CSC258: Intro to Digital Logic

Instructors: Steve Engels, sengels@cs.toronto.edu
Sajad Shirali-Shahreza, shirali@cs.toronto.edu

CSC258 Course Details

- Course Goals:
  - Understand the underlying architecture of computer systems.
  - Learn how to use this architecture to store data and create behaviour.
  - Use the principles of hardware design to create digital logic solutions to given problems.

CSC258 Course Details

- Lectures / tutorials
- Labs (28%):
  - 7 total (4% each)
  - Must complete pre-lab exercises ahead of time.
- Project (14%):
  - Large, cool digital creation.
  - Proposal and 3 demos.
- Exams:
  - Midterm (18%) – Feb 28th.
  - Final exam (40%)
    - must get 40% to pass the course.

Let the learning begin

Why take CSC258?

- To understand computers better!
    1. How does a computer store "`false`" or "`true`" in memory?
    2. Why is there a maximum value for an `int`?
    3. Where do the operators "`and`", "`or`" and "`not`" come from?
    4. What actually happens when you press `Ctrl-Alt-Delete` on your keyboard?
    5. What happens when a Java or C program is compiled?

CSC258 has the answers!

- Computers are physical things, therefore they have certain behaviours and limitations:
  - Data values are finite.
  - All data is stored as ones and zeroes at some level.
  - Many high-level operations depend on low-level ones.

- The way computers are today take their origins from how computers were created in the past.

Example #1: Booleans

- How are boolean values stored?
- Example: `if` statements:

```
if x:
    print 'Hello World'
    # what values can x have
    # that make this happen?
```

  - What if `x` is a boolean?
  - What if `x` is an int?
  - What if `x` is a string?

All comes down to hardware in the end!

Text

Example #2: Integers

- How are integers stored?
  - Again, as ones and zeroes.

Decimal

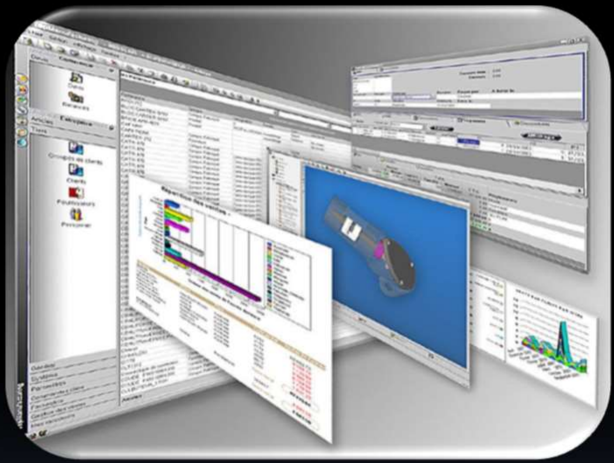
```
1234 → 00110000011001
```

Binary

- How many values can integers have?
  - This can vary based on language and architecture, but generally integers have 2^{32} different values.
  - <u>Signed integers</u>: range from $\boxed{-2^{31} \text{ to } 2^{31}-1}$
  - <u>Unsigned integers</u>: range from 0 to $2^{32}-1$
  - Different ranges for `long`, `short` and `byte`.

Programming hardware

- Computers do on the hardware level what programs do on the software level.



- In CSC258, designing circuits follows many of the the same ideas behind creating boolean logic in Python or Java.

Programming parallels

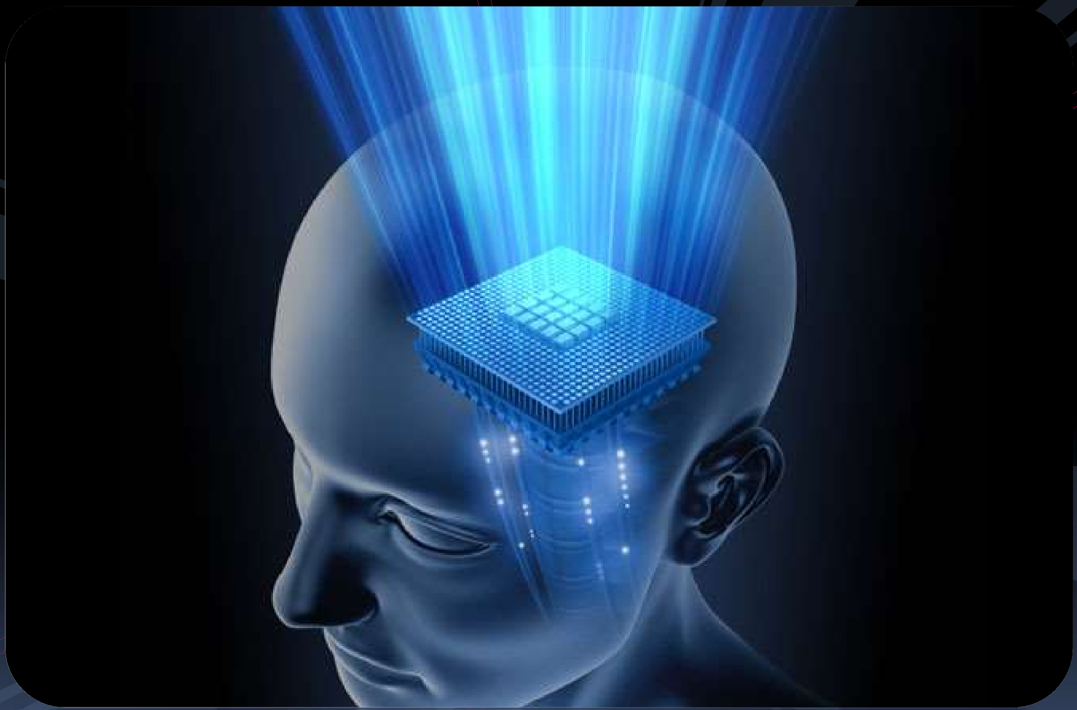
Python/Java

- Boolean variables
- Boolean operations (and, or, not, etc)
- Integers, doubles, chars
- Addition, subtraction, multiplication
- Storing values
- Executing instructions

Computer hardware

- High and low wire values
- Logic gates (AND, OR, NOT, etc)
- Registers
- Adder circuits, multiplier circuits
- Memory
- Processors

What you already know



Programming from CSC148

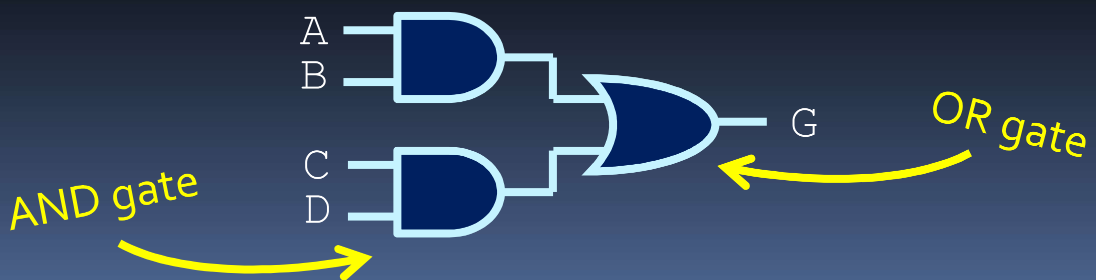
- Need to have basic programming literacy.
- For CSC258, be prepared to let that all go.
  - Verilog → specification language
  - Assembly → low-level programming
- Trying to connect these languages to CSC148 will only hold you back.
  - Practice practice practice

Logic from CSC165

- What you may not realize, is that you already know how some of the basics of logic gates.
- <u>Example:</u> Create an expression that is true if the variables `A` and `B` are true, or `C` and `D` are true.

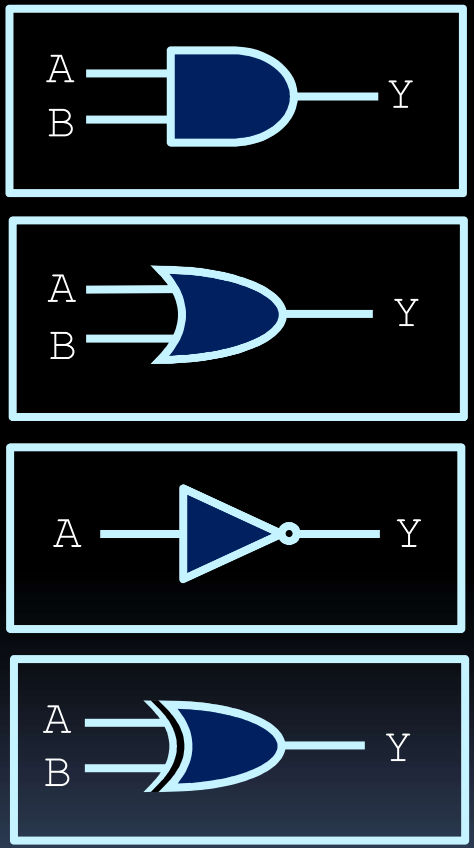
```
G = A & B | C & D
```

- Now create a circuit that does the same thing:



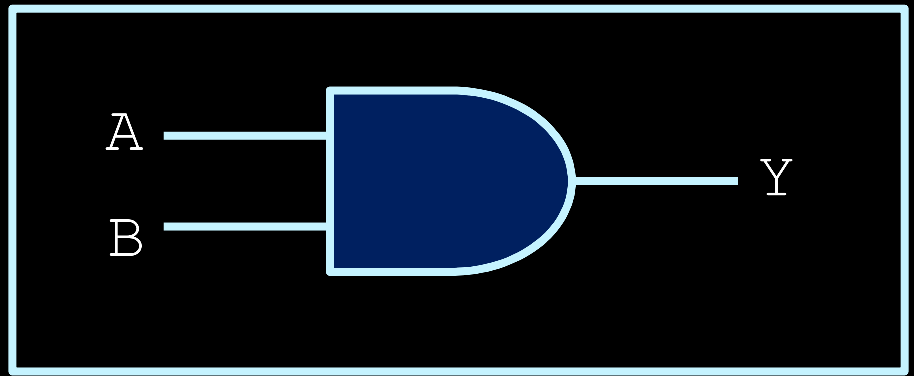
Gates = Boolean logic

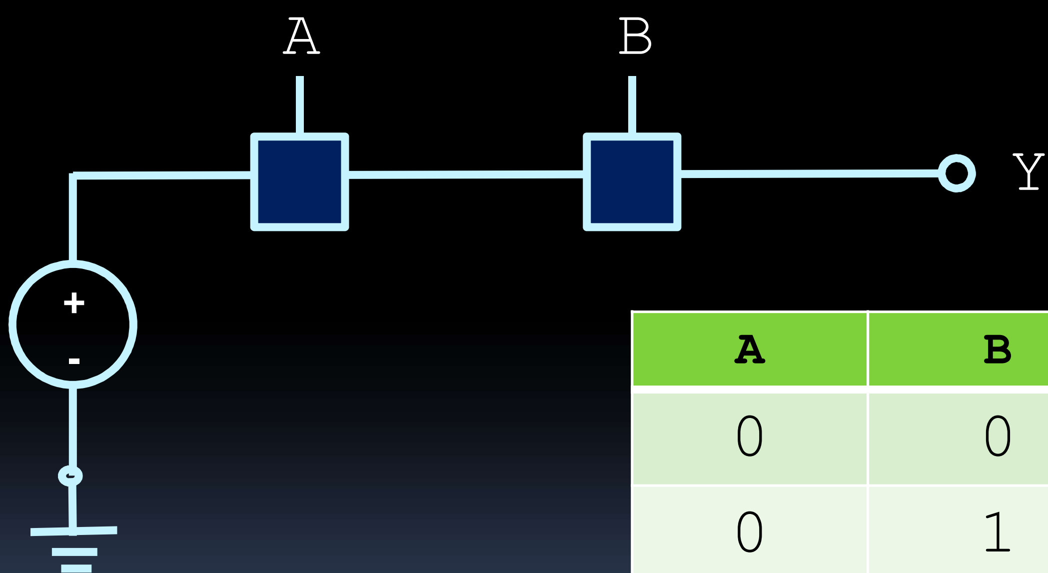
- If you know how to create simple logical expressions, you already know the basics of putting logic gates together to form simple circuits.
- Just need to know which logic operations are represented by which gate!



XOR

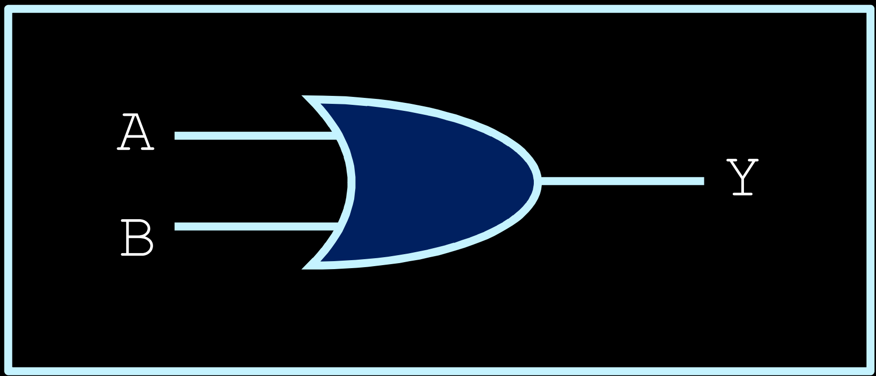
AND Gates

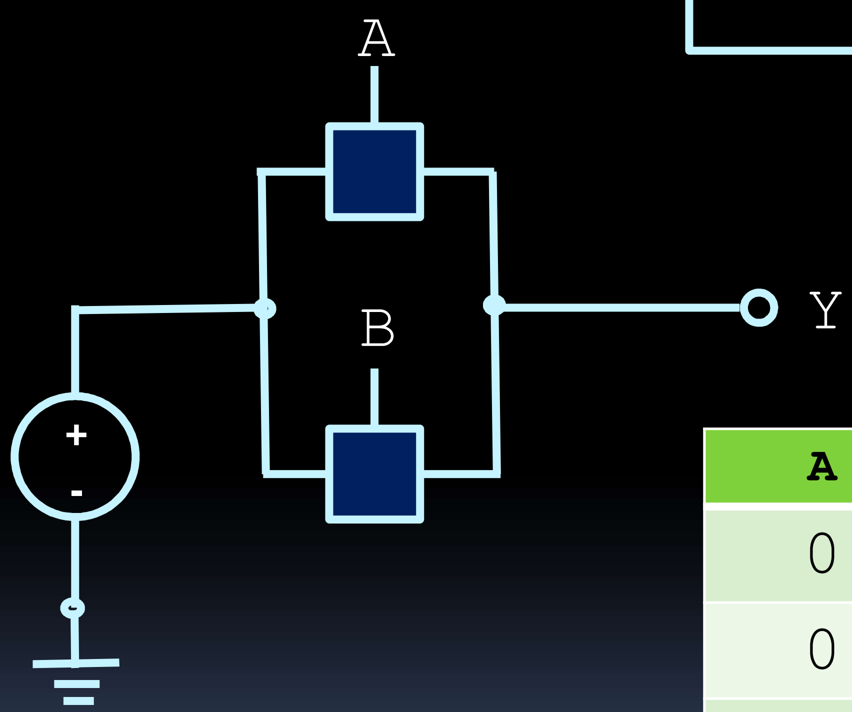




A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

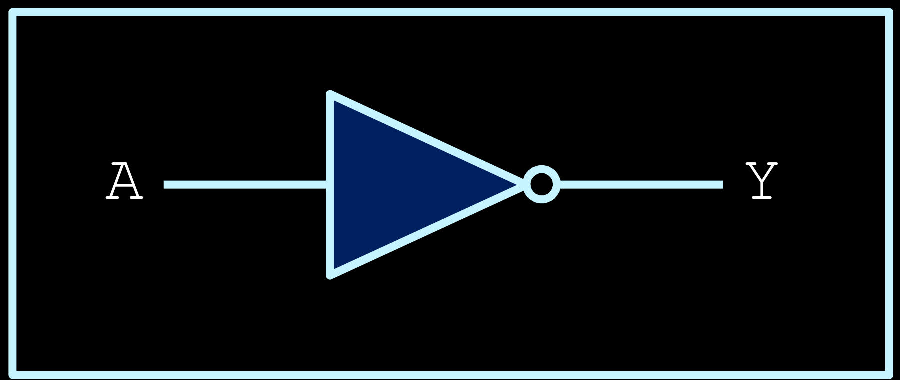
OR Gates

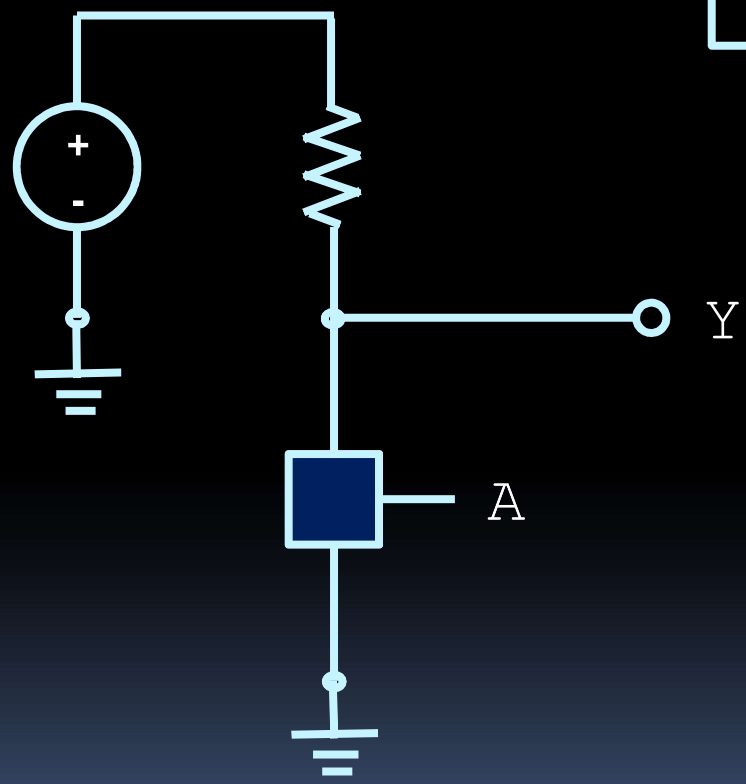




A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gates

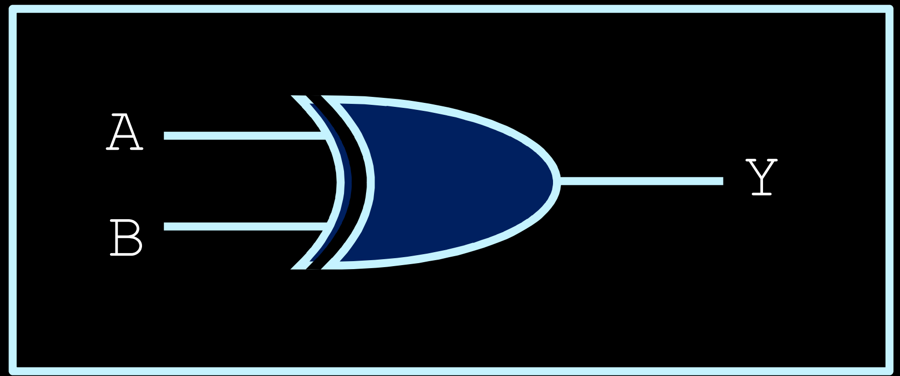


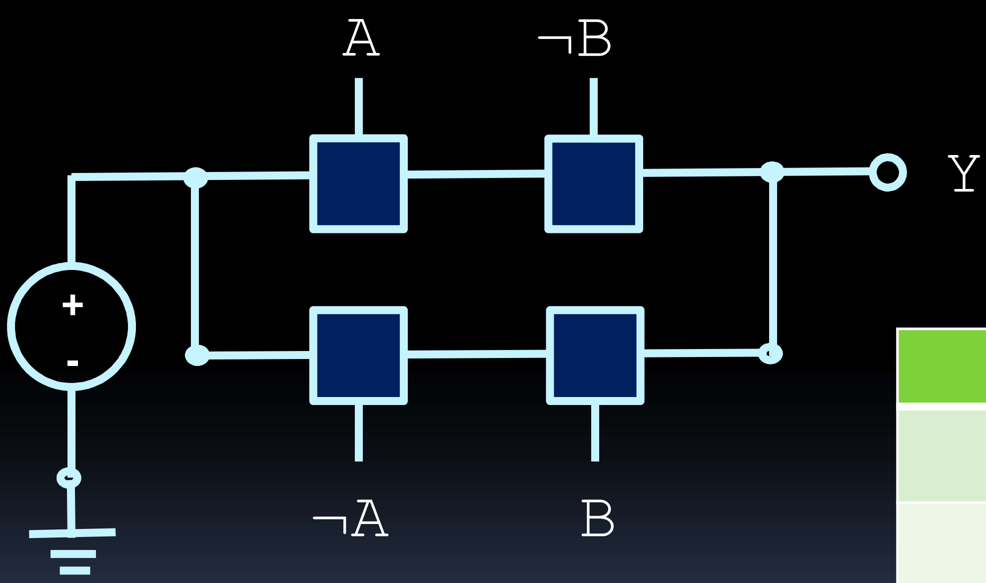


A	Y
0	1
1	0

XOR Gates

either but not both inputs are high





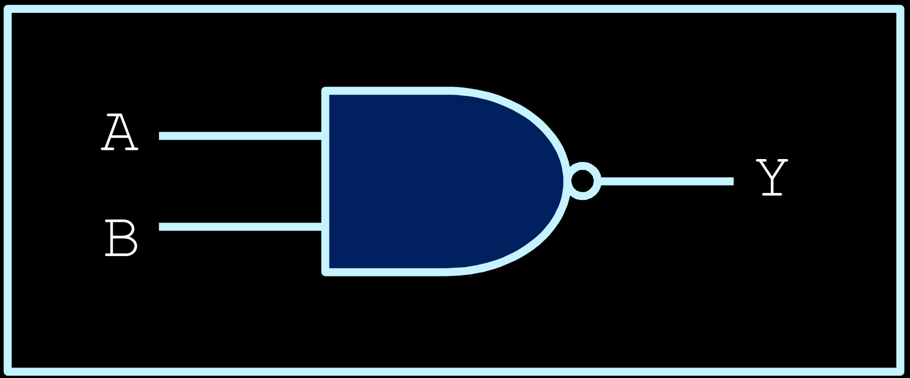
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

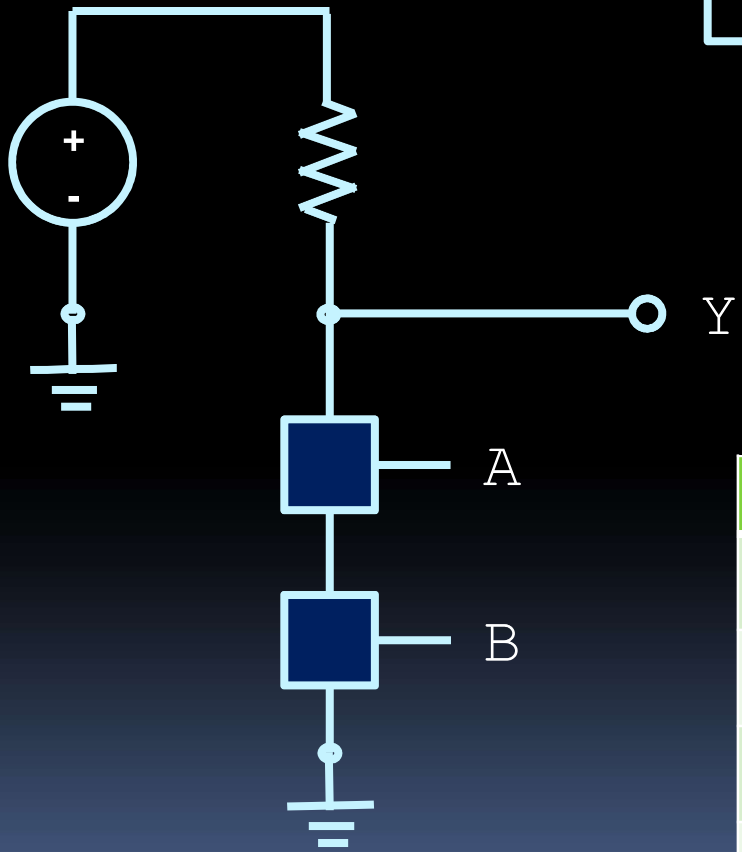
Bill Gates

- …ha ha…moving on….

NAND Gates

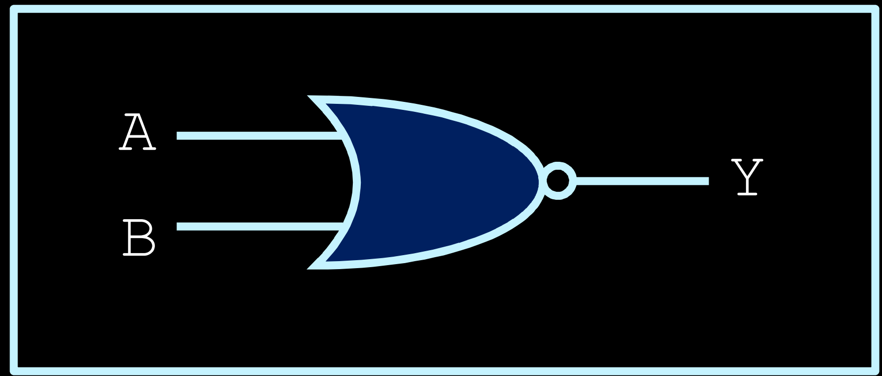
AND gate followed by a NOT gate

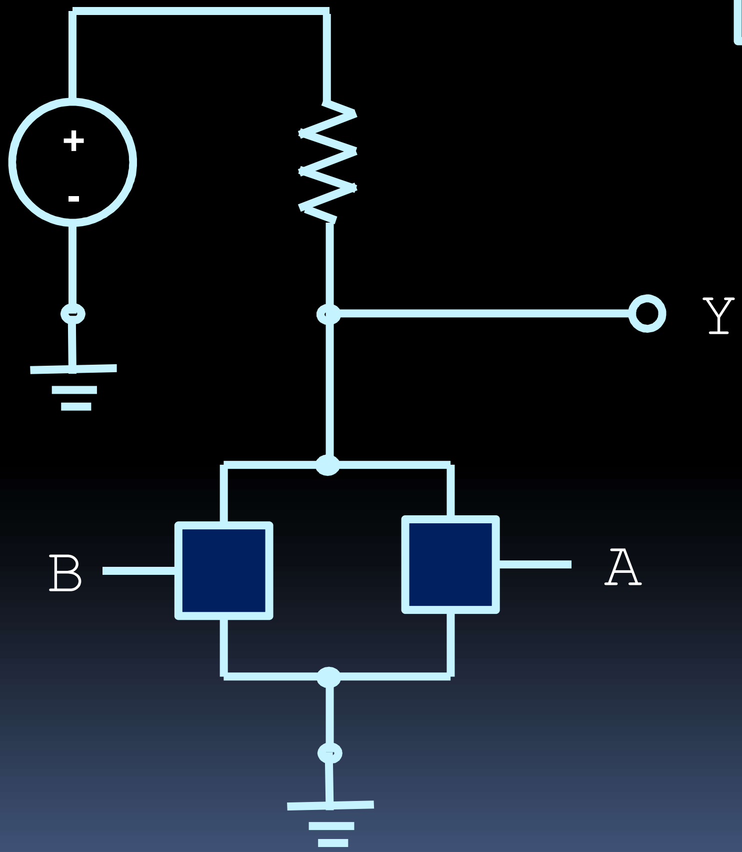




A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

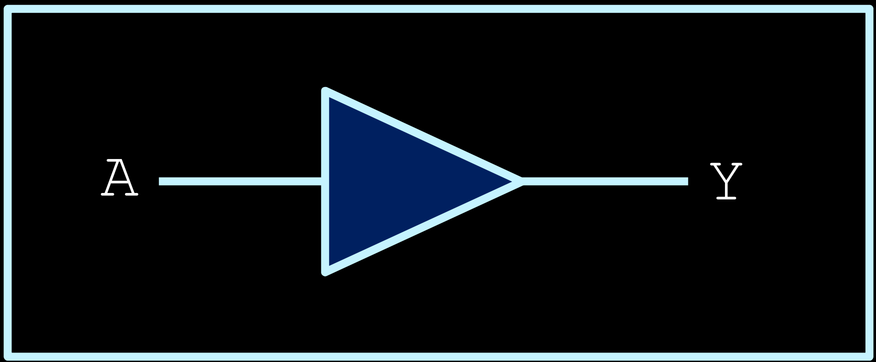
NOR Gates

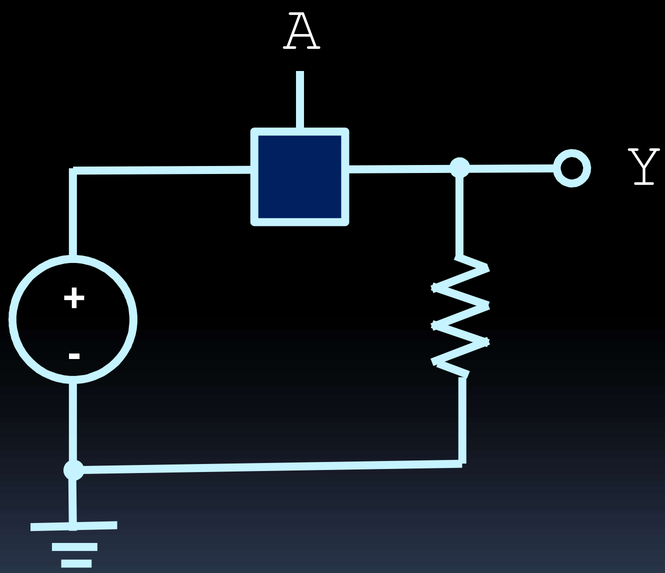




A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Buffer



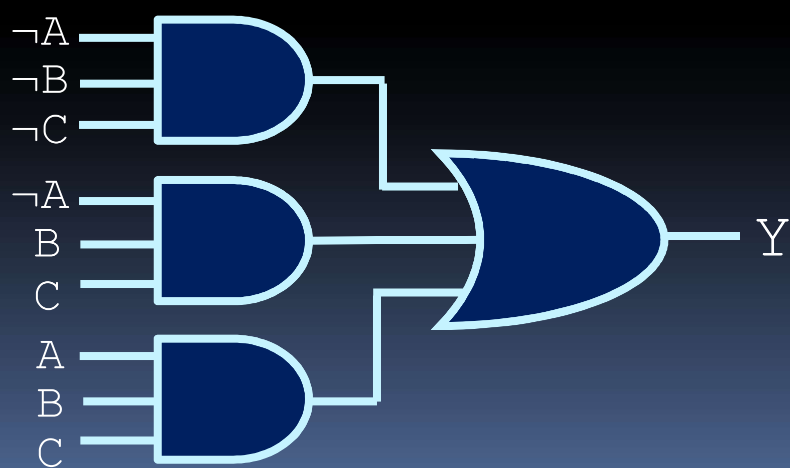


A	Y
0	0
1	1

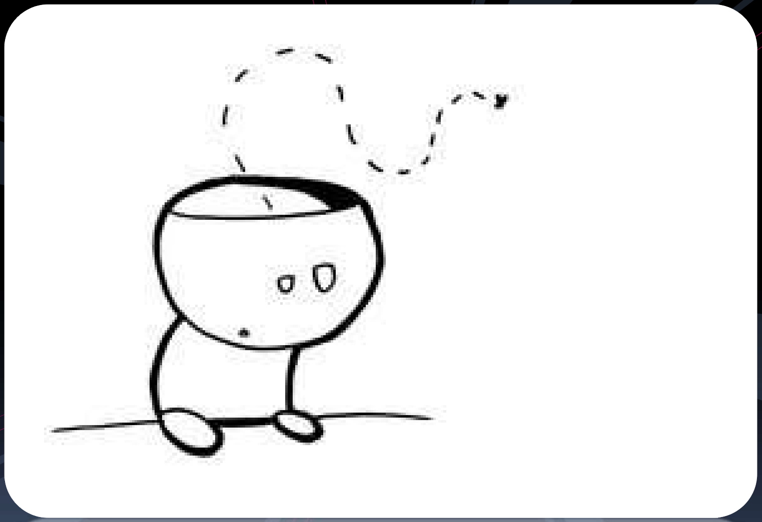
Creating complex circuits

- Creating circuit logic is the same as working with boolean logic in Python, C or Java:

```
Y = (!A and !B and !C) or (!A and
    B and C) or (A and B and C)
```



What you might not know yet

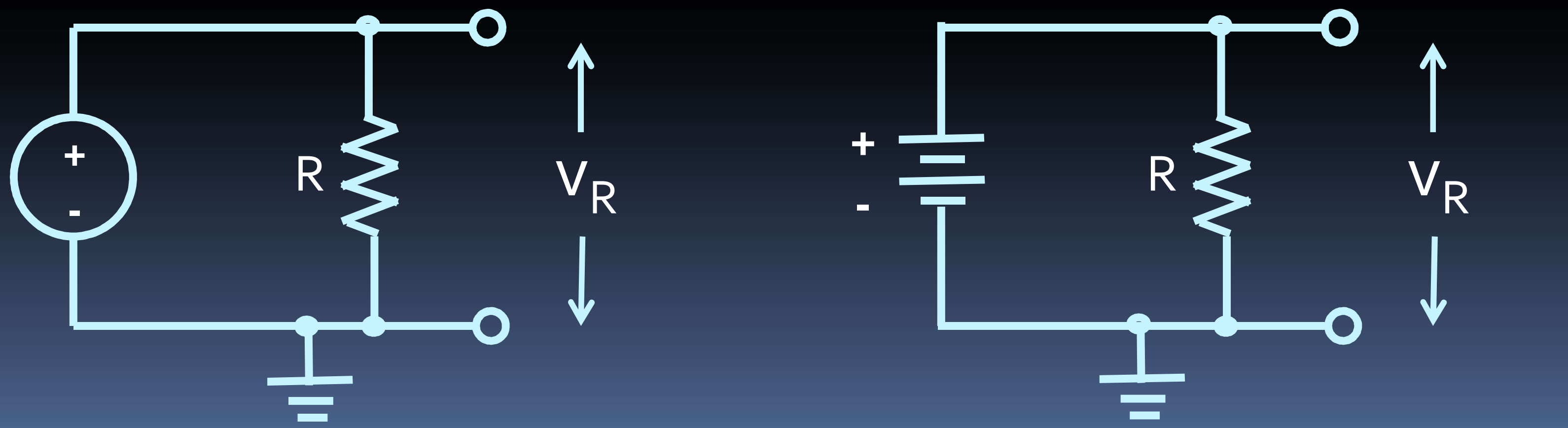


Thinking in hardware

- CSC258 is very, very different.
- Unlike software, CSC258 is not about creating programs and algorithms, but rather devices and machines.
  - Very important concept to grasp early in this course!
  - Need to understand what certain terms mean in the context of hardware.

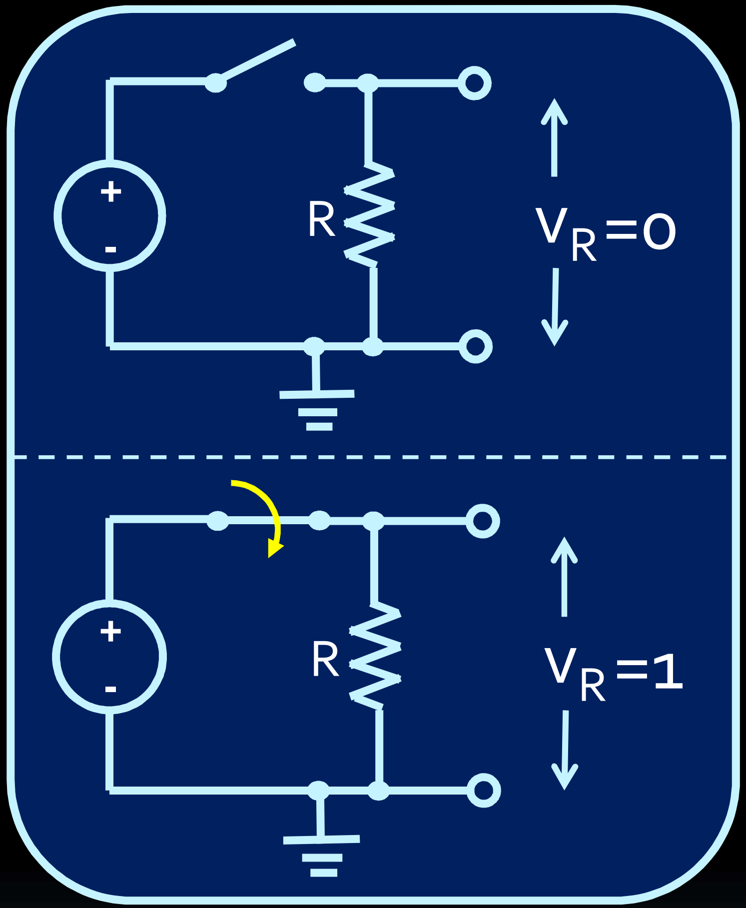
What is “true” and “false”?

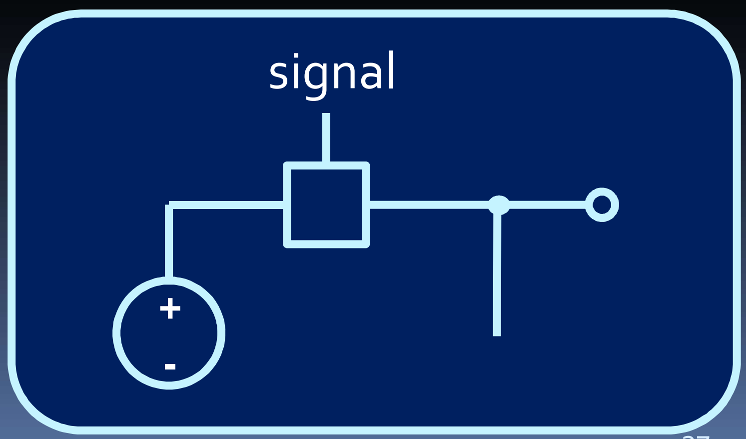
- Once you know the logical operation a circuit performs, all that’s left is supplying the input values.
  - How do we represent boolean values like “true” and “false”?
- In hardware, “true” and “false” refers to the electrical voltage values on the wires.
  - <u>Zero</u>: little to no voltage at that point.
  - <u>One</u>: typically a voltage difference of 5 volts, relative to the ground.



What are gates?

- Gates are like switches, which control whether an output wire will have a high value (5V) or a low value (0V)
  - Switches are physical devices for manually closing a circuit.
  - Gates are semi-conductor devices that close a circuit electrically.

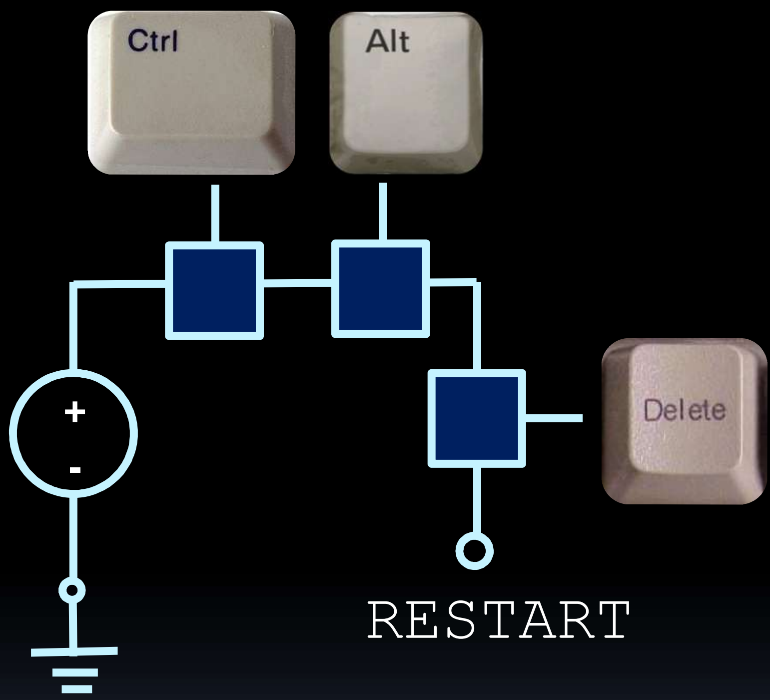




What are circuits?

- Assuming that certain signals can be turned on (one) or off (zero), we need have ways to combine these signals together.
  - <u>Example #1:</u> If the Ctrl, Alt and Delete buttons are being pressed, restart the computer.
  - <u>Example #2:</u> If three train tracks converge onto a single track, only turn on the green light if a single track has a train waiting.
- Every electronic device uses gates to combine input signals to create output signals.
- Very similar to CSC165 problems, but in hardware.

Example: Ctrl-Alt-Delete

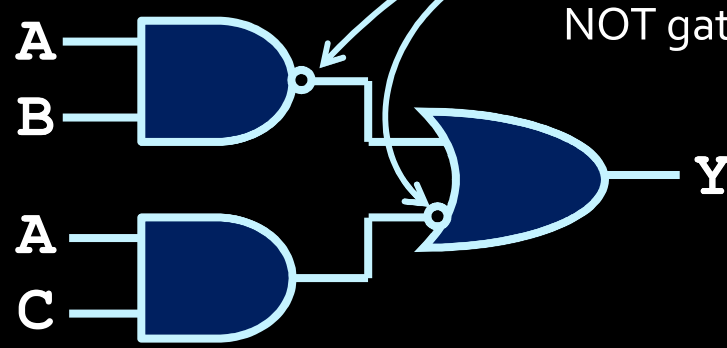


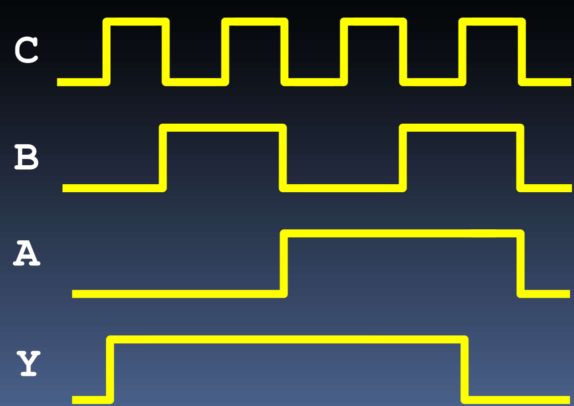
Ctrl	Alt	Del	RESTART
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Expressing digital logic

- Given a logic problem or circuit, truth tables are used to describe its behavior (as in CSC165).
- Sometimes illustrated using a waveform.

(the little circle here acts like a NOT gate)





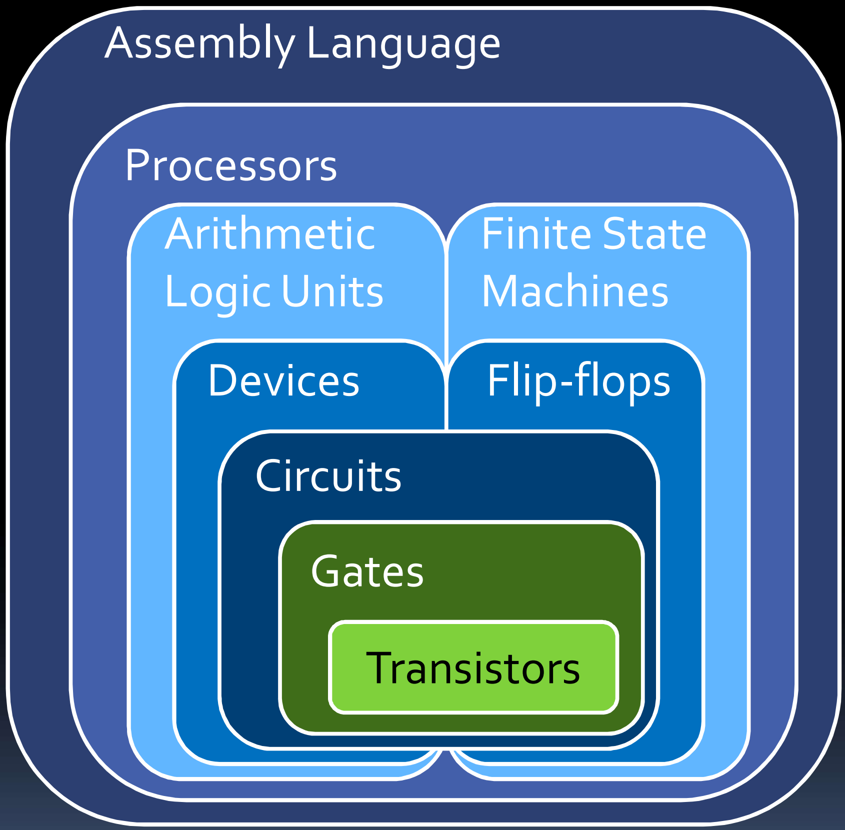
A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Digital logic tasks (e.g. Lab 0)

- Given a truth table or circuit description, determine the circuit that creates it.
- Look at the conditions that cause high output signals.
- Express the high conditions as a boolean statement, then convert this to gates.

A	B	C	MOVE
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

The course at a glance



Starting from the bottom

- Gates can combine values together like logical operators in C or Java.
- But how do gates work?
  - First, we need to understand electricity.
  - Then, we need to understand transistors.