Basic Electronics and Digital Logic
Computer Science 240

Laboratory 1

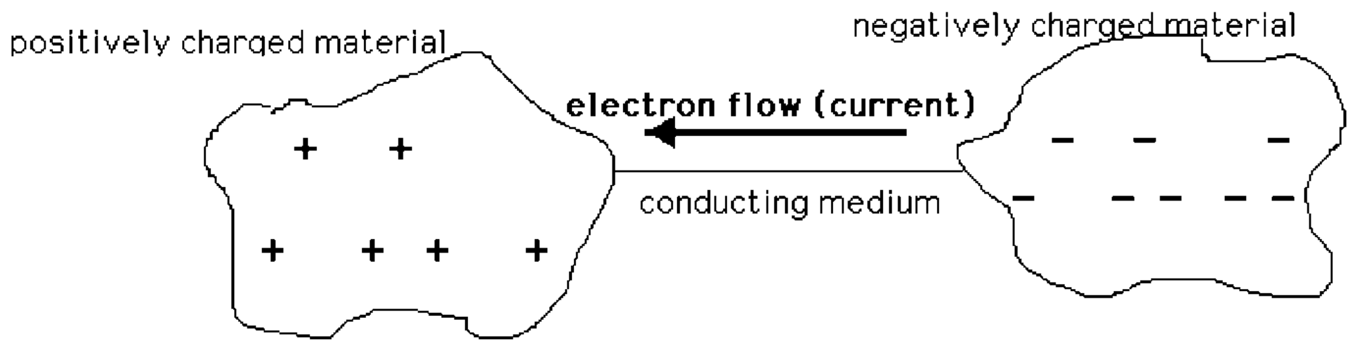
- **Administrivia**
- **Lab Environment**
- **Basic Electronics (Ohm's law, transistors, logic gates)**
- **Sum-of-Products and Equivalence**
- **Integrated Circuits**
- **Protoboard (for building physical circuits)**
- **LogicWorks (for simulating circuits)**

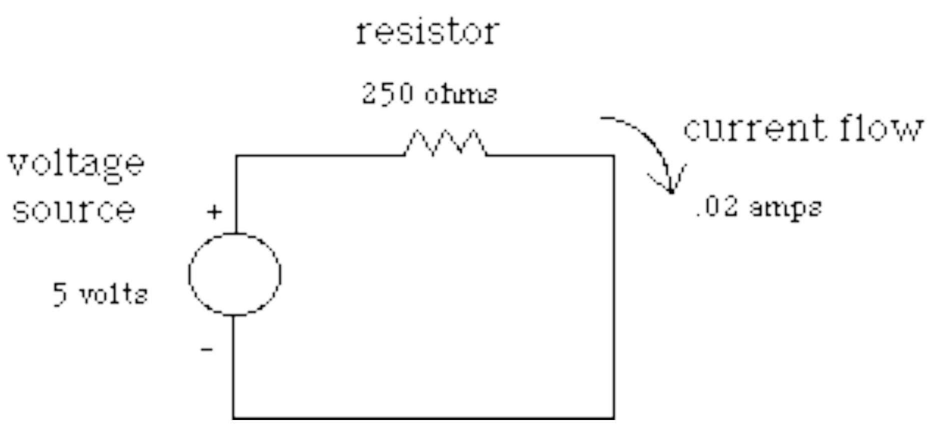
Lab Environment

- All lab exercises and reports will be *Google Docs*, and should be shared with lab partner and the instructor.
- You will switch partners each lab.
- To log in to a machine booted to Windows, use your Wellesley network username and password.
- To log in to a machine booted to Linux, use your CS username and password (should be the same as your account for CS 111 and/or CS 230).

Basic Concepts of Electricity

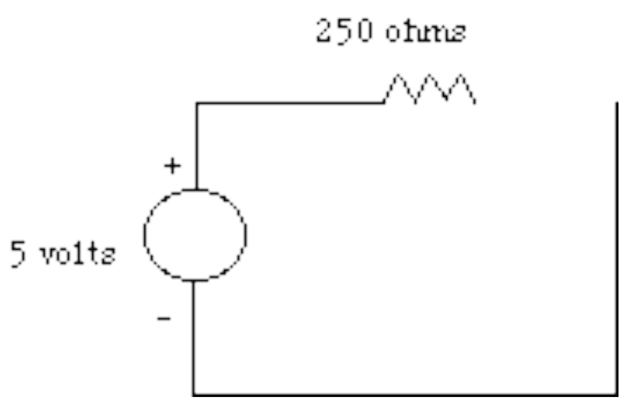
- Electricity = **the movement of electrons** in a material
- Materials tend to have a net negative or positive charge
- Difference of charge between two points = **potential difference/voltage** (**V**, measured in **V**olts**)**
- When you connect two materials with a potential difference using a conducting medium (such as a wire), the electrons will flow to try to balance the charge
- Rate at which flow of electrons is called **current I (**measured in **A**mps**)**.
- The conducting material has an integral ease of conduction to the flow of electrons called **resistance R (**measured in Ohms **Ω)**



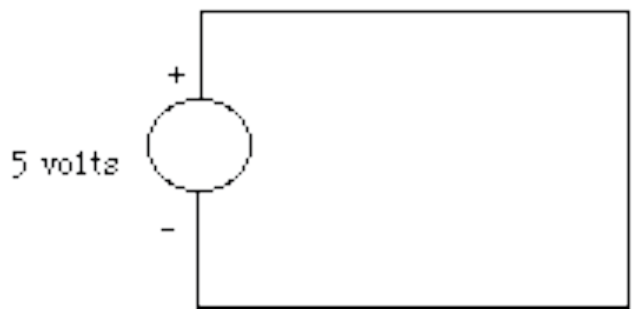


Ohm's Law, $V = IR$.

Open circuit = no current:



Short circuit = infinite current, since $V/0$ = infinite current:



Infinite current swiftly results in the destruction of the circuit!

The basis of electronic computers is that we can specify a voltage measured in a circuit as either *high (*close to the voltage source) or *low* (close to 0 volts, or ground).

Therefore, using electronic circuits, we can represent Boolean values (high = true, low = false), and we can also represent numbers using the binary number system, with high = 1 and low = 0.

Basic Gates and Truth Tables

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

A **truth table** specifies the output for all the given input combinations of a Boolean function. We represent a value of *true* with a 1 and *false* with a 0.

If a function has two inputs **A** and **B** (called *literals*), and the function is true when both input are true, for example:

when **A** = 1 **AND B**=1

that can be represented by the *minterm* **AB**, which implies **A AND B**, since an **AND** operation is implied by two inputs placed next to one another.

AB is only true when **A** = 1 **AND B** = 1

For a function that is true only when one of the inputs is false, for example:

when **A** = 1 and **B** = 0

we use the *inverse* of **B** in the minterm, **AB'** (**B'** means **NOT B**).

B' is true only when **B** = 0

AB' means **A AND NOT B**, and is only true when **A** = 1 and **B** = 0.

An **OR** operation is expressed by the + operator (such as **A + B**, meaning **A OR B**).

There are several basic logic functions which are fundamental to our study of digital electronics (including symbols used to represent the function):

NOT	**NAND**	**NOR**	**AND**	**OR**
F = A’	**F = (AB)’**	**F = (A+B)’**	**F = AB**	**F = A + B**
A F 0 1 1 0	**A B F** 0 0 1 0 1 1 1 0 1 1 1 0	**A B F** 0 0 1 0 1 0 1 0 0 1 1 0	**A B F** 0 0 0 0 1 0 1 0 0 1 1 1	**A B F** 0 0 0 0 1 1 1 0 1 1 1 1
				

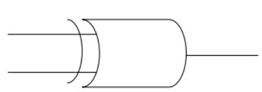
Although NOT, AND, and OR are the only functions needed for expressing sum-of-products, it turns out that NAND (NOT AND, the opposite of AND) and NOR (NOT OR, the opposite of OR) are also very useful.

In addition, the Exclusive-OR function (XOR) is also considered a basic logic function, because it can be used for comparison of bits, which is quite useful for many tasks, including addition!

XOR

F = AB’ + A’B

A	**B**	**F**
0	0	0
0	1	1
1	0	1
1	1	0



Sum-of-Products
Boolean functions be expressed in a **sum-of-products** form, which uses AND, OR, and NOT basic functions. For example, given the following truth:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

By sum-of-products, F is true if:

A = 0 AND B=1 (A'B)
-OR-
A = 1 AND B= 0 (AB')
-OR-
A = 1 AND B=1 (AB)

so, F = A'B + AB' + AB

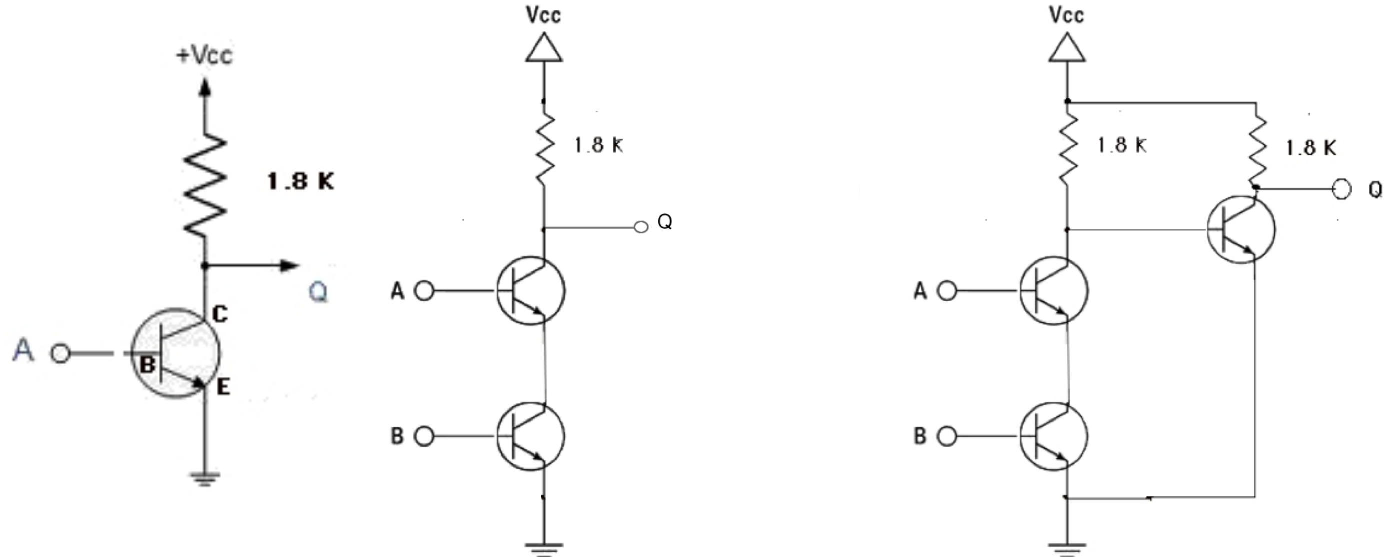
Since a Boolean function can be expressed using only NOT, AND, and OR basic functions using sum-of-products, if we had an electronic circuit which could produce these basic functions (assuming a high voltage measurement can represent true and a low voltage measurement can represent false), that means that we can build a circuit for any Boolean function.

Transistors

A key to the development of modern computers was the invention of devices that can act like a switch (can be turned on or off). Although the early devices were large (such as vacuum tubes), and used a variety of technologies, eventually **transistors**, miniature electric switches, were developed.

In addition to acting as a switch, transistors can be used to produce the basic logic functions:

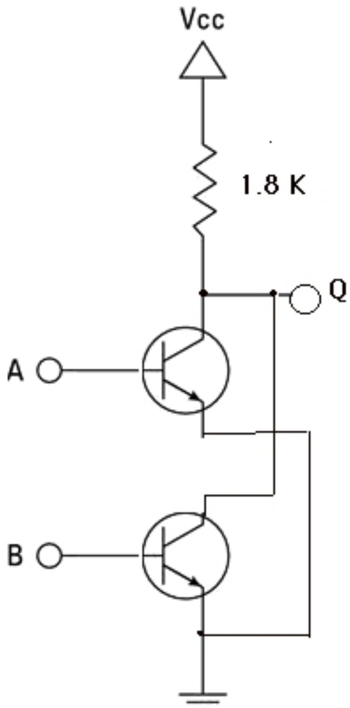
NOT – 1 transistor **NAND – 2** transistors **AND** – 3 transistors

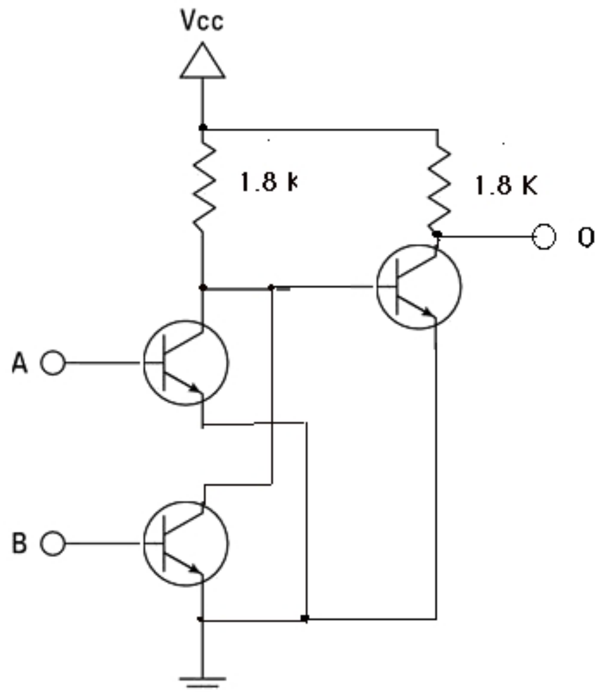


The **AND** gate uses 3 transistors and is basically a **NOT NAND** (it sends the output of a NAND through another transistor acting as a NOT gate to complement the result):

Similarly, these are the transistor circuits for a NOR and OR gate:

NOR – 2 transistors **OR** – 3 transistors





Equivalence

Two functions which produce the same truth table are considered *equivalent*.

For example, the functions F and Q can be shown to be equivalent:

F = A'B' + A'B

Q = A' + A'B + A'B':

A	B	A'B'	A'B	A'B' + A'B
0	0	1	0	1
0	1	0	1	1
1	0	0	0	0
1	1	0	0	0

A	B	A'	A'B	A' B'	A'+A'B+A'B'
0	0	1	0	1	1
0	1	1	0	0	1
1	0	0	0	0	0
1	1	0	0	0	0

When there is an equivalent function/circuit that uses fewer gates, transistors, or chips, it is preferable (cheaper and faster) to use that circuit in a design.

Equivalence can also be proven through use of Boolean algebra, which you will soon be covering in lecture.

Integrated Circuits

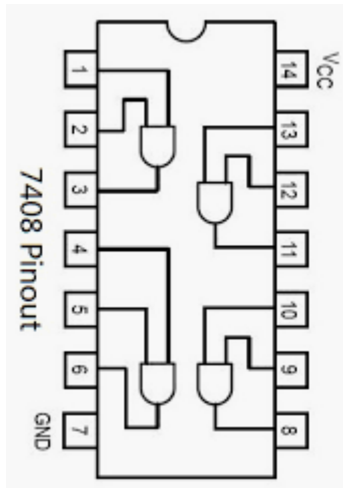
Integrated circuits (chips) contain transistors which perform a specific function.

The **pinout** (found in a datasheet from a TTL Data Book or online) shows the physical layout of the pins and the purpose of the device:

Pins are numbered, starting with "1" at the top left corner and incremented counter-clockwise around the device.

Top left pin is pin 1, always to left of notch in chip, is often marked with a dot.

Bottom left pin is often connected to the negative terminal of the power supply (called **Ground**, and assumed to be 0 Volts).

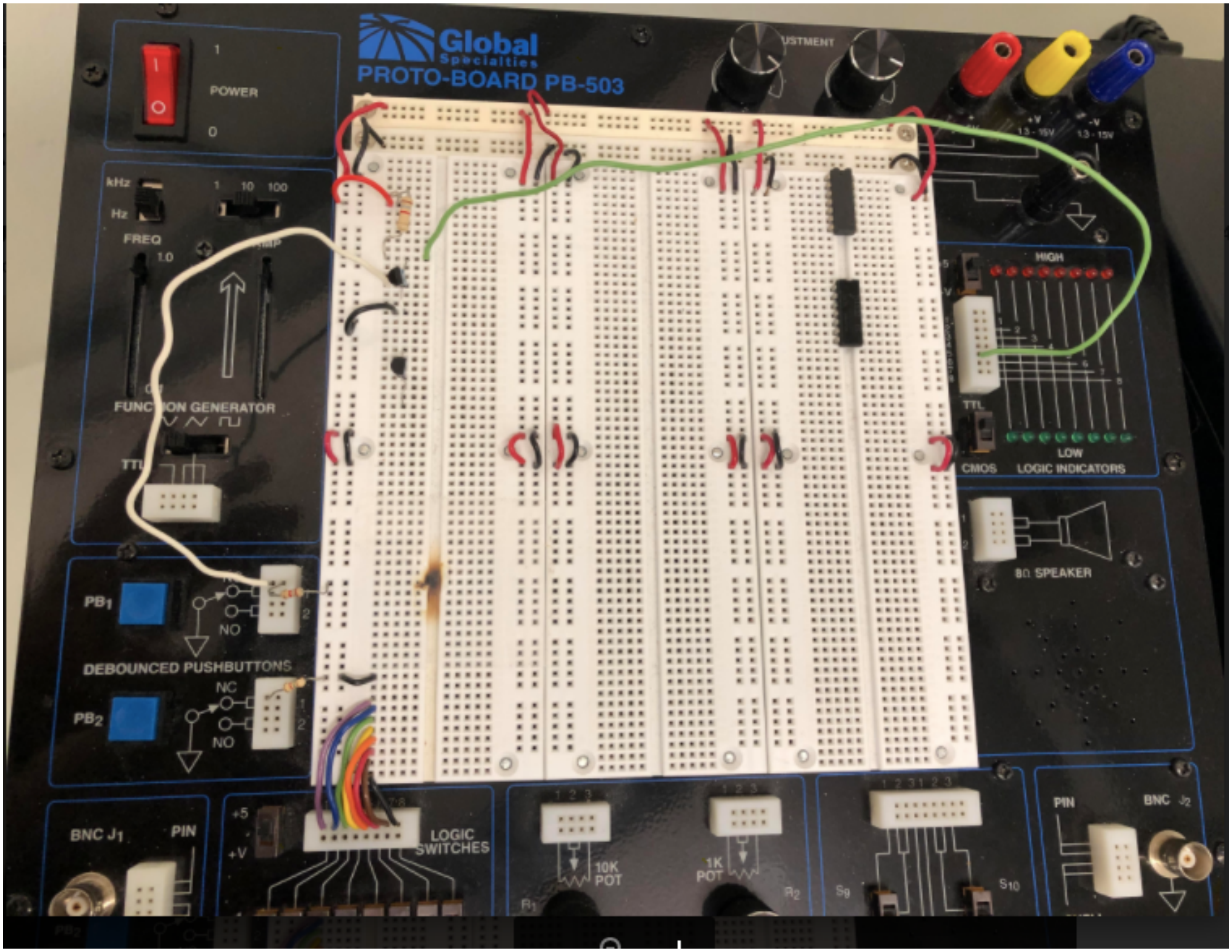


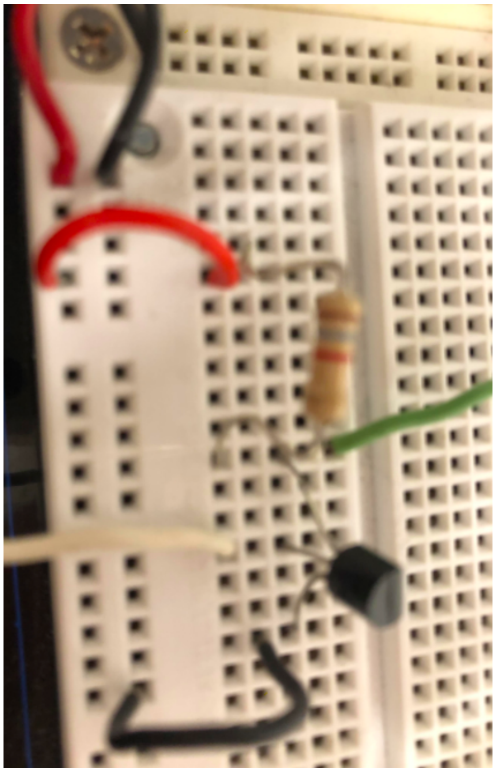
Top right pin is often connected to the positive terminal of the power supply (called **Vcc**, and assumed to be +5 Volts for our experiments).

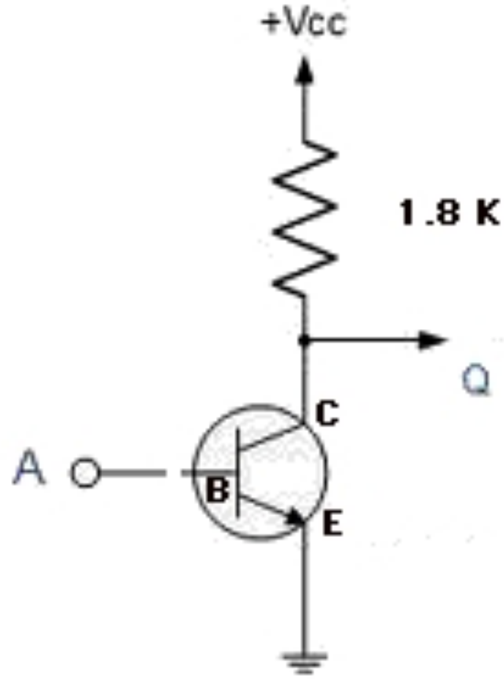
The chip will not work if it is not connected to power and ground!

Protoboard for building circuits

A protoboard is a tool to create prototype circuits. It contains a built-in power supply, switches to supply inputs to circuits, Logic Indicators to display outputs of circuits, and an array of holes/tie points in which components and wires can easily be inserted to connect circuits:







Transistor circuit for a NOT gate

Circuit Simulation/LogicWorks