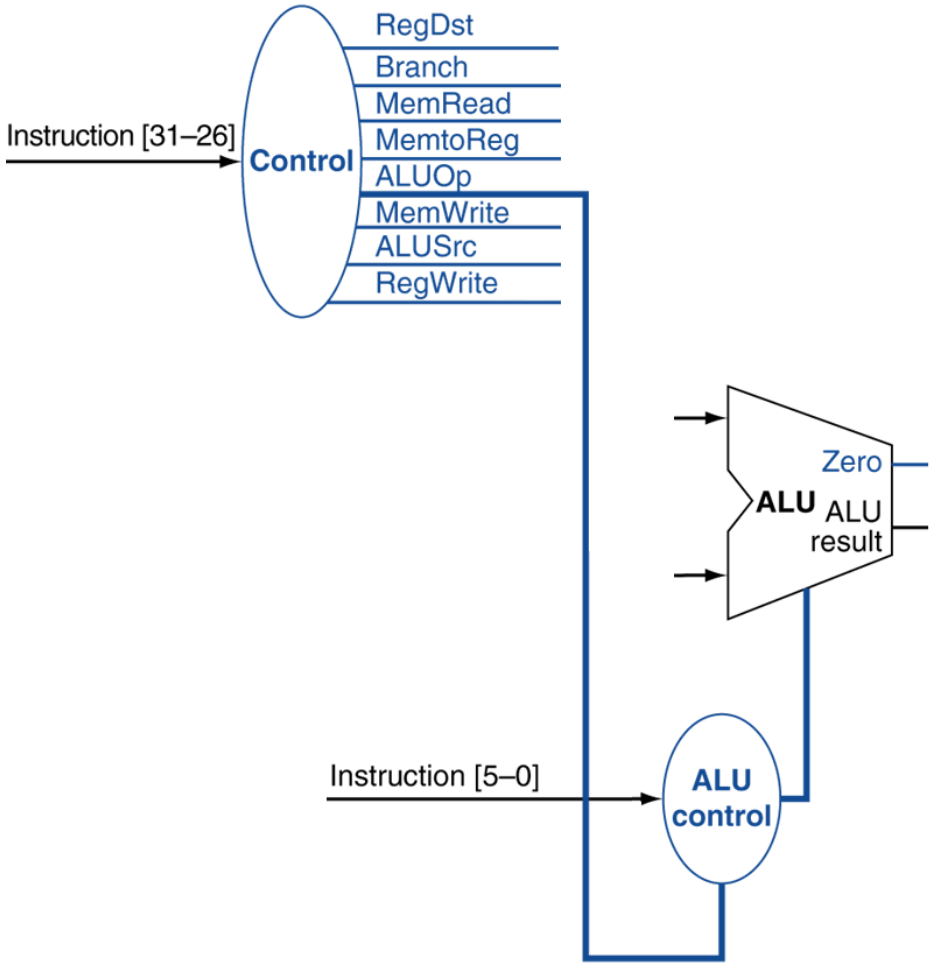
Jump address = PC+4 [31-28] : address × 4

Main Control Unit



Instr[31-26]	RegDst	RegWrite	ALUSrc	ALUOp	MemRead	MemWrite	MemtoReg	Branch	Jump
R-type (000000)	1	1	0	funct (10)	0	0	0	0	0
addi (001000)	0	1	1	add (00)	0	0	0	0	0
lw (100011)	0	1	1	add (00)	1	0	1	0	0
sw (101011)	d	0	1	add (00)	0	1	d	0	0
beq (000100)	d	0	0	sub (01)	0	0	d	1	0
j (000010)	d	0	d	d	0	0	d	d	1

ALU Control Unit



ALUOp	Instr[5-0]	ALU Control Line
funct (10)	add (100000)	add (0010)
	sub (100010)	sub (0110)
	slt (101010)	slt (0111)
add (00)	d	add (0010)
sub (01)	d	sub (0110)

Textbook Sections

- The content in these slides corresponds to:
  - Textbook:
    - *Computer Organization and Design, 5th Edition by David Patterson and John Hennessy, Morgan Kaufmann, 2014.*
  - Sections:
    - 4.4