

# Lecture 16: Basic CPU Design

---

- Today's topics:
  - Single-cycle CPU
  - Multi-cycle CPU

# Basic MIPS Architecture

---

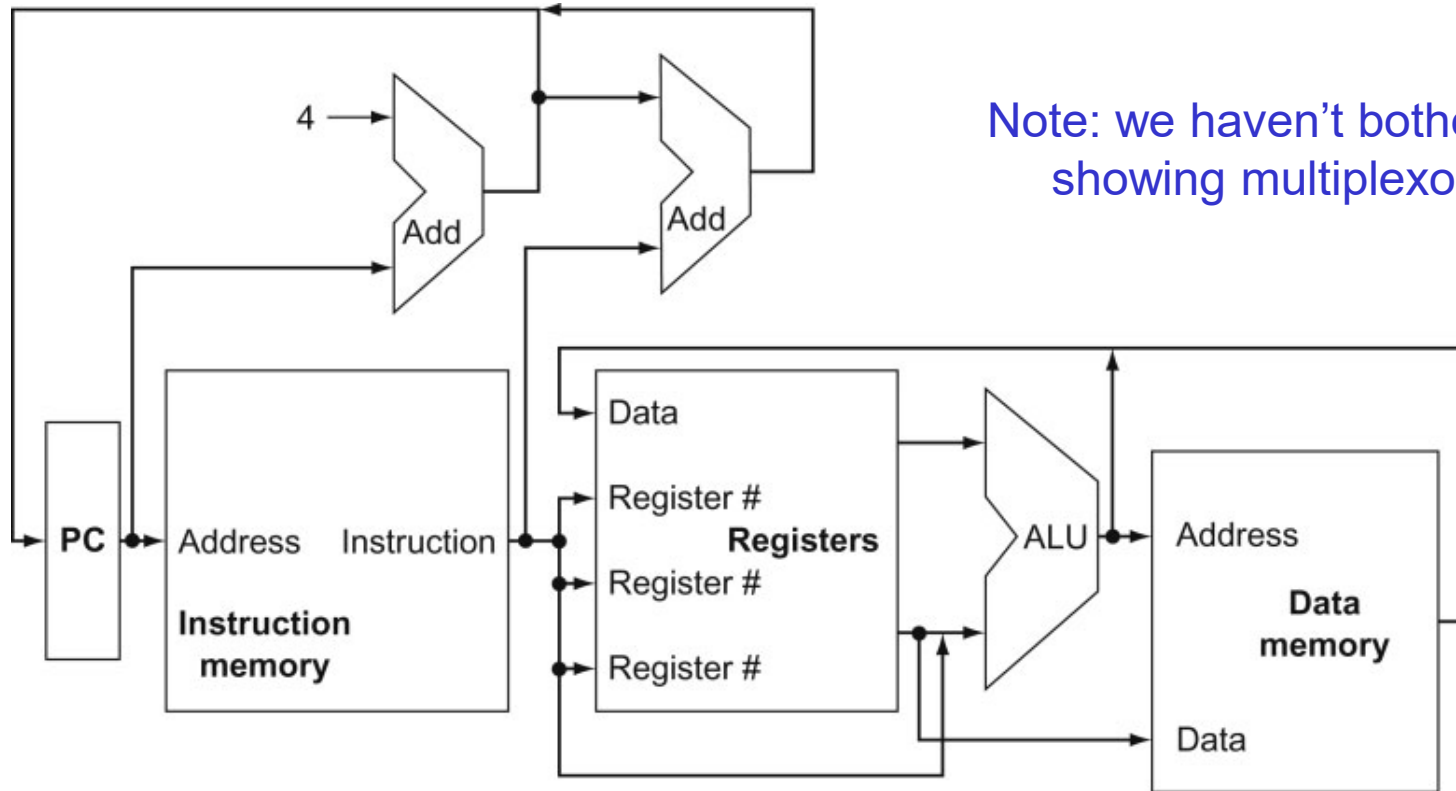
- Now that we understand clocks and storage of states, we'll design a simple CPU that executes:
  - basic math (add, sub, and, or, slt)
  - memory access (lw and sw)
  - branch and jump instructions (beq and j)

# 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

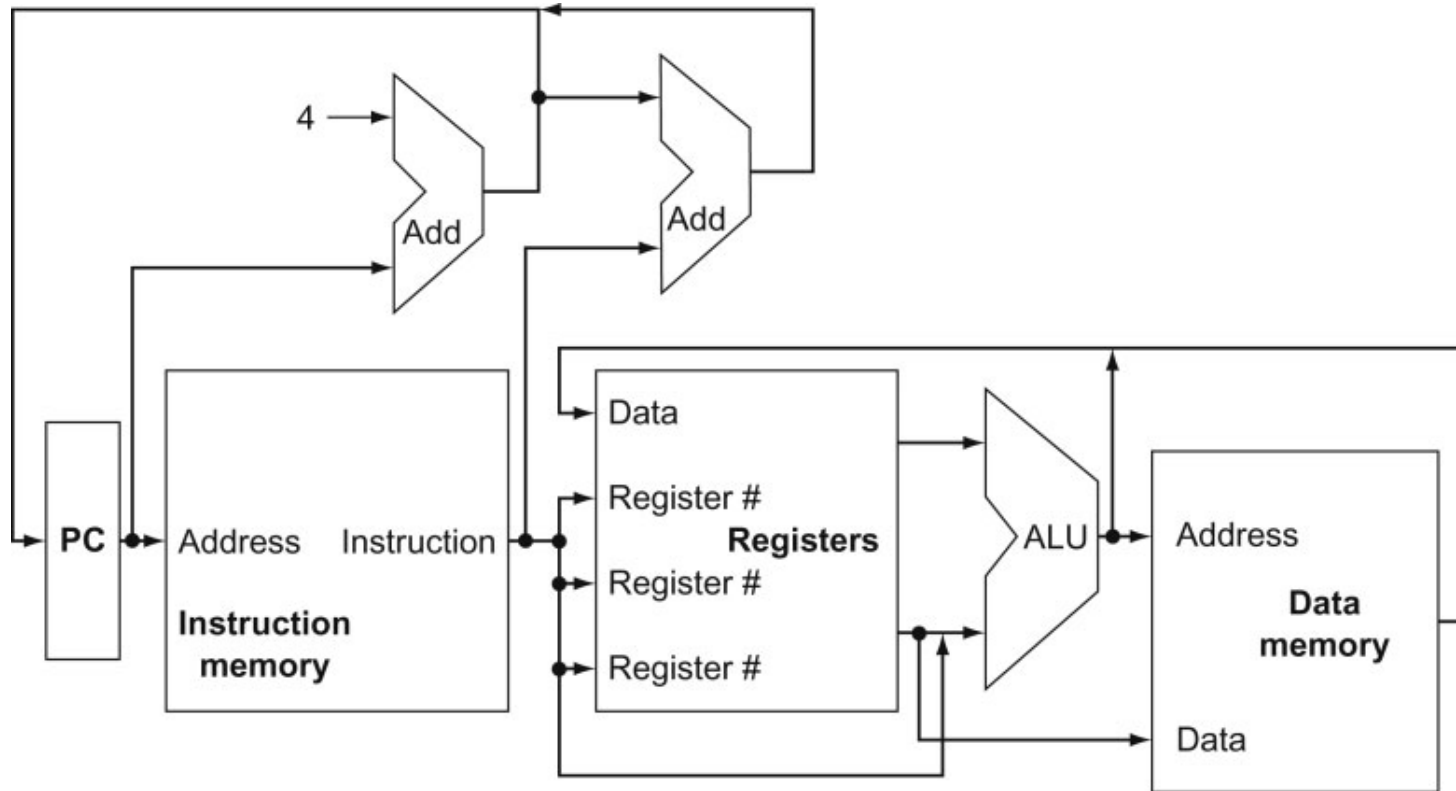


Note: we haven't bothered showing multiplexors

- 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

Source: H&P textbook

# Clocking Methodology



Source: H&P textbook

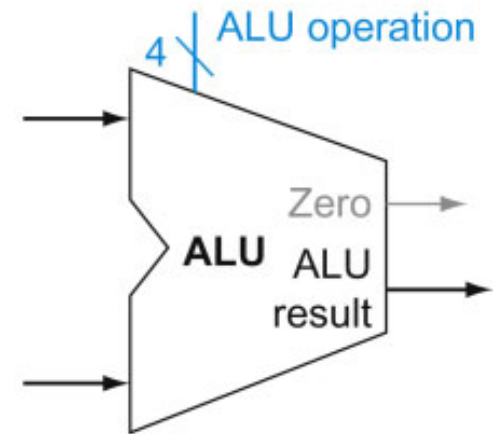
- Which of the above units need a clock?
- What is being saved (latched) on the rising edge of the clock?  
Keep in mind that the latched value remains there for an entire cycle

# Implementing R-type Instructions

- Instructions of the form `add $t1, $t2, $t3`
- Explain the role of each signal



a. Registers

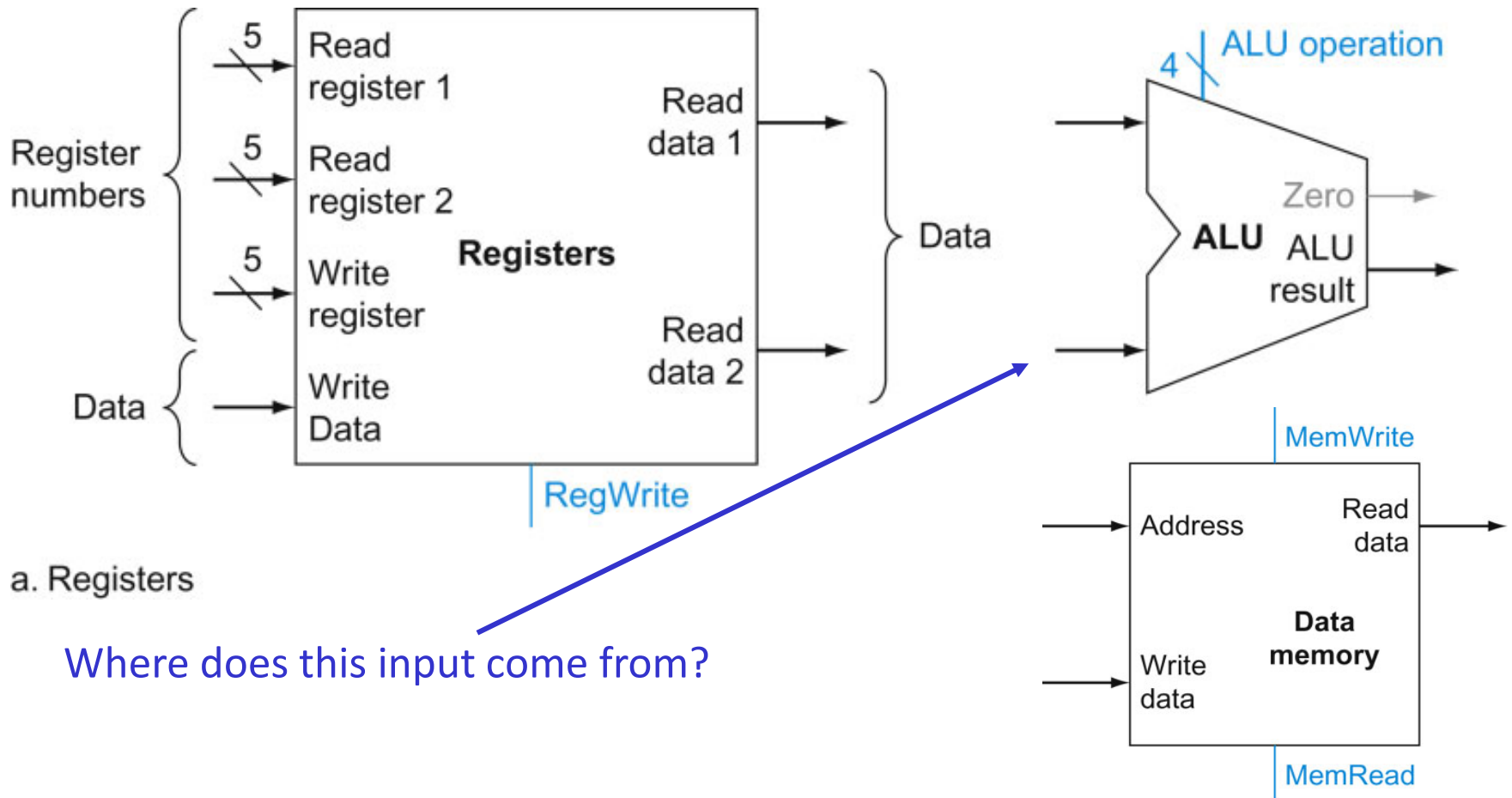


b. ALU

Source: H&P textbook

# Implementing Loads/Stores

- Instructions of the form `lw $t1, 8($t2)` and `sw $t1, 8($t2)`

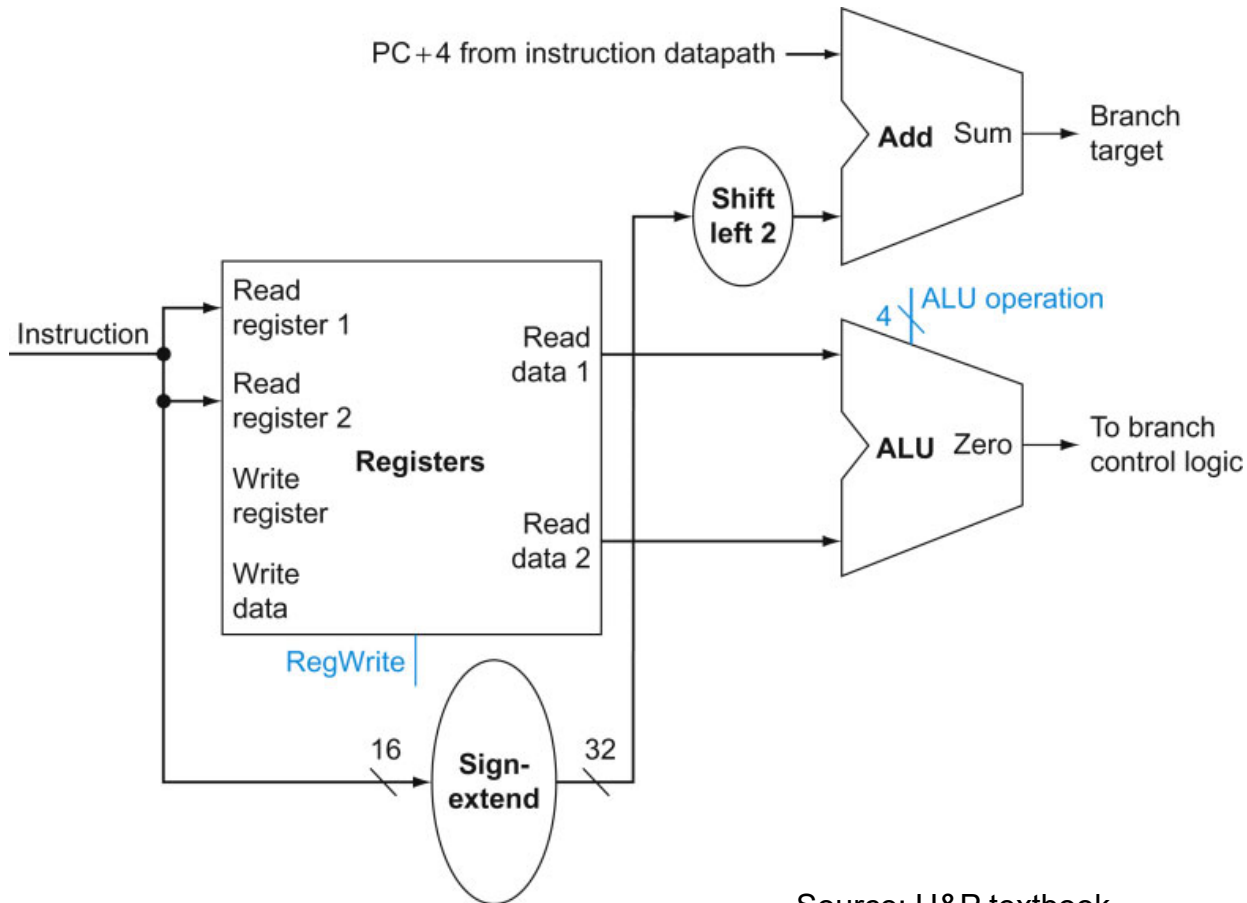


a. Registers

a. Data memory unit Source: H&P textbook

# Implementing J-type Instructions

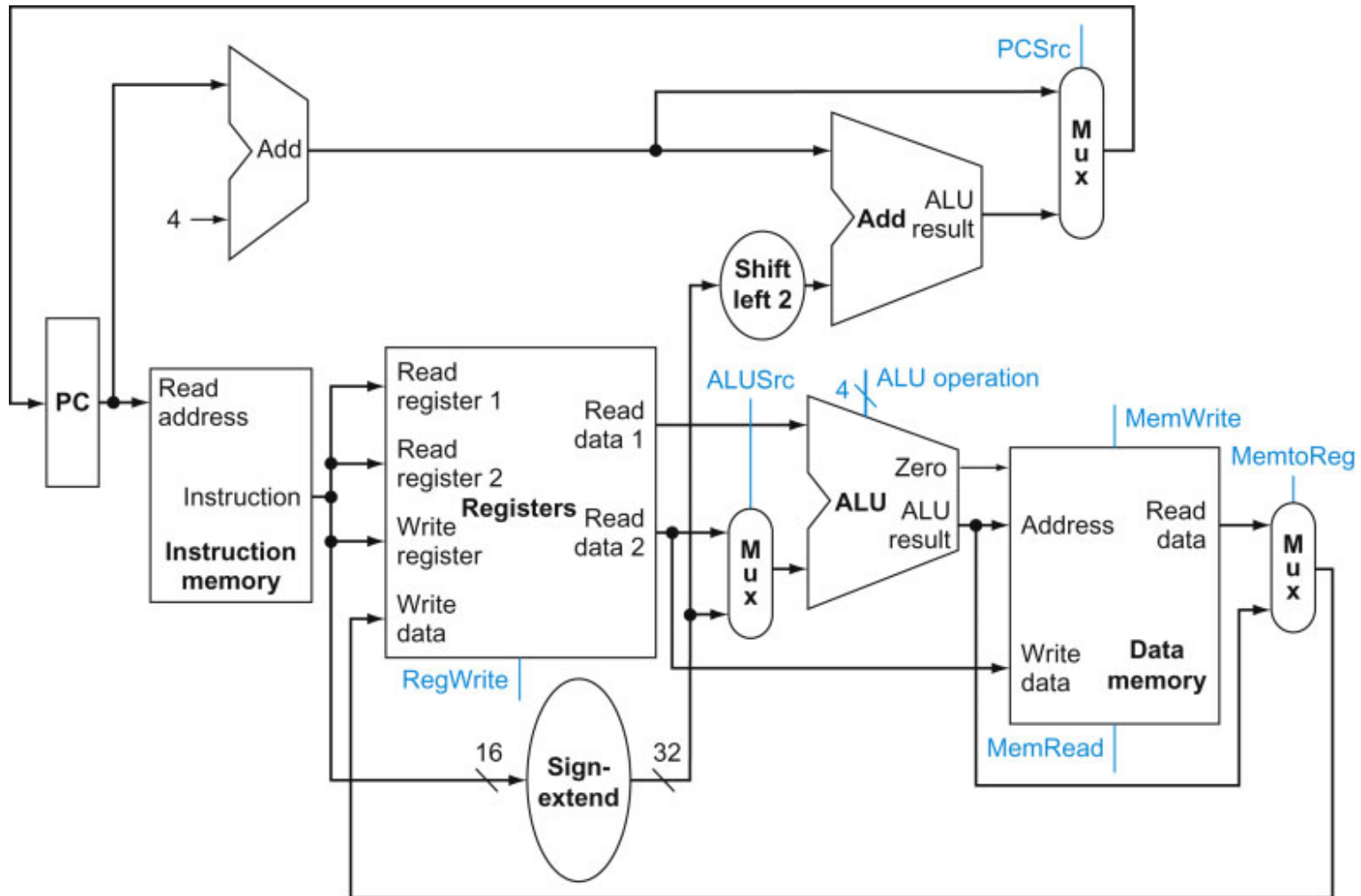
- Instructions of the form `beq $t1, $t2, offset`



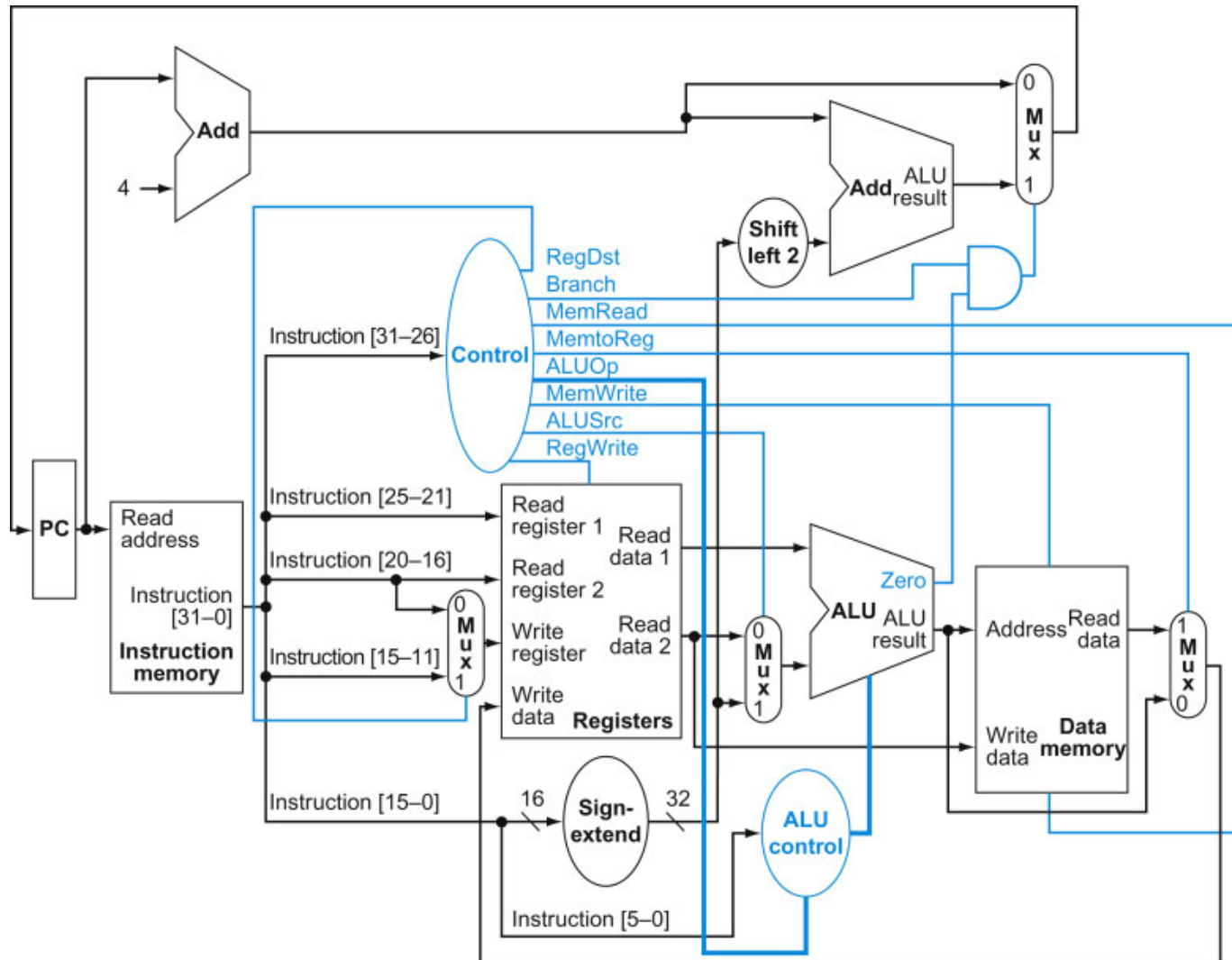
Source: H&P textbook



# View from 10,000 Feet

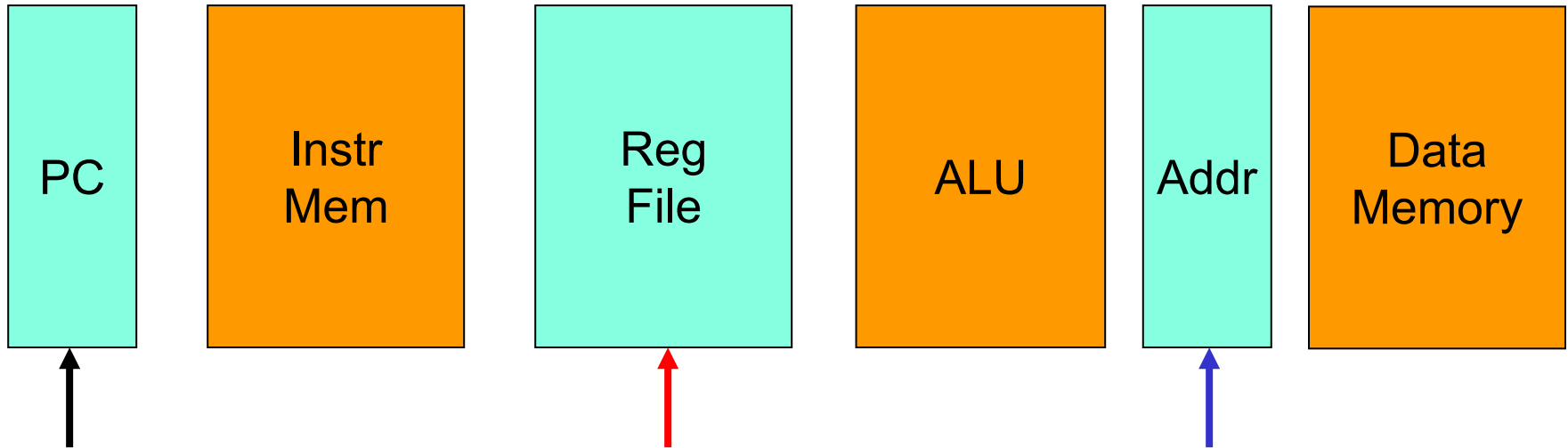






# View from 5,000 Feet



# Latches and Clocks in a Single-Cycle Design

---

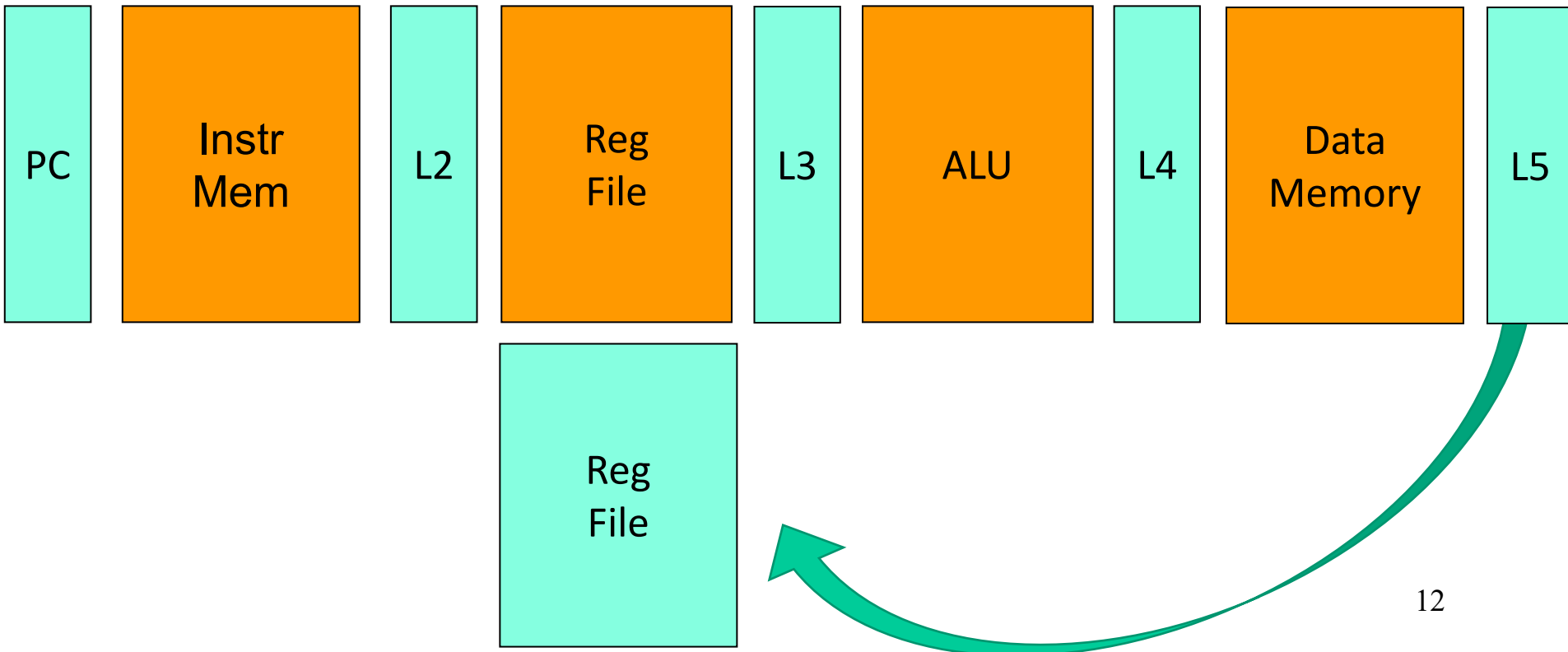


- The entire instruction executes in a single cycle
- Green blocks are latches
- At the rising edge, a new PC is recorded 
- At the rising edge, the result of the previous cycle is recorded 
- At the falling edge, the address of LW/SW is recorded so  we can access the data memory in the 2<sup>nd</sup> half of the cycle 

# Multi-Stage Circuit

---

Instead of executing the entire instruction in a single cycle (a single stage), let's break up the execution into multiple stages, each separated by a latch



# The Assembly Line

