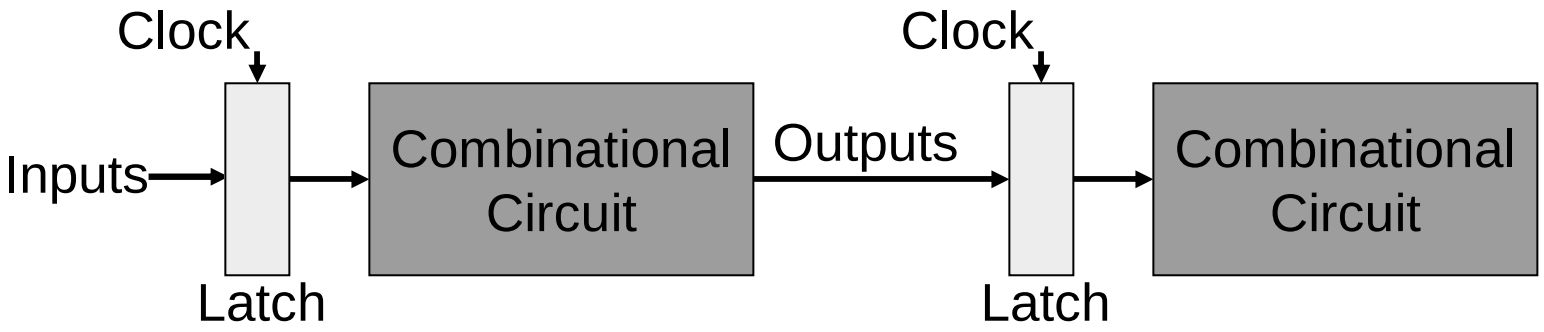
Lecture 14: FSM and Basic CPU Design

- Today's topics:
  - Finite state machines
  - Single-cycle CPU
- Reminder: midterm on Tue 10/24
  - will cover Chapters 1-4, App A, B
  - shorter than last year
  - if you understand all slides, assignments, you will ace 90% of the test

Sequential Circuits

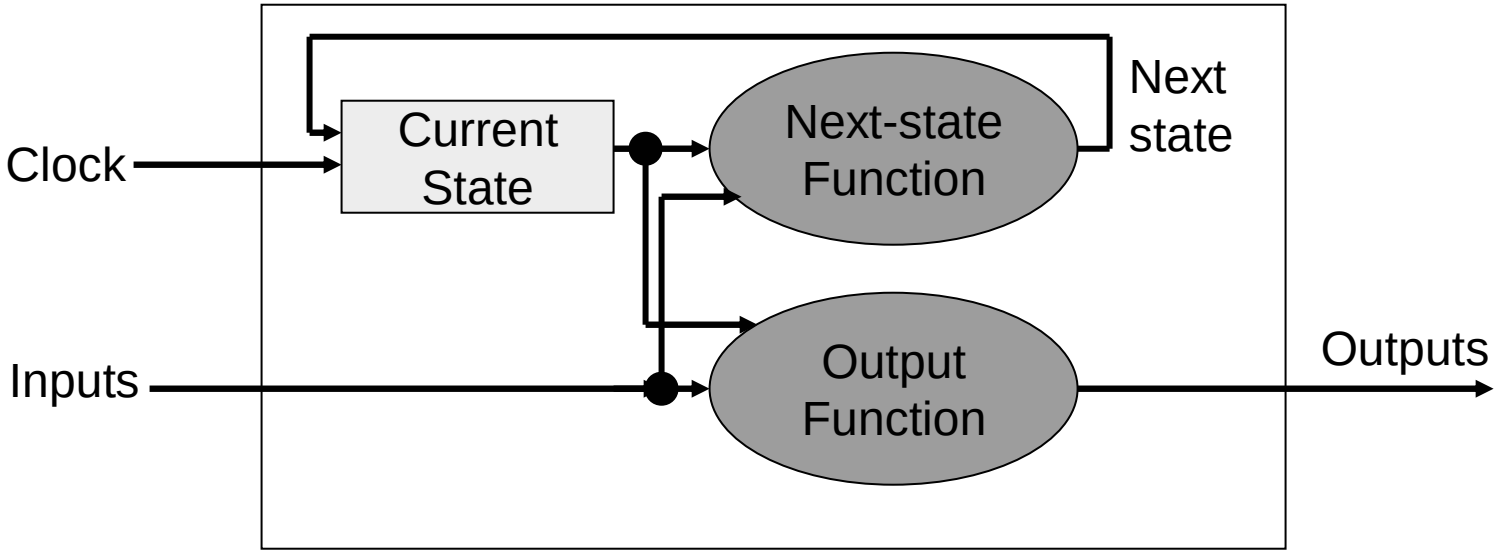
- We want the clock to act like a start and stop signal – a “latch” is a storage device that stores its inputs at a rising clock edge and this storage will not change until the next rising clock edge



Finite State Machine

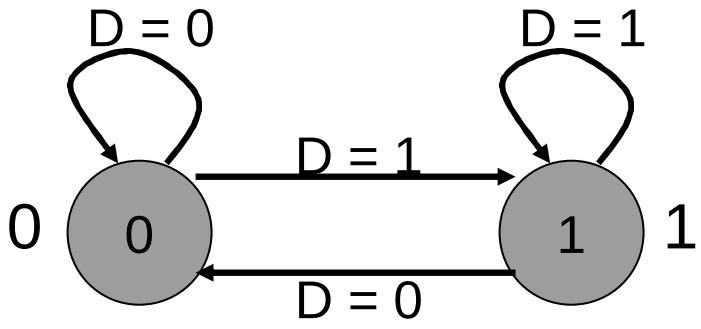
- A sequential circuit is described by a variation of a truth table – a finite state diagram (hence, the circuit is also called a finite state machine)

- Note that state is updated only on a clock edge



State Diagrams

- Each state is shown with a circle, labeled with the state value – the contents of the circle are the outputs
- An arc represents a transition to a different state, with the inputs indicated on the label



This is a state diagram for ___?

3-Bit Counter

- Consider a circuit that stores a number and increments the value on every clock edge – on reaching the largest value, it starts again from 0

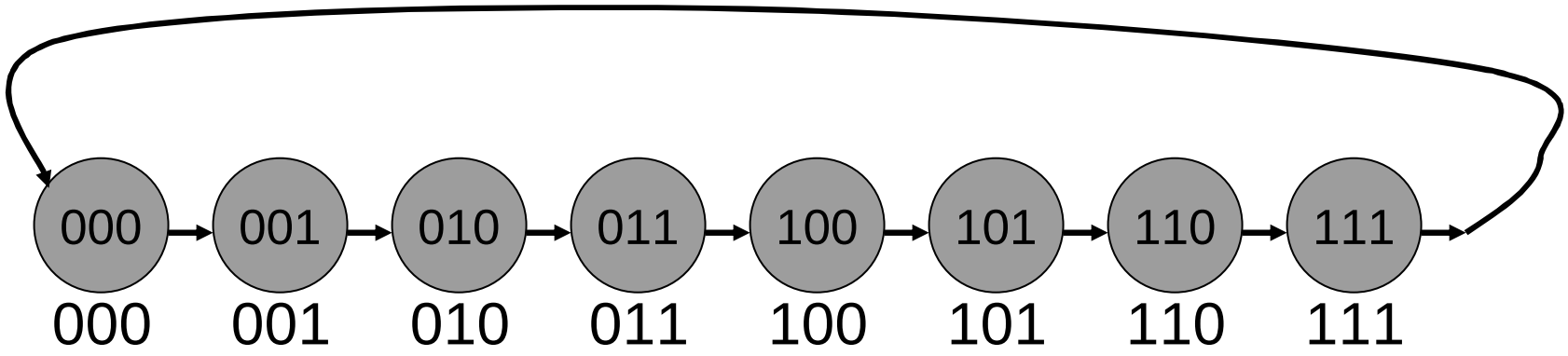
  Draw the state diagram:
  - How many states?
  - How many inputs?

3-Bit Counter

- Consider a circuit that stores a number and increments the value on every clock edge – on reaching the largest value, it starts again from 0

Draw the state diagram:

- How many states?
- How many inputs?



Traffic Light Controller

- Problem description: A traffic light with only green and red; either the North-South road has green or the East-West road has green (both can't be red); there are detectors on the roads to indicate if a car is on the road; the lights are updated every 30 seconds; a light need change only if a car is waiting on the other road

  State Transition Table:
  How many states?
  How many inputs?
  How many outputs?

State Transition Table

- Problem description: A traffic light with only green and red; either the North-South road has green or the East-West road has green (both can't be red); there are detectors on the roads to indicate if a car is on the road; the lights are updated every 30 seconds; a light must change only if a car is waiting on the other road

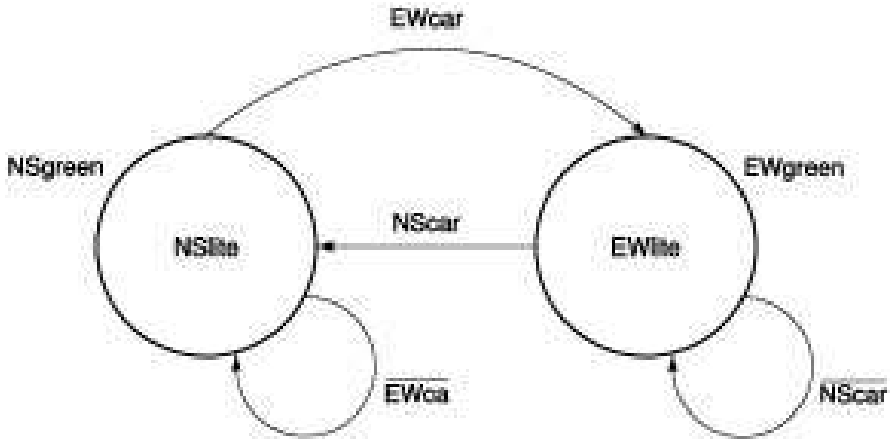
State Transition Table:

CurrState	InputEW	InputNS	NextState=Output
N	0	0	N
N	0	1	N
N	1	0	E
N	1	1	E
E	0	0	E
E	0	1	N
E	1	0	E
E	1	1	N

State Diagram

State Transition Table:

CurrState	InputEW	InputNS	NextState=Output
N	0	0	N
N	0	1	N
N	1	0	E
N	1	1	E
E	0	0	E
E	0	1	N
E	1	0	E
E	1	1	N



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)

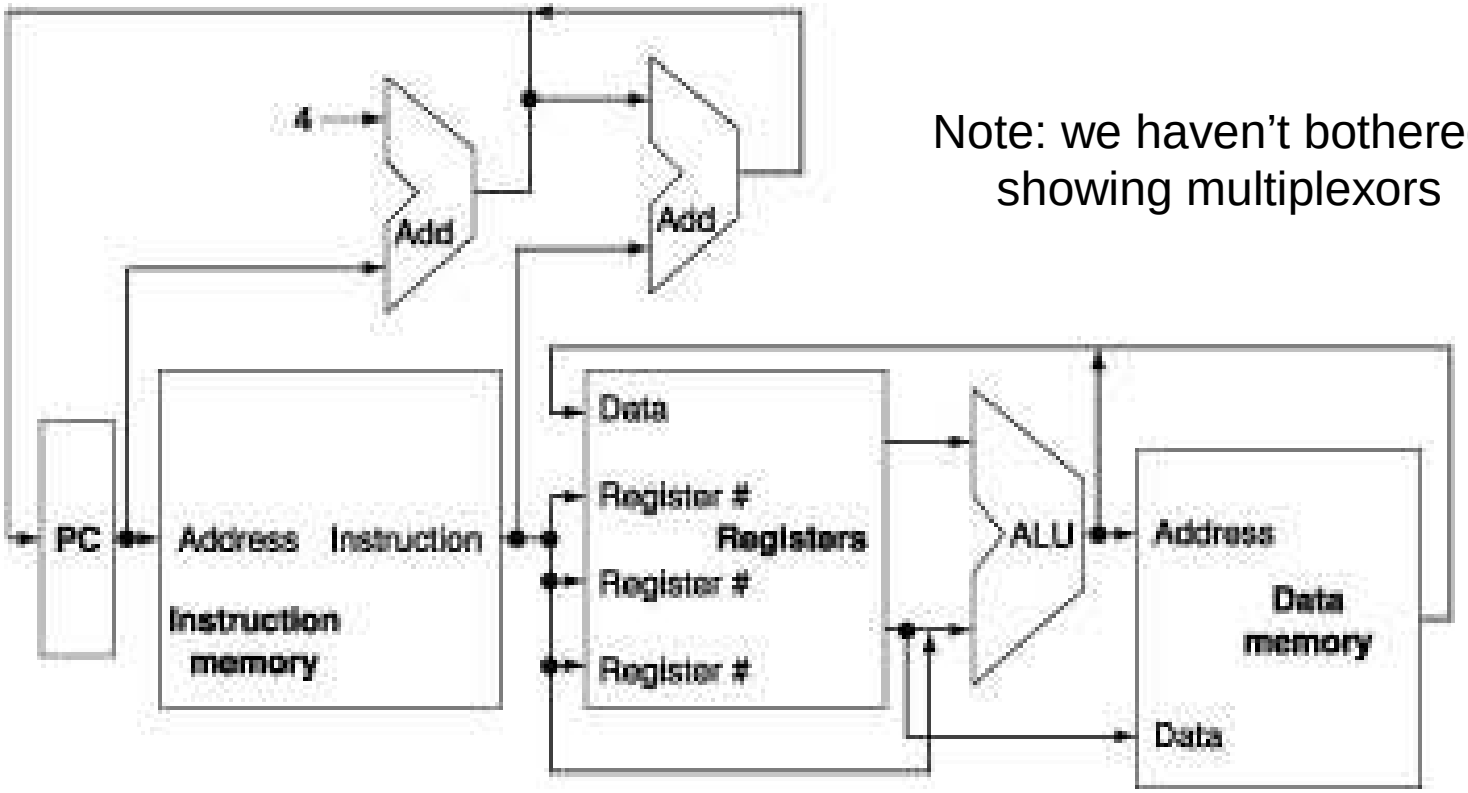
Implementation Overview

- 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

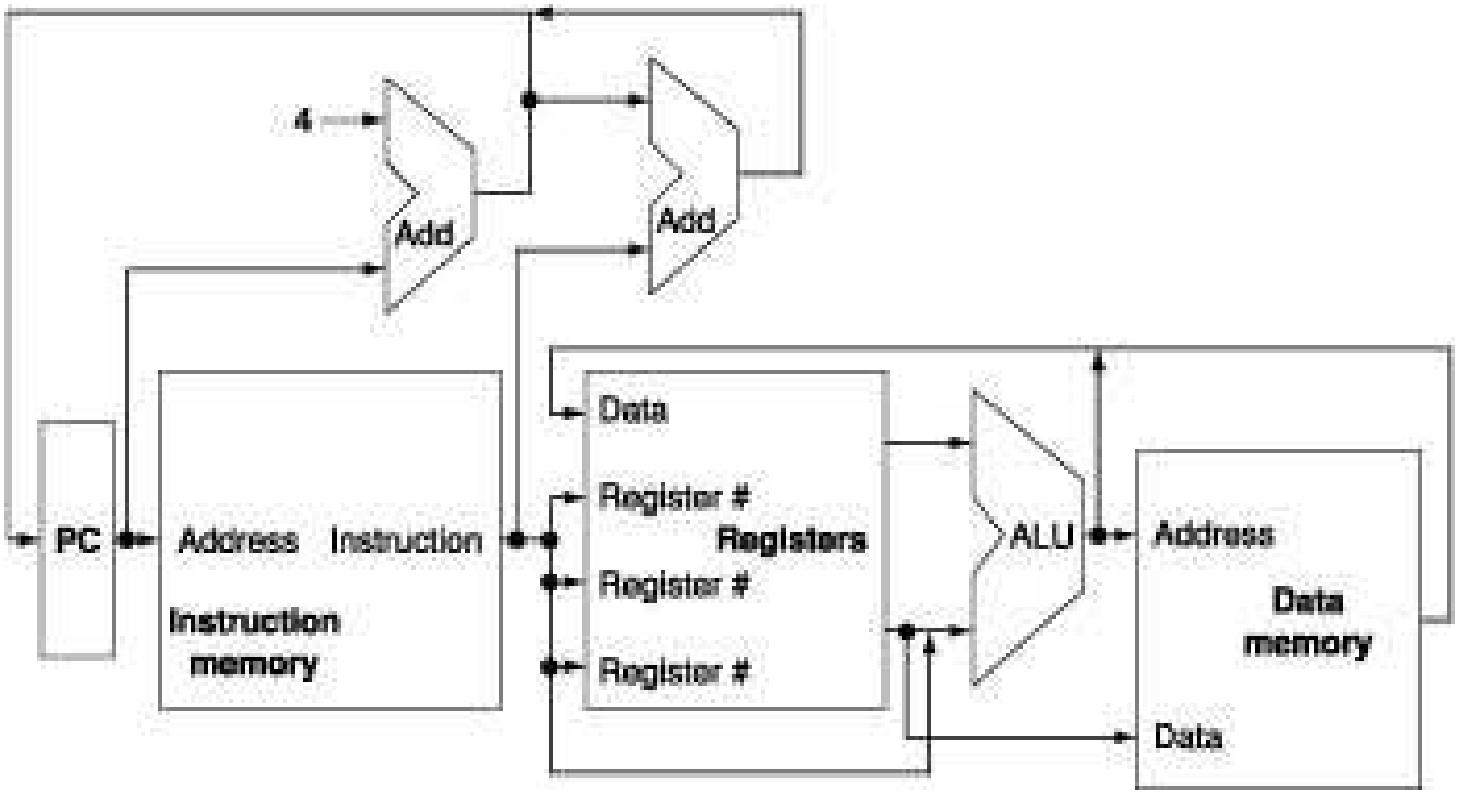
View from 30,000 Feet



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

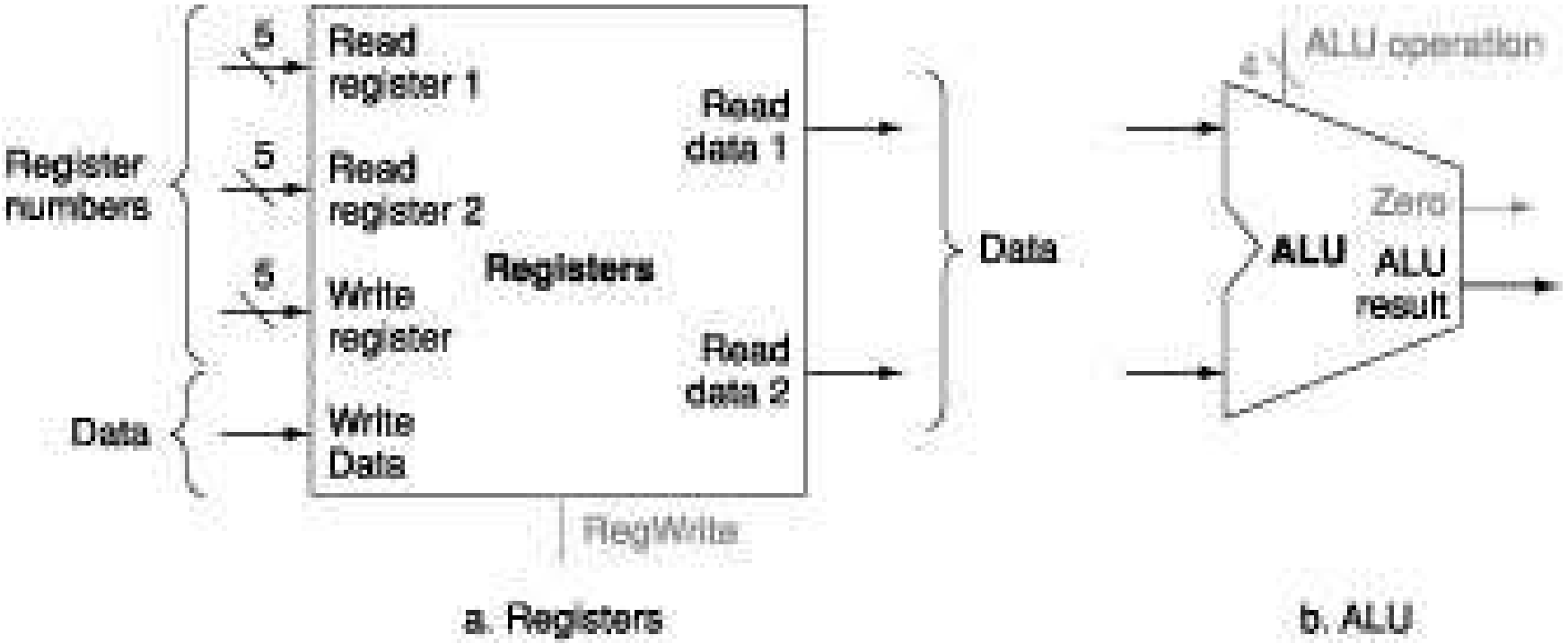
Clocking Methodology



- 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

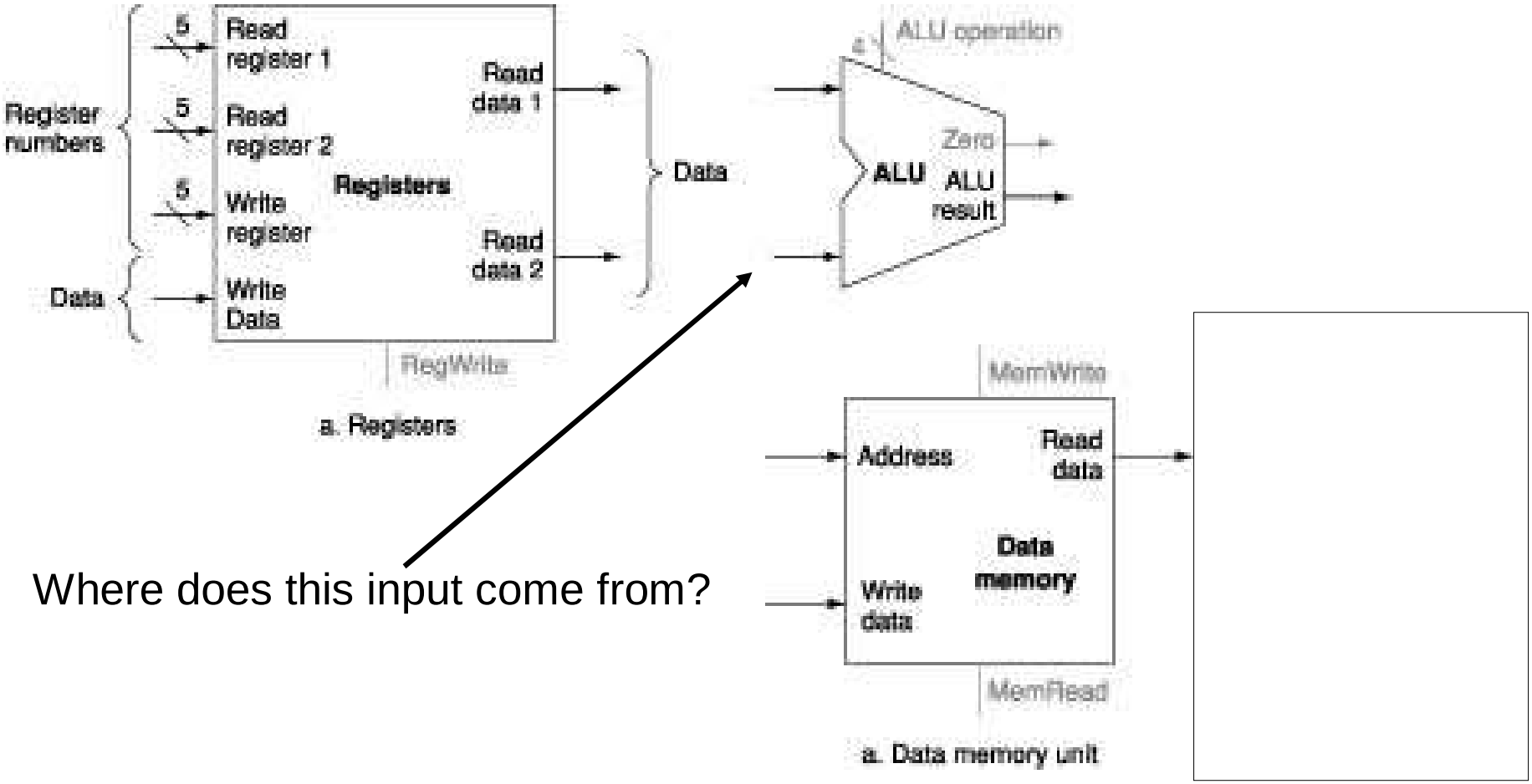
Implementing R-type Instructions

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



Implementing Loads/Stores

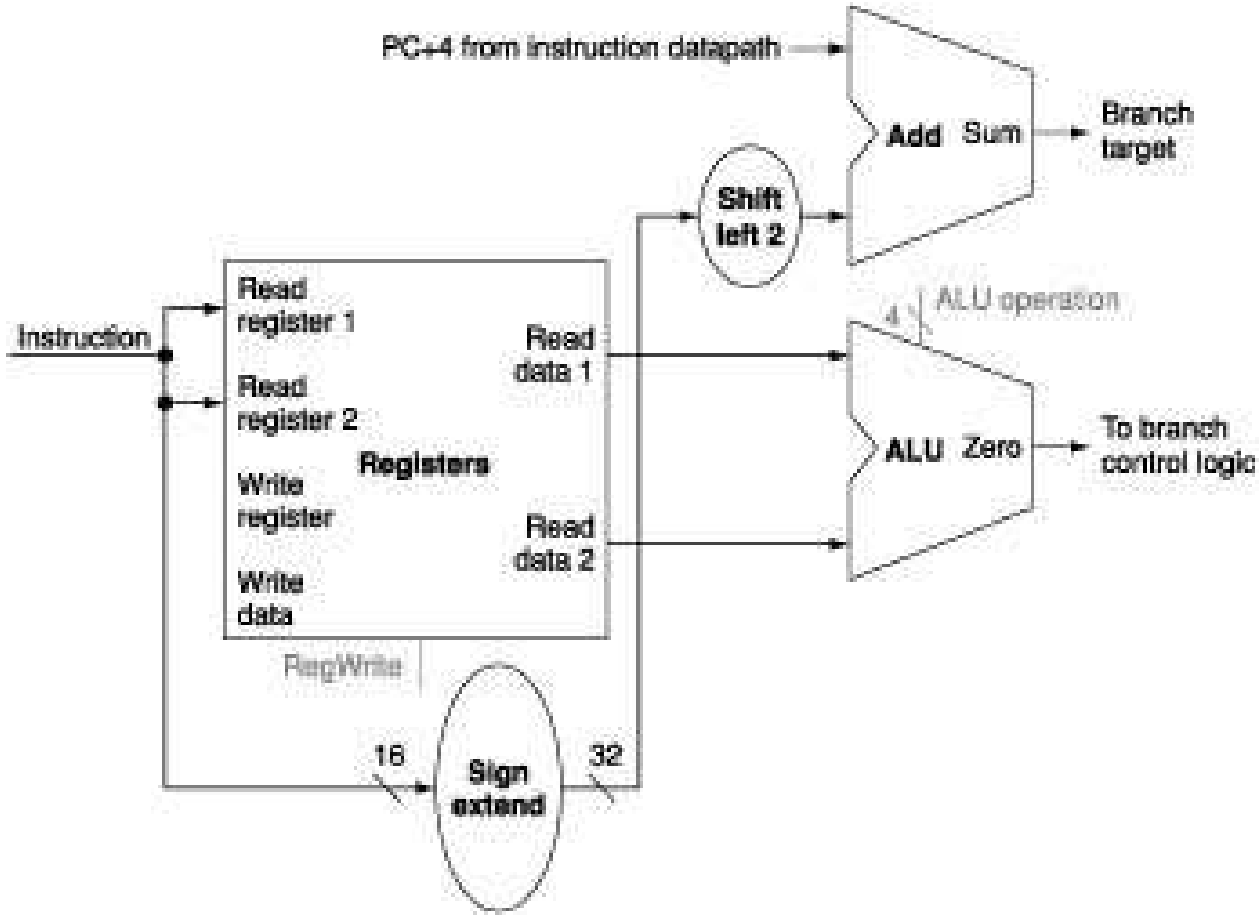
- Instructions of the form lw $t1, 8($t2) and sw $t1, 8($t2)



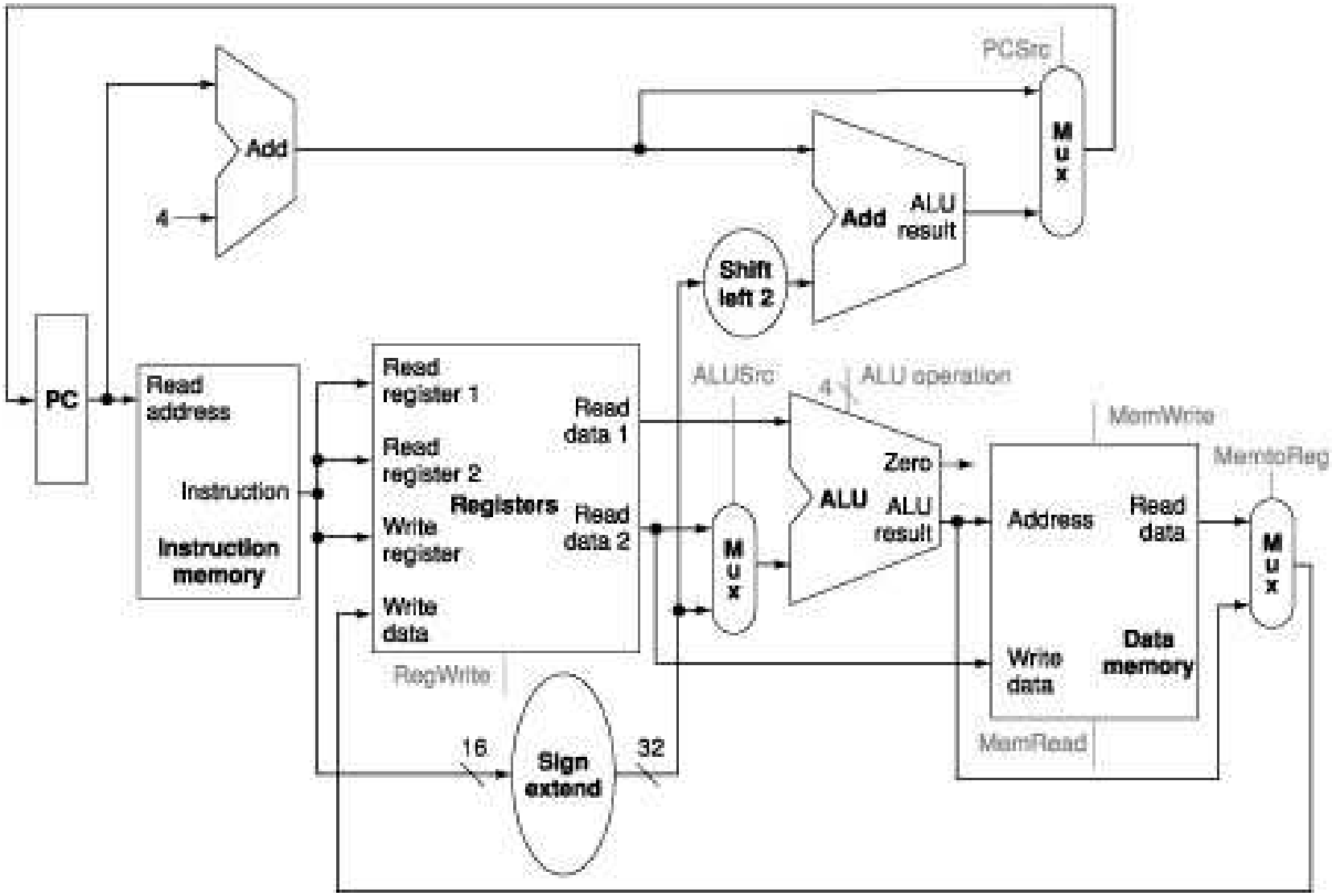
Where does this input come from?

Implementing J-type Instructions

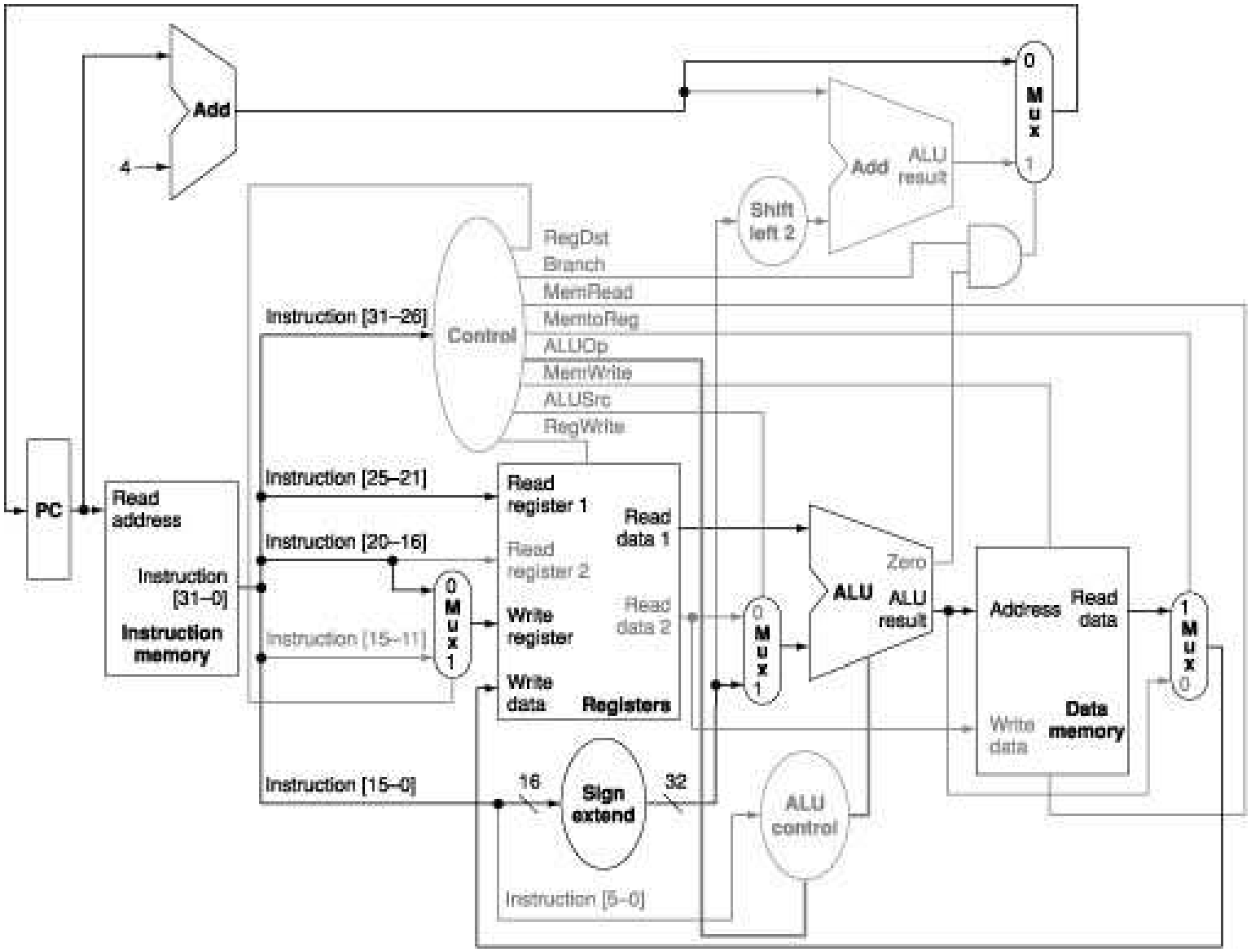
- Instructions of the form beq $t1, $t2, offset



View from 10,000 Feet



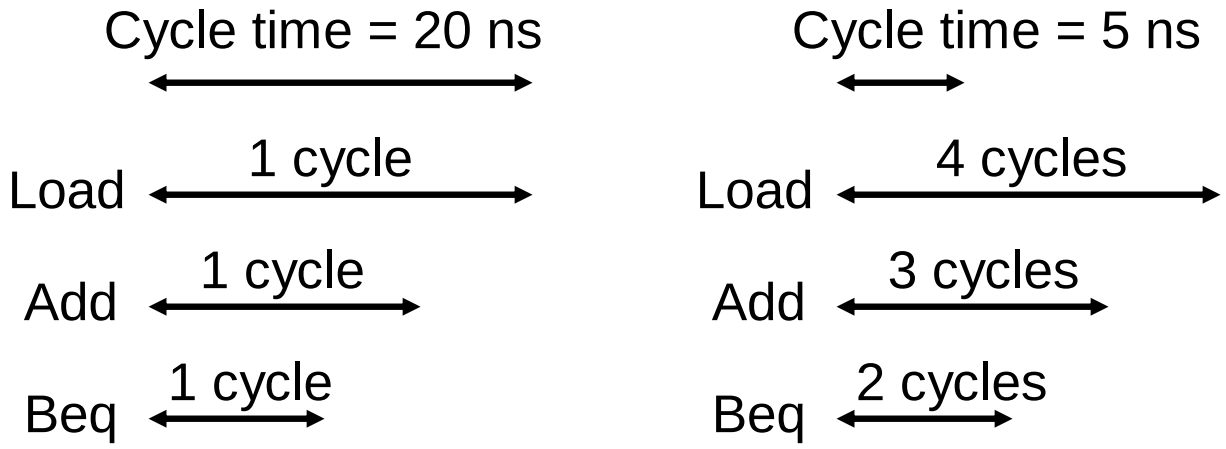
View from 5,000 Feet



Single Vs. Multi-Cycle Machine

- In this implementation, every instruction requires one cycle to complete ➔ cycle time = time taken for the slowest instruction

- If the execution was broken into multiple (faster) cycles, the shorter instructions can finish sooner



Time for a load, add, and beq = 60 ns 45 ns

Title

- Bullet