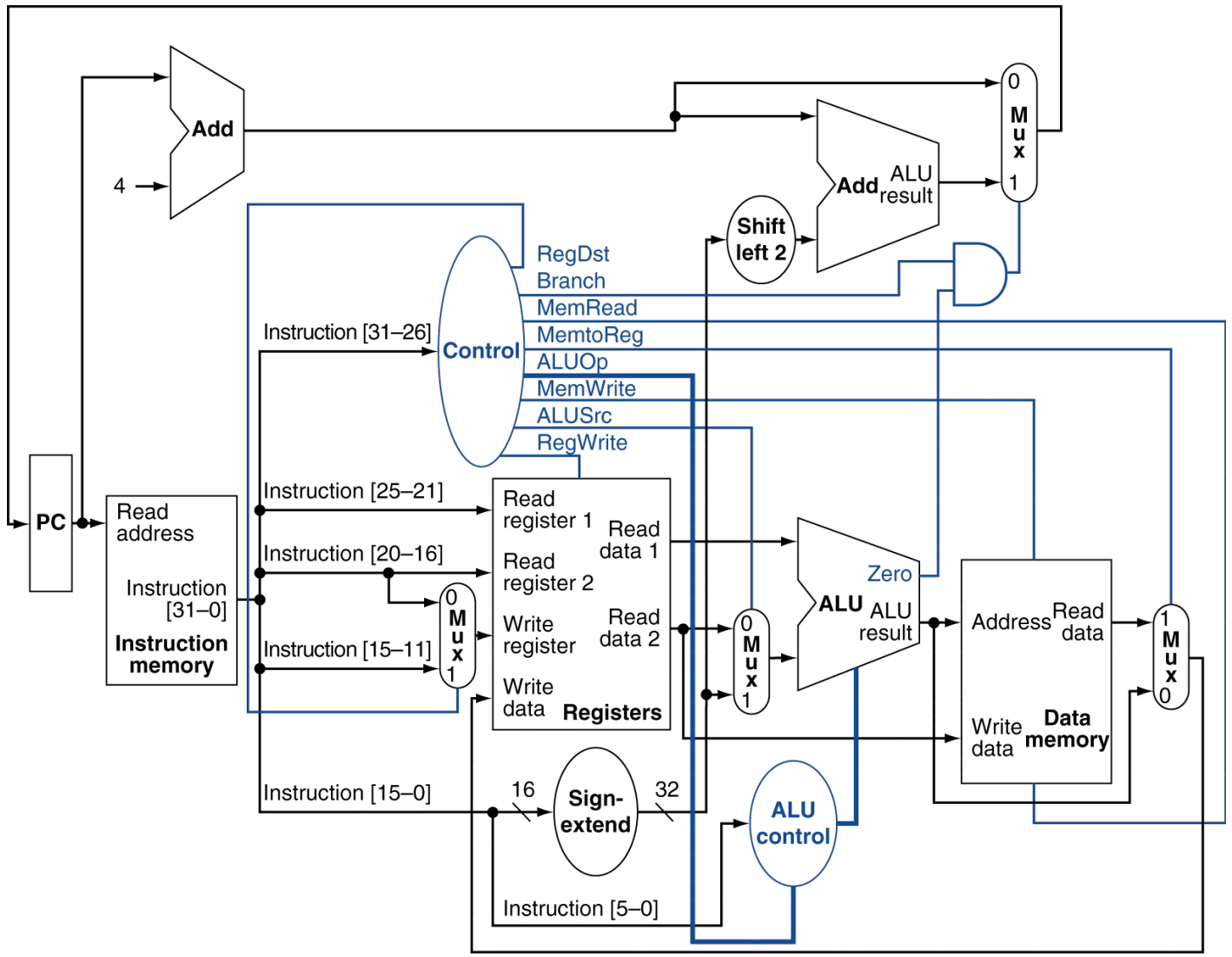
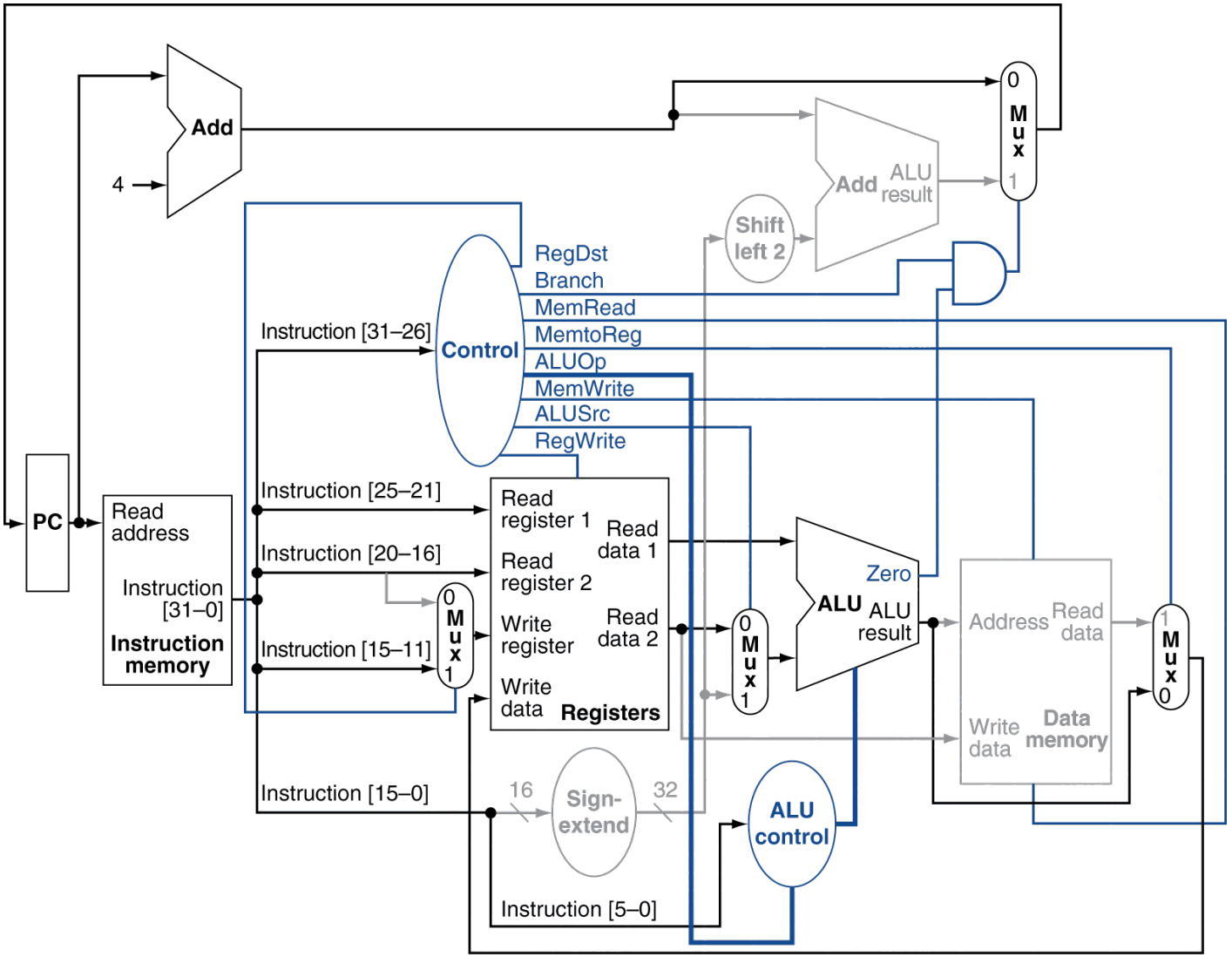
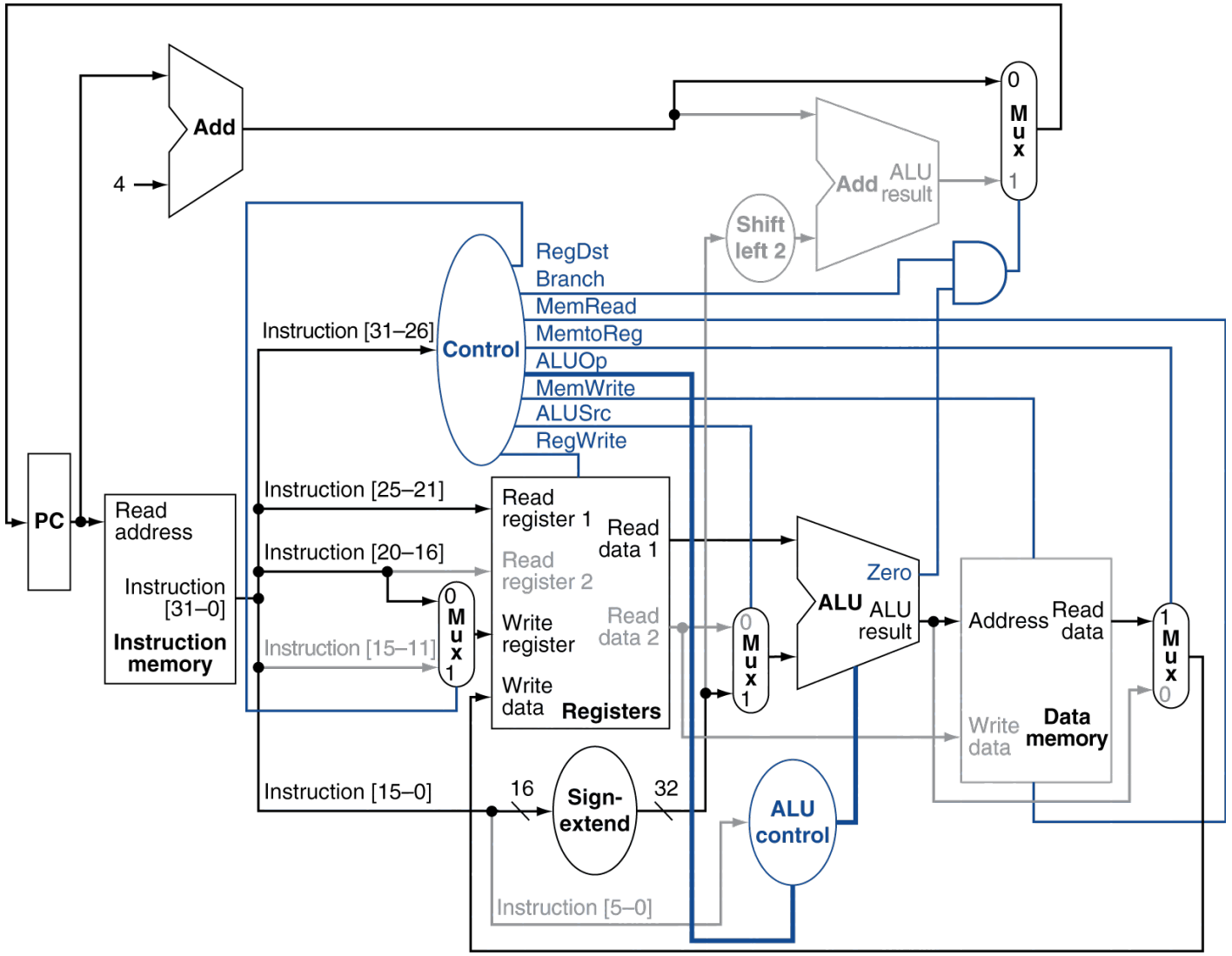
Datapath with Control



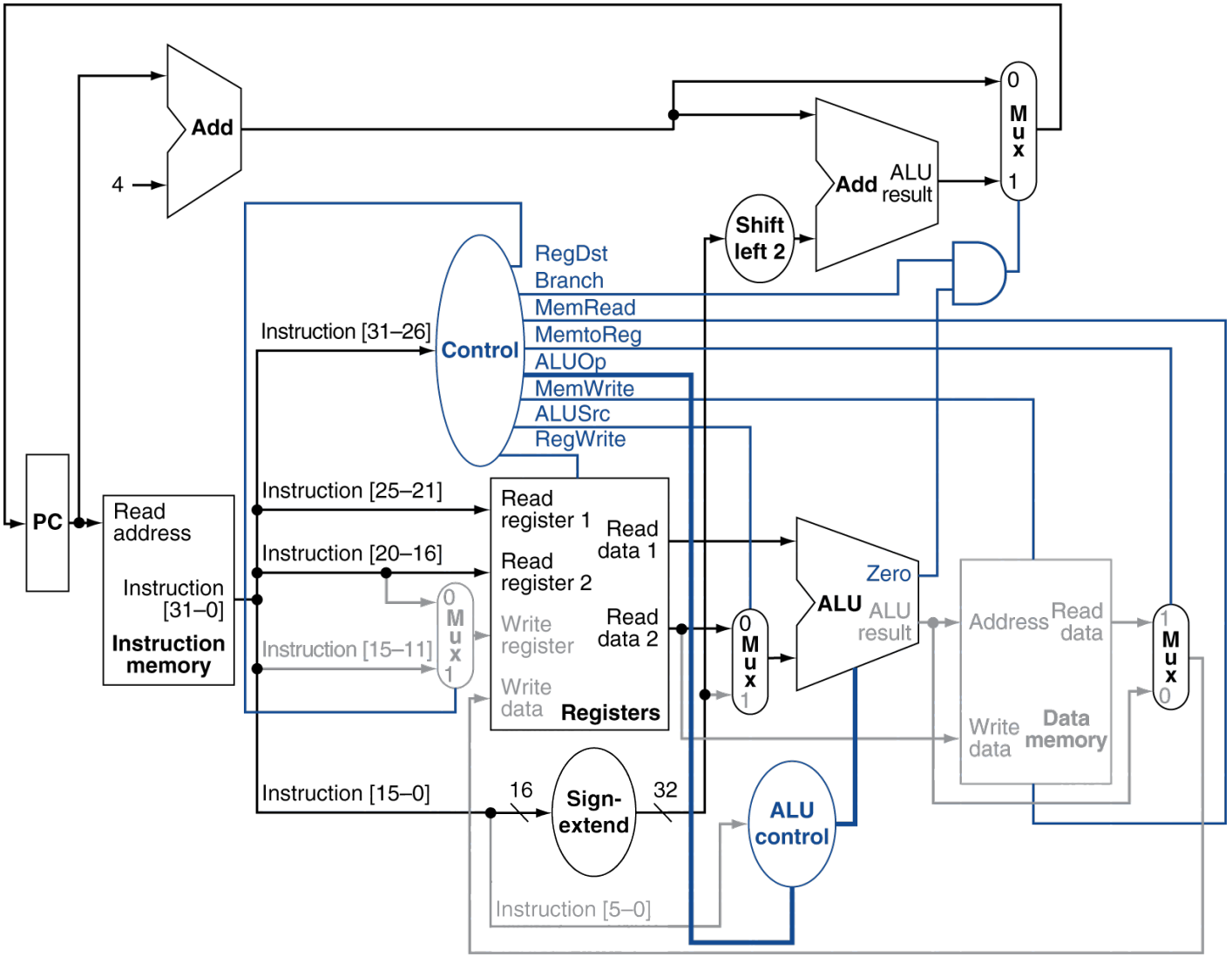
R-type Control



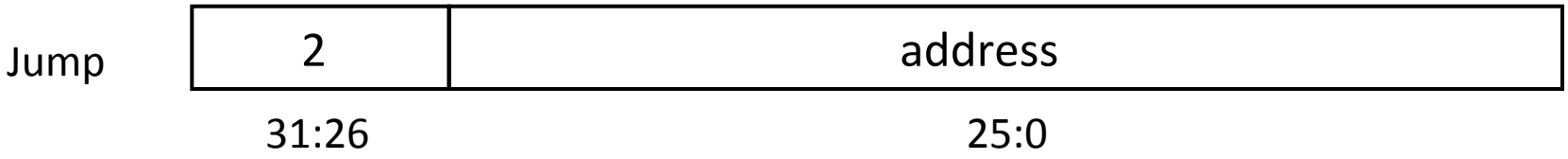
Load Instruction



Beq Instruction

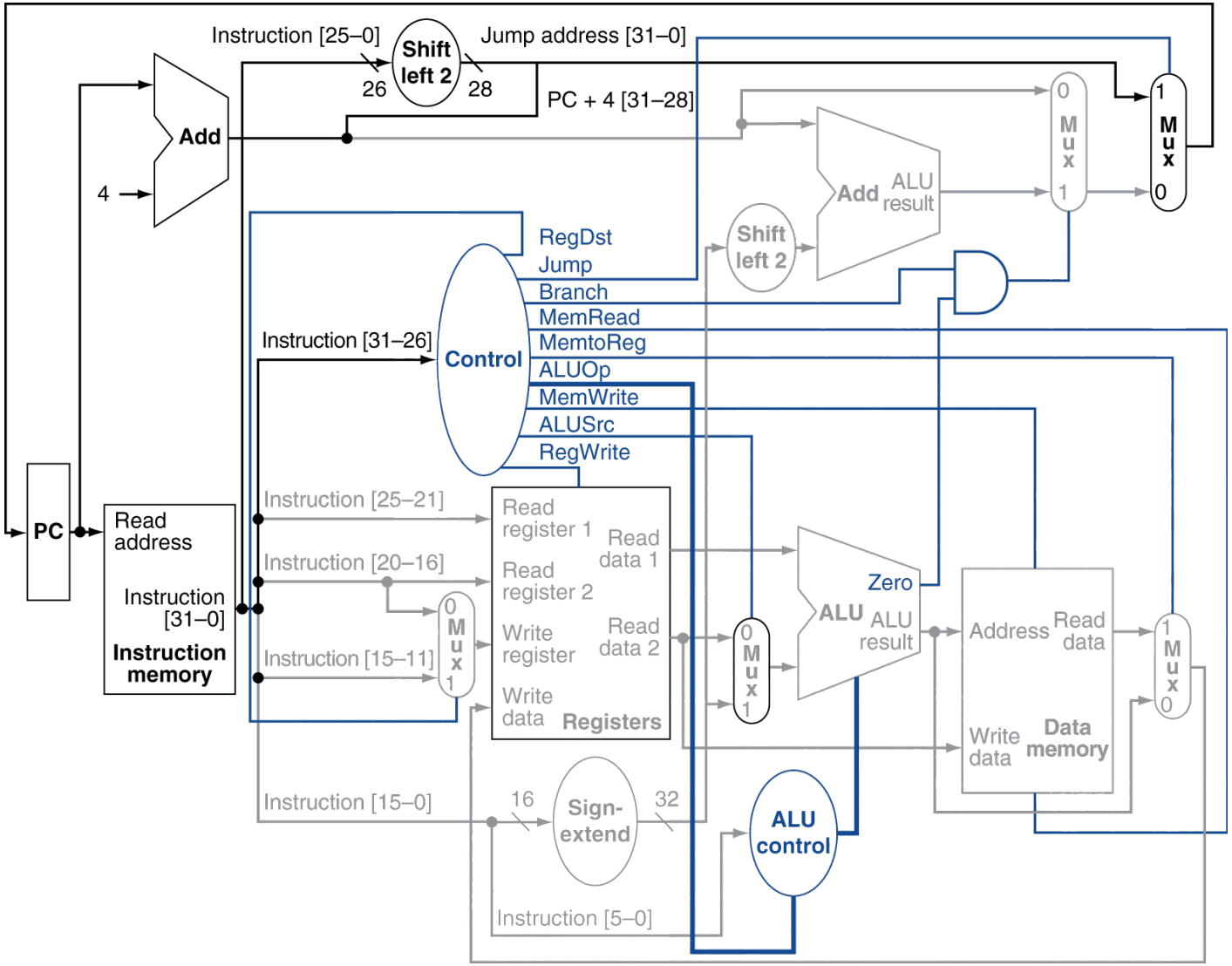


Jump Instruction



- Jump uses word address (instruction boundary)
- Sets PC = (PC & 0xF0000000) + 4*(26-bit jump address)
- Needs an extra control signal decoded from opcode

Datapath with Jump Added



Mapping Control to Gates

- Use of instruction subset simplifies control
- Only 9 control signals have to be generated

Signal name
RegDst
ALUSrc
MemtoReg
RegWrite
MemRead
MemWrite
Branch
ALUOp1
ALUOp0

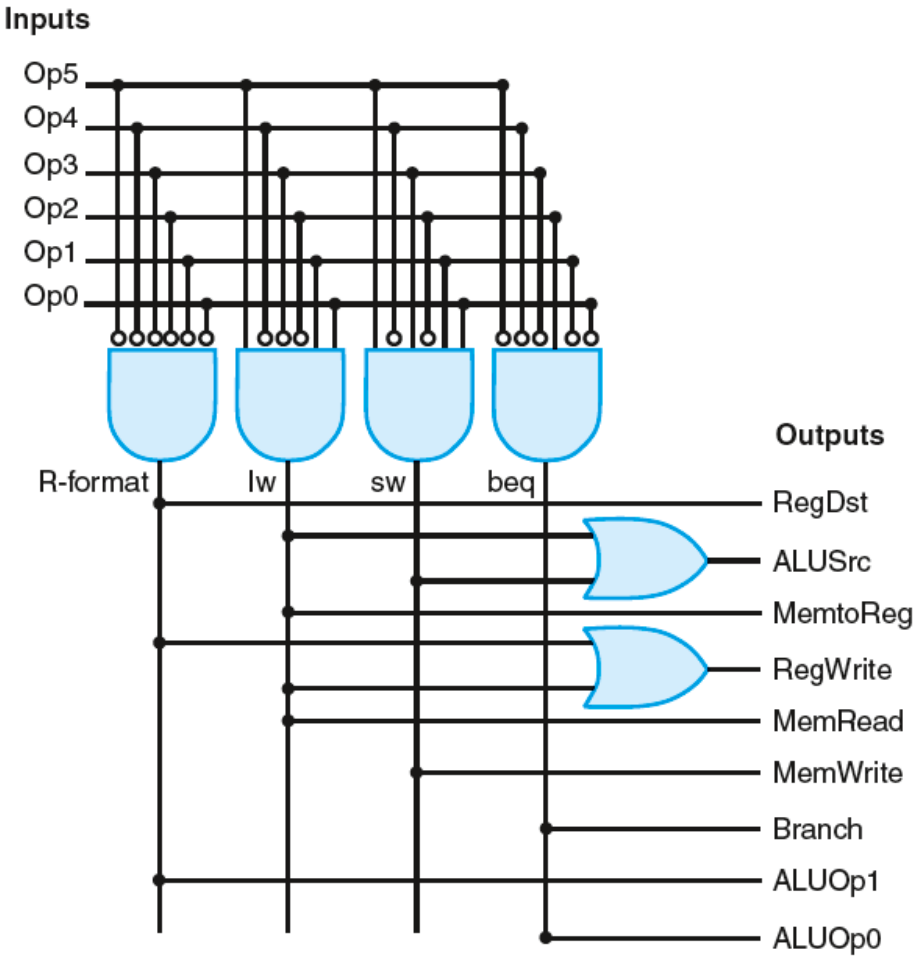
Mapping Control to Gates

- A truth table defines the outputs as function of opcode

Control	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Mapping Control to Gates

- Simple logic circuit derived from truth table



Performance Issues

- With this option, instruction executes in a single clock cycle
- Longest instruction 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