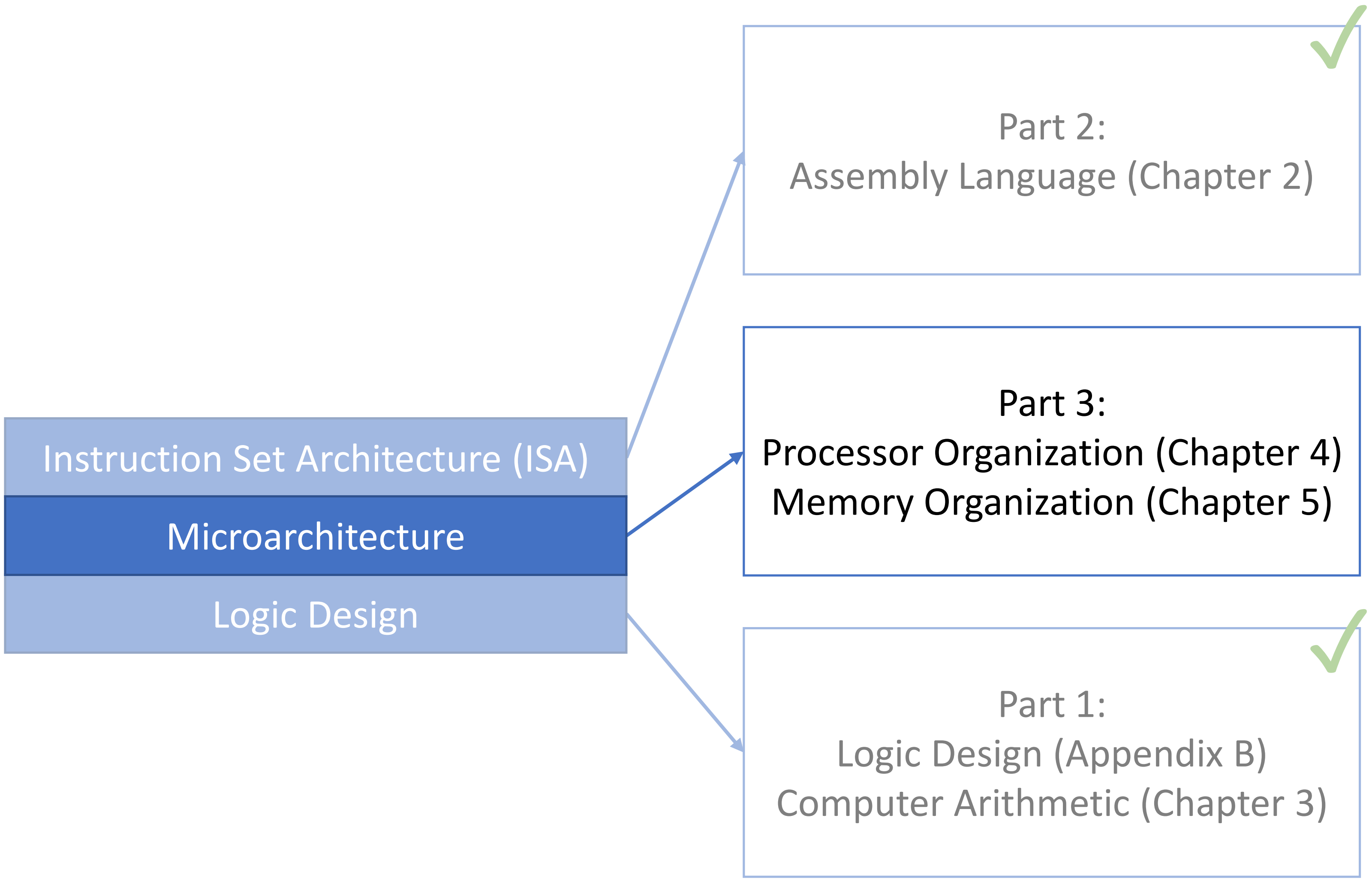
Lecture 21: Building a Datapath

CMPS 221 – Computer Organization and Design

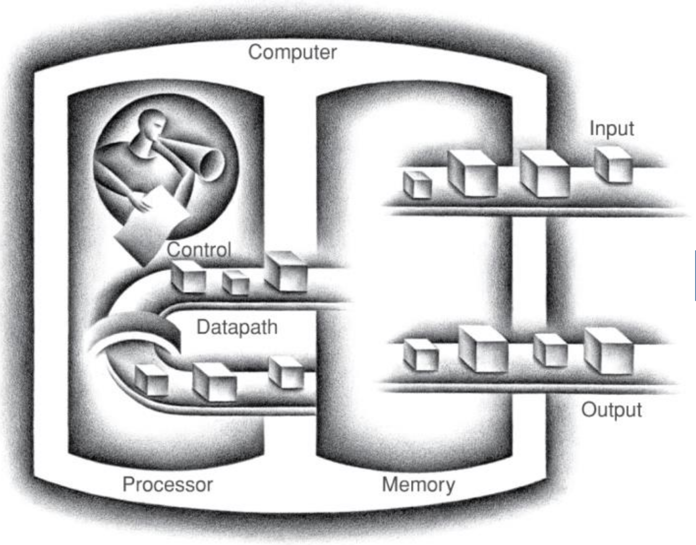
Adapted by Izzat El Hajj from textbook slides for Computer Organization and Design, by David Patterson and John Hennessy, Morgan Kaufmann.

Part 3: Microarchitecture


Instruction Set Architecture (ISA)
Microarchitecture
Logic Design
✓
Part 2:
Assembly Language (Chapter 2)
Part 3:
Processor Organization (Chapter 4)
Memory Organization (Chapter 5)
✓
Part 1:
Logic Design (Appendix B)
Computer Arithmetic (Chapter 3)

Microarchitecture

Adder	Multiplier
Multiplexer	Register File
Memory	Controller



```
sll $2, $5, 2
add $2, $4, $2
lw  $15, 0($2)
lw  $16, 4($2)
sw  $16, 0($2)
sw  $15, 4($2)
jr  $31
```

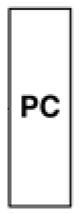
Introduction

- CPU performance factors
  - Instruction count
    - Determined by ISA and compiler
  - CPI and Cycle time
    - Determined by CPU hardware
- We will examine two MIPS implementations
  - A simple version
  - A more realistic pipelined version
- Focus on a simple subset (shows most aspects)
  - Arithmetic/logical: `add`, `addi`
  - Memory reference: `lw`, `sw`
  - Control transfer: `beq`, `j`

Building a Datapath

- A **datapath** is a set of elements that process data and addresses in the CPU
- Components of a MIPS datapath:
  - PC register, instruction memory (fetch instructions)
  - Register file (read/write registers)
  - ALUs:
    - Arithmetic/logic operations
    - Memory address for load/store
    - Branch target address
  - Data memory (load/store data)

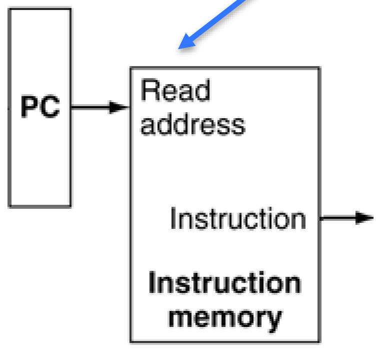
Instruction Fetch



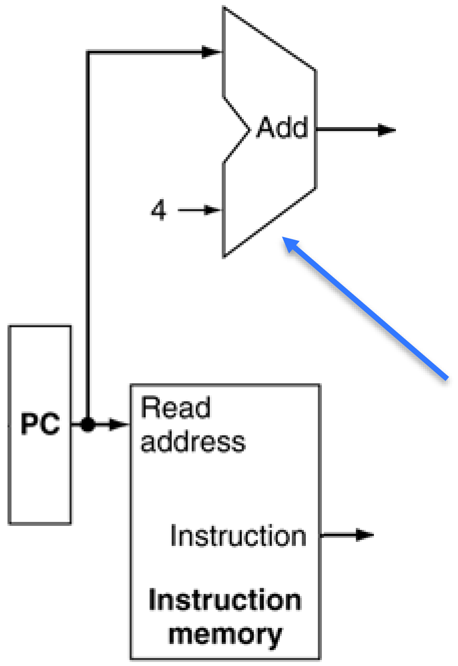
PC register points to the current instruction to be executed

Instruction Fetch

Use PC to fetch the instruction from memory

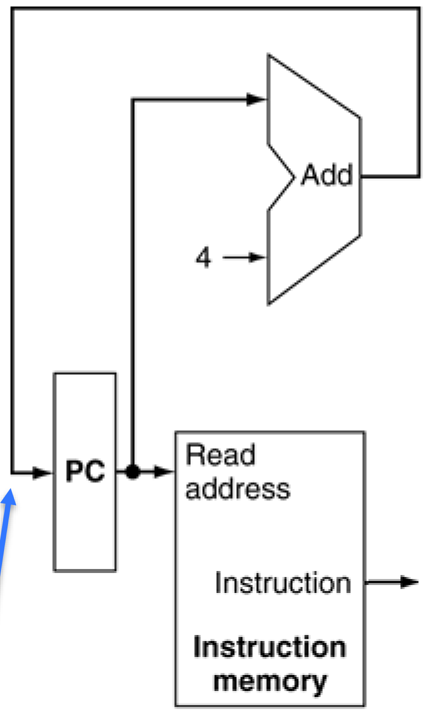


Instruction Fetch



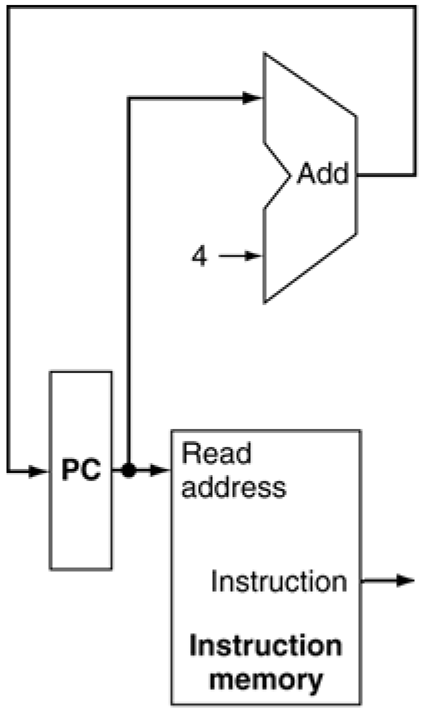
Increment PC by 4 to get the address of the next instruction

Instruction Fetch



Update the PC register at the end of the cycle to fetch the next instruction on the next cycle

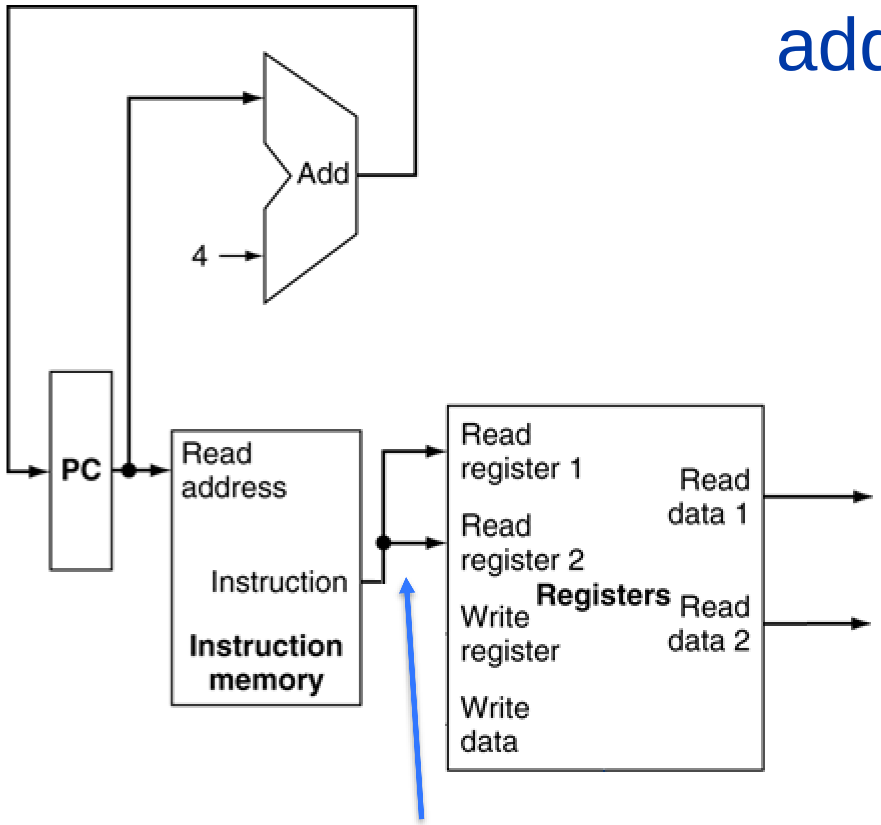
R-type Instructions



add $t1, $a0, $s2

R-type Instructions

add $t1, $a0, $s2

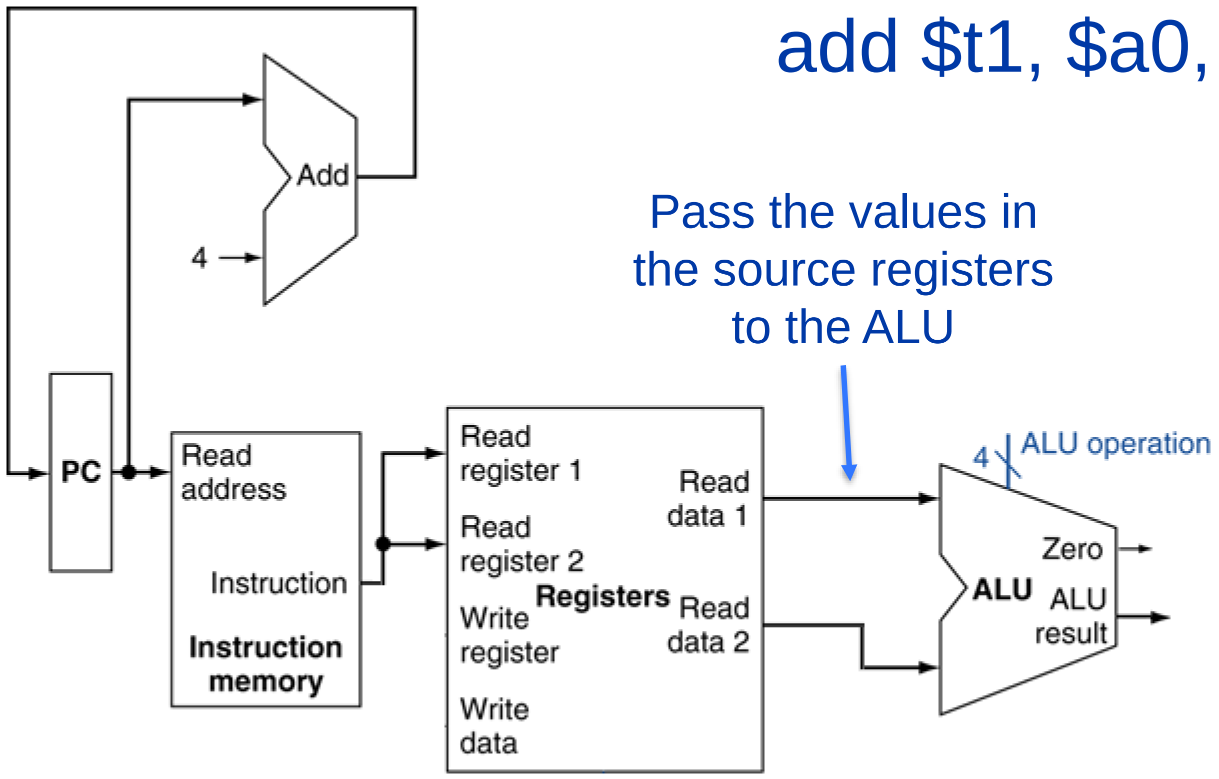


Use the source register numbers to read the source registers from the register file

R-type Instructions

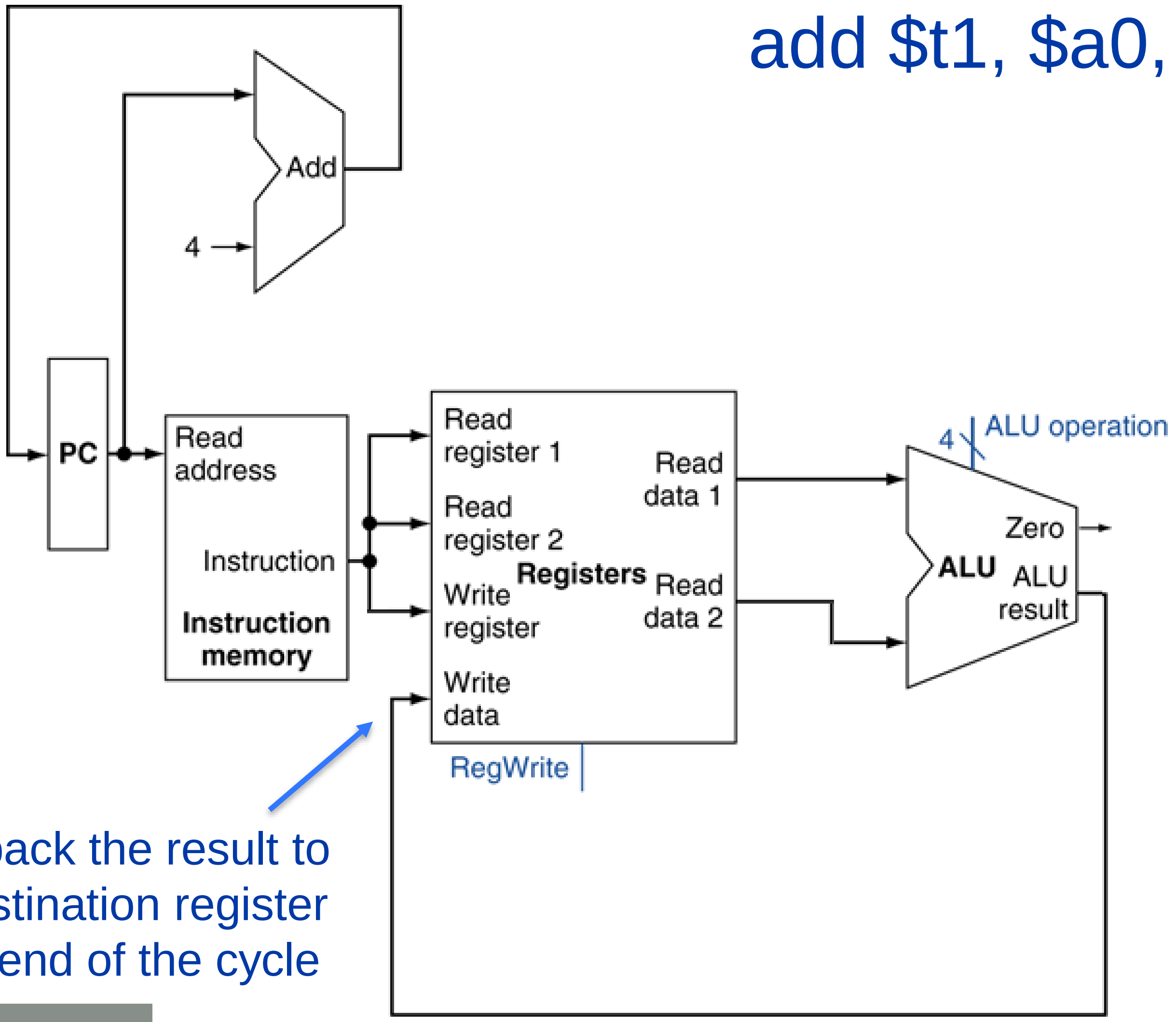
add $t1, $a0, $s2

Pass the values in the source registers to the ALU



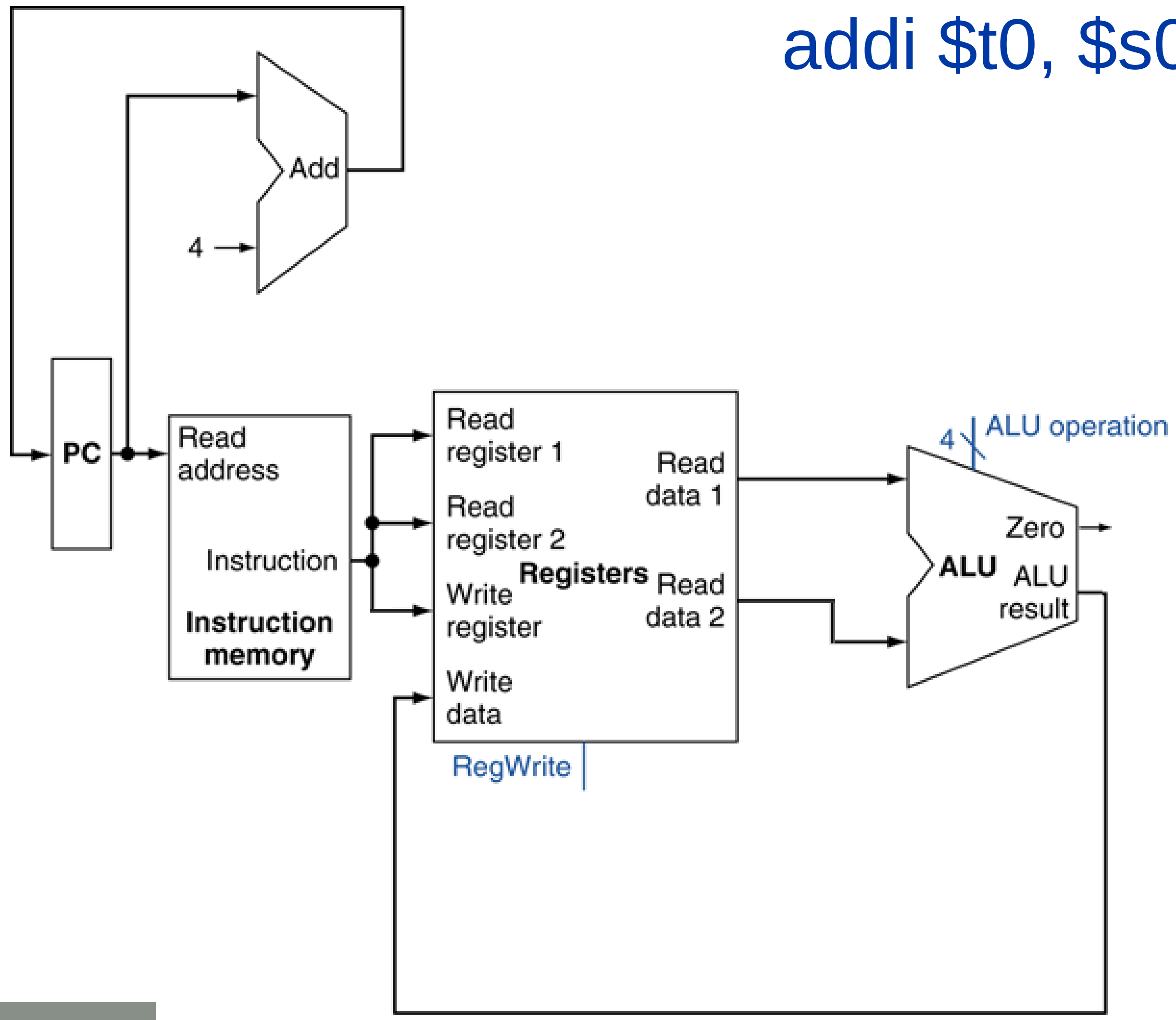
R-type Instructions

add $t1, $a0, $s2



Write back the result to the destination register at the end of the cycle

Immediate Instructions

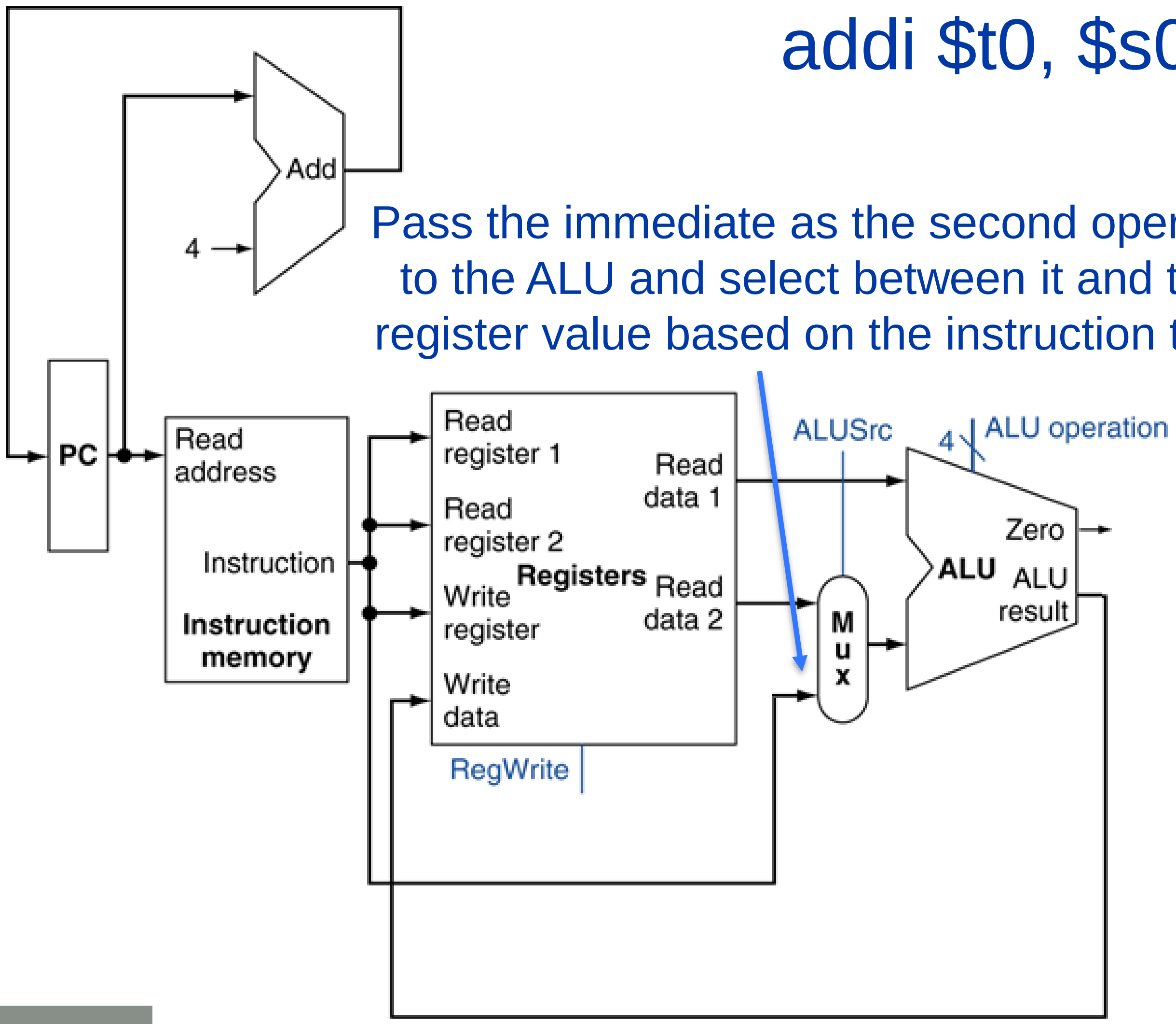


addi $t0, $s0, 4

Immediate Instructions

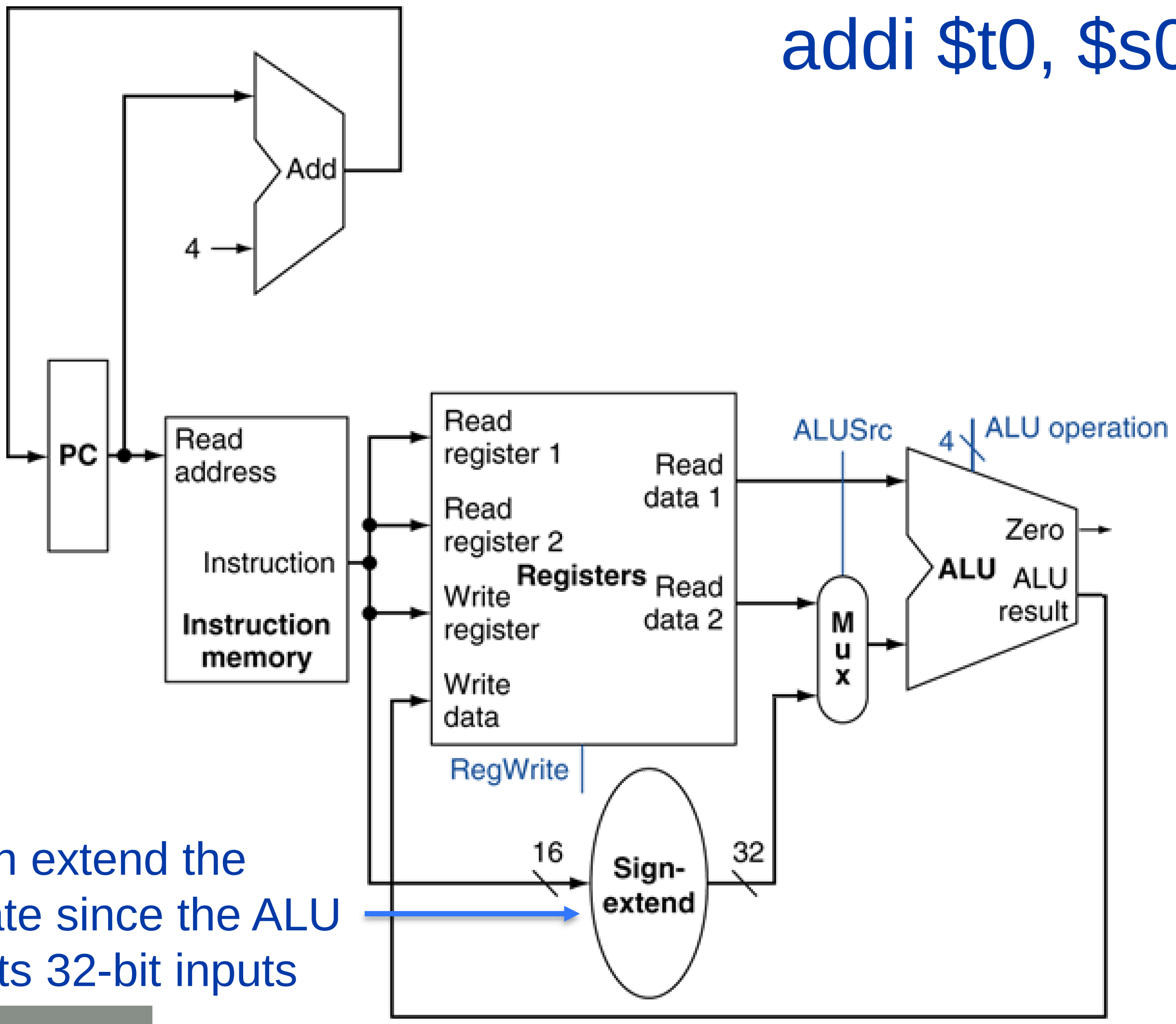
addi $t0, $s0, 4

Pass the immediate as the second operand to the ALU and select between it and the register value based on the instruction type



Immediate Instructions

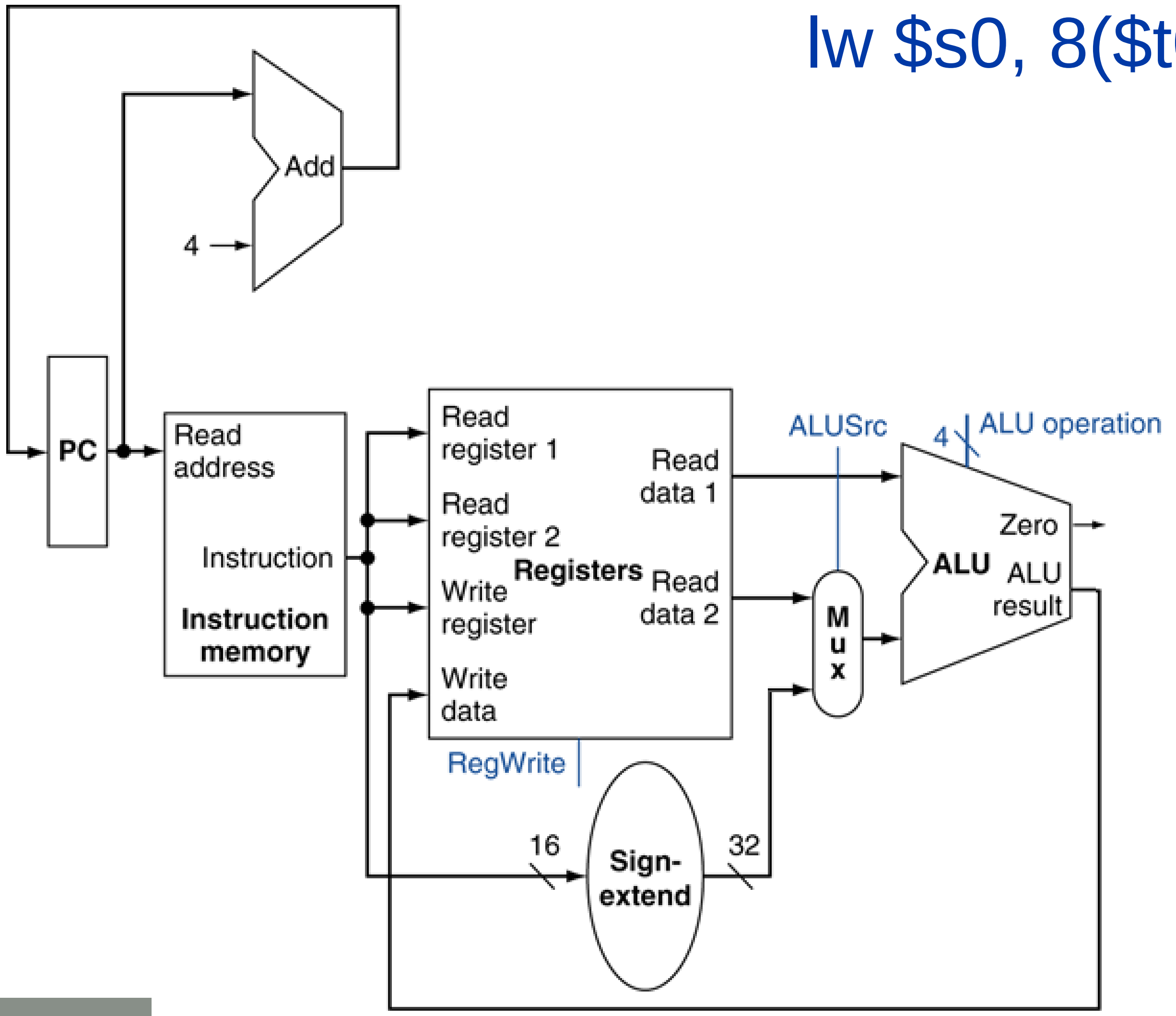
addi $t0, $s0, 4



Sign extend the immediate since the ALU expects 32-bit inputs

Load Instructions

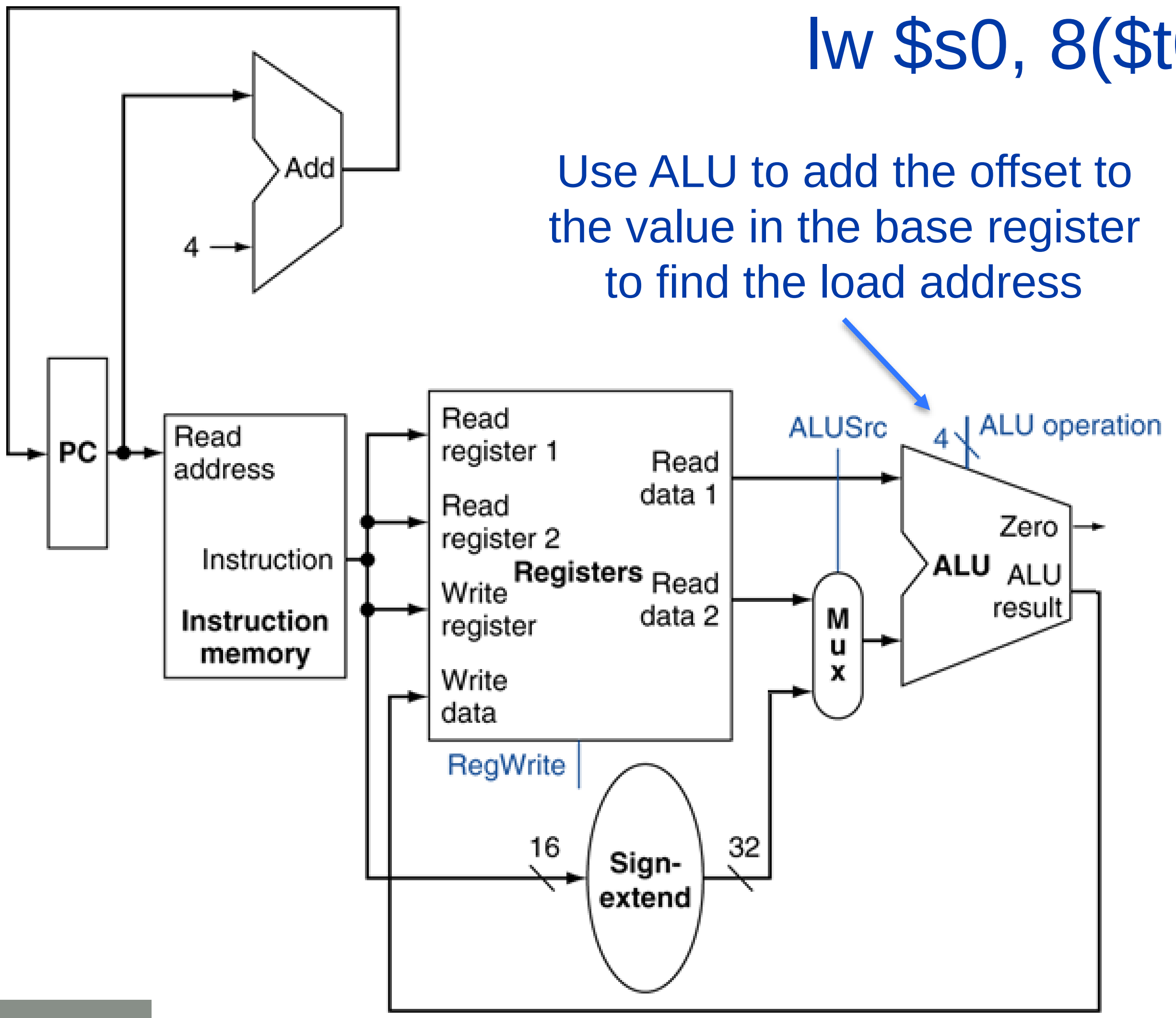
lw $s0, 8($t0)



Load Instructions

lw $s0, 8($t0)

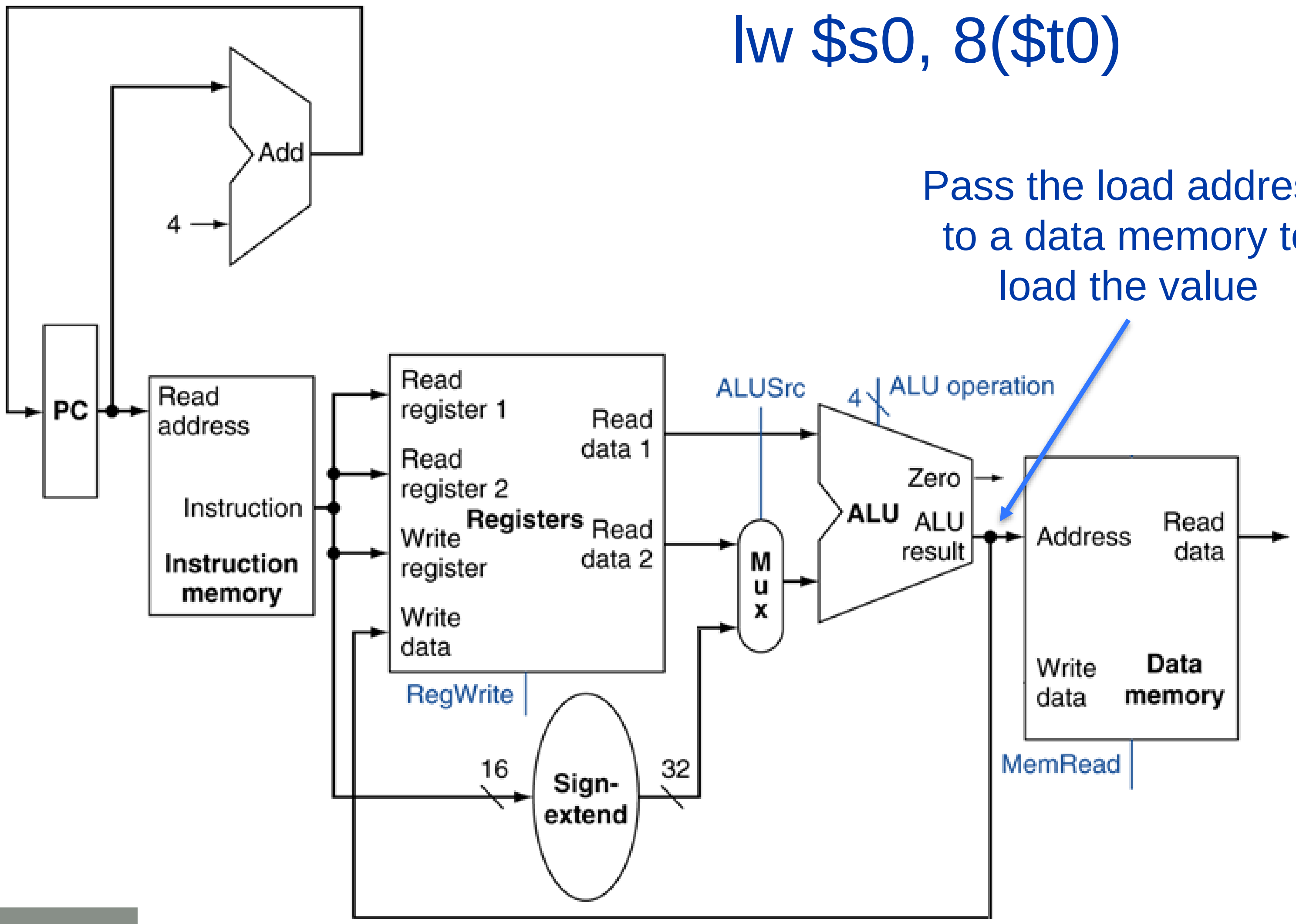
Use ALU to add the offset to the value in the base register to find the load address



Load Instructions

lw $s0, 8($t0)

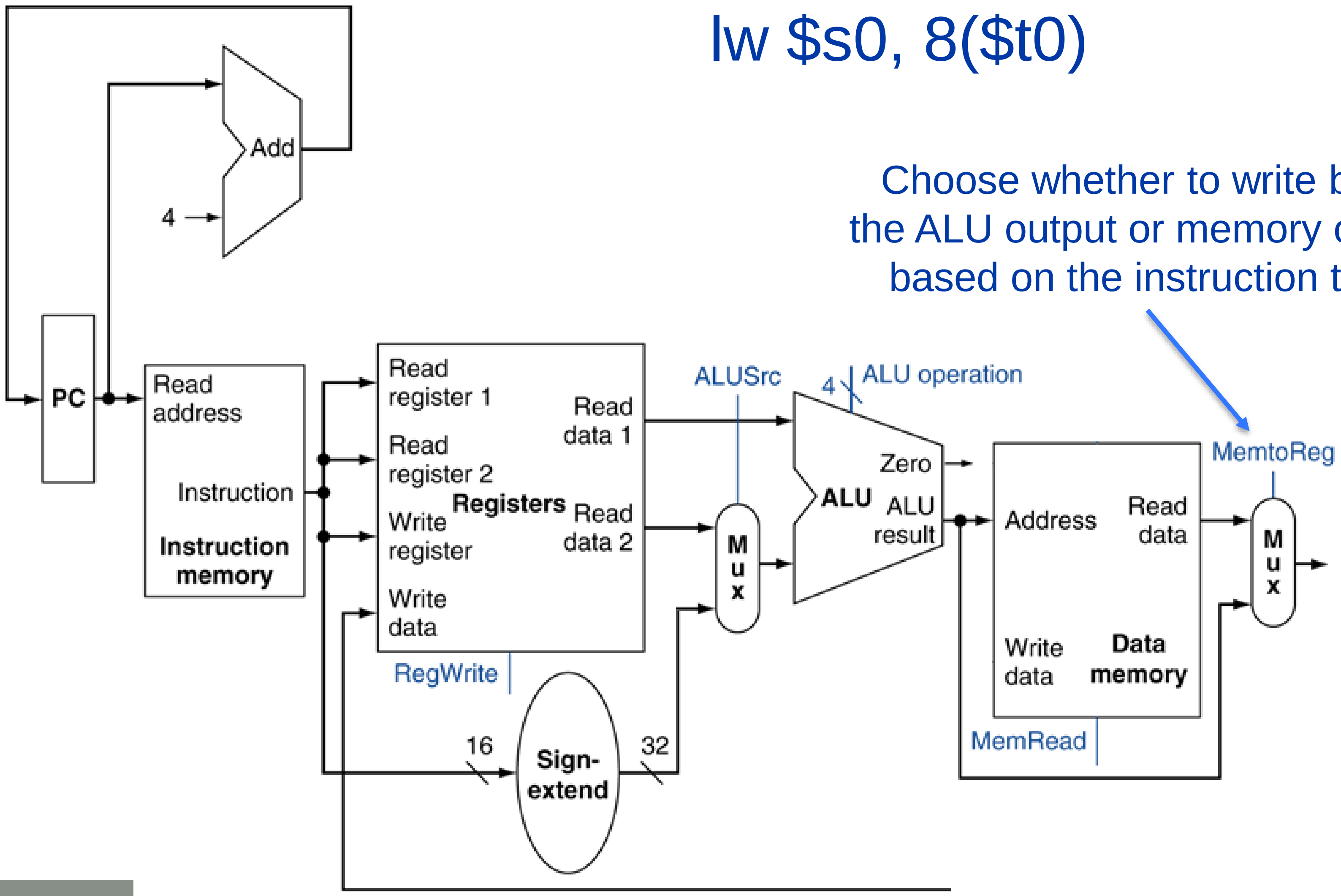
Pass the load address to a data memory to load the value



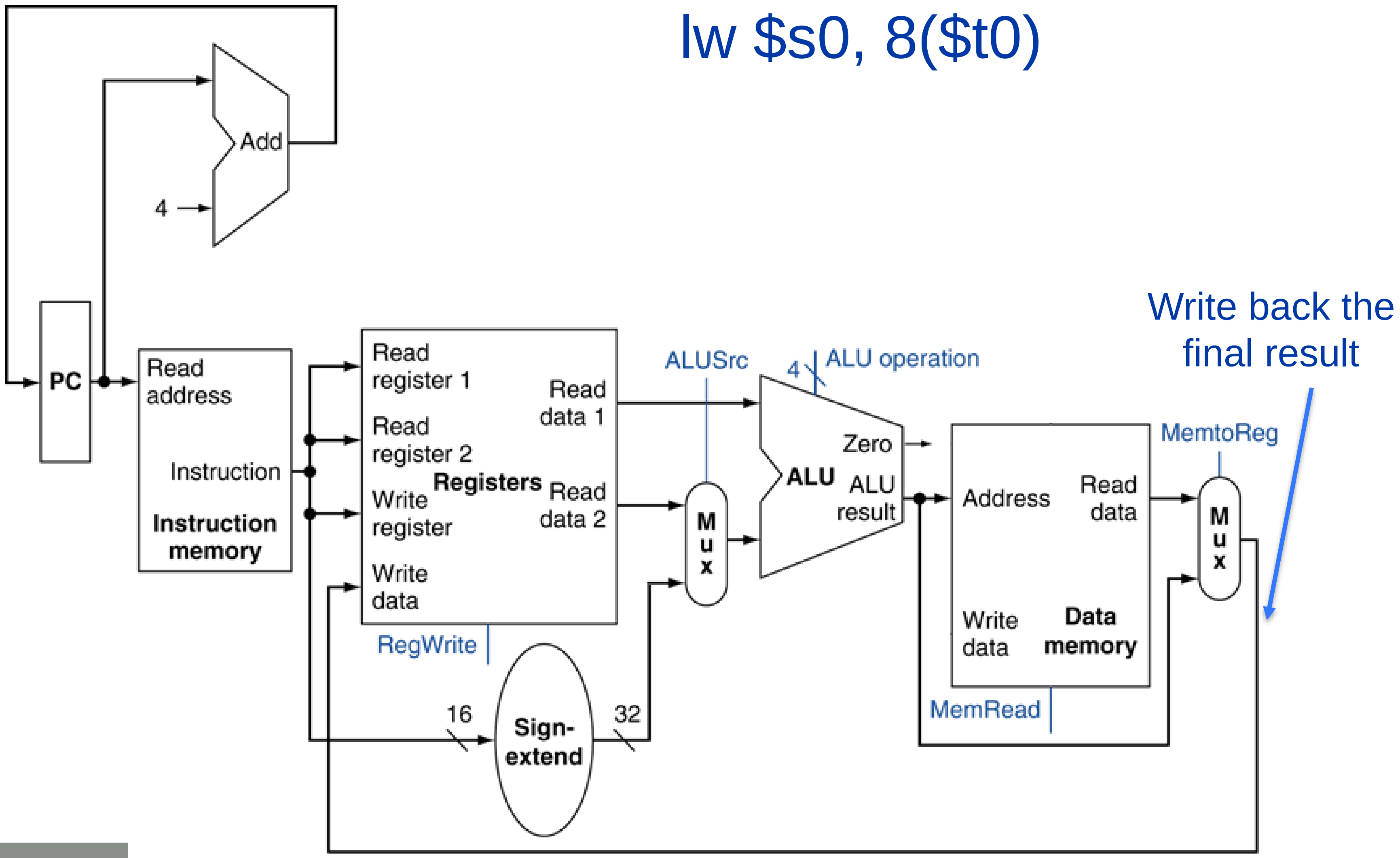
Load Instructions

lw $s0, 8($t0)

Choose whether to write back the ALU output or memory output based on the instruction type

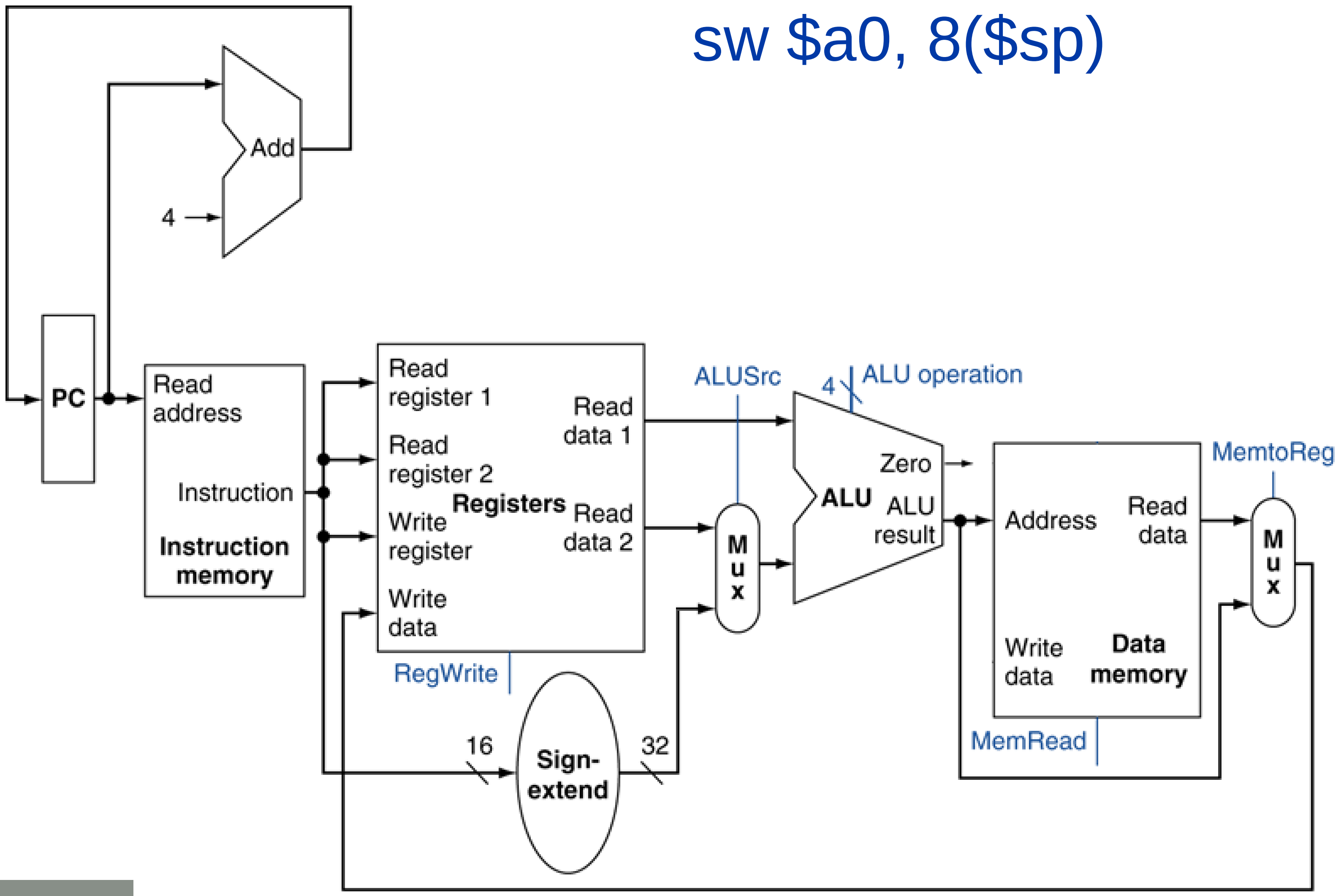


Load Instructions



Store Instructions

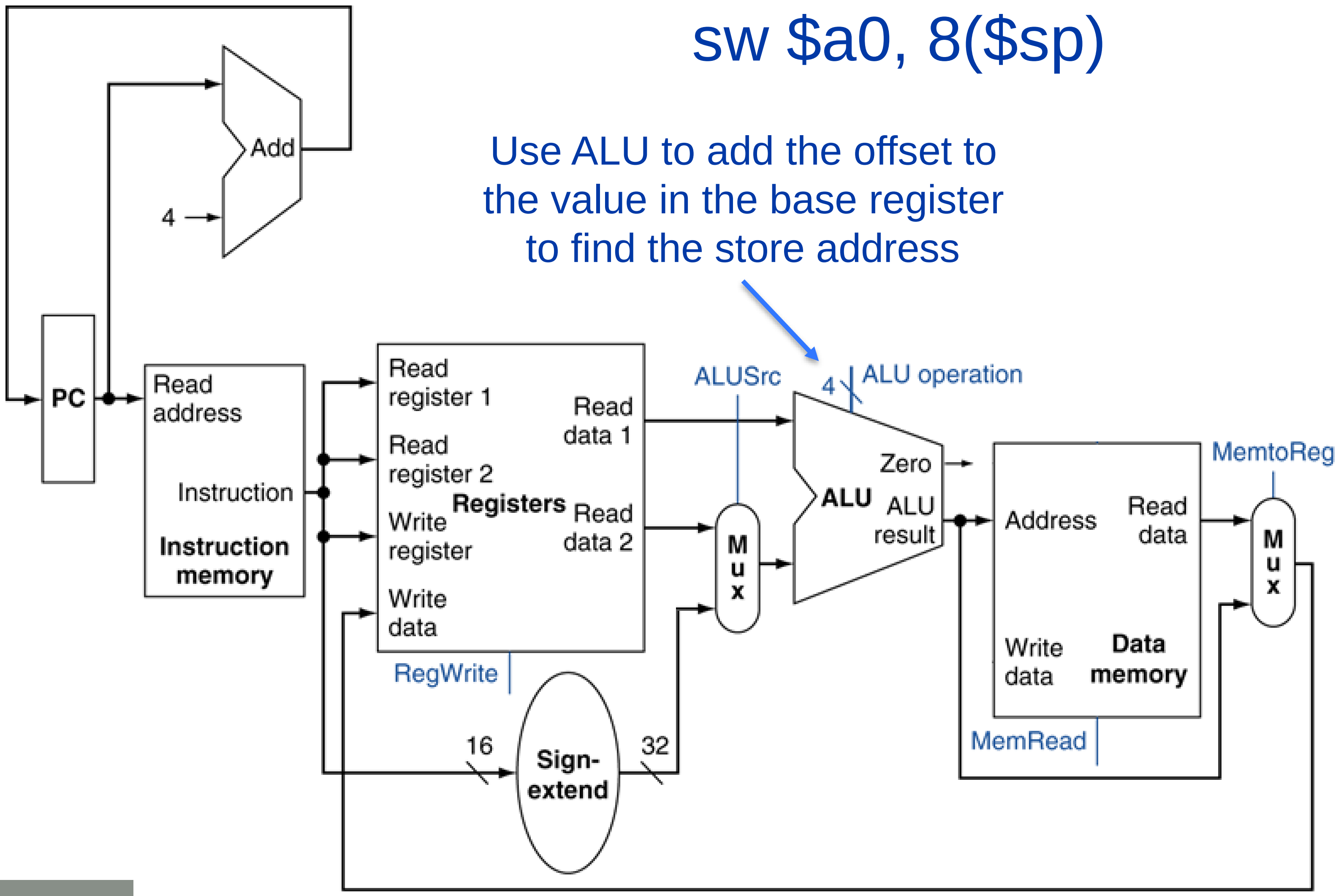
sw $a0, 8($sp)



Store Instructions

sw $a0, 8($sp)

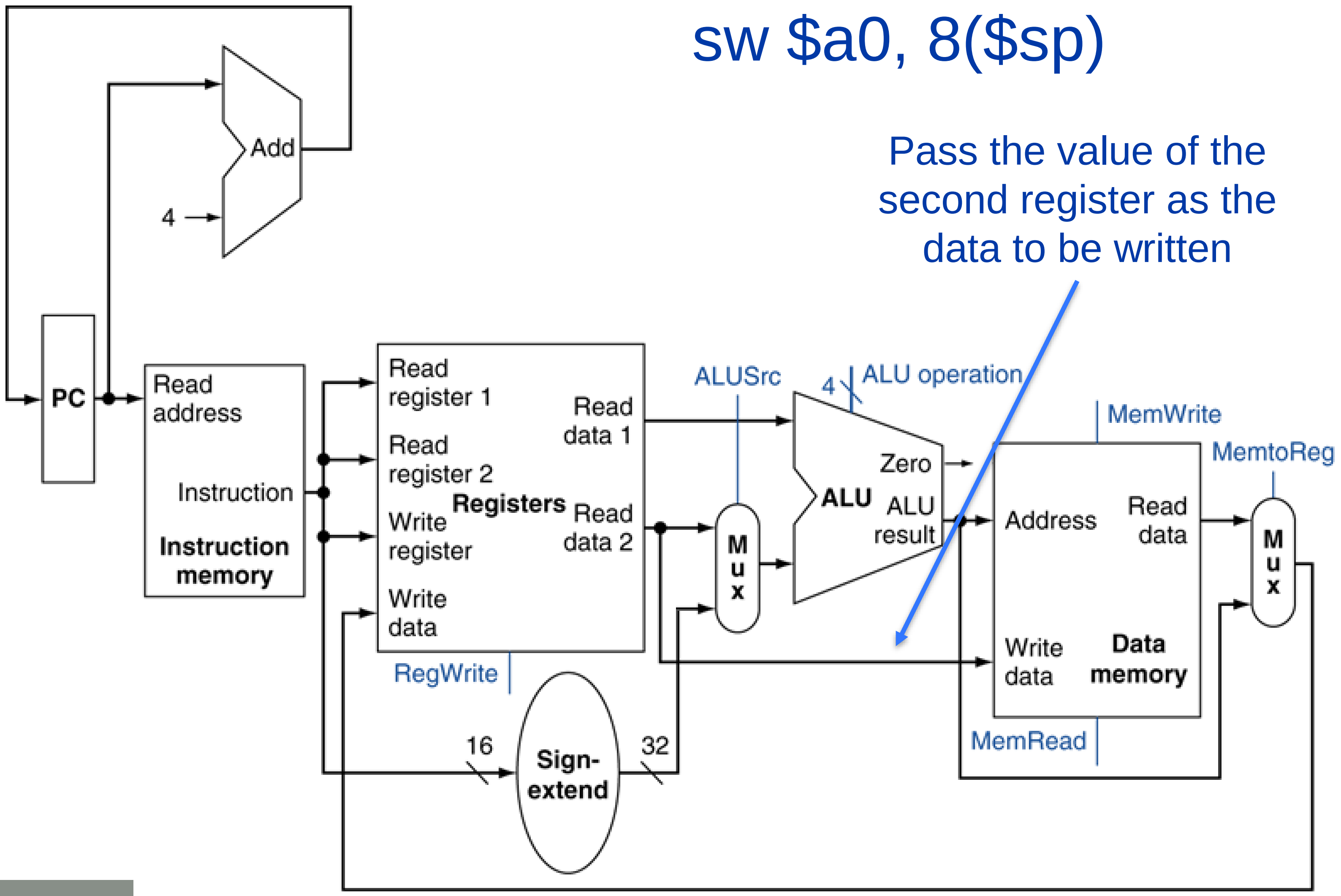
Use ALU to add the offset to the value in the base register to find the store address



Store Instructions

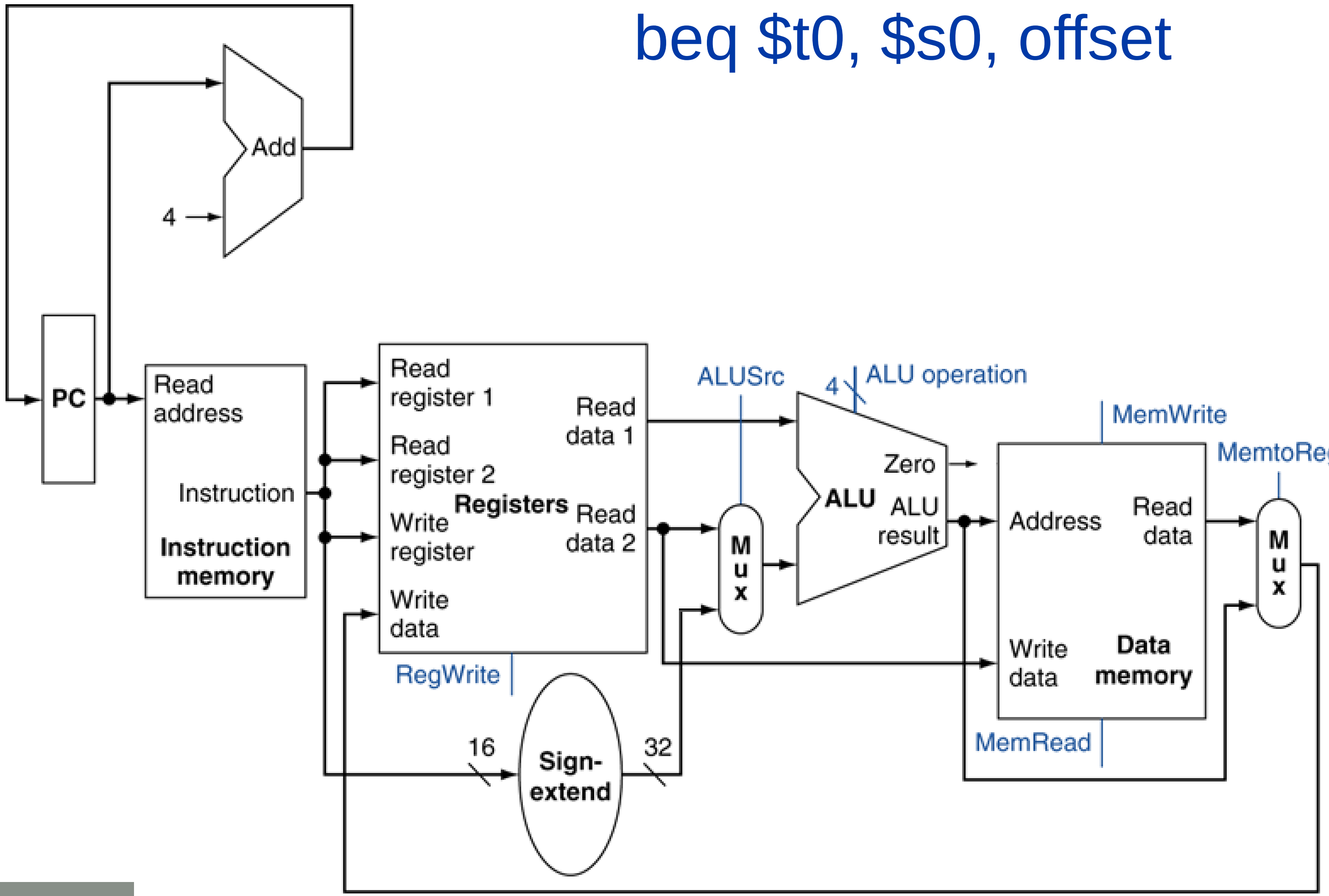
sw $a0, 8($sp)

Pass the value of the second register as the data to be written



Branch Instructions

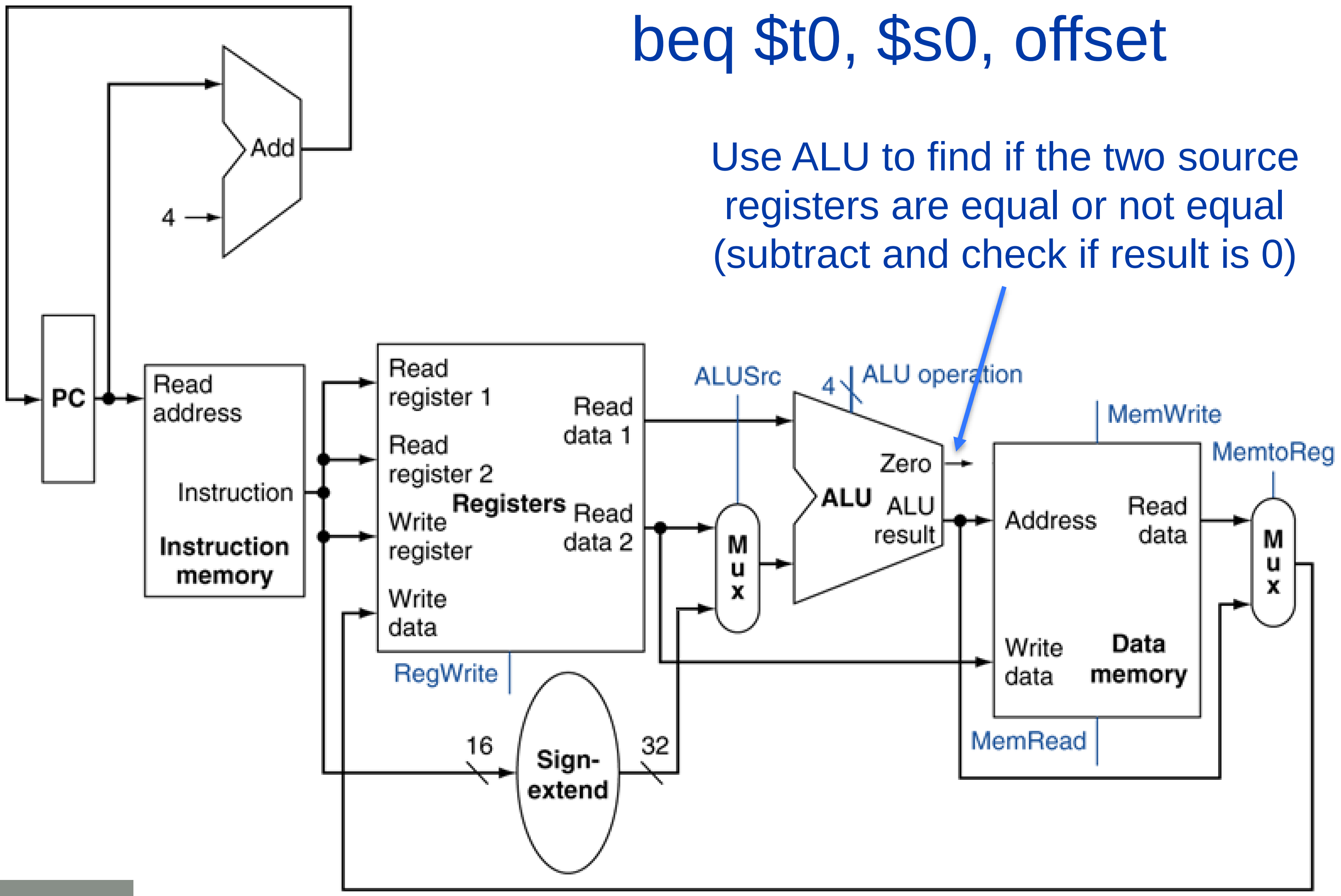
beq $t0, $s0, offset



Branch Instructions

beq $t0, $s0, offset

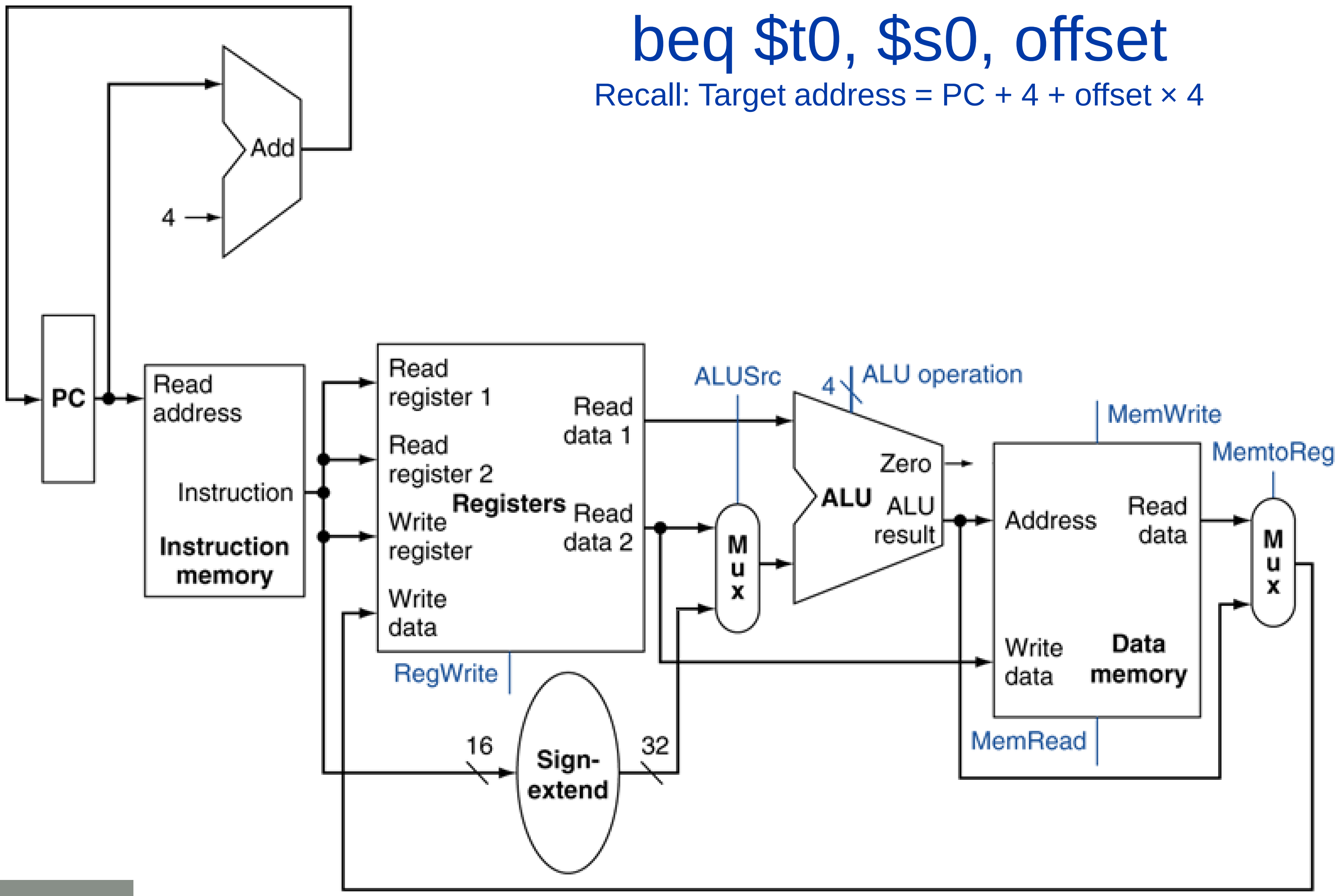
Use ALU to find if the two source registers are equal or not equal (subtract and check if result is 0)



Branch Instructions

beq $t0, $s0, offset

Recall: Target address = PC + 4 + offset × 4

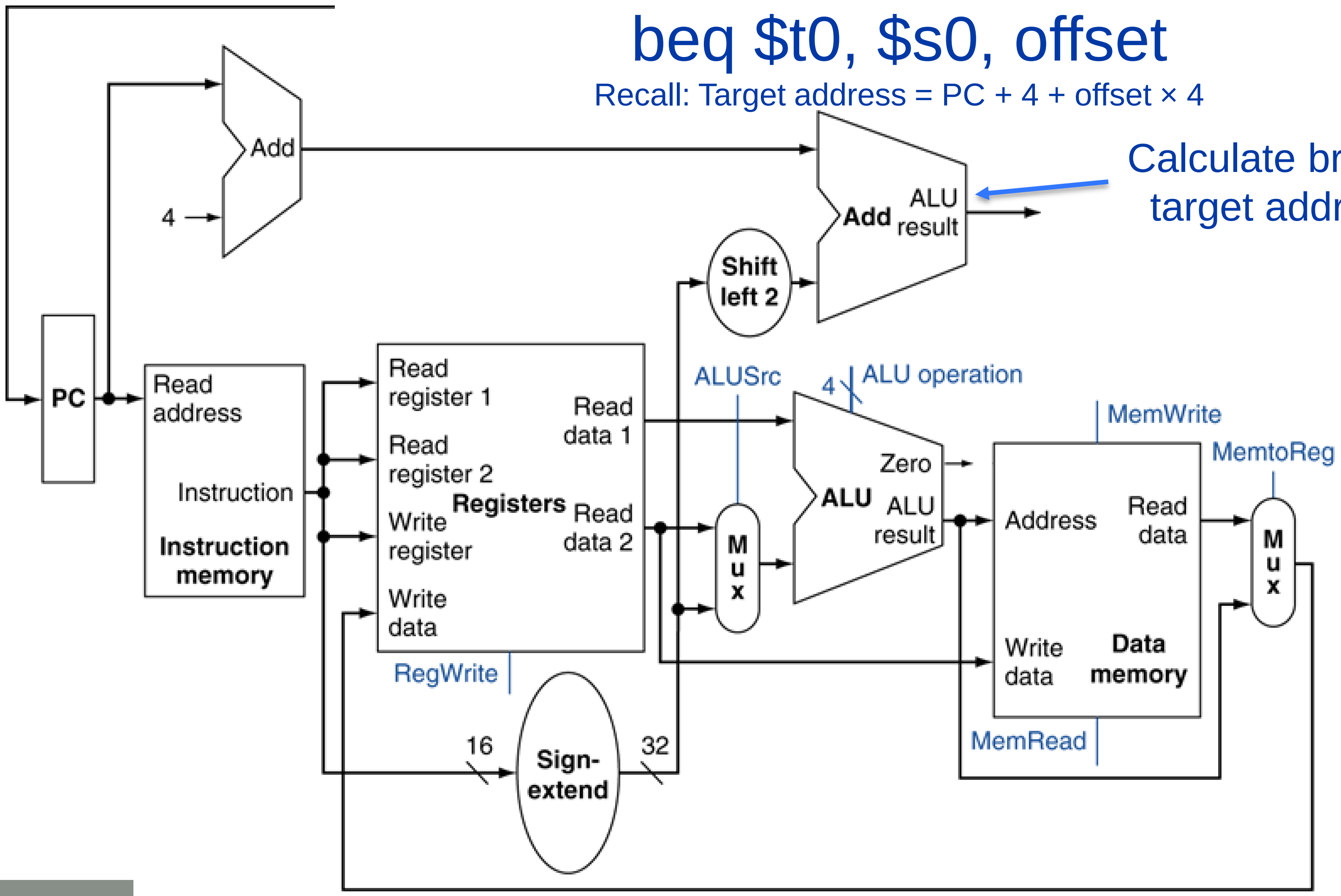


Branch Instructions

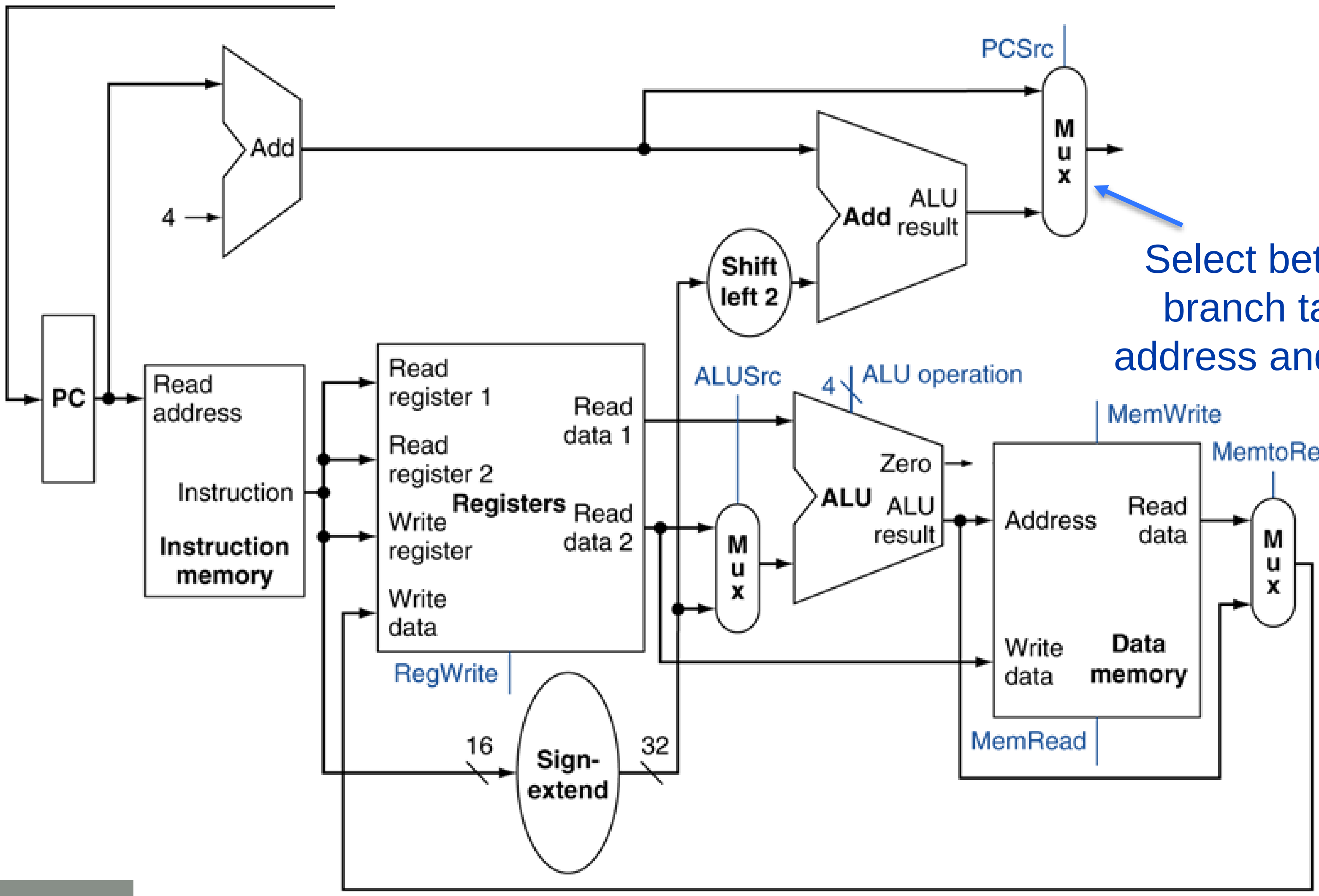
beq $t0, $s0, offset

Recall: Target address = PC + 4 + offset × 4

Calculate branch target address

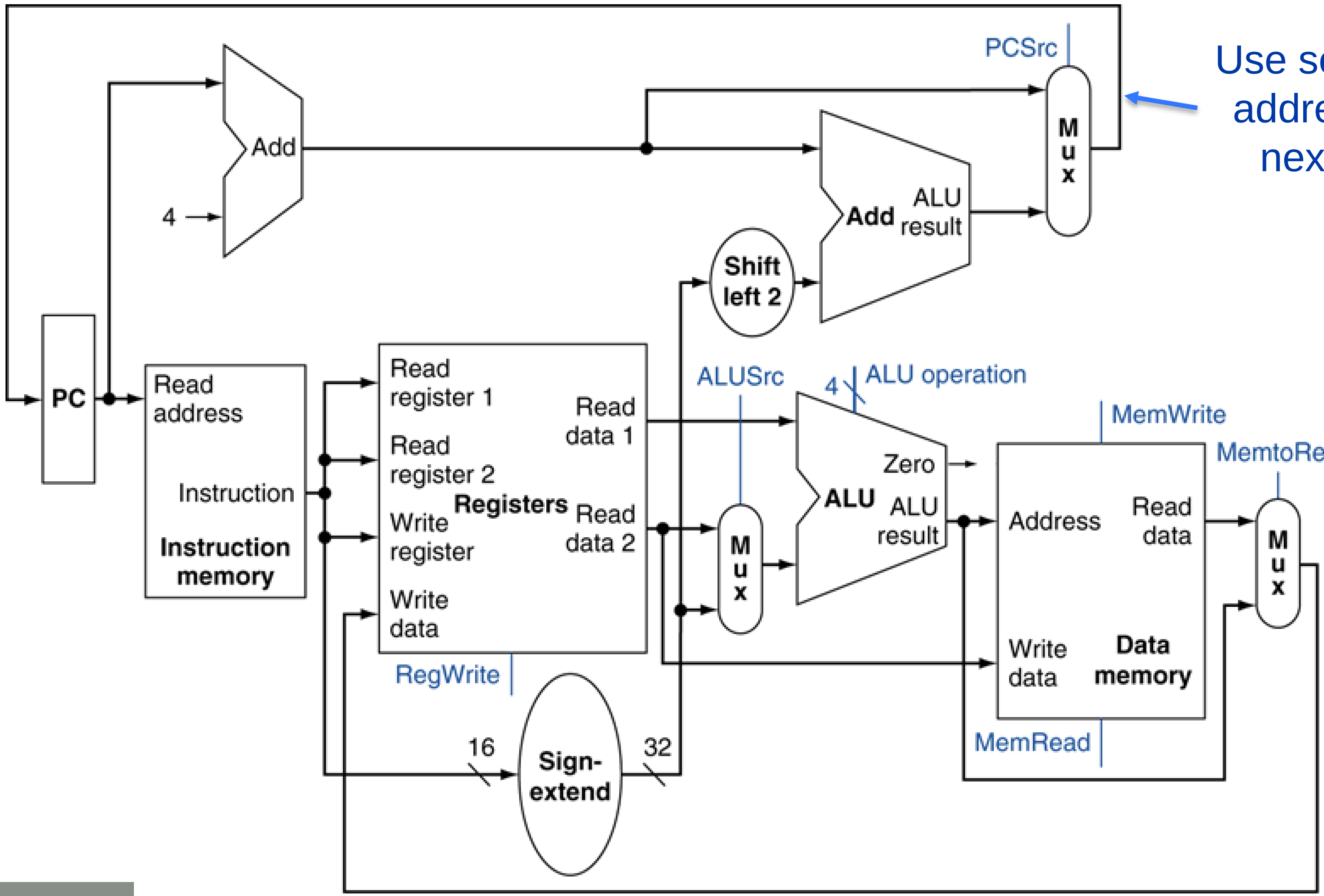


Branch Instructions



Select between branch target address and PC+4

Branch Instructions



Use selected address as next PC

Textbook Sections

- The content in these slides corresponds to:
  - Textbook:
    - *Computer Organization and Design, 5th Edition by David Patterson and John Hennessy, Morgan Kaufmann, 2014.*
  - Sections:
    - 4.1, 4.3