3810: Computer Organization

Lecture 3: Basic MIPS Architecture

Anton Burtsev
October, 2022

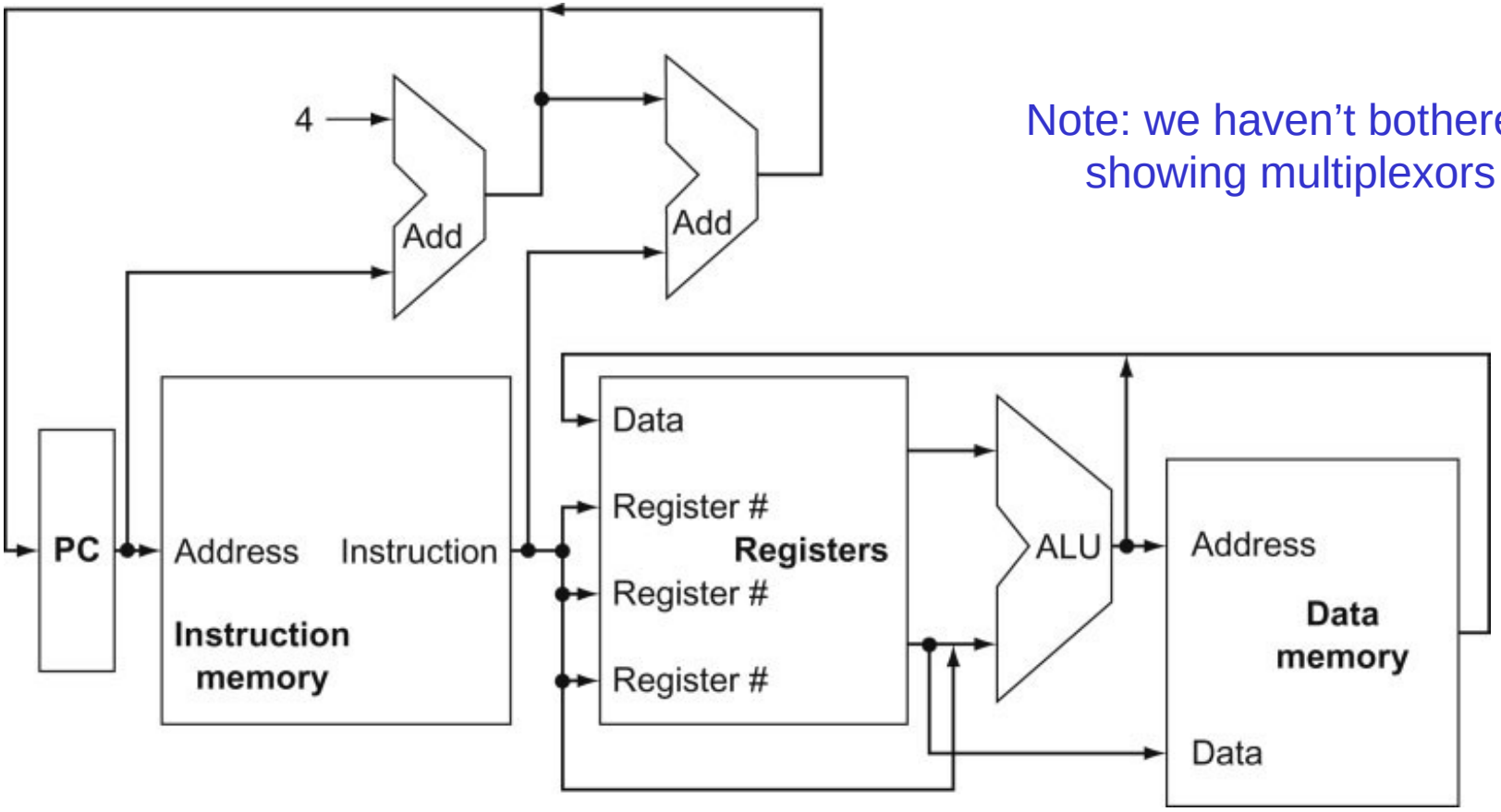
Basic MIPS Architecture

- Now that we understand clocks and storage of states,
- we'll design a simple CPU that executes:
  - basic math (add, sub, and, or, slt)
  - memory access (lw and sw)
  - branch and jump instructions (beq and j)

Implementation Overview

- We need memory
  - to store instructions
  - to store data
  - for now, let’s make them separate units
- We need registers, ALU, and a whole lot of control logic
- CPU operations common to all instructions:
  - use the program counter (PC) to pull instruction out of instruction memory
  - read register values

View from 30,000 Feet



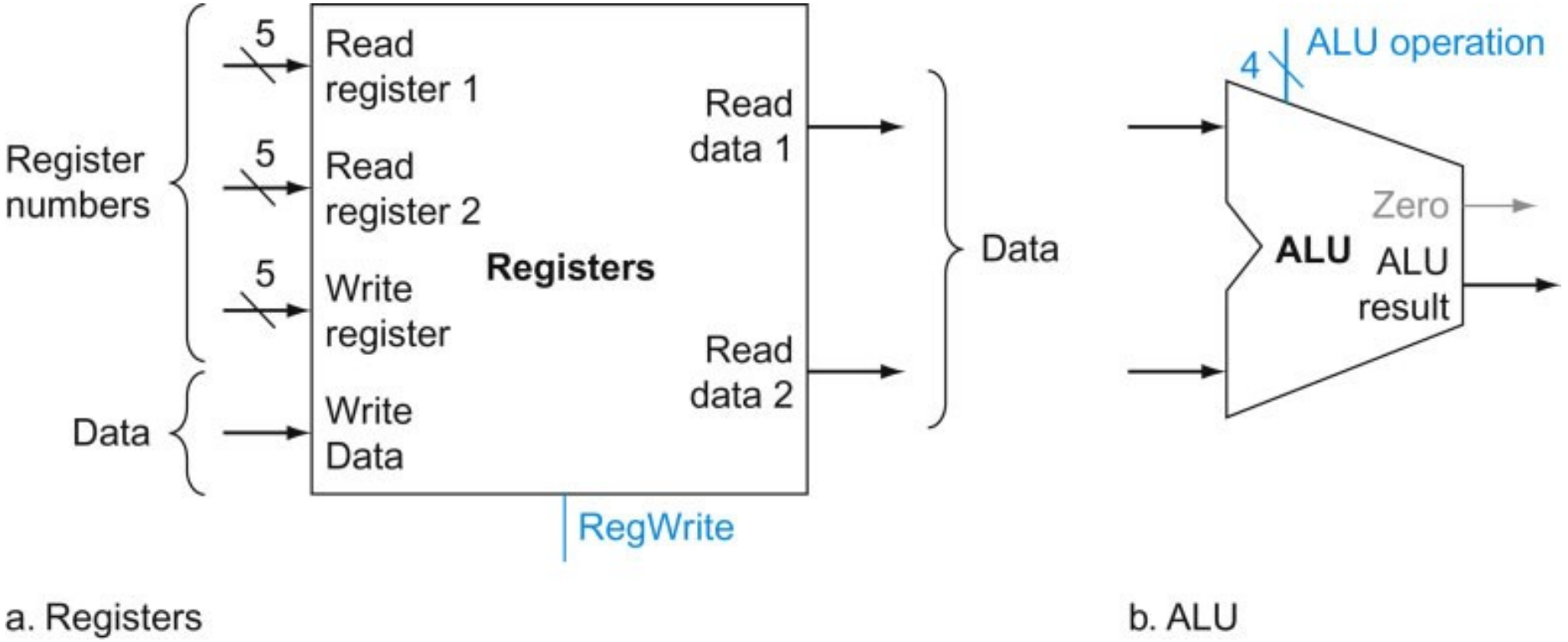
Note: we haven't bothered showing multiplexors

Source: H&P textbook

- What is the role of the Add units?
- Explain the inputs to the data memory unit
- Explain the inputs to the ALU
- Explain the inputs to the register unit

Implementing R-type Instructions

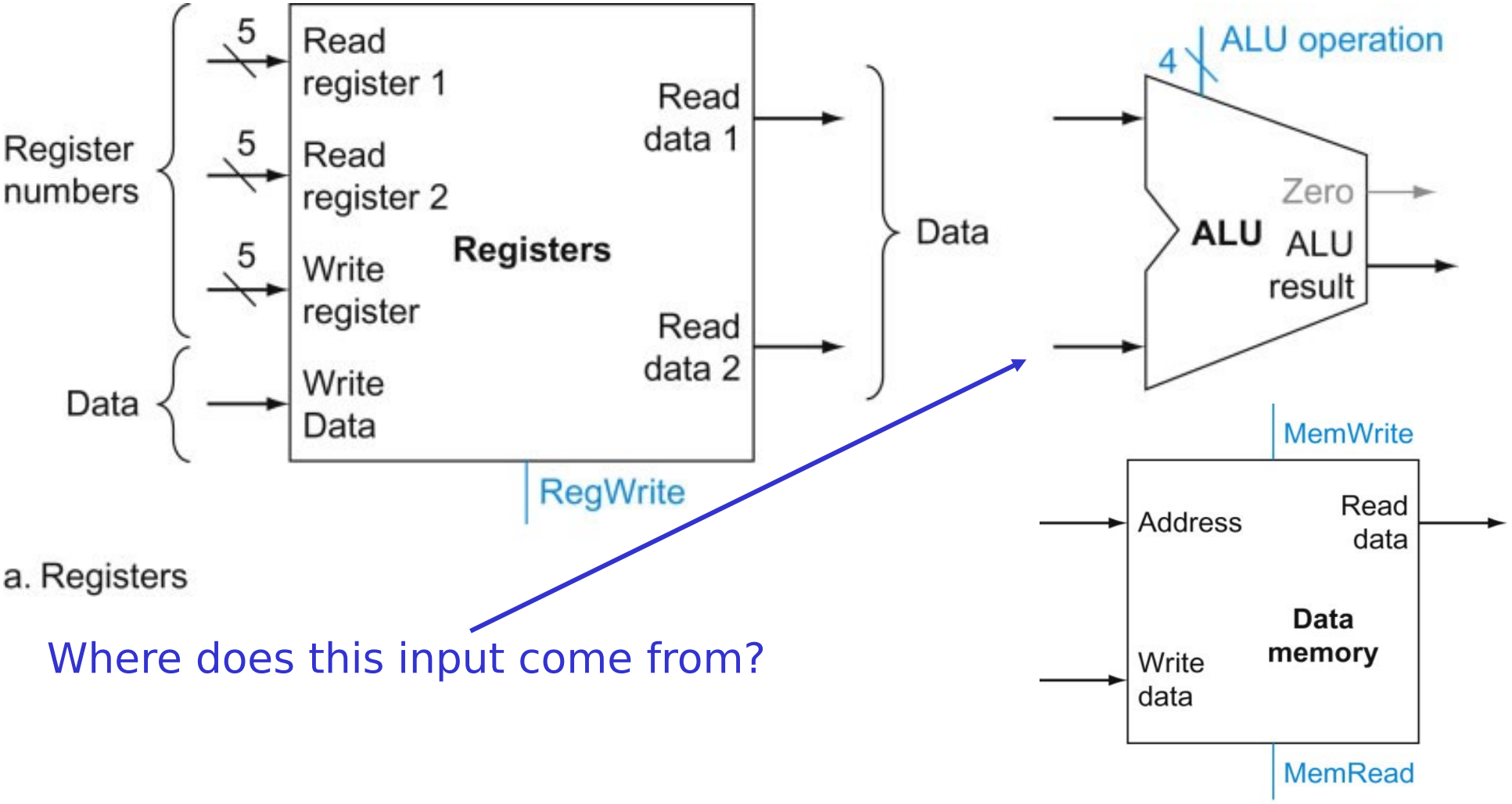
- Instructions of the form add $r1, $r2, $r3
- Explain the role of each signal



Source: H&P textbook

Implementing Loads/Stores

- Instructions of the form lw $r1, 8($r2) and sw $r1, 8($r2)
- Explain the role of each signal

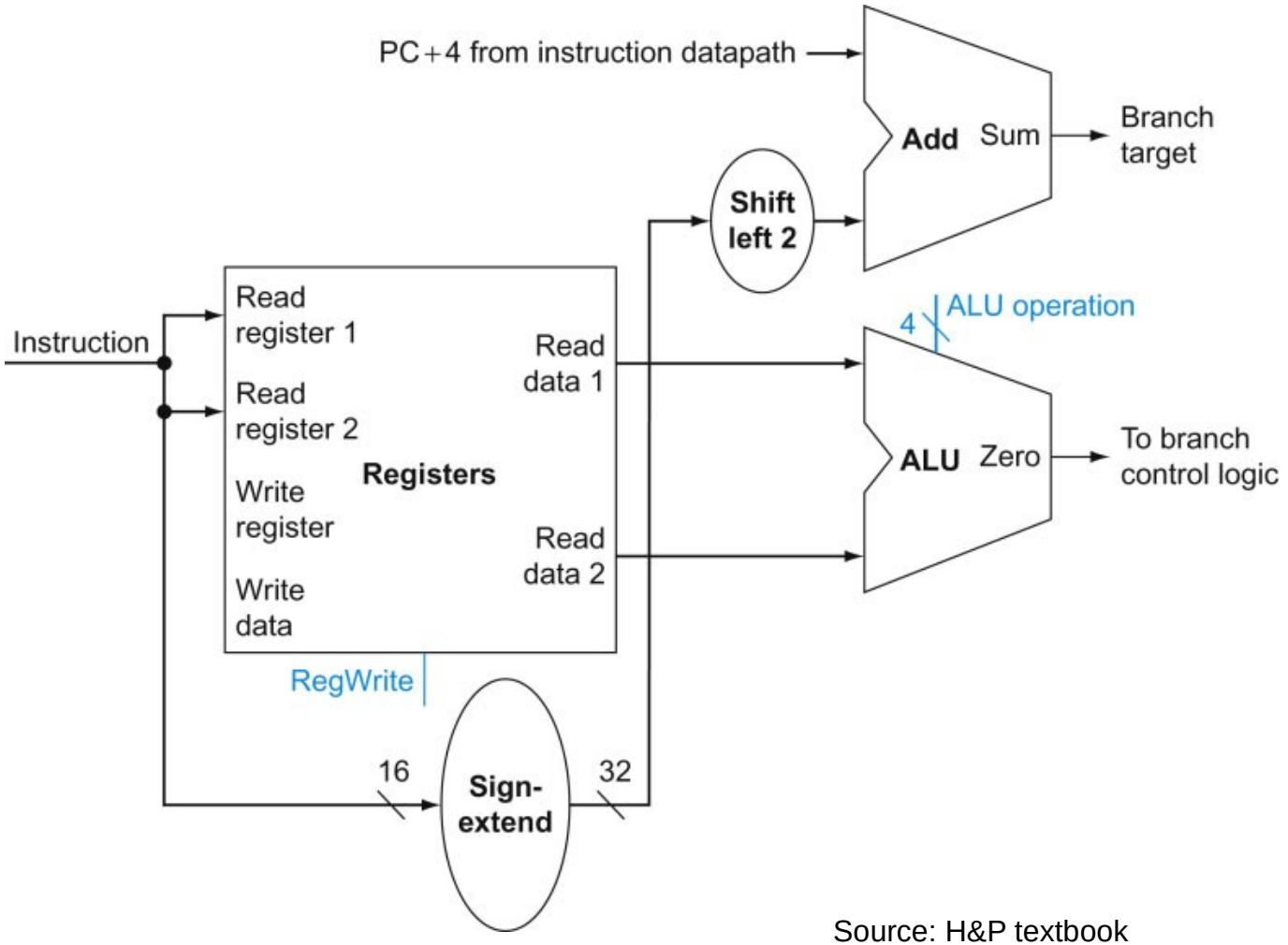


Where does this input come from?

Source: H&P textbook

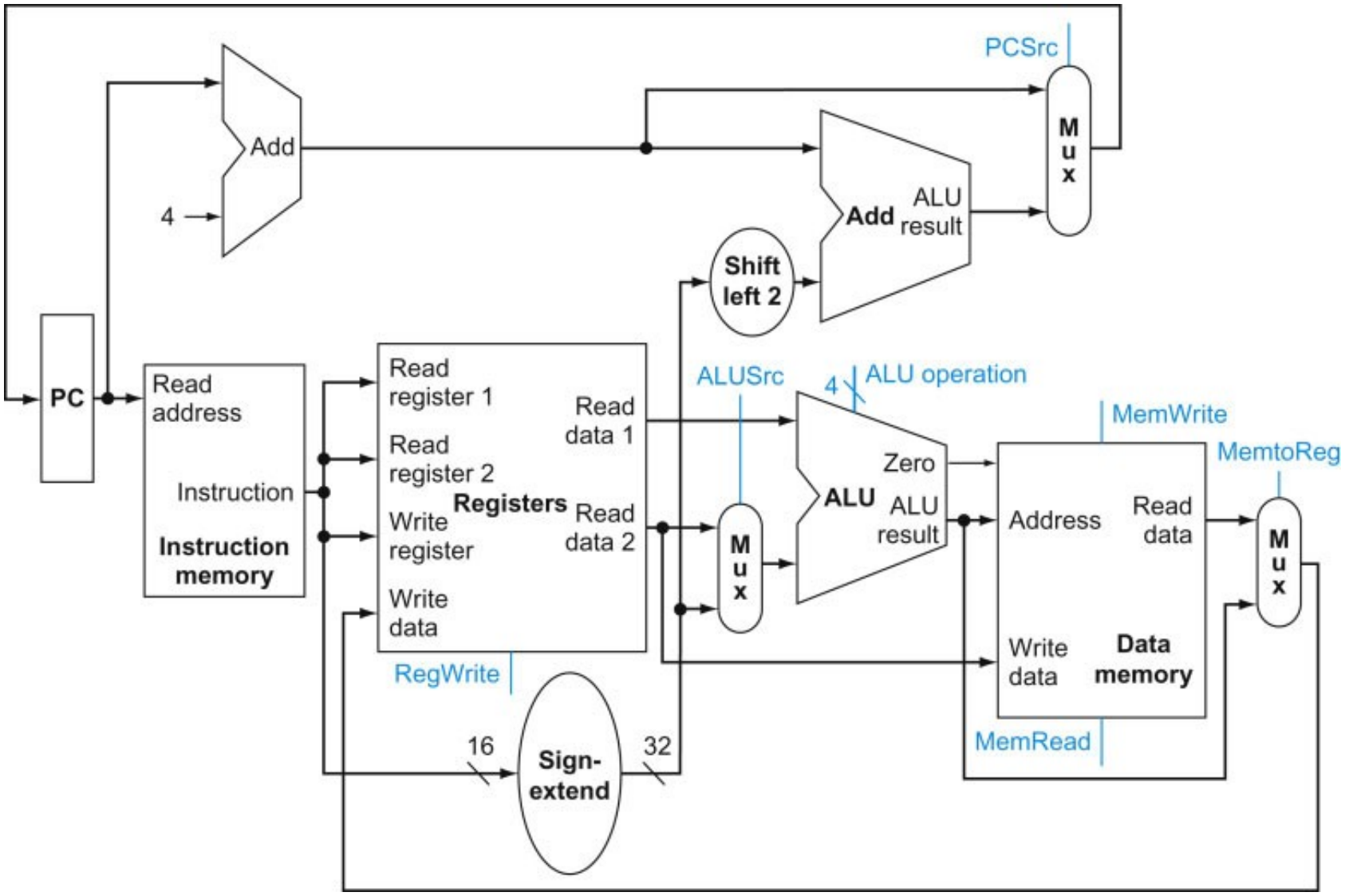
Implementing J-type Instructions

- Instructions of the form beq $r1, $r2, offset



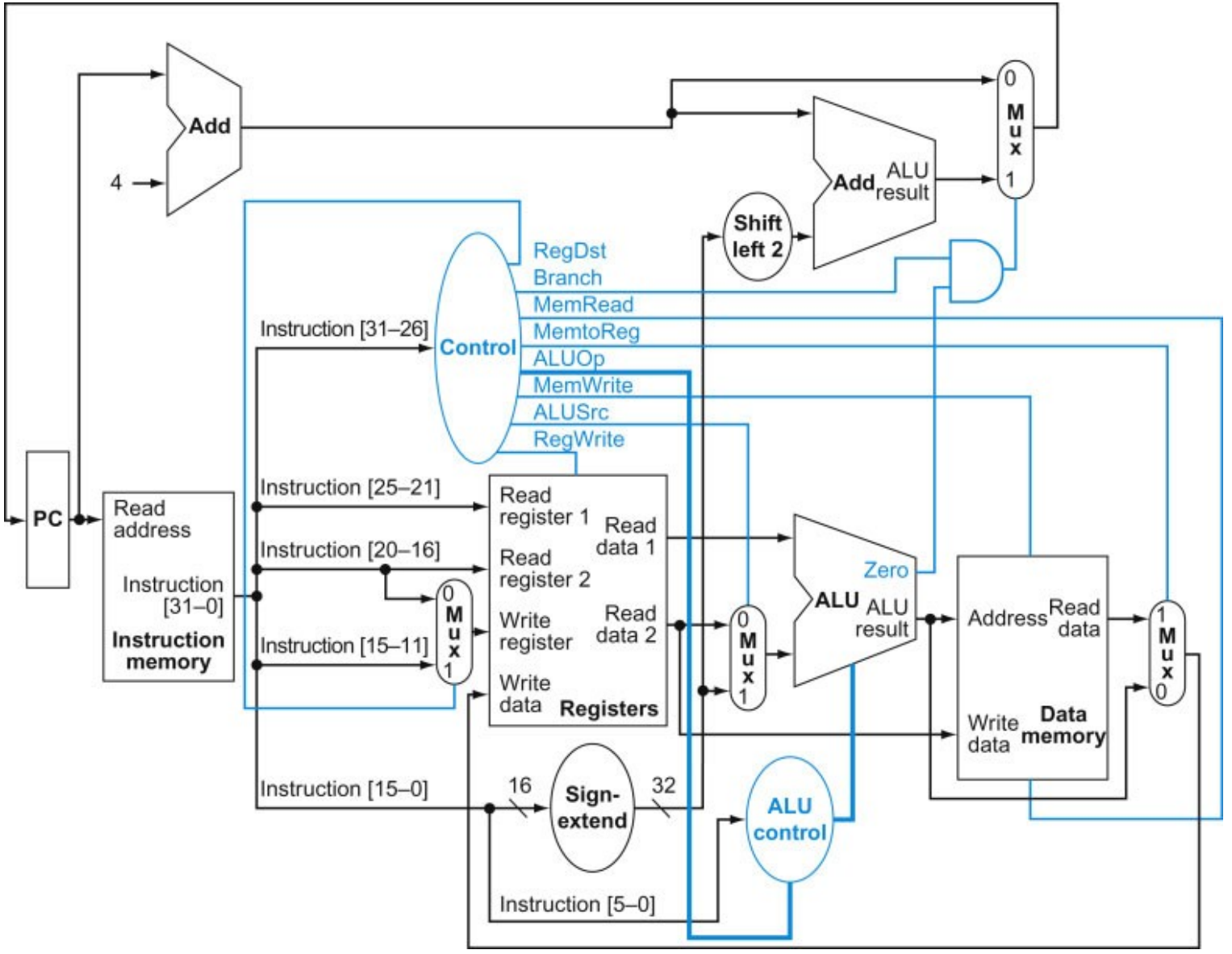
Source: H&P textbook

View from 10,000 Feet



Source: H&P textbook

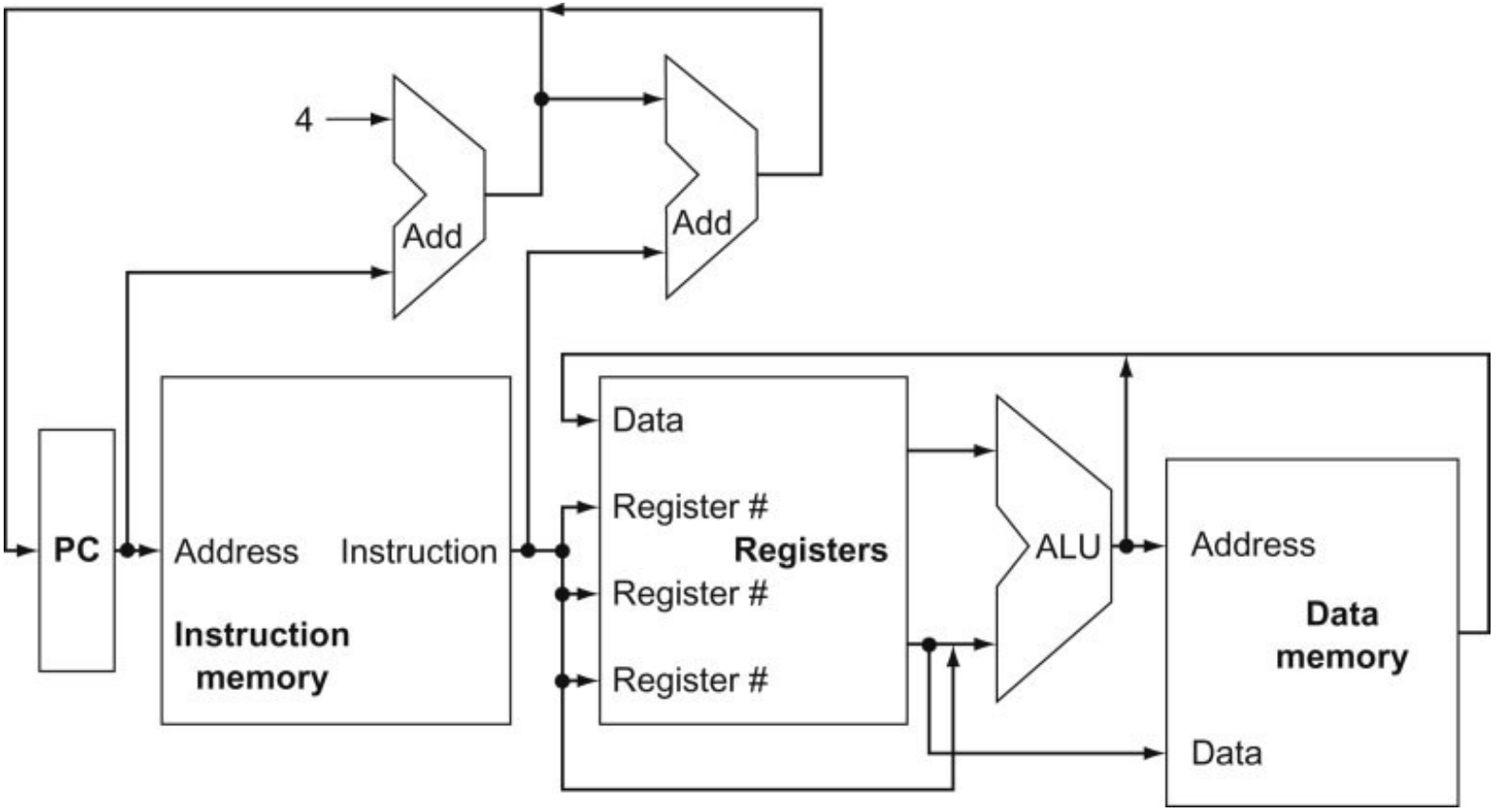
View from 5,000 Feet



Source: H&P textbook

Thank you!

Clocking Methodology



Source: H&P textbook

- Which of the above units need a clock?
- What is being saved (latched) on the rising edge of the clock?
- Keep in mind that the latched value remains there for an entire cycle