COMP2611: Computer Organization

The Processor: Datapath & Control

Major Goals

- To present the design of MIPS processor
    - **Single-cycle implementation for this class**
- To illustrate to datapath & control

What Do We Study in this Chapter?

- Focus on implementing of a subset of the core MIPS instruction set
  - **Memory-reference instructions**: `lw`, `sw`
  - **Arithmetic-logical instructions**: `add`, `sub`, `and`, `or`, `slt`
  - **Branch and jump instructions**: `beq`, `j`

- Instructions not included:
  - Integer instructions such as those for multiplication and division
  - Floating-point instructions

Big Picture

- First, understand how an instruction is executed before the design
- Next, split the execution of an instruction into multiple steps common to all instructions
- Next, implement each part separately
- Finally, put all these parts back together

How Is an Instruction Executed?

1. Fetch the instruction from memory location indicated by program counter (PC)

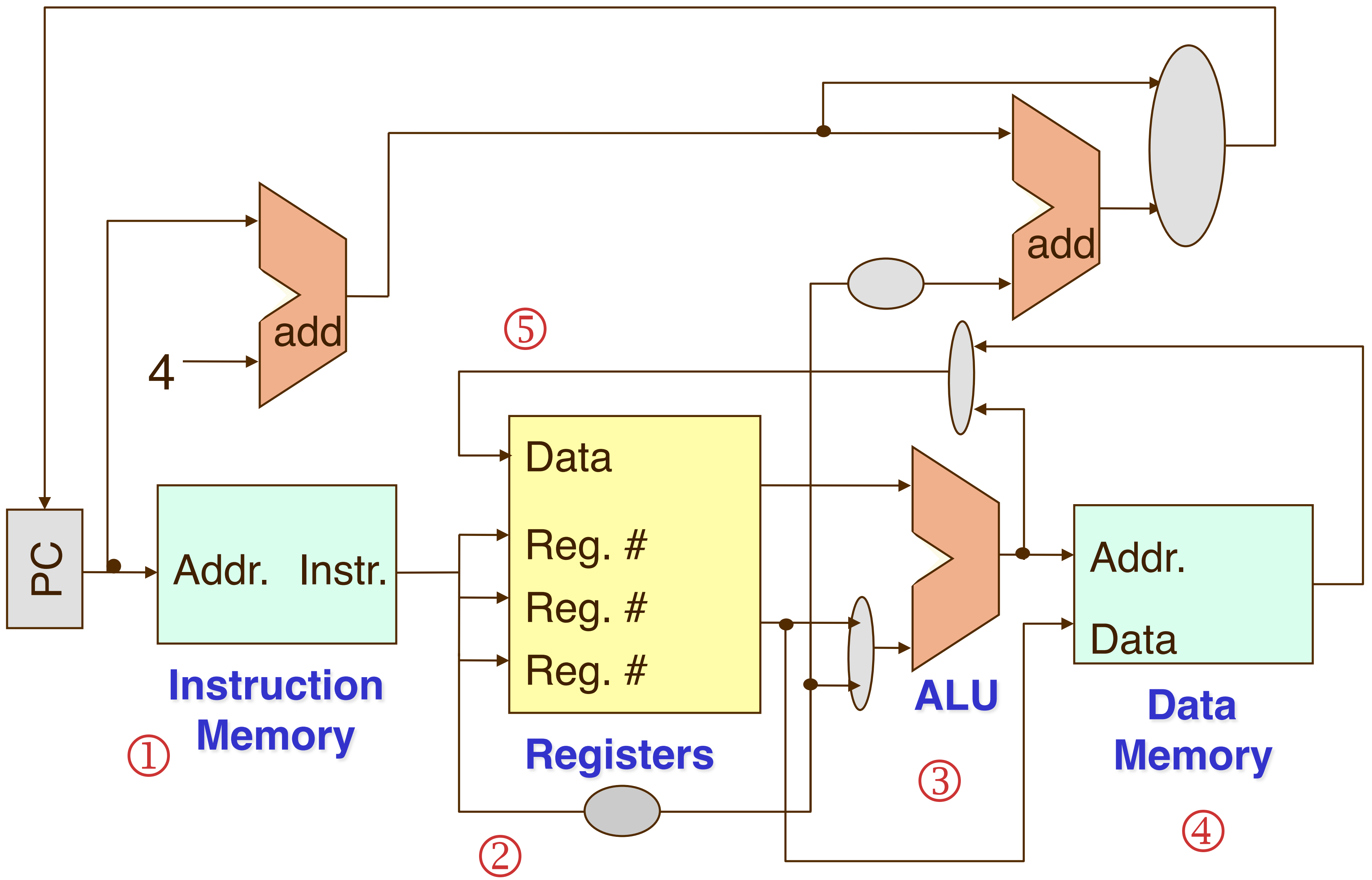
2. Decode the instruction – to find out what to perform
   Meanwhile, read source registers specified in the instruction fields
   - `lw` instruction require reading only one register
   - most other instructions require reading two registers

3. Perform the operation required by the instruction using the ALU
   - Arithmetic & logical instructions: execute
   - Memory-reference instructions: use ALU for address calculation
   - Conditional branch instructions: use ALU for comparison

4. Memory access: `lw` and `sw` instructions

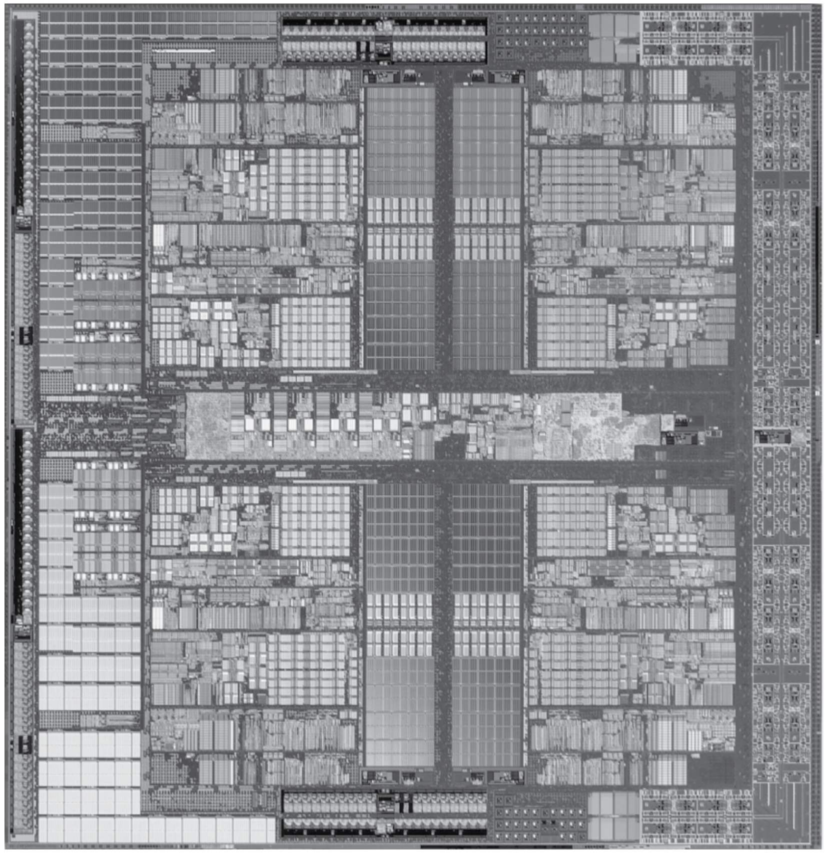
5. Write back the result to the destination register
   Increment PC by 4 or change PC to branch target address

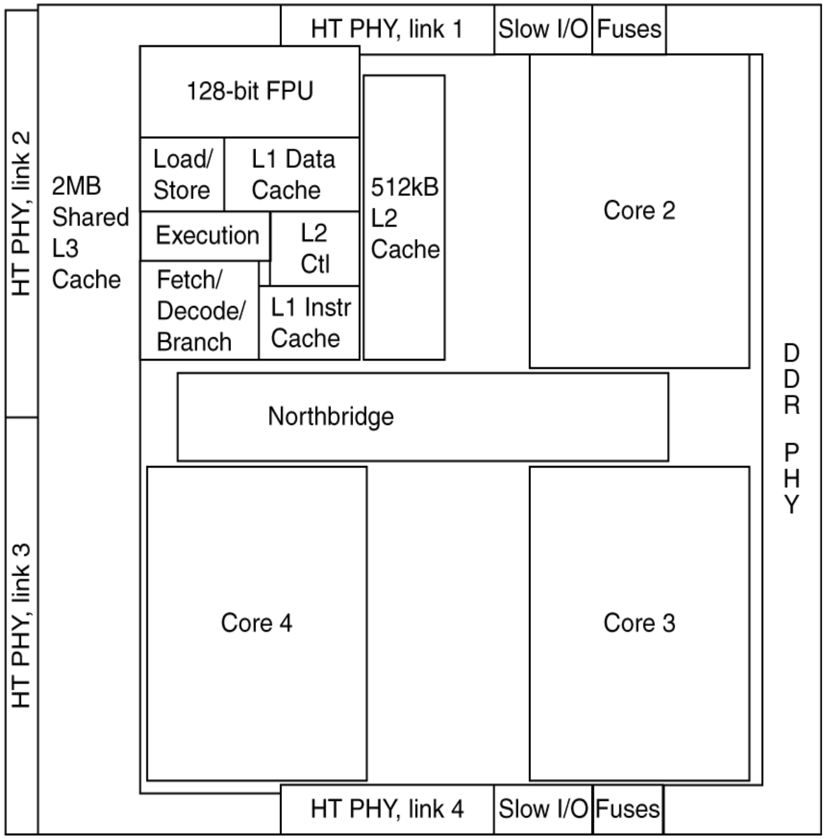
Brief Overview of MIPS Processor Implementation



Anatomy of a Computer: Inside the Processor

- AMD Barcelona: 4 processor cores

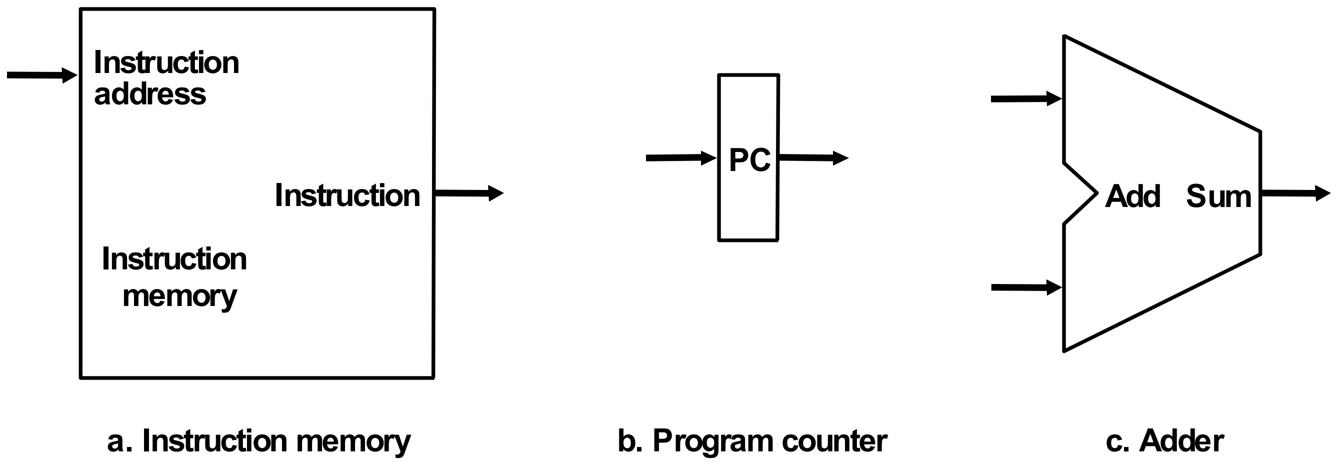




1. Building a Datapath

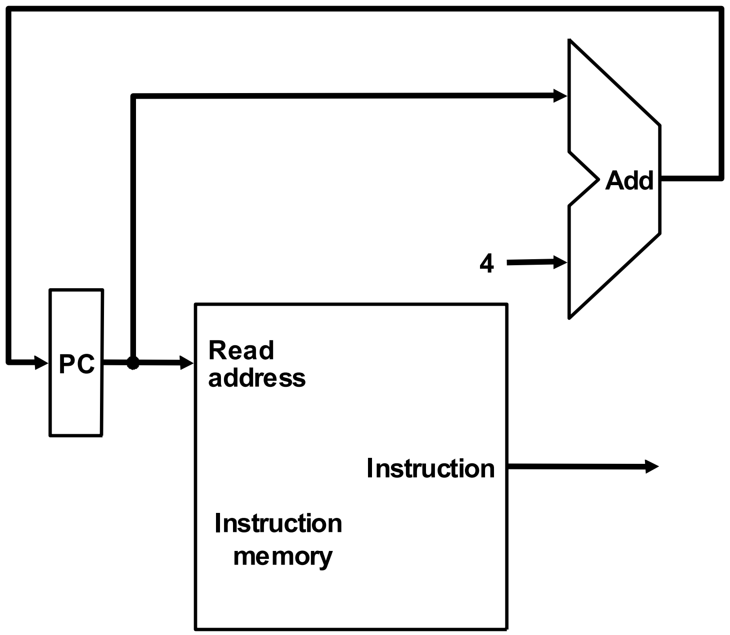
Basic Building Blocks

- **Instruction memory**:
  - A memory unit that stores the instructions of a program
  - Supplies an instruction given its address
- **Program counter**:
  - A register storing the address of the instruction being executed
- **Adder**:
  - A unit that increments PC to form the address of next instruction



Executing Instruction

1. Fetch the current instruction from memory using PC
2. Prepare for the next instruction
   - ➢ By incrementing PC by 4 to point to next instruction (base case)
   - ➢ Will worry about the branches later

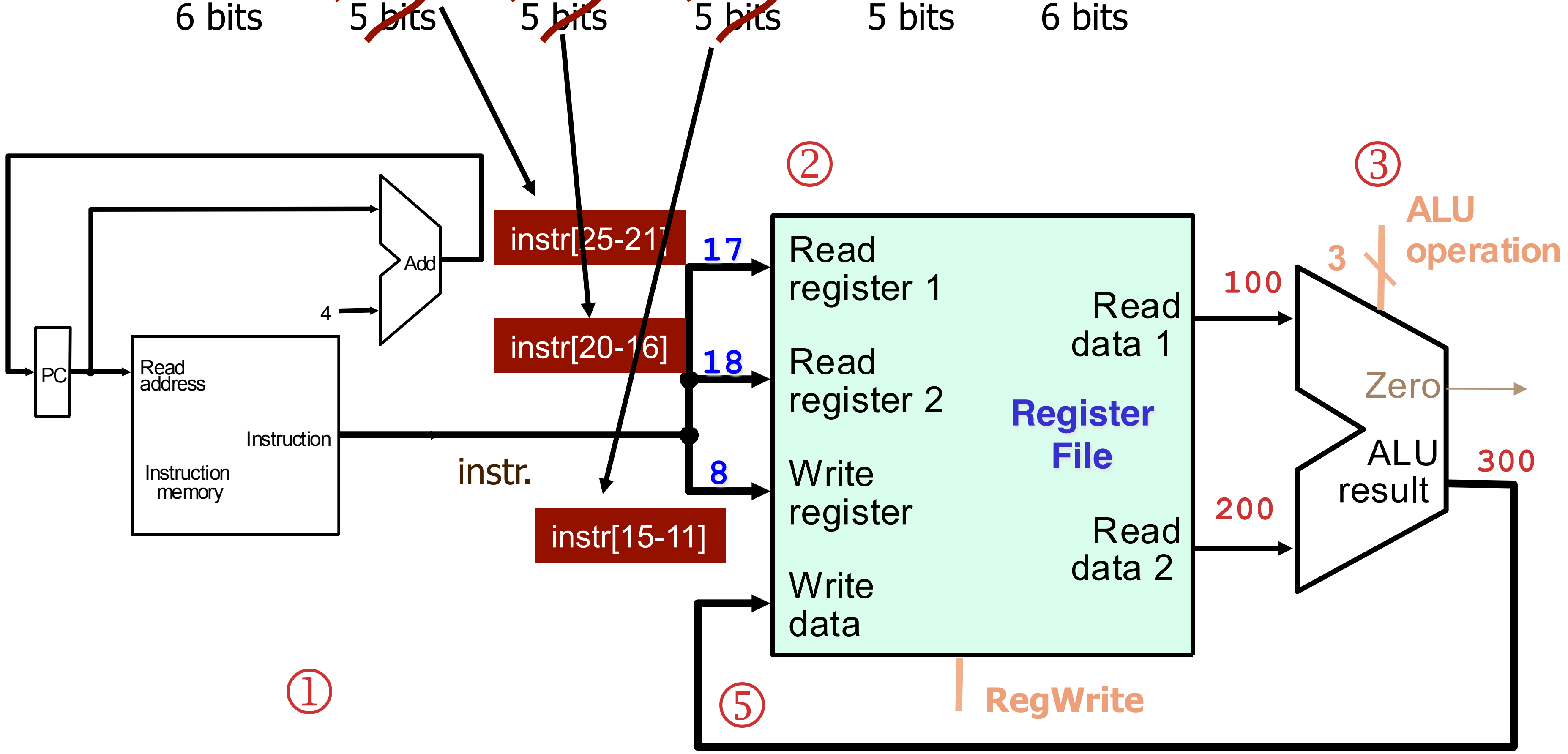


Datapath for Arithmetic/Logical (R-Type) Instr.

- **E.g.:** `add $t0, $s1, $s2`

0	17	18	8	0	32
000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Register	Value
s1	100
s2	200



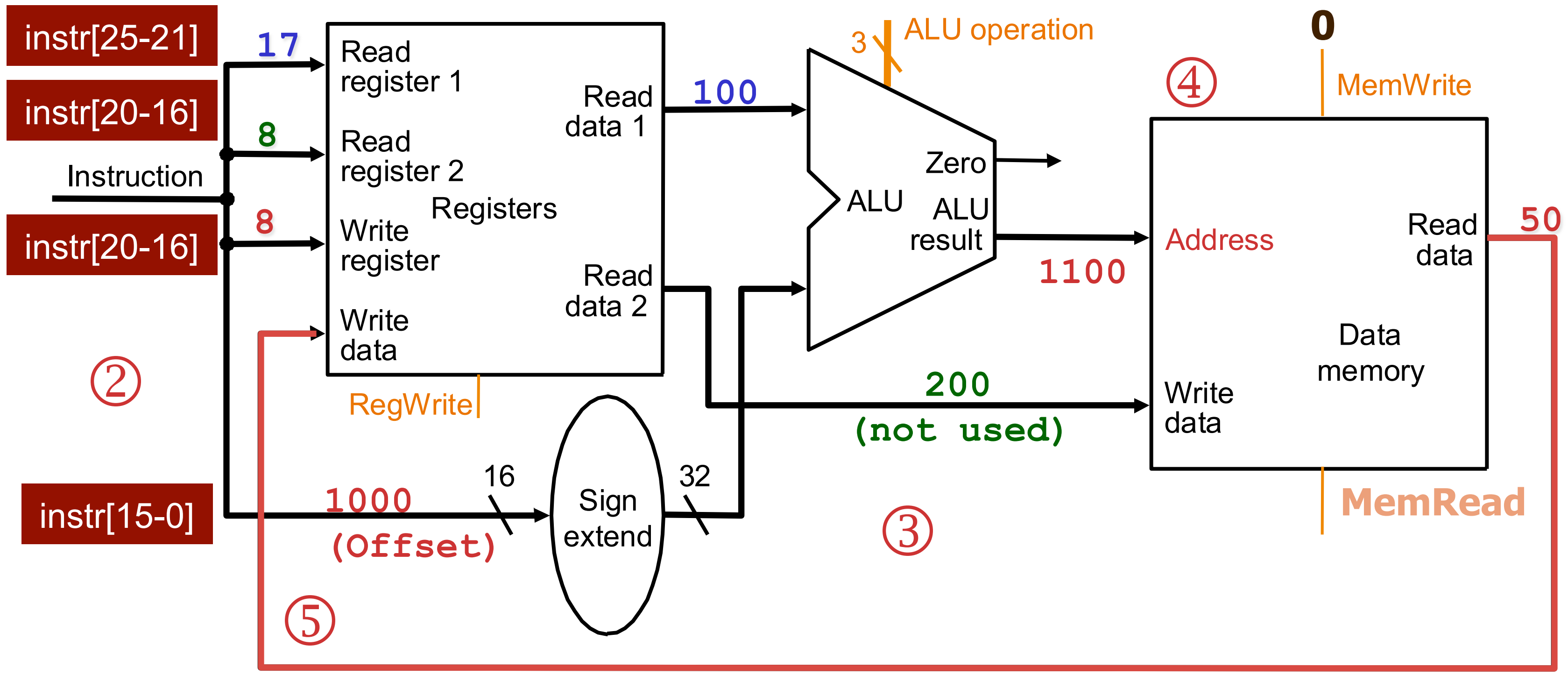
Datapath for Load/Store (I-Format) Instr.

- **E.g.:** `lw $t0, 1000($s1)`

35	17 (s1)	8 (t0)	1000 (immediate)
100011	10001	01000	0000 0011 1110 1000
6 bits	5 bits	5 bits	16 bits

Register	Value
t0	200
s1	100

Memory[1100]
50



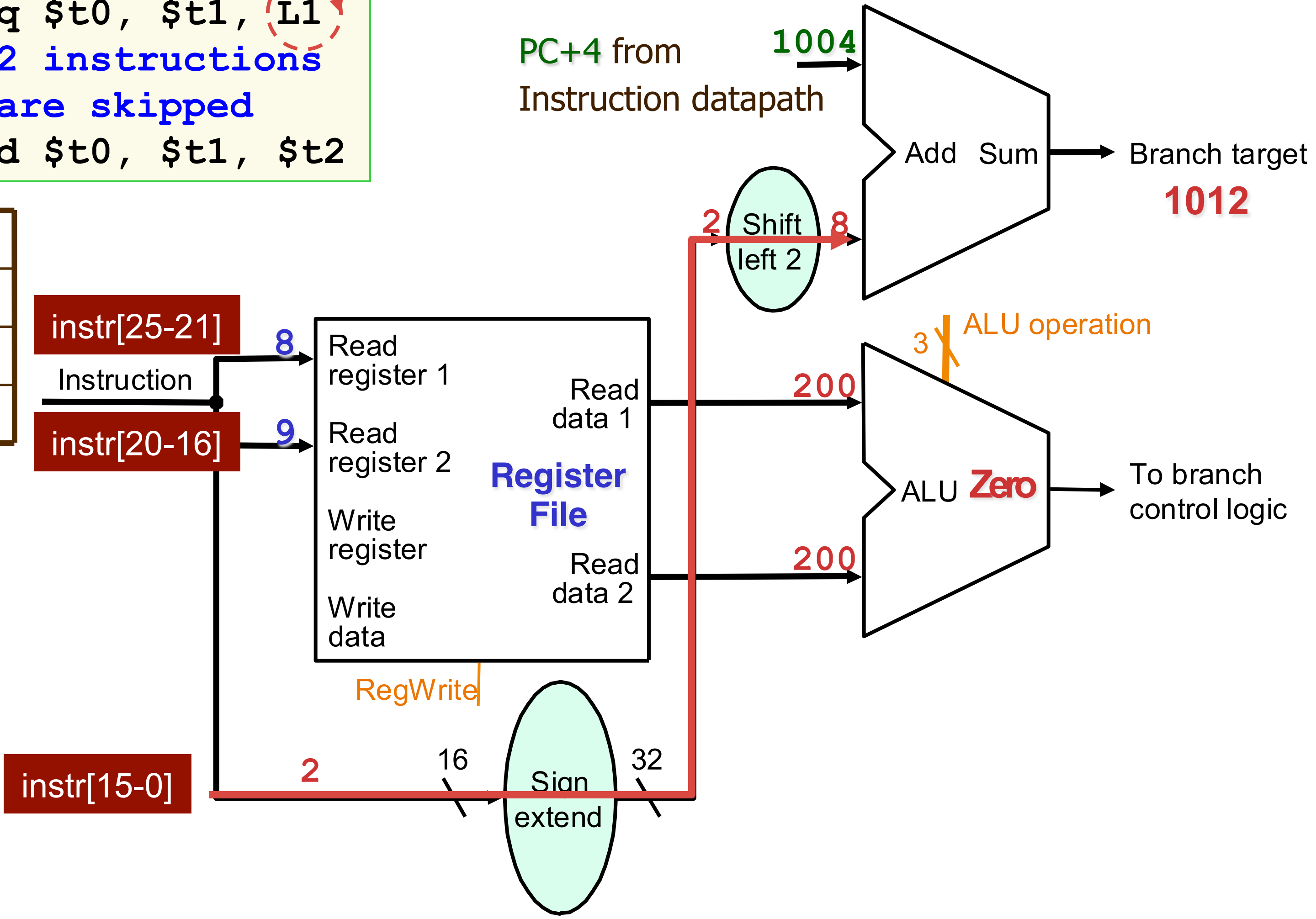
Note: For load word instruction, **MemWrite** has to be de-asserted so that the memory will not be modified by incoming write data.

Datapath for Branch (I-Format) Instr.

- **E.g.:** `beq $t0, $t1, 2`

```
1000:     beq $t0, $t1, L1
1004:     . 2 instructions
1008:     . are skipped
1012: L1: add $t0, $t1, $t2
```

Register	Value
t0	200
t1	200
PC	1000



A Simple Single-Cycle Implementation

- We have already built a datapath for each instruction separately
- Now, we need to combine them into a **single datapath**
- **Key to combine**

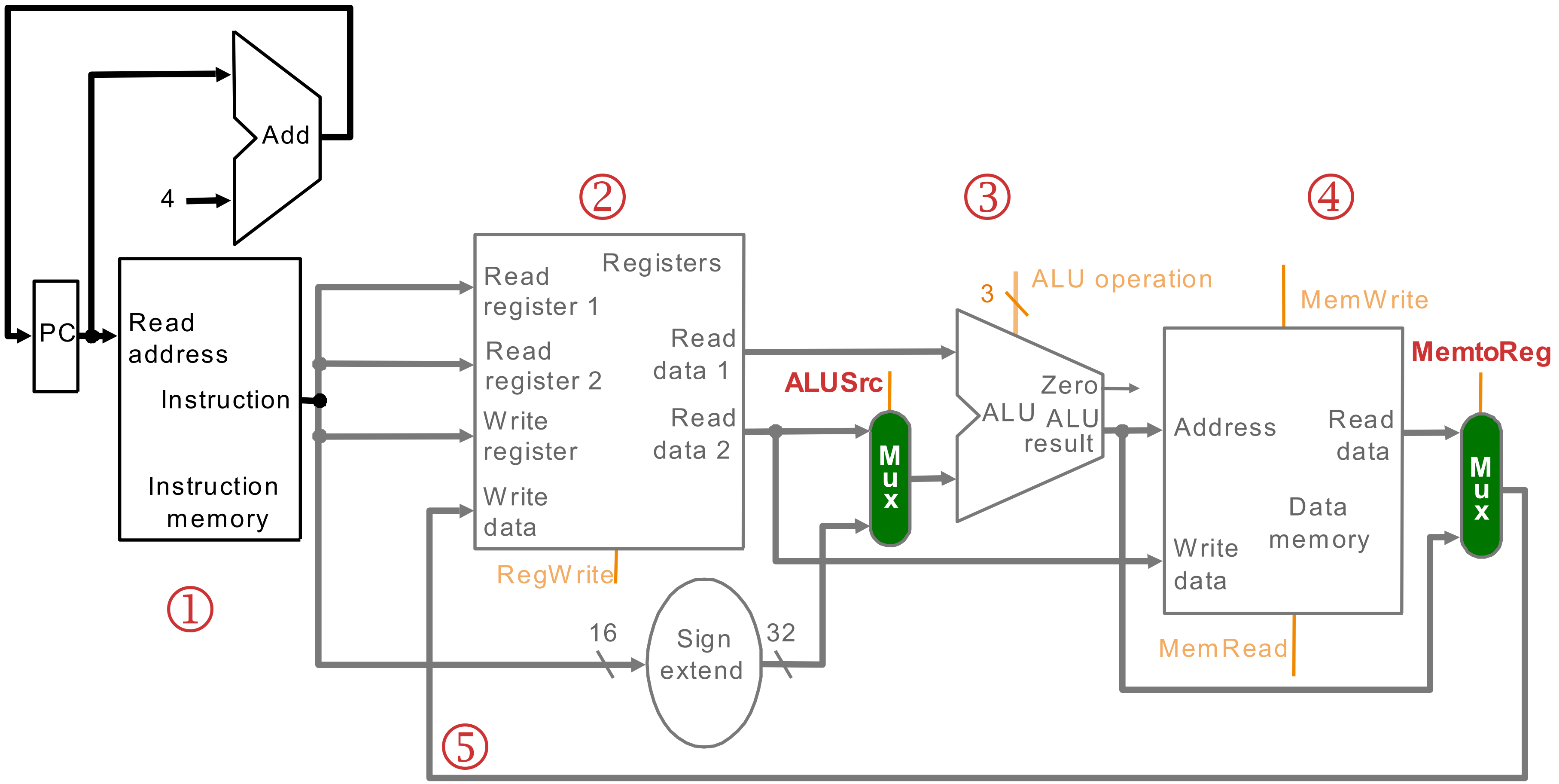
Share some of the resources (e.g., ALU) among different instructions

Note:

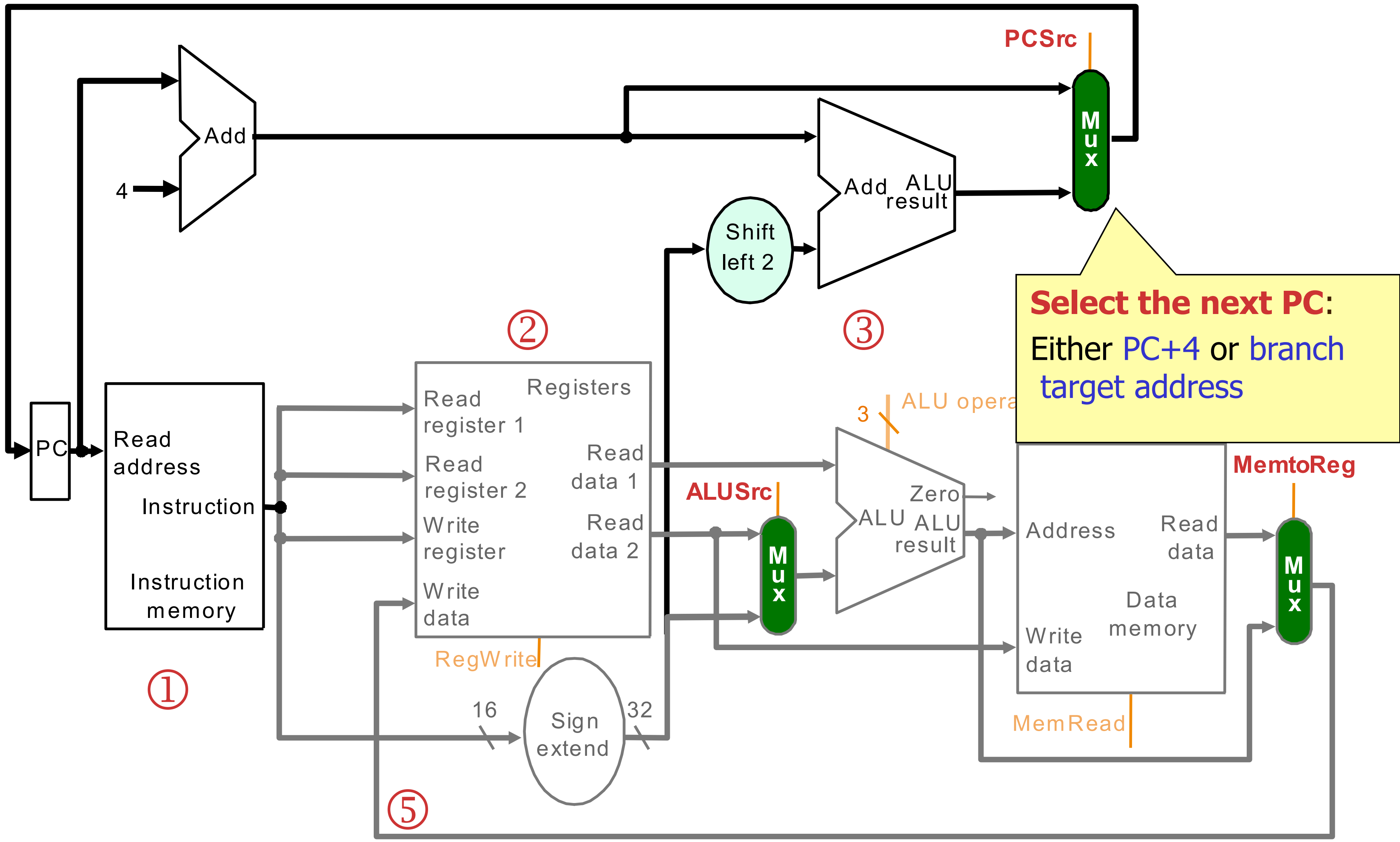
- This simple implementation is based on the (unrealistic) assumption
  - i.e. all instructions take just one clock cycle each to complete
- Implication:
  - No datapath resource can be used more than once per instruction
  - Any element needed more than once must be duplicated
  - ➪ Instructions and data have to be stored in separate memories

Combined Datapath for R-Type & Memory Instr.

- Use **ALUSrc** to decide which source will be sent to the ALU
- Use **MemtoReg** to choose the source of output back to dest. register

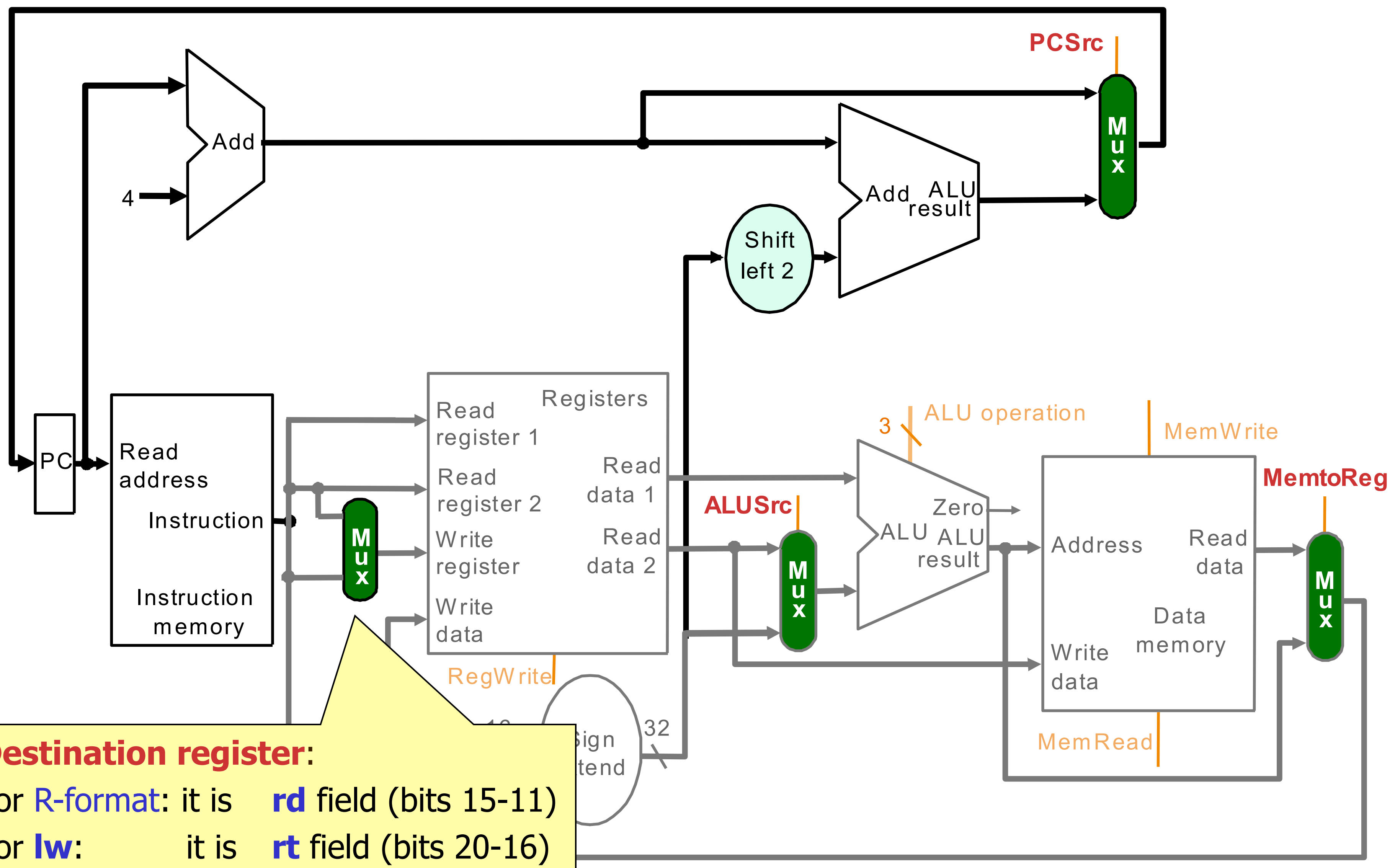


Combined Datapath for Different Instr. Classes



Select the next PC:
Either PC+4 or branch target address

Muxing Two Possible Destination Registers



Destination register:

For **R-format**: it is **rd** field (bits 15-11)

For **lw**: it is **rt** field (bits 20-16)