Lecture – 15

The ALU Control

- The main control unit reads instruction opcode and generates the necessary control signal.
- ALU Control is a small control unit separate from the main control unit.
- It takes 2-bit control signal ALUOp from the main control unit and the instruction function field [5:0] and generates the necessary control signal for the ALU.

Multi-level Decoding:

- It reduces the size of main control unit.
- Increases the speed of control unit.

The ALU Control

- ALU has four control inputs. Only 6 of the 16 possible input combinations are used in this subset.

ALU control lines	Function
0000	AND
0001	OR
0010	Add
0110	Subtract
0111	set on less than
1100	NOR

- For load word and store word instructions, we use the ALU to compute the memory address by addition.
- For the R-type instructions, the ALU needs to perform one of the five actions (AND, OR, subtract, add, or set on less than) depending on the value of the 6-bit funct field.
- For branch equal, the ALU perform a subtraction.

The ALU Control

- A 2-bit control field called *ALUOp* indicates what type operation is carried out.
- ALUOp is (00) to perform add for load and store instruction.
- ALUOp is (01) to perform subtract for beq.
- ALUOp is (10) for arithmetic-logical instruction.

The Truth Table

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	and	0000
R-type	10	OR	100101	or	0001
R-type	10	set on less than	101010	set on less than	0111

The Truth Table

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

Instruction Format

Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

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

b. Load or store instruction

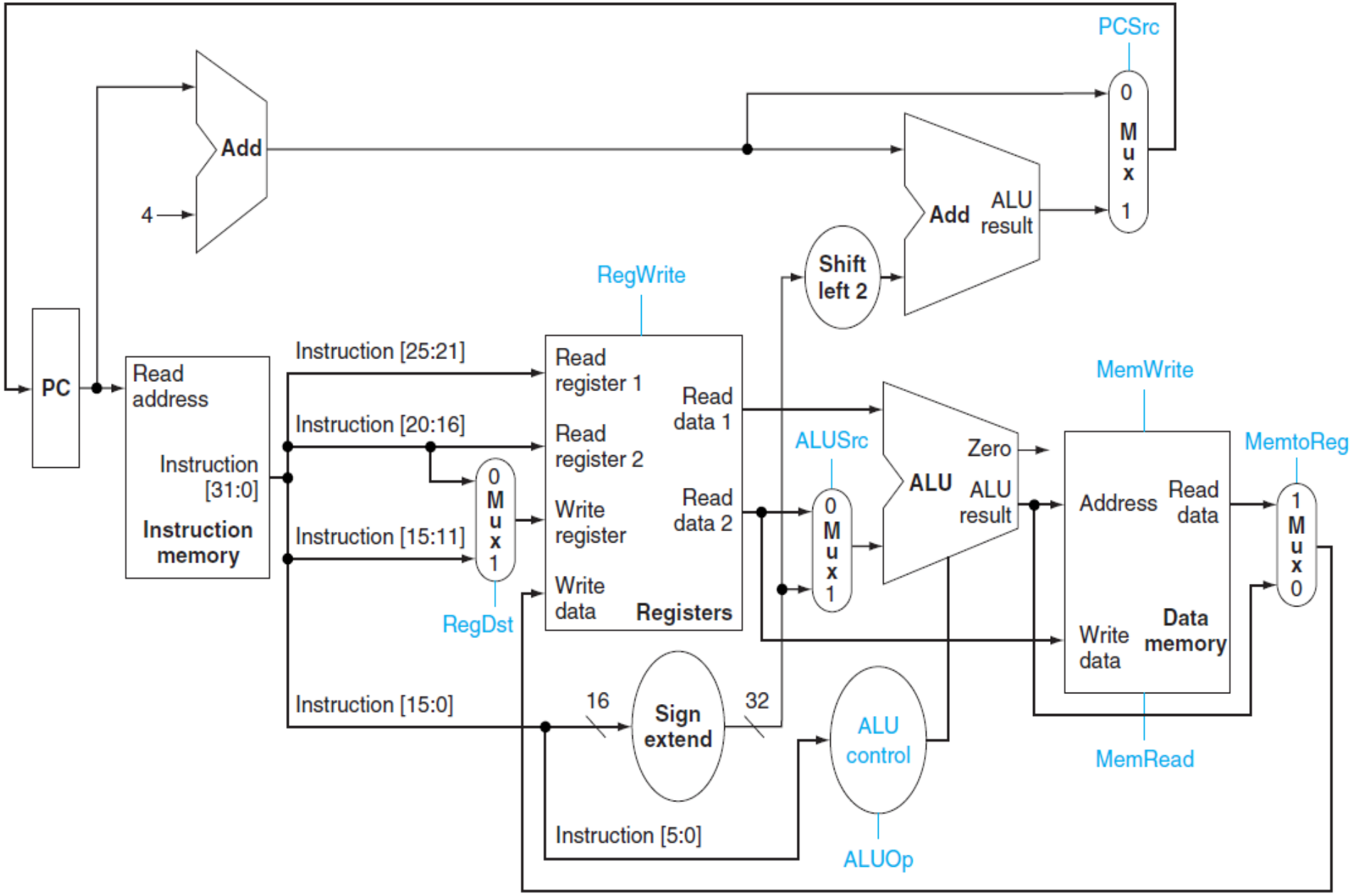
Field	4	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

c. Branch instruction

Several Major Observations

- The **opcode** is always contained in bits 31:26.
- Two registers to be read always specified by the **rs**[25:21] and **rt** [20:16]. This is true for R-type, branch equal and store instruction.
- The base register for load and store instruction is always in **rs**[25:21].
- The 16-bits offset for branch equal, load and store is always in position 15:0.
- The destination register for load is **rt**[20:16] and for R-type instruction it is **rd**[15:11].

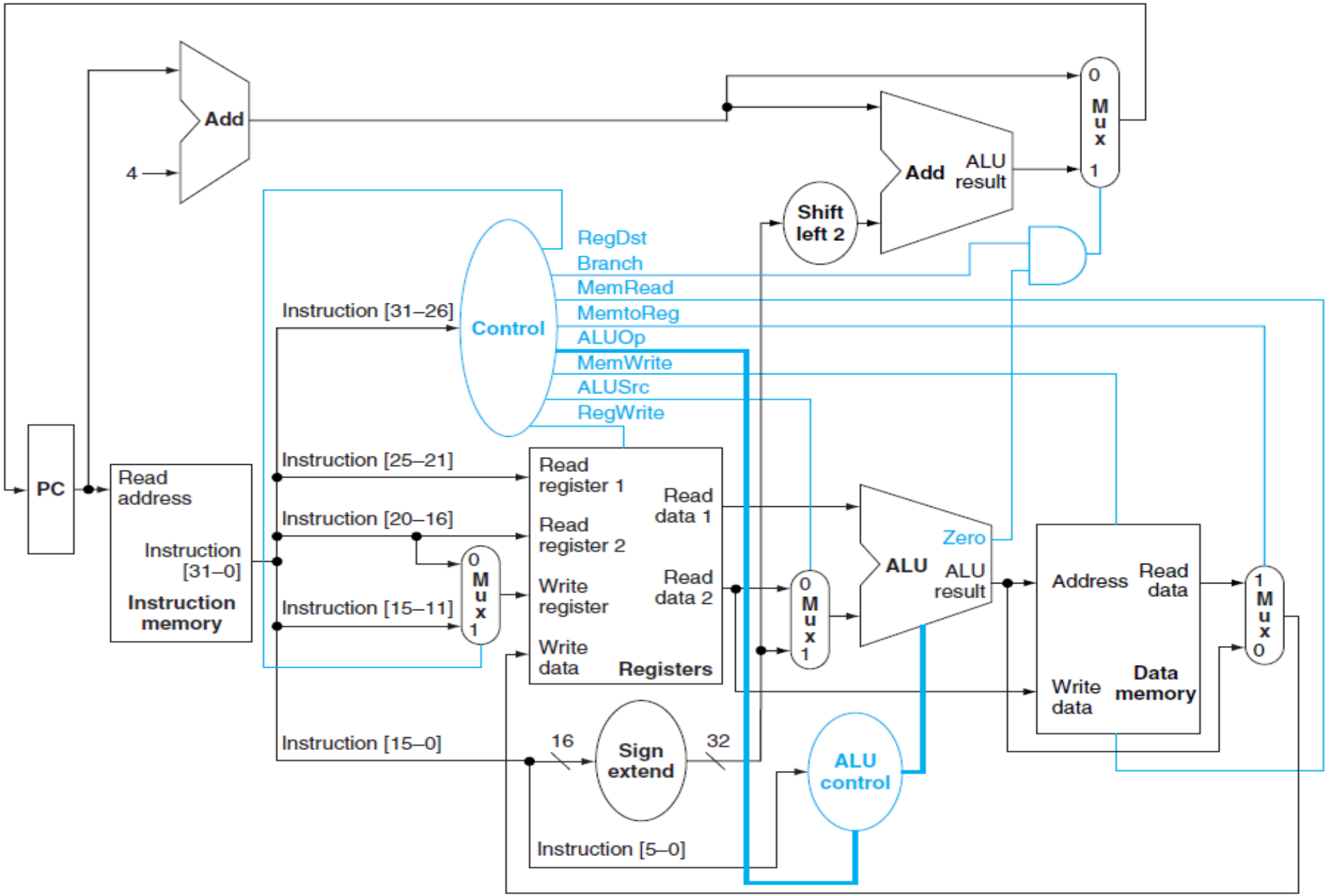
Single Cycle Processor with Necessary Control Signal



The 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.

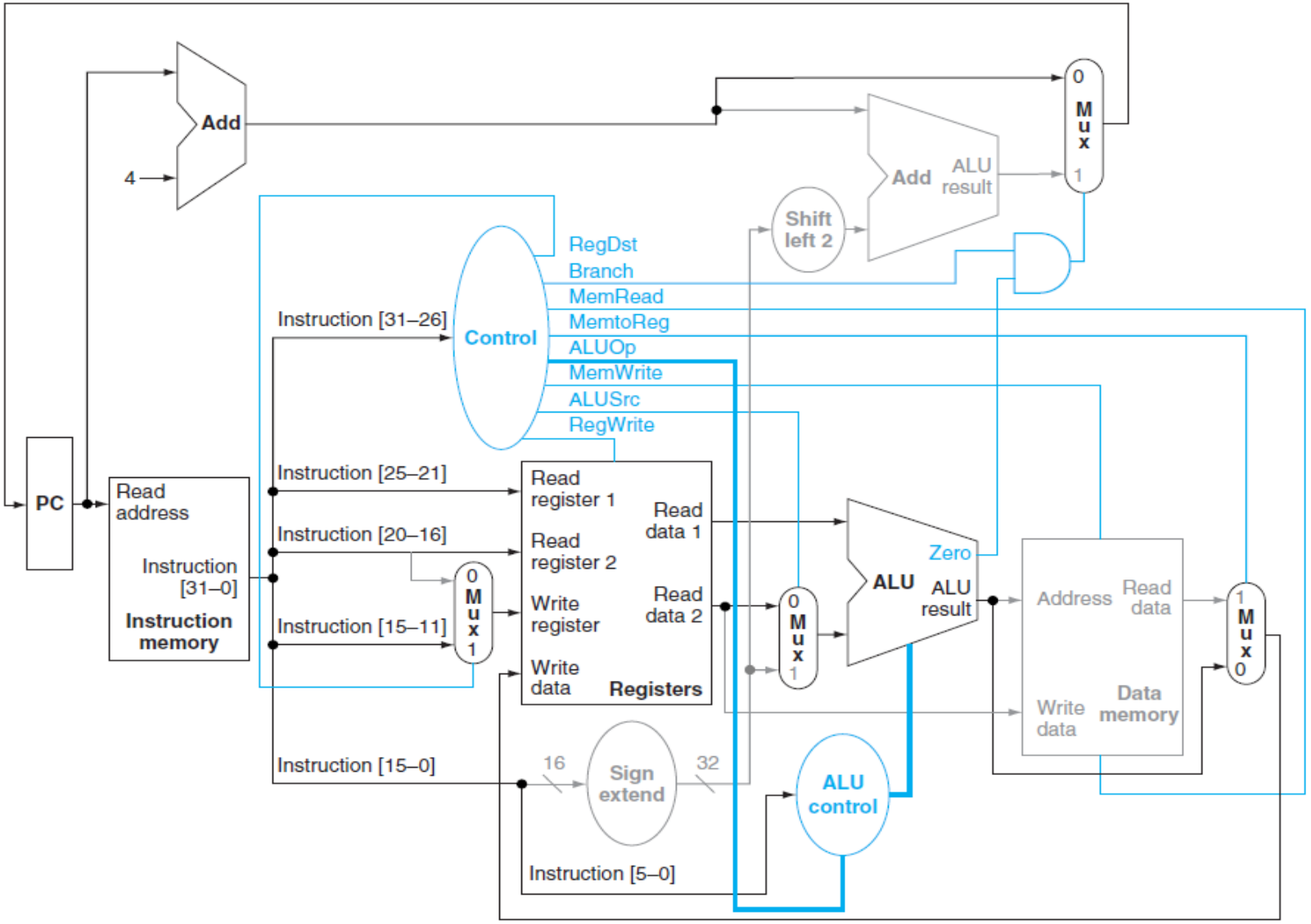
The Simple Datapath with Control Unit



The Setting of Control Lines

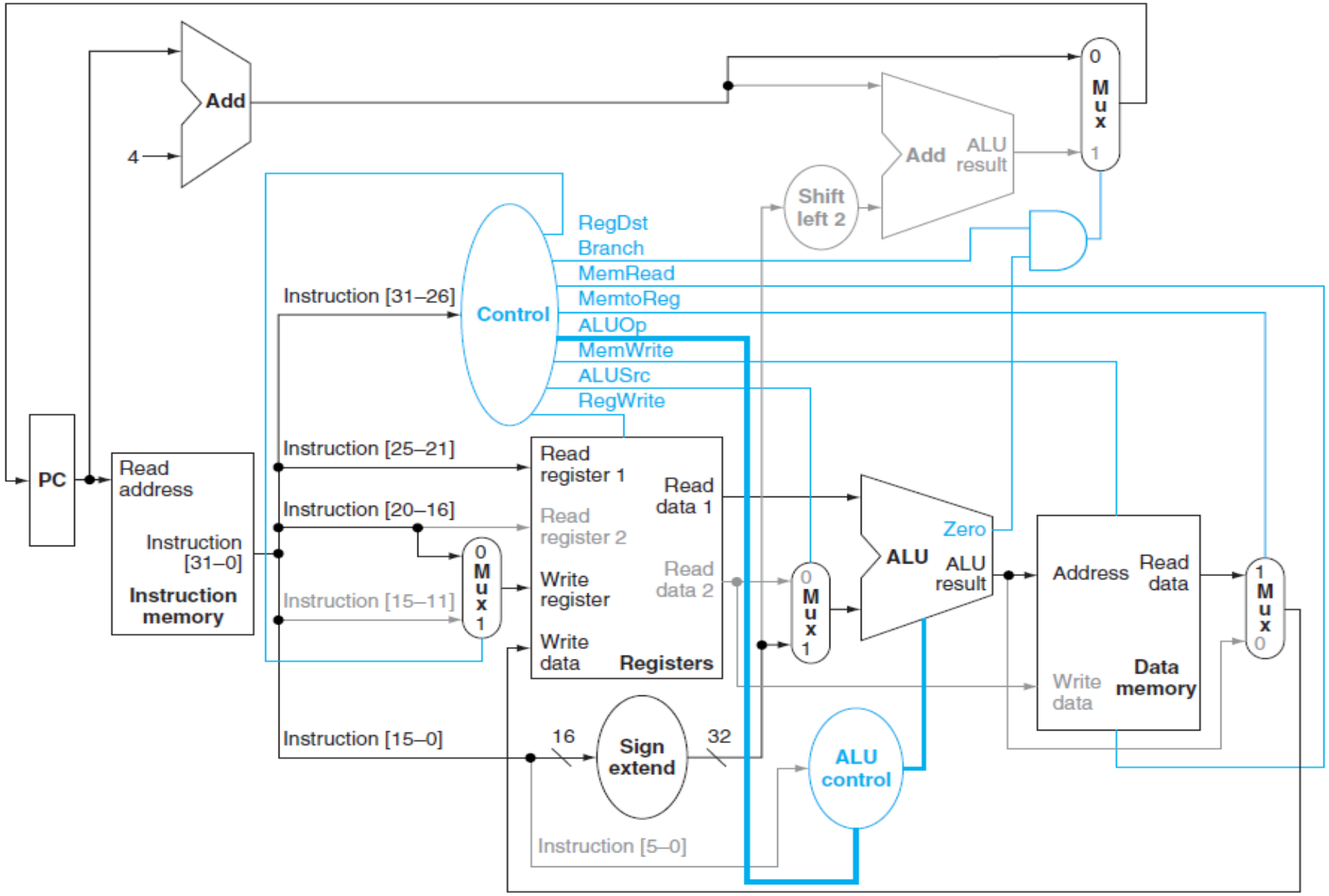
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

The Datapath in Operation for an R-type Instruction



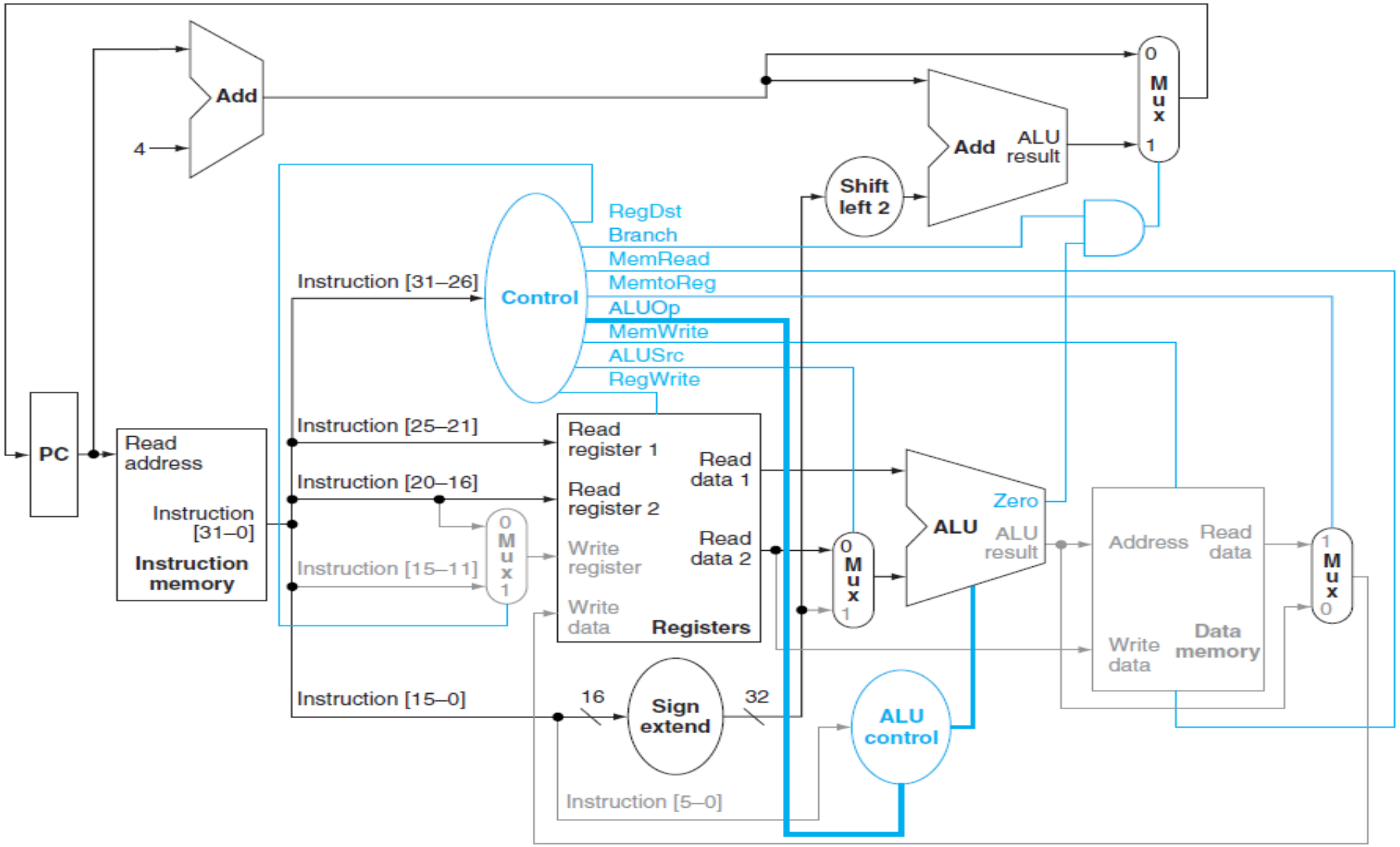
RegDst=1 **ALUSrc=0** **MemtoReg=0** **RegWrite=1**
MemWrite=0 **MemRead=0** **Branch=0** **ALUOp=10**

The Datapath in Operation for a Load Instruction



RegDst=0 **ALUSrc=1** **MemtoReg=1** **RegWrite=1**
MemWrite=0 **MemRead=1** **Branch=0** **ALUOp=00**

The Datapath in Operation for a Branch Equal Instruction



RegDst=X **ALUSrc=0** **MemtoReg=X** **RegWrite=0**
MemWrite=0 **MemRead=0** **Branch=1** **ALUOp=01**

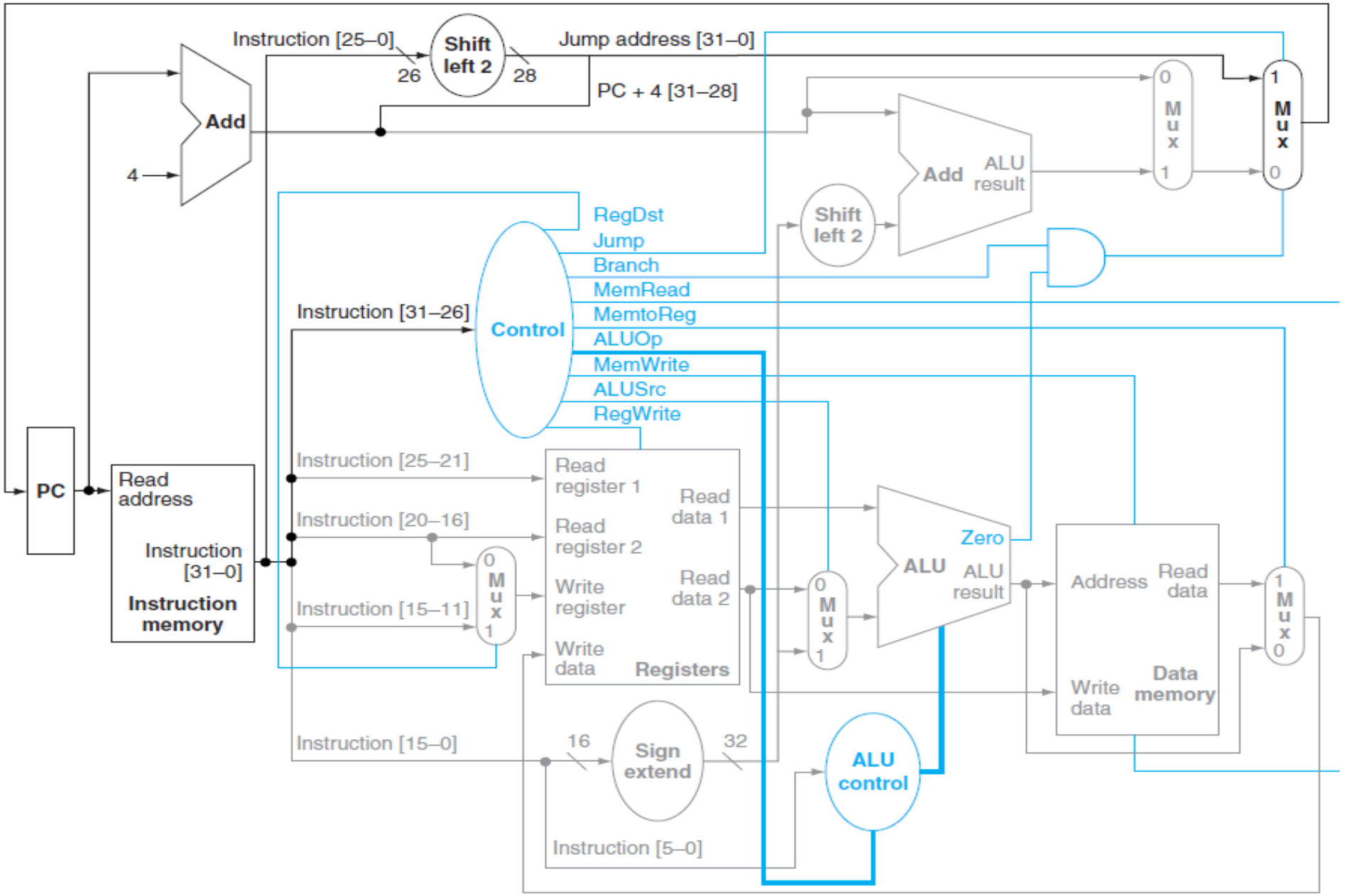
Implementing Jumps

Format of J-type Instruction:

- A jump is calculated by storing into the PC the concatenation of:
  1. The upper 4-bits of the current PC+4
  2. The 26-bit immediate field of the Jump instruction
  3. The bits 00_2

- Implementation of Jump requires:
  1. An additional multiplexor
  2. Control signal *Jump* from the main control unit.

Control and Datapath to Handle the Jump Instruction



Why a Single Cycle Instruction Is Not Used Today?

- The clock cycle must have same length for every instruction.
- The cycle time must be long enough for the load instruction.
- The performance is not good since, several of the instruction classes could fit in a shorter clock cycle.