Chapter 4

The Processor

Dr. Randa Mohamed

Agenda

- Introduction
- Logic design basics
- MIPS Datapath: R-Type & I-Type

Introduction

- Processor design is composed of 2 modules: Datapath and Control
- We will examine

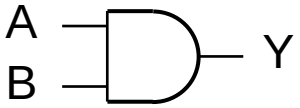
1. Two MIPS implementations
   - A simplified version (Single cycle MIPS)
   - A more realistic pipelined version (Pipelined MIPS)
2. Implementation that supports
   - R-type instructions, like ADD, SUB
   - I-type instructions LW, SW
   - Conditional branch instruction BEQ
   - J-type branch instruction J

Logic Design Basics

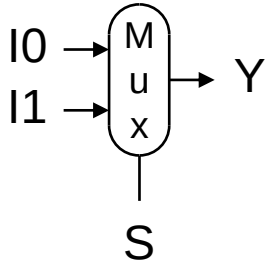
- Information encoded in binary
  - Low voltage = 0, High voltage = 1
  - One wire per bit
  - Multi-bit data encoded on multi-wire buses
- Combinational element
  - Operate on data
  - Output is a function of input
- State (sequential) elements
  - Store information

Combinational Elements

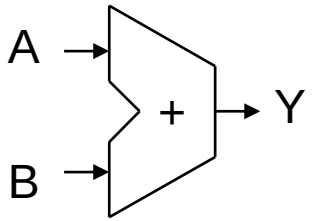
- AND-gate
  - Y = A & B



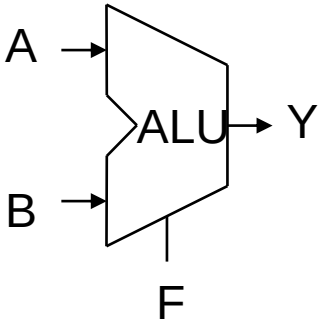
- Multiplexer
  - Y = S ? I1 : I0



- Adder
  - Y = A + B

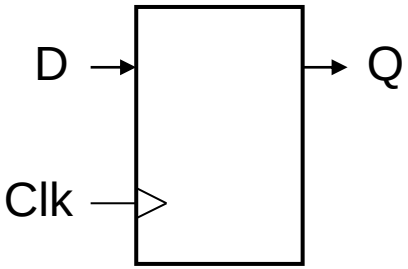


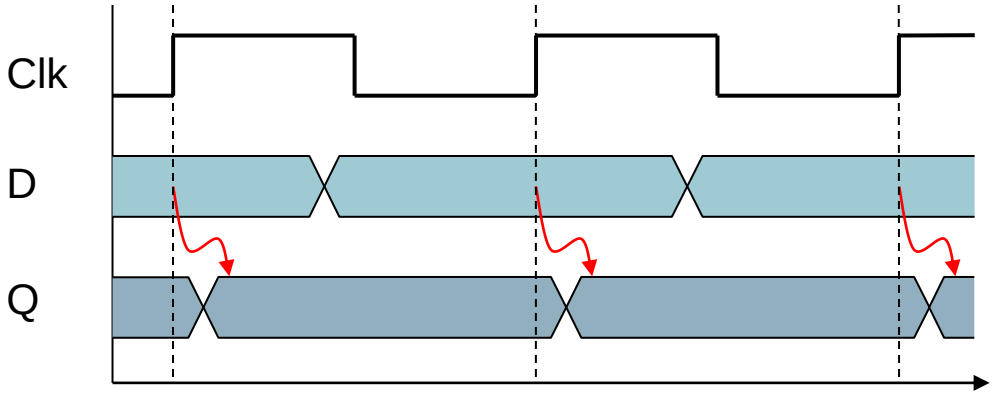
- Arithmetic/Logic Unit
  - Y = F(A, B)



Sequential Elements

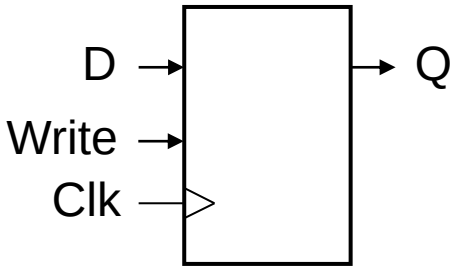
- Stores data in a circuit
  - Uses a clock signal to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1

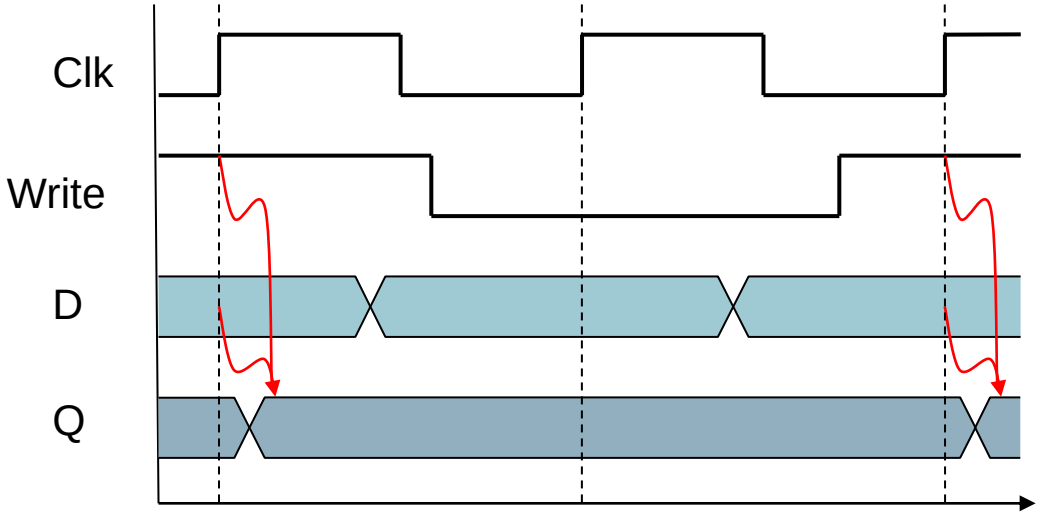




Sequential Elements

- Storing with write control
  - Only updates on clock edge when write control input is 1
  - Used when stored value is required later

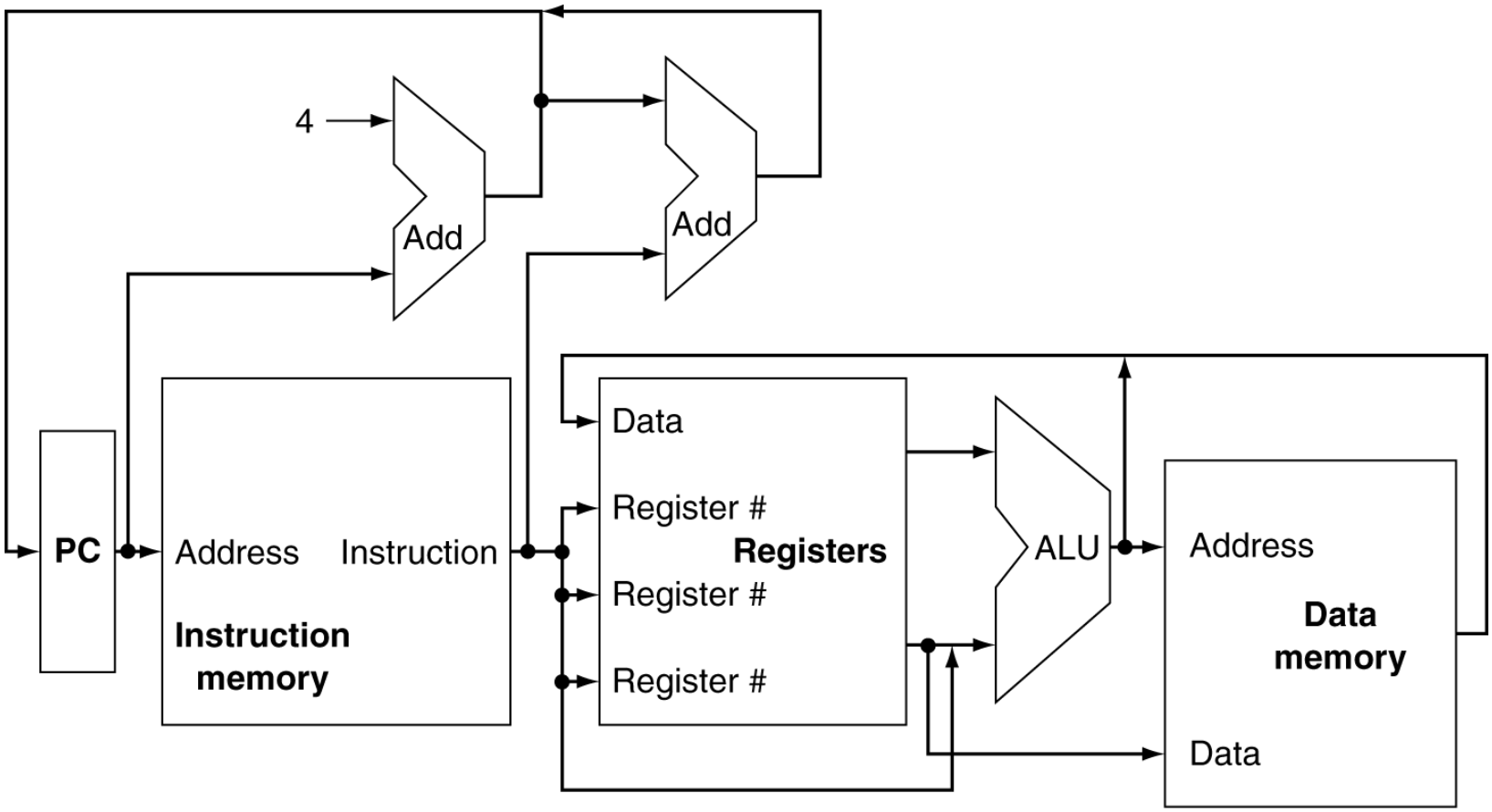




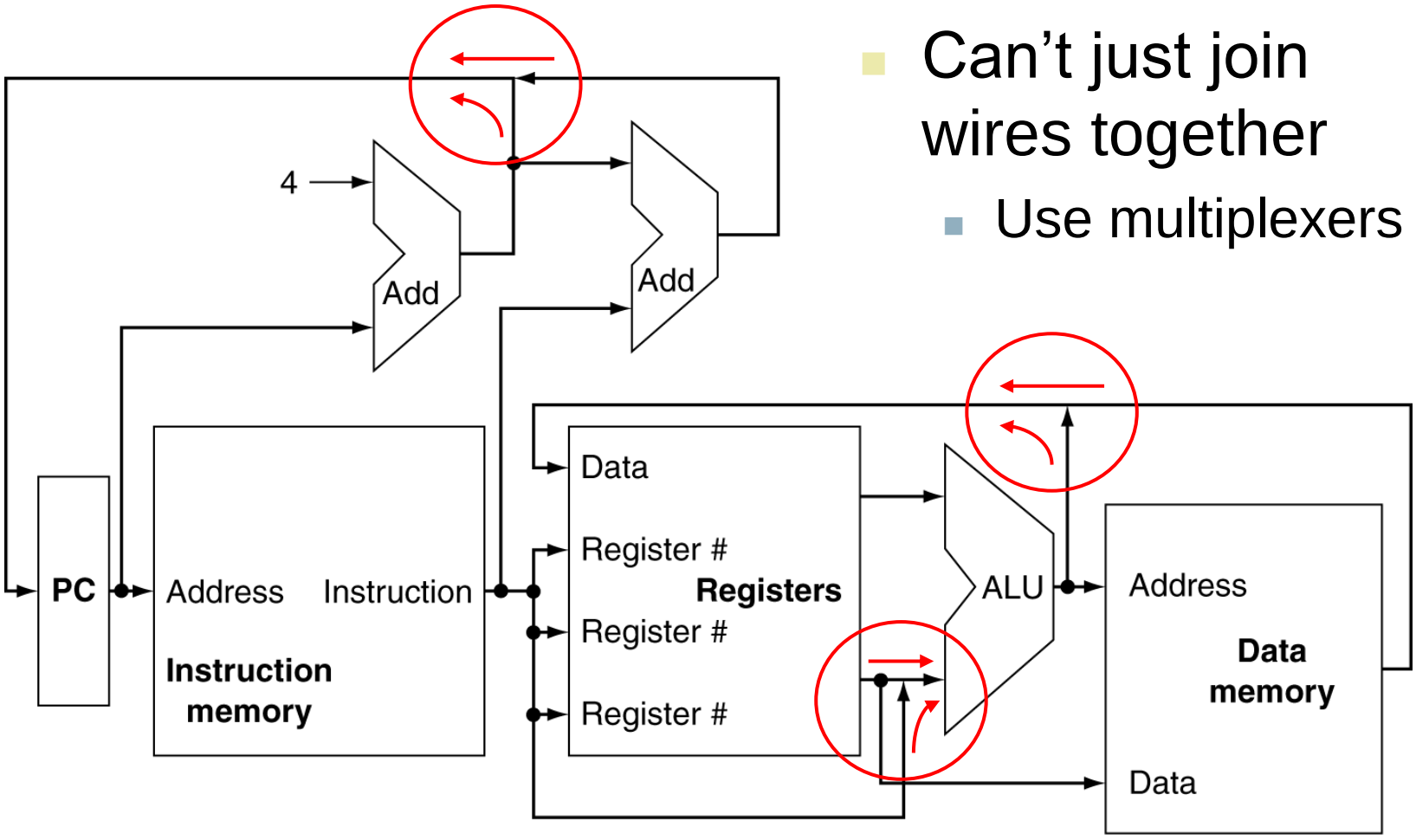
Building a Datapath

- Datapath
  - Elements that process data and addresses in the CPU
    - Registers, ALUs, mux's, memories, …
- We will build a MIPS datapath incrementally
  - Refining the overview design

Datapath

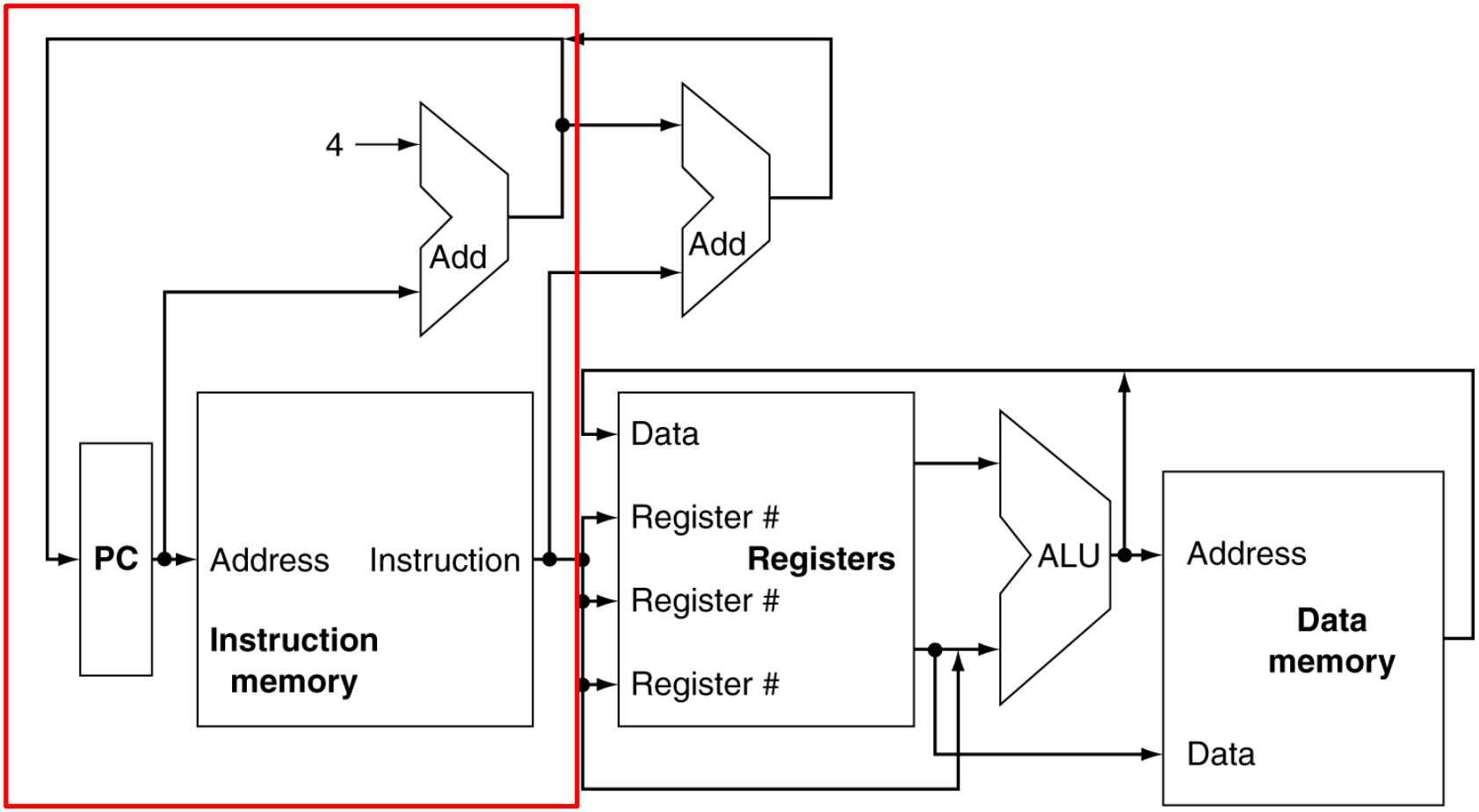

4
Add
Add
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
ALU
Address
Data memory
Data

Multiplexers

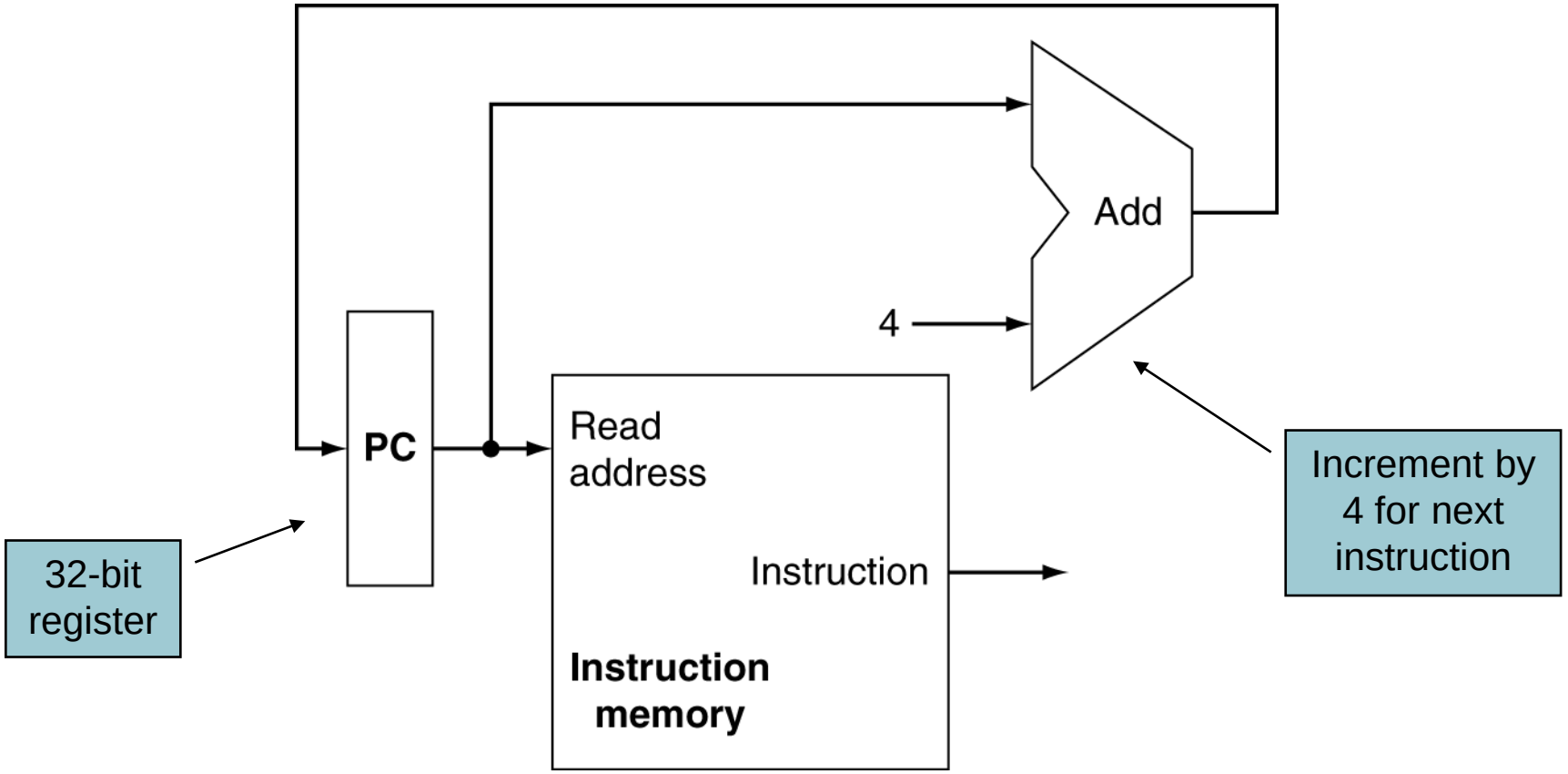


- Can't just join wires together
  - Use multiplexers

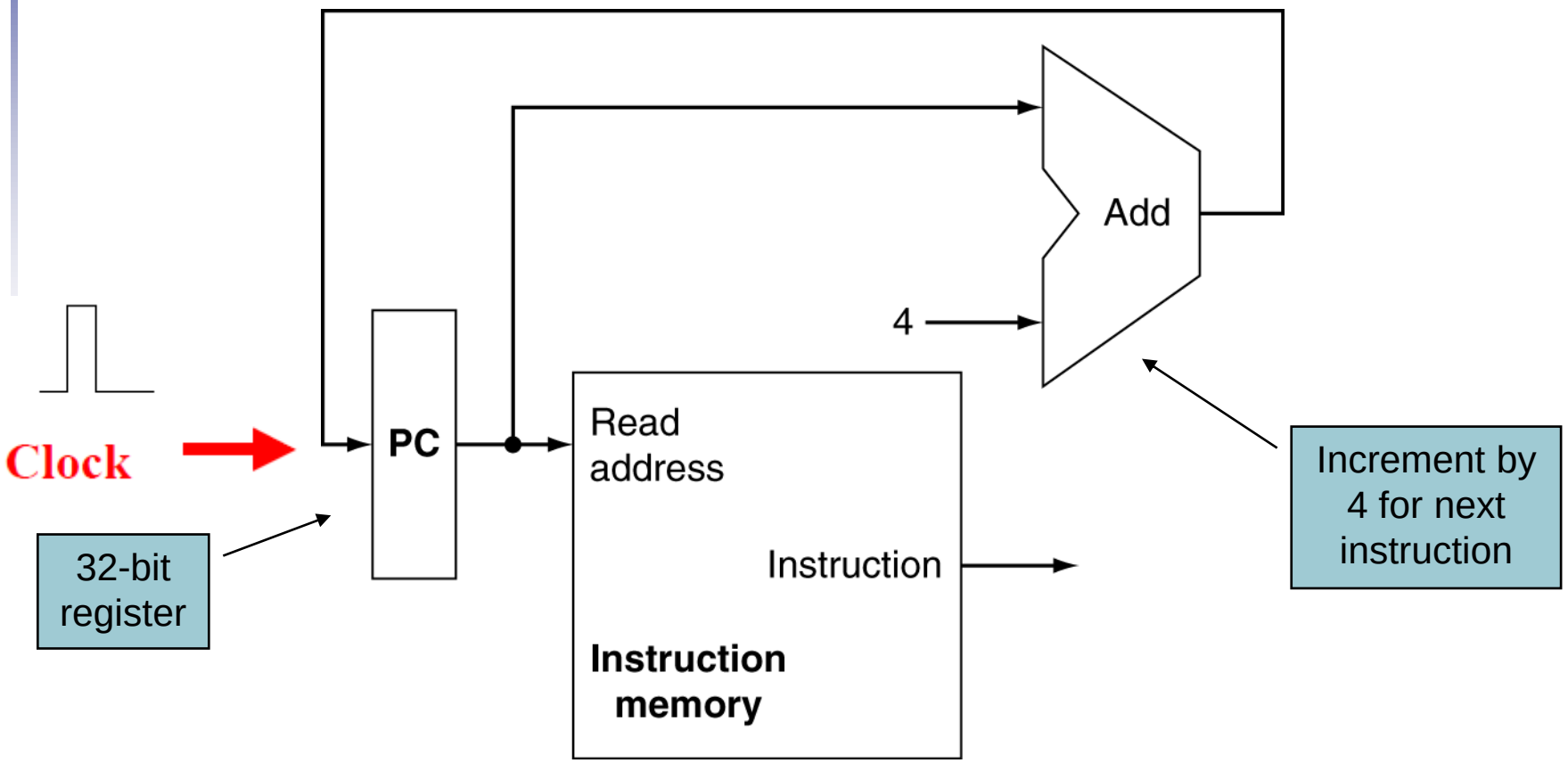
Datapath


4
Add
Add
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
ALU
Address
Data memory
Data

Instruction Fetch



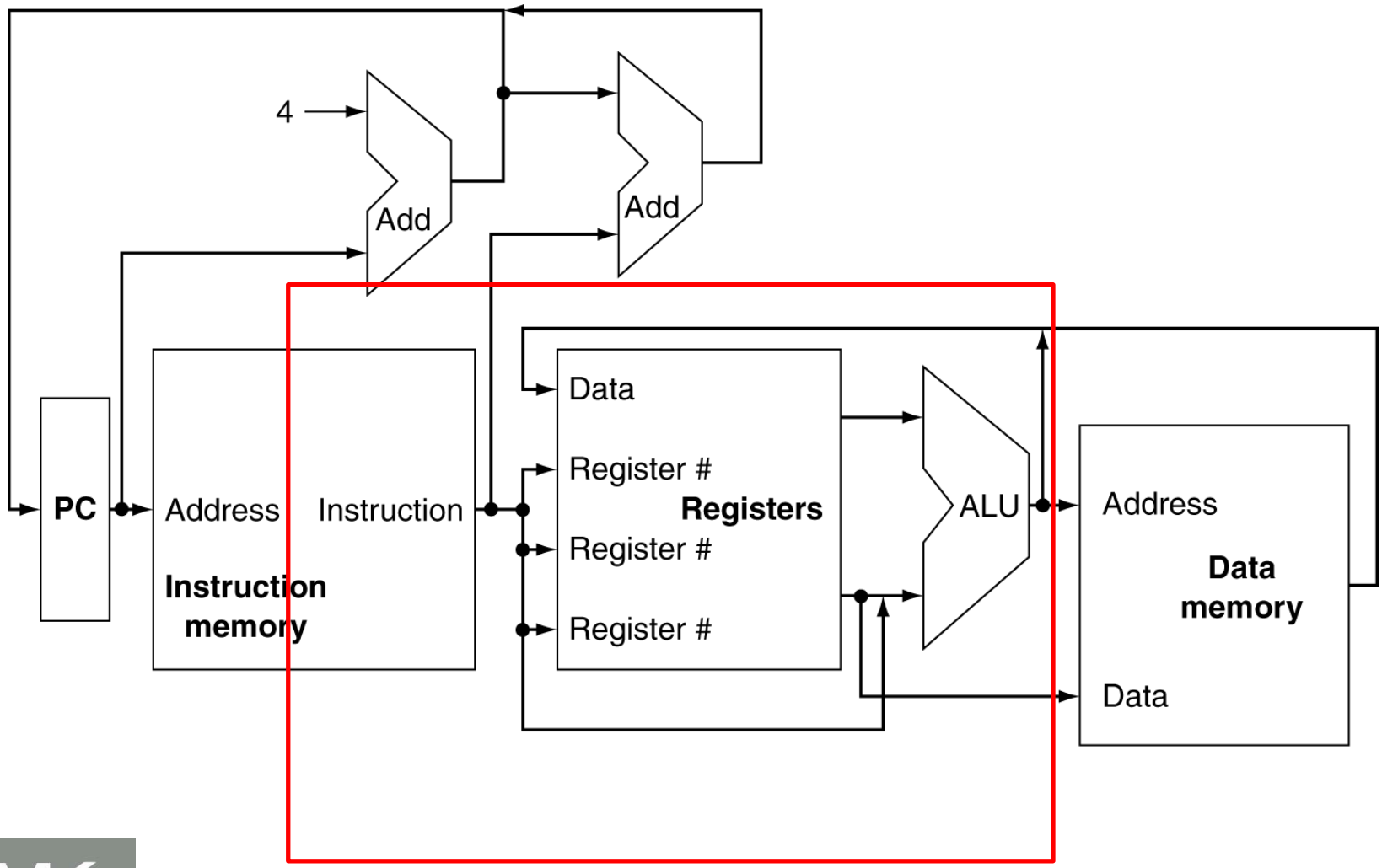
Instruction Fetch



Instruction Fetch

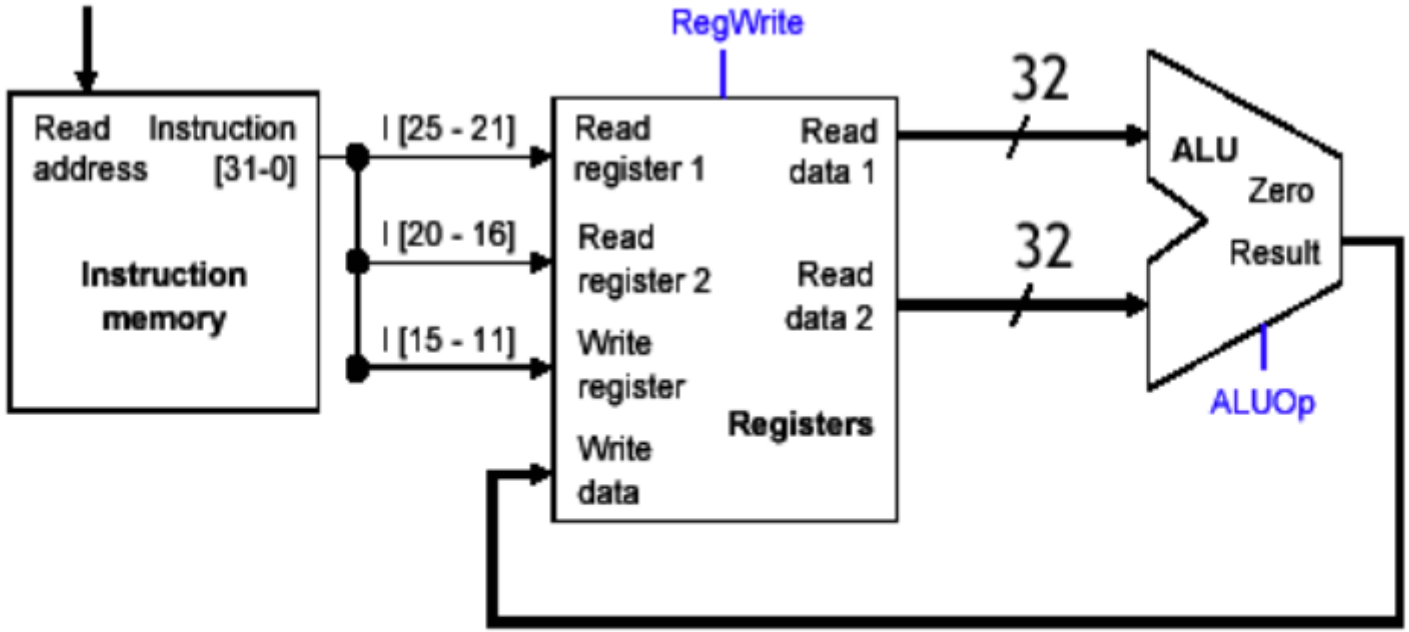
Field bit positions	31-26	25-21	20-16	15-11	10-6	5-0
No. of bits	6	5	5	5	5	6
R-type	op	rs	rt	rd	shamt	funct
I-type	op	rs	rt	16- bit immediate/address		

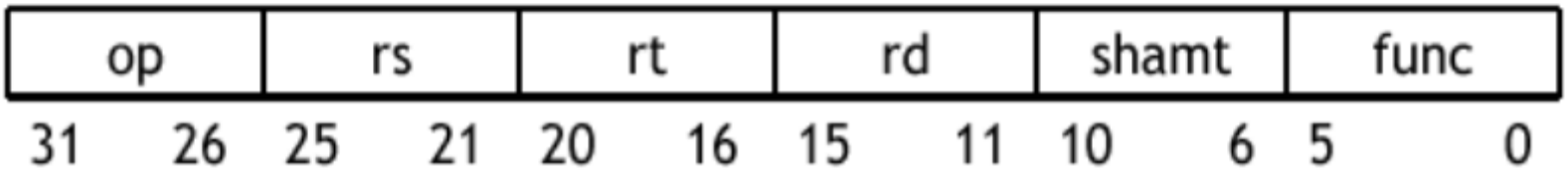
R-Format Instructions



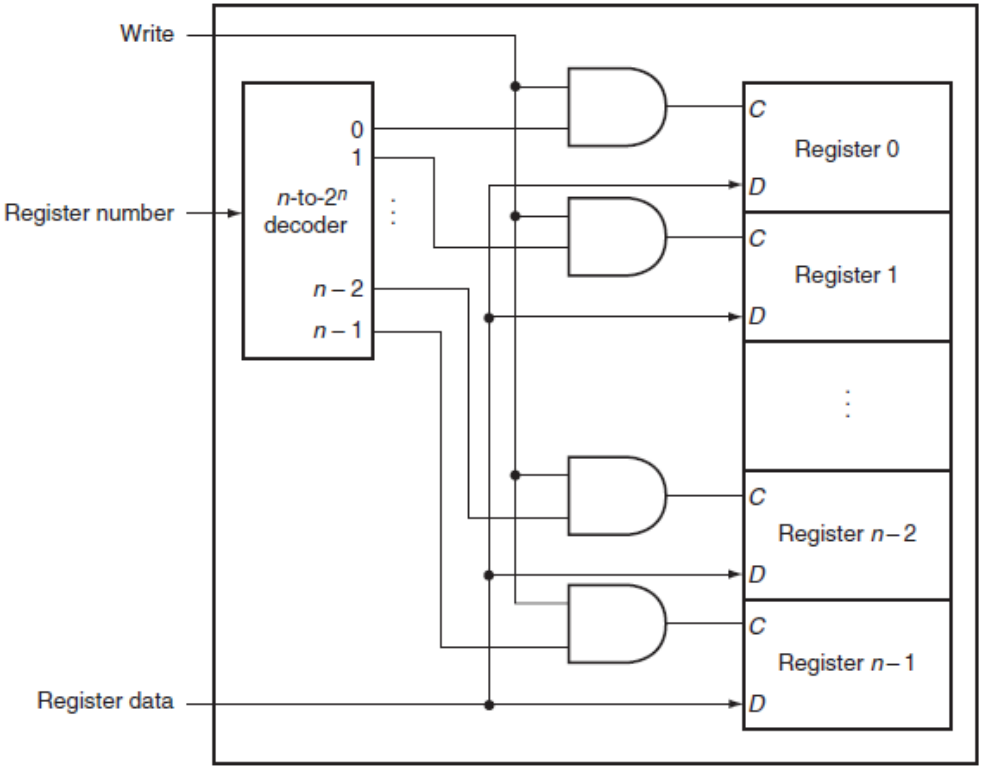
R-Format Instructions

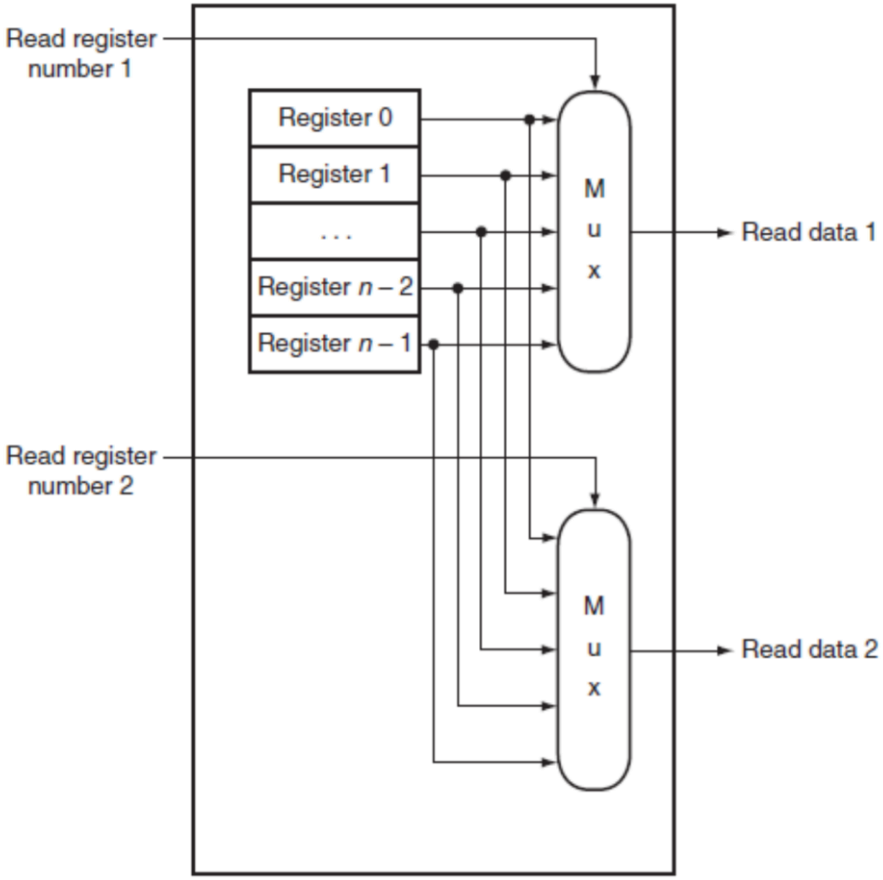
- Read two source register operands (defined by rs,rt)
- Perform arithmetic/logical operation
- Write result in destination register (defined by rd)



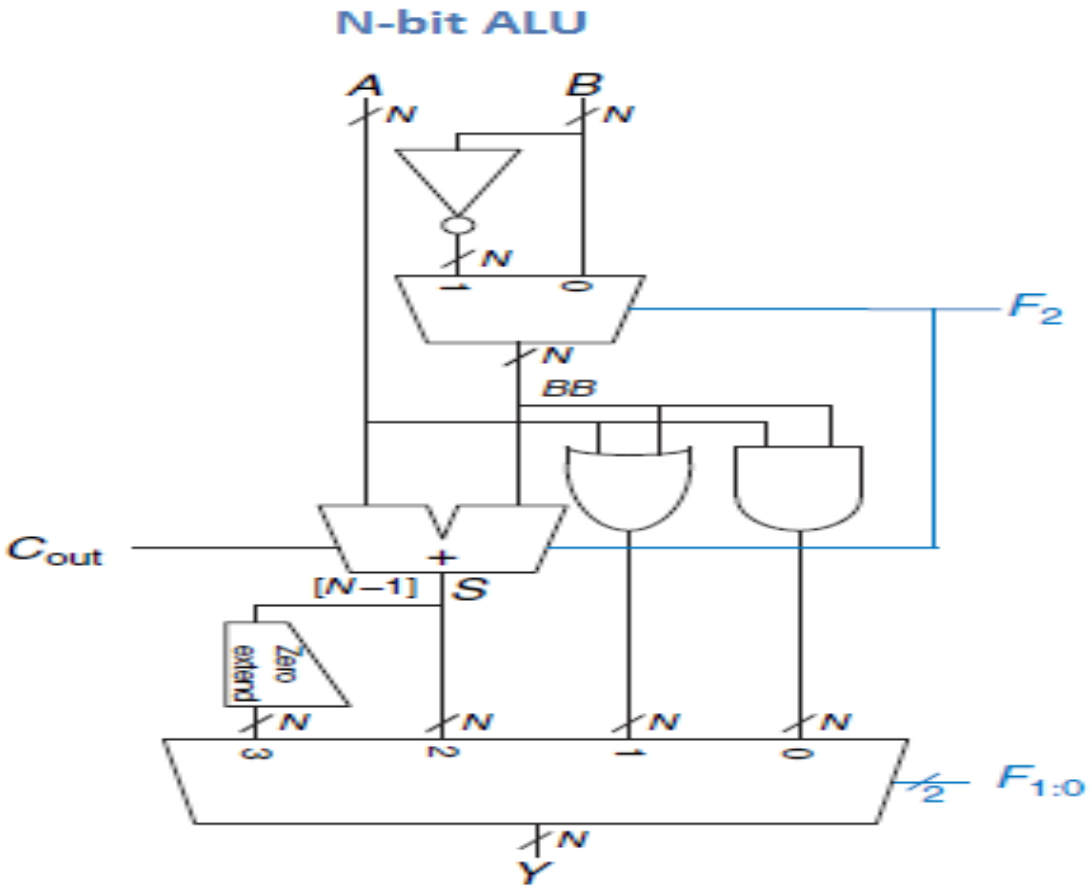


Register File

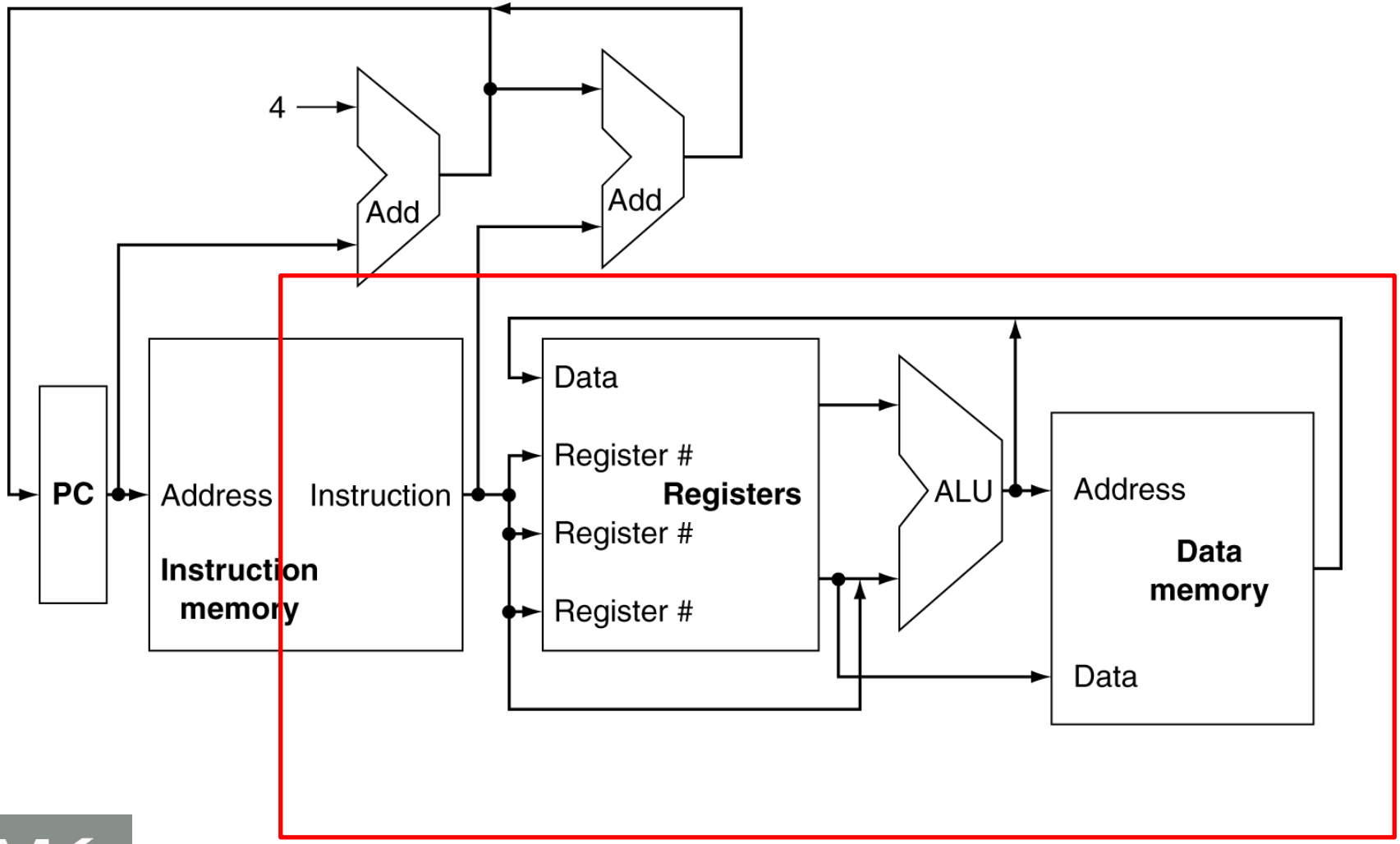




ALU

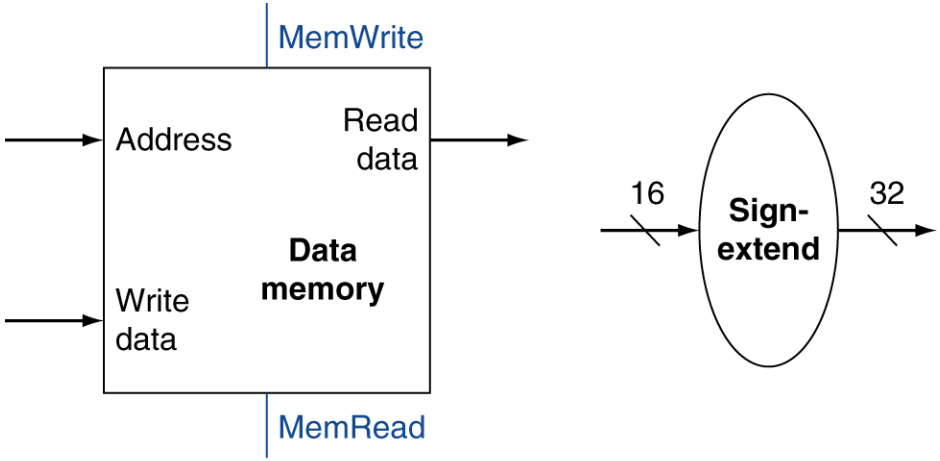

N-bit ALU
A
B
N
N
N
1
0
F2
N
BB
Cout
+
[N–1]
S
Zero extend
N
N
N
N
3
2
1
0
2
F1:0
N
Y

Load/Store Instructions



Load/Store Instructions

- To read from the data memory, set Memory read =1
- To write into the data memory, set Memory write =1



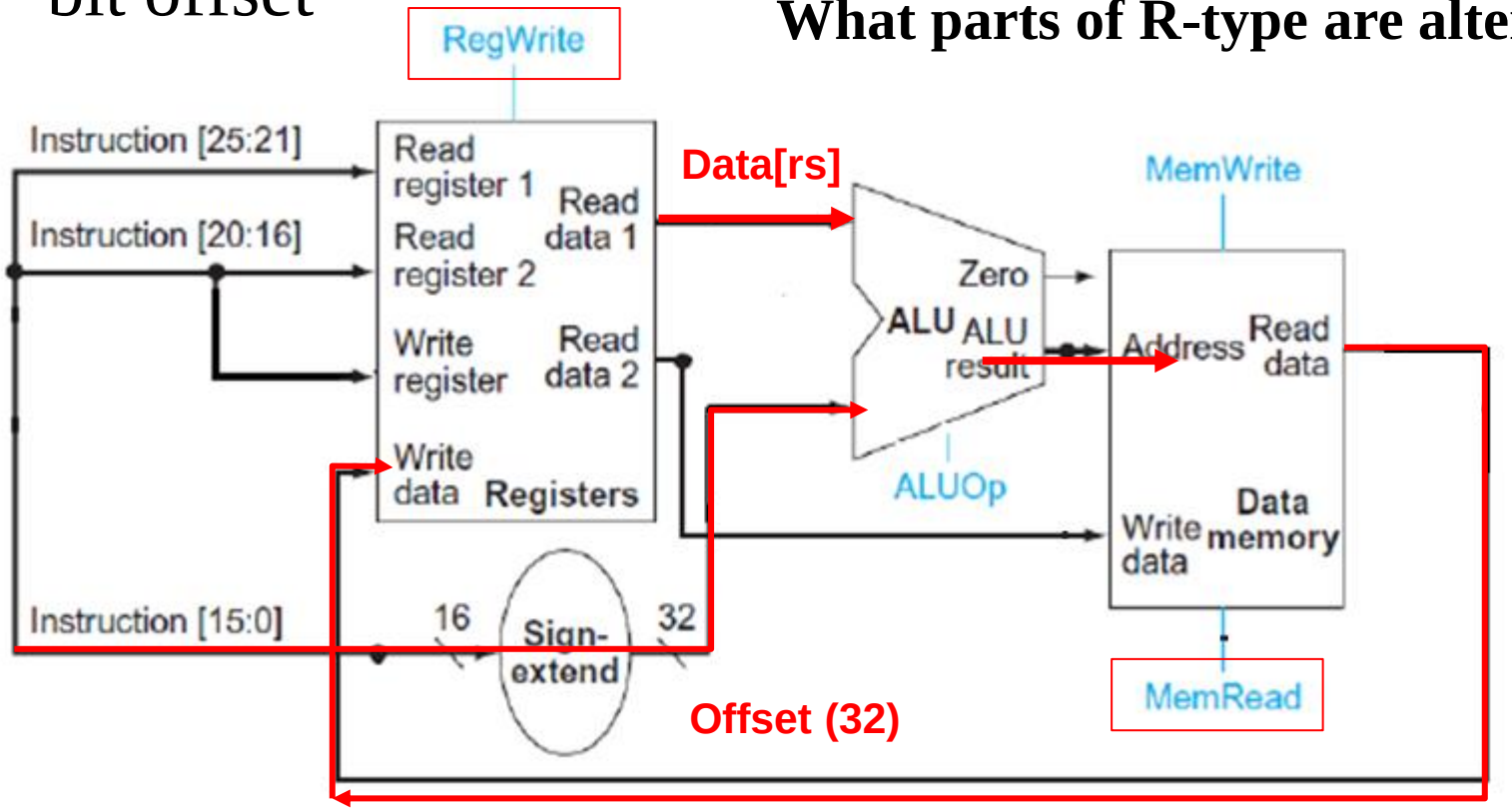
a. Data memory unit

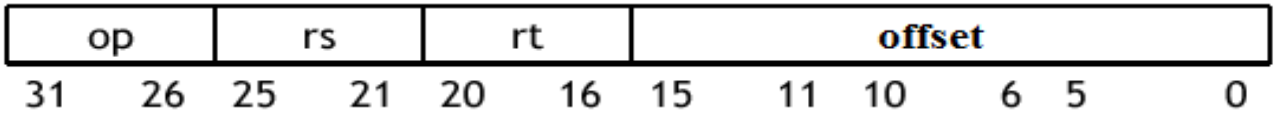
b. Sign extension unit

Load Instruction

- Read register operands & calculate address using 16-bit offset

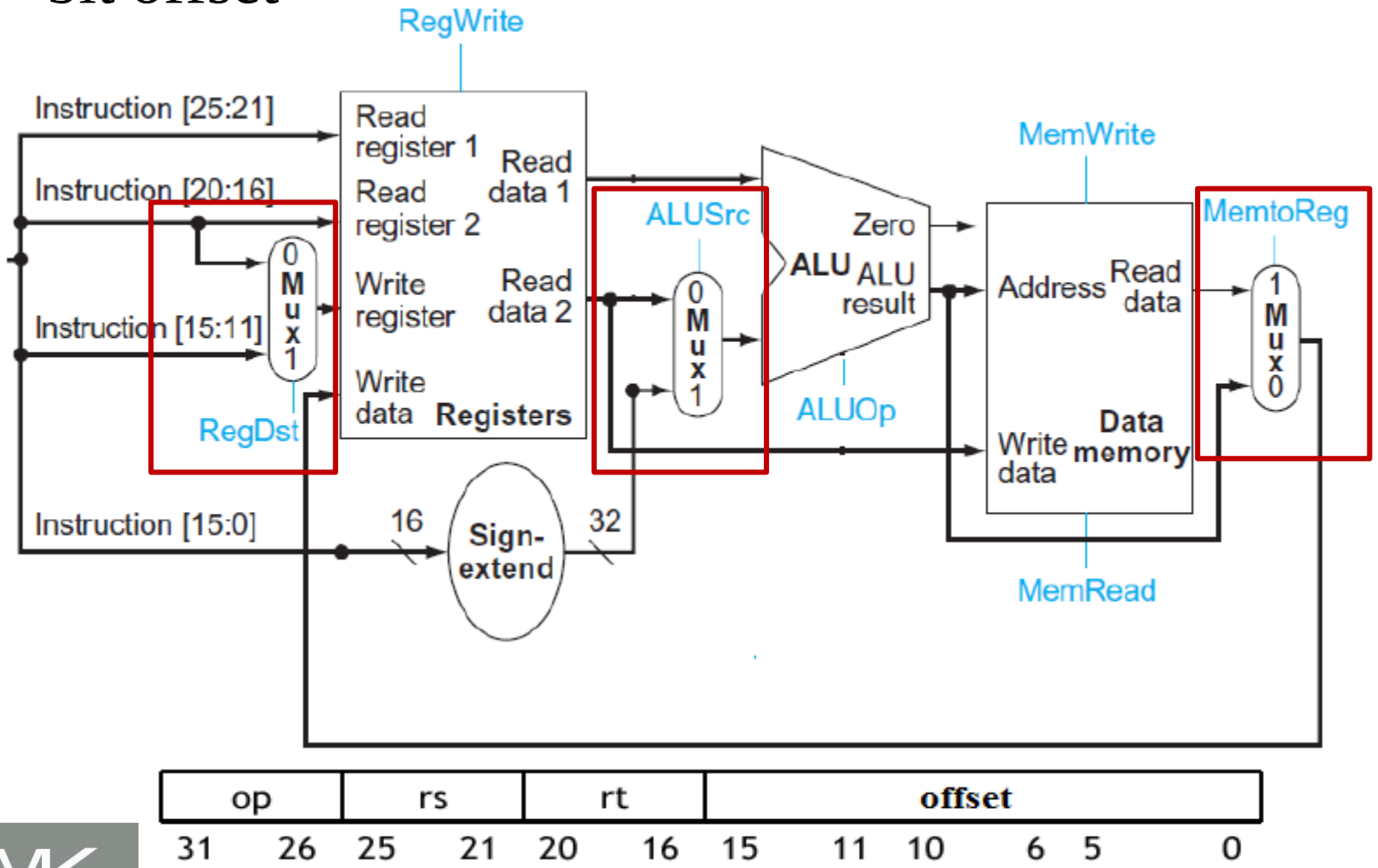
What parts of R-type are altered?





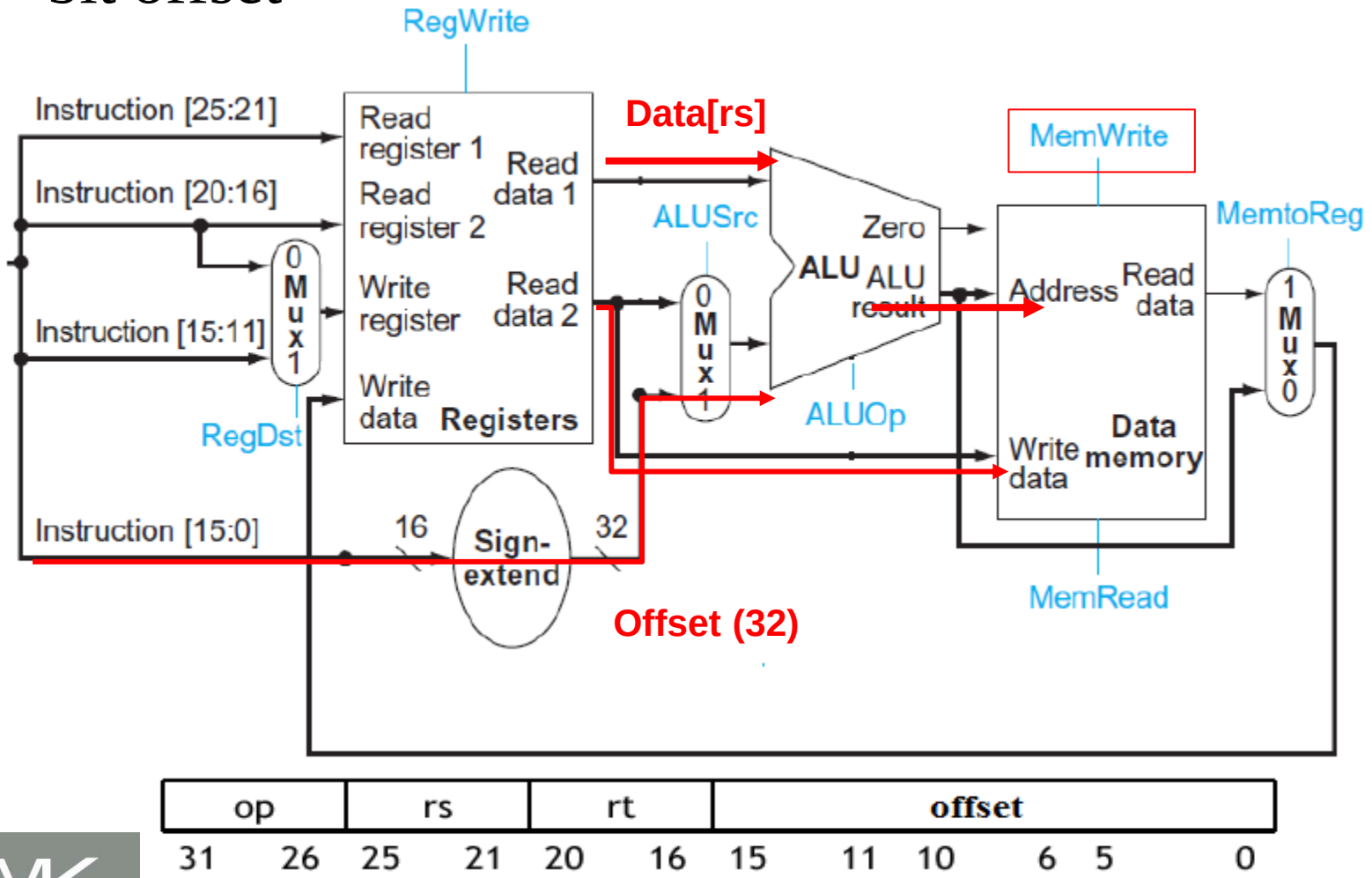
R-Type/Load Instruction

- Read register operands & calculate address using 16-bit offset

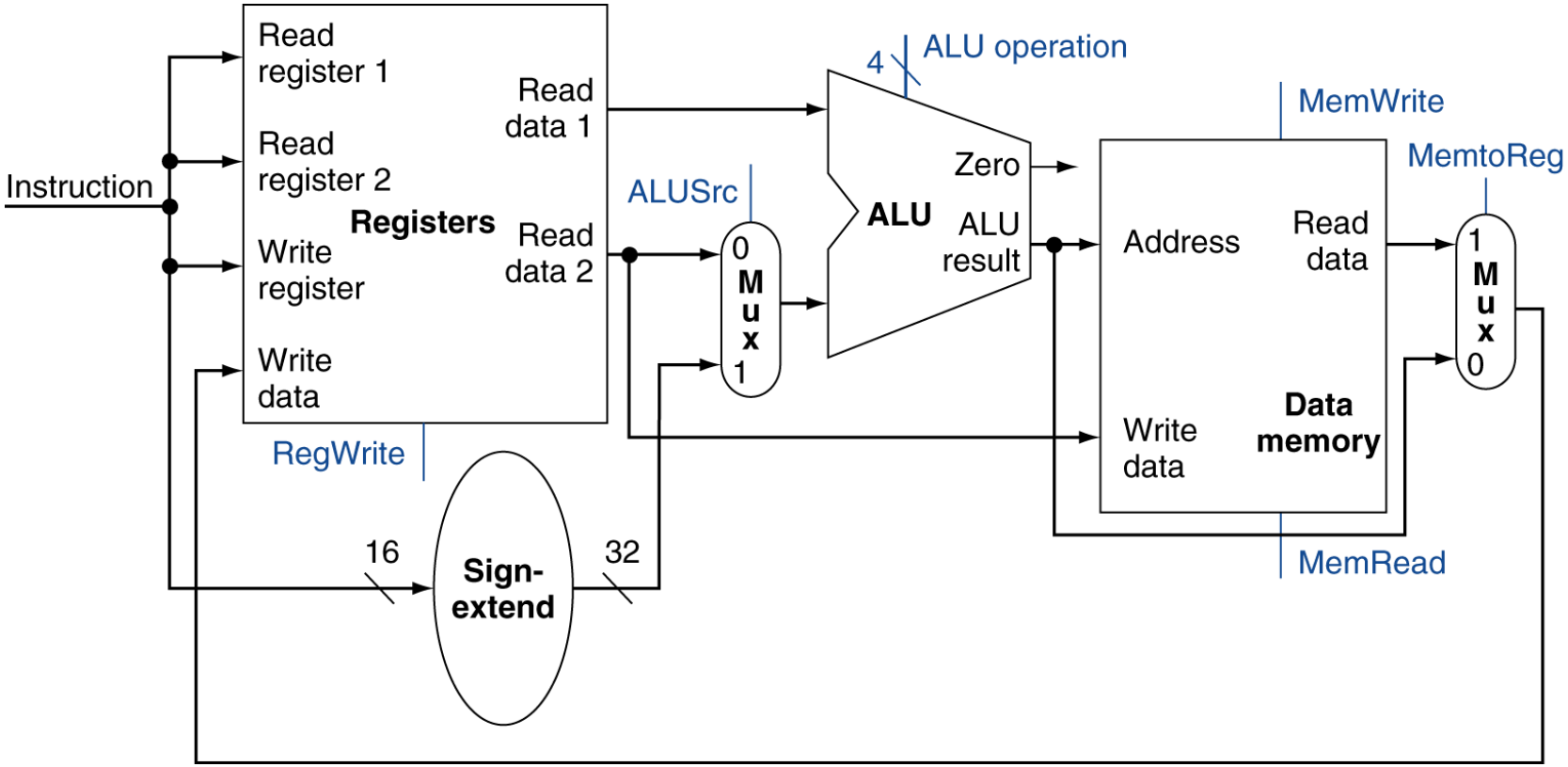


Store Instruction

- Read register operands & calculate address using 16-bit offset



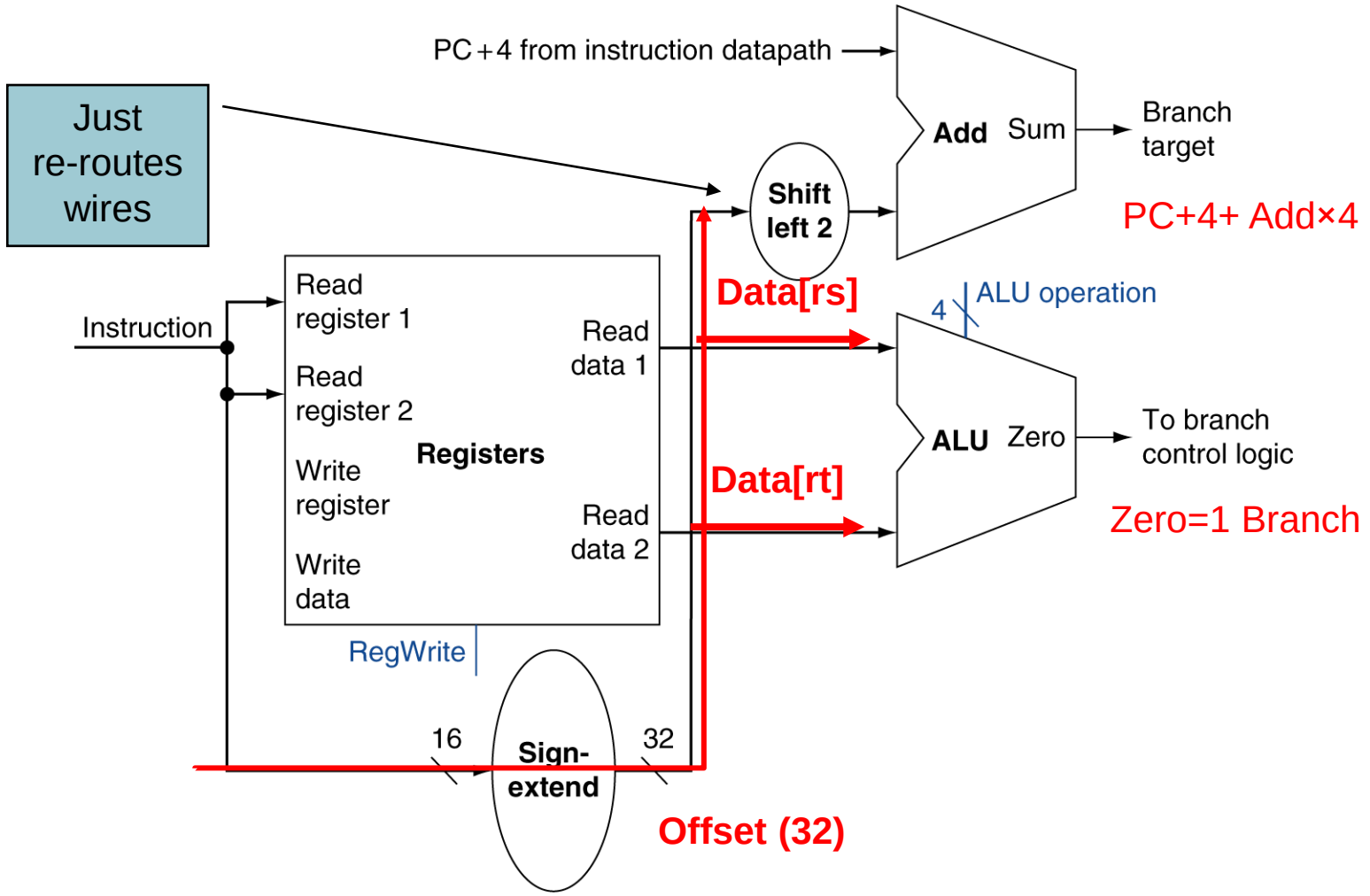
R-Type/Load/Store Datapath



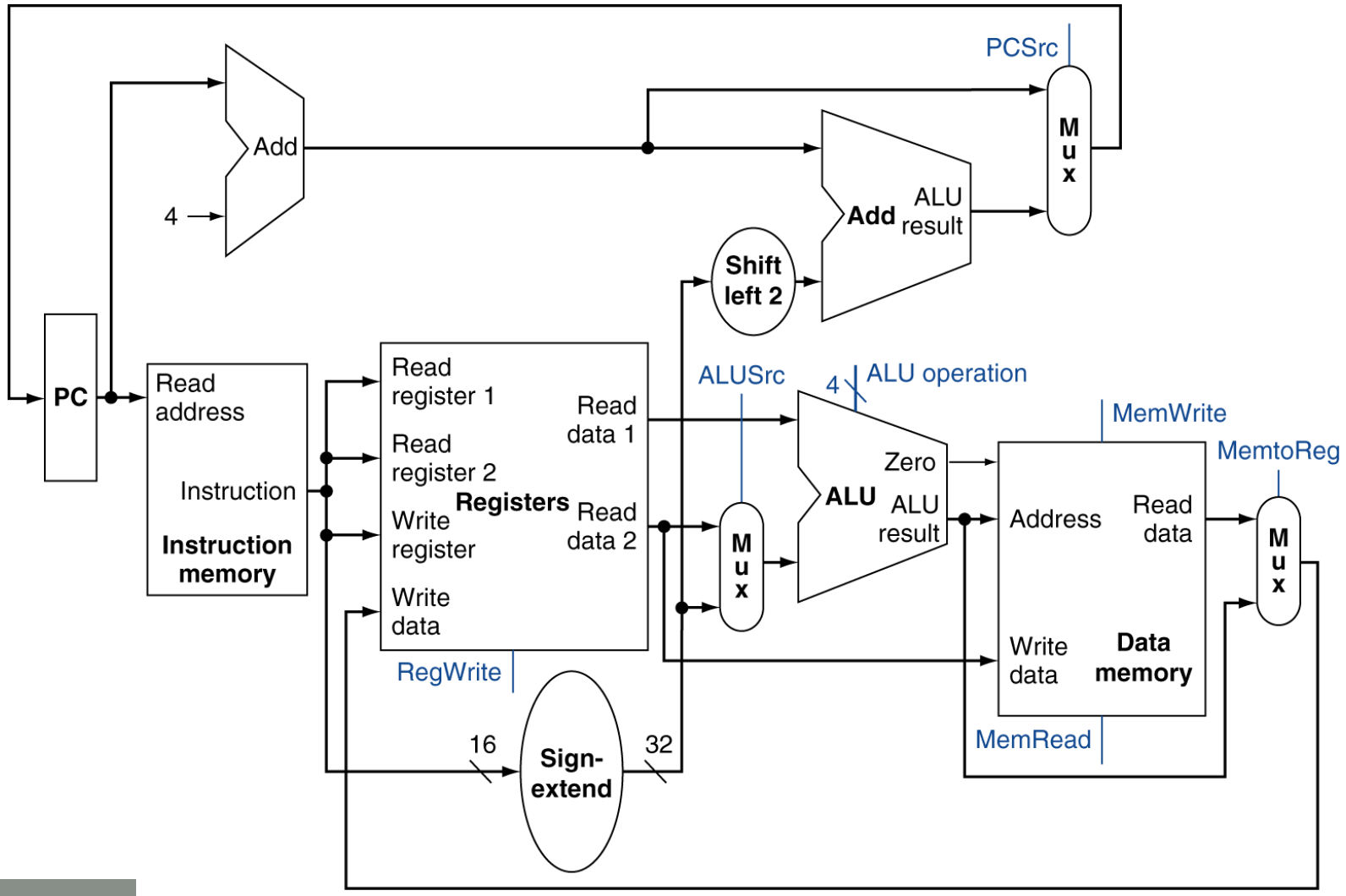
Branch Instructions

- Read register operands
- Compare operands
  - Use ALU, subtract and check Zero output
- Calculate target address (PC +4 +Add×4)
  - Sign-extend displacement
  - Shift left 2 places (word displacement)
  - Add to PC + 4
    - Already calculated by instruction fetch

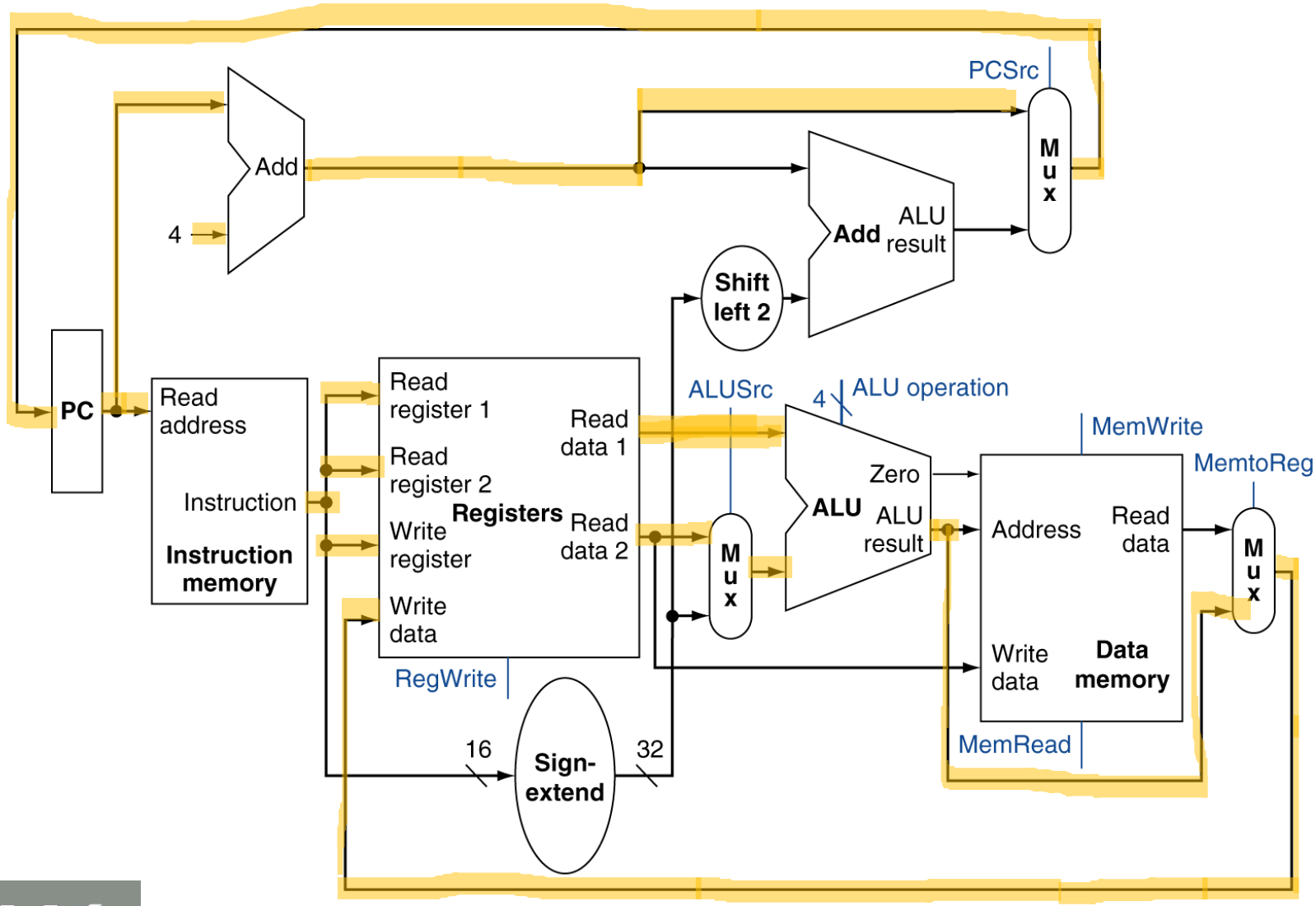
Branch Instructions



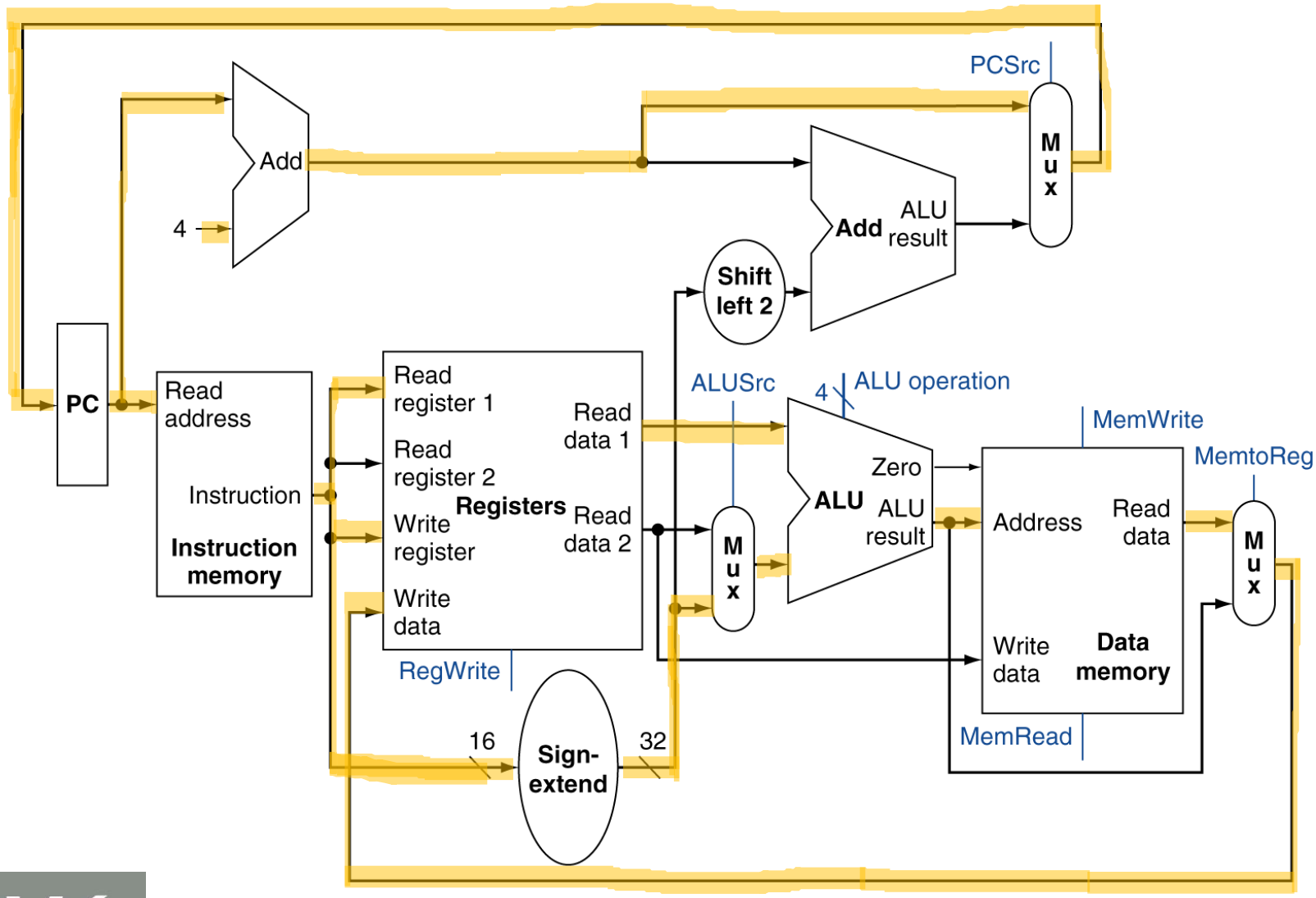
Full Datapath



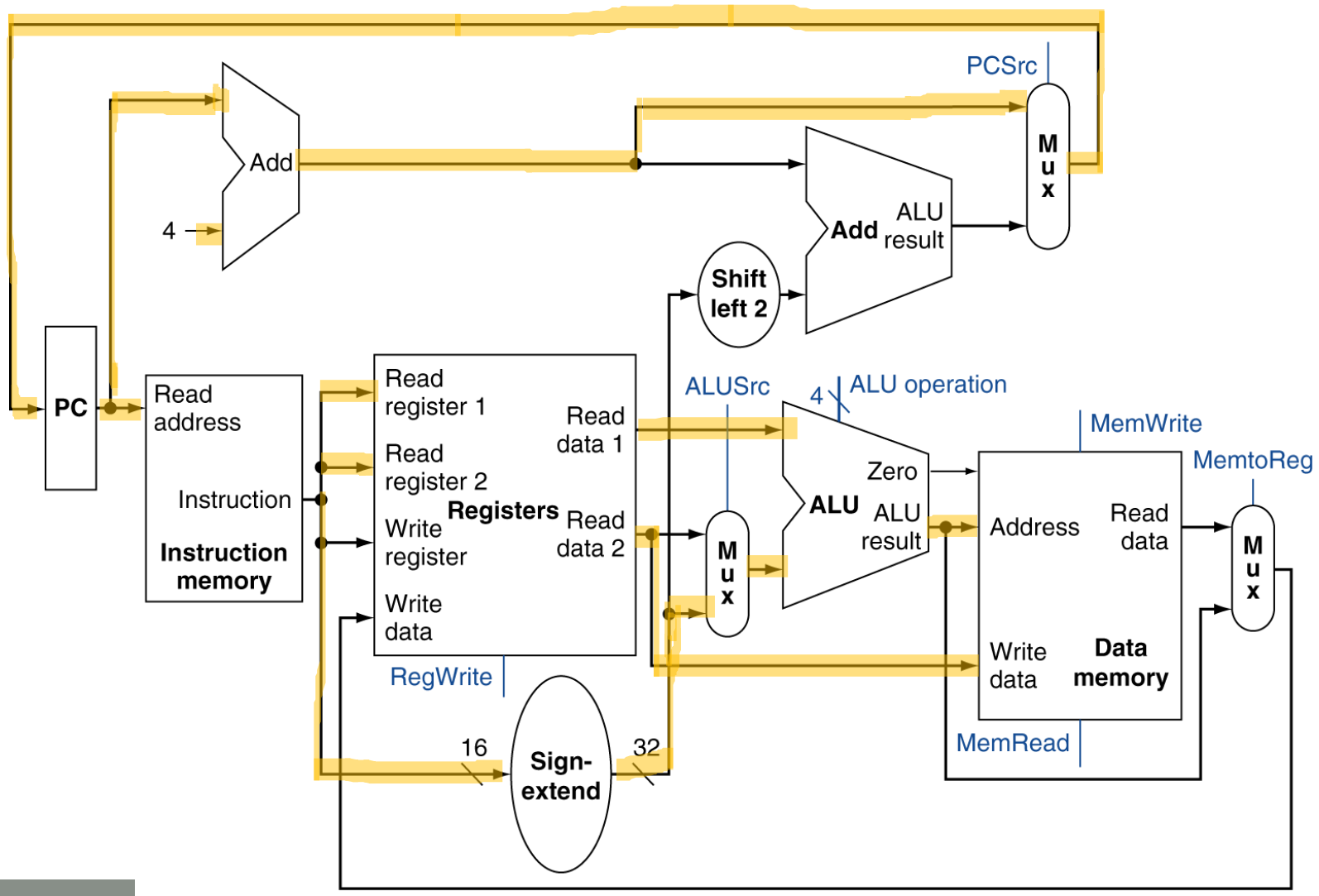
Full Datapath: R-Type



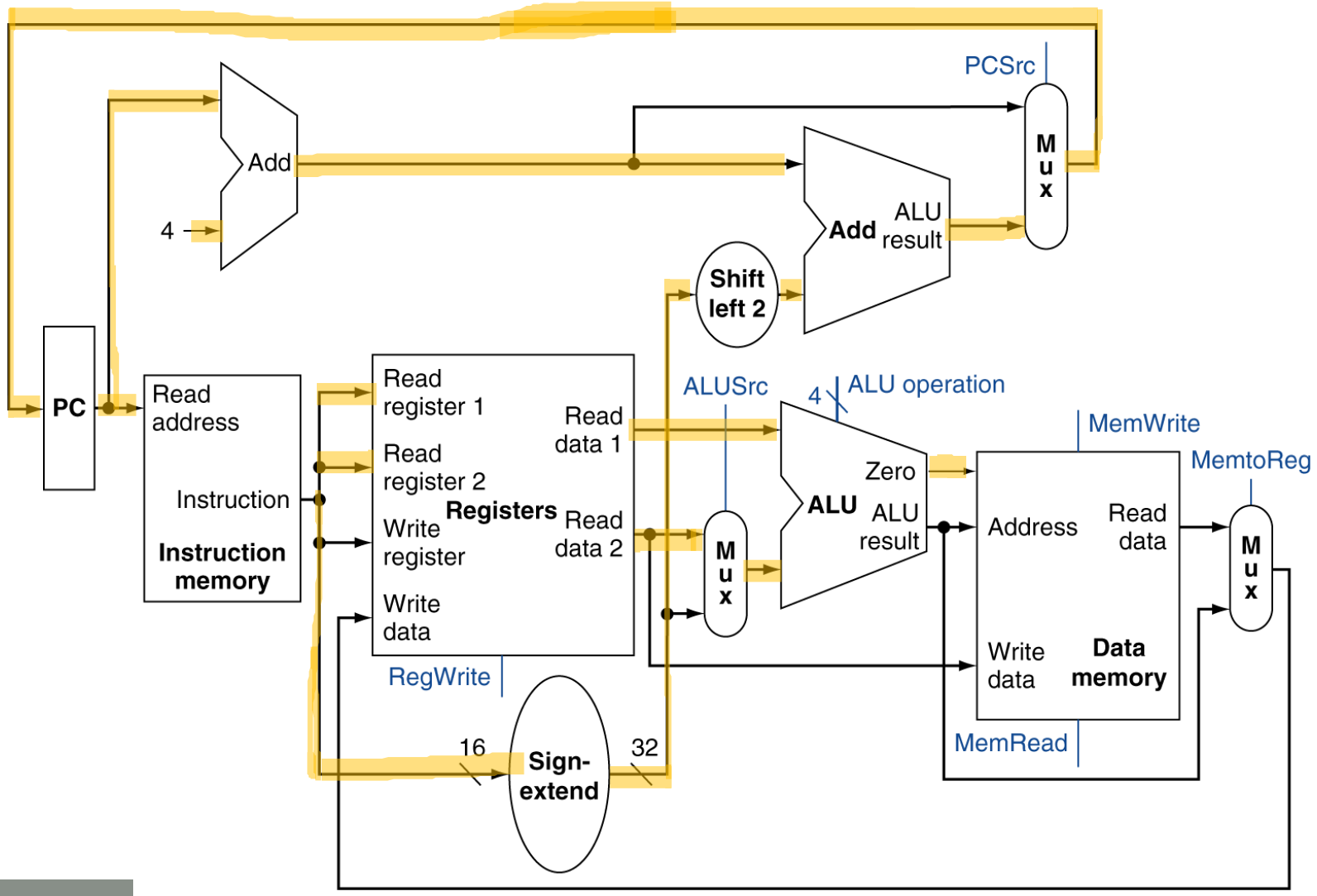
Full Datapath: I-Type (Load)



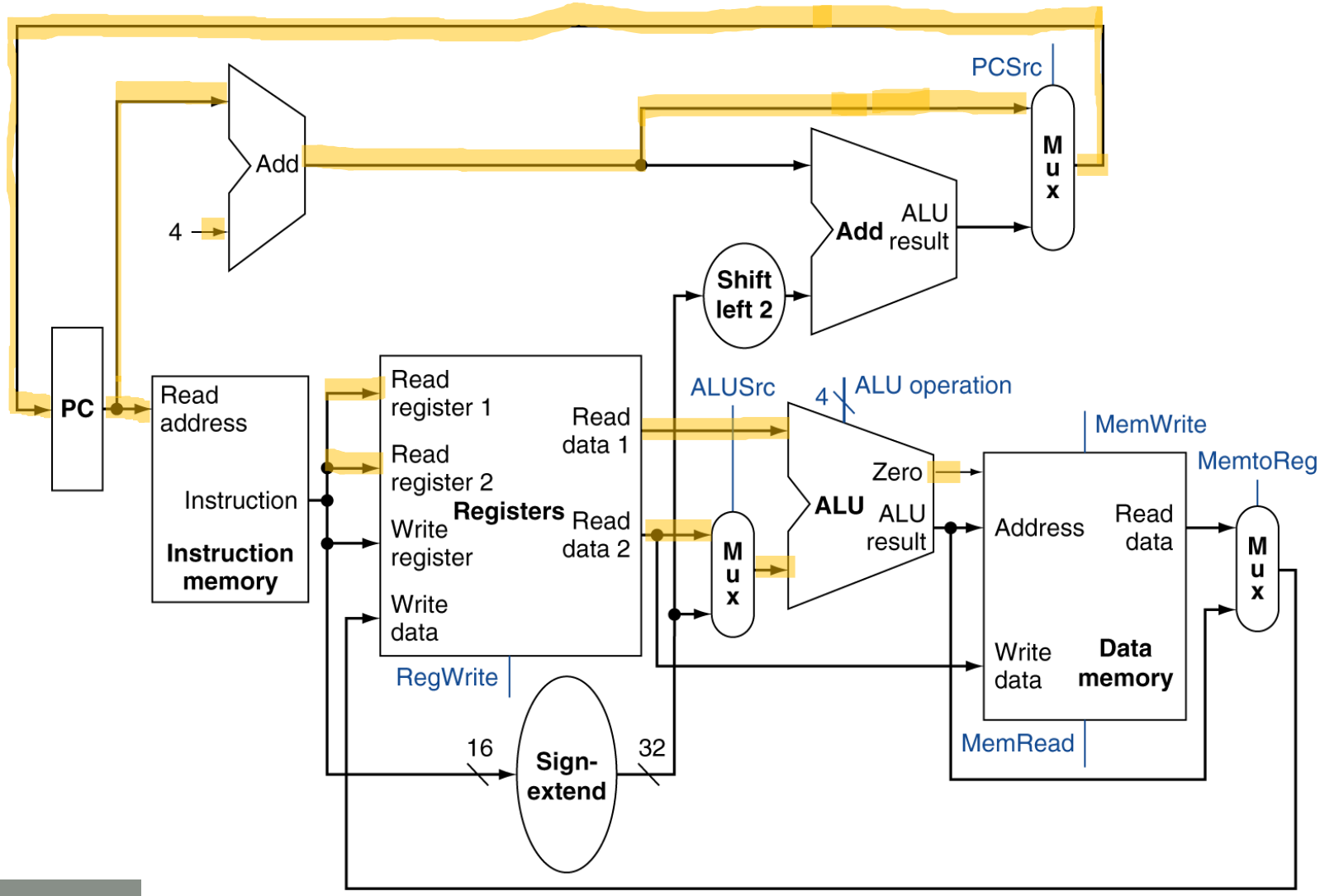
Full Datapath: I-Type (Store)



Full Datapath: I-Type (Beq:T)



Full Datapath: I-Type (Beq:F)



Instruction Execution Summary

Data path does an instruction in one clock cycle

1. Fetch: PC → instruction memory → instruction
2. Decode: Register numbers → register file → read registers
3. Execute: Depending on instruction class
   - Use ALU to calculate
     - Arithmetic result
     - Memory address for load/store
     - Branch target address
   - Access data memory for load/store
   - PC ← target address or PC + 4

Problems to Solve

4.1, 4.7