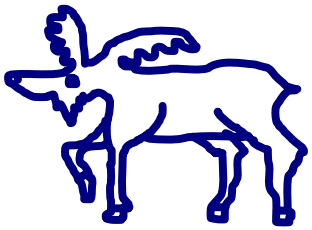
Lecture 12
Control Unit Design

Byung-gi Kim
School of Computer Science and Engineering
Soongsil University



4. The Processor

Designing the Main Control Unit

- Control signals derived from instruction

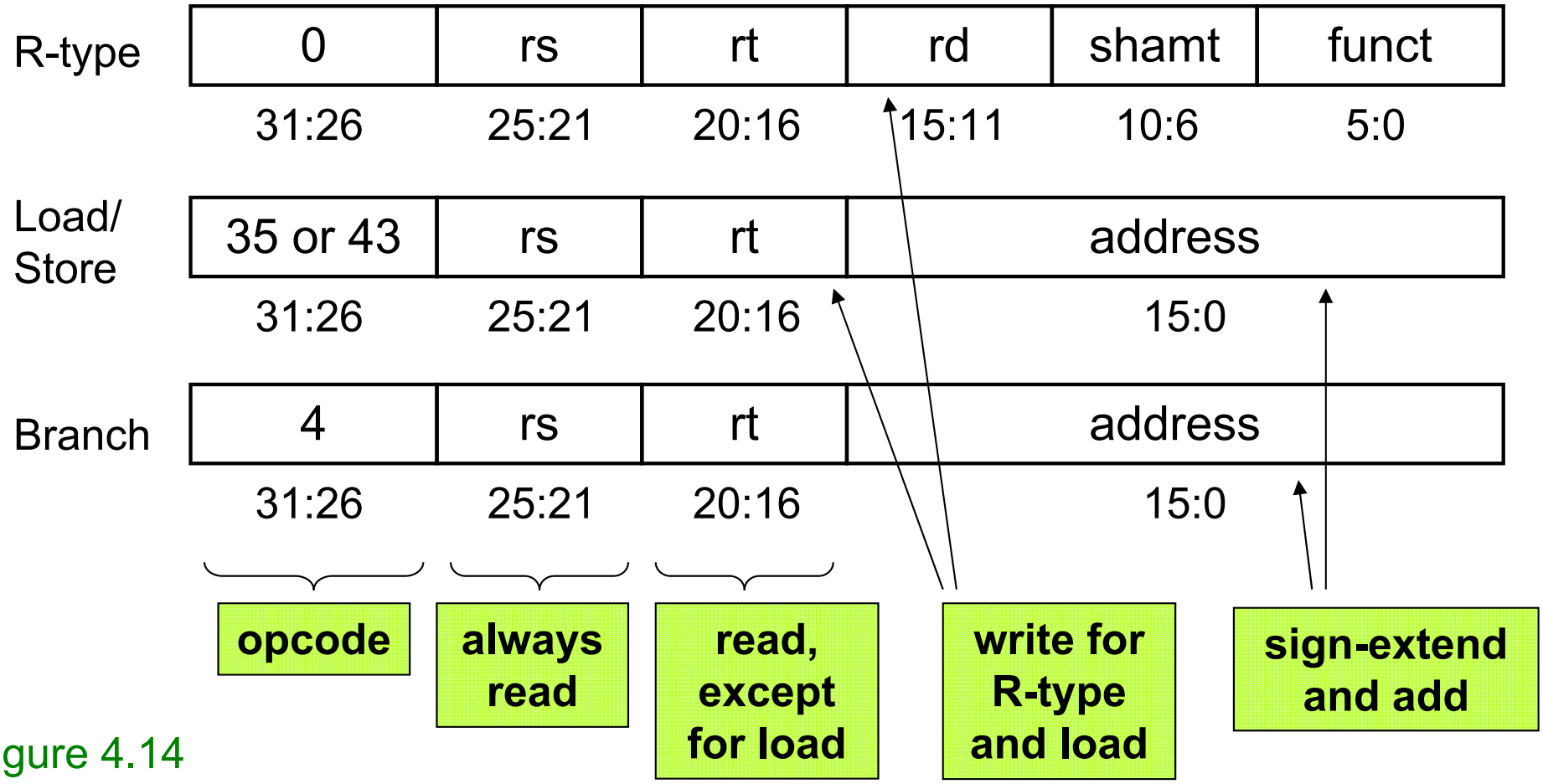


Figure 4.14

Datapath with Control Lines

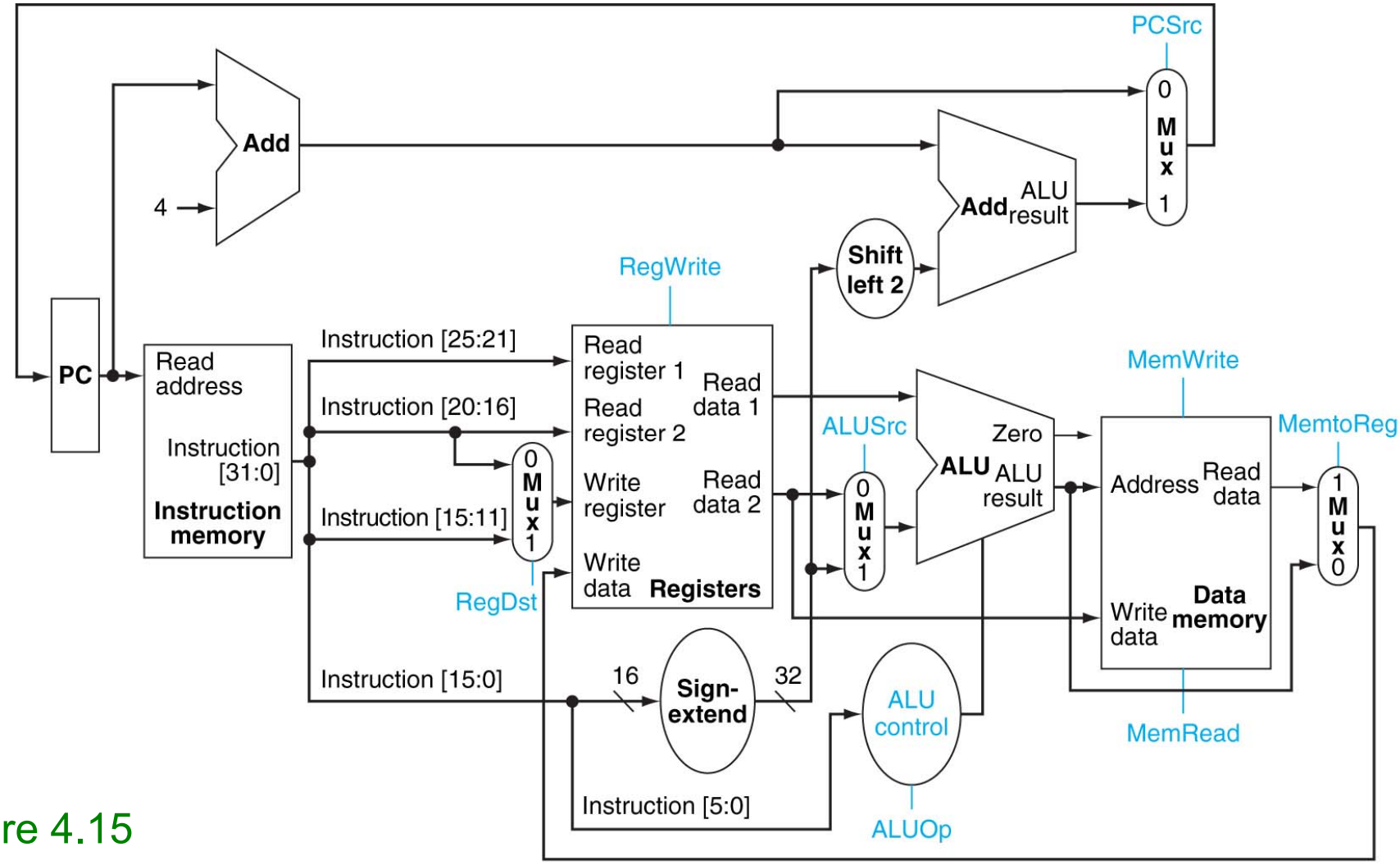


Figure 4.15

The Function 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.

Figure 4.16

Control Unit

- **PCSrc = Branch · Zero (from ALU)**

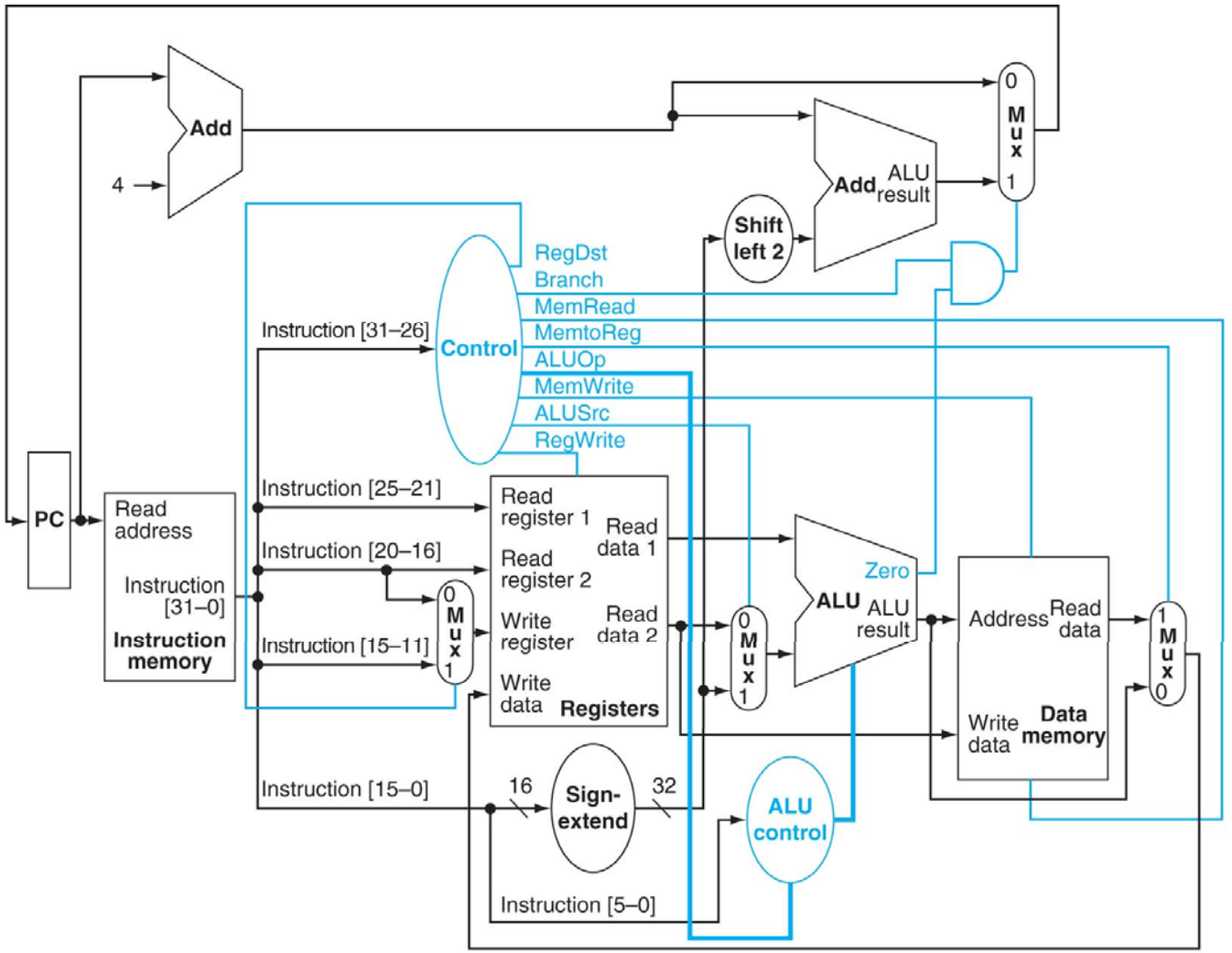


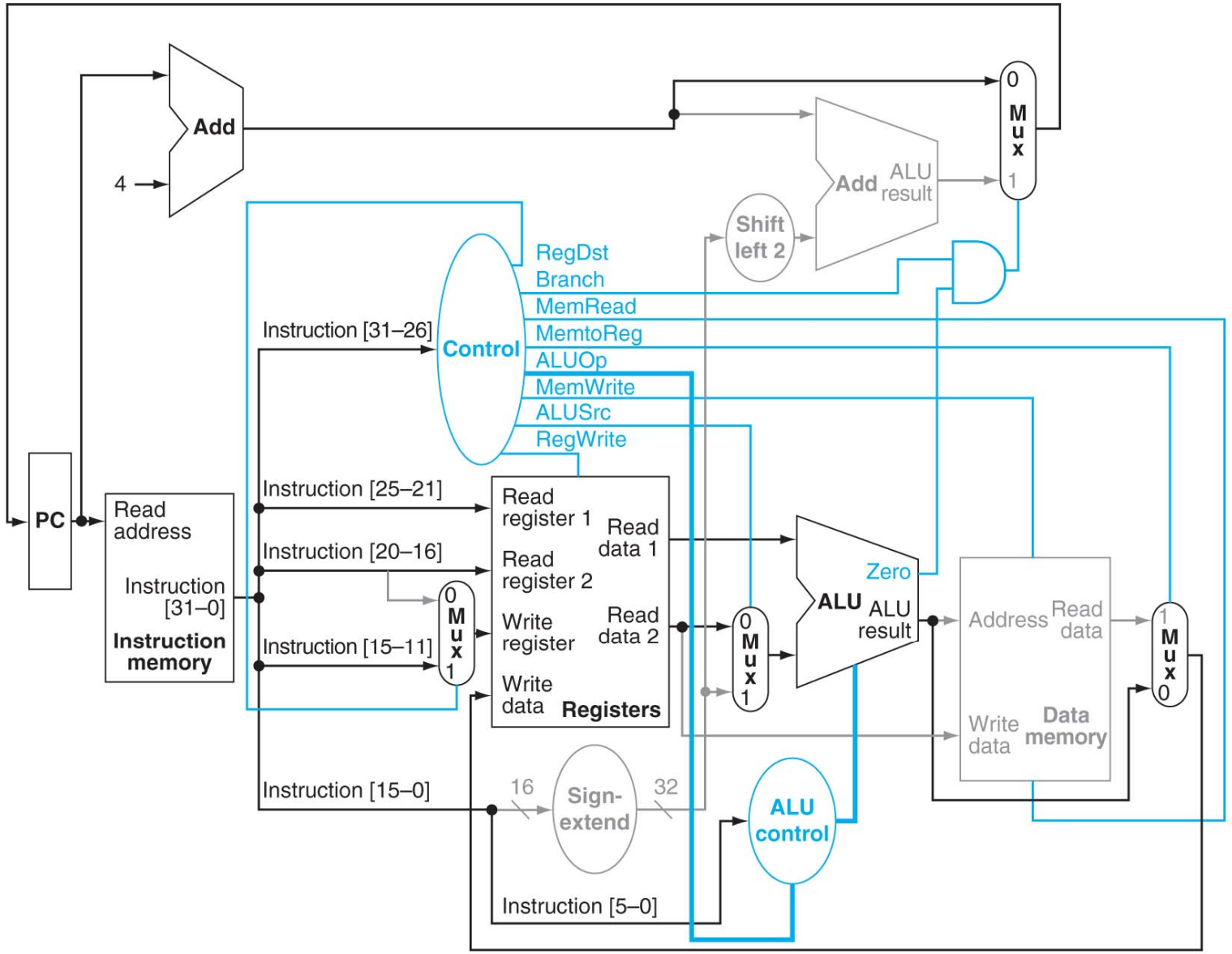
Figure 4.17

Execution of an R-type Instruction

- e.g. **`add $t1,$t2,$t3`**

1. Fetch instruction and increment PC.
2. Read two registers (**`$t2`** and **`$t3`**) and generate control signals in the main control unit.
3. ALU operates on the data, using the function code.
4. ALU result is written into **`$t1`**.

Datapath for an R-type Instruction



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

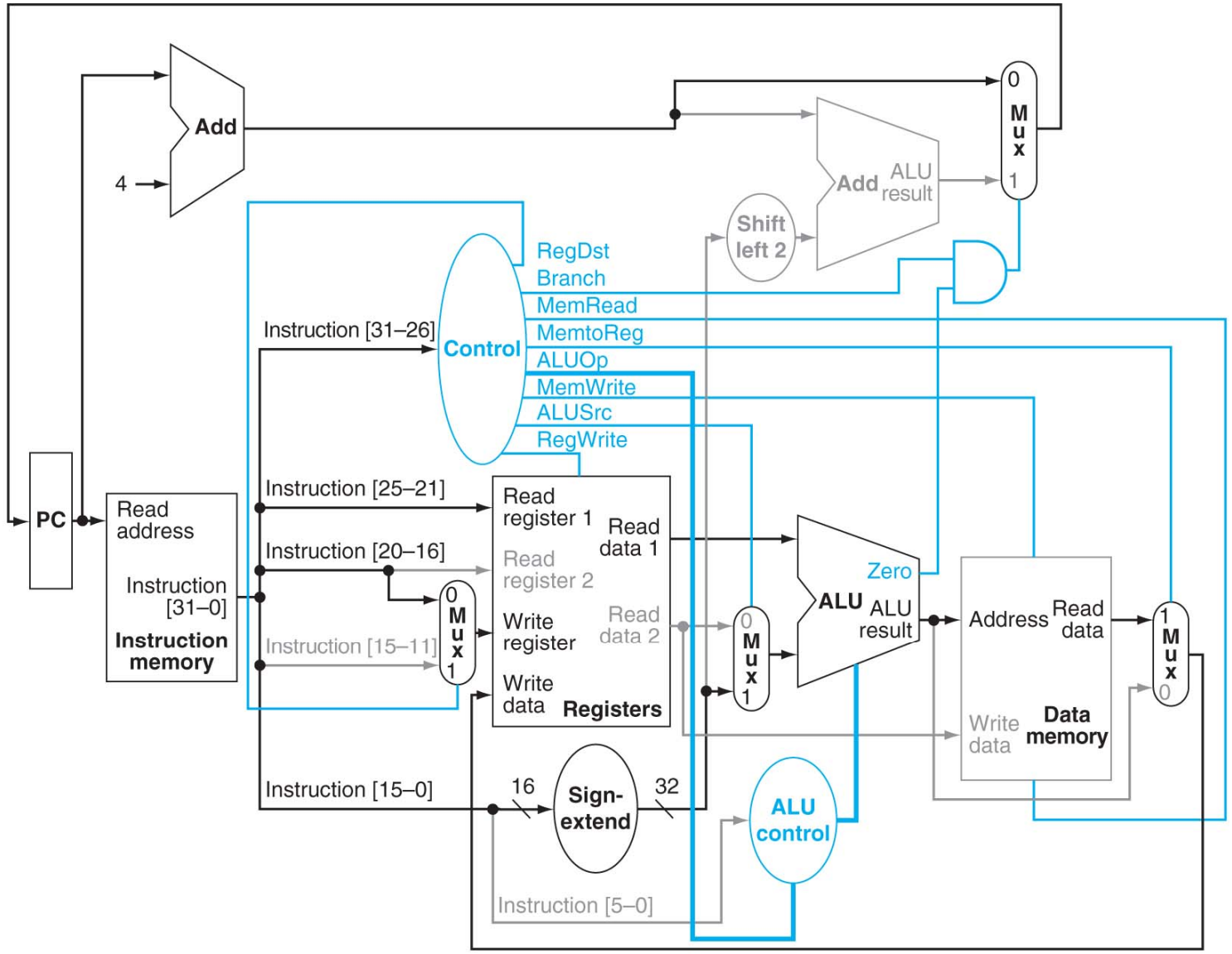
Figure 4.19

Execution of a load Instruction

- e.g. `lw $t1,offset($t2)`

1. Fetch instruction and increment PC.
2. Read a register (`$t2`) and generate control signals in the main control unit.
3. ALU computes the effective address by adding `$t2` and sign-extended offset.
4. Use the ALU output as the address for the data memory.
5. Write the data from the memory into `$t1`.

Datapath for a load Instruction



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

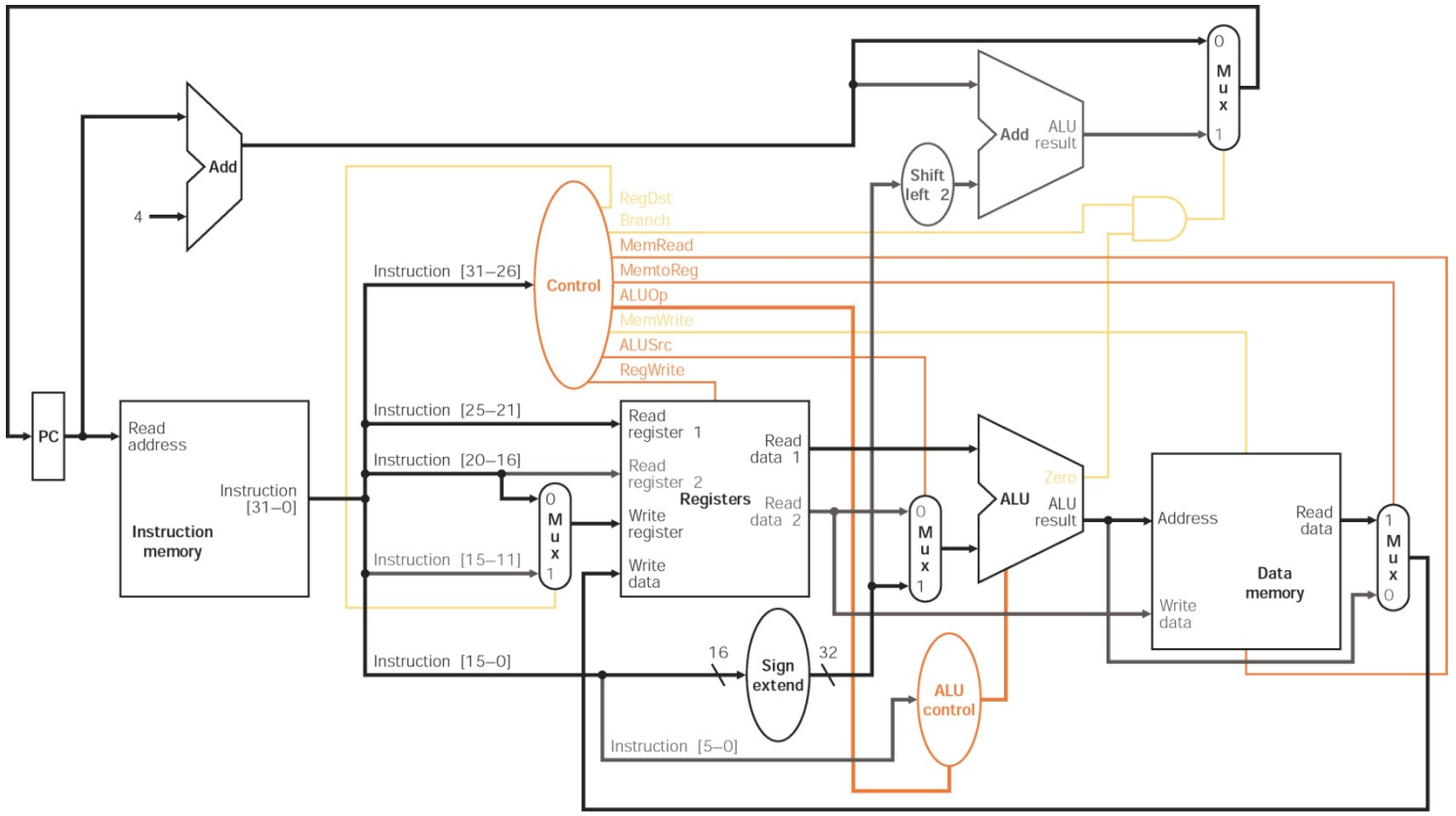
Figure 4.20

Execution of a store Instruction

- e.g. **`sw $t1,offset($t2)`**

1. Fetch instruction and increment PC.
2. Read two registers (**`$t1 and $t2`**) and generate control signals in the main control unit.
3. ALU computes the effective address by adding **`$t2`** and sign-extended offset.
4. Write **`$t1`** into memory using the ALU output as the address for the data memory.

Datapath for a store Instruction



RegDst	X
ALUSrc	1
MemtoReg	X
RegWrite	0
MemRead	0
MemWrite	1
Branch	0
ALUOp	00

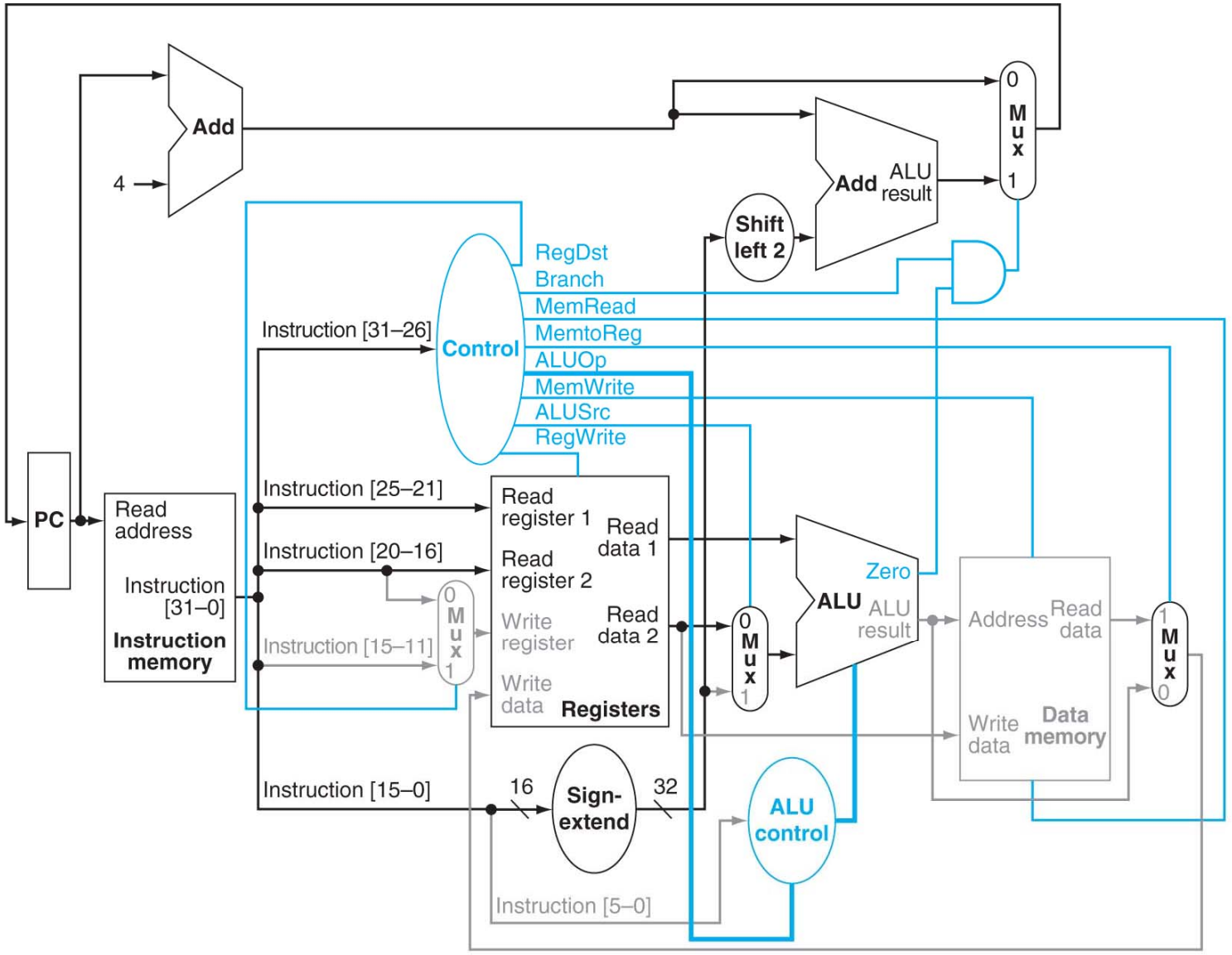
Figure 5.25 in 2ed

Execution of a branch Instruction

- e.g. **`beq $t1,$t2,offset`**

1. Fetch instruction and increment PC.
2. Read two registers (**`$t1`** and **`$t2`**) and generate control signals in the main control unit.
3. Subtract the two data from registers. And compute branch target address by adding (PC+4) and sign-extended and shifted-left **`offset`**.
4. Decide which adder result to store into the PC using the Zero result from ALU.

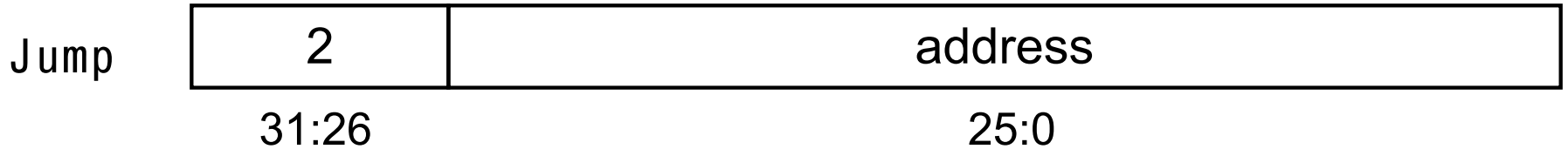
Datapath for a branch Instruction



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

Figure 4.21

Example: Implementing Jumps



- Jump uses word address
- Update PC with concatenation of
  - Top 4 bits of (PC+4) ⇨ PC[31-28] ← PC+4[31-28]
  - 26-bit jump address ⇨ PC[27-2] ← instruction[25-0]
    (=immediate field)
  - 00 ⇨ PC[1-0] ← 00
- Need an extra control signal decoded from opcode

Datapath With Jumps Added

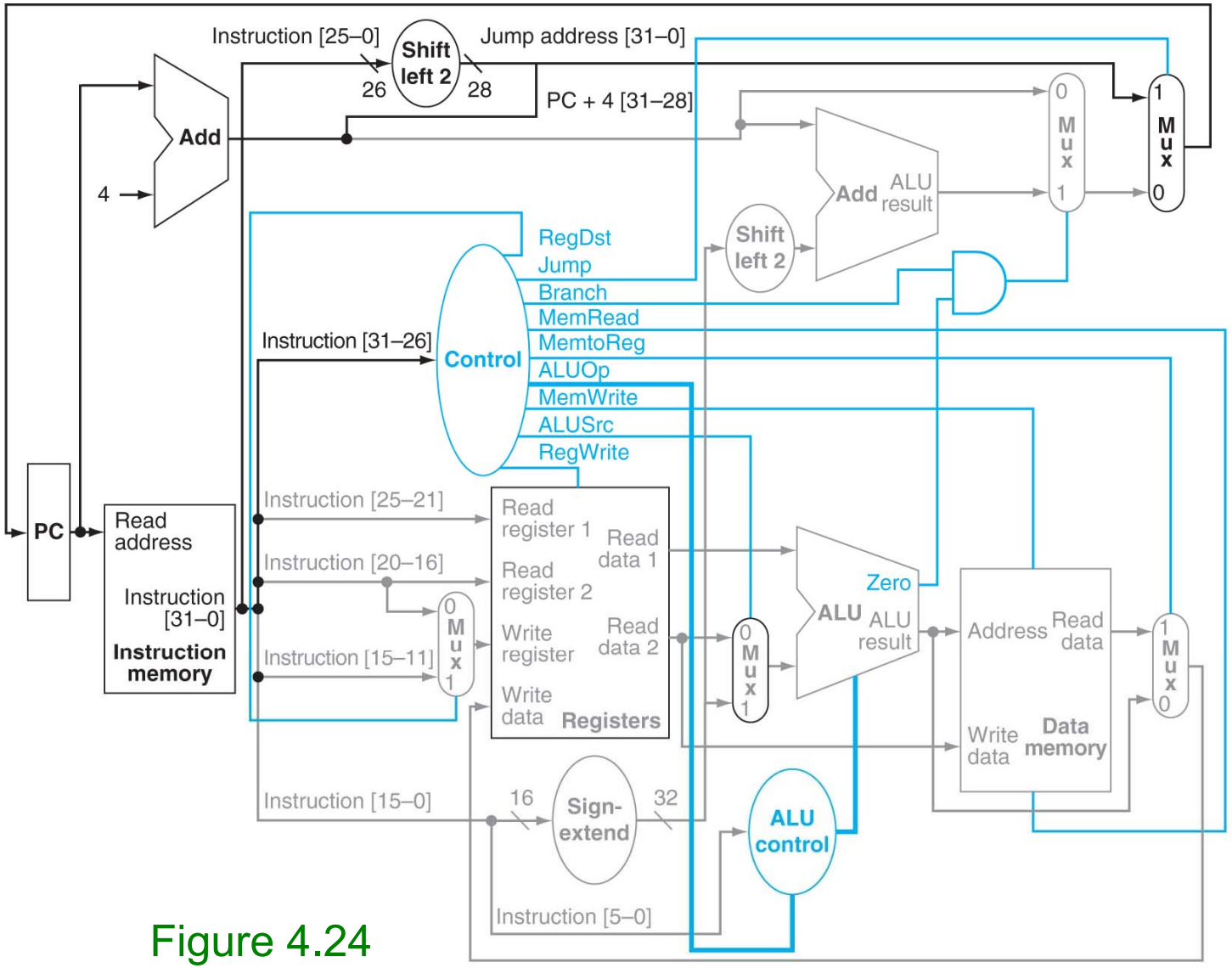


Figure 4.24

RegDst	X
ALUSrc	X
MemtoReg	X
RegWrite	0
MemRead	0
MemWrite	0
Branch	X
ALUOp	XX
Jump	1

Control Unit Outputs for Each Instruction

Instruction	Reg Dst	ALU Src	Mem toReg	Reg Write	Mem Read	Mem Write	Branch	ALU-Op1	ALU-Op0	Jump
R-format	1	0	0	1	0	0	0	1	0	0
lw	0	1	1	1	1	0	0	0	0	0
sw	X	1	X	0	0	1	0	0	0	0
beq	X	0	X	0	0	0	1	0	1	0
j	X	X	X	0	0	0	X	X	X	1
addi										

Modified Figure 4.18