Lecture 15: Basic CPU Design

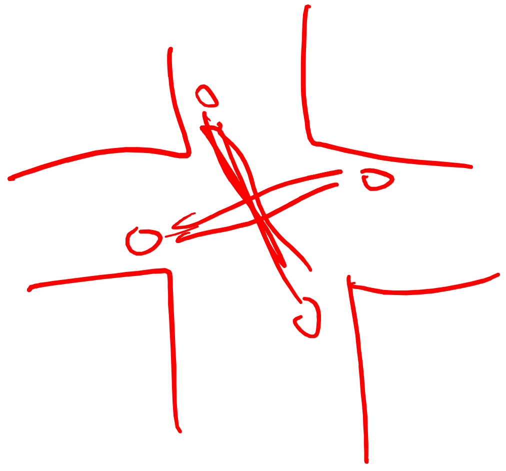
- Today's topics:
  - FSM examples
  - Single-cycle CPU
  - Multi-cycle CPU

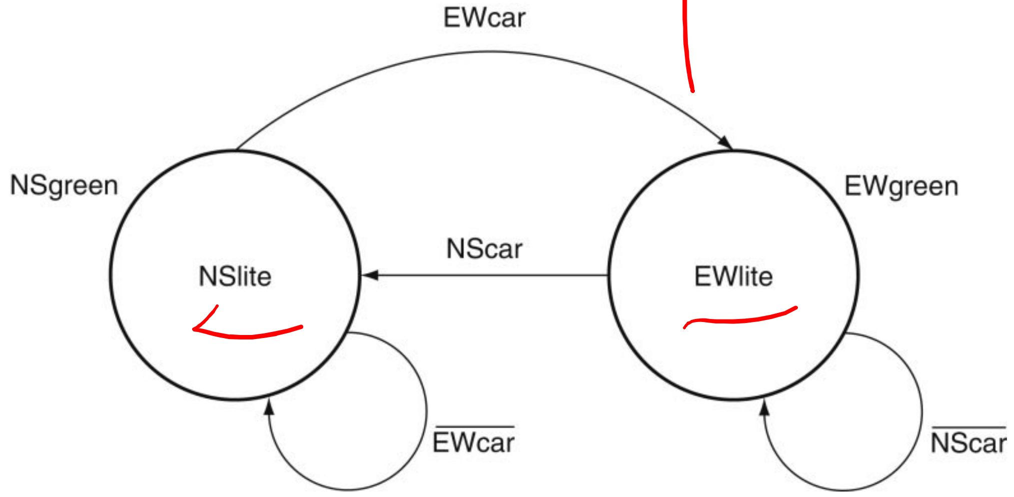
Traffic light example

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





$\overline{\text{NScar}}$

Source: H&P textbook

Tackling FSM Problems

- Three questions worth asking:
  - What are the possible output states? Draw a bubble for each.
  - What are inputs? What values can those inputs take?
  - For each state, what do I do for each possible input value? Draw an arc out of every bubble for every input value.

#rows in table = $\underset{\text{states}}{3} \times \underset{\text{EXT Val}}{2} \times \underset{\text{INT}}{3} = 18$

~~Val~~

Example – Residential Thermostat

70°

INT HT ← 69° 71° → AC

- Two temp sensors: internal and external
- If internal temp is within 1 degree of desired, don't change setting → EXT 65° ← OFF → 75°
- If internal temp is > 1 degree higher than desired, turn AC on; if internal temp is < 1 degree lower than desired, turn heater on
- If external temp and desired temp are within 5 degrees, disregard the internal temp, and turn both AC and heater off

3 diff actions for int ther

# inputs = 2

input1 (INT) values = 3
C G H

input2 (EXT) values = 2
Under Desired range

HT ON
AC ON
BOTH OFF

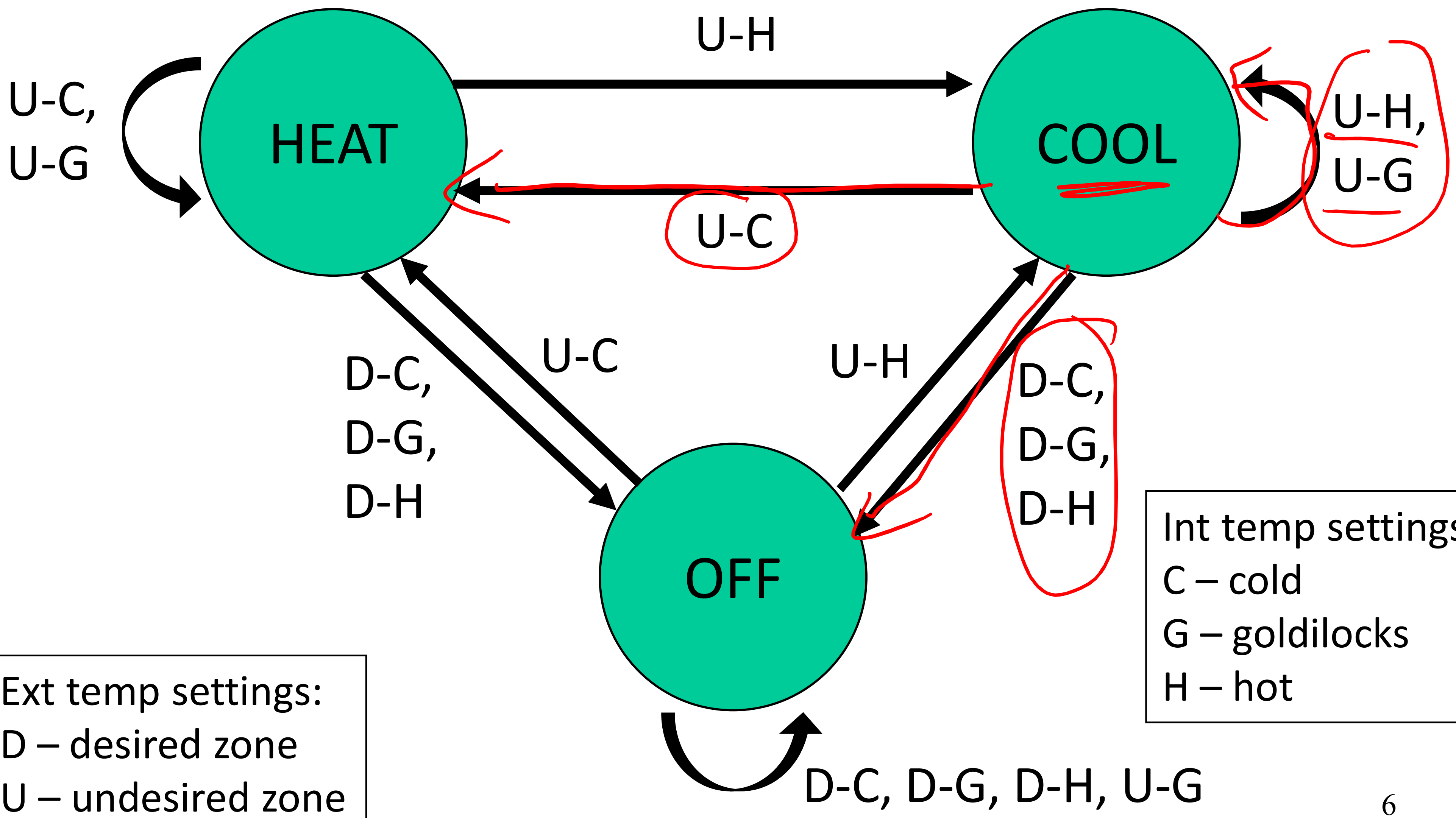
3 output states

Finite State Machine Table

Current State	Input E	Input I	Output State
	D U	C G H	
HEAT	D	C	OFF
HEAT	D	G	OFF
HEAT	D	H	OFF
HEAT	U	C	HEAT
HEAT	U	G	HEAT
HEAT	U	H	COOL
COOL	D	C	OFF
COOL	D	G	OFF
COOL	D	H	OFF
COOL	U	C	HEAT
COOL	U	G	COOL
COOL	U	H	COOL
OFF	D	C	OFF
OFF	D	G	OFF
OFF	D	H	OFF
OFF	U	C	HEAT
OFF	U	G	OFF
OFF	U	H	COOL

6 possible inputs

Finite State Diagram



Int temp settings:
C – cold
G – goldilocks
H – hot

Ext temp settings:
D – desired zone
U – undesired zone

Lazy allig SnappyAlligator

Latch vs. Flip-Flop

RS

2 back-2-back latches

- Recall that we want a circuit to have stable inputs for an entire cycle – so I want my new inputs to arrive at the start of a cycle and be fixed for an entire cycle

- A flip-flop provides the above semantics (a door that swings open and shut at the start of a cycle)

- But a flip-flop needs two back-to-back D-latches, i.e., more transistors, delay, power

- You can reduce these overheads with just a single D-latch (a door that is open for half a cycle) as long as you can tolerate stable inputs for just half a cycle

Basic MIPS Architecture

ALU

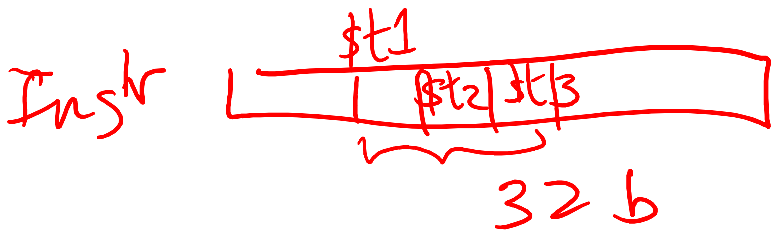
- Now that we understand clocks and storage of states, we'll design a simple CPU that executes:
  - basic math (add, sub, and, or, slt) ALU
  - memory access (lw and sw)
  - branch and jump instructions (beq and j)

PRE

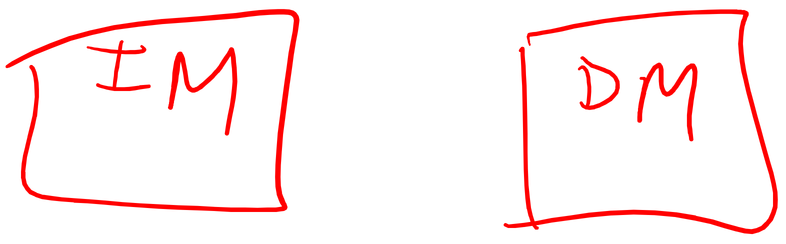
MID TERM

POST MIDTERM

Implementation Overview

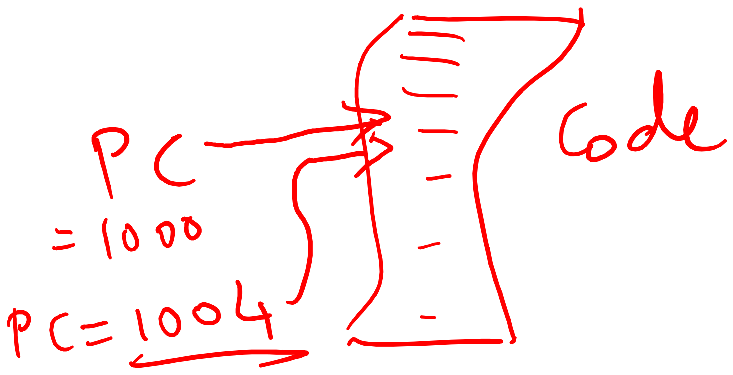


- We need memory
  - to store instructions
  - to store data
  - for now, let's make them separate units



- We need registers, ALU, and a whole lot of control logic

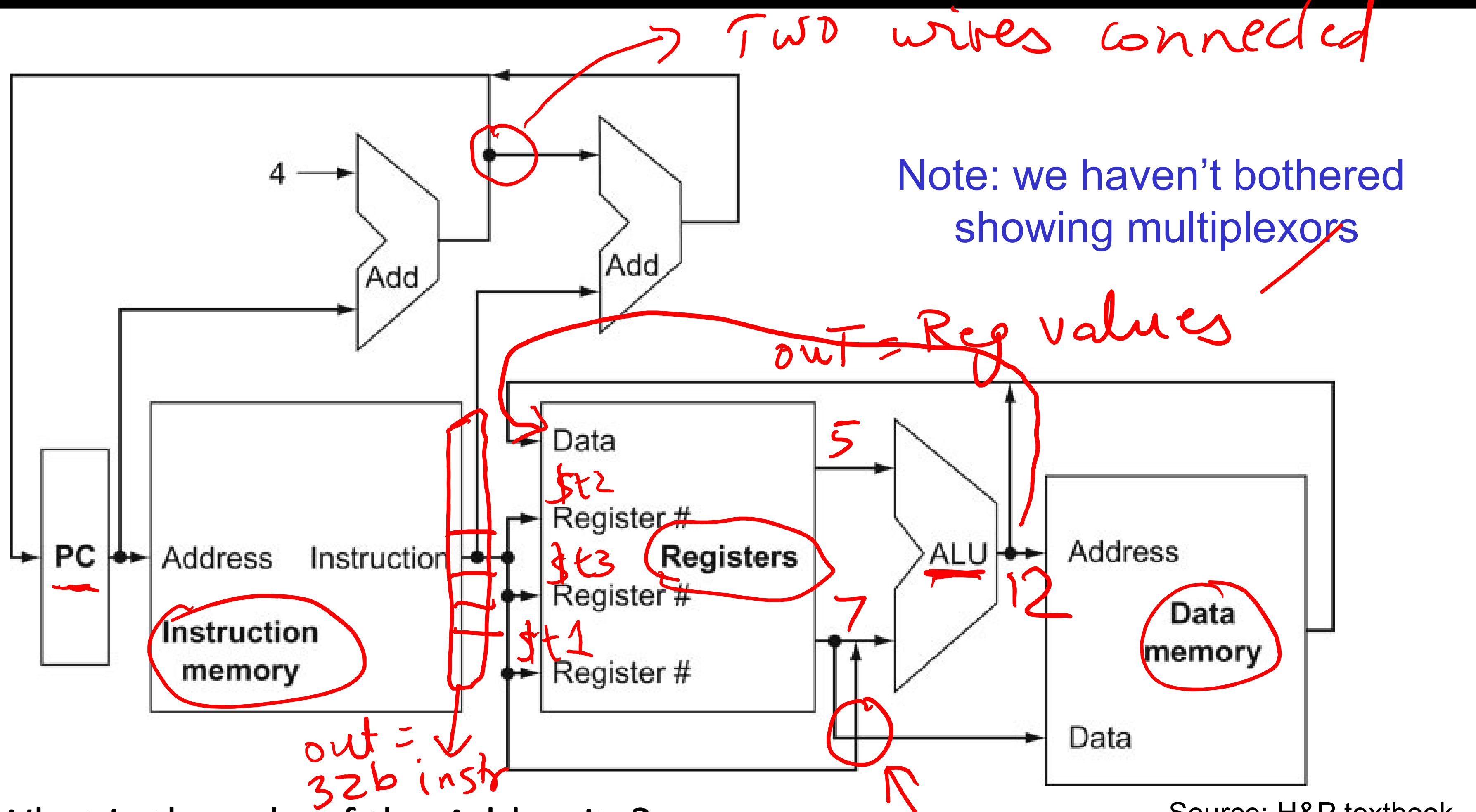
- CPU operations common to all instructions:
  - use the program counter (PC) to pull instruction out of instruction memory
  - read register values



add $t1, $t2, $t3

View from 30,000 Feet

Note: we haven't bothered showing multiplexors



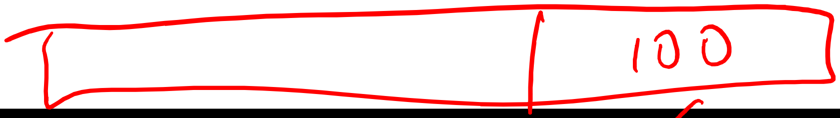
Source: H&P textbook

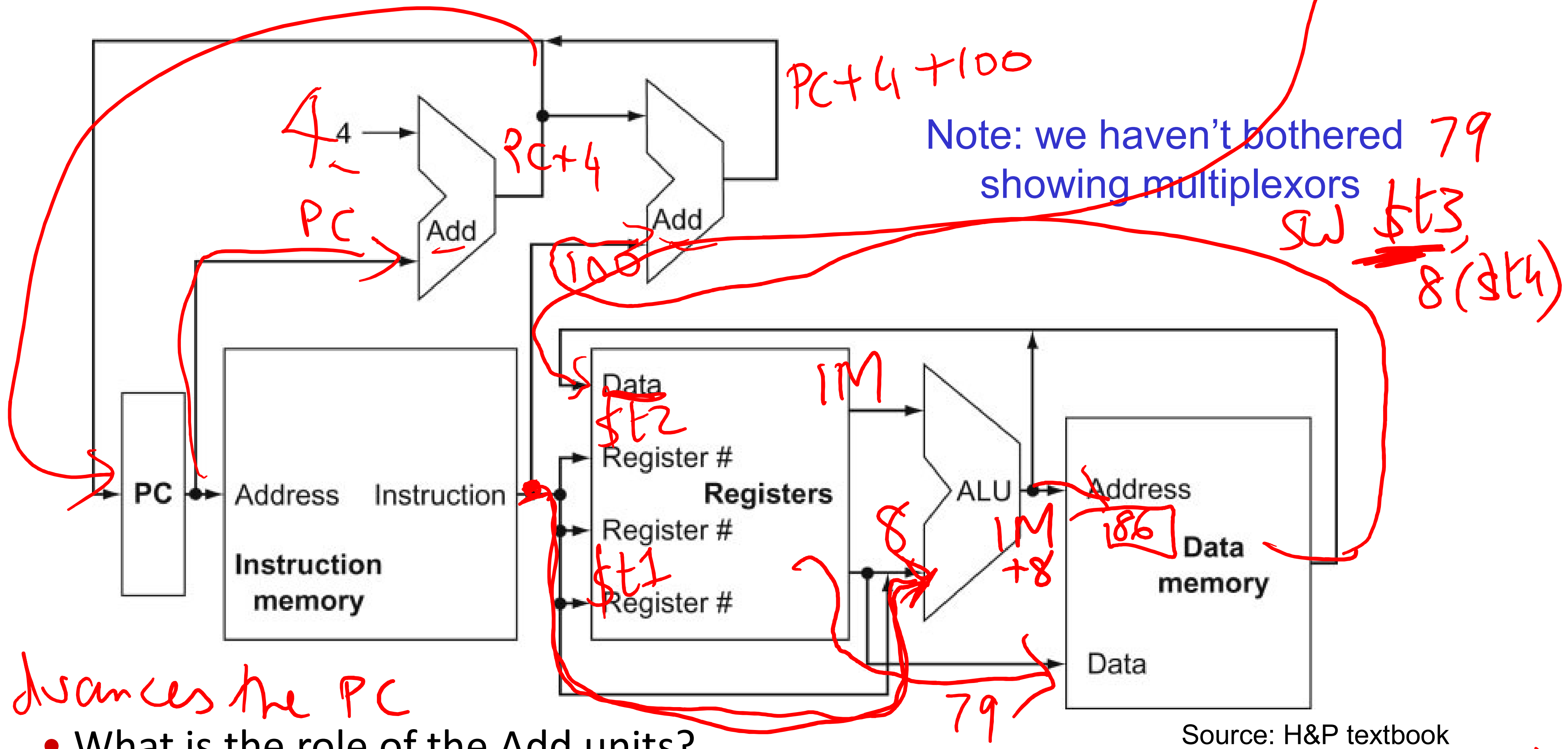
- What is the role of the Add units?
- Explain the inputs to the data memory unit
- Explain the inputs to the ALU
- Explain the inputs to the register unit

wires flying over each other

View from 30,000 Feet

BEQ, $t1, $t2, 100





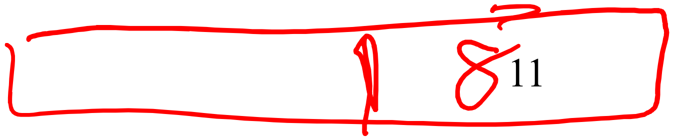
Note: we haven't bothered showing multiplexors

Source: H&P textbook

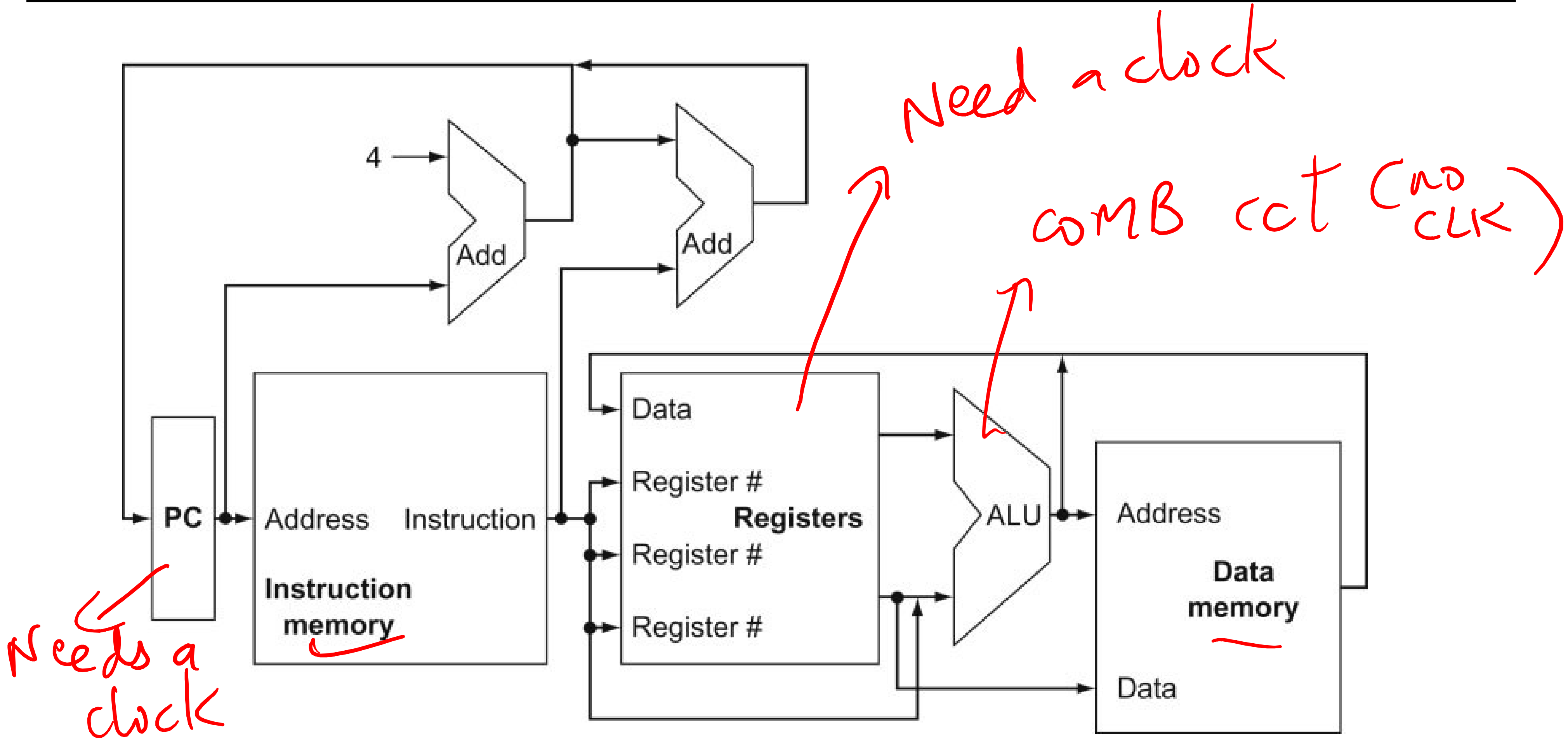
advances the PC

- What is the role of the Add units?
- Explain the inputs to the data memory unit
- Explain the inputs to the ALU either reg values or immediate
- Explain the inputs to the register unit

LW $t1, 8($t2)



Clocking Methodology

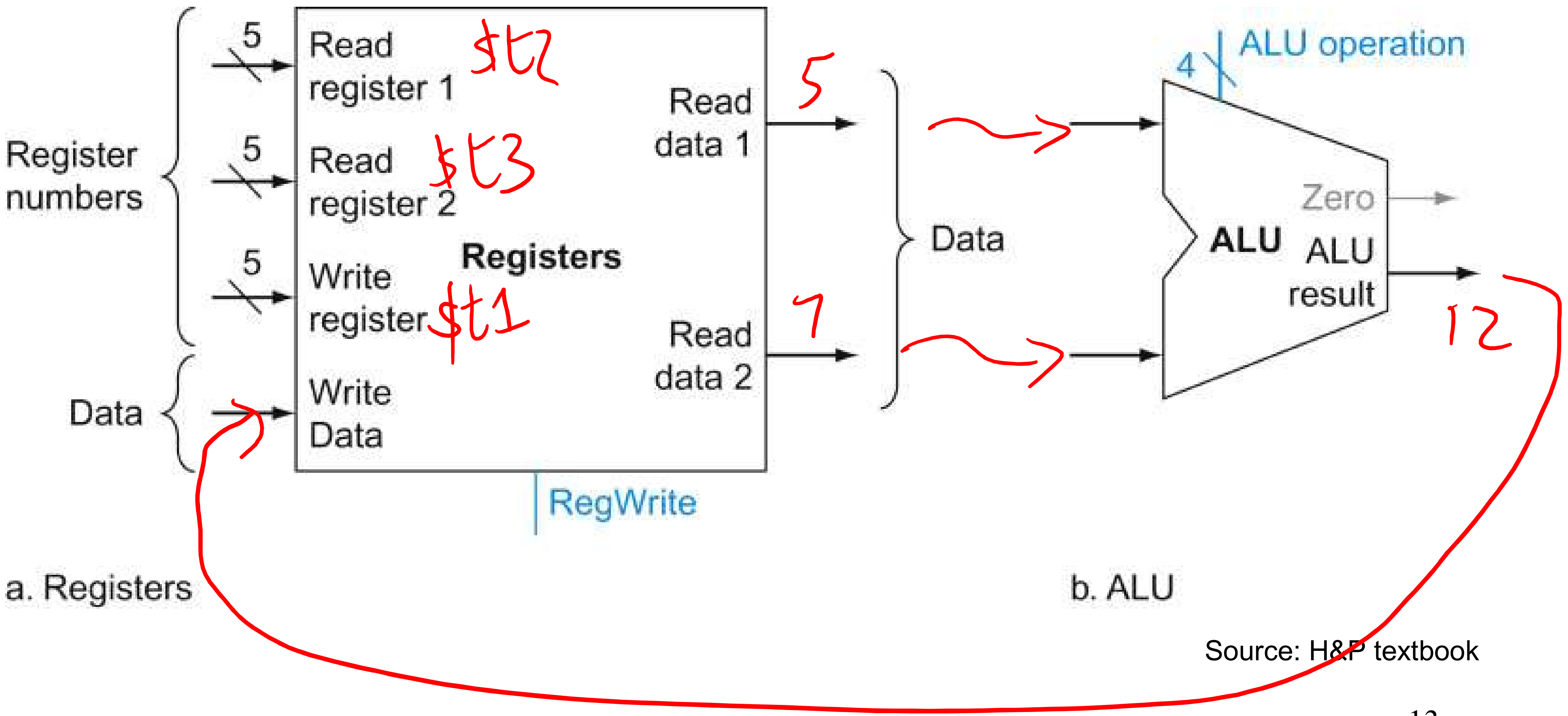


Source: H&P textbook

- 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

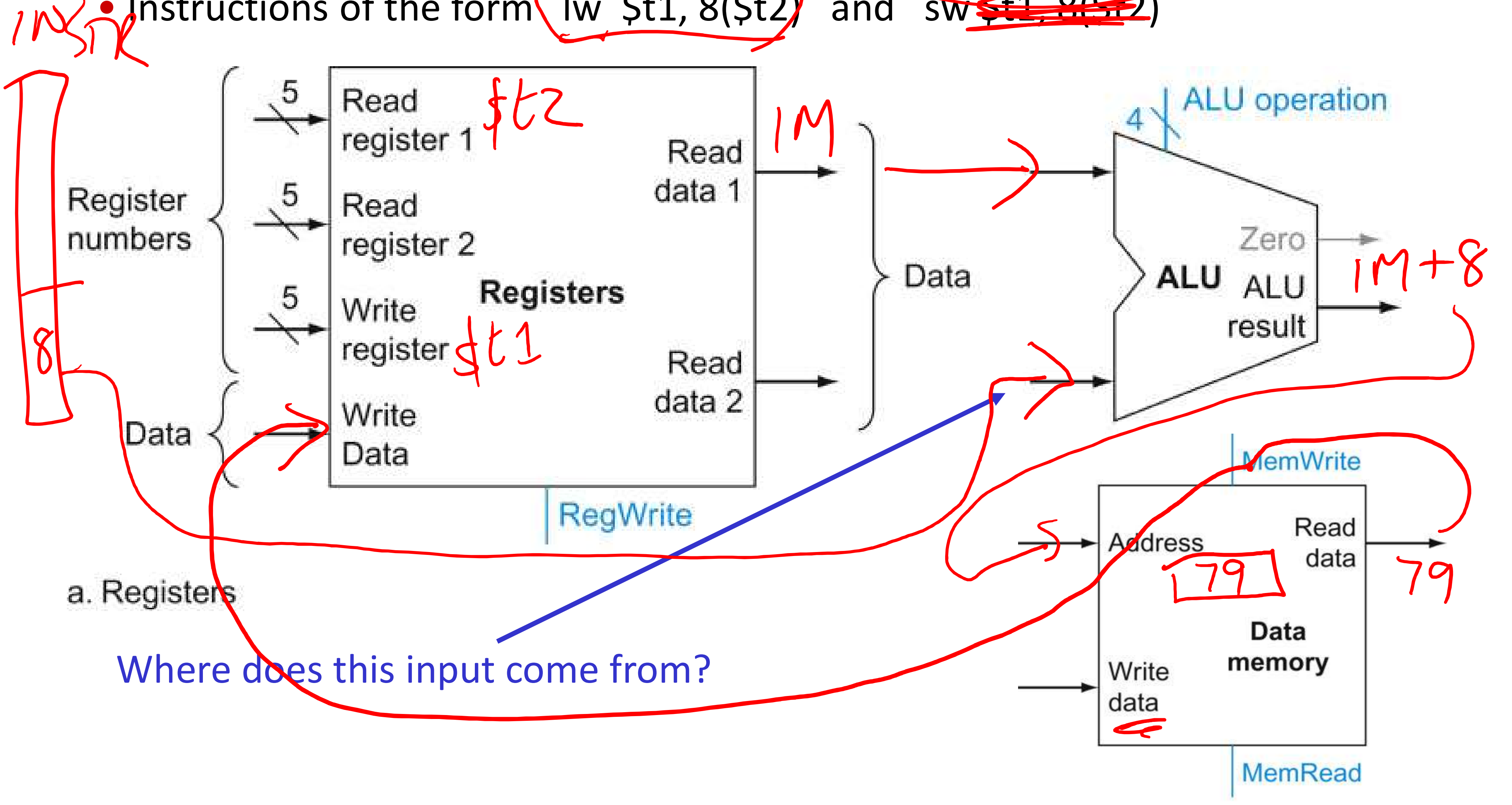
- Instructions of the form add $t1, $t2, $t3
- Explain the role of each signal



Source: H&P textbook

Implementing Loads/~~Stores~~

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

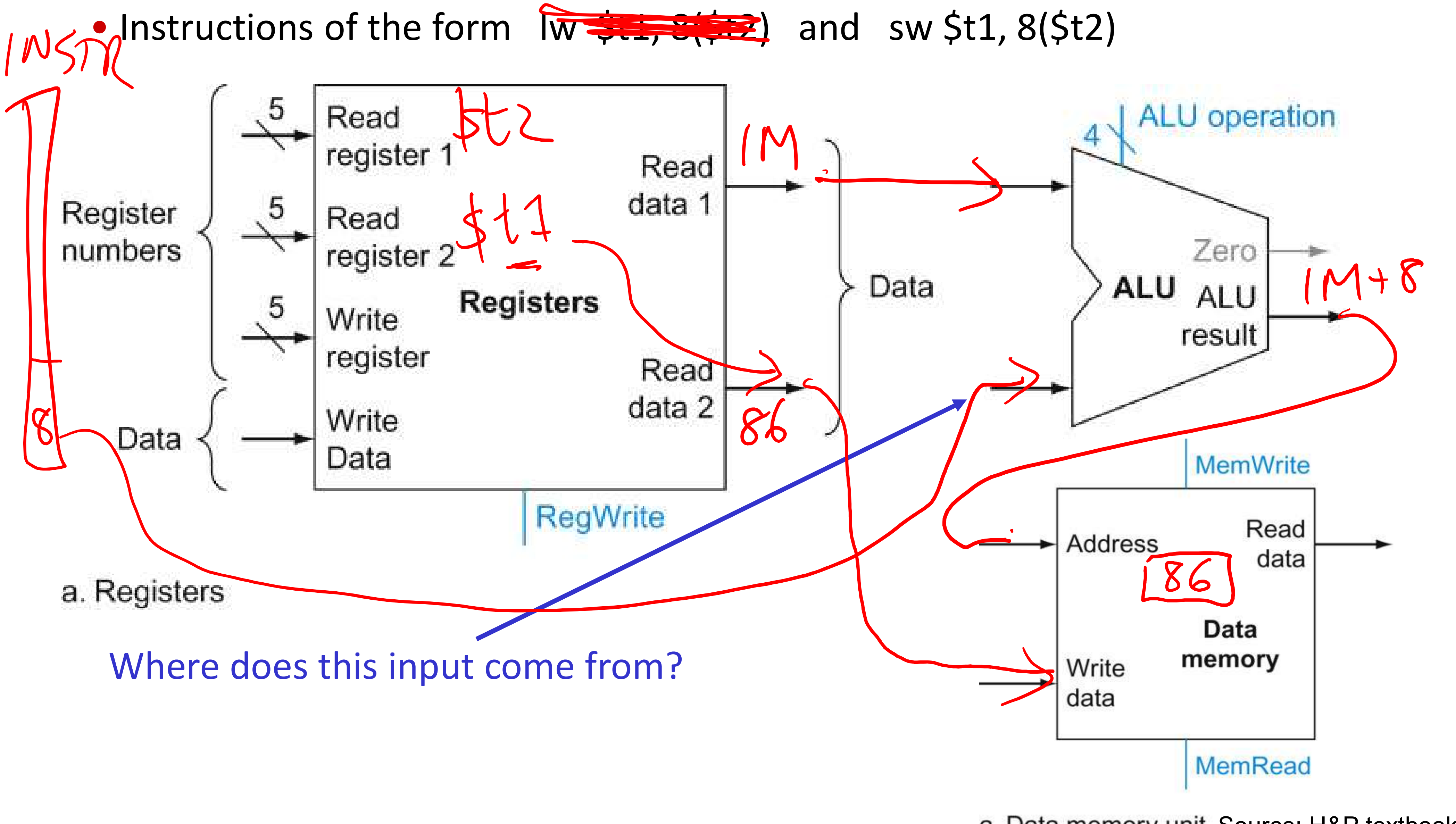


Where does this input come from?

Source: H&P textbook

Implementing ~~Loads~~s/Stores

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

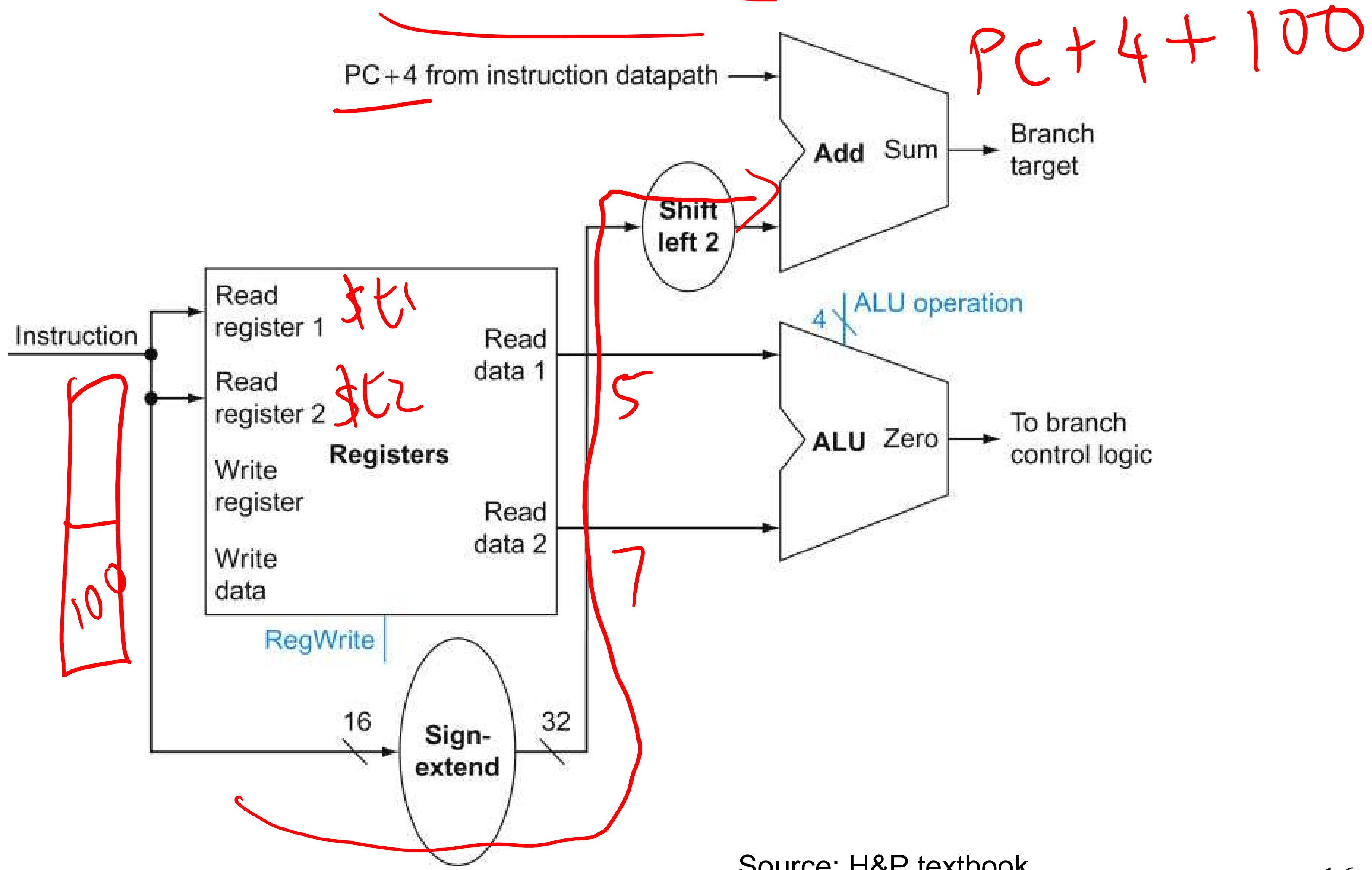


Where does this input come from?

Source: H&P textbook

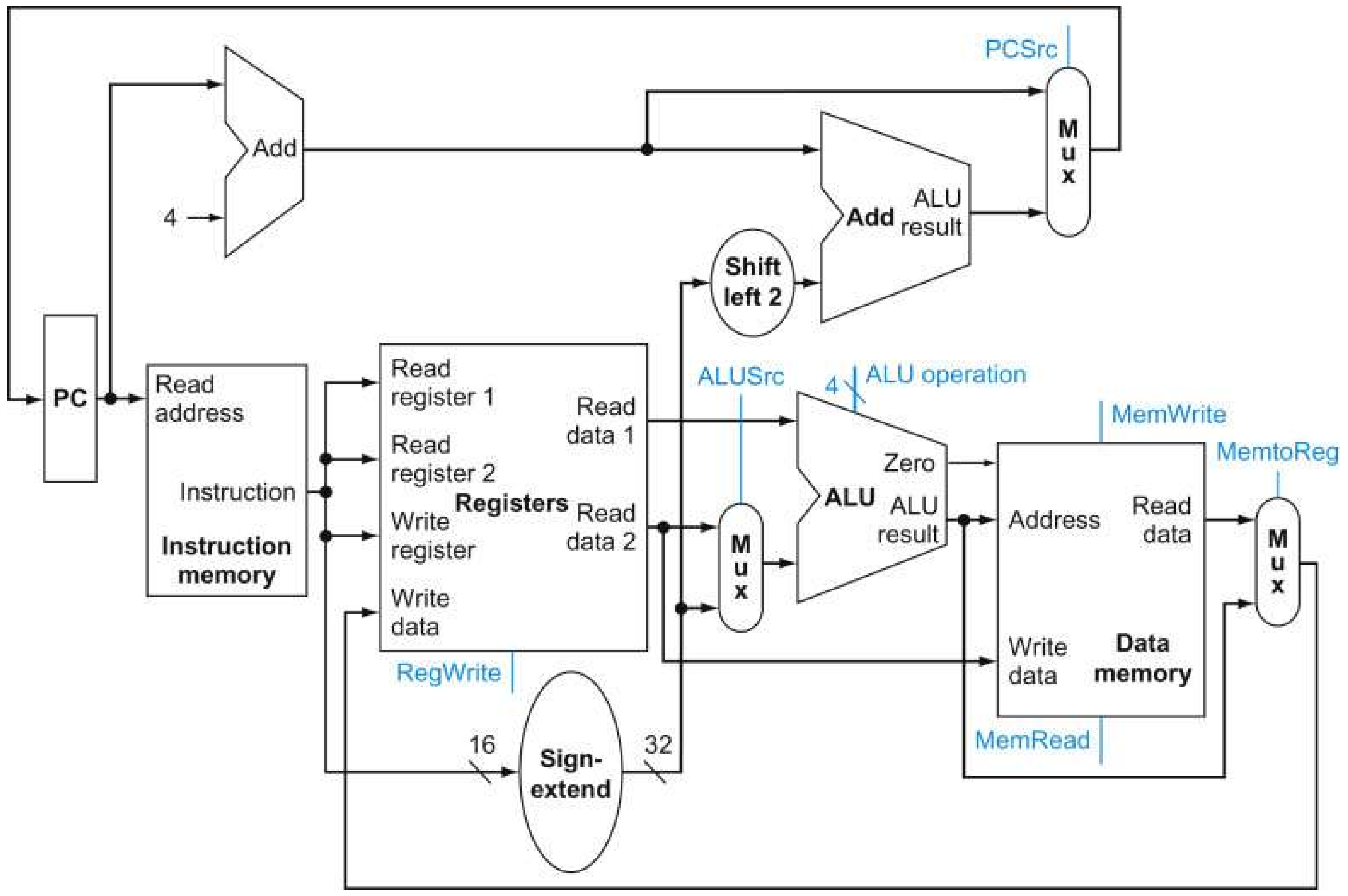
Implementing J-type Instructions

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



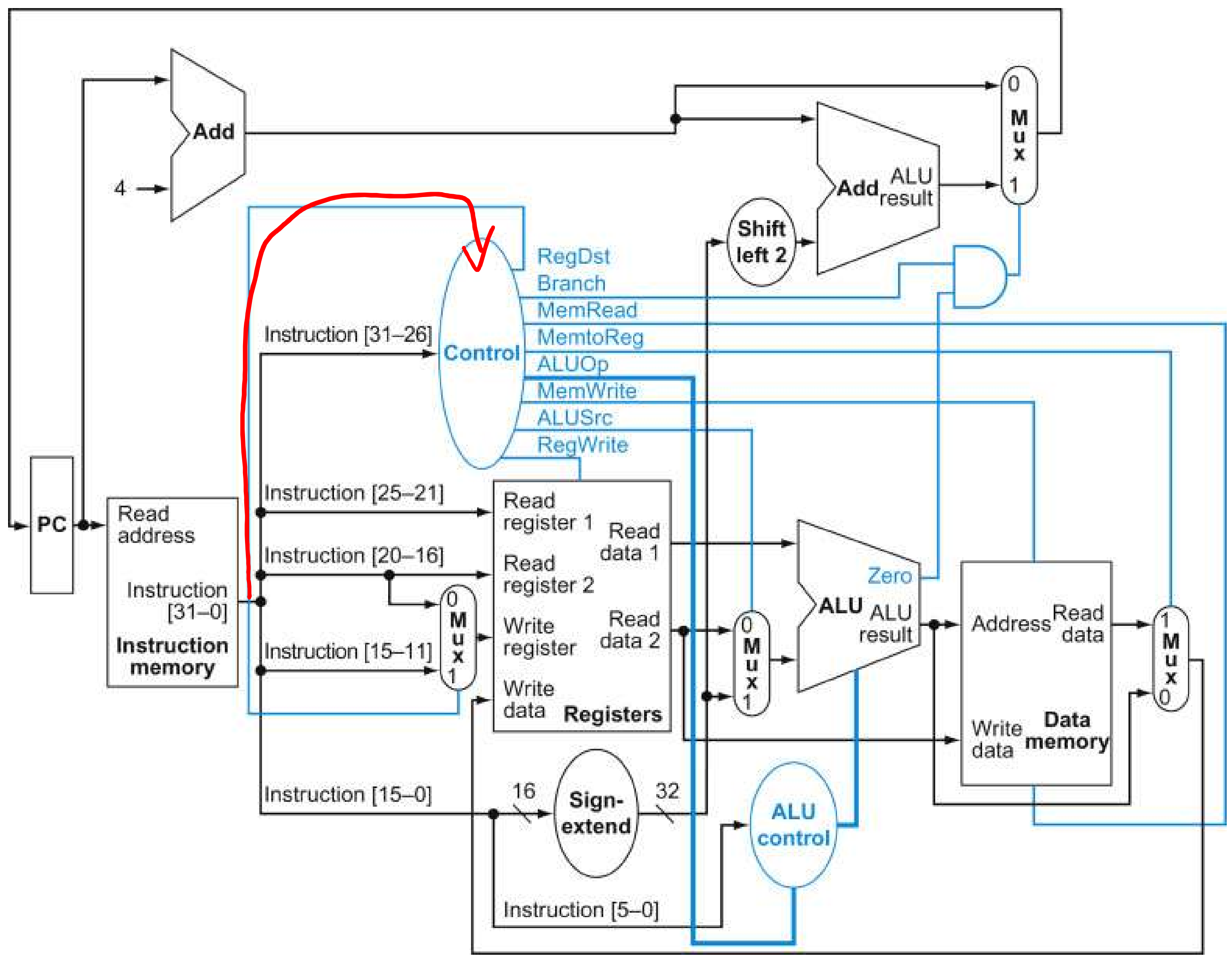
Source: H&P textbook

View from 10,000 Feet



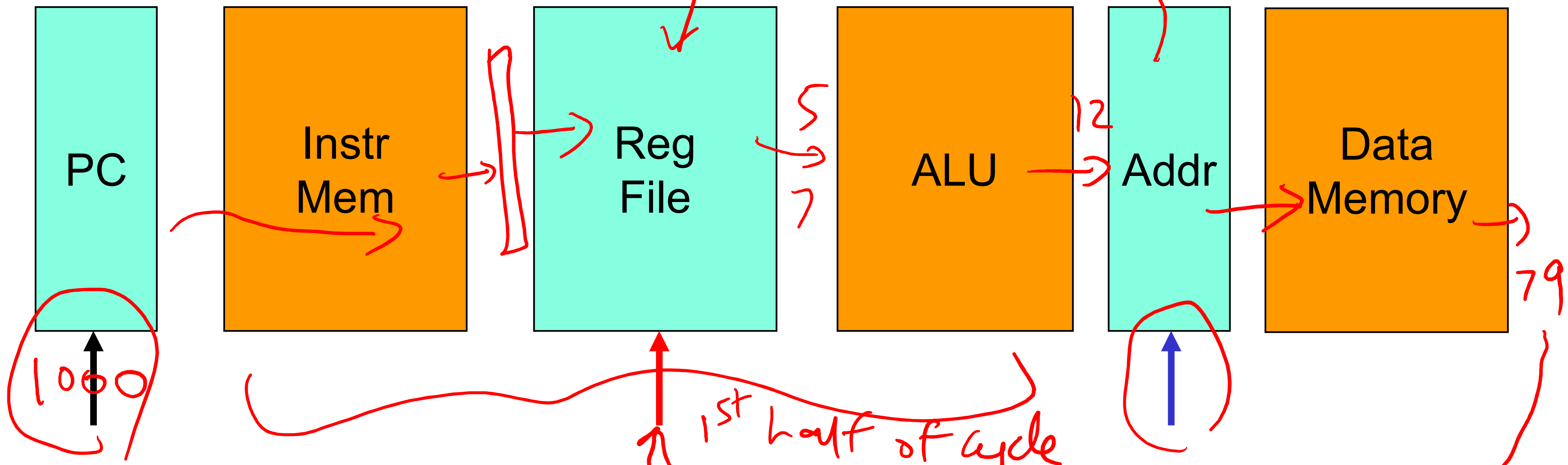
Source: H&P textbook

View from 5,000 Feet



Source: H&P textbook

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 2nd half of the cycle