The Processor Data Path and Control

Chapter 4 Single Cycle

The Big Picture: The Performance Perspective

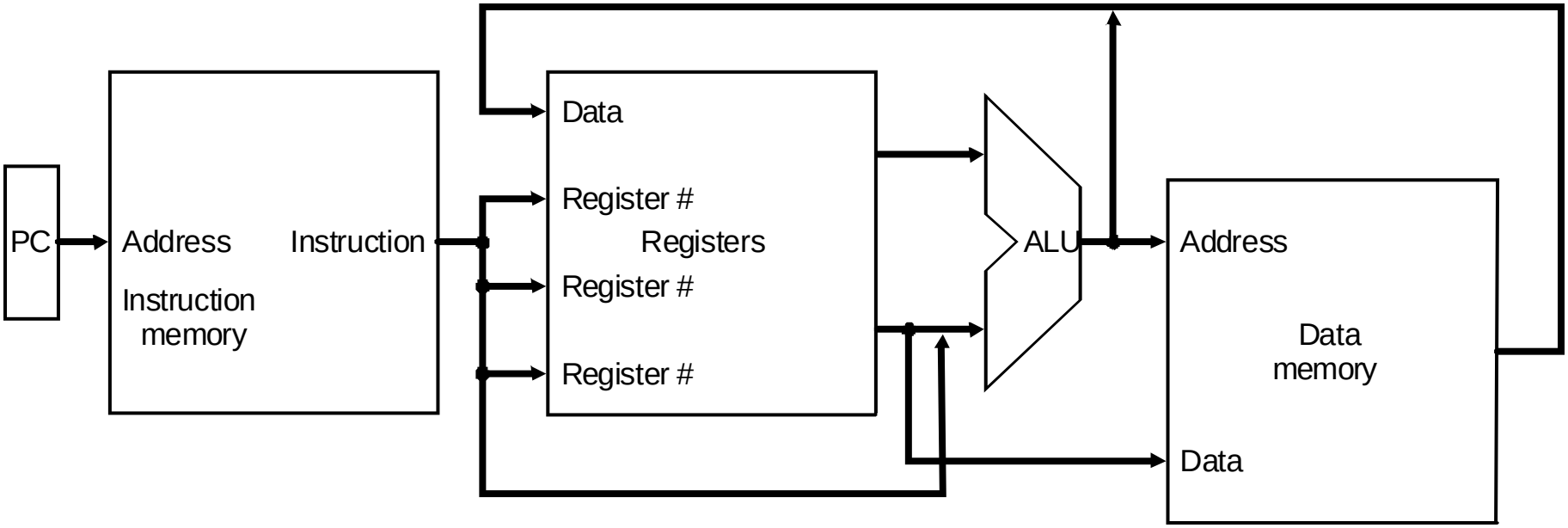
- Performance of a machine is determined by:
  - Instruction count
  - Clock cycle time
  - Clock cycles per instruction
- Processor design (datapath and control) will determine:
  - Clock cycle time
  - Clock cycles per instruction
- Today:
  - Single cycle processor:
    - Advantage: One clock cycle per instruction
    - Disadvantage: long cycle time

Introduction

- Building blocks of the processor
- Implementation of a subset
  - Load & store word
  - Arithmetic & logic instructions (add, sub, and, or, slt)
  - Branch equal & jump
- Logic and clocking
  - Edge triggered
  - New value at the rising edge of the clock
  - Period determined by the longest module
  - Registers: automatically updated, except those with an explicit write signal

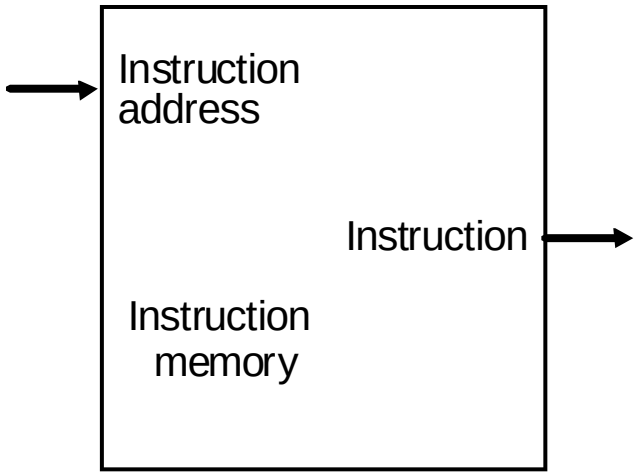
The abstract view

- Program counter: next instruction
- Instruction memory
- Register file
- Data memory
- Bus structure: information transfer

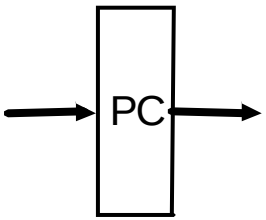


Building the data path

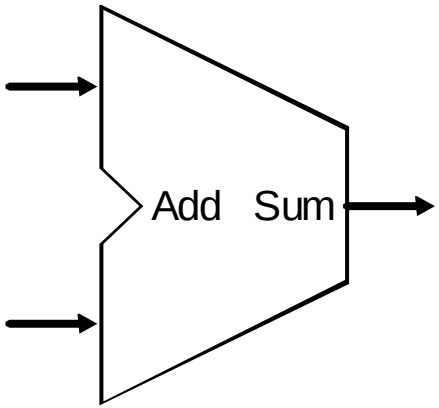
- Instruction memory
  - Contains only instructions
- Program counter
  - points to the next instruction
- Arithmetic & logic unit
  - Here: an adder only



a. Instruction memory



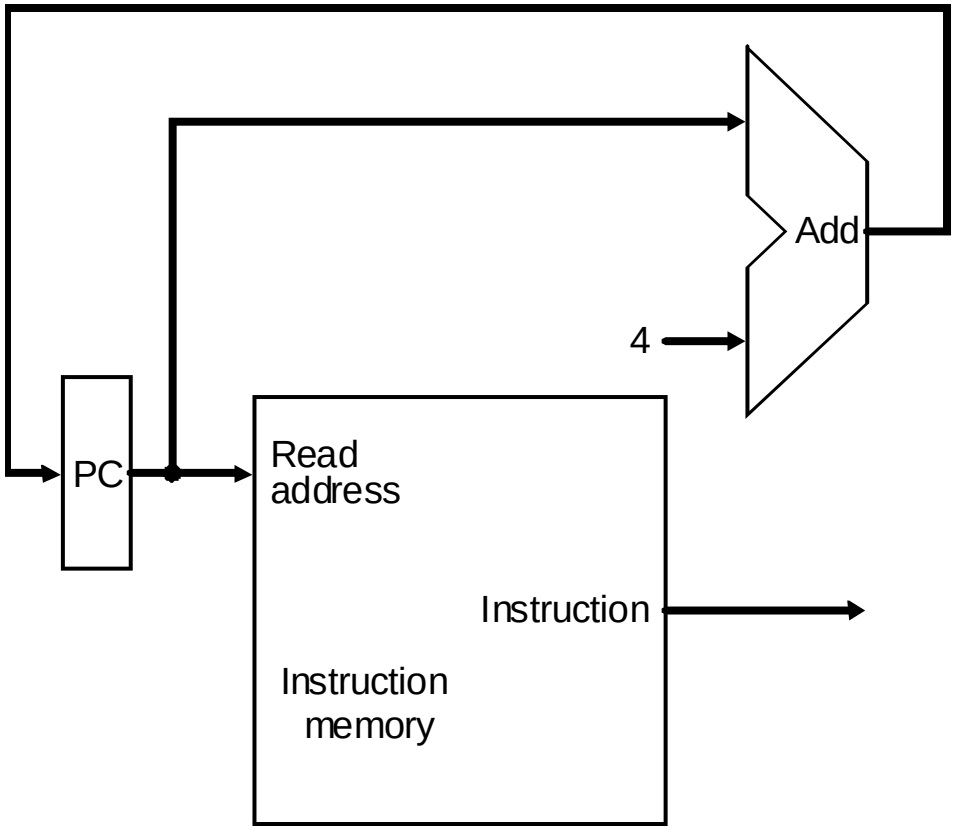
b. Program counter



c. Adder

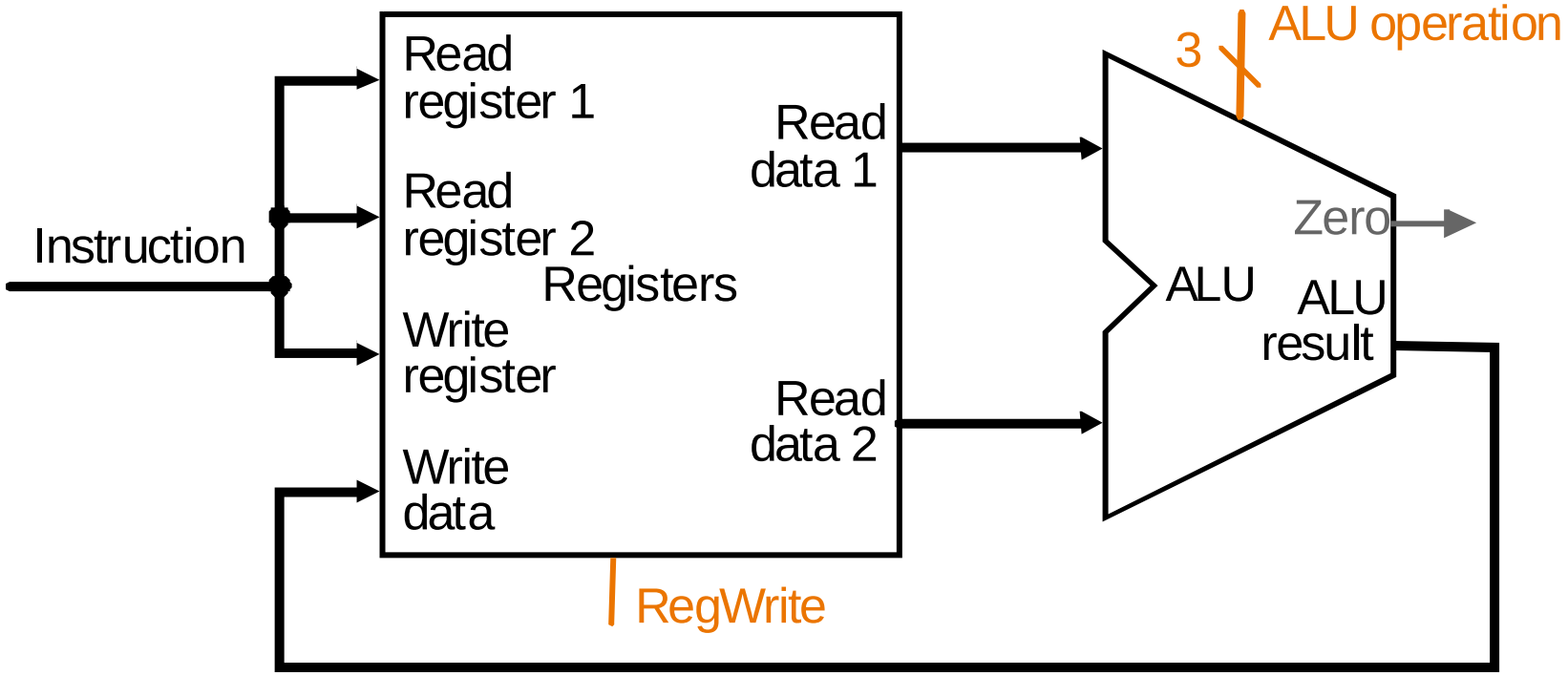
Instructions & PC

- Increment: add 4 to the PC to get to the address of the next instruction
- Needs extension for Branch and Jump



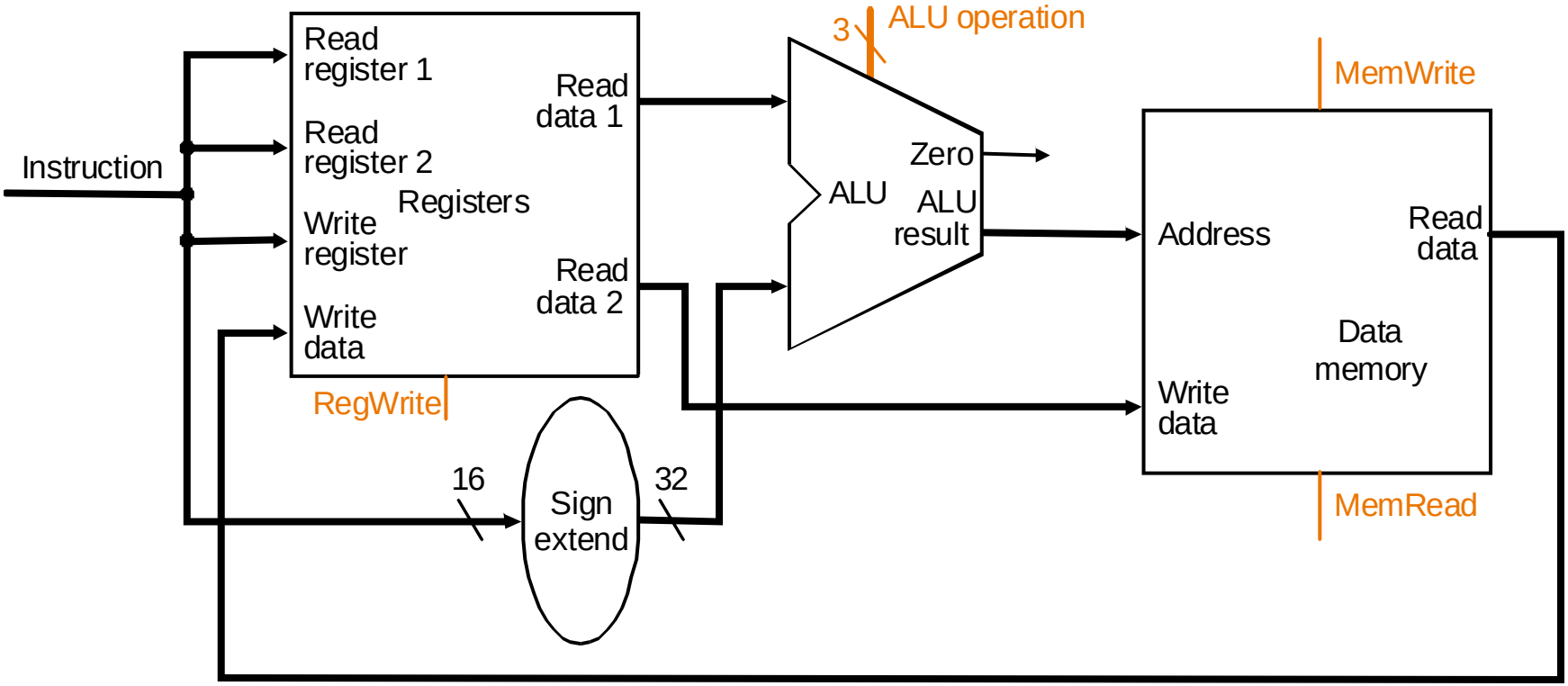
Registers and ALU

- Register file for R-format instructions as discussed previously
- ALU as designed in the previous chapter



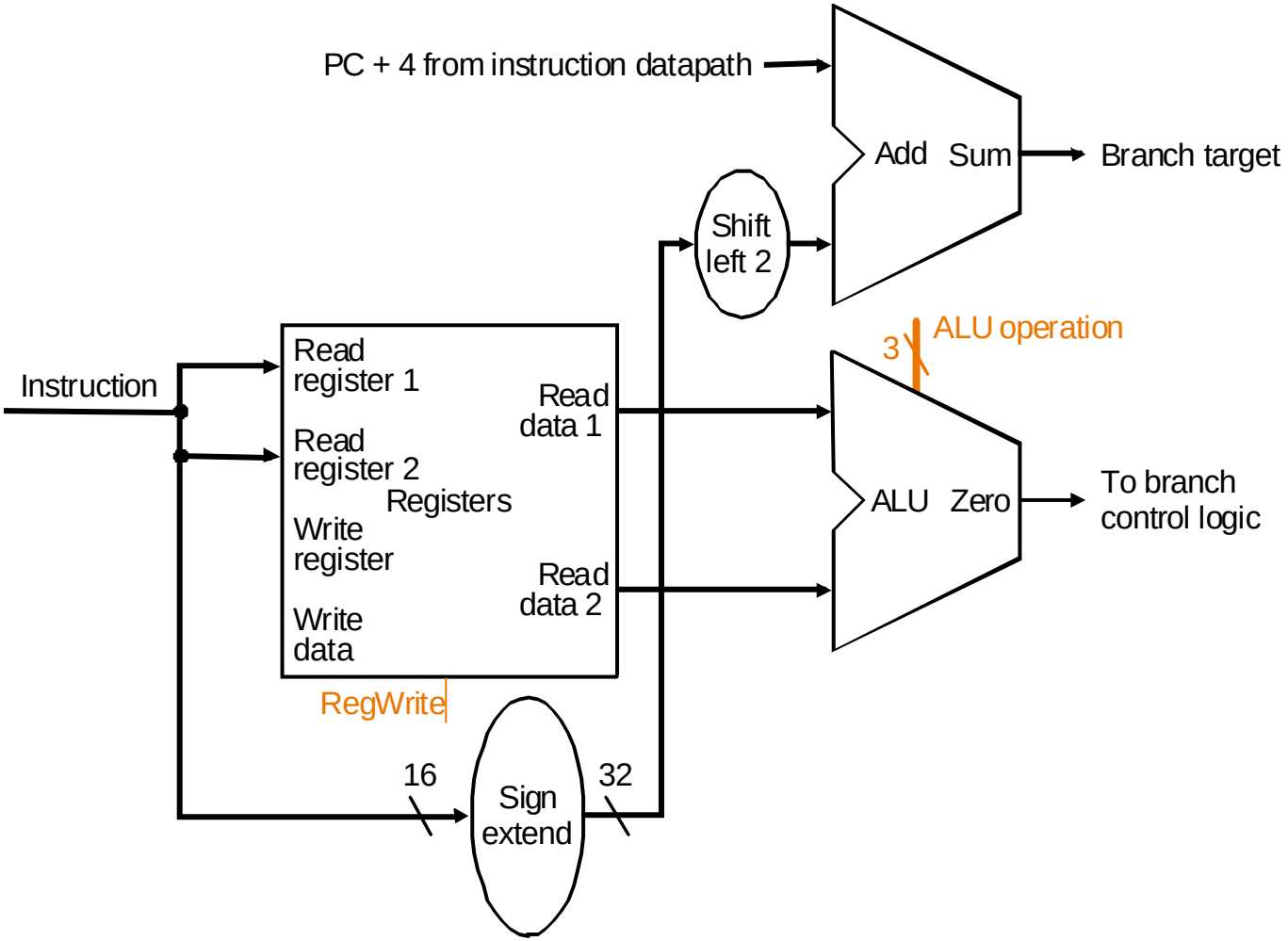
Adding the data memory

- Transfer address to the memory
- Transfer data to the register file
- Sign extension unit for address calculation



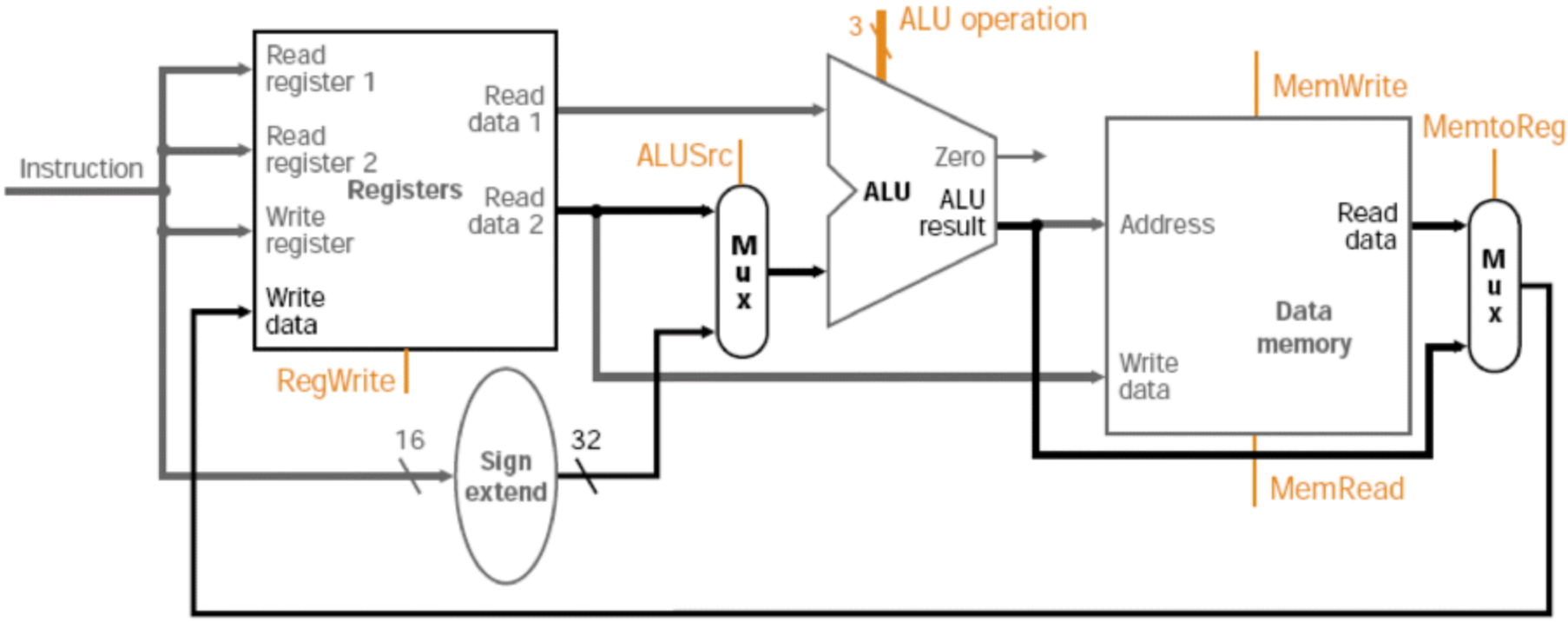
Branch data path

- PC + 4 + sign extended & shifted address



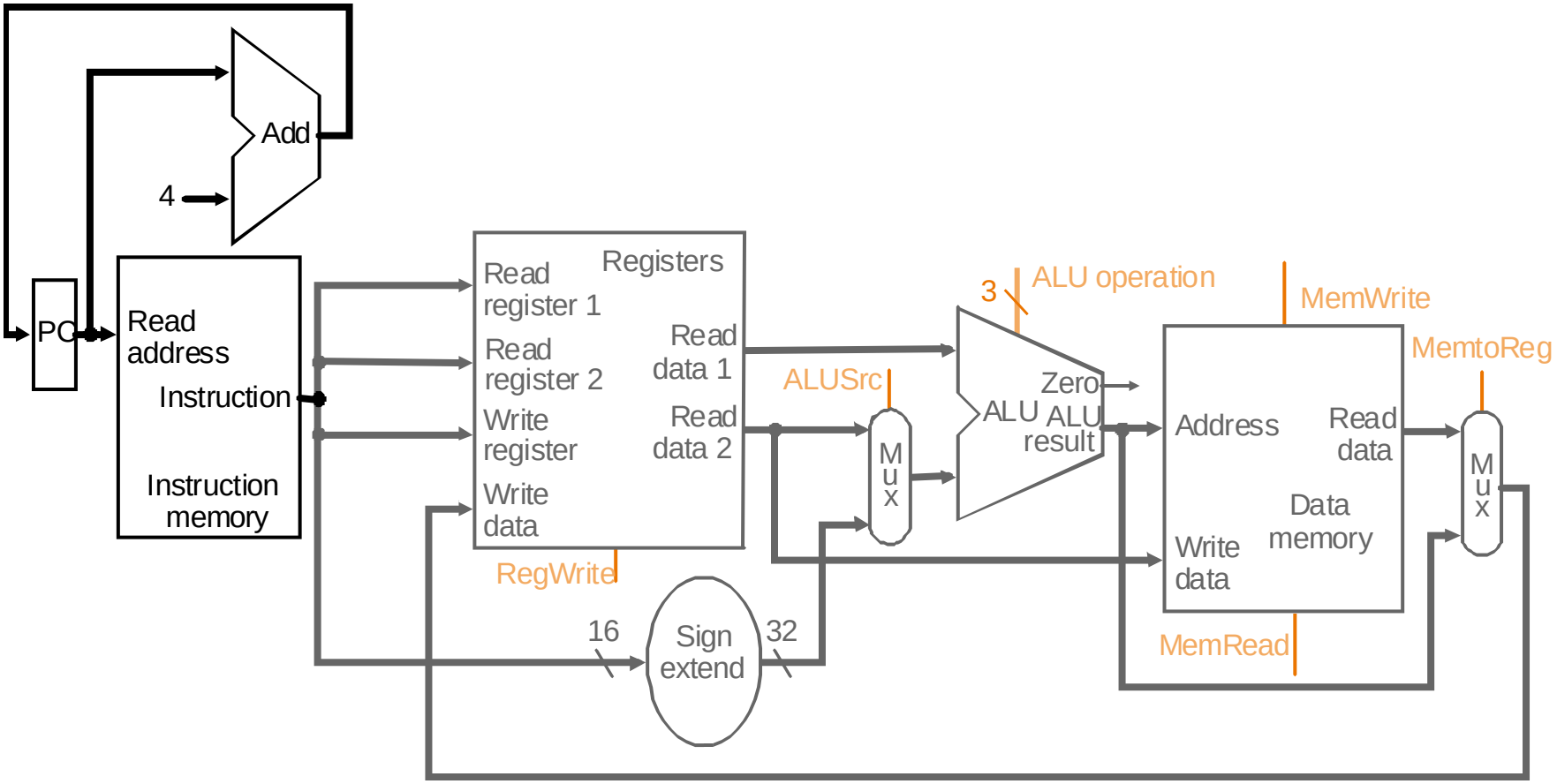
Putting things together

- Register set
- ALU for operand and address calculation
- Data memory



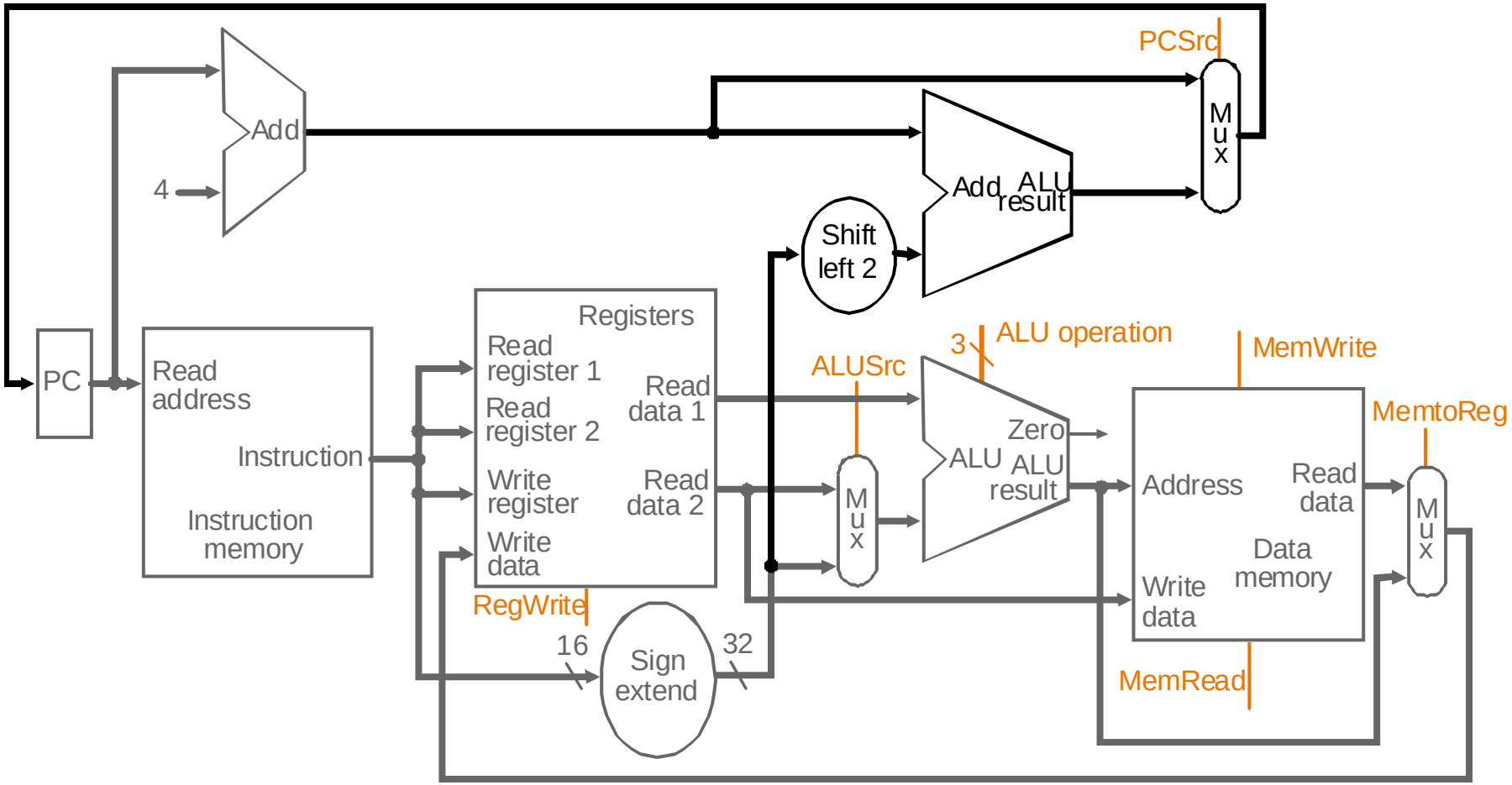
Adding the instruction fetch unit

- The instruction fetch unit provides the instruction



Adding branch logic

- Supports now all basic instructions



Basic control

- Review of ALU functions / control lines
- ALU control bits: origin of
  - 00: Addition (load & store word)
  - 01 : Subtraction (branch on equal)
  - 10 : Operations according the value in the function field (R-type instructions)

ALU Control lines	Function
000	And
001	Or
010	Add
110	Subtract
111	Set-on-less-than

ALU control

Instruction opcode	ALUOp	Instruction Operation	Funct Field	Desired ALU Action	ALU control input
LW	00	load word	XXXXXX	add	010
SW	00	store word	XXXXXX	add	010
Branch equal	01	branch equal	XXXXXX	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	and	000
R-type	10	OR	100101	or	001
R-type	10	set on less than	101010	set on less than	111

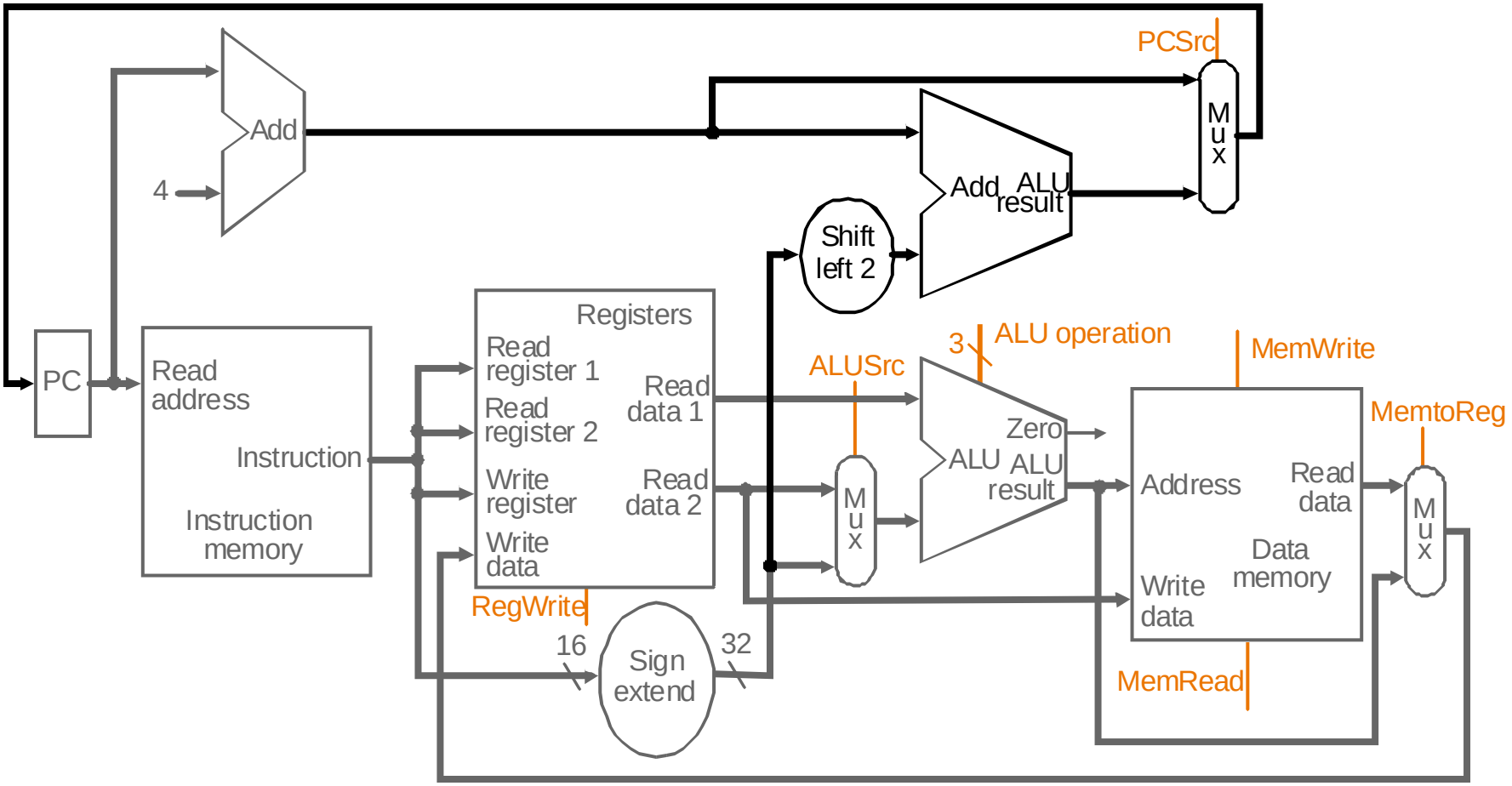
ALU Operation		Funct Field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Main control

- Source information: instruction
- Operation code: Op [31-26]
- Read registers: rs [25-21], rt [20-16]
- Base register (lw, sw): rs [25-21]
- Destination register
  - Load: rt [20-16]
  - R-type: rd [15-11]
  - Requires a multiplexor

Extended data path

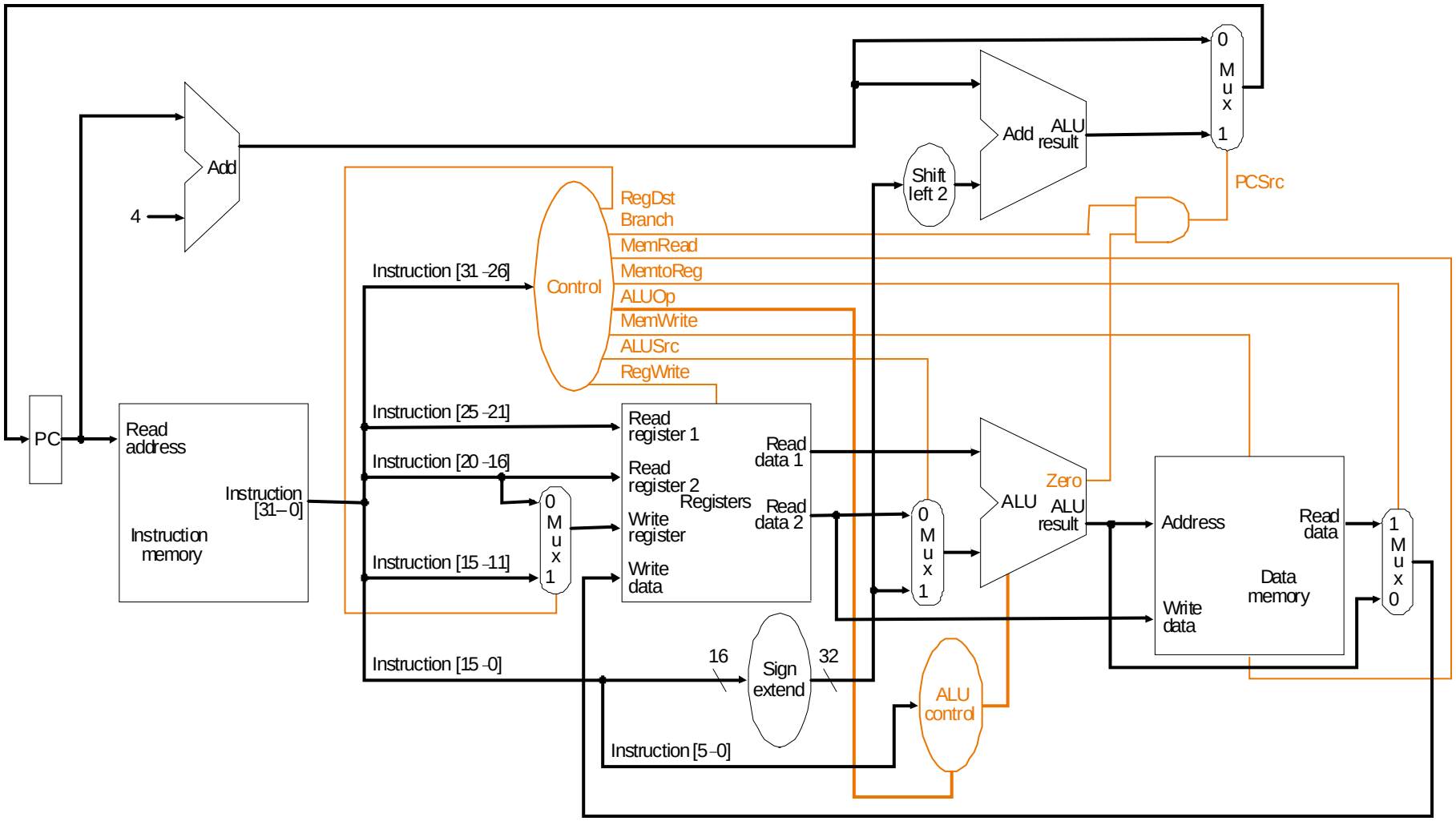
- Multiplexors for write register select
- All control lines



Summary of control lines

- RegDest
  - Source of the destination register for the operation
- RegWrite
  - Enables writing a register in the register file
- ALUsrc
  - Source of second ALU operand, can be a register or part of the instruction
- PCsrc
  - Source of the PC (increment [PC + 4] or branch)
- MemRead / MemWrite
  - Reading / Writing from memory
- MemtoReg
  - Source of write register contents

Data path & control

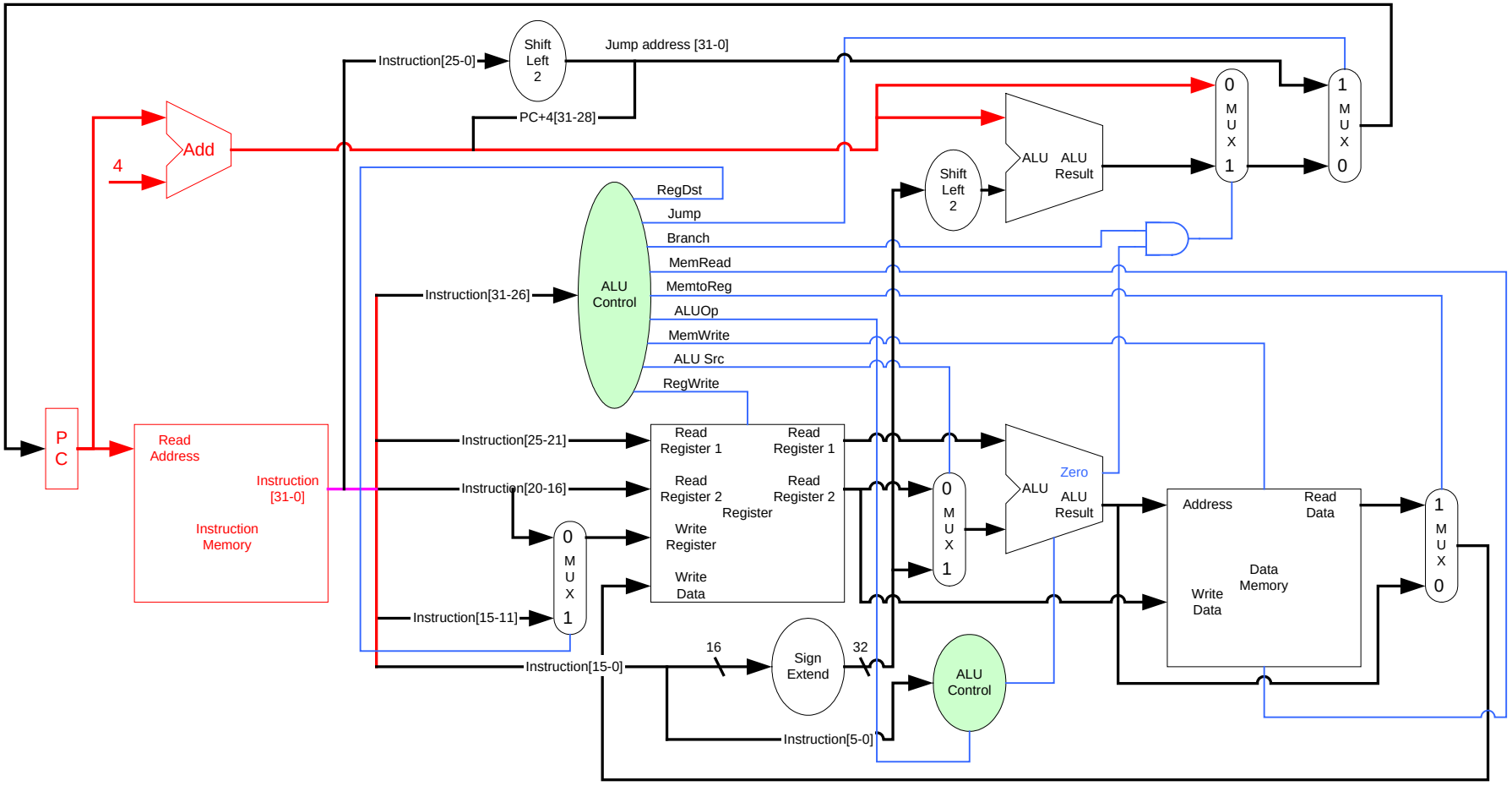


Operation of data path

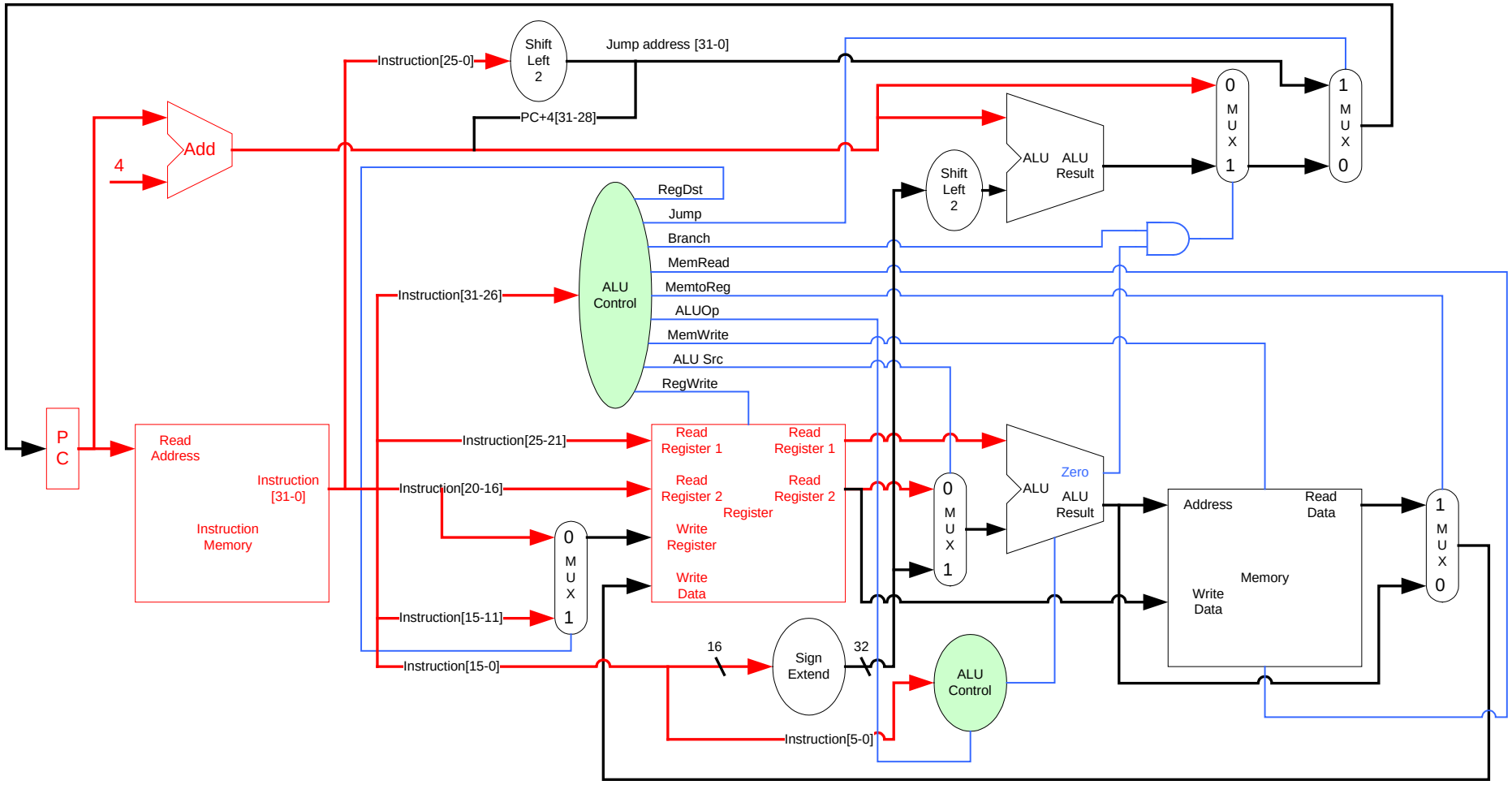
- Example flow: R-type instruction
  - Instruction fetch and PC increment
  - Reading of the registers
  - Processing of the data in the ALU
  - Writing the result to the register file

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

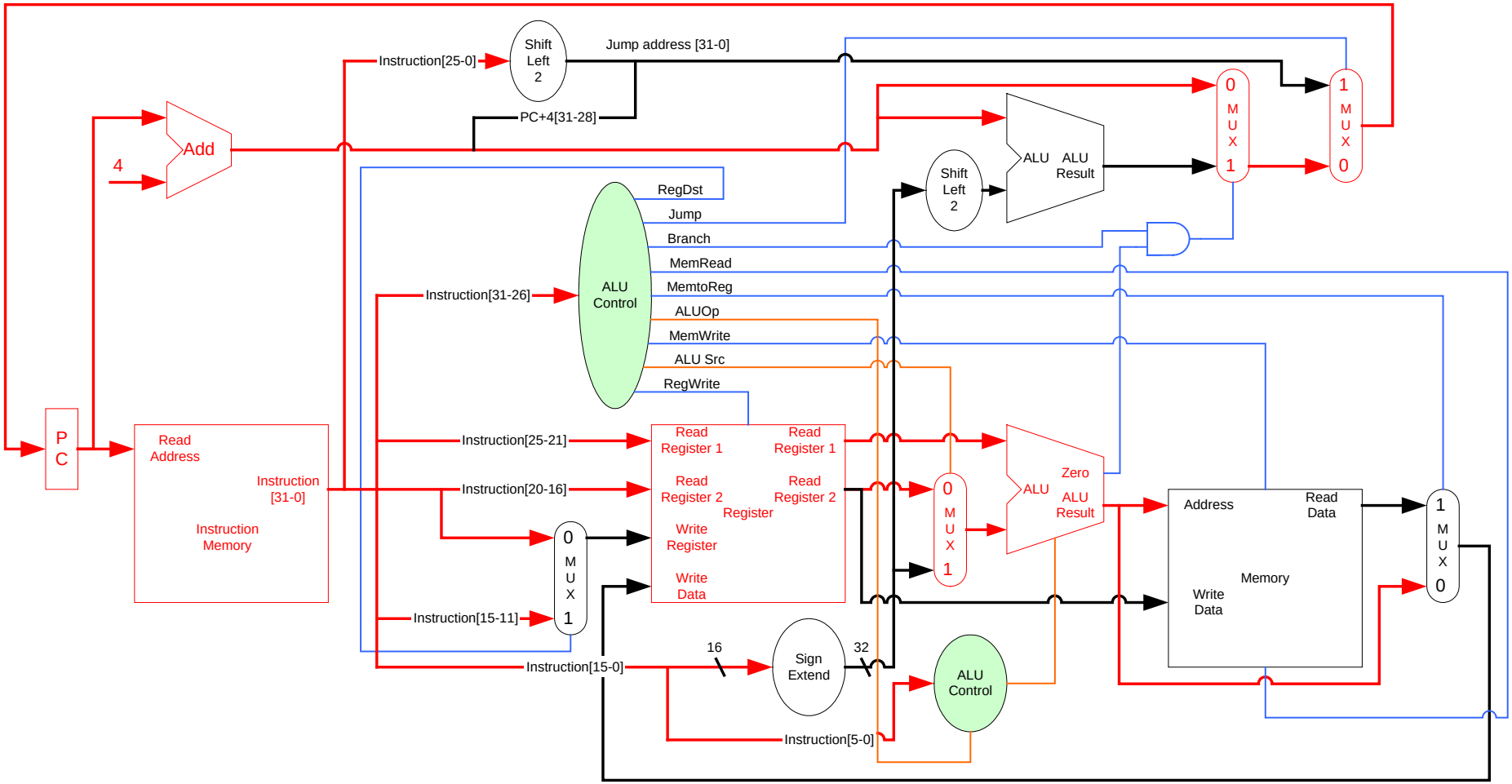
Example flow: R-type instruction



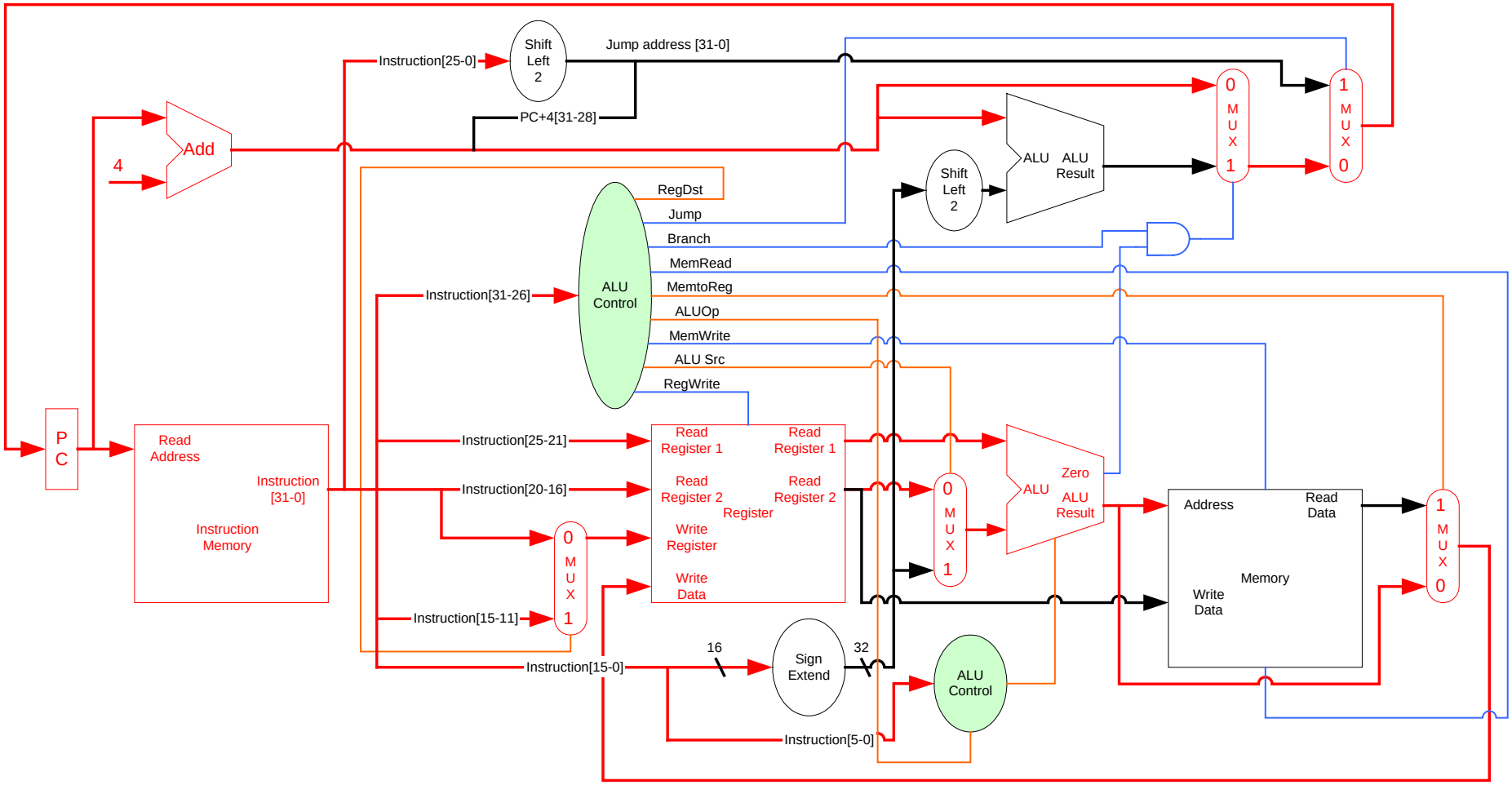
R-type flow phase 2



R-type flow phase 3



R-type flow phase 4



Example flow: load word

- lw $3, offset ($2)
  - Fetch instruction from memory, increment PC
  - Read register value from register $2
  - Compute the final address
  - Use address for for addressing the memory
  - Write the data into the register file
- beq $10, $11, offset
  - Fetch instruction form memory, increment PC
  - Read the two register values ($10, $11)
  - Subtract the values, calculate the branch target

Finalizing the control

Name	Opcode in decimal	Opcode in binary					
		Op5	Op4	Op3	Op2	Op1	Op0
R-format	0_{10}	0	0	0	0	0	0
lw	35_{10}	1	0	0	0	1	1
sw	43_{10}	1	0	1	0	1	1
beq	4_{10}	0	0	0	1	0	0

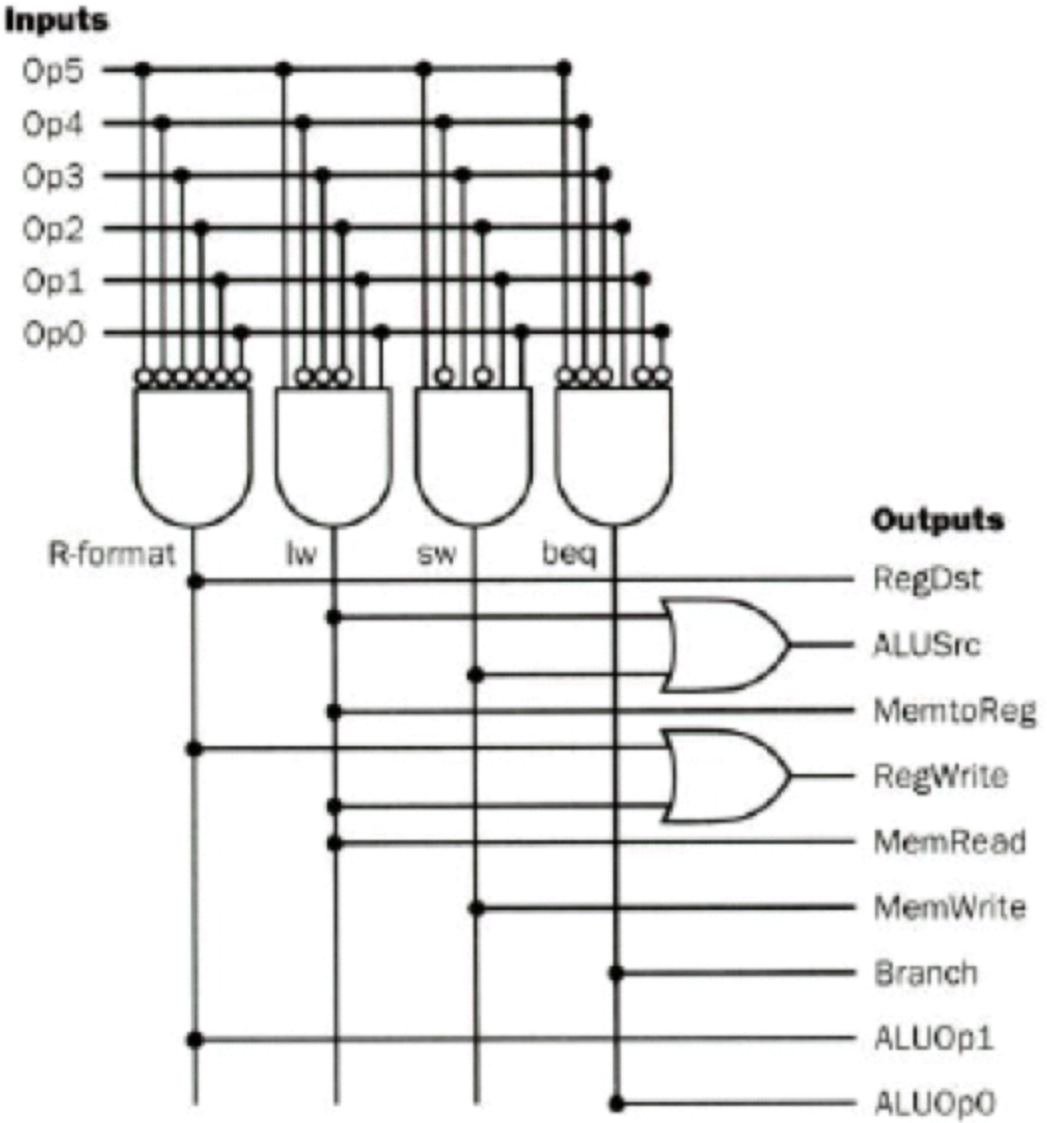
Control function

- Choose the right implementation

	Op[5..0]						RegDst	ALUSrc	MemtoReg	Regwrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
R-format	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0
lw	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0
sw	1	0	1	0	1	1	X	1	X	0	0	1	0	0	0
beq	0	0	0	1	0	0	X	0	X	0	0	0	1	0	1

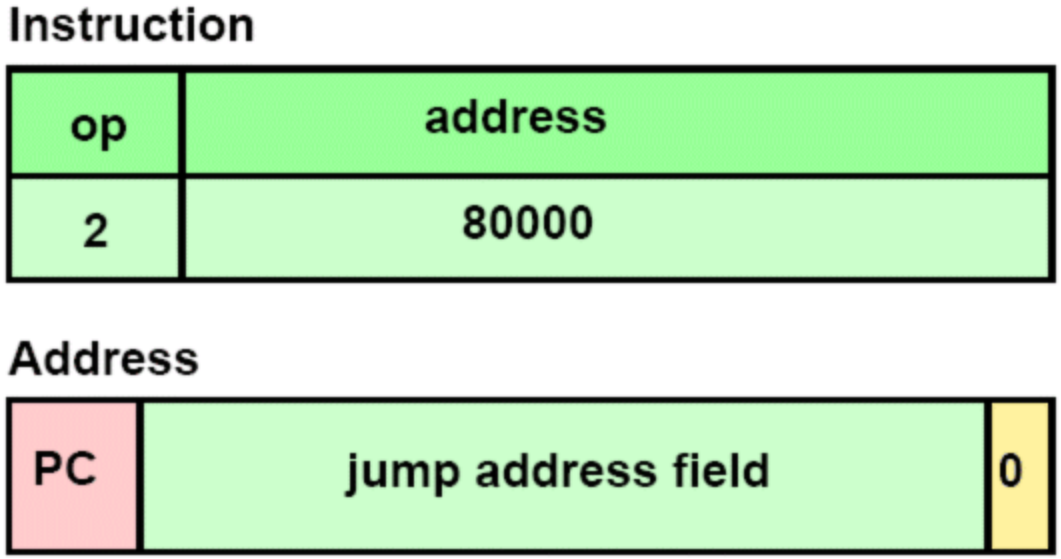
Implementation as PLA

- Custom logic
- PLA
- ROM
- ....

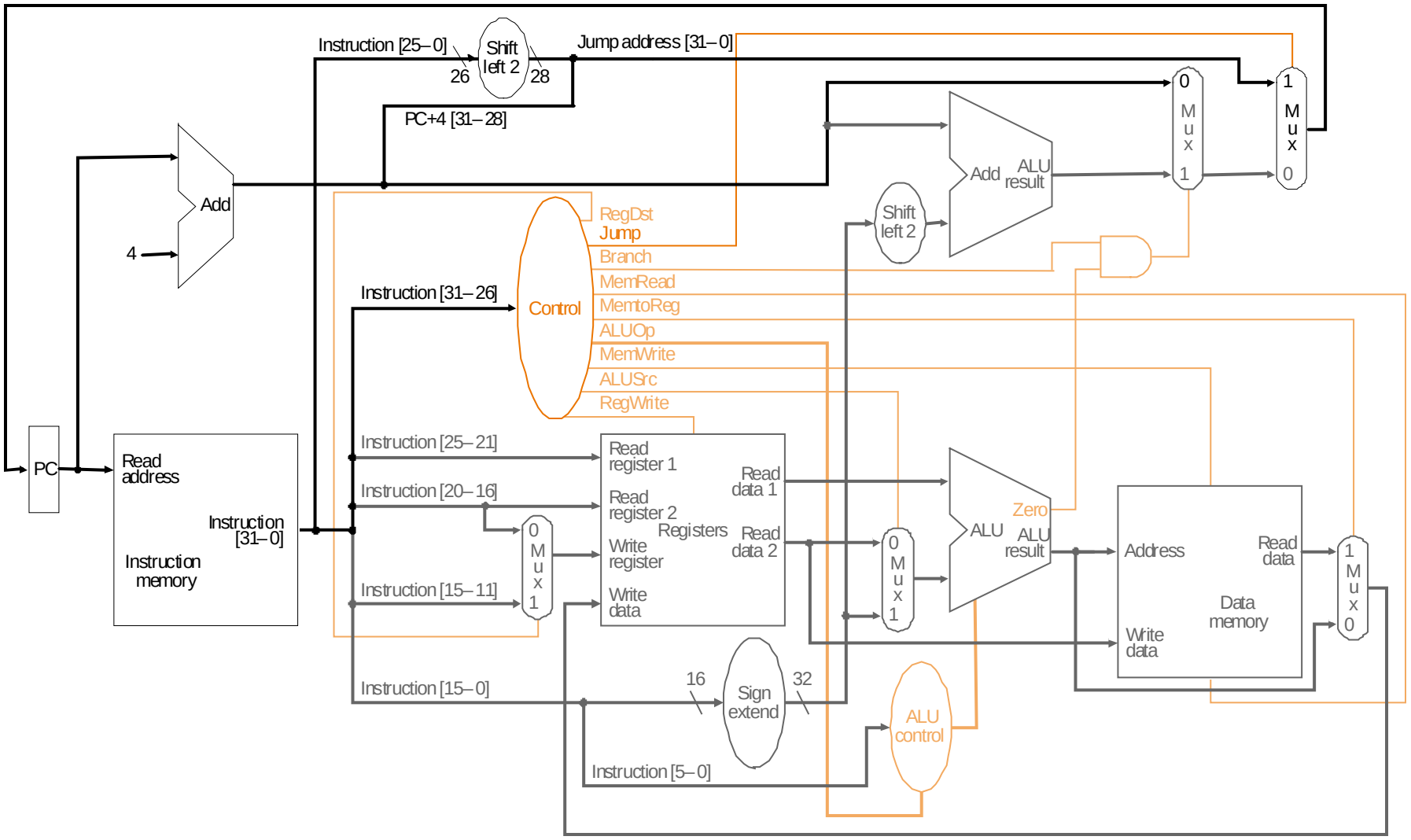


Jump instruction

- Extension of the previous architecture
- Jump address
- Lower 2 bits always 0
- Immediate field: bits 2-27
- Current PC: bits 28-31



Data path supporting jump



Limitation of single cycle approach

- Access times for various operations vary
  - Memory access: 10 ns
  - Register access: 2 ns
  - Add: 3 ns
  - Sub: 3.5 ns
  - Multiplication: 20 ns
  - FP multiplication: 24 ns
  - Transcendent functions: 40 ns
- Time depends on implementation
- The slowest module defines the cycle duration

Example

- List of actions per instruction
- Number of cycles / instruction
- Faster MIPS
- Take a benchmark and the probability of instructions

Instruction type	Instruction memory	Register read	ALU operation	Data memory	Register write	Total
R-Format op.	8	2	4	0	2	16
Load word	8	2	4	8	2	24
Store word	8	2	4	8	0	22
Branch	8	2	4	0	0	14
Jump	8	0	0	0	0	8

Example

- Probabilities
  - R-format 49% 16ns
  - Load word 22% 24ns
  - Store word 11% 22ns
  - Branch 16% 14ns
  - Jump 2% 8ns
- Average CPU clock cycle time for this benchmark: 18 ns
- Clock cycle time by slowest block: 24 ns (ignoring really long operations)