Memory; Sequential Logic Circuits

CS 350: Computer Organization & Assembler Language Programming

A. Why?

- Memory lets us access and update state values.
- Sequential logic circuits combine combinatorial circuits, memory, and a clock to perform calculations that take inputs, use and update the state, and produce outputs.

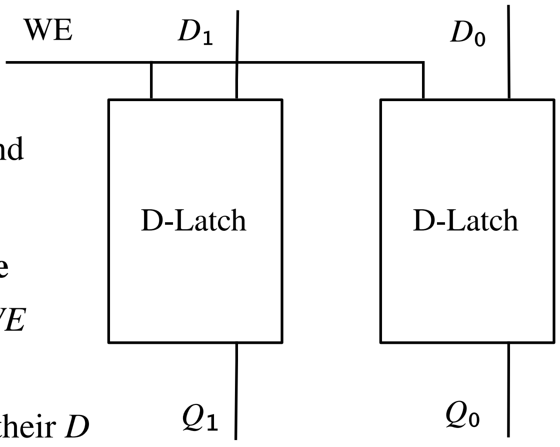
B. Outcomes

After this lecture, you should

- Know how memory combines address decoders and data storage/update.
- What sequential logic circuits, clocks, and flip-flops do and why we have them.

C. Registers Hold Bitstrings

- An ***n*-bit wide register** is a circuit that can store an *n*-bit bitstring.
- We can build an *n*-bit register by combining *n* *D*-latches. (Shown to the right is a 2-bit register.)



- The register has *n* input data bits *D*[*n*–1:0] and *n* output data bits *Q*[*n*–1:0].
- The register also has one "control" bit, **Write Enable** (*WE*), shared by all the latches. If *WE* = 1, then the latches update their states (and outputs). If *WE* = 0, then the latches ignore their *D* inputs and continue to output the same *Q* values.

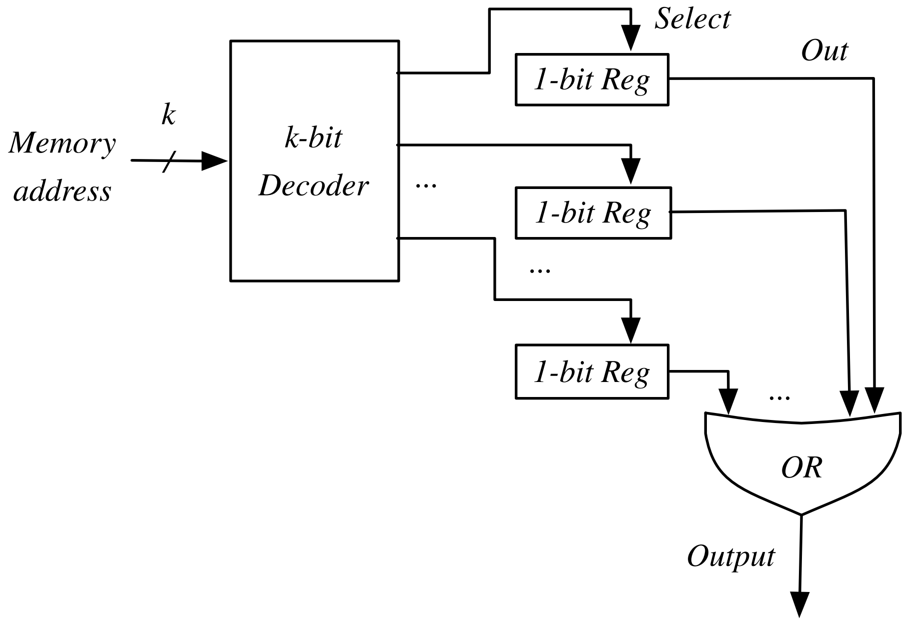
D. Memory Bank

- We can combine multiple registers to produce a memory bank. We give each register an unsigned bitstring **address** (i.e., a name) to identify it.

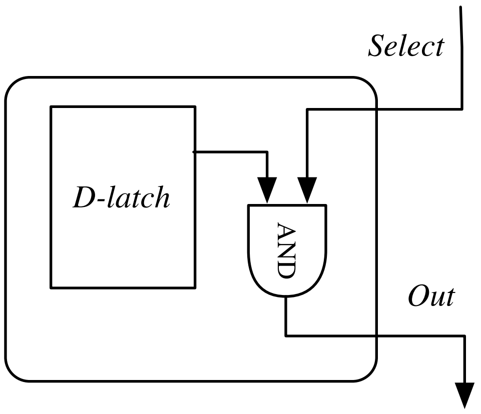
- **Definition**: The **address size** or **address width** is the number of bits in an address. The **address space** is the set of legal addresses; it has size $\leq 2^k$ where k is the address width. "***k*-bit addresses**" means "**has address of width *k***".
    - Address size corresponds to the number of digits in a street address for the U.S. Mail. ("10 West 31st Street" requires $\geq$ 2 decimal digits of address size.)
- **Definition**: The **addressability** of a memory bank is the amount of data stored at each address. If we have m-bit addressability and k-bit addresses, then the memory bank contains $\leq m \times 2^k$ bits.
    - Addressability corresponds to the amount of data you can put in the mailbox for a given street address.
- Computers generally have one of two kinds of addressability.
    - **Byte-addressability** means it has 8-bit addressability.
    - **Word-addressability** means its addressability equals its **word size** (the amount of memory it takes to store an integer, typically).
- **Note 1**: For memory banks in general, the address size and addressability are **unrelated**: You could have a very large set of small mailboxes (i.e., k is large and m is small) or a small set of very large mailboxes (small k, large m).
    - **Examples**: A calculator with one memory location has $k = 0$ and $m =$ some number of decimal digits. An array of 1024 bits has $k = 10$ and $m = 1$.
- **Note 2**: For computer memories with word addressability, we often have $k \leq m$, which means you can store an entire memory address at a memory address.
    - The LC-3 computer has 16-bit addresses and 16-bit addressability.
    - The SDC computer has 4-decimal digit addresses and 2-decimal digit addressability.
- **Alignment**: If a memory bank is byte-addressable, the addresses of words are often restricted to be multiples of the addressability. E.g., if a word is 4 bytes wide, then the "**aligned**" word addresses are 0, 4, 8, 12, etc. (Such addresses are **word-aligned**.) Trying to access an **unaligned** word can cause an error or a slowdown in execution (depends on the hardware design).

E. Reading a Memory Bank

- With the goal of understanding how a read/write memory bank works, let's start by looking at reading from a memory with *k*-bit addresses and 1-bit addressability.
- The *k*-bit memory address feeds into a decoder with 2^k one-bit outputs. The output specified by the value of the address will be 1; the others will be 0.

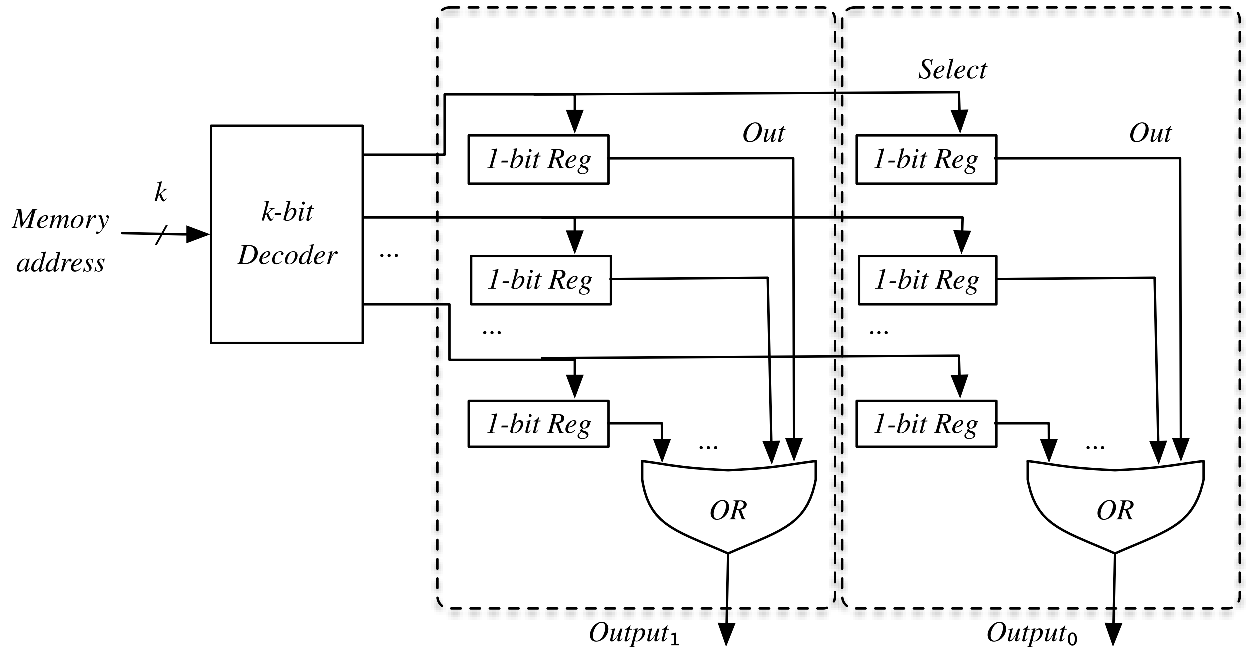


- A 1-bit register is a *D*-latch whose output is ANDed with a *Select* line: If the select line is 1, the register outputs its stored value, otherwise it outputs 0. For the select line, we use the decoder output line for this address.
- The outputs of the 2^k registers get ORed together to form the output of our memory bank: The output matches the value of the register selected by the address.
- To store multiple bits of data at each address, we use a register for each bit. (We only need one decoder, however.) The diagram below supports 2 bits of data per address. The dashed rectangles indicate the circuitry being repeated for each "column" of output. Note the registers



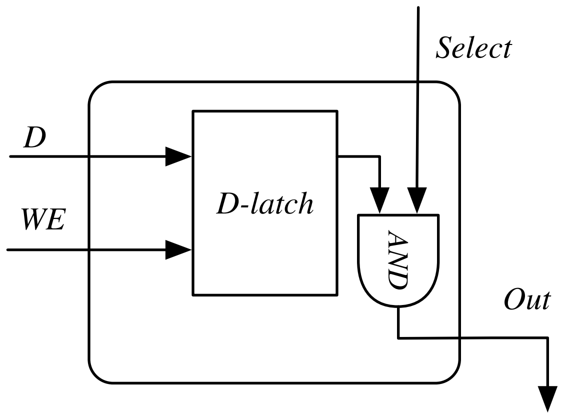
Reading a 1-bit Register

for each address share the same select line.



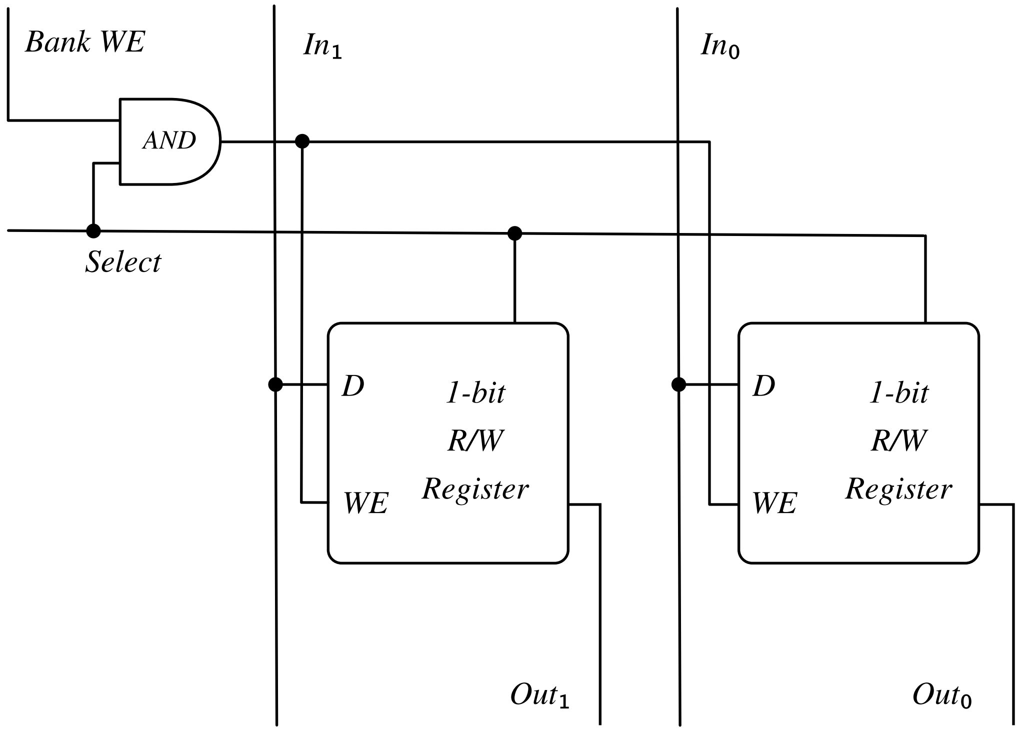
F. Writing To Memory

- With read-write memory, the 1-bit registers gain two inputs: a write enable line and an input data line. The latch state is set to the input if the write enable line is on, otherwise the input is ignored.
- As before, the output of the latch is only sent out if the select line is on, otherwise a 0 is output.



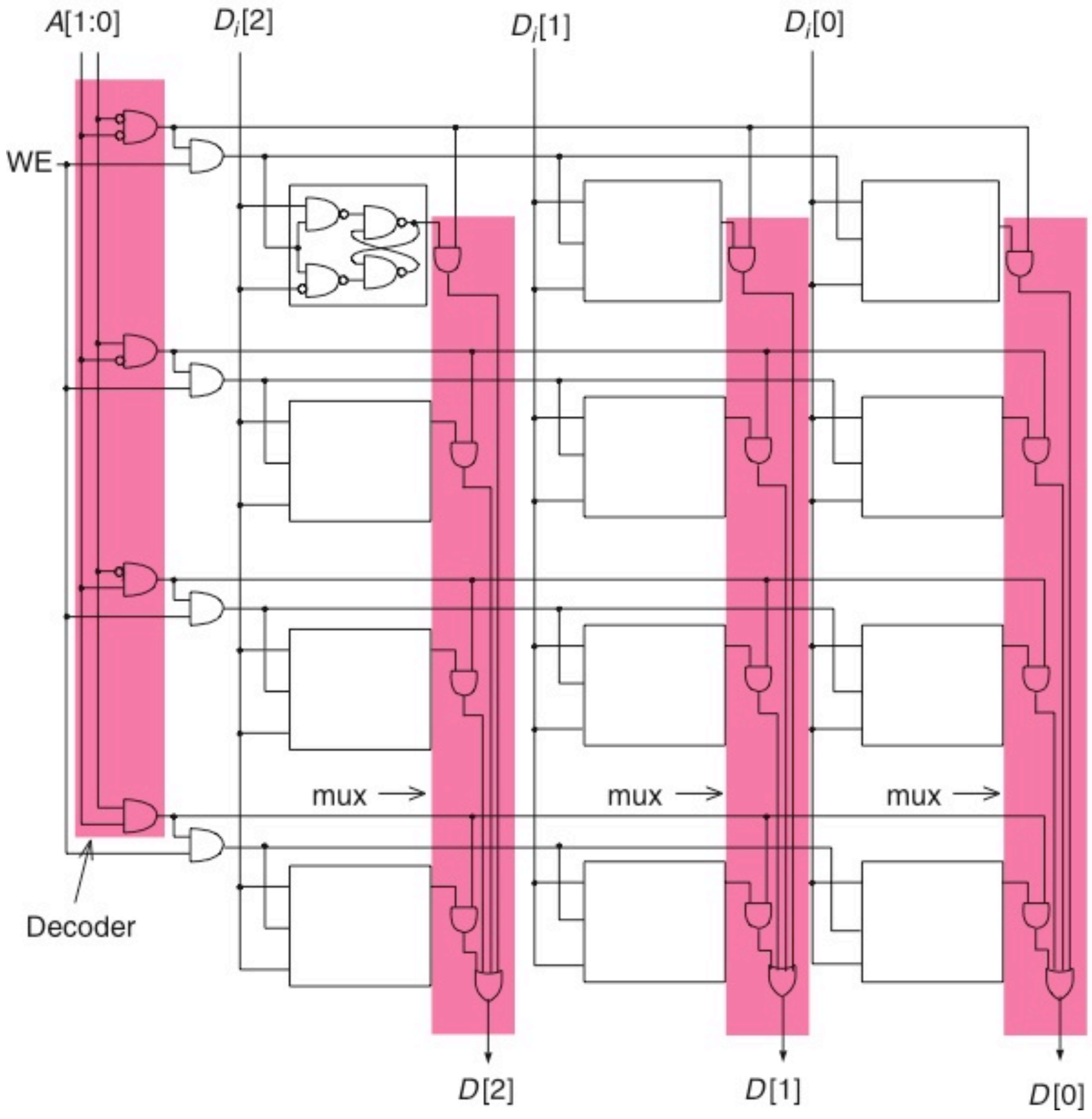
A 1-bit Read/Write Register

- If we concentrate on all the bits stored at one address, we only write data to these bits if this address is the one selected and the memory bank is supposed to write its input data into the bank.



The 2 bits At an Address

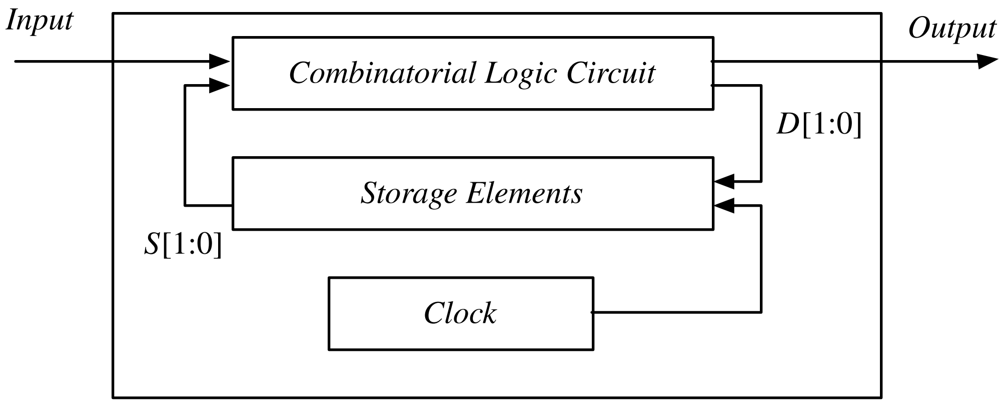
- For a fuller example, Patt & Patel's Figure 3.21 shows a memory bank with 2-bit address size and 3-bit addressability.



G. Sequential Logic Circuits

- A **Sequential Logic Circuit** combines combinatorial circuits and storage elements. The combinatorial circuit calculations take input values and memory values and produce output values and updated memory values.

- It turns out that we also need a way to control when the calculations occur relative to when memory gets updated. Consider the following block diagram for a sequential logic circuit; ignore the clock for now. There are two storage elements, with inputs $D[1{:}0]$ and outputs $S[1{:}0]$.



- If we think of calculating D and setting S as four operations (calculate D_1 and D_0 and set S_1 and S_0), then different orders of operations might produce different results. As a concrete example, say we have input X and we calculate $D_1 = \overline{S}_1 * S_0$ and $D_0 = S_1 + X * S_0$. Suppose $X = 0$ and $S = 10$ (i.e., $S_1 = 1$ and $S_0 = 0$).
  - If we calculate D_1 and D_0 before setting S_1 and S_0, then S becomes 01.

    $D_1 = \overline{S}_1 * S_0 = 1 * 0 = 0$
    $D_0 = S_1 + X * S_0 = 1 + 0 * 0 = 1$
    $S_1 \leftarrow D_1 = 0$
    $S_0 \leftarrow D_0 = 1$

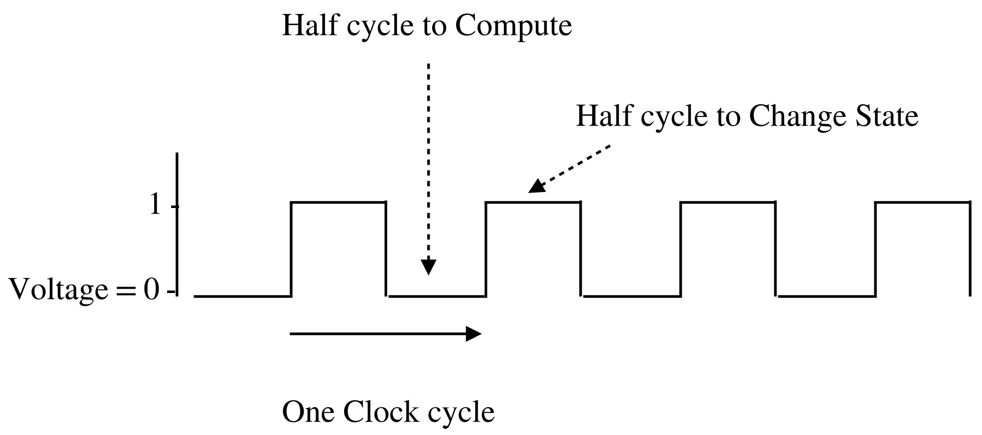
  - But if we calculate D_1 and set S_1 before calculating D_0 and setting S_0, then S becomes 00.

    $D_1 = \overline{S}_1 * S_0 = 1 * 0 = 0$
    $S_1 \leftarrow D_1 = 0$
    $D_0 = S_1 + X * S_0 = 0 + 0 * 0 = 0$
    $S_0 \leftarrow D_0 = 0$

  - This situation is bad enough, but if the times to calculate D_1 and D_0 can vary, then we could have even more problems.

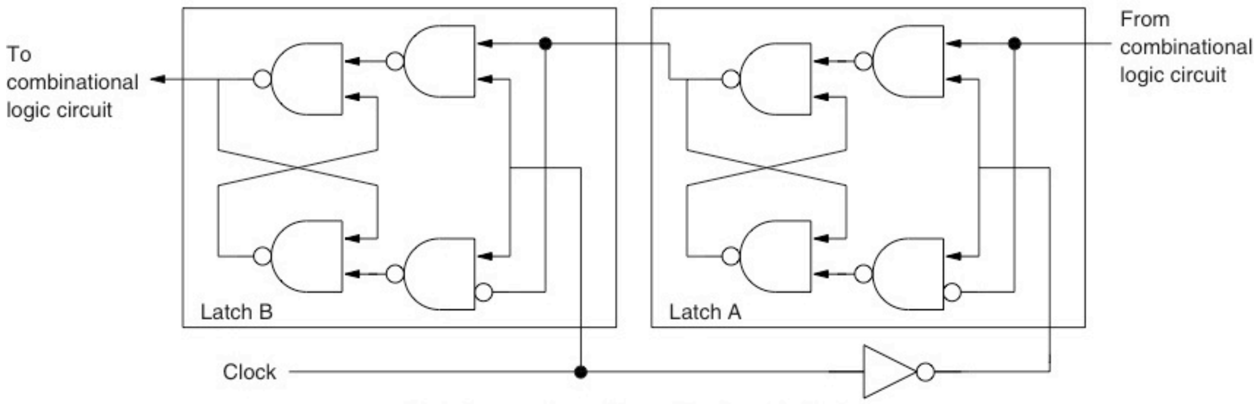
H. Clocks and Flip-Flops

- To the solve the problem caused by allowing calculations and memory updates to intermingle, we will alternate them. To do this, we'll use a **Clock**, a device that cycles its output (the **Clock Signal**) from 0 to 1 and back, at a regular speed.
- During one half of the cycle, we'll simultaneously read memory, get the input values, and do calculations, but we won't allow memory to be updated. Memory updates are only allowed during the other half cycle, and while memory is being updated, we'll ignore the inputs and calculation results.



- To store a memory bit, we can use a **Master-Slave Flip-Flop**. It uses the clock signal to alternate between two modes. In the first mode, it reads and stores one bit of input without changing its output. In the second mode, it changes its output to match its internal state while ignoring its input.
- A master-slave flip-flop has two *D*-latches and a clock connection. The right-hand *D*-latch output is the left-hand *D*-latch input.
- When the clock signal is 0
  - The combinatorial circuit does its calculation based on the value in the left-hand latch and sends its output to the right-hand latch
  - The right-hand latch sets its state to the value of the combinatorial circuit.
  - The left-hand latch sends its value as the flip-flop output but ignores the value of the right-hand latch (as it possibly changes).
- When the clock signal is 1

- The left-hand latch reads the value in the right-hand latch.
- The right-hand latch ignores the value being produced by the combinatorial circuit.
- The combinatorial circuit does its calculation based on the changing value of the left-hand latch.


To combinational logic circuit
From combinational logic circuit
Latch B
Latch A
Clock
(c) A storage element (a master-slave flip-flop)

- For an *m*-bit register, we have *m* flip-flops all connected to the same clock. Either *m* bits of data are being read into the register (without changing the register's output). Or, *m* bits of data are being output from the register (while ignoring the data entering the register).
- In Patt & Patel's Figure 3.43, we have two flip-flops connected to the clock, so D_0 and D_1 get calculated, then S_0 and S_1 get set, then D_0 and D_1 get calculated, and so on.

Memory; Sequential Logic Circuits

CS 350: Computer Organization & Assembler Language Programming

A. Why?

- Memory lets us access and update state values.
- Sequential logic circuits combine combinatorial circuits, memory, and a clock to perform calculations that take inputs, use and update the state, and produce outputs.

B. Outcomes

After this activity, you should be able to:

- Show how memory combines address decoders, multiplexers, and data storage/update.

C. Questions

1. In a memory bank, what kind of circuit is used to map a memory address to the enable line for that memory address?
2. In a memory bank, how are enable lines shared? Write enable lines?
3. What is the difference between byte addressability and word addressability?
4. What is the relationship between address size and addressability for computer memory banks? For memory banks in general?
5. Say we have a machine with k-bit addresses and m-byte addressability. (a) What is the address space and how large is it? (b) How many bytes of memory can we have in total? (c) What, if any, is the relationship between k and m?
6. Say a byte-addressable machine has 32-bit memory addresses. How many gigabytes of memory can we access? What if memory addresses are 64-bits? Hint: The standard prefixes are kilo-, mega-, giga-, tera-, peta-, exa-, zetta-, and yotta-.
7. How are combinatorial and sequential logic circuits different? Which uses state information?

8. Why do sequential logic circuits synchronize calculations and memory updates? How are clocks used to do synchronization? How fast can the clock be relative to the calculation part of a sequential logic circuit?
9. What comprises a master-slave flip-flop and how does it work?