The Processor: Datapath & Pipelining

Dr. Ahmed Zahran

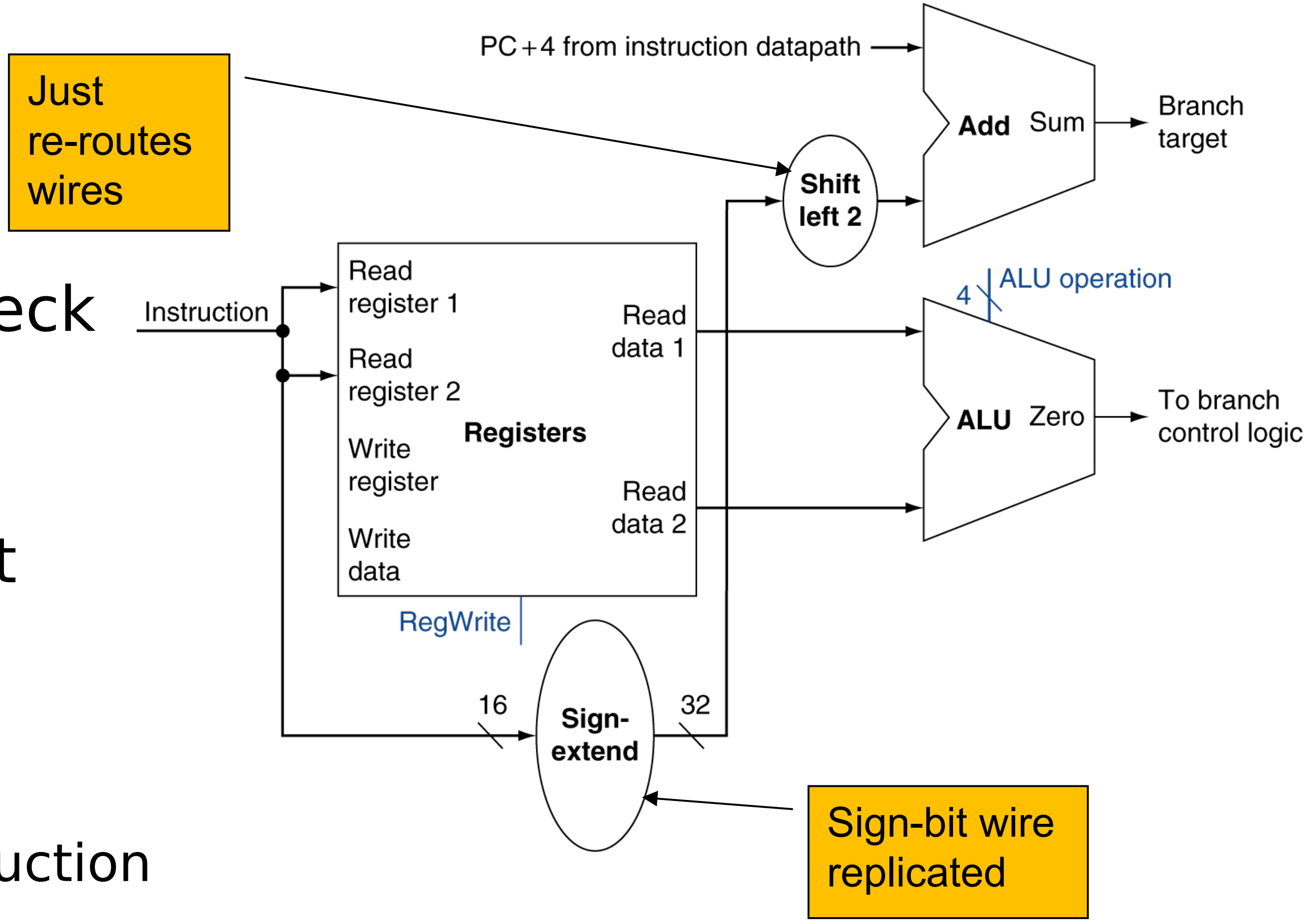
WGB 182

a.zahran@cs.ucc.ie

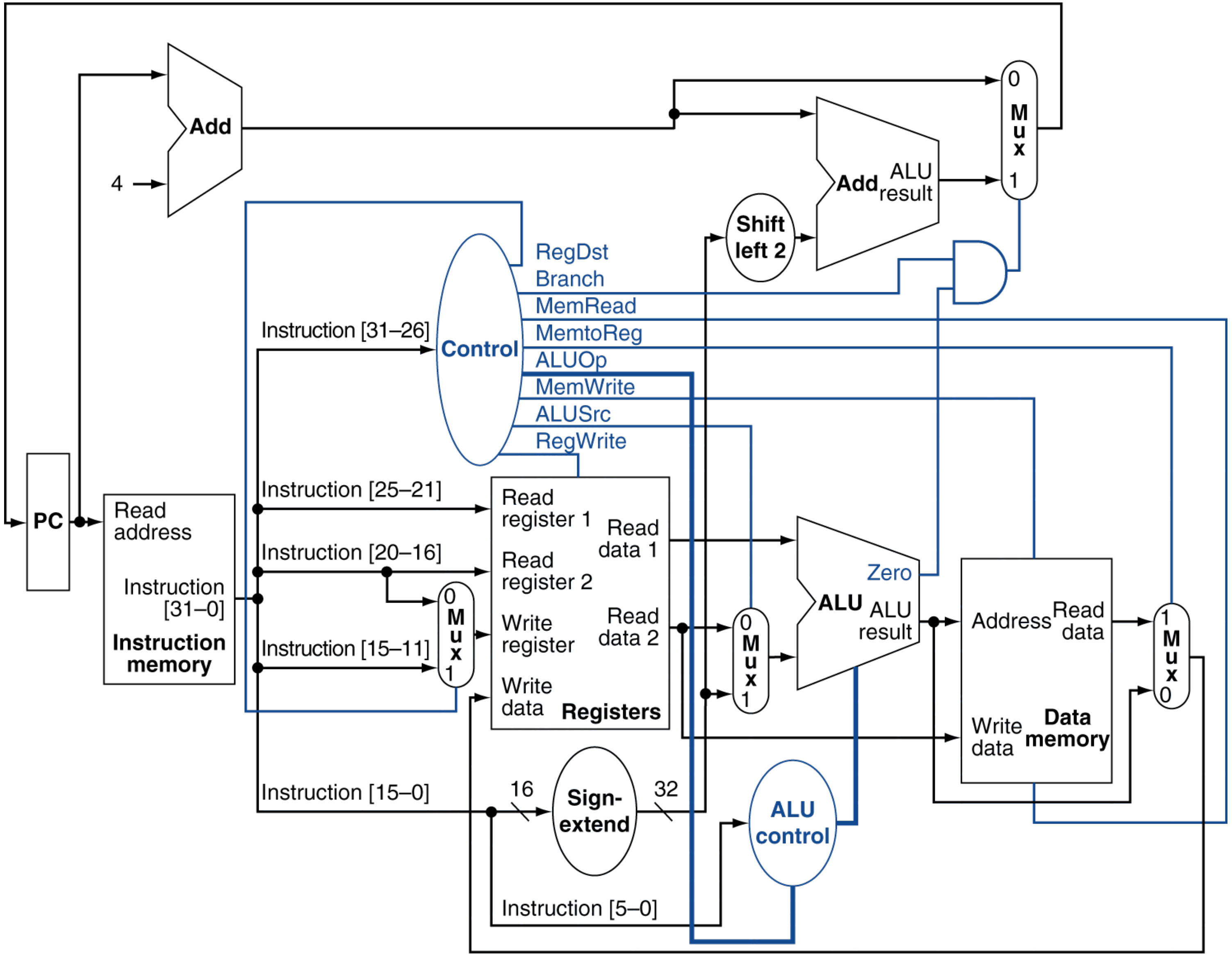
Branch Instructions

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

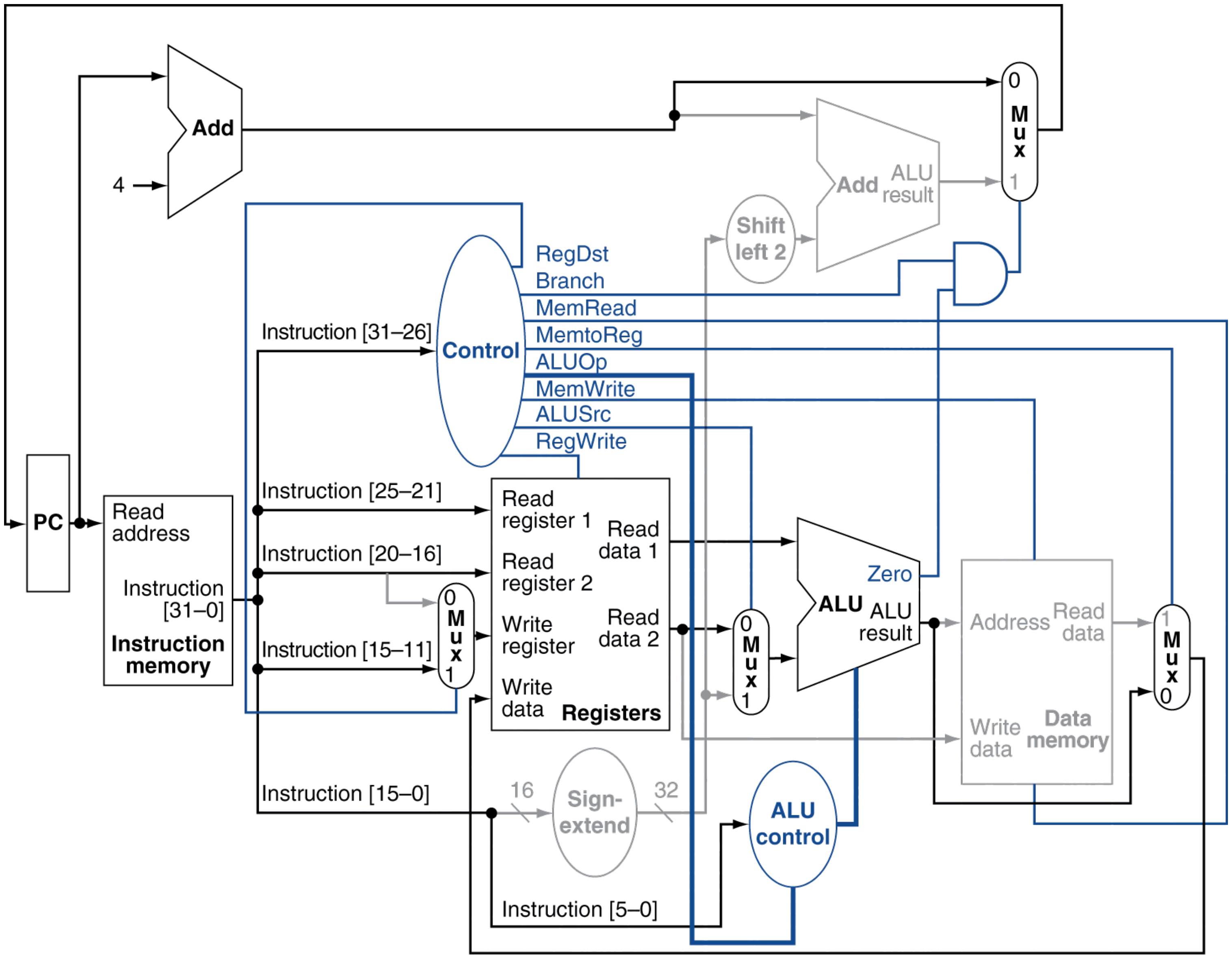
- Read register operands
- Compare operands
  - Use ALU, subtract and check Zero output
- Calculate target address
  - Sign-extend displacement
  - Shift left 2 places (word displacement)
  - Add to PC + 4
    - Already calculated by instruction fetch



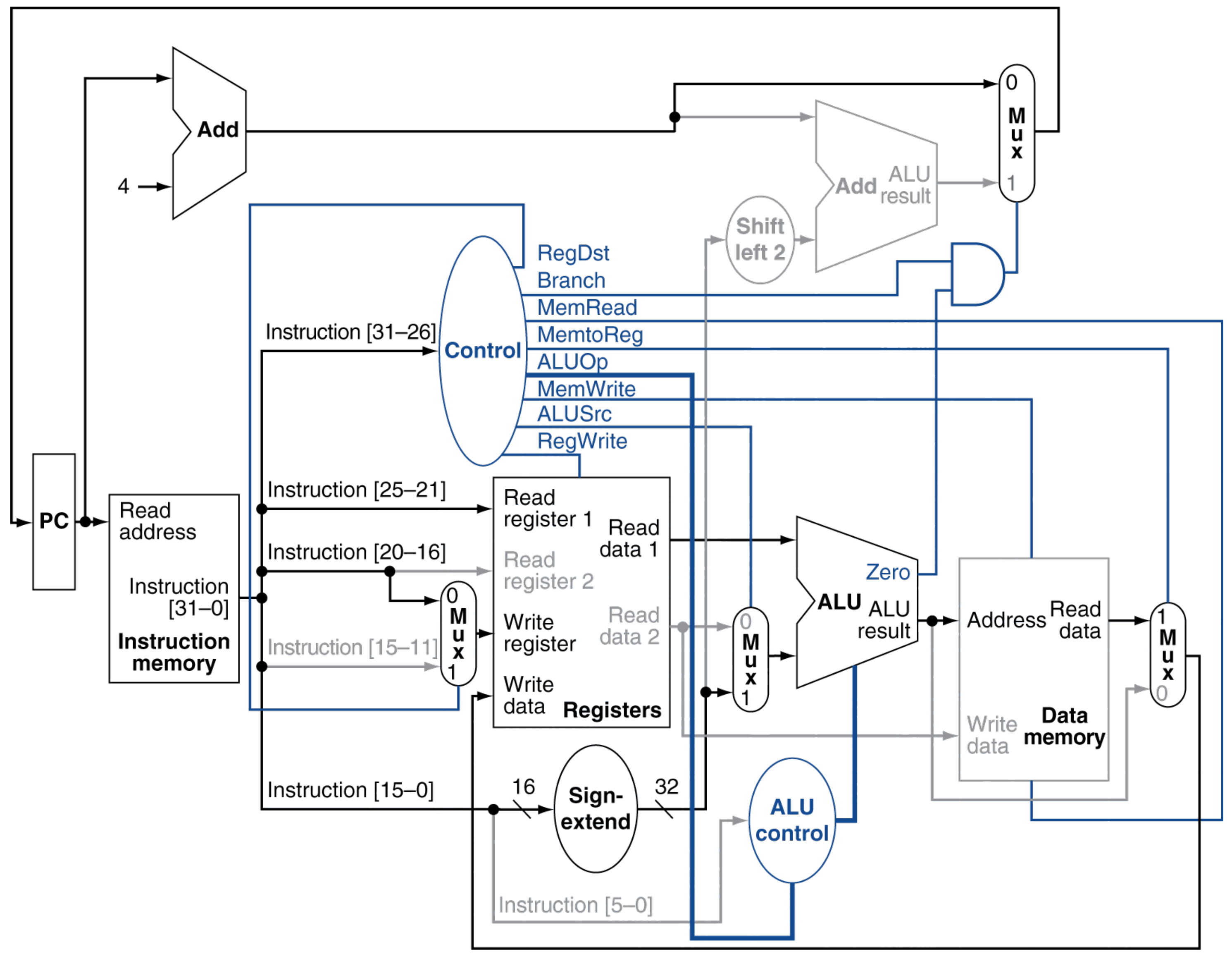
Integrated Datapath With Control



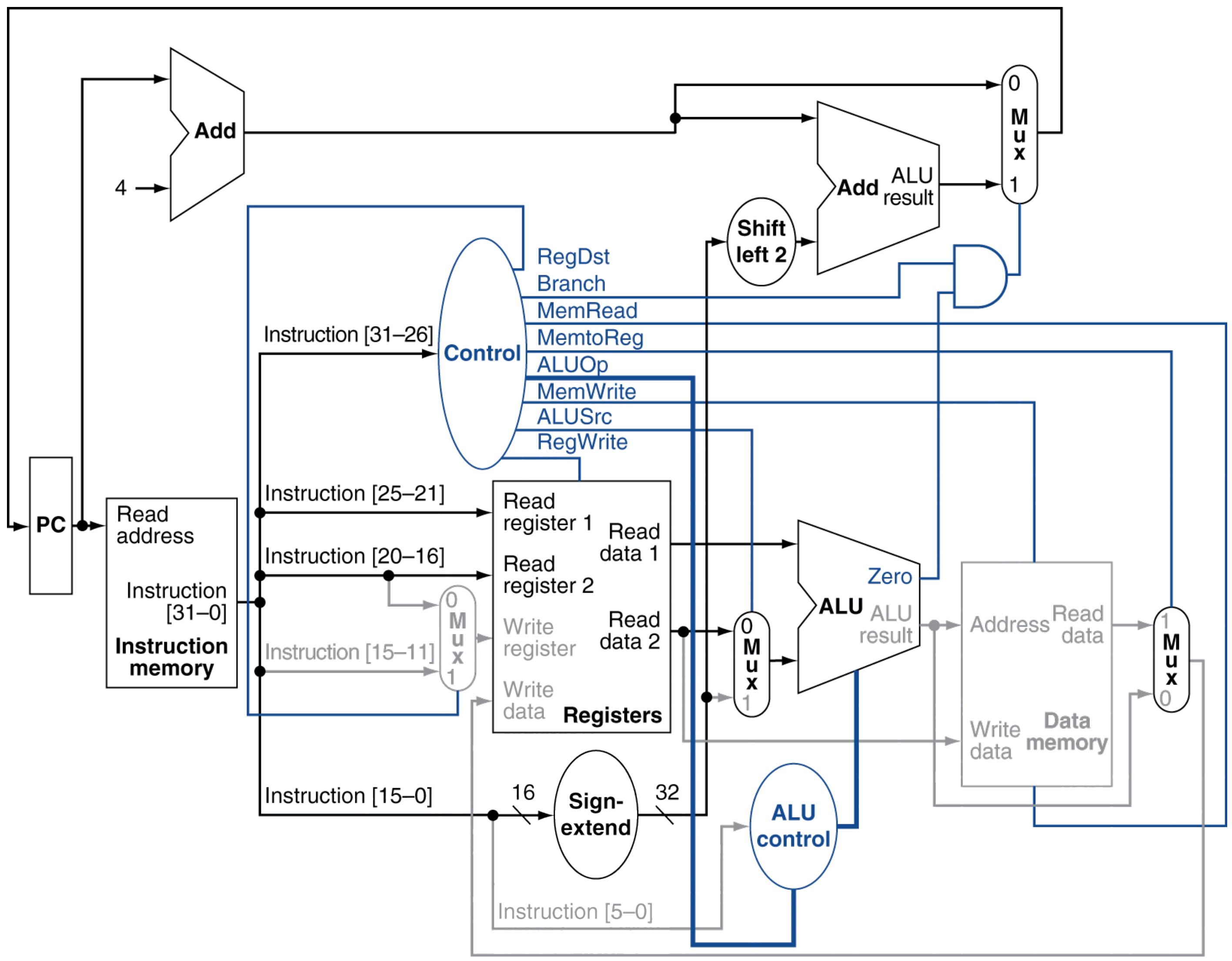
R-Type Instruction



Load Instruction



Branch-on-Equal Instruction

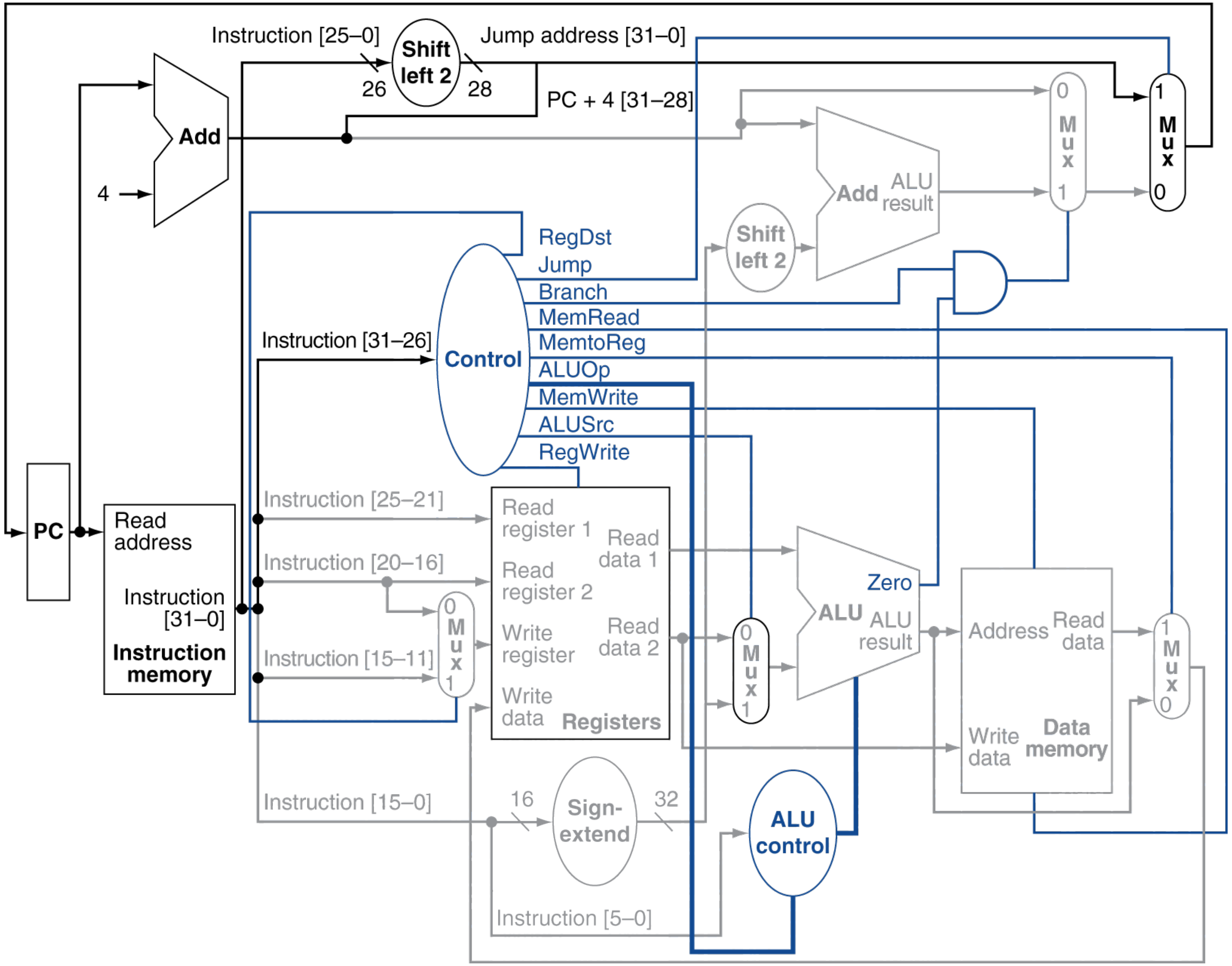


Implementing Jumps

Jump	2	address
	31:26	25:0

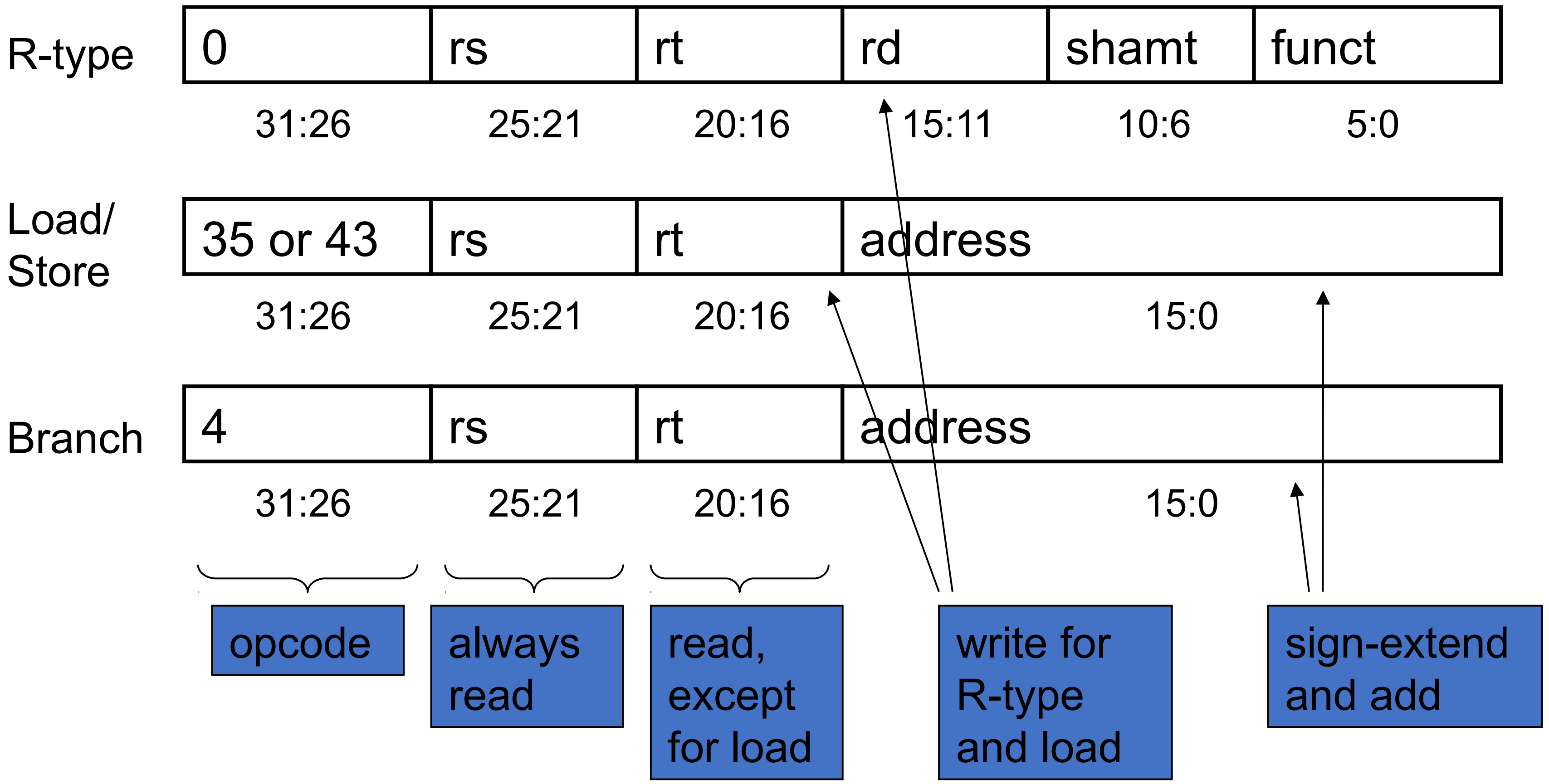
- Jump uses word address
- Update PC with concatenation of
  - Top 4 bits of old PC
  - 26-bit jump address
  - 00
- Need an extra control signal decoded from opcode
  - Called a jump control
  - Asserted (set) only when opcode has the value of 2

Datapath With Jumps Added



The Main Control Unit

- Control signals derived from instruction



ALU Control

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

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

ALU Control

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

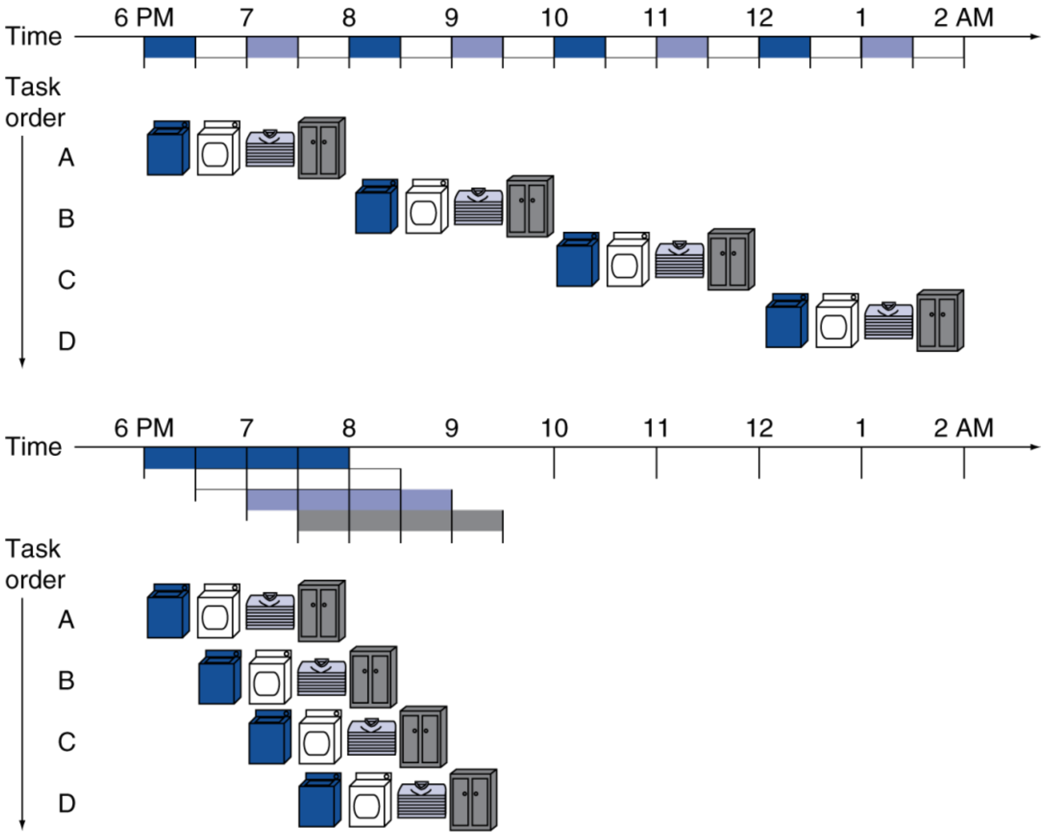
opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

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
- A solution to improve performance is by pipelining

Pipelining Analogy

- Pipelined laundry: overlapping execution
  - Parallelism improves performance



- Four loads:
  - Sequential execution
- Non-stop:
  - Speedup achieved with pipelining

Reading

- Sections 4.4, 4.5