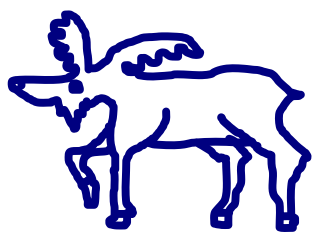
# Lecture 11
# Datapath Design

**Byung-gi Kim**

**School of Computer Science and Engineering**

**Soongsil University**



# 4. The Processor

# 4.3 Building a Datapath

- **Datapath elements for instruction fetch**
  - Instruction memory: Store the instructions of a program
  - Program Counter (PC): Store the address of the instruction
  - Adder: Increment the PC to the address of the next instruction

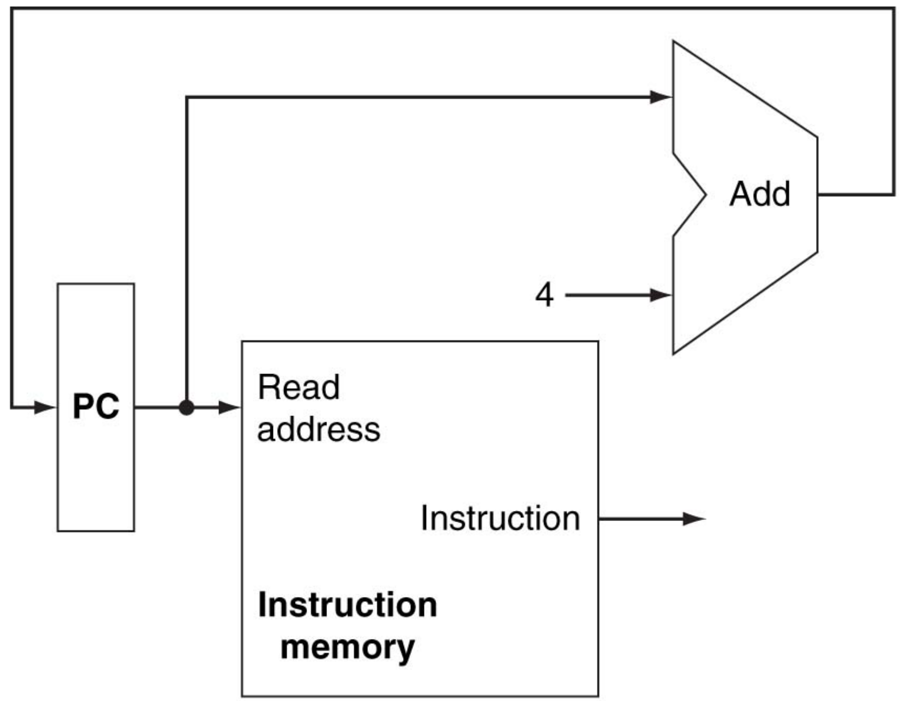



Figure 4.6

# Executing R-format Instructions

- **Step 3**

  Use ALU for the opcode execution

- **Step 4**

  Write the data from the ALU back into a register

- **Major components**

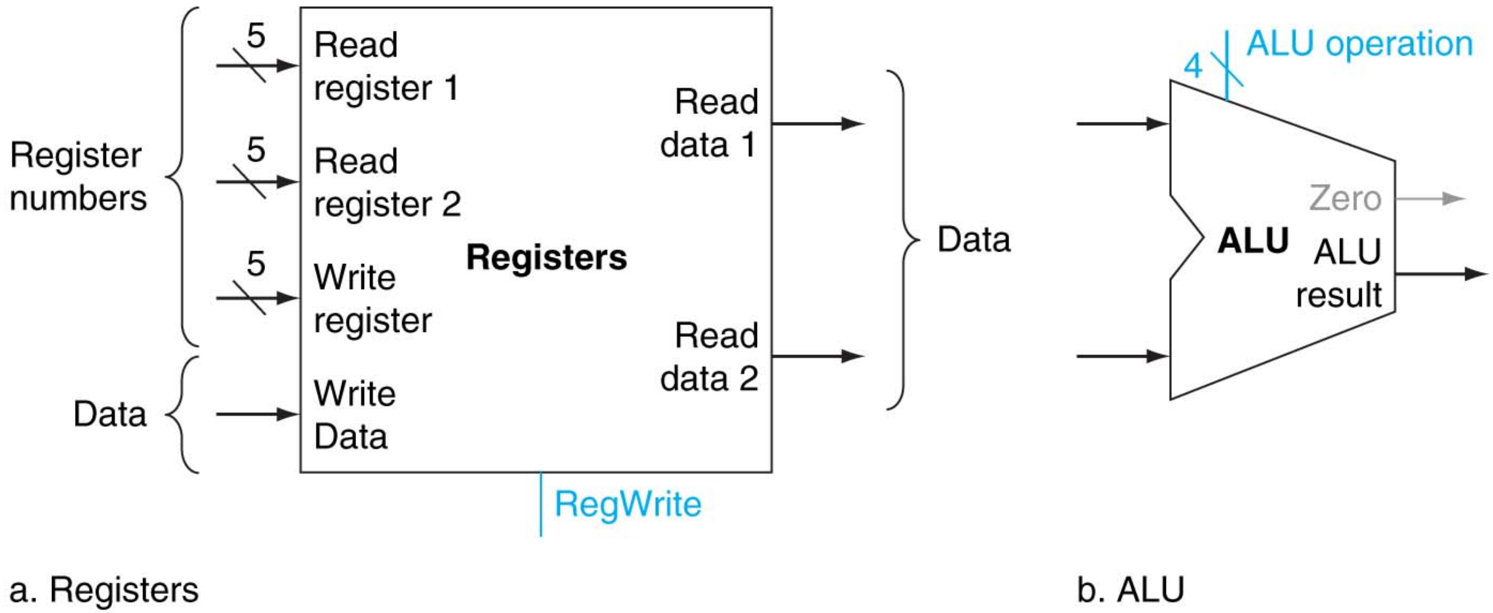
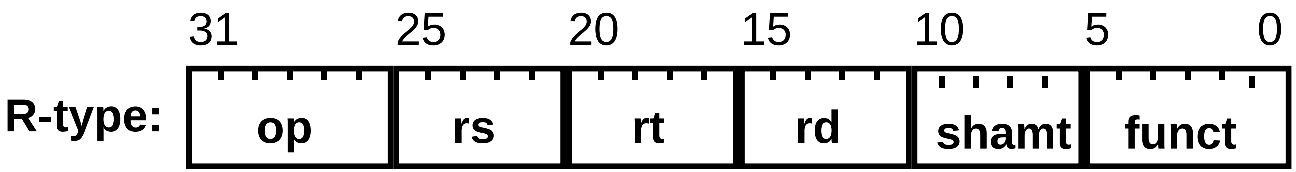
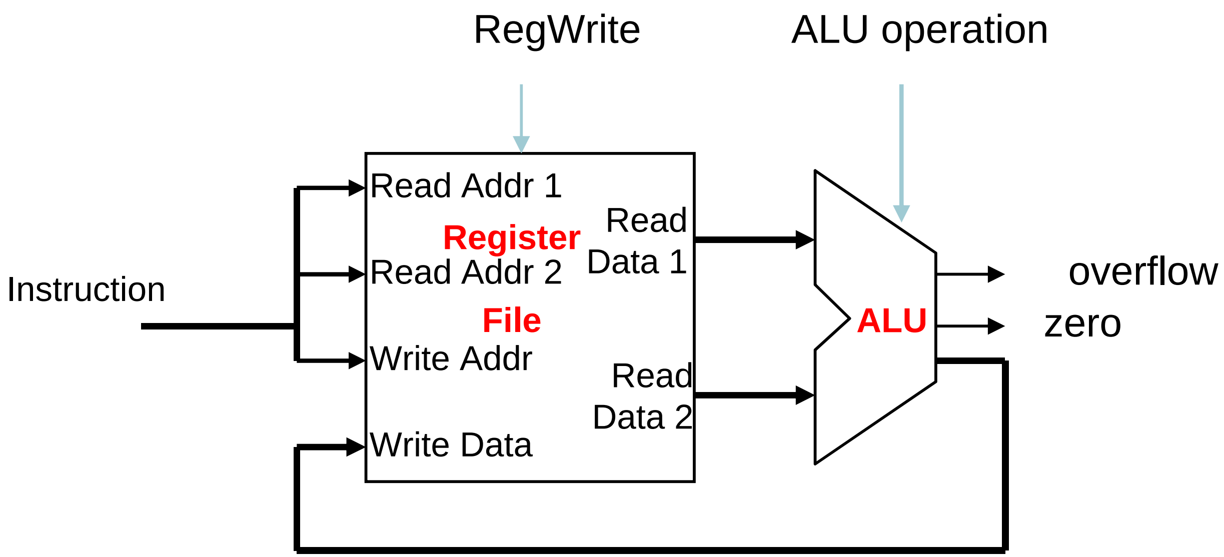



Figure 4.7

# Datapath for R-format Instructions




- Note that Register File is not written every cycle (e.g. **`sw`**), so we need an explicit write control signal for the Register File

# Executing Memory Reference Instructions

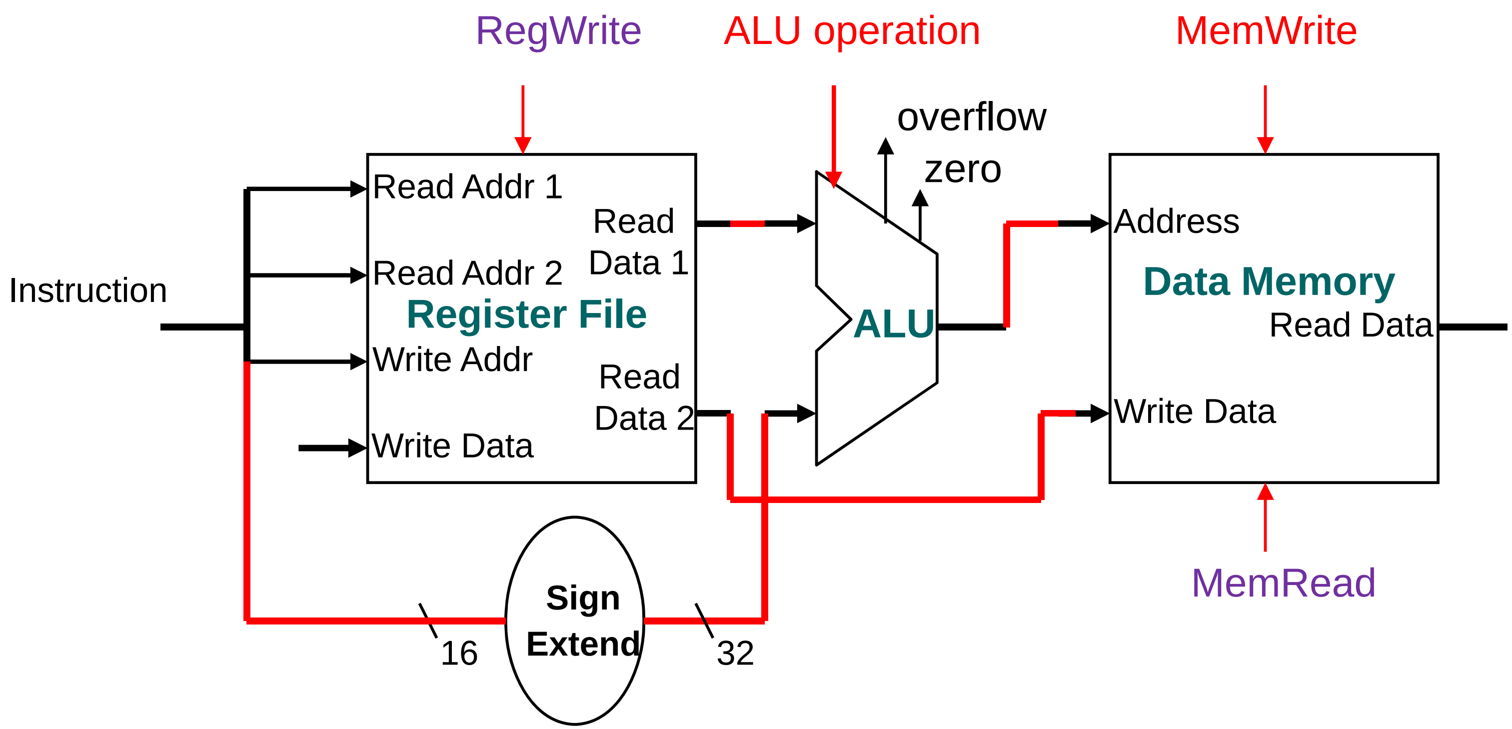
- **Step 3**

  Compute a memory address by adding the base register to the sign-extended 16-bit offset

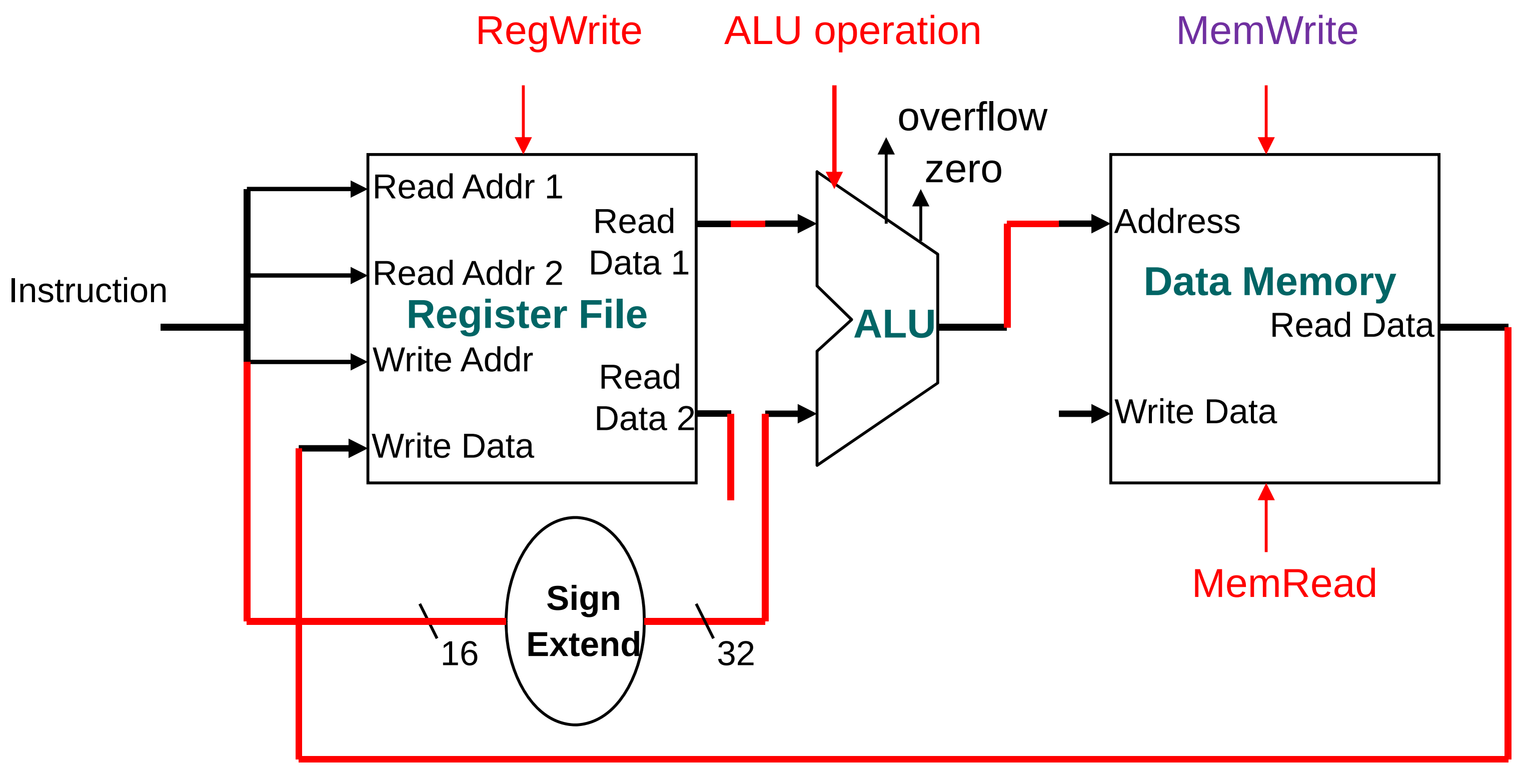
- **Step 4**
  - **`sw $t1,offset_value($t2)`**
    - Write **`$t1`** into memory
  - **`lw $t1,offset_value($t2)`**
    - Read a value from memory
    - Write it into **`$t1`**

- **Major components**
  - Sign extension unit and data memory

# Datapath for store Instructions



# Datapath for load Instructions



# Executing Branch Instructions

- **Step 3**

  Use ALU for comparison

  Compute branch target address by adding the sign-extended offset to the PC

- **Step 4**

  Change PC based on the comparison

- **Major components**
  - Shift-left-2 unit
  - Separate adder

# Datapath for Branch Instruction

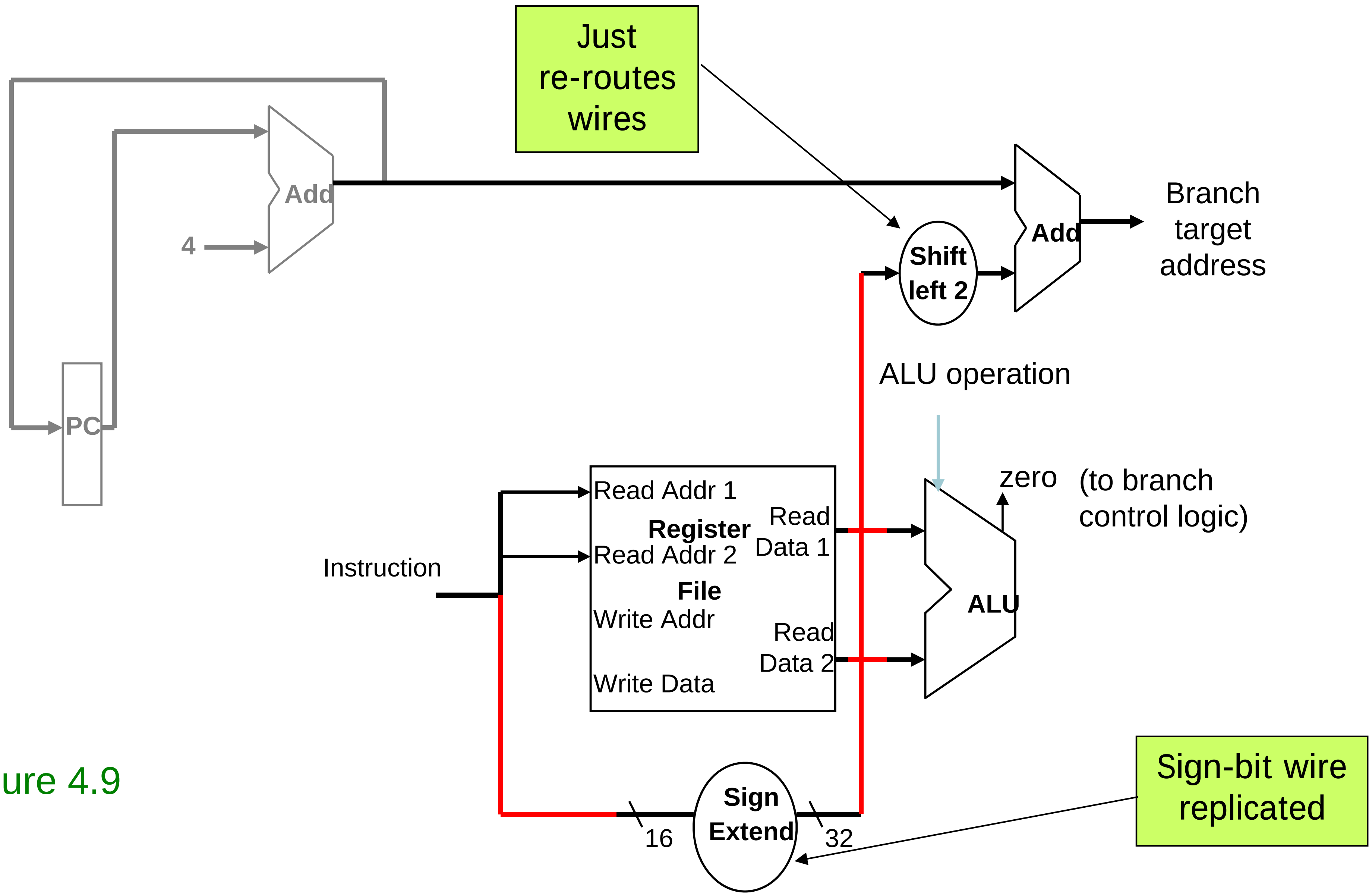
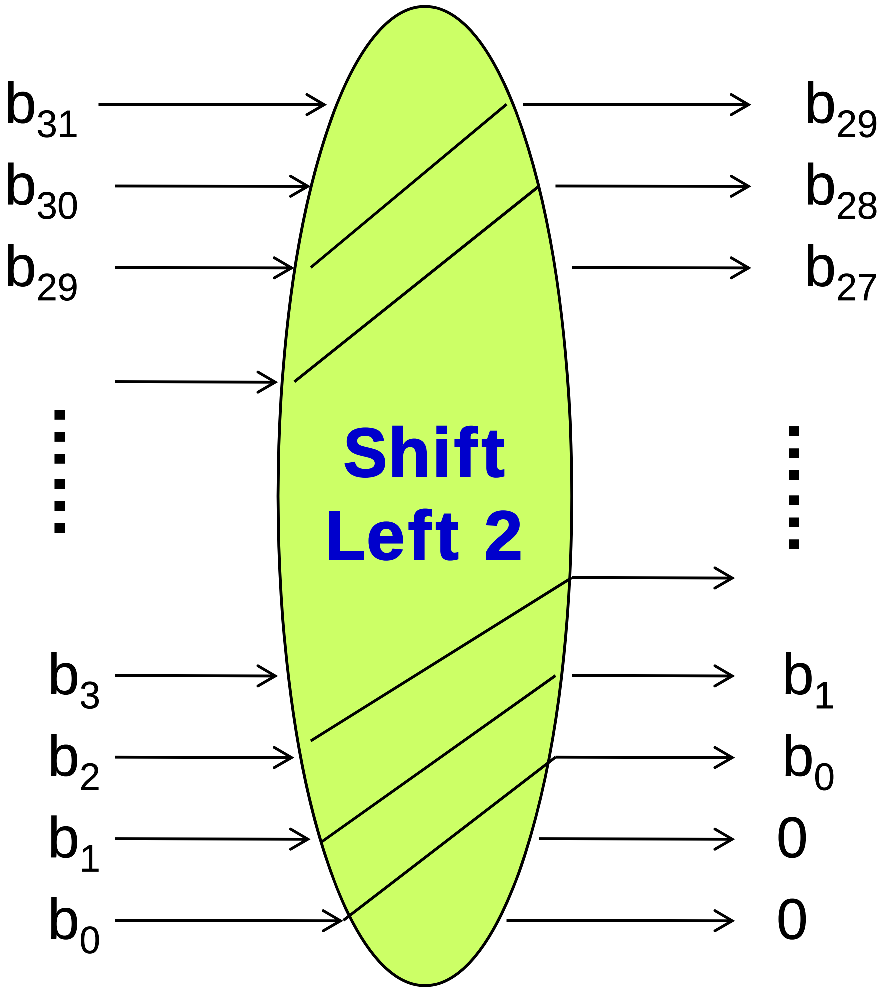
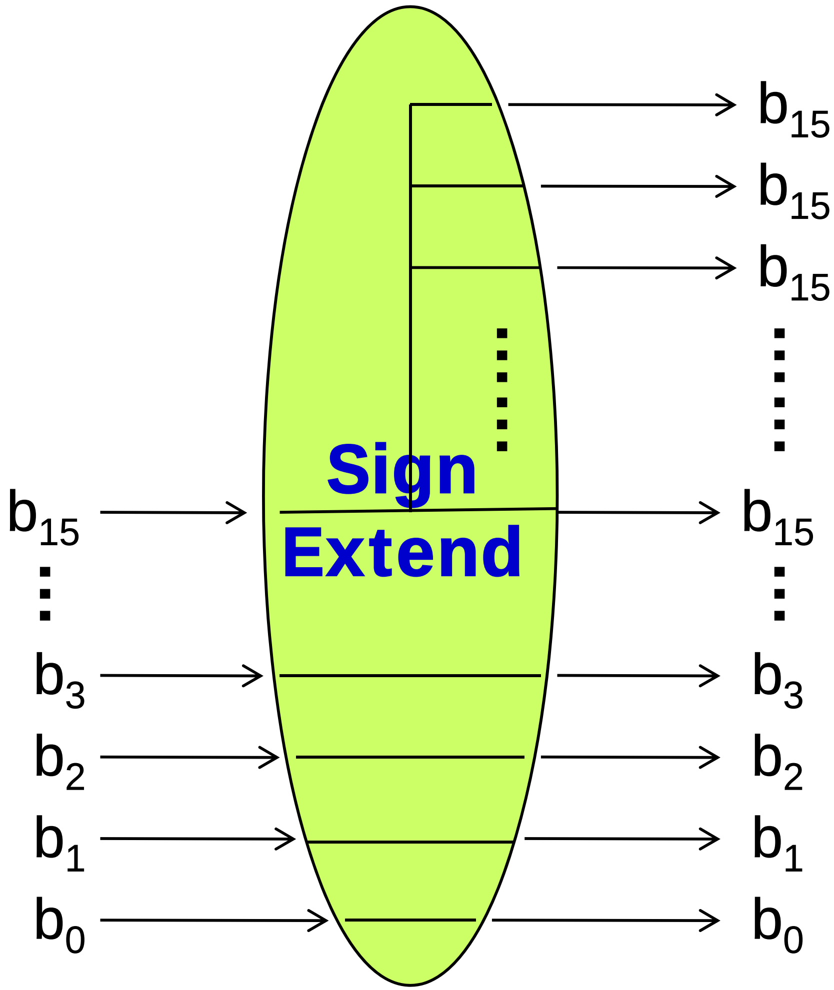



Figure 4.9

# Units Implementation



# Creating a Single Datapath

- **The simplest datapath**
  - Single-cycle datapath ( ↔ Multi-cycle datapath)
    - Execute all instructions in one clock cycle
    - No datapath resource can be used more than once per instruction.
    - Any element needed more than once must be duplicated.
    - Separate instruction and data memories
  - Sharing a datapath between two different instruction types
    - Use multiplexor

# Example: Building a Datapath

- **Combine the arithmetic-logical instruction datapath and the memory instruction datapath.**

**[Answer]**

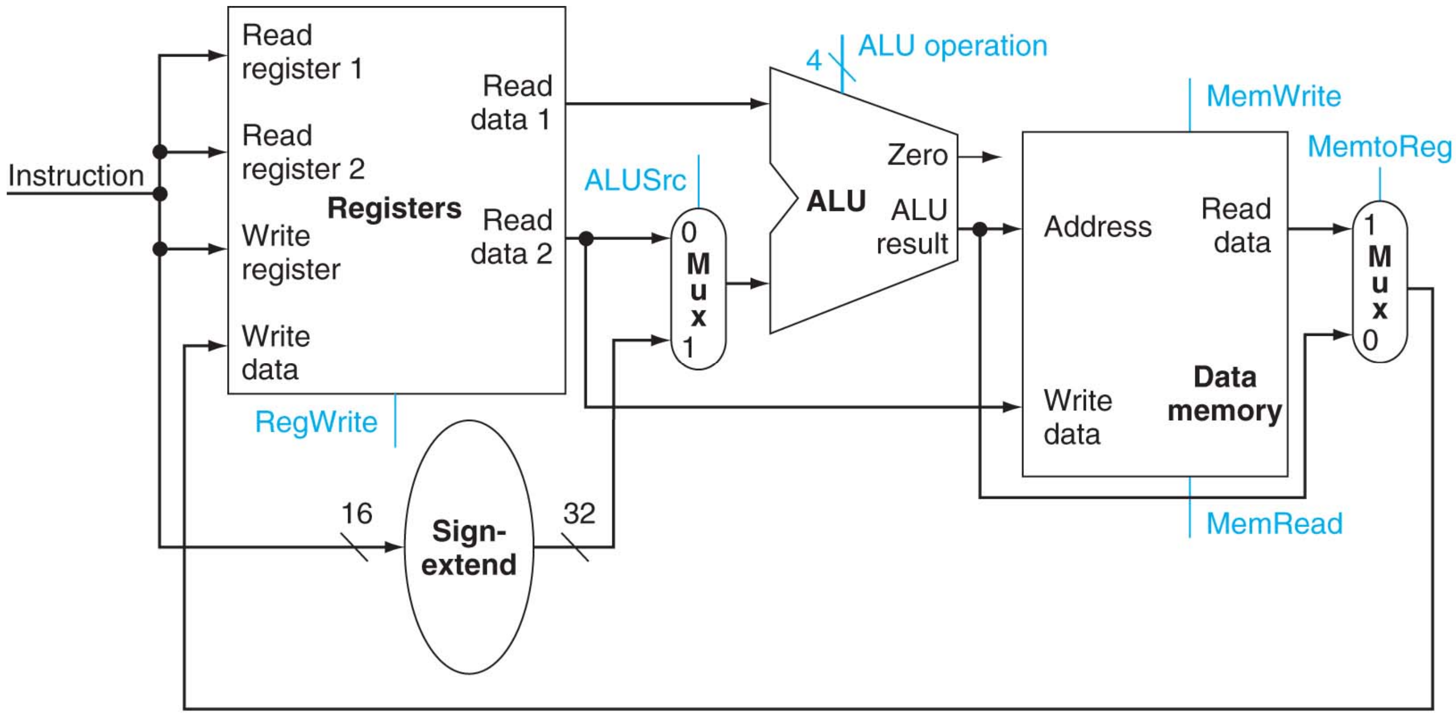



Figure 4.10

# Simple Datapath for MIPS Architecture

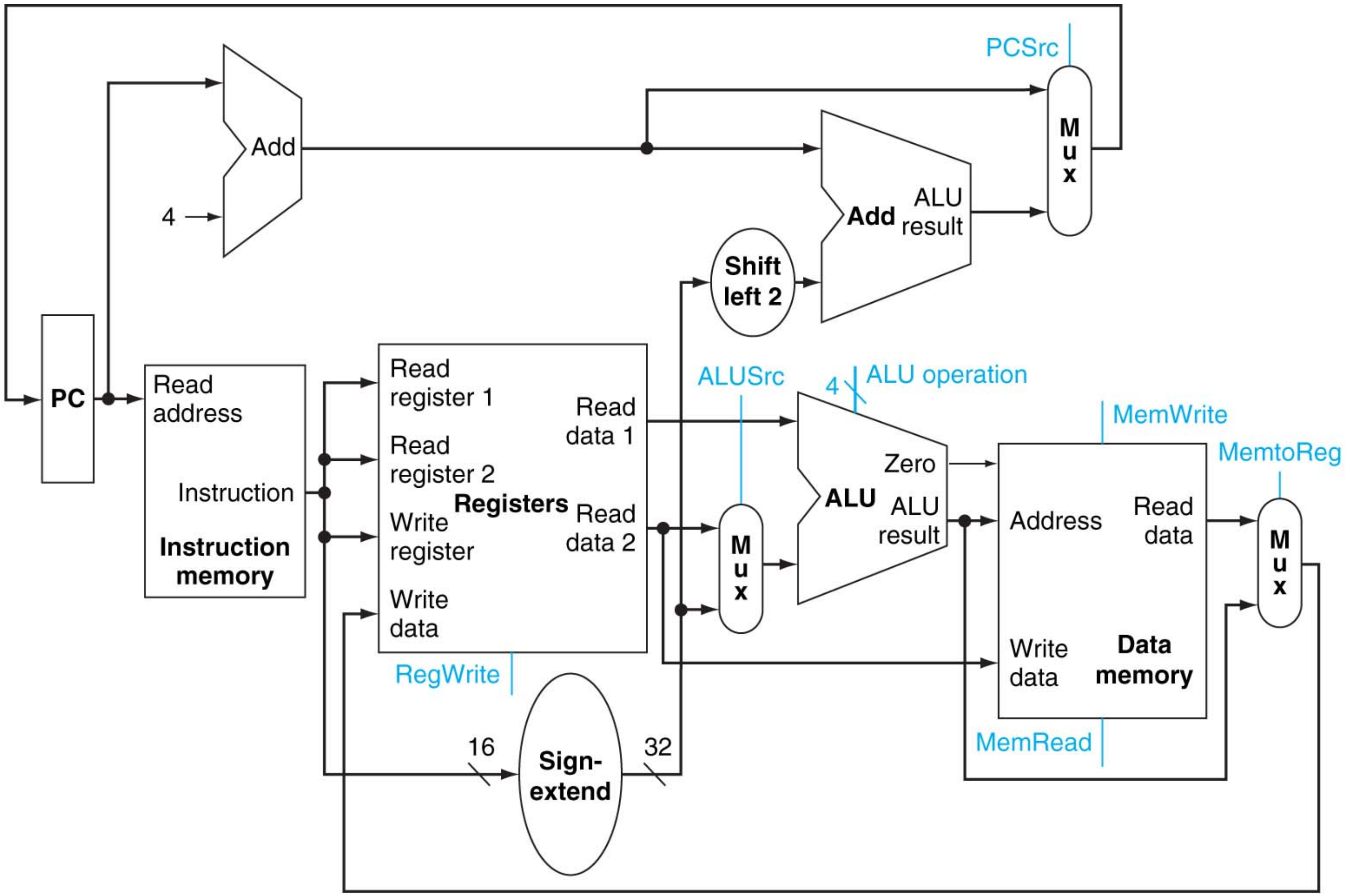



Figure 4.11

# 4.4 A Simple Implementation Scheme

## The ALU Control

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

| Instruction opcode | ALU Op | Instruction operation | Function field | Desired ALU action | ALU operation |
|---|---|---|---|---|---|
| lw | 00 | load word | XXXXXX | add | 0010 |
| sw | 00 | store word | XXXXXX | add | 0010 |
| beq | 01 | branch on 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 |

Figure 4.12

# Main Control and ALU Control

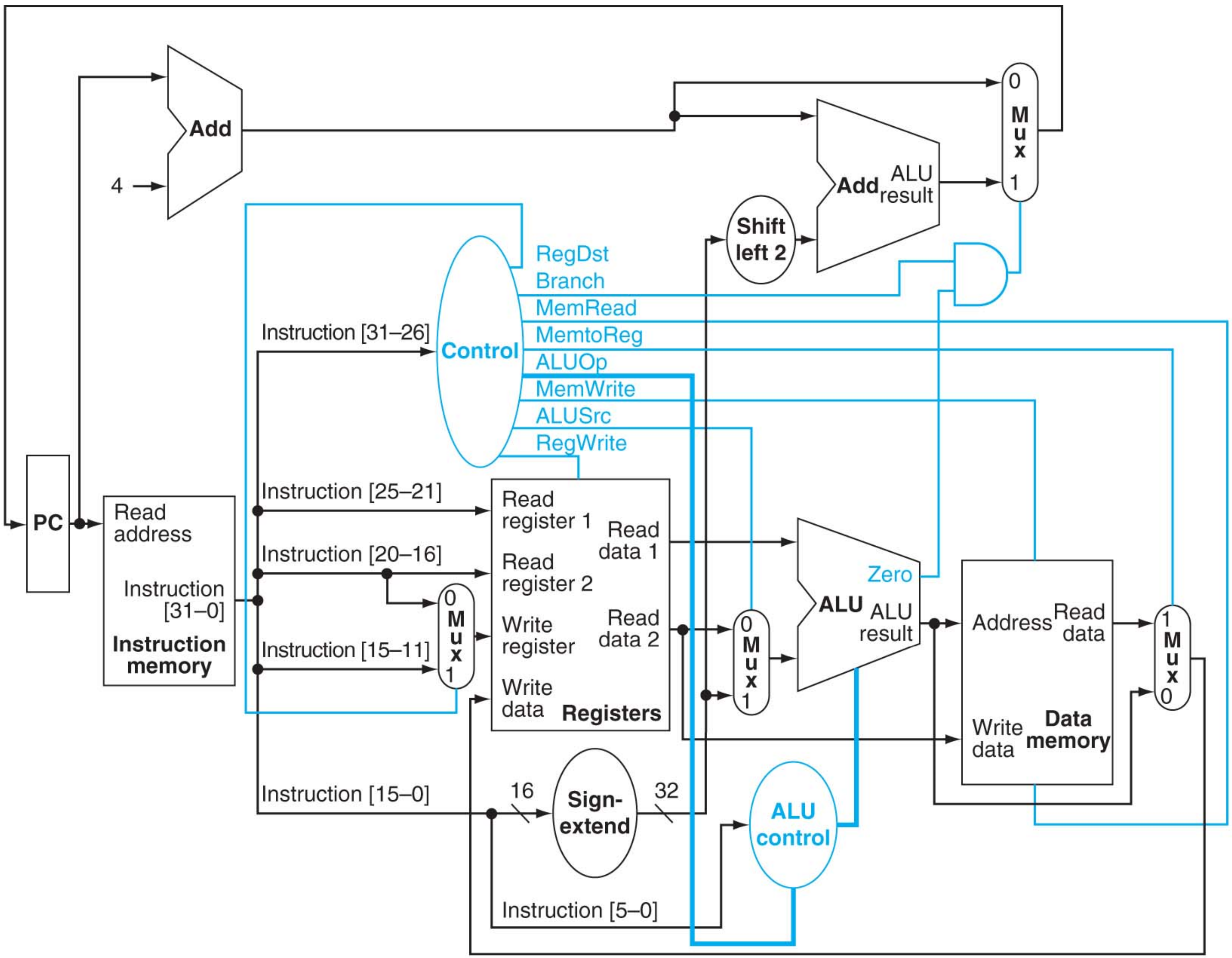



Figure 4.17

# Truth Table for the ALU Control Bits

| ALUOp | | Function field | | | | | | ALU 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 |

Figure 4.13

# Implementation of ALU Control

**Operation3 = 0**

**Operation2 = ALUOp0 + ALUOp1 · F1**

**Operation1 = ALUOp1' + F2'**

**Operation0 = ALUOp1 · (F0 + F3)**

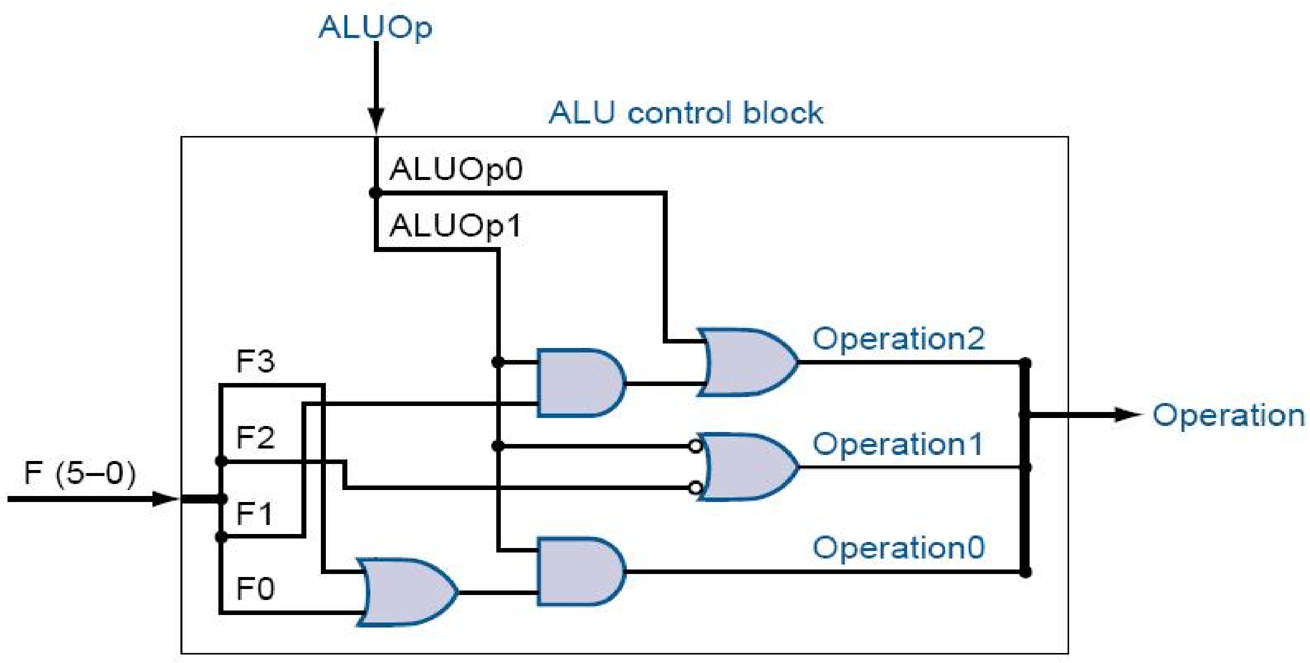



Figure C.2.3