CSE 390 B Spring 2021

Cornell Notes Review & Memory

Building memory and revisiting cornell note-taking

Significant material adapted from www.nand2tetris.org. © Noam Nisan and Shimon Schocken.

Agenda

- Cornell Note-Taking Review
- Building a Bit
  - DFF Example Review
  - Building a bit exercise
- Building Memory
  - Reading Review and Q&A
  - Building memory: Bit to RAM
  - Program Counter
- Project 3 Overview

Agenda

- **Cornell Note-Taking Review**
- Building a Bit
  - DFF Example Review
  - Building a bit exercise
- Building Memory
  - Reading Review and Q&A
  - Building memory: Bit to RAM
  - Program Counter
- Project 3 Overview

Breakouts: Cornell Note-Taking Discussion

In your small groups, compare and contrast your Cornell Notes from Tuesday's lecture.

- What are some of the key points you wrote in your summary?
- What were some of the questions you came up with?
- What are you still left feeling confused/uncertain about after Tuesday's lecture?

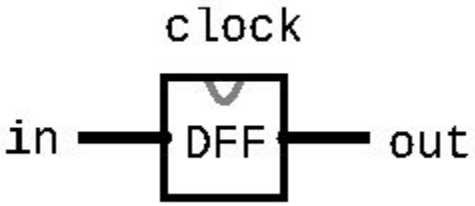
Agenda

- Cornell Note-Taking Review
- **Building a Bit**
  - **DFF Example Review**
  - Building a bit exercise
- Building Memory
  - Reading Review and Q&A
  - Building memory: Bit to RAM
  - Program Counter
- Project 3 Overview

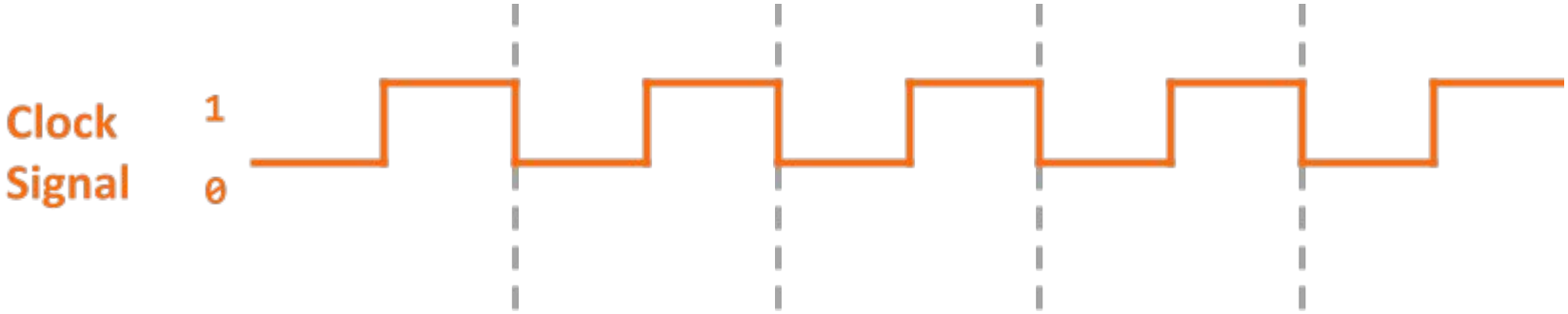
Review: Sequential Chips & DFF

- Sequential Chips
  - A category of chips that utilize the clock signal, in addition to any combinational logic
  - Maintains state
  - Constructed from DFF and combinational logic

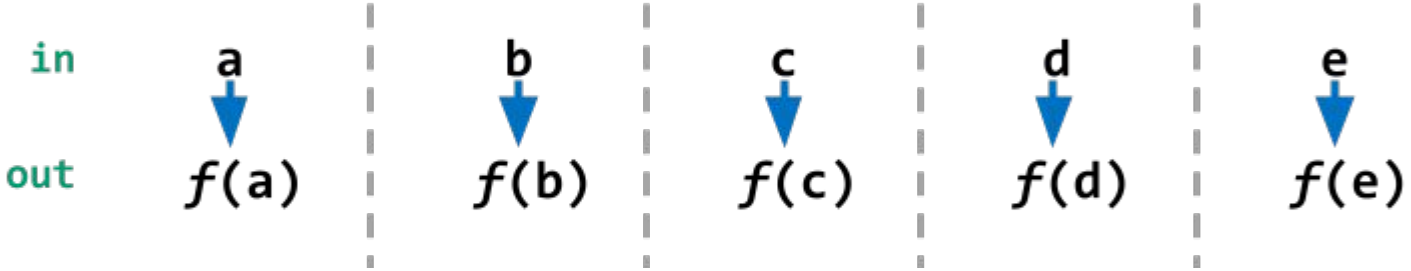
- Data Flip Flop (DFF)
  - Foundational state-keeping component
  - 1-bit input, 1-bit output
  - Wired to the clock signal
  - Always outputs its previous input: `out(t) = in(t-1)`



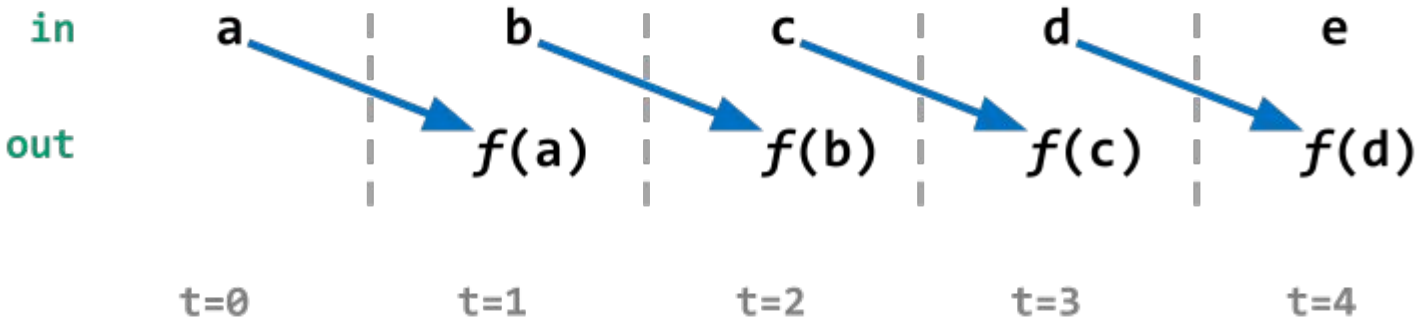
Combinational vs. Sequential Abstraction



Combinational: a function of the inputs from the current time cycle



Sequential: a function of inputs from the previous cycle (has "memory")



DFF Example 2 Specification

- Example specification:

```
out(t) = Xor(out(t-1), in(t-1))
```

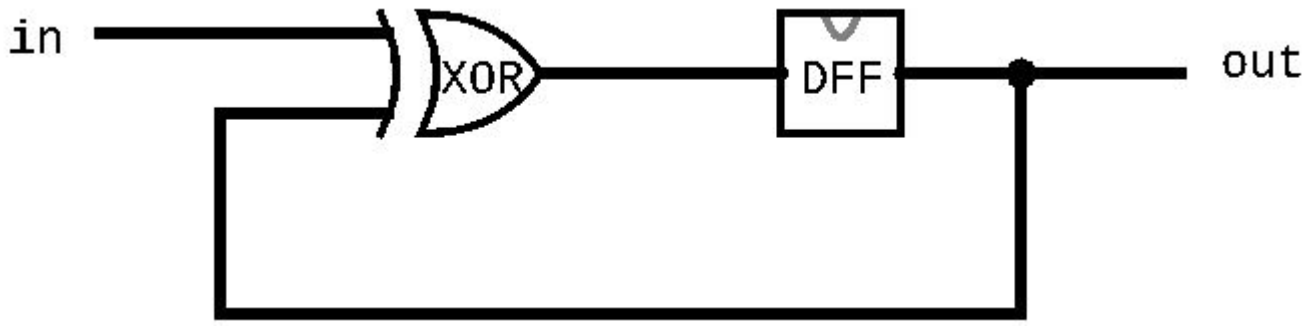
- Notice how the specification uses `out(t-1)` as an input for `out(t)`
  - Need some sort of circular wiring, separated by a DFF

DFF Example 2 Circuit Diagram

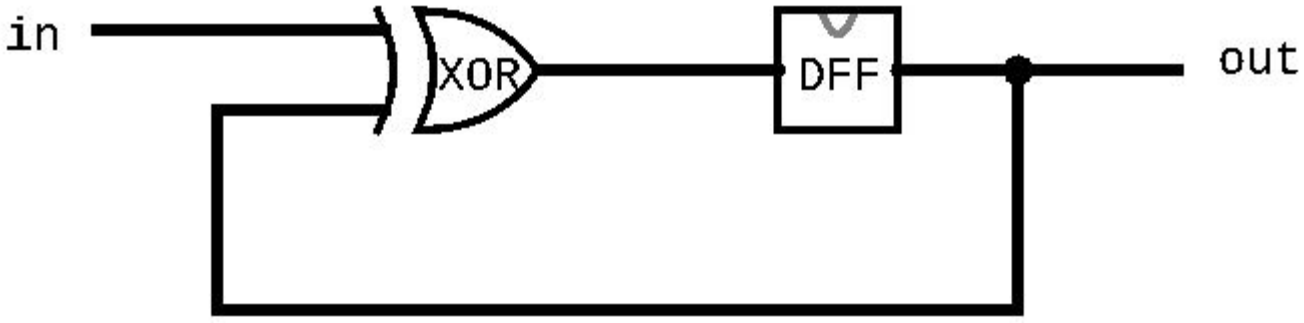
- Example specification:

```
out(t) = Xor(out(t-1), in(t-1))
```

- Circuit diagram:



DFF Example 2 Time Series

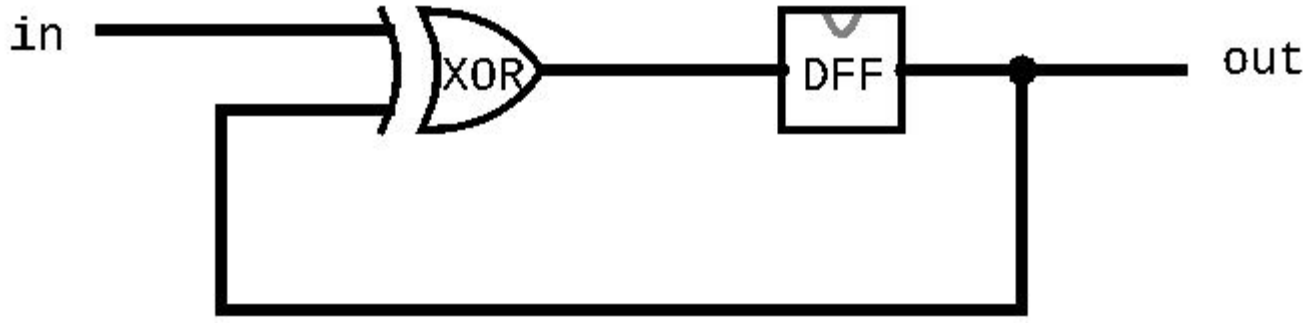


pin	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...
in	0	0	1	1	1	0	0	...
out	0	0	0	1	0	1	1	...

Example:

```
out(t=3) = Xor(in(t=2), out(t=2))
```

DFF Example 2 HDL Implementation



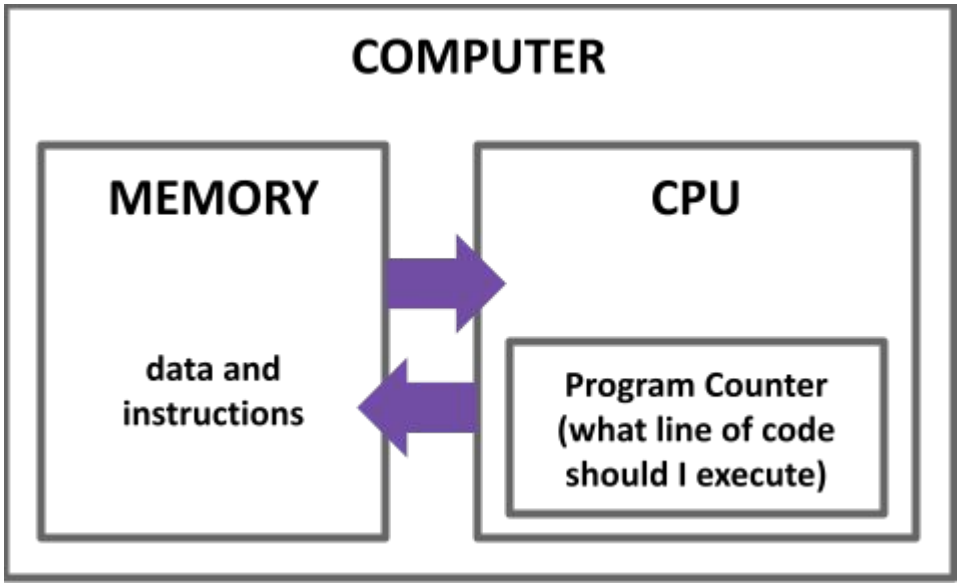

```
CHIP Example2 {
    IN in;
    OUT out;

    PARTS:
    Xor(a=in, b=prevout, out=xorout);
    DFF(in=xorout, out=prevout, out=out);
}
```

Agenda

- Cornell Note-Taking Review
- **Building a Bit**
  - DFF Example Review
  - **Building a bit exercise**
- Building Memory
  - Reading Review and Q&A
  - Building memory: Bit to RAM
  - Program Counter
- Project 3 Overview

Computer Overview



- CPU is the "brain" of our computer
  - Does necessary computations (add, subtract, multiply, etc.)

- Memory is used to store/remember values for later use
  - Needs to be able to persist across multiple computations!
  - Need to be able to change the values when we want to

Storing Data: Bit

- In memory, we want data to stay the same until we tell it to change
- Currently our DFFs always change based on the inputs!
- Need a new circuit: Bit
  - If we tell it to change, updates its value
  - Otherwise maintains its previous value
- Bit is the first chip you implement in project 3!

Storing Data: Bit

- More formal specification of Bit:

```
// if (load(t-1)): out(t) = in(t-1)
//              else: out(t) = out(t-1)
CHIP Bit {
    IN in, load;
    OUT out;

    PARTS:
    // TODO
}
```

Bit Time Series

```
if (load(t-1)): out(t) = in(t-1)
          else: out(t) = out(t-1)
```

pin	t=0	t=1	t=2	t=3	t=4	t=5	t=6	...
in	1	0	0	0	1	0	1	...
load	1	0	0	1	1	1	0	...
out	0	1	1	1	0	1	0	...

Example 1: `load(t=0) == 1` so `out(t=1) = in(t=0)`

Example 2: `load(t=2) == 0` so `out(t=3) = out(t=2)`

Poll Everywhere

pollev.com/cse390b

In order to implement Bit, what gates do you think we will need? (Select all that apply)

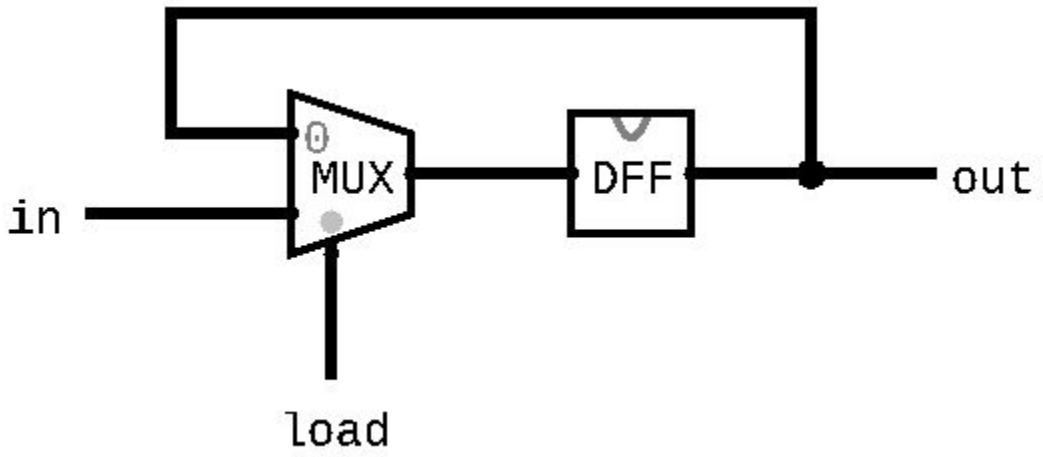
```
Bit Spec:
if (load(t-1)): out(t) = in(t-1)
           else: out(t) = out(t-1)
```

Options:

A. Mux
B. Xor
C. And
D. DFF

Implementing Bit Activity

- Now you know what gates to use!
- In groups, fill in the connections to the gates to create a circuit diagram of Bit
- We encourage you to reference examples from the previous lecture!



Agenda

- Cornell Note-Taking Review
- Building a Bit
  - DFF Example Review
  - Building a bit exercise
- **Building Memory**
  - **Reading Review and Q&A**
  - Building memory: Bit to RAM
  - Program Counter
- Project 3 Overview

Reading Review: Memory Abstraction

- Memory is just one big array!

- **Addresses** are indices into different memory slots
  - The width of an address is fixed for the system
  - The Nand2Tetris project will use 16-bit addresses

- Each slot in memory takes up a fixed width
  - Not the same as address width
  - The Nand2Tetris project uses 16-bit slots in memory

Reading Review: Memory Abstraction

- Can read and write to memory by specifying an address
  - More details next week!

- **Example:** `x = memory[01...00]`
  - Reads the value in memory at address `01...00` and stores it in `x`

- **Example:** `memory[01...00] = 7`
  - Writes the value 7 in the memory slot at address `01...00`

Agenda

- Cornell Note-Taking Review
- Building a Bit
  - DFF Example Review
  - Building a bit exercise
- **Building Memory**
  - Reading Review and Q&A
  - **Building memory: Bit to RAM**
  - Program Counter
- Project 3 Overview

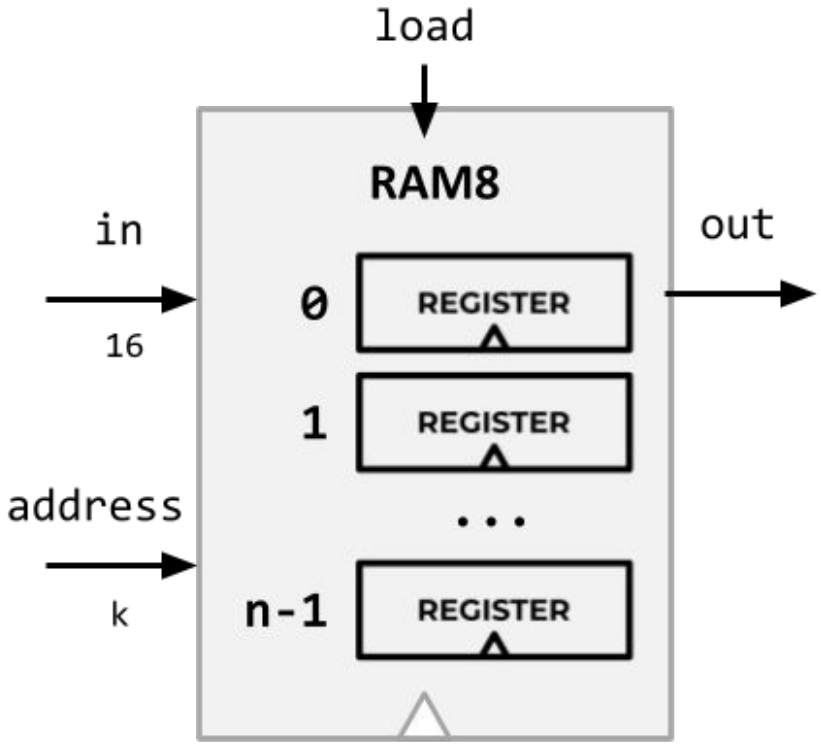
Building Memory: Register

- Bits can store a single value, 0 or 1, but in memory we need a way to store 16-bit values!
- Register: Same idea as a Bit but for 16 of them!
  - Allows us to store and change 16-bit values
  - Groups together 16 individual bits that share a load signal

```
// if (load(t-1)): out(t) = in(t-1)
//              else: out(t) = out(t-1)
CHIP Register {
    IN in[16], load;
    OUT out[16];
    ...
}
```

Building Memory: RAM8 From Registers

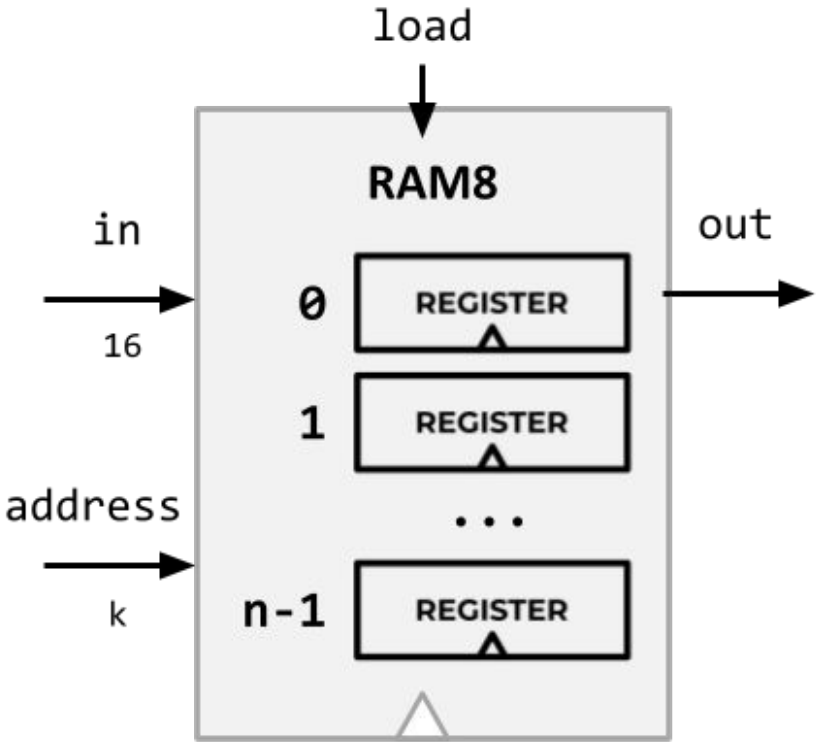
- RAM interface:
  - **`address`**: address used to specify memory slot
  - **`in`**: 16-bit input used to update specified memory slot if load is 1
  - **`load`**: if 1, then **`in`** should be written to specified memory slot
  - **`out`**: 16-bit output from the slot specified by address



- RAM8 can be built from 8 registers
  - address width is $\log_2(8) = 3$ bits

Building Memory: RAM8 From Registers

- Step 1: Route **in** to every register
  - But we don't want every register to update!
  - Solution: choose which register to enable w/ load
- Step 2: Choose which register to use for the output
- These are choices we need to make in circuits: perfect tasks for Mux and Demux



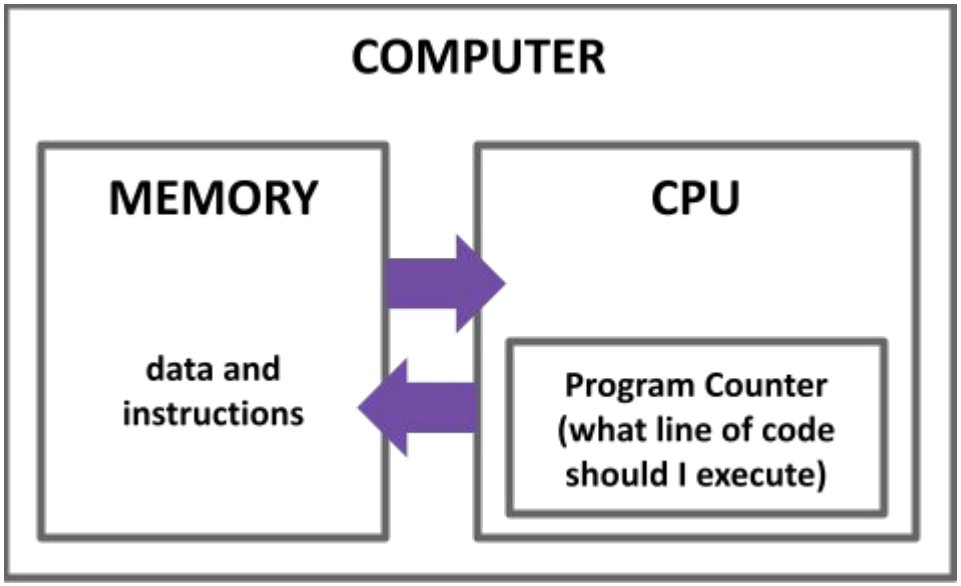
Building Memory: The rest of RAM

- After RAM8, can build larger RAM chips from a combination of smaller RAM chips
  - e.g. RAM64 can be built using eight RAM8 chips
- Technique will be similar to RAM8 but will have to use different portions of the address
- The blocks section of the reading will likely be helpful!
  - For example, can think of each RAM8 as a block of RAM64

Agenda

- Cornell Note-Taking Review
- Building a Bit
  - DFF Example Review
  - Building a bit exercise
- **Building Memory**
  - Reading Review and Q&A
  - Building memory: Bit to RAM
  - **Program Counter**
- Project 3 Overview

Program Counter

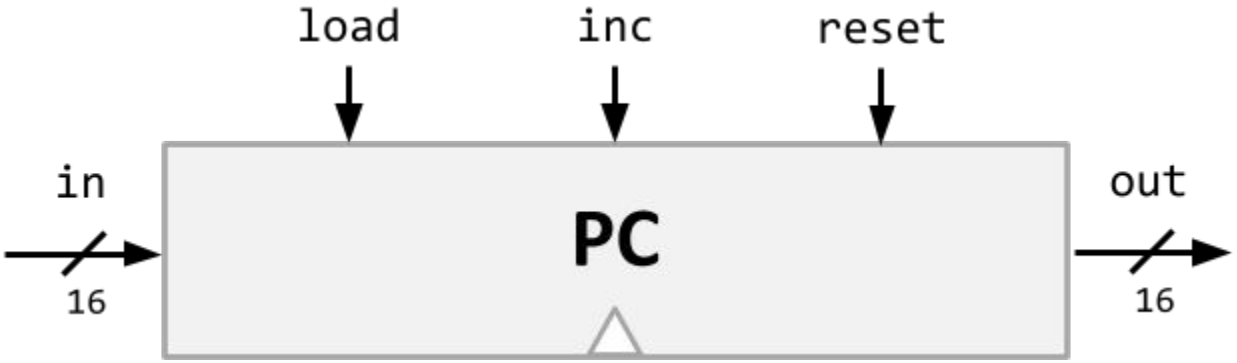


- Memory is used to store data but it also stores code!
- Instructions/operations are stored at different addresses in memory
- Program Counter in the CPU keeps track of which address contains the instruction that should be executed next

Program Counter

- Keeps track of what instruction we're executing
  - If it outputs 24, on the next clock cycle the computer runs the instruction at address 24 in the code segment
- Program counter interface:

```
if       (reset[t] == 1) out[t+1] = 0
else if (load[t] == 1)   out[t+1] = in[t]
else if (inc[t] == 1)    out[t+1] = out[t] + 1
else                     out[t+1] = out[t]
```



Agenda

- Cornell Note-Taking Review
- Building a Bit
  - DFF Example Review
  - Building a bit exercise
- Building Memory
  - Reading Review and Q&A
  - Building memory: Bit to RAM
  - Program Counter
- **Project 3 Overview**

Project 3 Overview

- Cornell Note-Taking Method
  - Practice taking detailed notes in another class. Think critically about the technique.
- Memory & Sequential Logic
  - Build our first sequential chips, from a 1-bit register to a 16K RAM module.
- Program Counter
  - Build a counter that can keep track of where we are in a program, with support for several operations we'll need later.

- Note: folder split for performance reasons only!

Wrapping Up

What's in store for Week 4?

- Assembly Languages
- Annotation
- Project 4 Released

Reminders

- Project 1 Grades Released on Gradescope
- Project 2 Due Tonight 4/15 11:59PM PDT