Lecture 13: Sequential Circuits, FSM

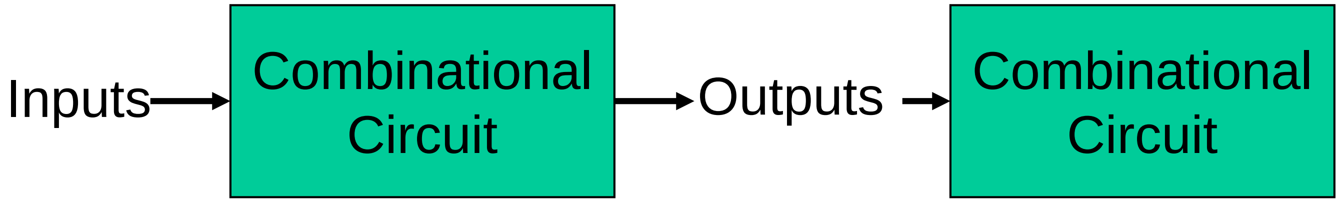
- Today's topics:

  - Sequential circuits
  - Finite state machines

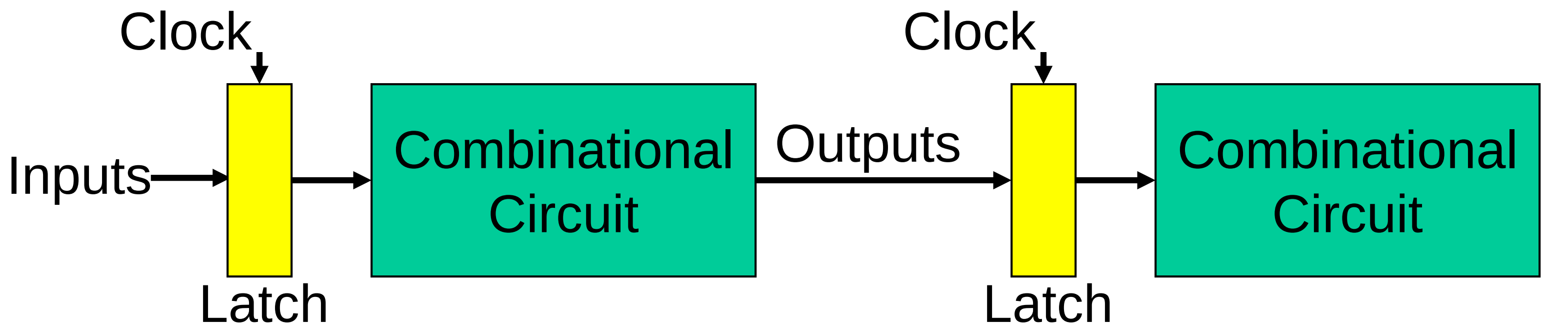
- Reminder: midterm on Tue 2/28
  - will cover Chapters 1-3, App A, B
  - if you understand all slides, assignments, you will ace 90% of the test

Sequential Circuits

- Until now, circuits were combinational – when inputs change, the outputs change after a while (time = logic delay thru circuit)



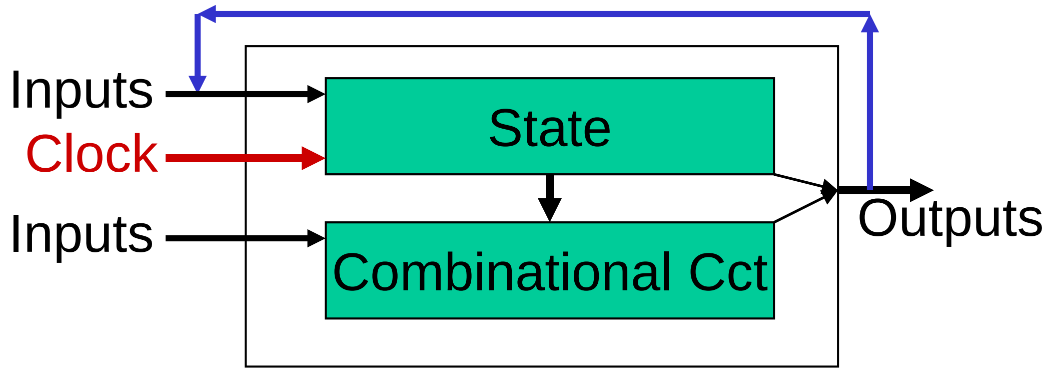
- We want the clock to act like a start and stop signal – a “latch” is a storage device that separates these circuits – it ensures that the inputs to the circuit do not change during a clock cycle



Sequential Circuits

- Sequential circuit: consists of combinational circuit and a storage element

- At the start of the clock cycle, the rising edge causes the “state” storage to store some input values



- This state will not change for an entire cycle (until next rising edge)

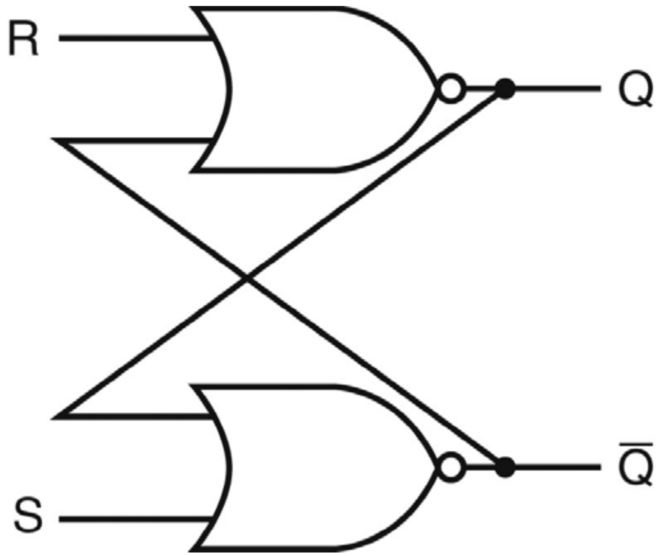
- The combinational circuit has some time to accept the value of “state” and “inputs” and produce “outputs”

- Some of the outputs (for example, the value of next “state”) may feed back (but through the latch so they’re only seen in the next cycle)

Designing a Latch

- An S-R latch: set-reset latch
  - When Set is high, a 1 is stored
  - When Reset is high, a 0 is stored
  - When both are low, the previous state is preserved (hence, known as a storage or memory element)
  - Both are high – this set of inputs is not allowed

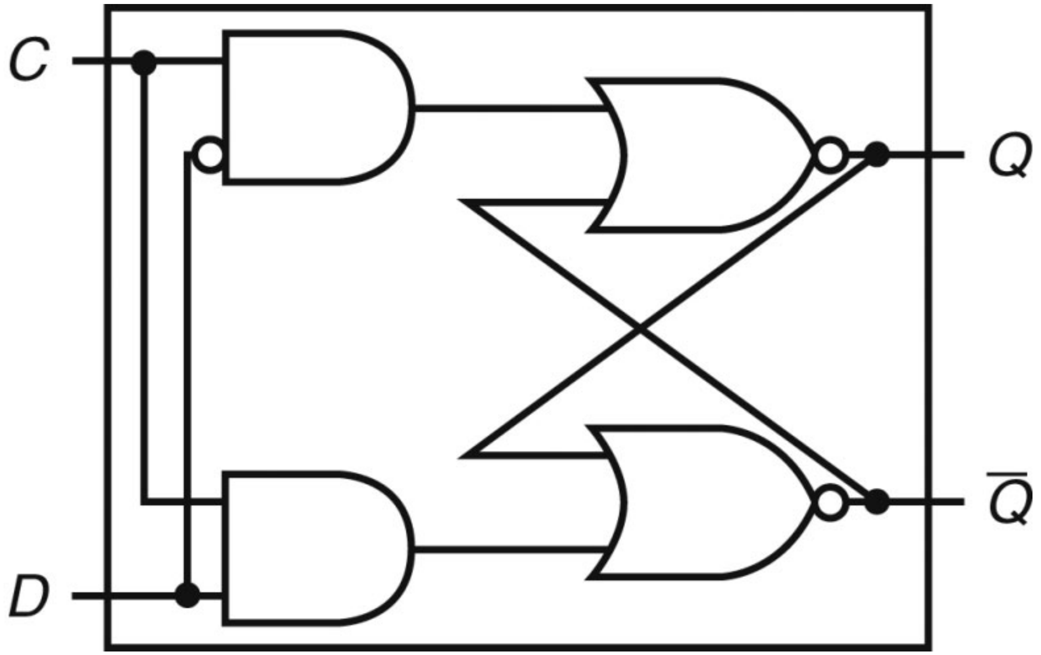
Verify the above behavior!



Source: H&P textbook

D Latch

- Incorporates a clock
- The value of the input D signal (data) is stored only when the clock is high – the previous state is preserved when the clock is low

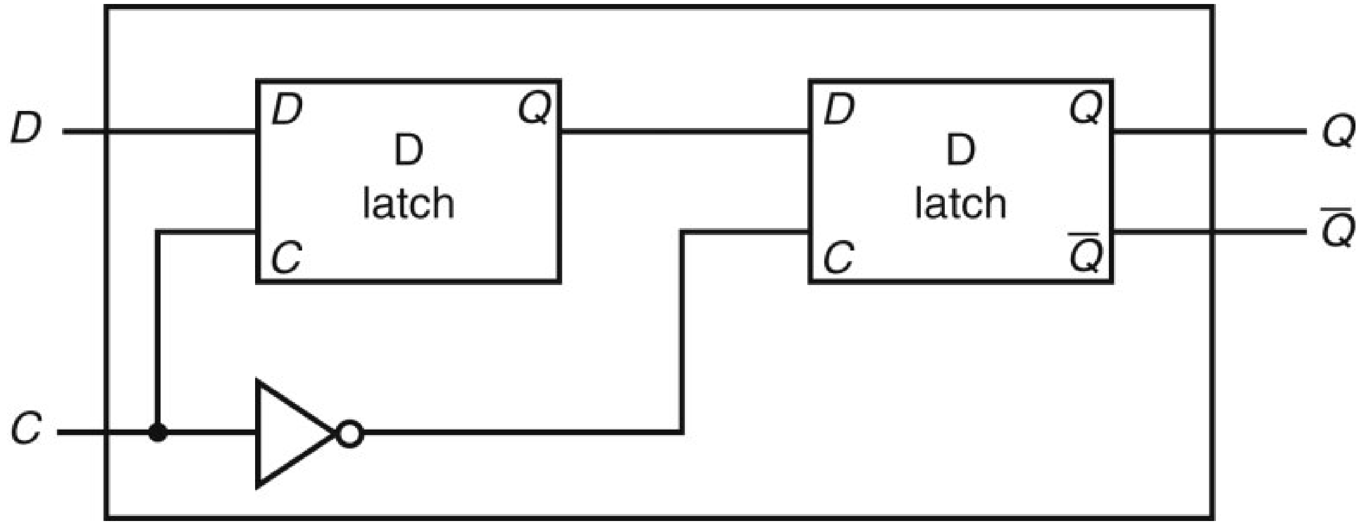


Source: H&P textbook

D Flip Flop

- Terminology:
  Latch: outputs can change any time the clock is high (asserted)
  Flip flop: outputs can change only on a clock edge

- Two D latches in series – ensures that a value is stored only on the falling edge of the clock

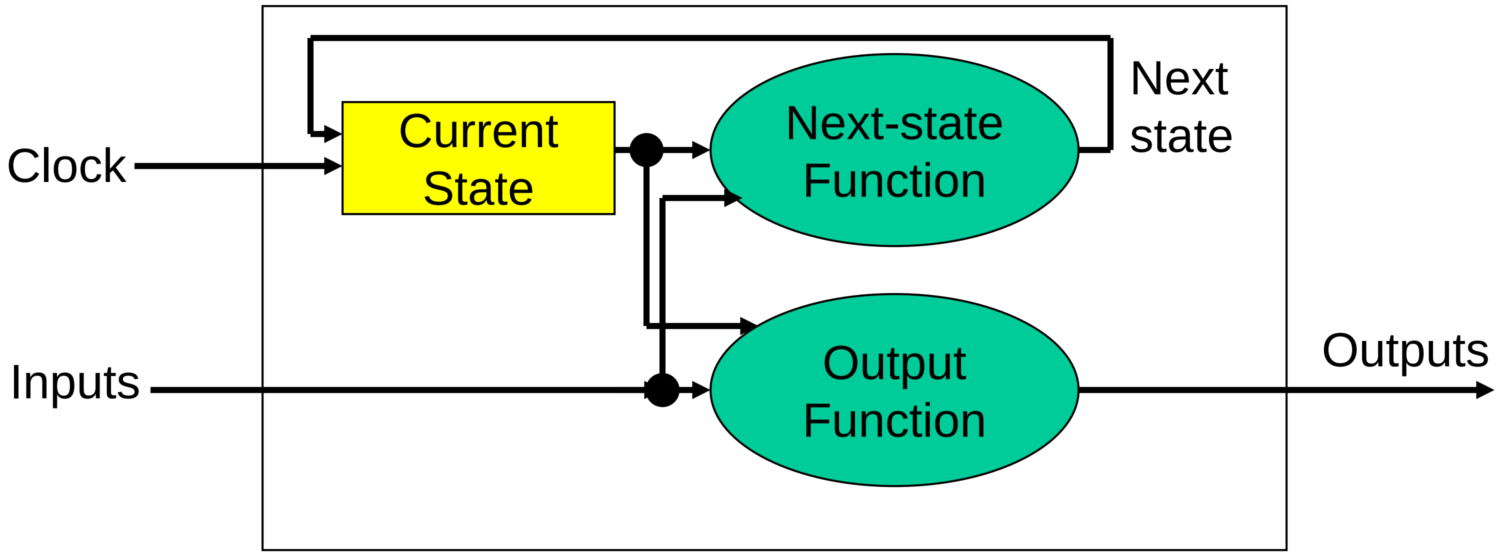


Source: H&P textbook

Finite State Machine

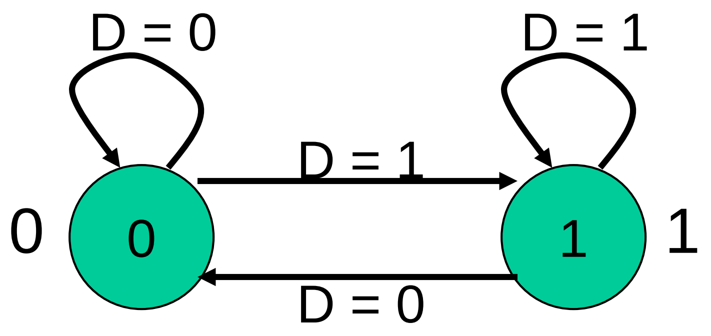
- A sequential circuit is described by a variation of a truth table – a finite state diagram (hence, the circuit is also called a finite state machine)

- 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

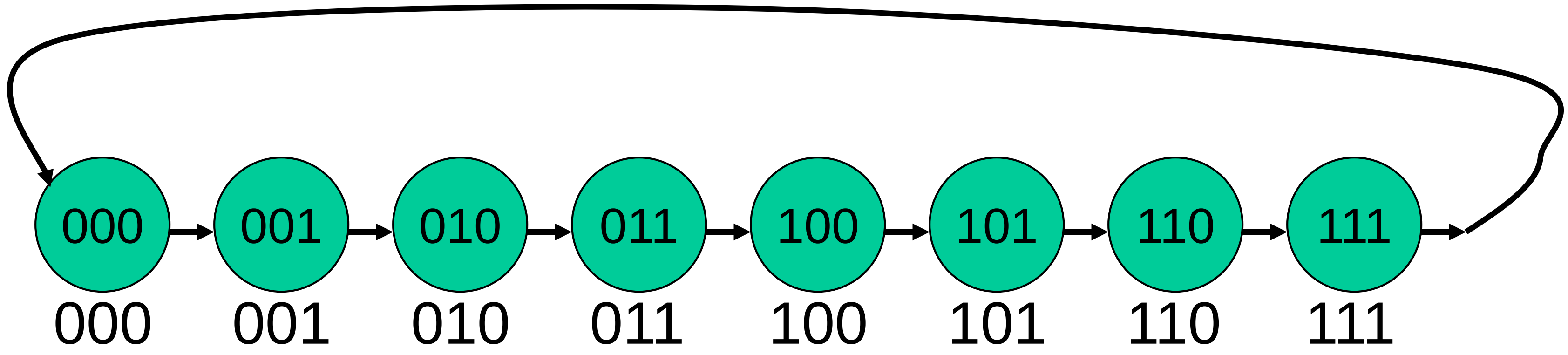
- Consider a circuit that stores a number and increments the value on every clock edge – on reaching the largest value, it starts again from 0

  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

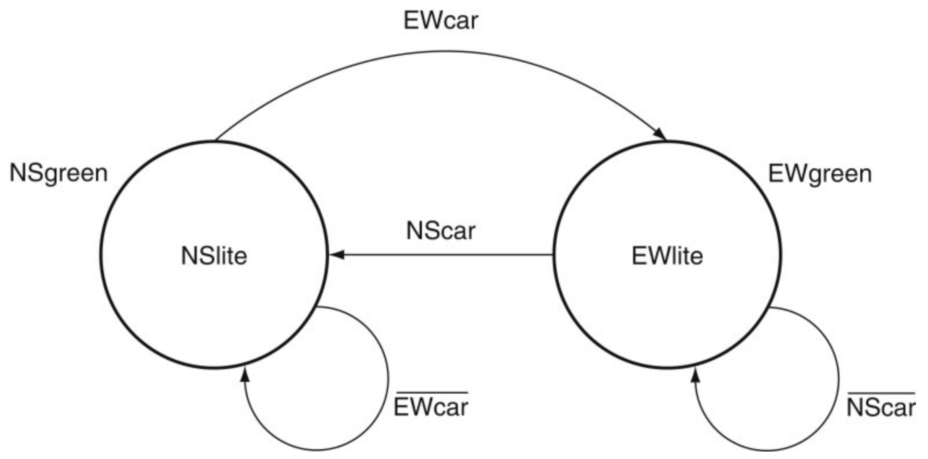
State Transition Table:

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



Source: H&P textbook

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)

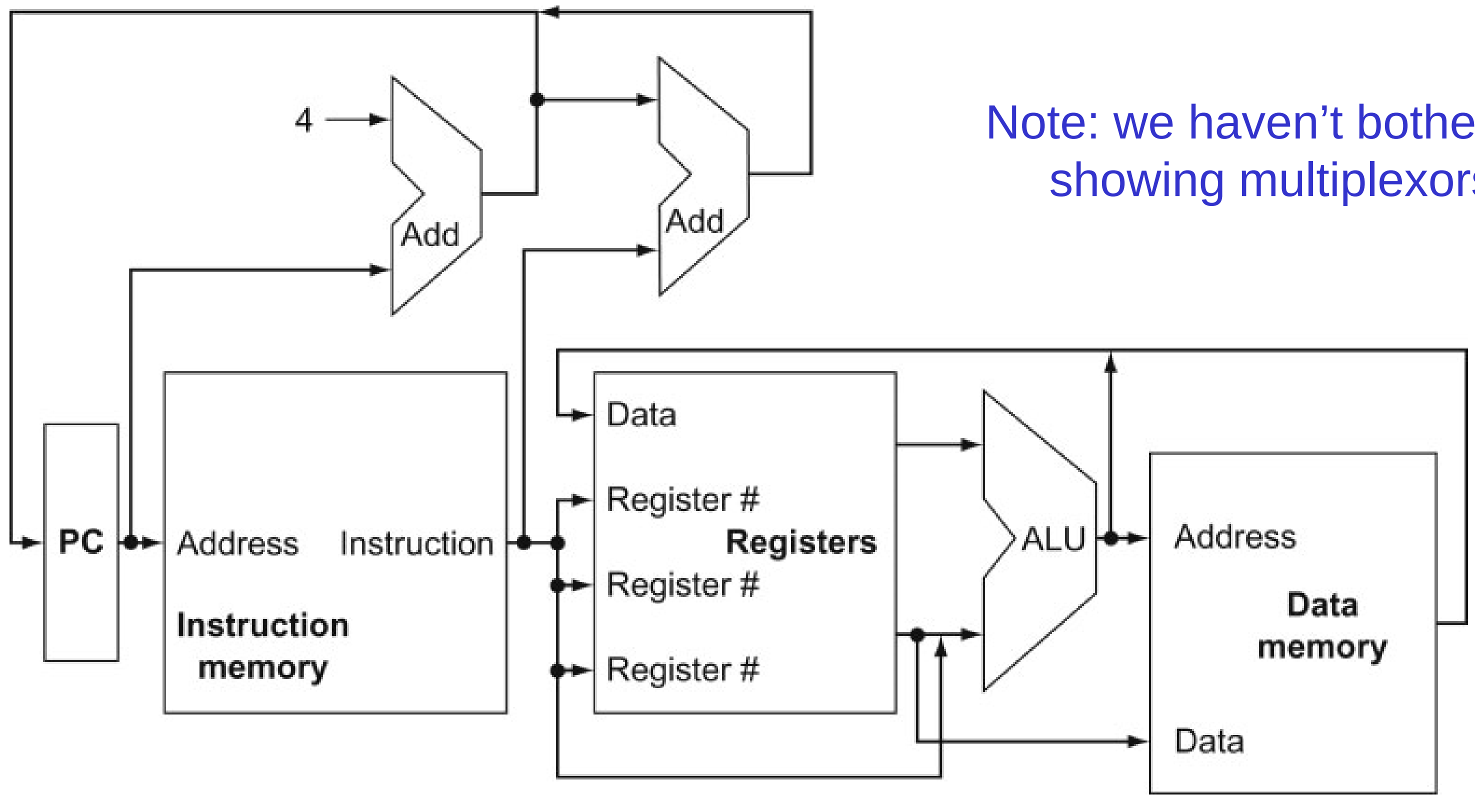
Implementation Overview

- 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

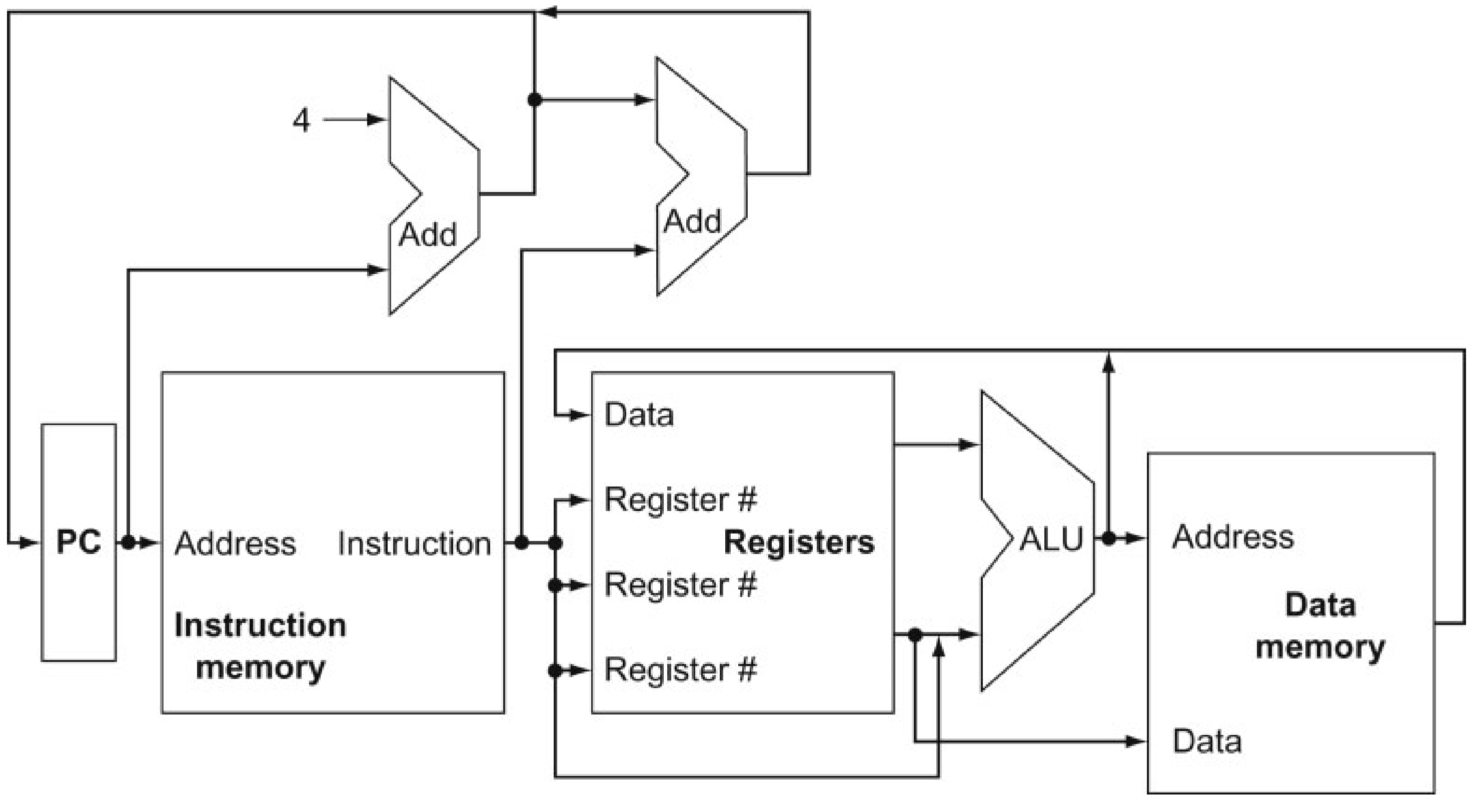


Note: we haven't bothered showing multiplexors

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

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

Title

- Bullet