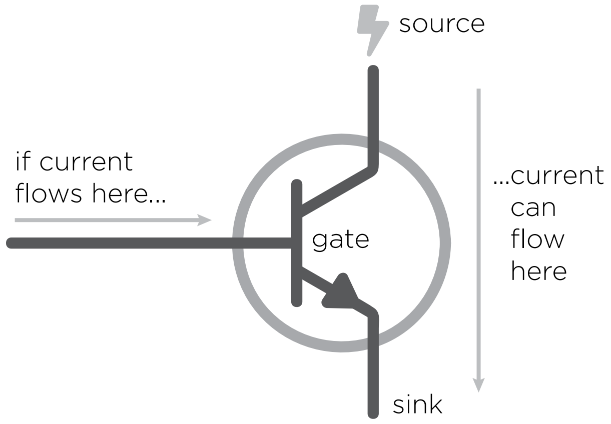
CS50 Transistors and Logic

Overview

Computers represent and process 1s and 0s using sequences of physical components called transistors. By linking different configurations of these transistors, computers can perform everything from basic arithmetic to playing a video. However, thanks to layers of abstraction, we don't need to constantly think about at the level of binary and transistors to program a computer. Sequences of transistors can be represented by Boolean logic, and these sequences of logic gates can then be packaged into chips (hardware) and code (software).

Key Terms

- transistor
- semiconductor
- true
- false
- Boolean logic



Transistors

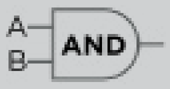
Transistors are small hardware devices made of semiconductors that act as switches for electric current. Because transistors are made of **semiconductors**, transistors can behave as both insulators (materials that inhibit electron flow) and conductors (materials that enable electron flow). When a small current is provided into a transistor's gate, the gate "opens" and current can flow from the source to the sink. When no current is available at the gate, the gate "closes" and current cannot flow from the source to the sink. By controlling current flowing into the gate, transistors can manipulate how current flows and therefore which signals are sent.

Boolean Logic

Since transistors either enable or disable the flow of electricity, we can use transistors to represent the binary values 1 and 0, or **true** and **false**. By linking these transistors in complicated webs, we can implement complex processes using a branch of math known as **Boolean logic**, created by the mathematician George Boole.

Boolean logic is built upon the two Boolean values, true and false, and the fundamental operators AND, OR, and NOT. Similar to arithmetic operations, these operations take value(s) as input and output one value. For example, in the statement 2 + 3 = 5, 2 and 3 are our inputs, + is our operator, and 5 is our output. Boolean logic looks very similar. In the statement "true AND false = false," true and false are our inputs, AND is our operator, and false is our output. The trick here is that while you know what the + sign does, you may not be familiar with the rules of Boolean operators. Luckily, they operate similarly to how these words are used in English. The AND operator requires that the first and second input are true to output true. Otherwise, it returns false. For example, the statement "the shirt is green AND striped" is only true if the shirt is both green and striped. The OR operator requires that either the first or second input is true for it to return true. The statement "the shirt is green OR striped" is true for a green shirt, a striped shirt, and a green striped shirt, but not true for any other shirt. The NOT operator, like a negative sign, only takes in one value and simply flips it, changing true to false and vice versa. The negation of the statement "the shirt is green AND striped" would be "the shirt is NOT green OR NOT striped."

By combining these fundamental operators in sequences, we can build gates like NAND (not and), NOR (not or), and XOR (exclusive or), and then eventually basic arithmetic, and then even more complex operations like editing a photo.

A	B	 AND	OR	NOT
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true