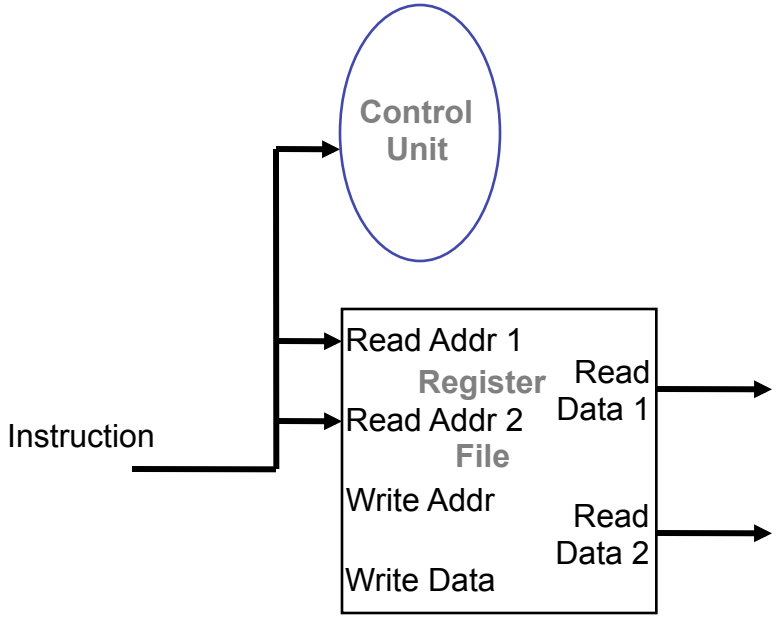
Decoding Instructions

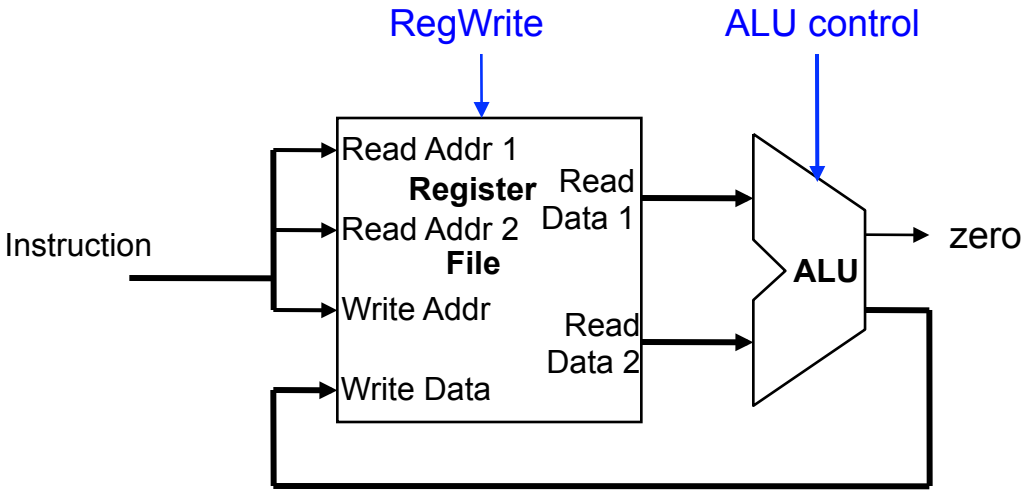
- Decoding instructions involves
  - Sending the instruction's fields to the control unit



  - Reading two values from the Register File

Executing R-Type Instructions

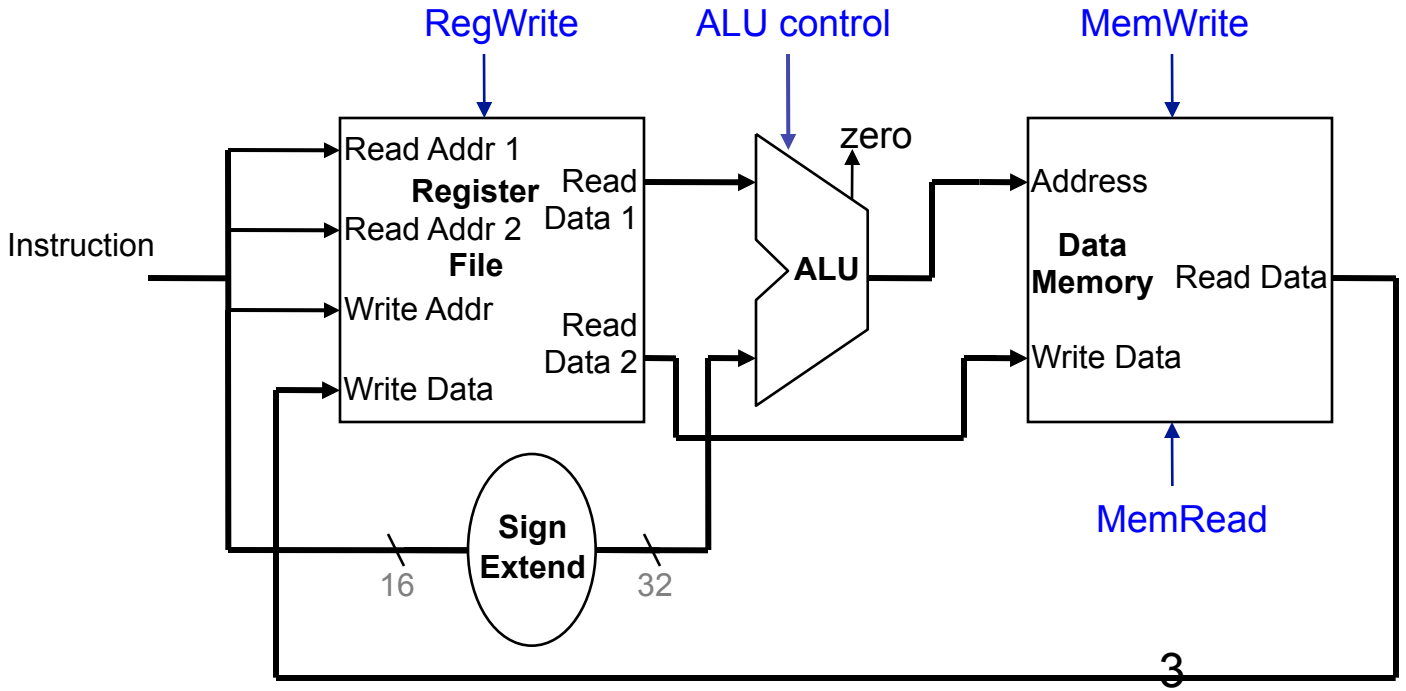
- R-type instructions (`add, sub, slt, and, or`)
  - Perform the operation on values in rs and rt
  - Store the result back into the Register File (rd)



  - The Register File is not written every cycle, so we need an explicit write control signal for the Register File

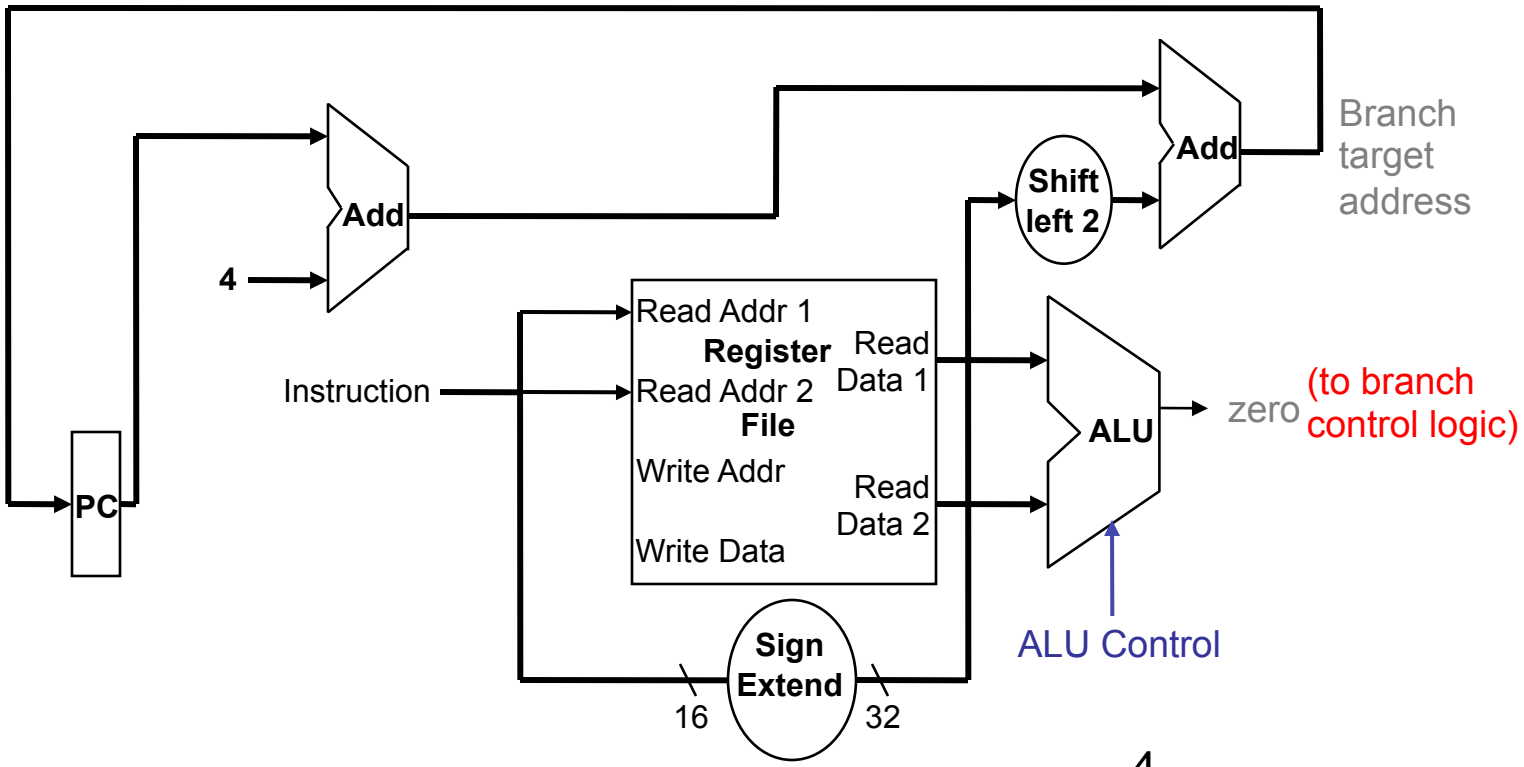
Executing Load and Store Operations

- Load and store operations involve
  - Compute memory address (base register plus the 16-bit signed-extended offset field in the instruction)
  - Store value to the Data Memory
  - Load value to the Register File



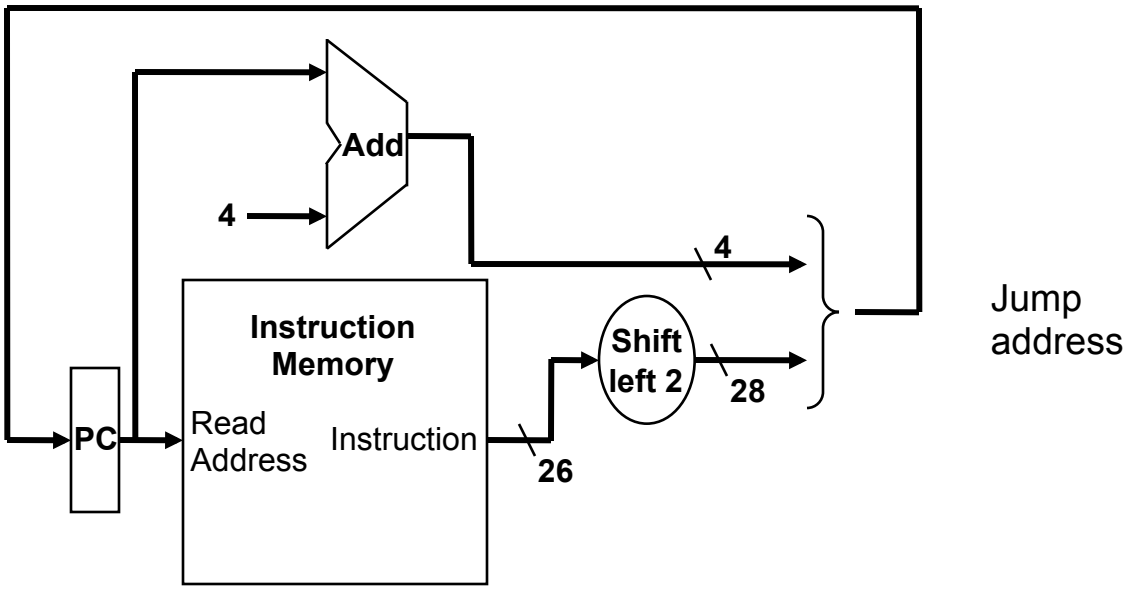
Executing Branch Operations

- Branch operations involve
  - Compare the operands for equality (**`zero`**)
  - Compute the branch target address (PC plus the 16-bit signed-extended offset field in the instruction)



Executing Jump Operations

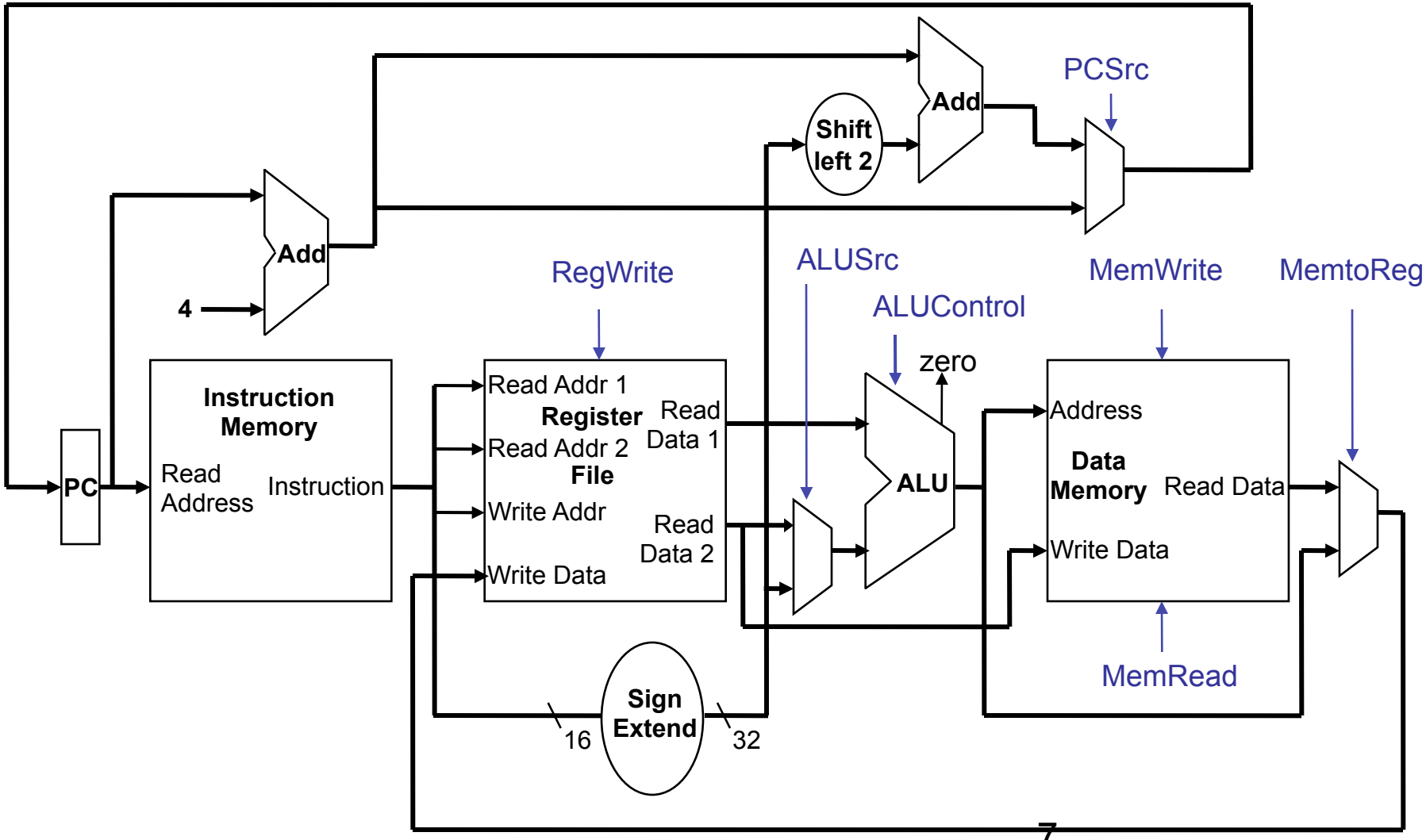
- Jump operation involves
  - Replace the lower 28 bits of the PC with the lower 26 bits of the instruction shifted left by 2 bits



Creating a Single Datapath

- Assemble the datapath segments and add control lines and multiplexers as needed
- Single cycle design – fetch, decode and execute each instructions in one clock cycle
  - No resource can be used more than once per instruction
    - Separate Instruction Memory and Data Memory
    - Multiple, parallel adders
  - Multiplexers needed at the input of shared elements
  - Write control signals for the Register File and Data Memory
- Cycle time is determined by length of the longest path
  - The “critical path”

The Complete Datapath



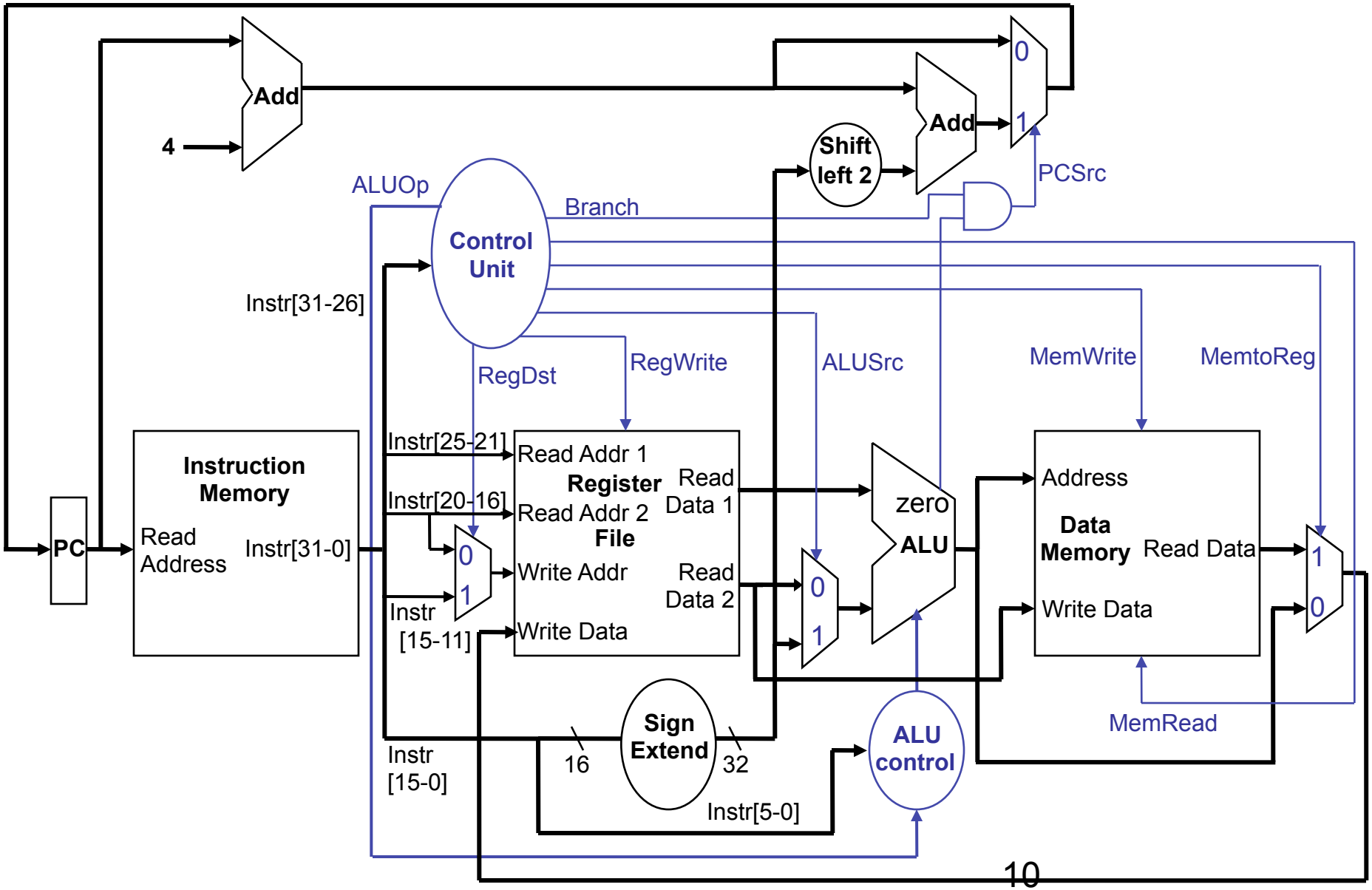
The Processor: Datapath & Control

- Our implementation of the MIPS is simplified
  - Memory-reference instructions: **`lw, sw`**
  - Arithmetic/Logical instructions: **`add, sub, and, or, slt`**
  - Control flow instructions: **`beq, j`**
- Generic implementation
  - Use the program counter (PC) to supply instruction address and fetch the instruction from memory (and update the PC)
  - Decode the instruction (and read registers)
  - Execute the instruction
- All instructions (except **`j`**) use the ALU

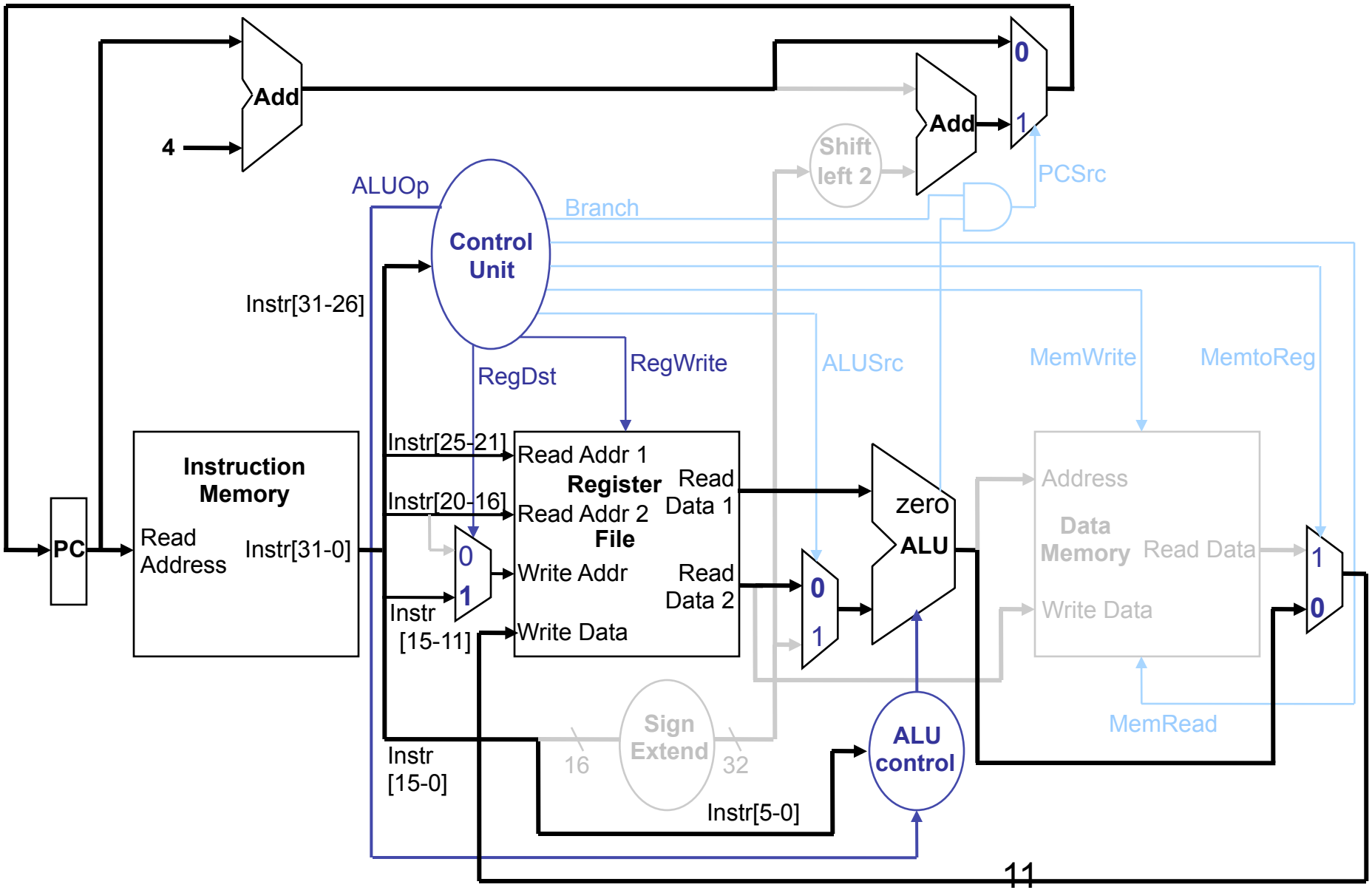
Adding the Control

- Selecting the operations to perform (ALU, Register File and Memory read/write)
- Controlling the flow of data (multiplexer inputs)
- Observations
  - op field always in bits 31-26
  - Address of registers to be read are always specified by the rs field (bits 25-21) and rt field (bits 20-16)
  - Address of register to be written is in one of two places
    - rt (bits 20-16) for I-type instructions
    - rd (bits 15-11) for R-type instructions
  - Offset for beq, lw, and sw always in bits 15-0

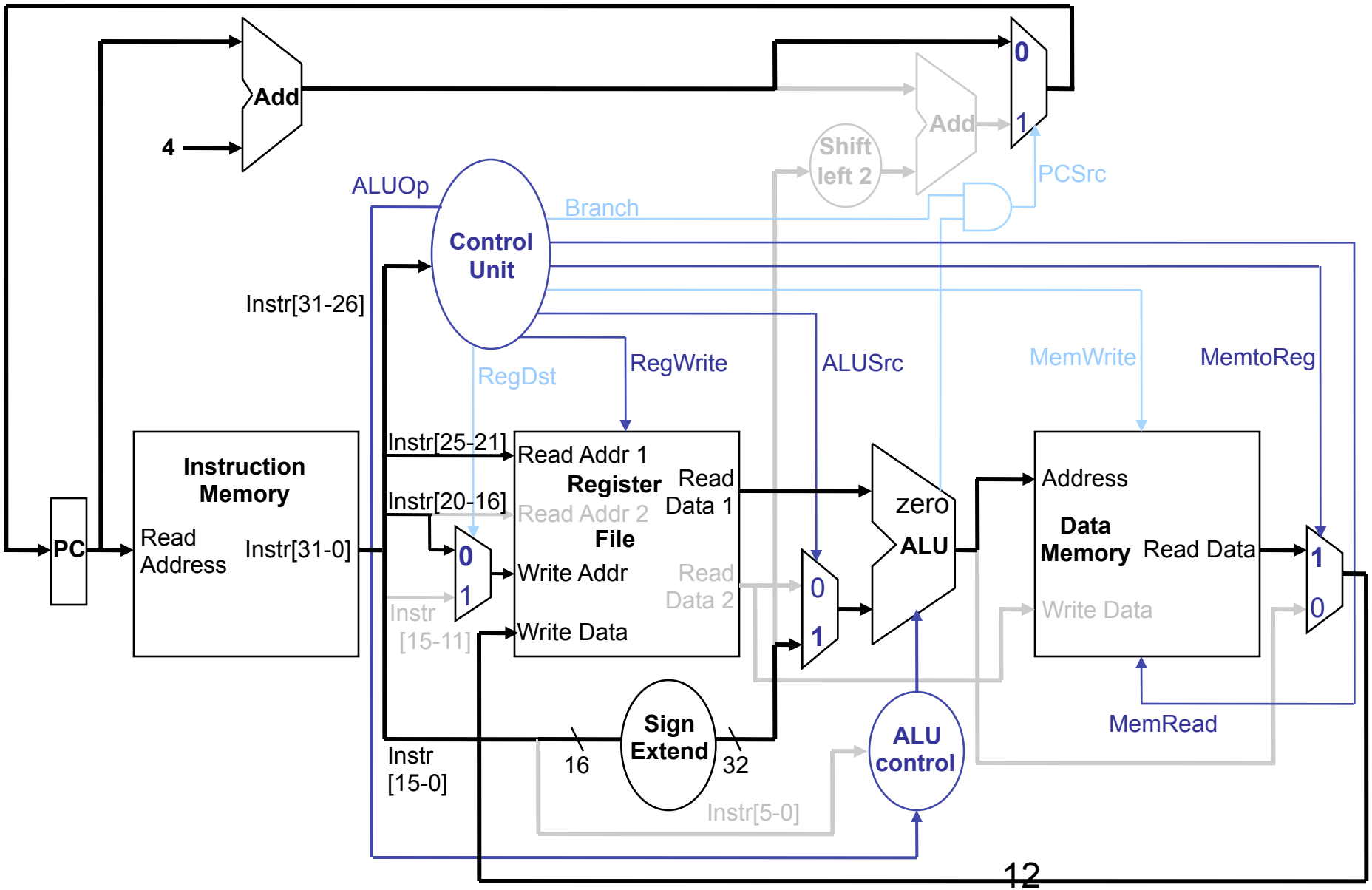
Single Cycle Datapath and Control



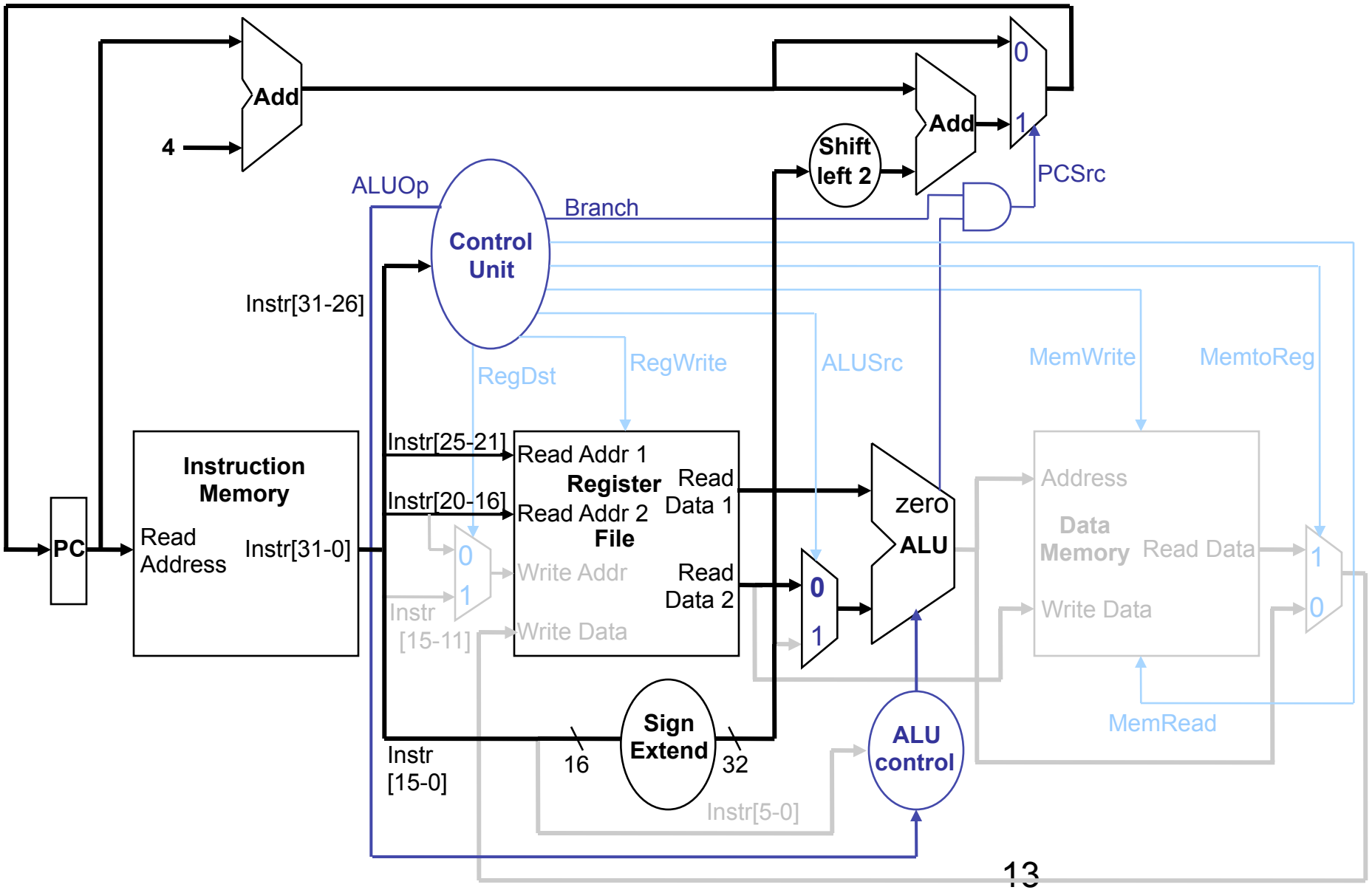
R-Type Instructions



Load Instructions

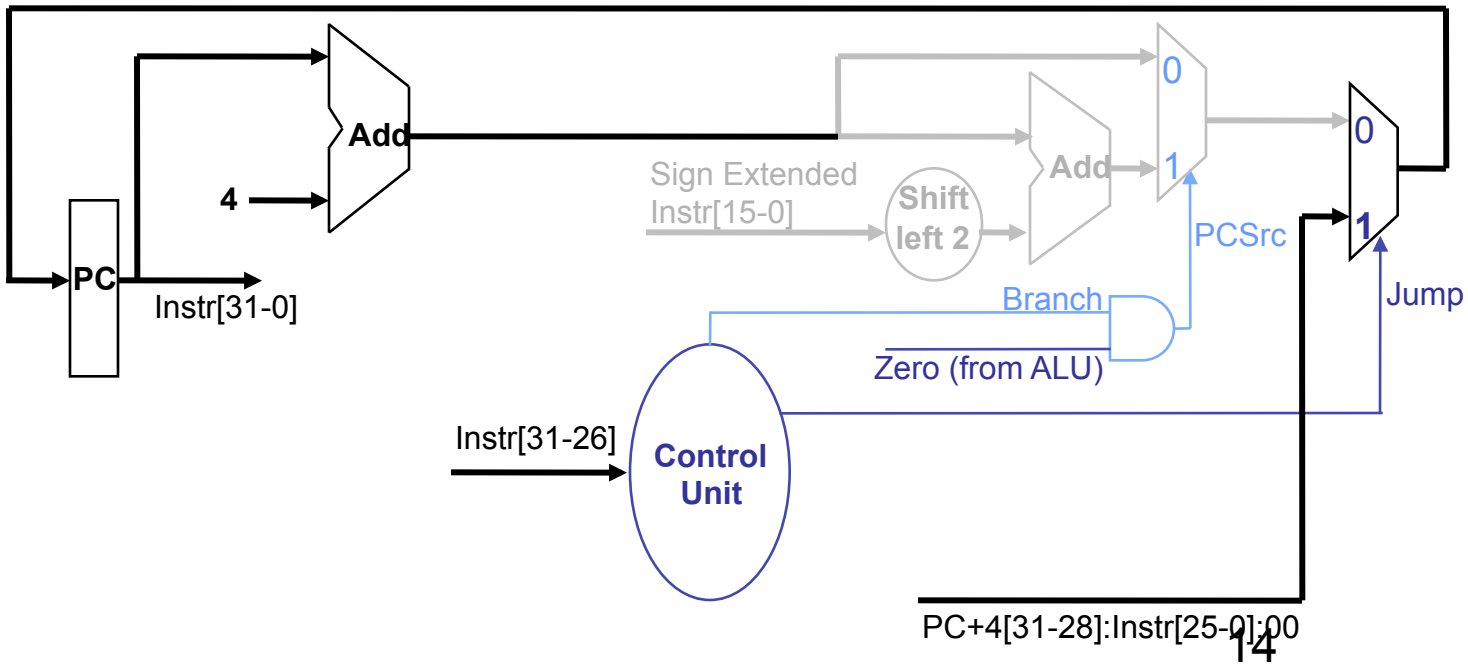


Branch on Equal Instruction



Adding the Jump Instruction

- Additional multiplexer and control signal
- Jump address
  - 4 Most significant bits from PC + 4
  - 26 bits from instruction's address field
  - 2 zero bits



Single Cycle Characteristics