Lesson 9

Clocks, Memory elements, (Flip-flops, Latches and Registers)

And Processors

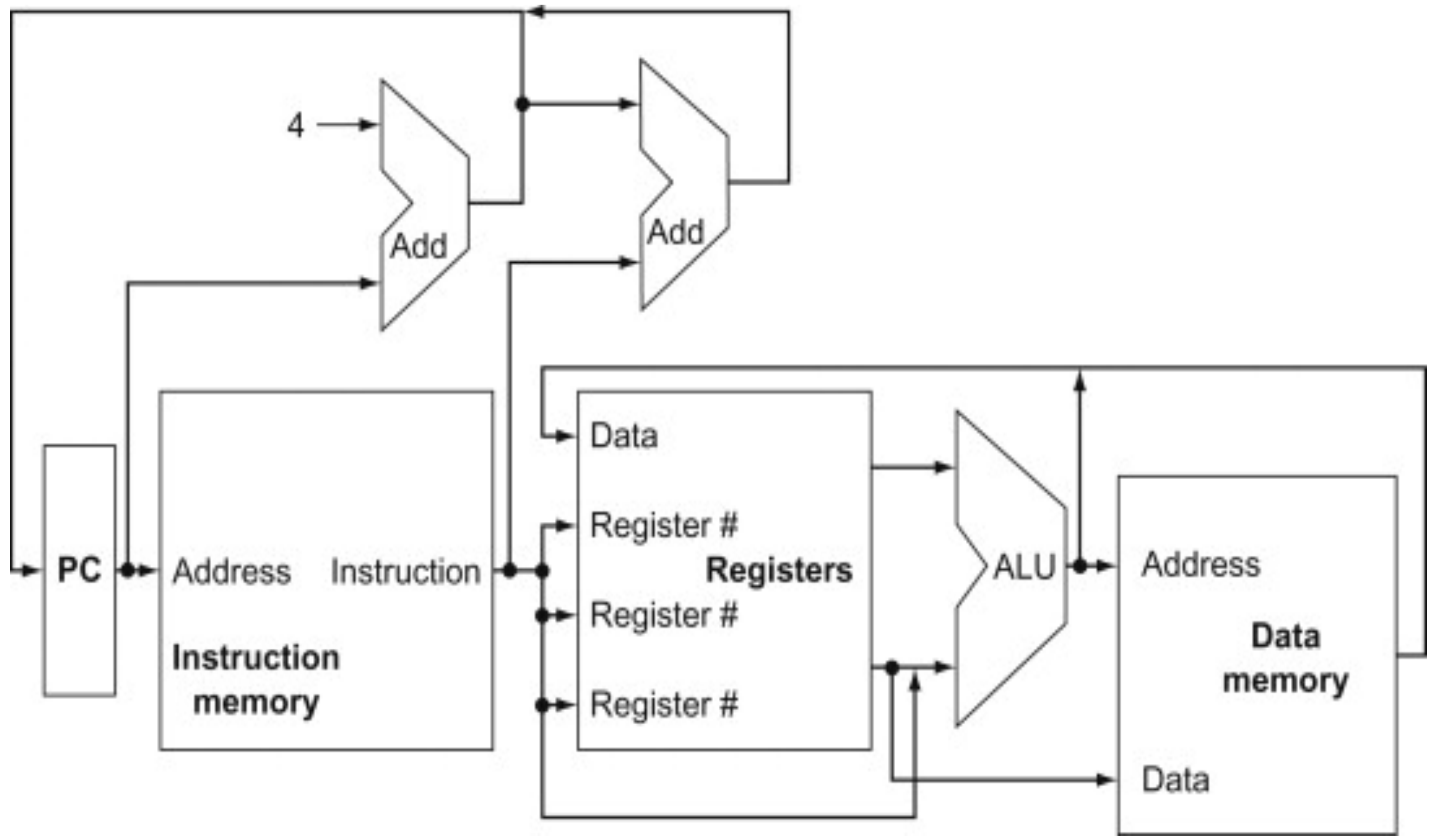
A basic MIPS implementation

- We examine implementation of the following instructions
  - The memory-reference instructions *load word* (lw) and *store word* (sw)
  - The arithmetic-logical instructions add, sub, AND, OR, and slt
  - The instructions *branch equal* (beq) and *jump* (j) (We will review these)

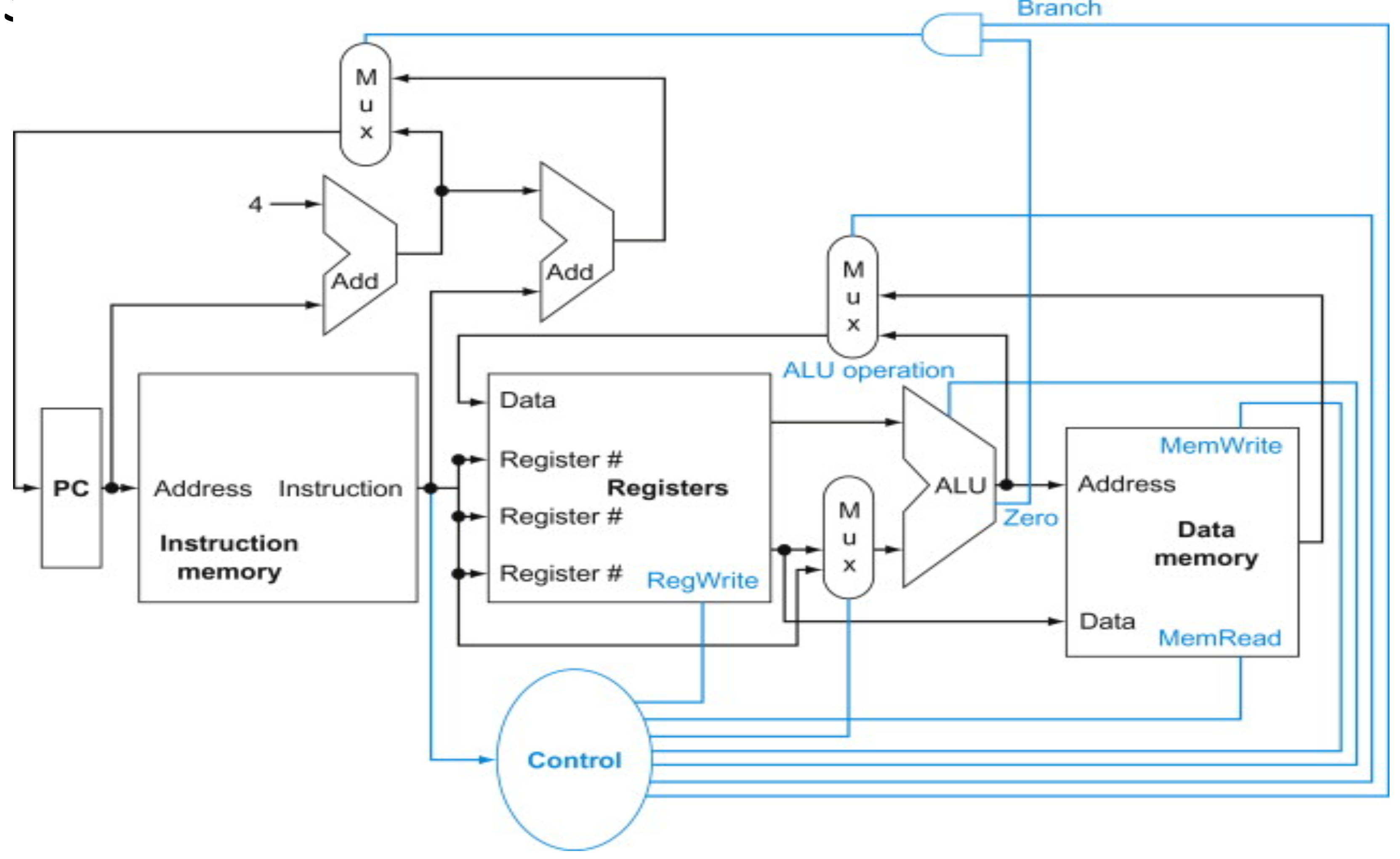
Steps for MIPS instructions

- The following are the steps required to implement MIPS instructions
  1. Send the *program counter* (PC) to the memory that contains the code and fetch the instruction from that memory.
  2. Read one or two registers, using fields of the instruction to select the registers to read. For the load word instruction, we need to read only one register, but most other instructions require reading two registers.
  3. Step 3 Varies:
     - The memory-reference instructions use the ALU for an address calculation, the arithmetic-logical instructions for the operation execution
     - Branches use ALU for comparison
     - Arithmetic-logic instructions use ALU for operation execution
  - The next steps after step 3 differ according to instruction type

An abstract view of the implementation of the MIPS



The basic implementation of the MIPS subset, including the necessary multiplexors and control lines



Logic design conventions

- The datapath elements in the MIPS implementation consist of two different types of logic elements:
    1. Elements that operate on data values
    2. elements that contain state.
- Elements that operate on data values are all *combinational*, which means that their outputs depend only on the current inputs.
- ***Combinational element***: An operational element, such as an AND gate or an ALU.
- *State element is an element that contains some internal storage such as register or memory*

State element

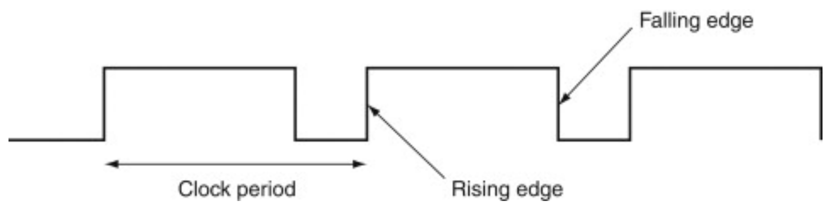
- A state element has at least two inputs and one output.
- The required inputs are the data value to be written into the element and the clock, which determines when the data value is written.
- The output from a state element provides the value that was written in an earlier clock cycle
- The clock is used to determine when the state element should be written; a state element can be read at any time.
- Logic components that contain state are also called *sequential*, because their outputs depend on both their inputs and the contents of the internal state

Clocks (Reading Appendix B – 8.7)

- Clocks are needed in sequential logic to decide when an element that contains state should be updated.
- A clock is a free-running signal with a fixed cycle time; the clock frequency is simply the inverse of the cycle time.
- The clock cycle time or clock period is divided into two portions:
  - when the clock is high
  - when the clock is low.
- we use only *edge-triggered clocking - t*his means that all state changes occur on a clock edge.
- Depending on the technology, it may or may not be the best choice for a *clocking methodology*.
- ***Edge-triggered clocking***: A clocking scheme in which all state changes occur on a clock edge.
- ***Clocking methodology***: The approach used to determine when data is valid and stable relative to the clock

A clock signal oscillates between high and low values

- In an edge-triggered methodology, either the rising edge or the falling edge of the clock is *active* and causes state changes to occur



Clocks continued

- The major constraint in a clocked system, also called a *synchronous system*, is that the signals that are written into state elements must be valid when the active clock edge occurs.
- ***Synchronous system***: A memory system that employs clocks and where data signals are read only when the clock indicates that the signal values are stable.
- ***Register file***: A state element that consists of a set of registers that can be read and written by supplying a register number to be accessed.

Memory elements: Flip-flops, latches, and registers

- The simplest type of memory elements are *unclocked*; that is, they do not have any clock input
- *Flip-flops* and *latches* are the simplest memory elements.
- In both flip-flops and latches, the output is equal to the value of the stored state inside the element.
- In a clocked latch, the state is changed whenever the appropriate inputs change and the clock is asserted, whereas in a flip-flop, the state is changed only on a clock edge.

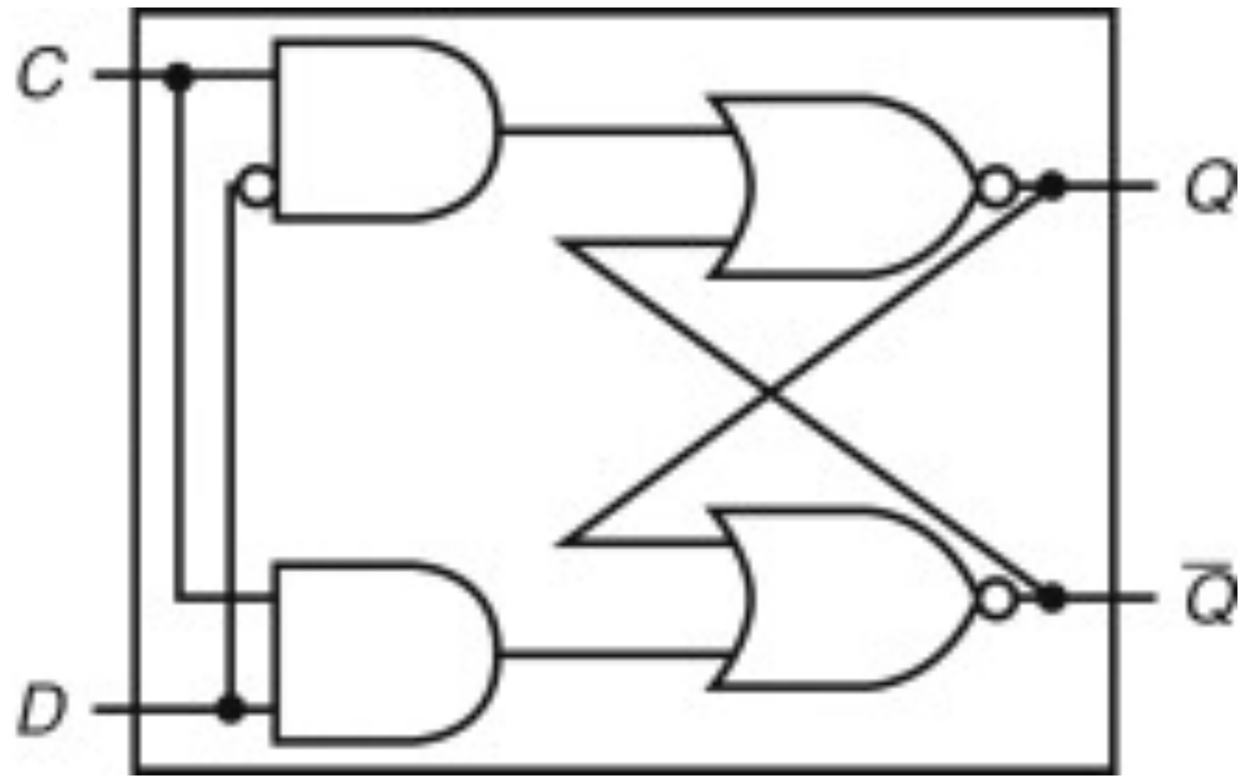
Flip-flop vs Latch

- ***Flip-flop*** is a memory element for which output is equal to the value of the stored state inside the element and for which the internal state is changed only on a clock edge.
- Latch is a memory element in which the output is equal to the value of the stored state inside the element and the state is changed whenever the appropriate inputs change and the clock is asserted.
- For computer applications, the function of both flip-flops and latches is to store a signal.

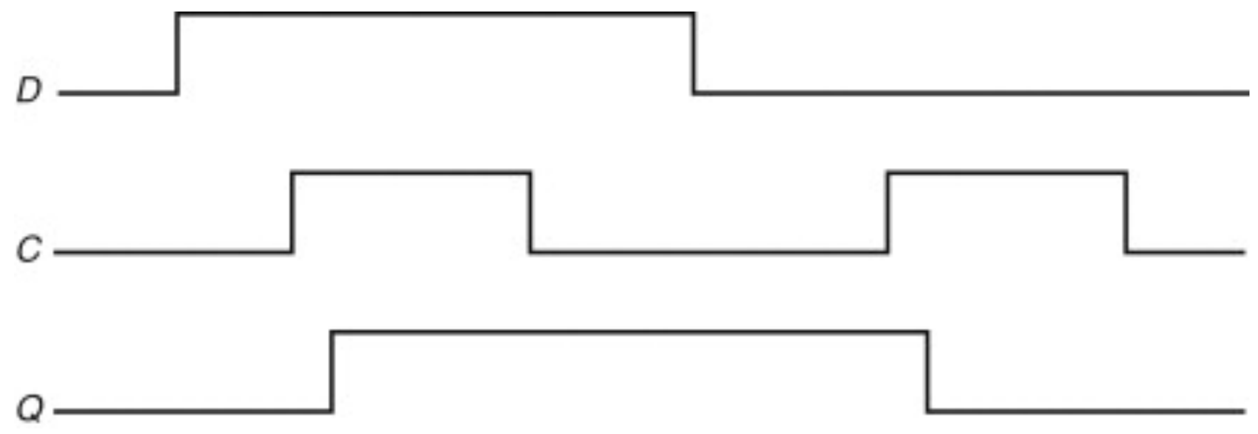
Flip-flop vs Latch

- A D latch or *D flip-flop* stores the value of its data input signal in the internal memory.
- A D latch has two inputs and two outputs.
- The inputs are the data value to be stored (called *D*) and a clock signal (called *C*) that indicates when the latch should read the value on the *D* input and store it.
- ***D flip-flop***: A flip-flop with on data input that stores the value of that input signal in the internal memory when the clock edge occurs.

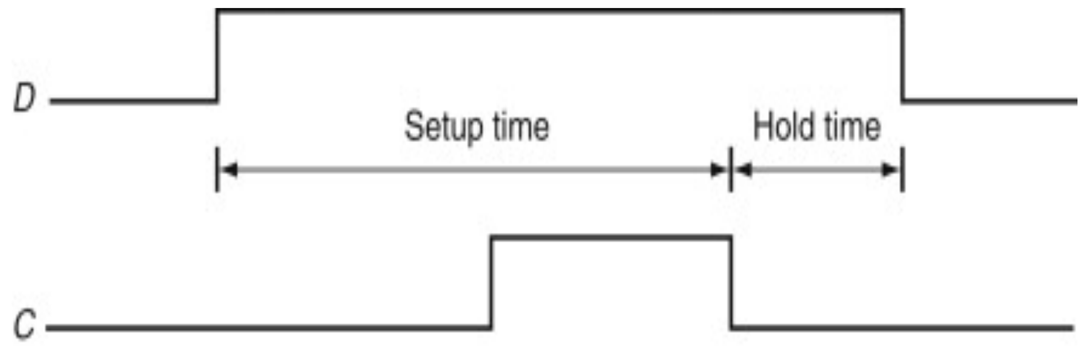
A D latch implemented with NOR gates



Operation of a D latch



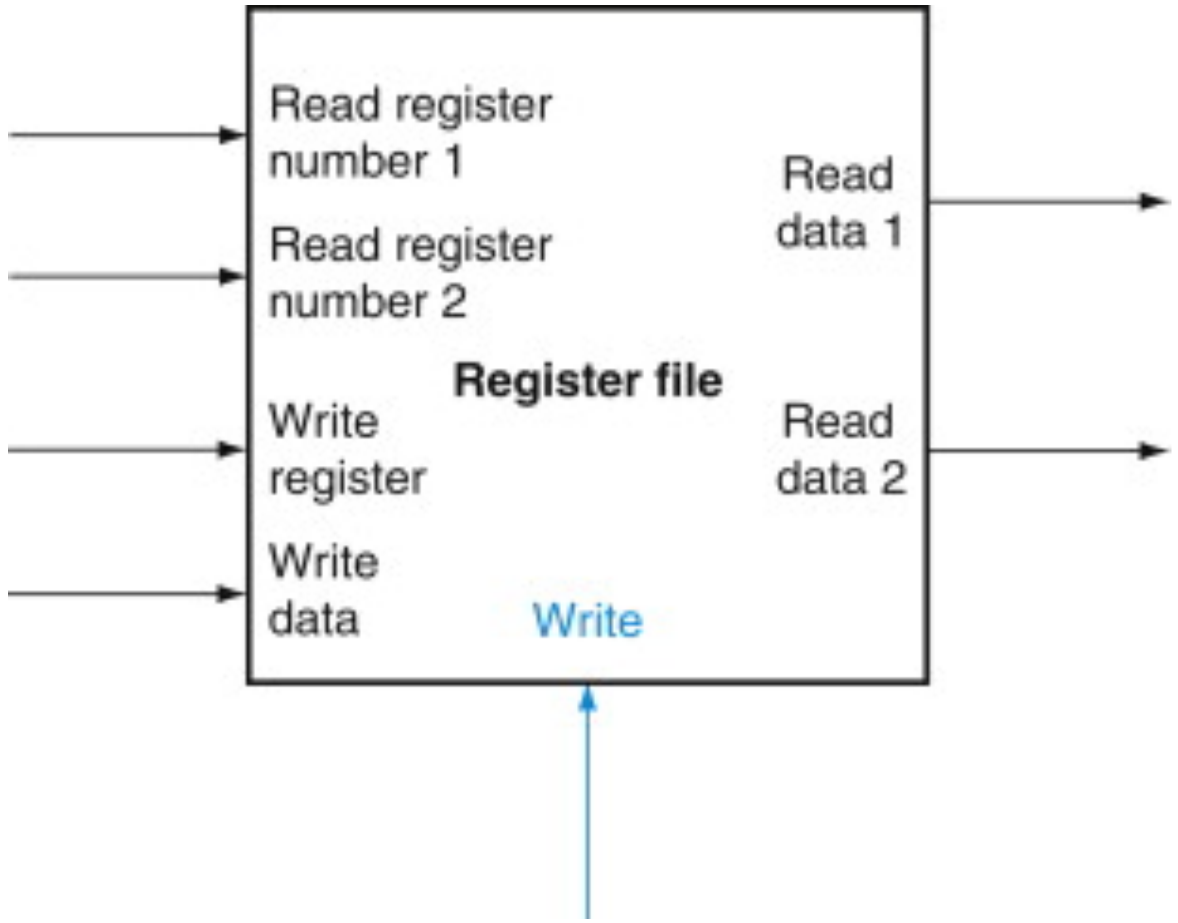
- ***Setup time***: The minimum time that the input to a memory device must be valid before the clock edge.
- ***Hold time***: The minimum time during which the input must be valid after the clock edge.



Register files

- A register file consists of a set of registers that can be read and written by supplying a register number to be accessed.
- A register file can be implemented with a decoder for each read or write port and an array of registers built from D flip-flops

A register file with two read ports and one write port has five inputs and two outputs



Reading

- Hennessy and Patterson Appendix B (8.7 and 8.8) and Chapter 4.1, 4.2 and 4.3