EEE304 – Digital Design with HDL (II)

Lecture 7

Dr. Ming Xu

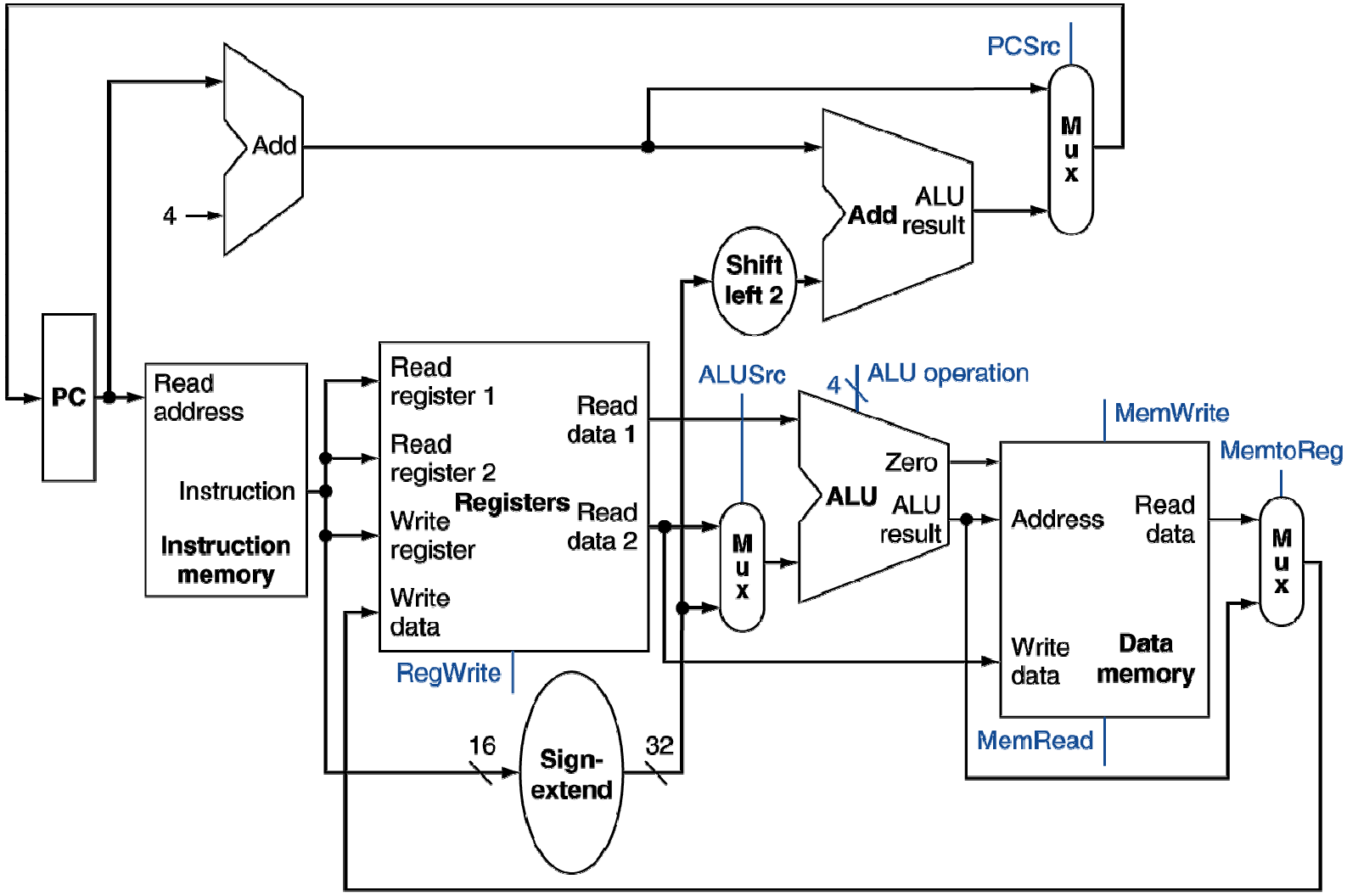
Dept of Electrical & Electronic Engineering

XJTLU

In This Session

- Adding the Control to Processors

Full Datapath



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

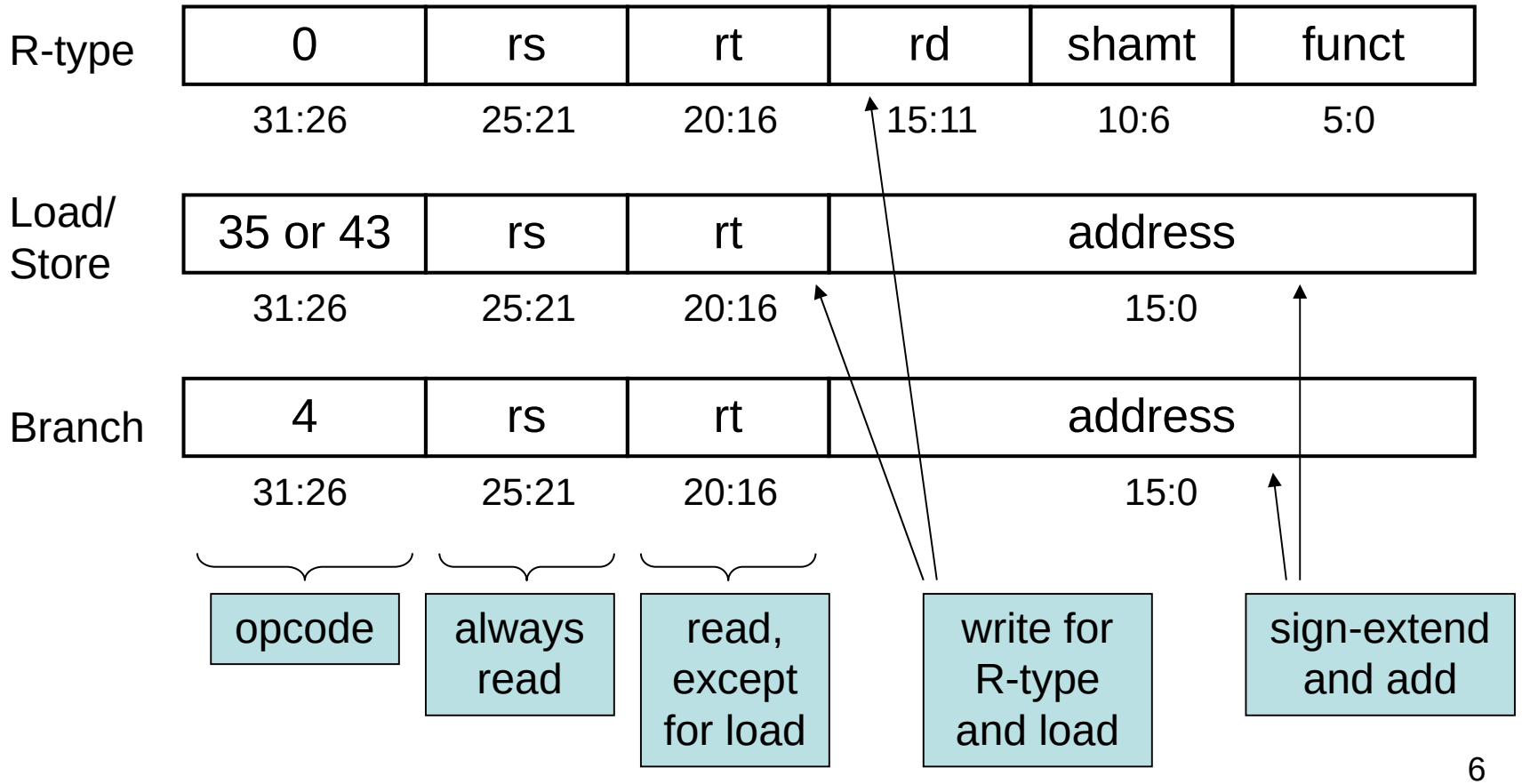
ALU Control

- Assume 2-bit ALUOp derived from opcode
  - ALU control bits depends on the ALUOp control bits and the function codes for the R-type instructions.

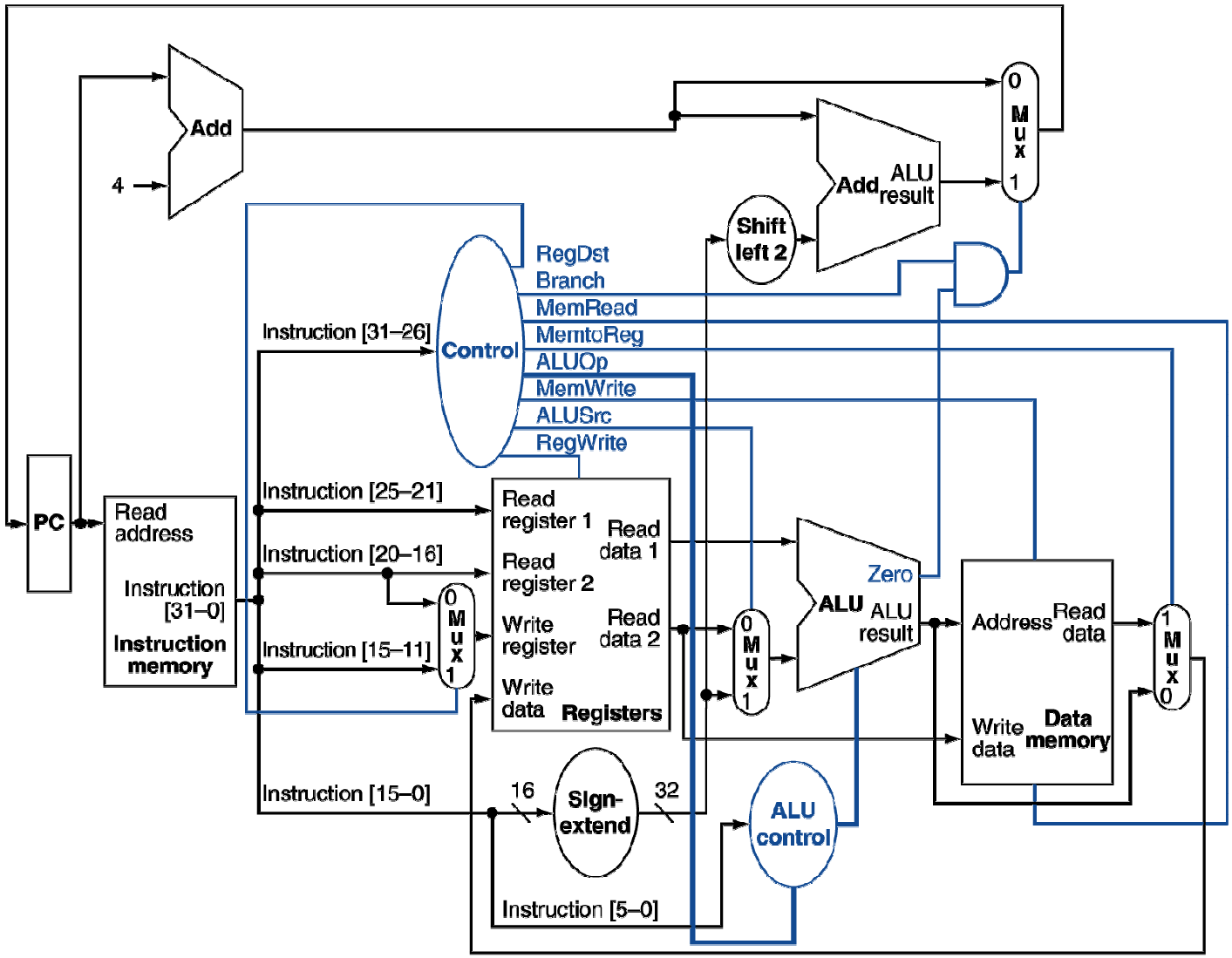
opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than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

The Main Control Unit

- Control signals derived from instruction



Datapath With Control



Effect of the Control Signals

Signal Name	Effect when deasserted	Effect when asserted
RegDst	The register destination number for the Write register comes from the **rt field** (bits 20:16)	The register destination number for the Write register comes from the **rd field** (bits 15:11)
RegWrite	None	The register on the Write register input is written with the value on the Write data input.
ALUSrc	The second ALU operand comes from the second register file output (Read data 2).	The second ALU operand is the sign extended, lower 16-bits of the instruction.
PCSrc	The PC is replaced by the output of the adder that computes the value of PC+4.	The PC is replaced by the output of the adder that computes the branch target.
MemRead	None	Data memory contents designated by the address input are put on the Read data output.
MemWrite	None	Data memory contents designated by the address input are replaced by the value on the write data input.
MemtoReg	The value fed to the register Write data input comes from the ALU.	The value fed to the register Write data input comes from the data memory.

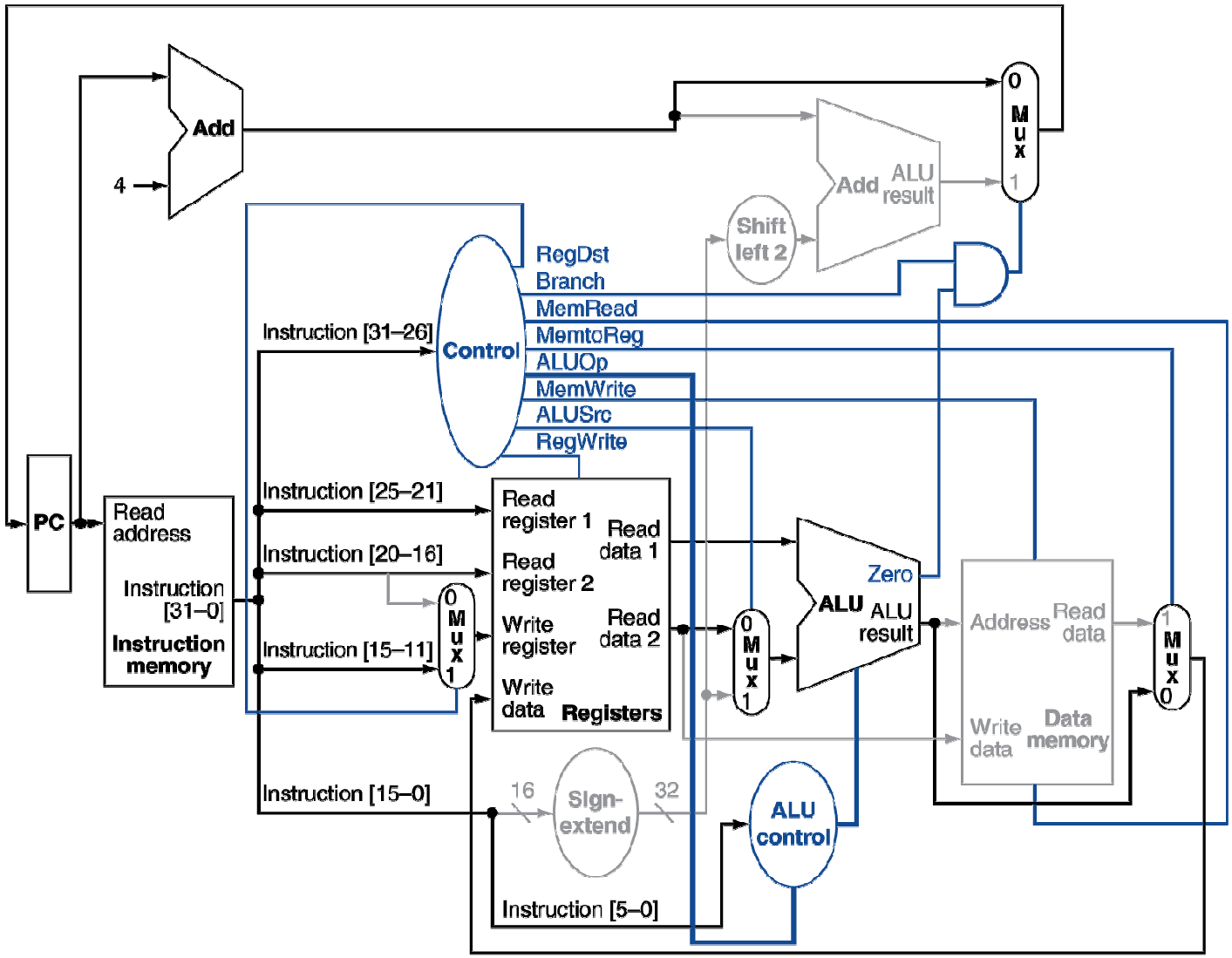
Effect of the Control Signals

The setting of the control lines is completely determined by the opcode field of the instructions.

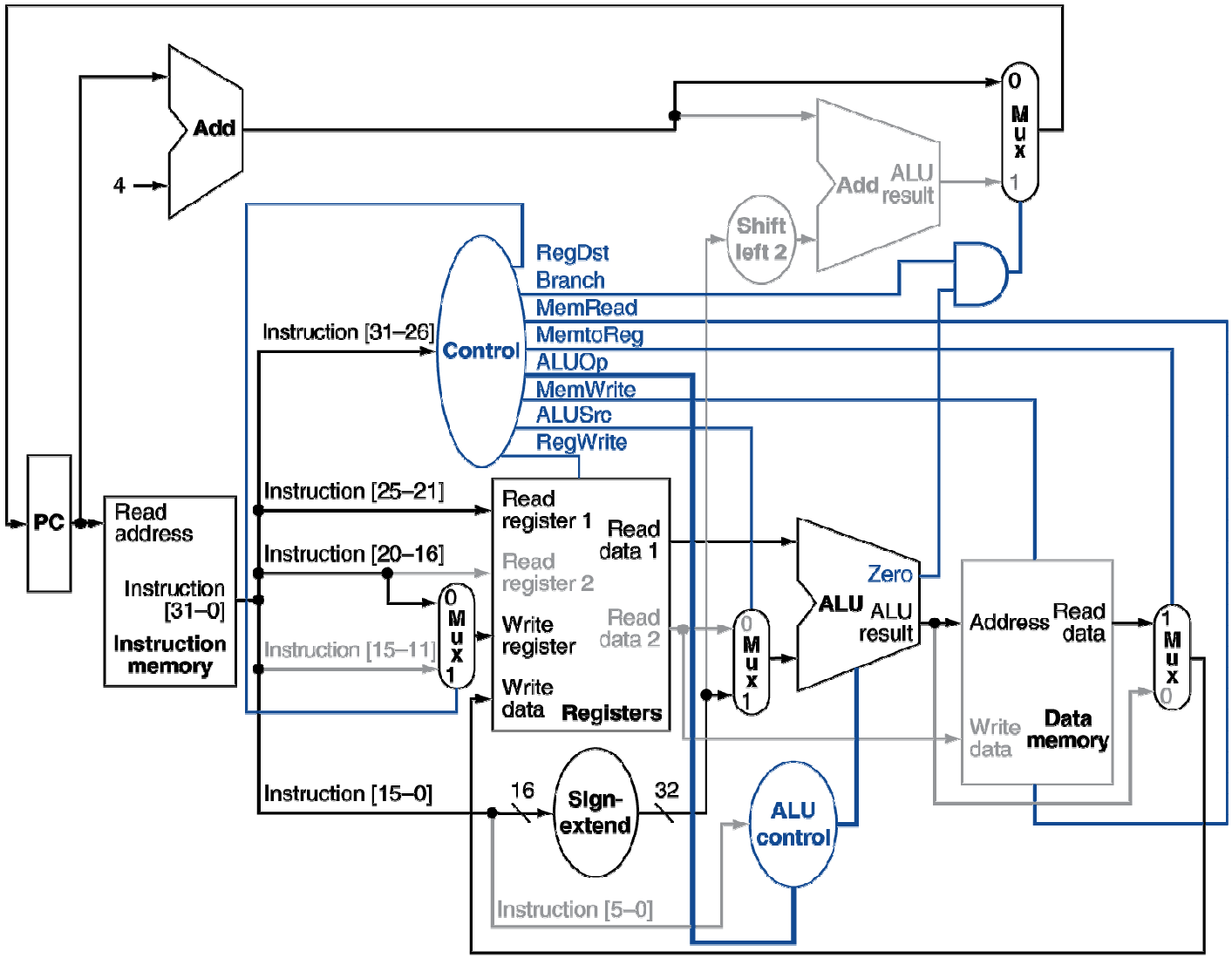
Instruction	Reg Dest	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUop1	ALUOp0
R-format 000000	1	0	0	1	0	0	0	1	0
Lw 100011	0	1	1	1	1	0	0	0	0
Sw 101011	X	1	X	0	0	1	0	0	0
Beq 000100	X	0	X	0	0	0	1	0	1

This function can be implemented directly in gates.

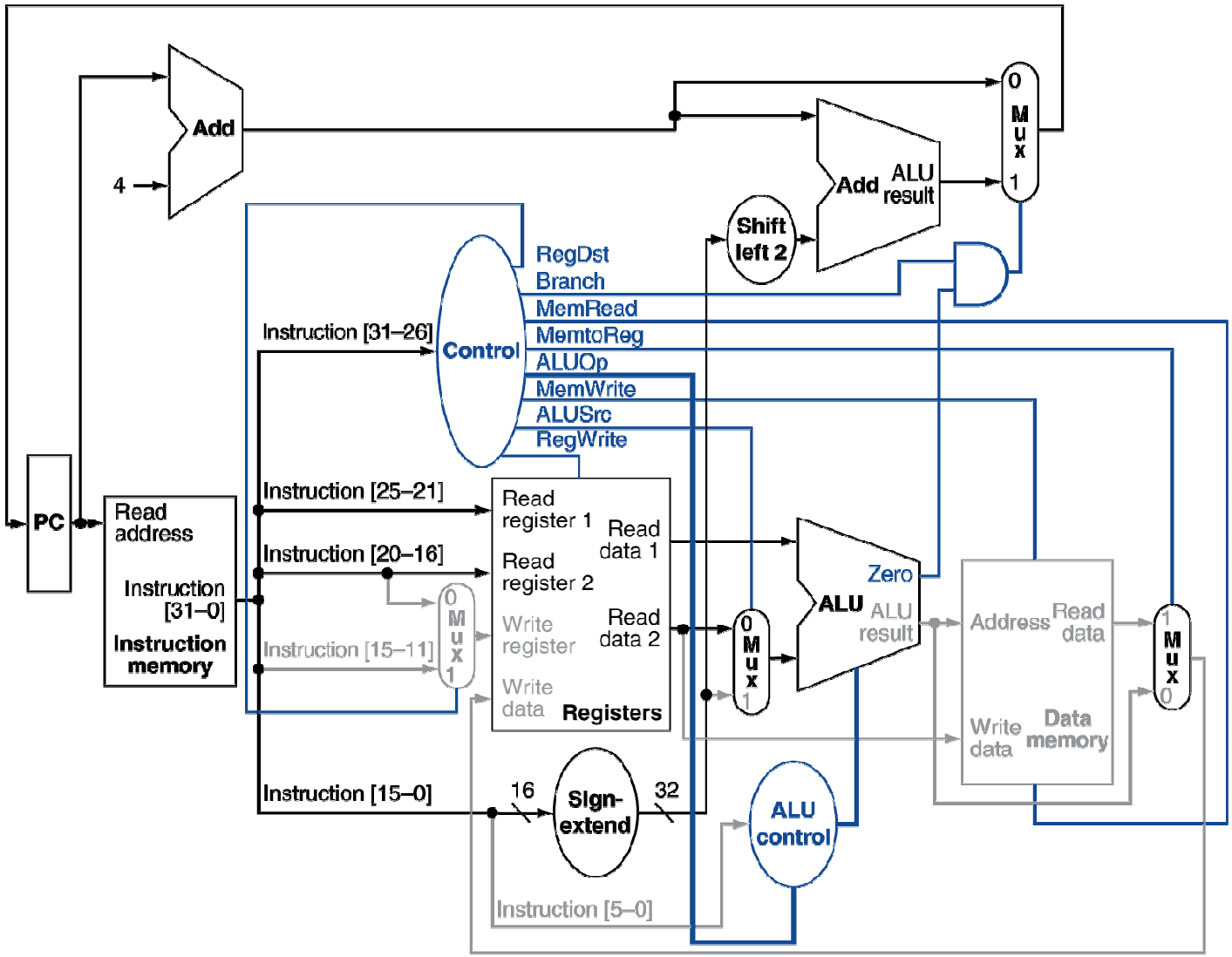
R-Type Instruction



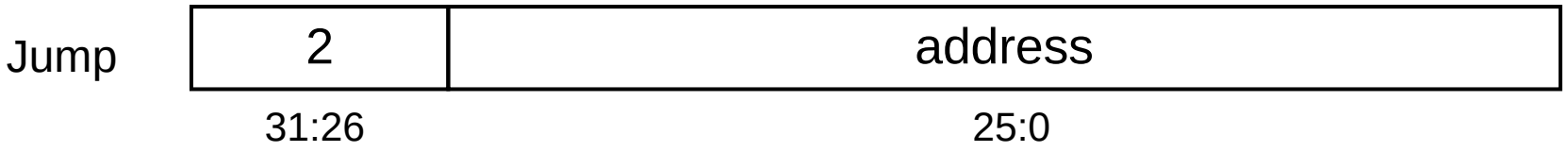
Load Instruction



Branch-on-Equal Instruction

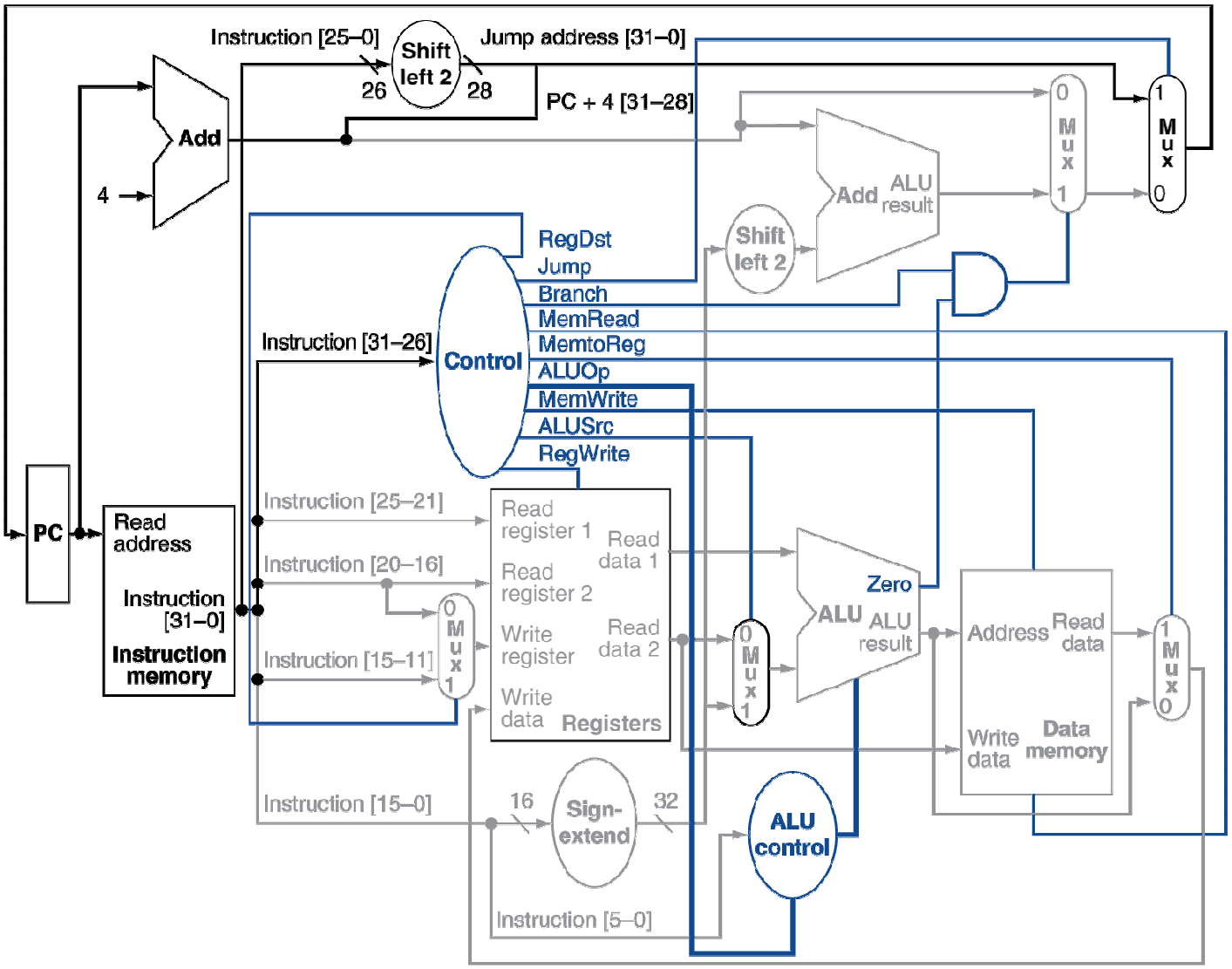


Implementing Jumps



- 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

Datapath With Jumps Added



Performance of Single-Cycle Machines

- Assume that the operation times for the major functional units in this implementation are the following:
  - Memory units: 200 picoseconds (ps)
  - ALU and adders:100 ps
  - Register file (Read or write):50 ps
  - Wires and other units:0 ps
- Assume the following instruction mix:
  - 25% load, 10% store,45% ALU instructions, 15% branches, and 5% jumps
- Which of the following implementations would be faster and by how much?
  - I.Every instruction operates in one clock cycle of a fixed length.
  - II.Every instruction operates in one clock cycle using a variable-length clock, which for each instruction is only as long as it needs to be.

Performance of Single-Cycle Machines

CPU execution time = Instruction count ×CPI ×Clock cycle time

Instruction class	Instruction memory	Register read	ALU operation	Data memory	Register write	Total
R-type	200	50	100	0	50	400 ps
Load word	200	50	100	200	50	600 ps
Store word	200	50	100	200		550 ps
Branch	200	50	100	0		350 ps
jump	200					200 ps

- The clock cycle in a single clock machine is determined by the longest instructionwhich is 600ps.
- The average time per instruction with a variable clock is:600×25% + 550 ×10% +400 ×45% +350 ×15% + 200 ×5%= 447.5ps600 / 447.5 = 1.34
- The variable clock implementation would be 1.34times faster. Unfortunately this implementation is extremely difficult.