

Inf2C - Computer Systems

Lecture 12

Processor Design – Single Cycle

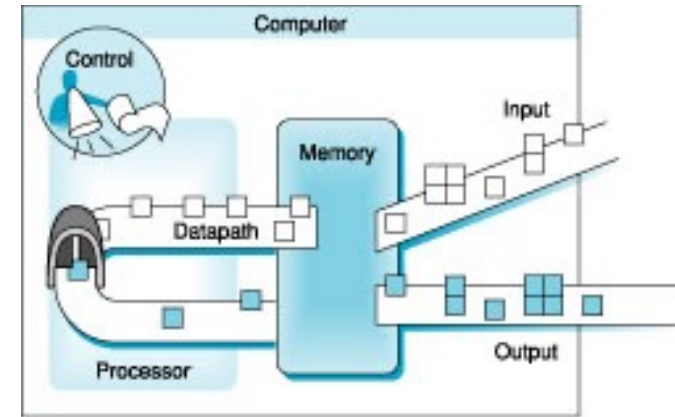
Vijay Nagarajan

School of Informatics
University of Edinburgh



Lecture 9: Processor design – single cycle

- Motivation:
 - Learn how to design a simple processor
- Two main parts:
 - Datapath: performs the data operations as commanded by the program instructions
 - Control: controls the datapath, memory and I/O according to the program instructions
- Using:
 - Combinational and sequential circuits described in previous lectures



Design steps / Lecture outline

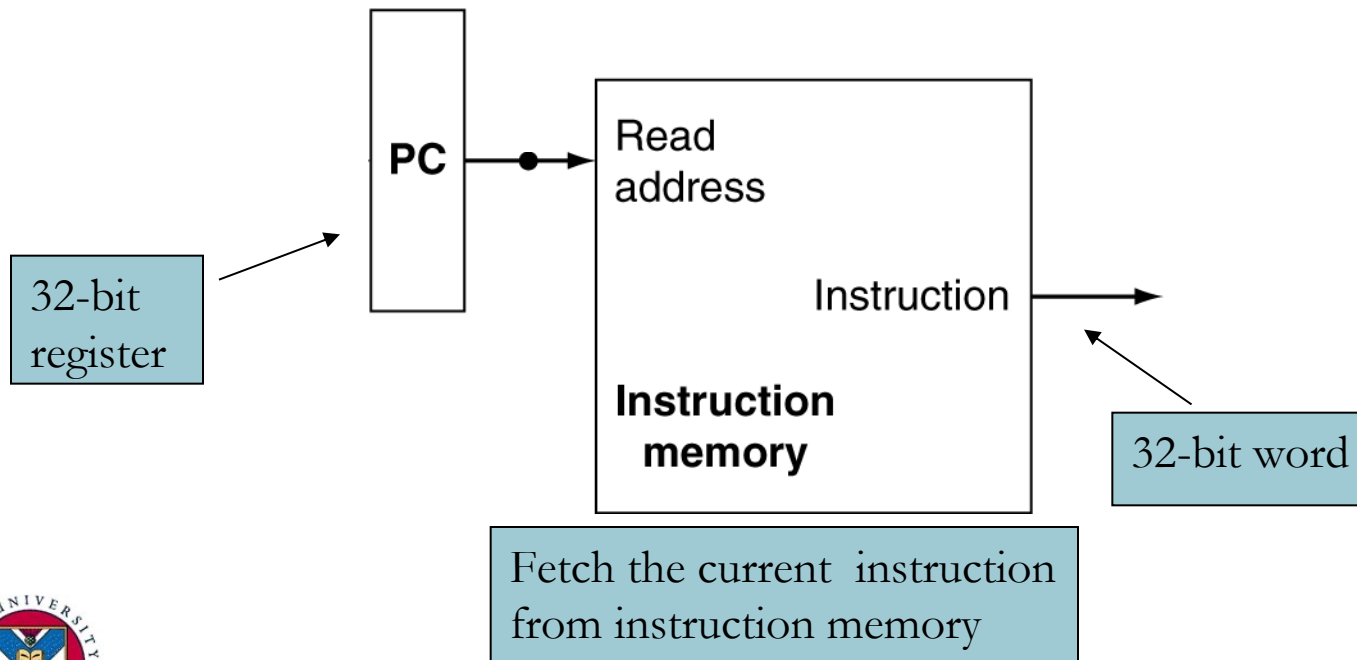
- Step 1: Determine the components required by understanding main processor functions
- Step 2: Build the datapath
- Step 3: Build the control

- Show the execution of a few instructions on the designed machine

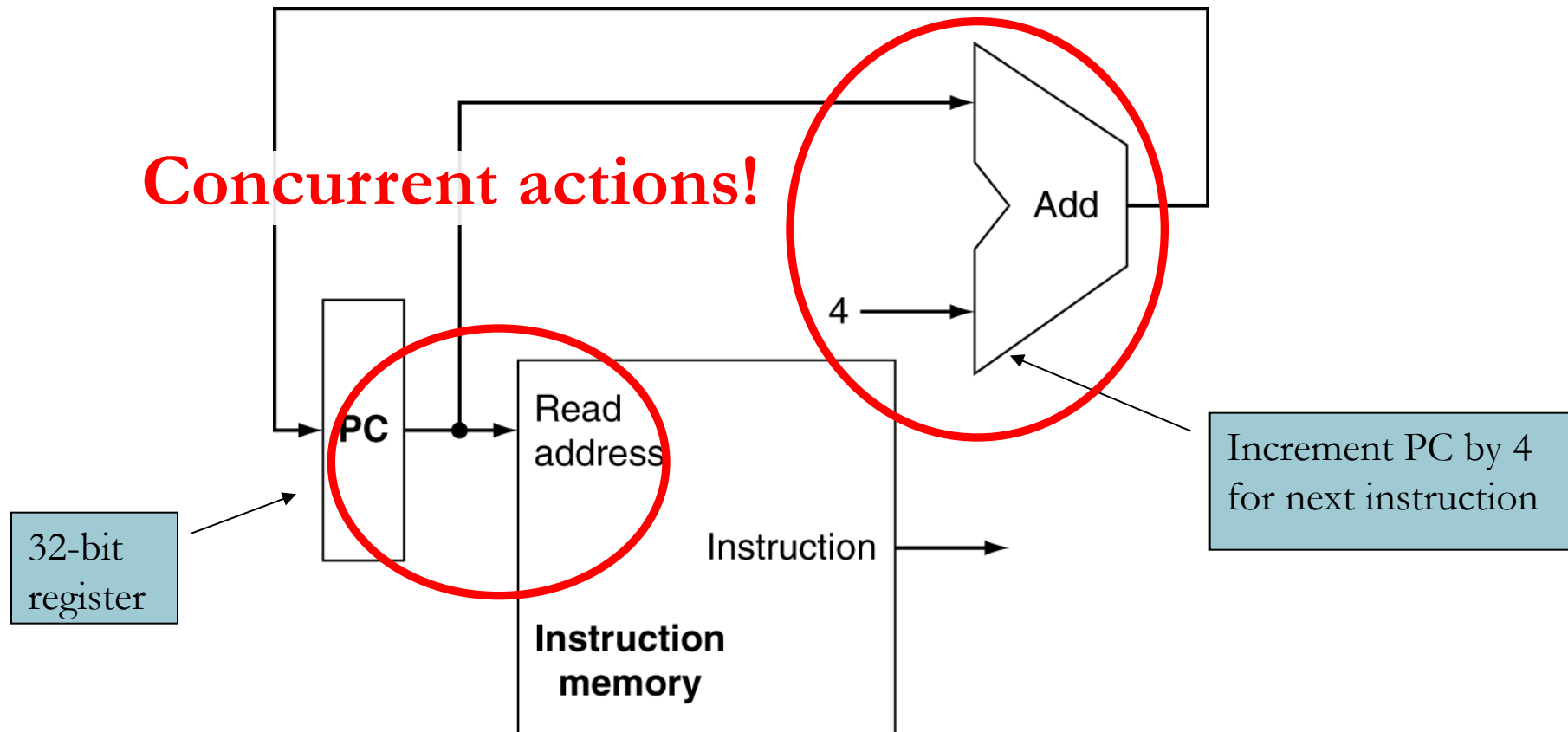
Main processor functions

- **Fetch** instruction from **instruction memory**
- Read the **register** operands
- Use the **ALU** for computation
 - Arithmetic, memory address, branch target address
- Access **data memory** for load/store
- Store the result of computation or loaded data into the destination **register**
- Update the **Program Counter (PC)**

Instruction Fetch: common to all insts



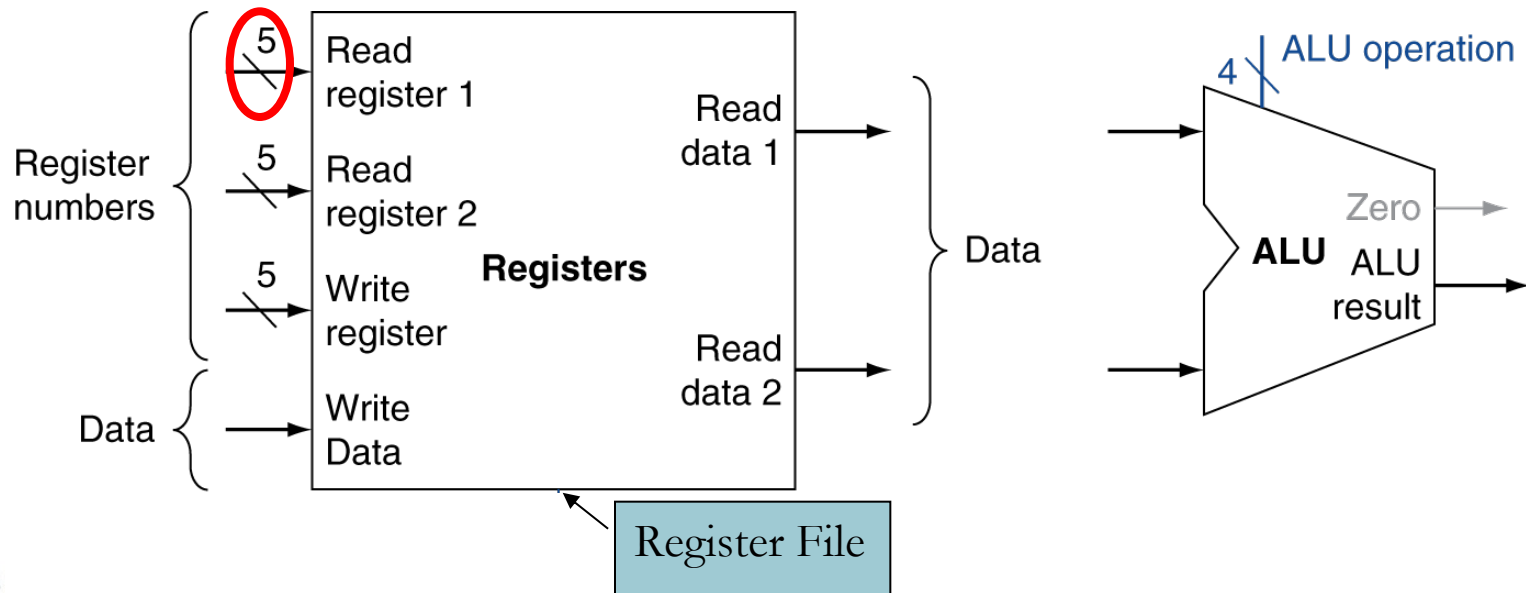
Instruction Fetch: common to all insts



R-Format Instructions

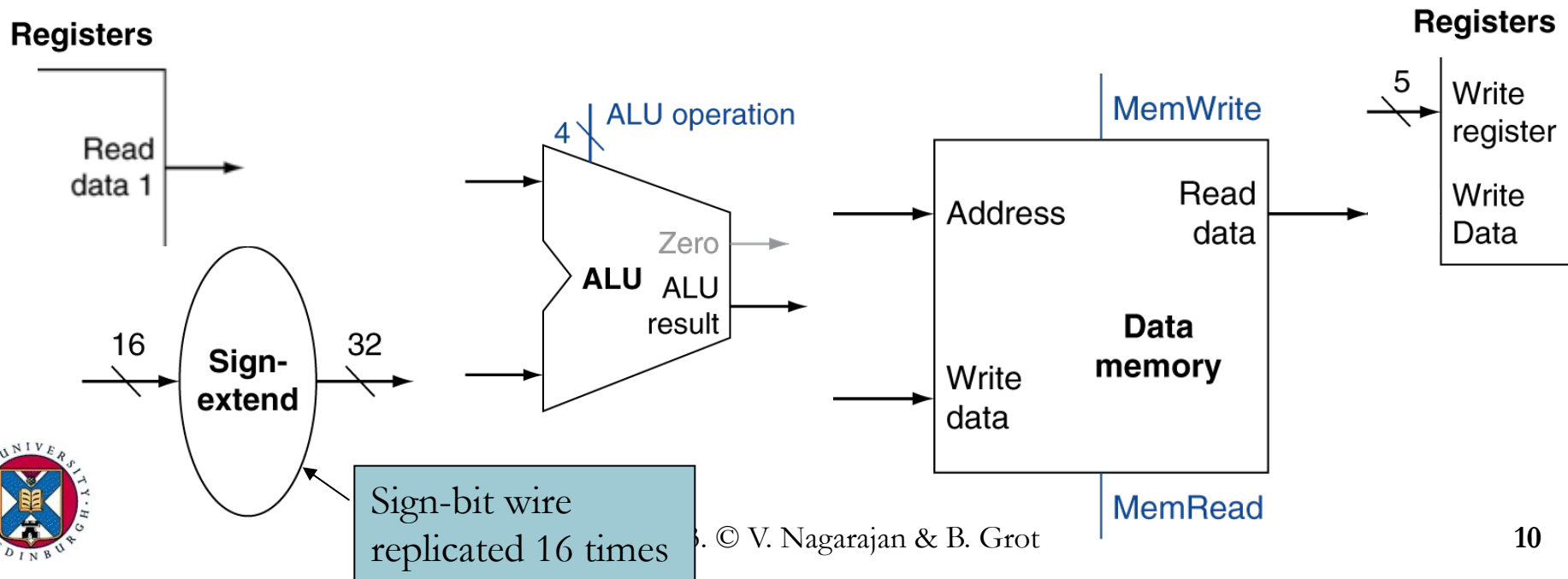
1. Read two register operands
2. Perform arithmetic/logical operation
3. Write register result

A bundle of N wires ($N=5$ here)



Load/Store Instructions

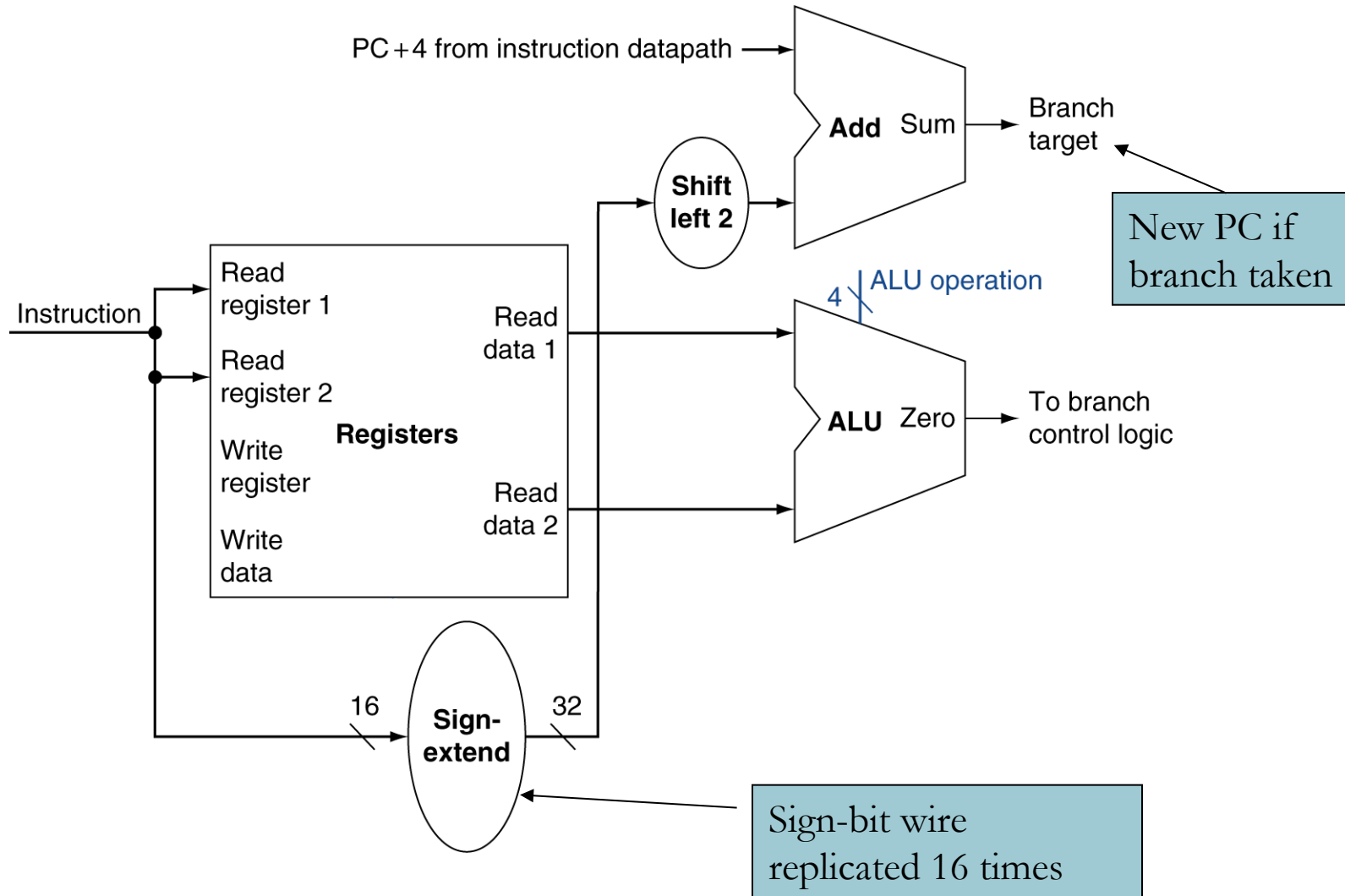
1. Read register operands
2. Calculate address using 16-bit offset
 - Use ALU, but first, sign-extend the offset
3. Read (for load) or write (for store) the memory
4. Load only: update destination register



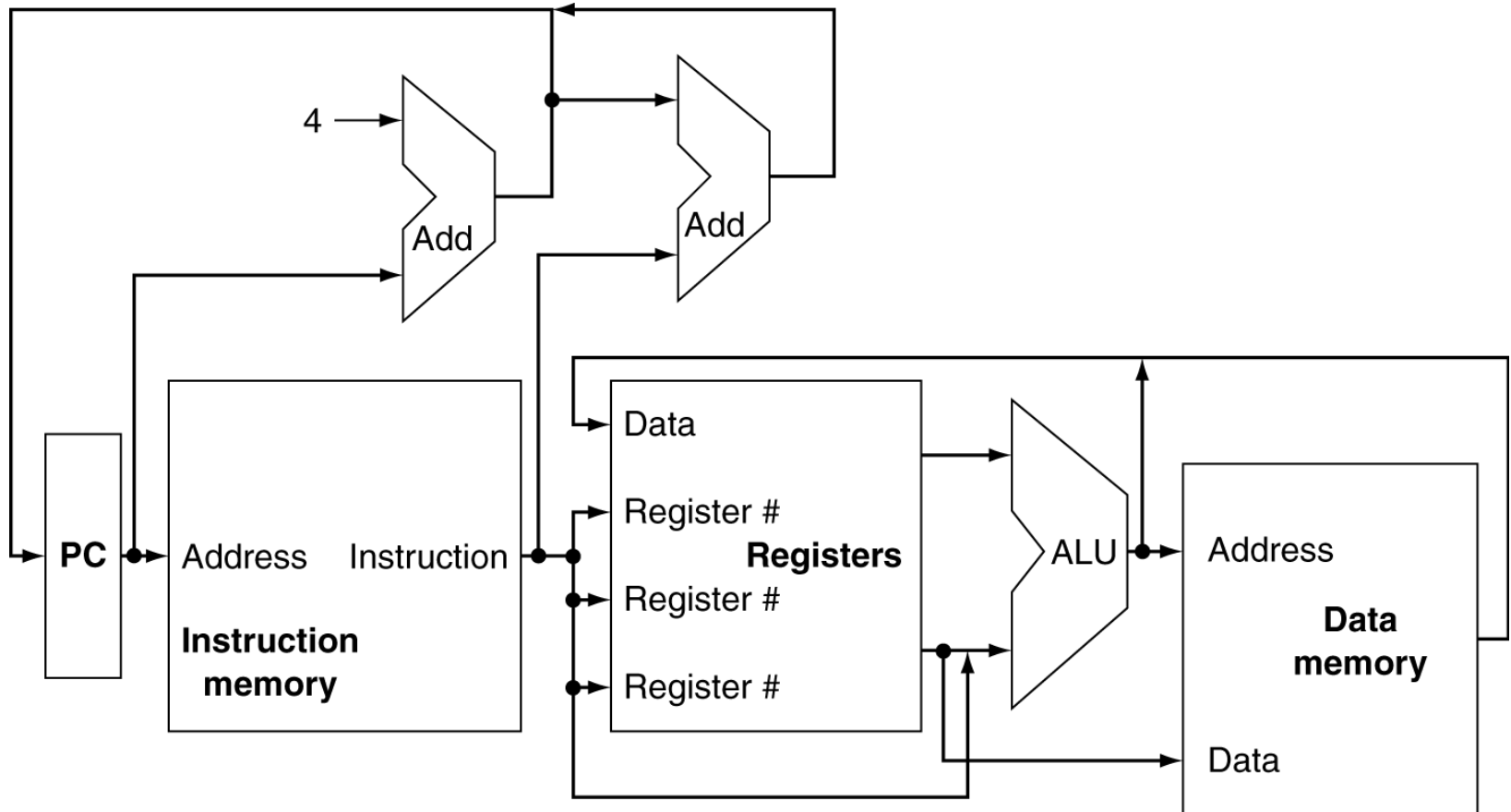
Branch Instructions (BEQ, BNE)

1. Read register operands
2. Compare operands
 - Use ALU, subtract and check ALU's **Zero** output
3. Calculate target address
 - Sign-extend the immediate (offset)
 - Shift left 2 places (word align)
 - Add to PC + 4
 - Already calculated by instruction fetch

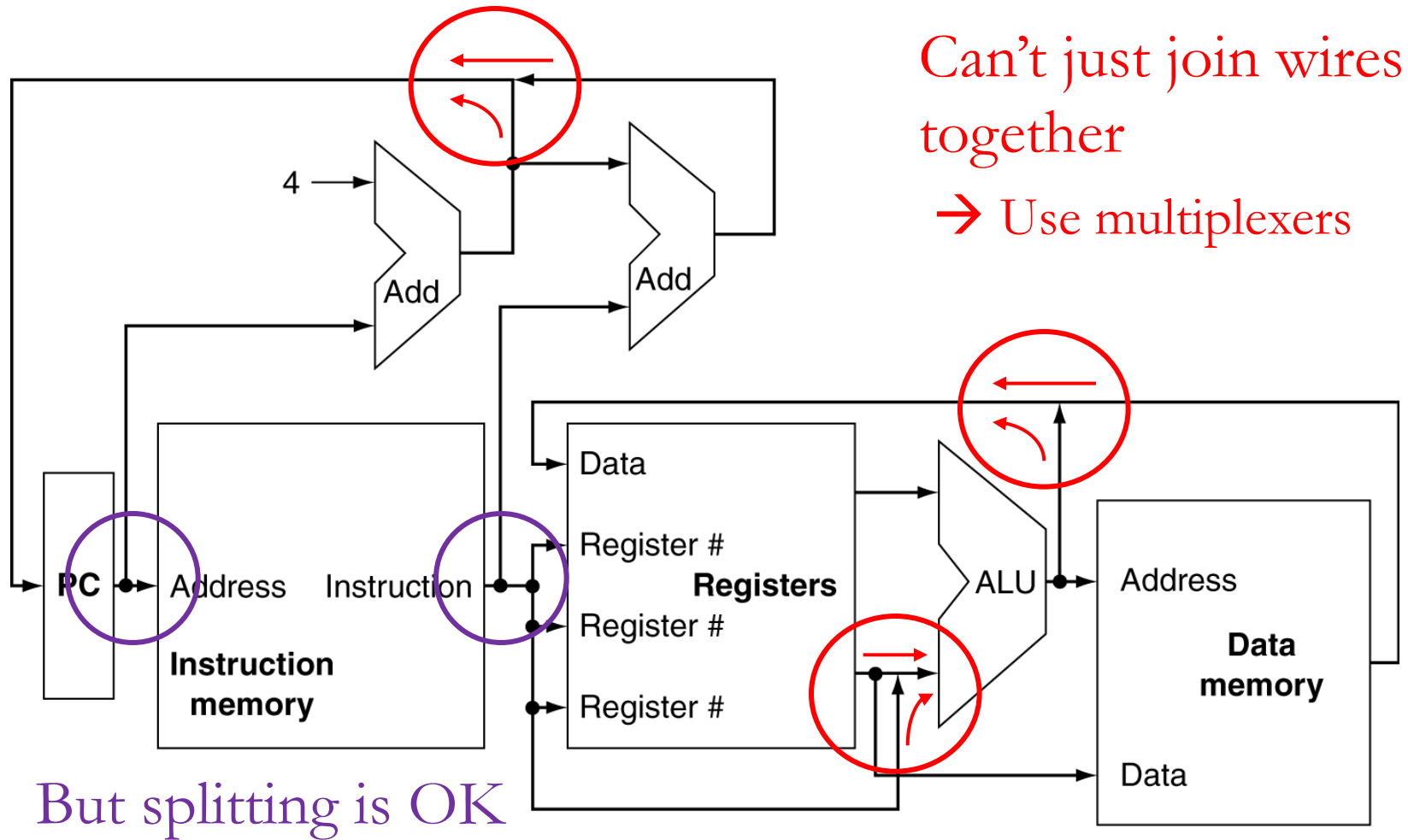
Branch Instructions



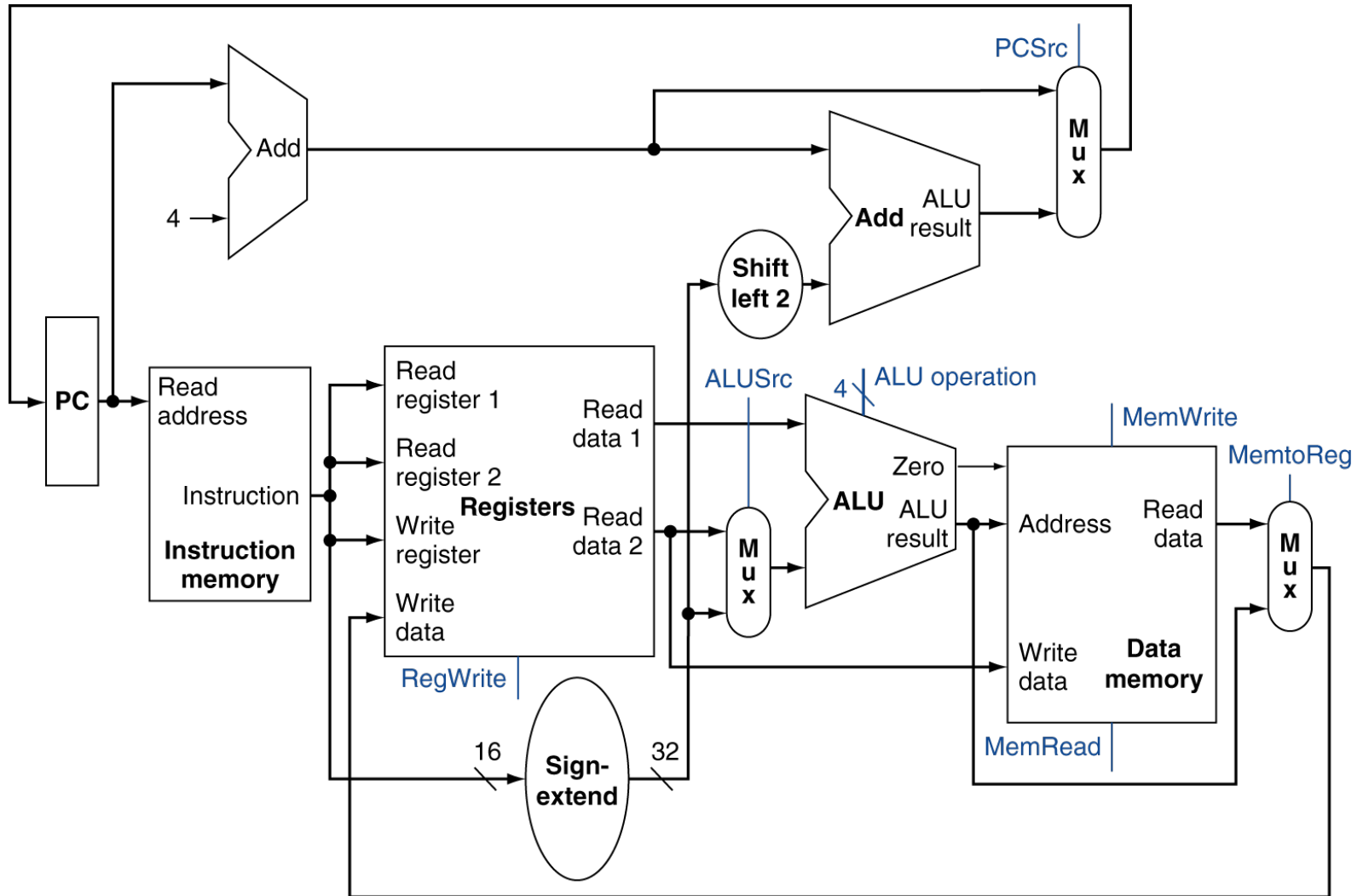
Putting It Together: Simplified Datapath



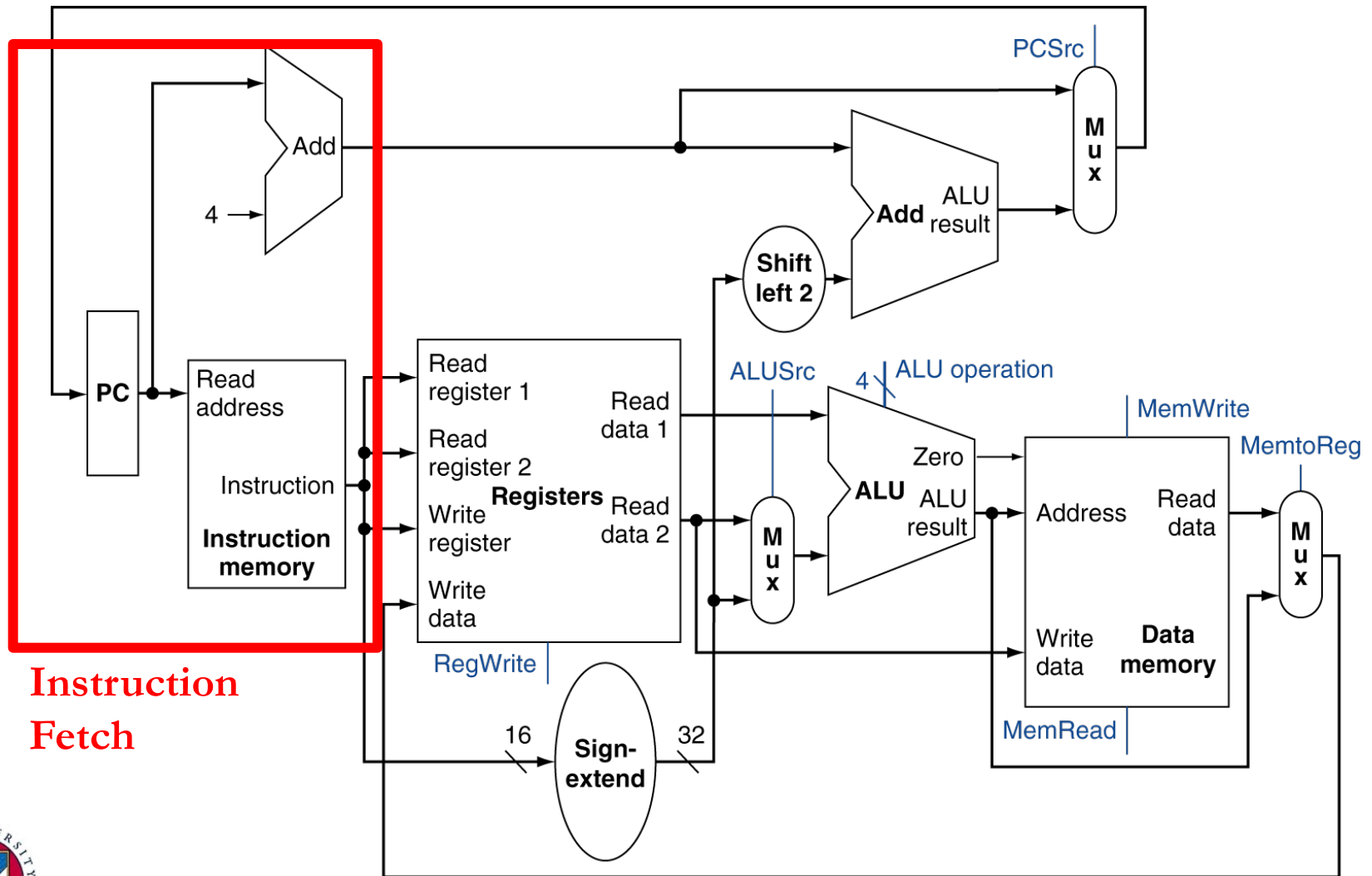
Putting It Together: Simplified Datapath



Full Datapath



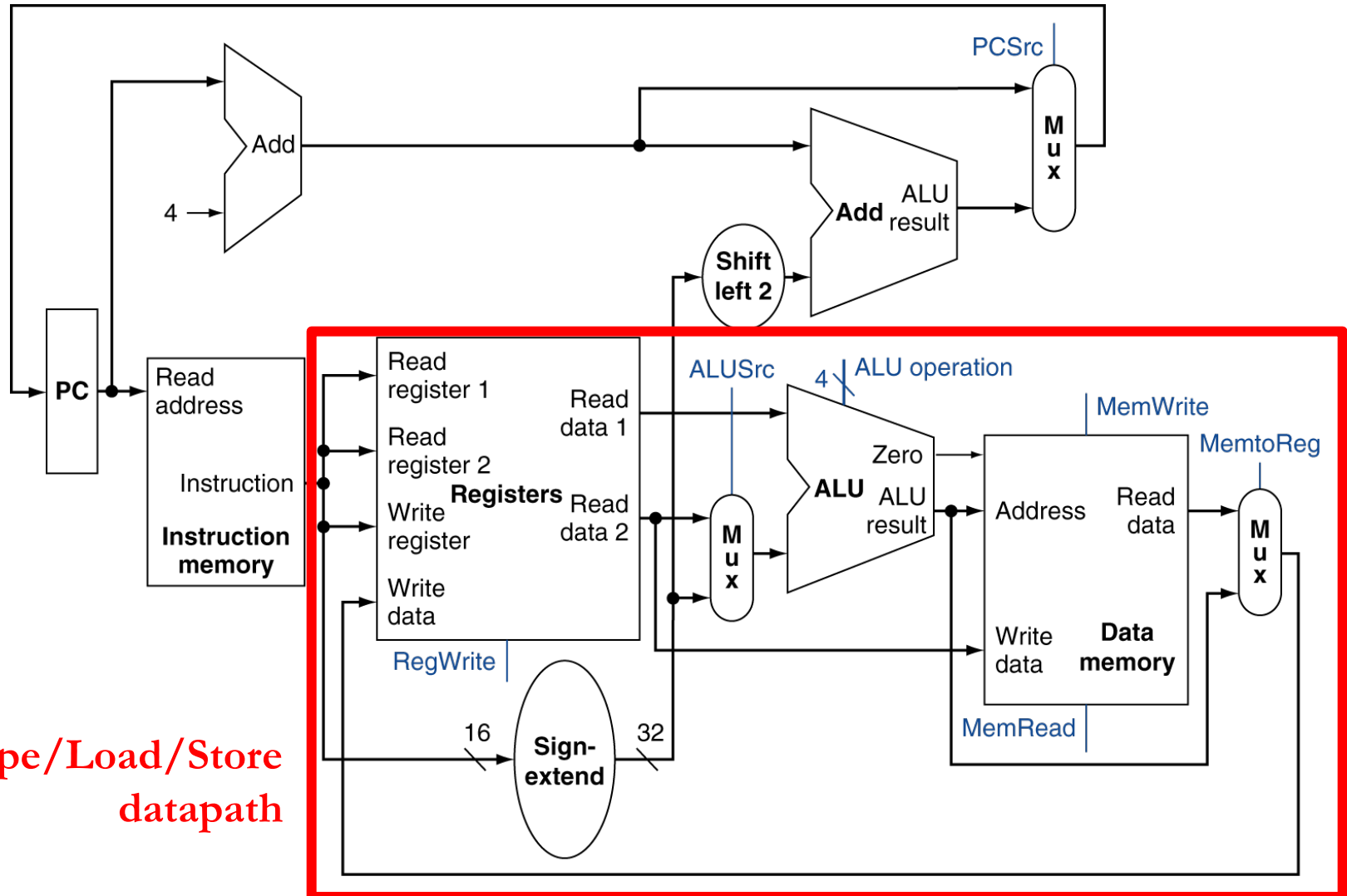
Full Datapath



Instruction
Fetch



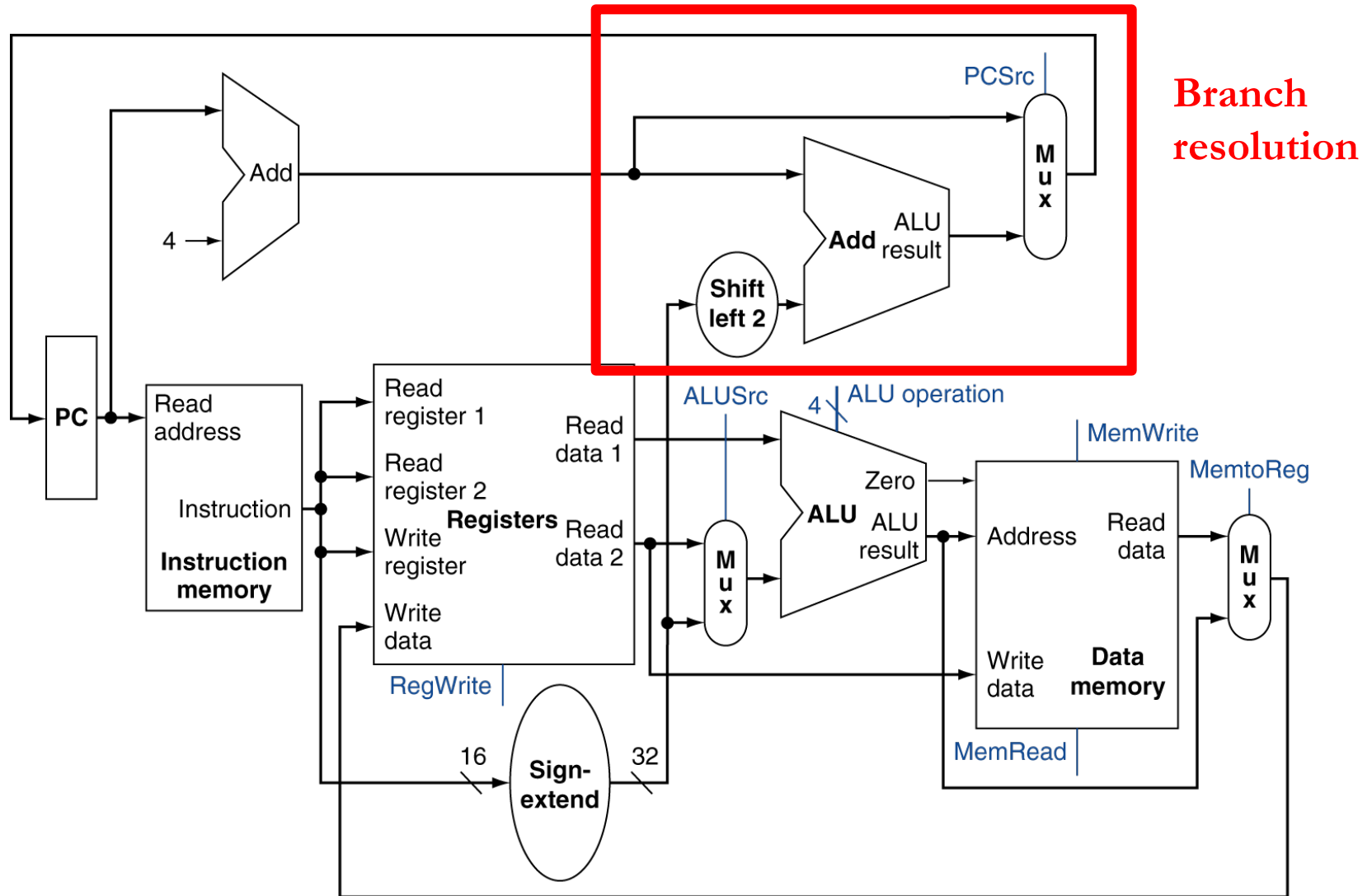
Full Datapath



**R-type/Load/Store
datapath**



Full Datapath

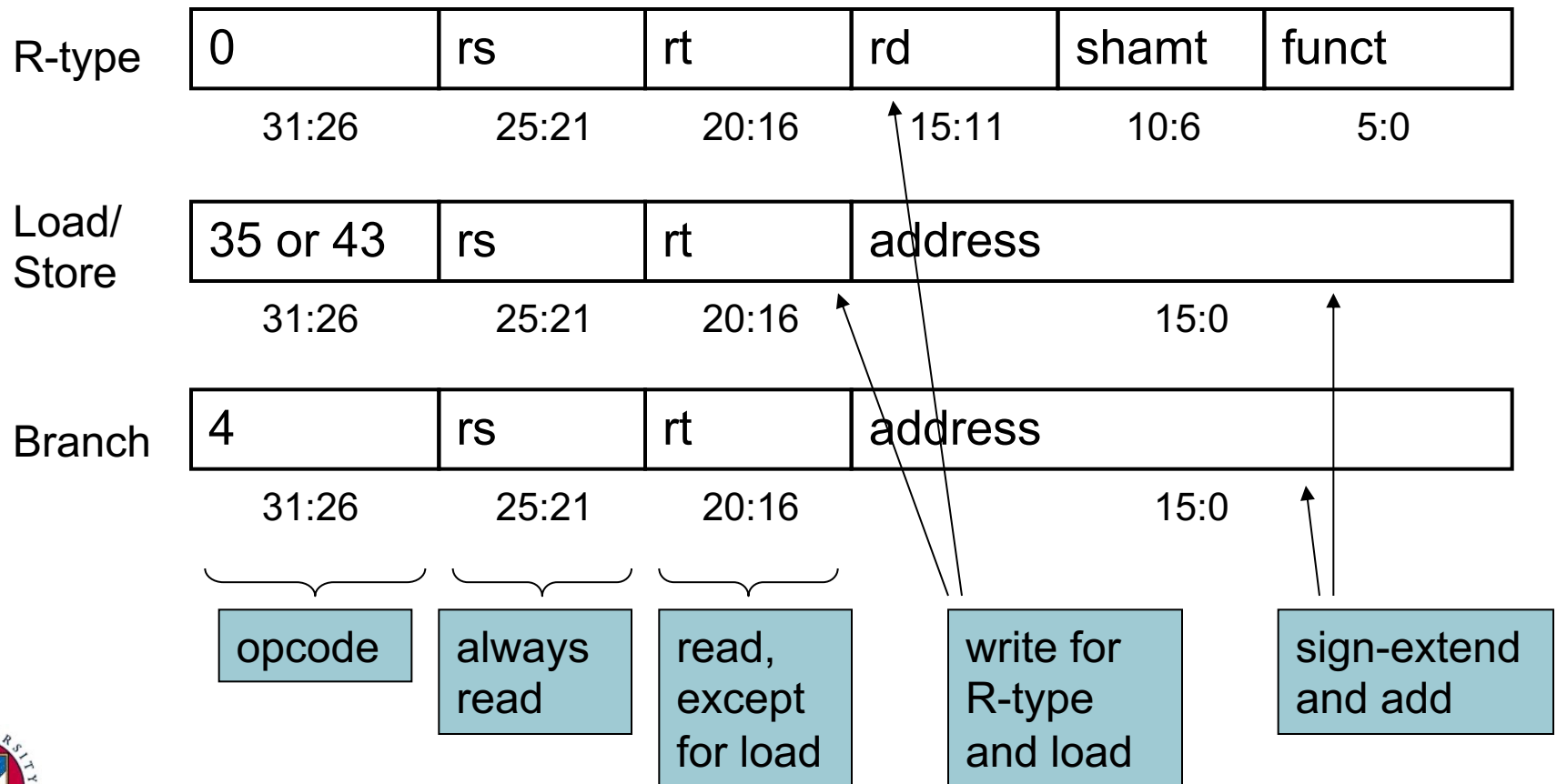


How to design the control part

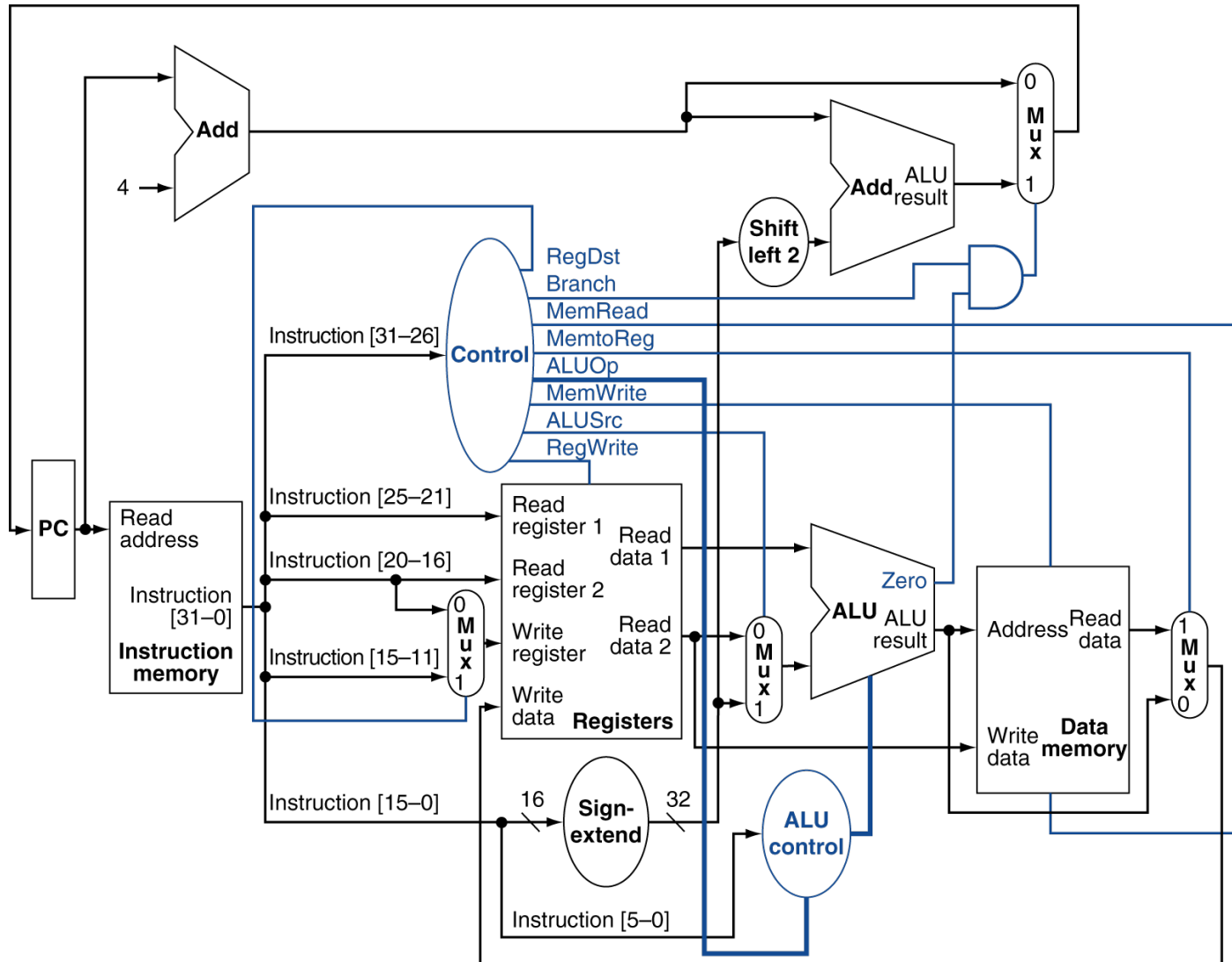
- For all control signals determine which value selects what operation, input, etc.
- Make truth table of control signal values for each instruction, or instruction group
- Convert table to combinational circuit

Designing the Main Control Unit

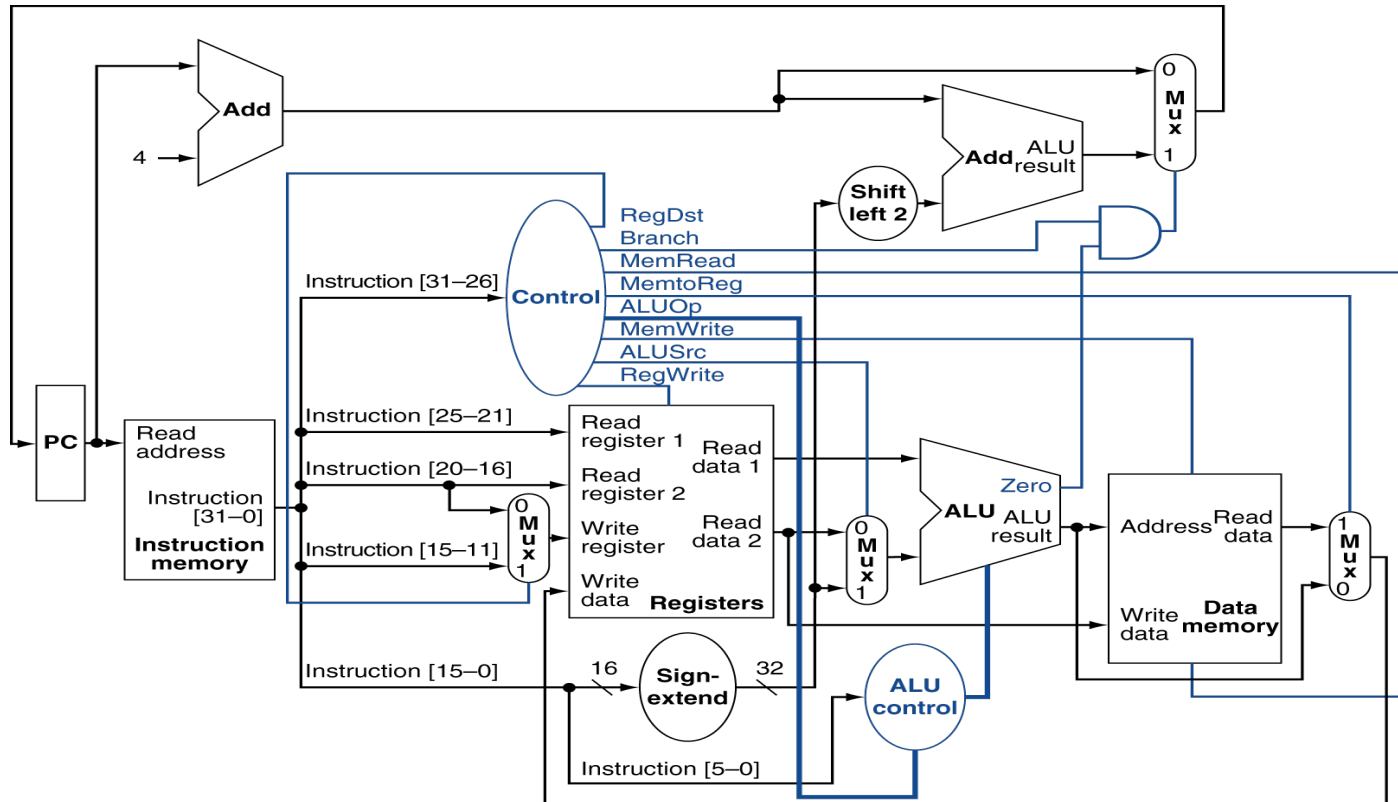
Control signals derived from instruction fields



Datapath with Control



Datapath and control truth table



Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1



Don't Care

ALU control

ALU operation:

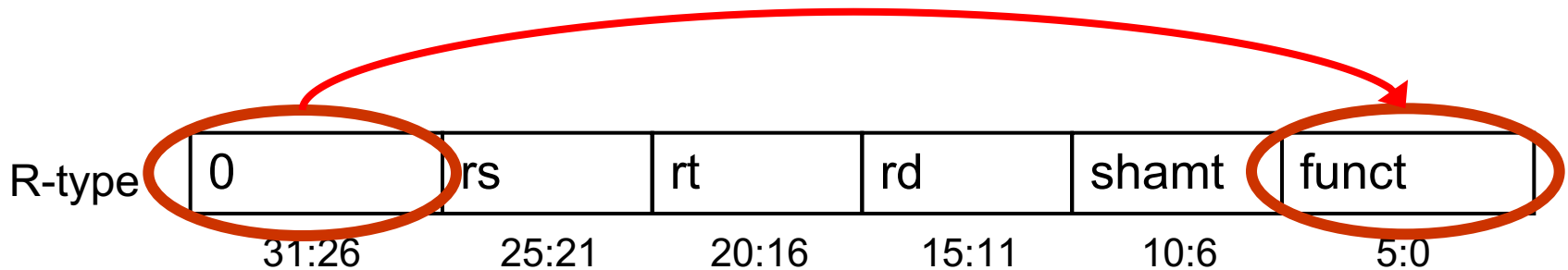
- Data transfers (ld/st)
- Branches
- All other

ALU control

ALU operation:

- Data transfers (ld/st) – add
- Branches – sub
- All other – determined by **funct** field, I[5:0]

derived
from
opcode



ALU control

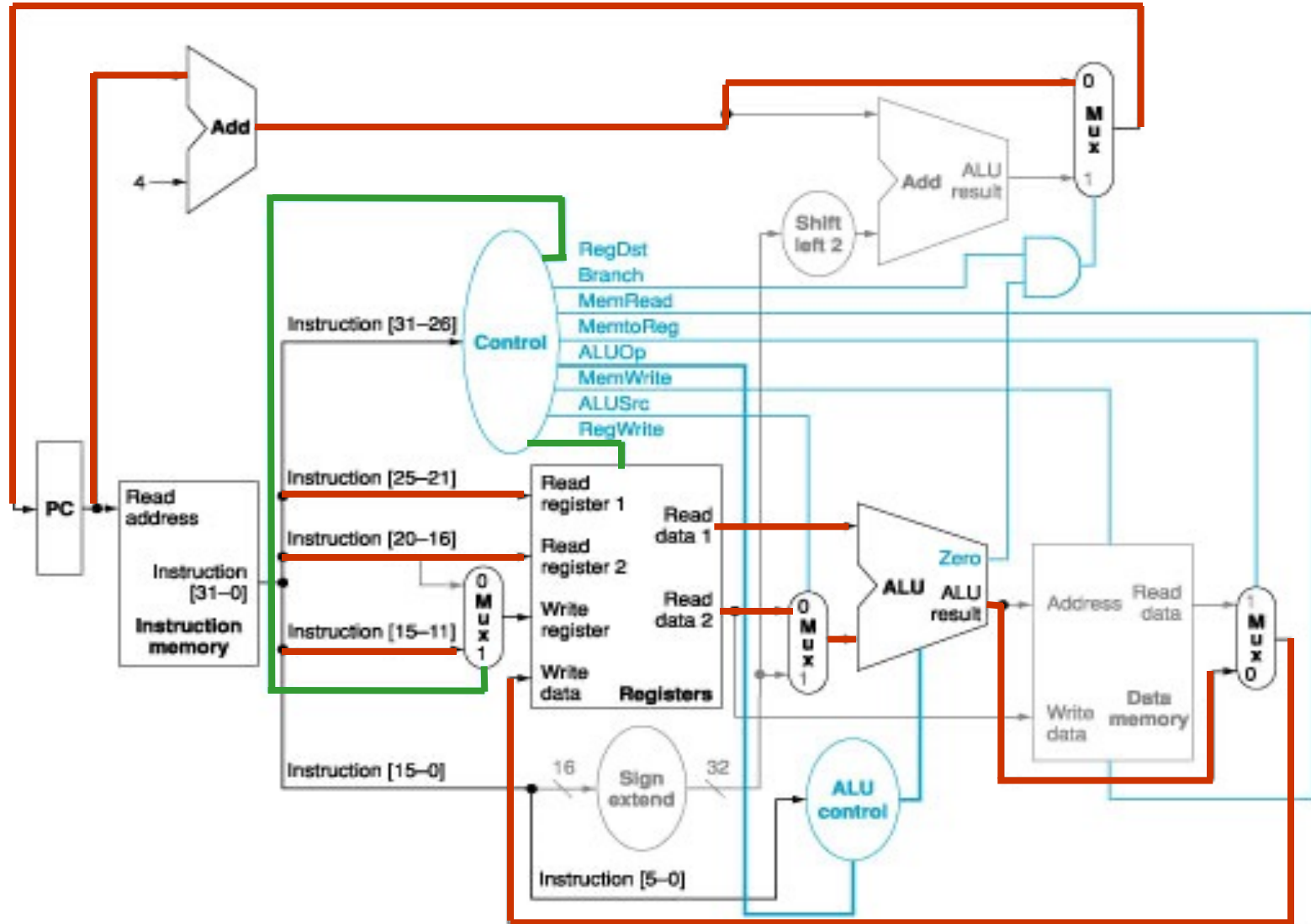
ALU control is **hierarchical**:

- Main control specifies which of the 3 op types
 - Add, Sub, or based on Funct bits
- Second level provides actual ALU control signal

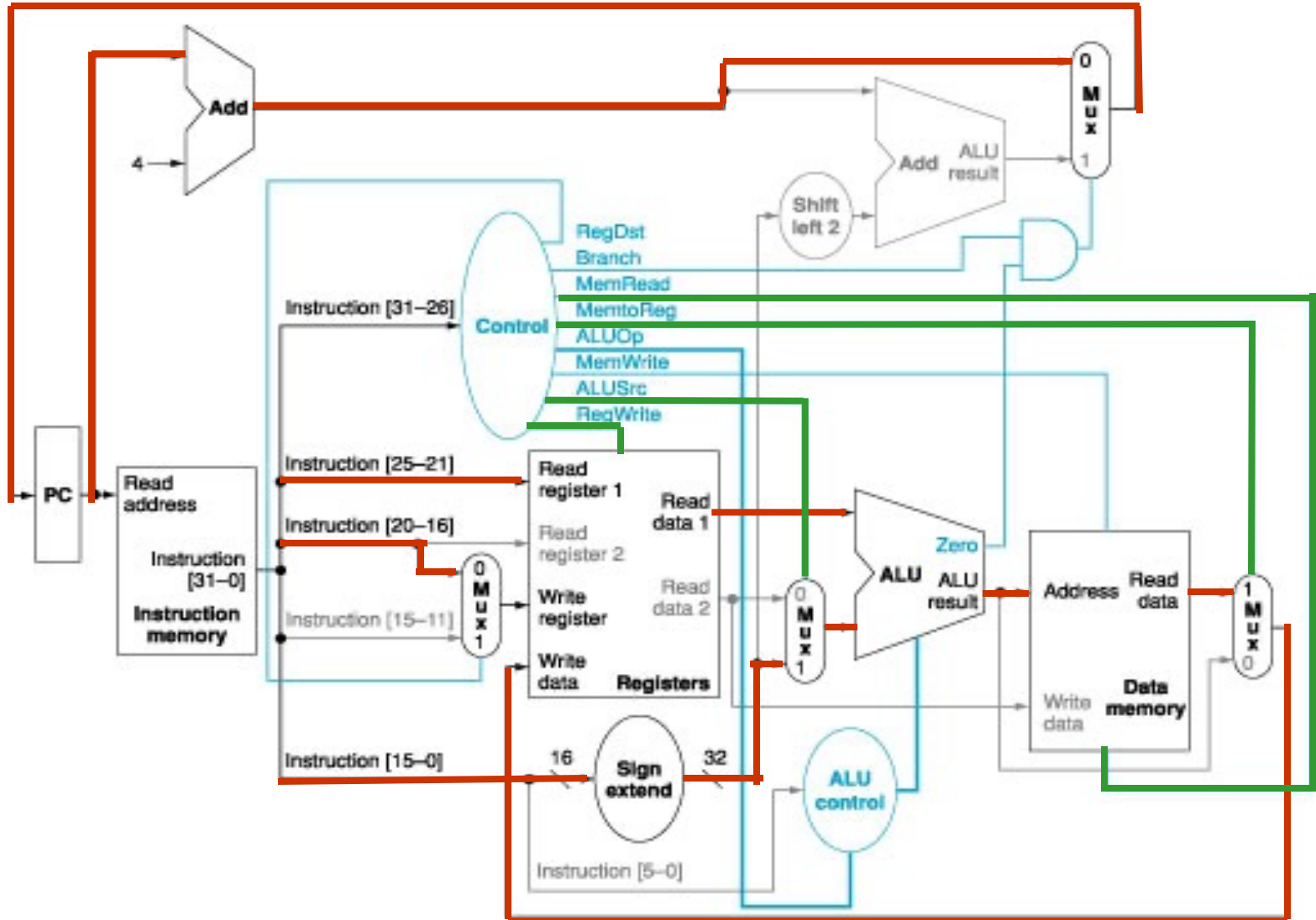
ALU operation	
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less than

} actual
ALU Control
signals

R-type instruction execution



lw execution



beq (taken) execution

