

# **Basic Electronics and Digital Logic Computer Science 240**

## **Laboratory 1**

- **Administrivia**
- **Lab Environment**
- **Basic Electronics (Ohm's law, transistors, logic gates)**
- **Truth Tables, Sum-of-Products**

## Lab Environment

- All lab exercises and reports will be *Google Docs*, and should be shared with lab partner and the instructor
- Bring a laptop to lab if you have it (helpful to use a second computer for the lab report)
- From lab machine booted to Linux, you can enter Linux commands using a **terminal / shell**
- You can also use a terminal from either Mac ( *Terminal* ) or PC ( *PuTTY* ) to open a remote connection to a Linux machine for command-line entry

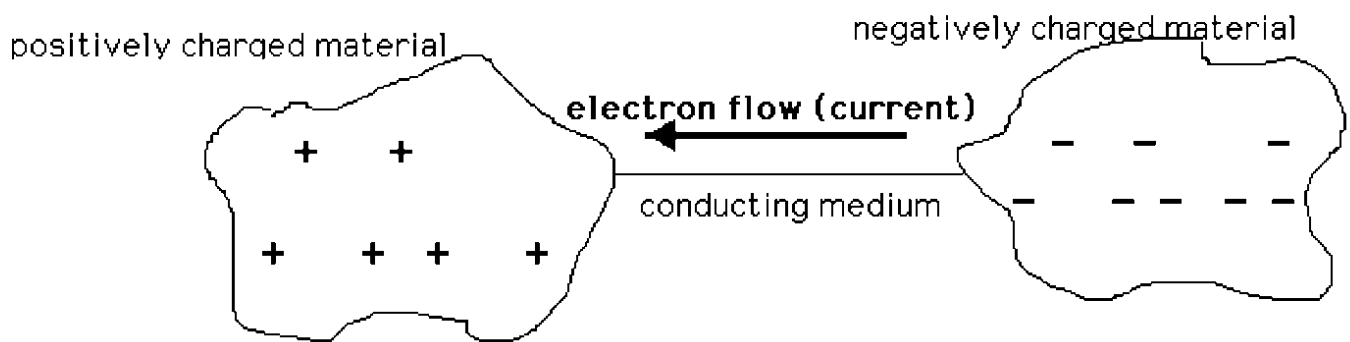
NOTE: for some exercises and assignments, you will be required to use the lab machines to compile and run your programs

## Basic Concepts of Electricity

Electricity = **the movement of electrons** in a material

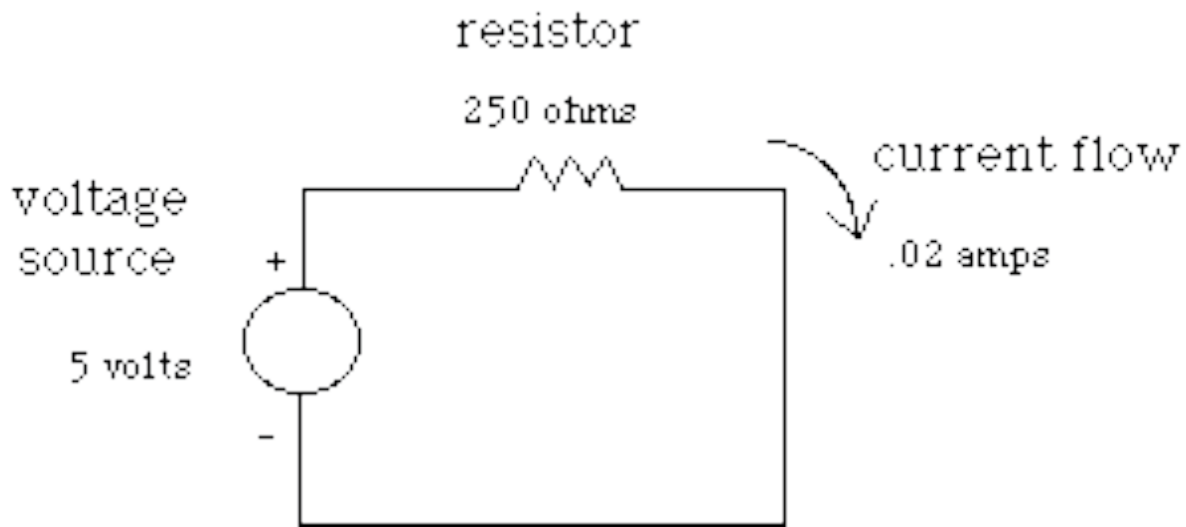
Materials tend to have a net negative or positive charge

Difference of charge between two points = **potential difference (V)**



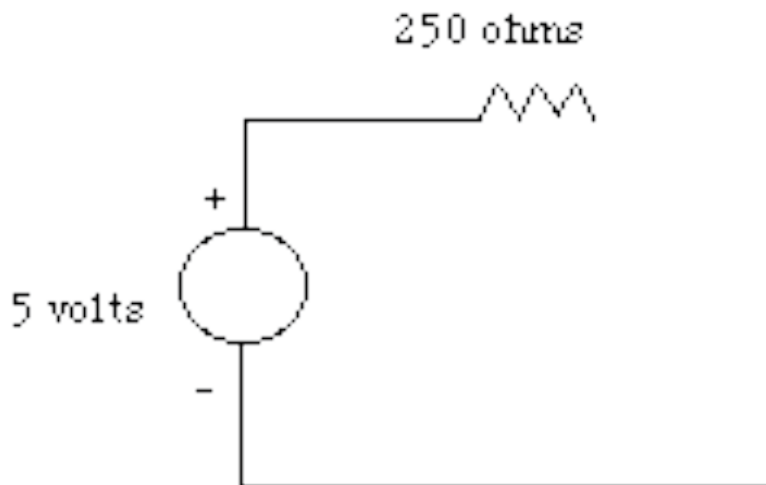
Rate at which electrons flow through = **current (A)**.

Ease of conduction, or current flow = **resistance ( $\Omega$ )**

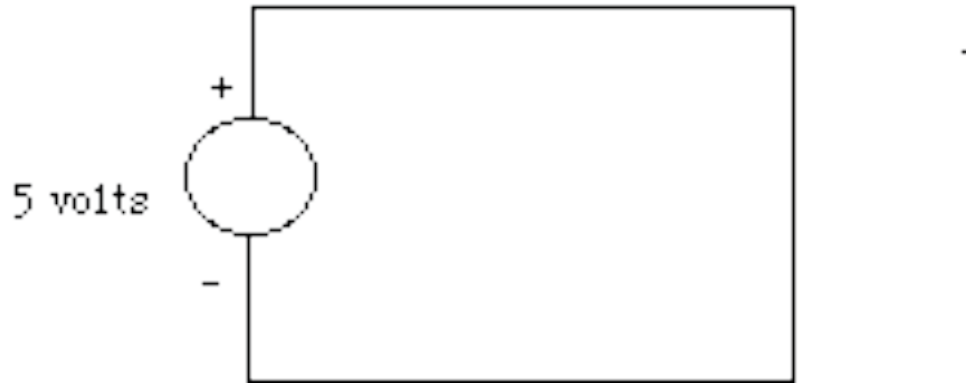


**Ohm's Law,  $V = IR$ .**

Open circuit = no current

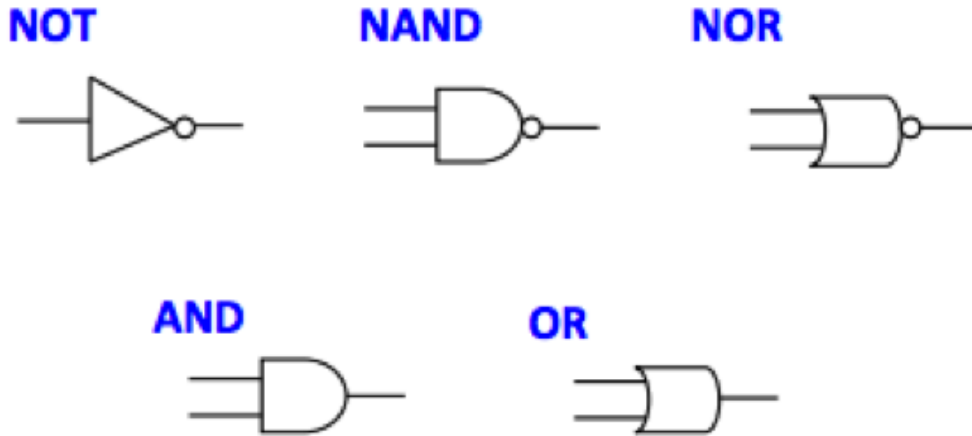


Short circuit = infinite current, since  $V/0 = \text{infinite current}$ :



Infinite current swiftly results in the destruction of the circuit!

## Basic Gate Symbols



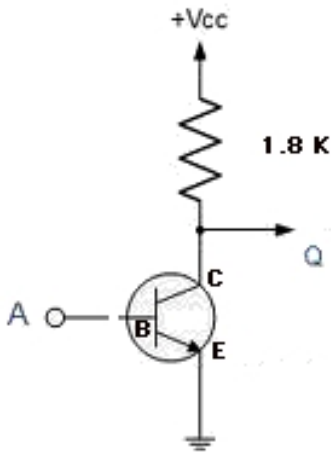
## Notation and Truth Tables for Basic Logic Gates

NOT $F = A'$	NAND $F = (AB)'$	NOR $F = (A+B)'$	AND $F = AB$	OR $F = A + B$
<u>A</u> <u>F</u>	<u>A</u> <u>B</u> <u>F</u>	<u>A</u> <u>B</u> <u>F</u>	<u>A</u> <u>B</u> <u>F</u>	<u>A</u> <u>B</u> <u>F</u>
0   1	0 0   1	0 0   1	0 0   0	0 0   0
1   0	0 1   1	0 1   0	0 1   0	0 1   1
	1 0   1	1 0   0	1 0   0	1 0   1
	1 1   0	1 1   0	1 1   1	1 1   1

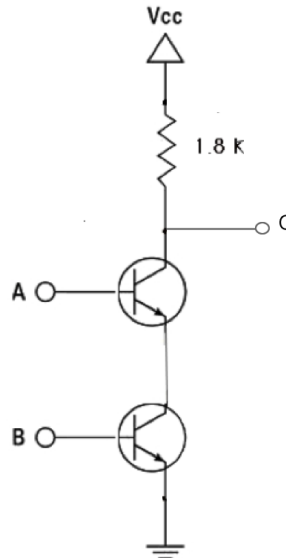
## Basic Gates are built using Transistors

You have seen the circuits for NOT and NAND in lecture:

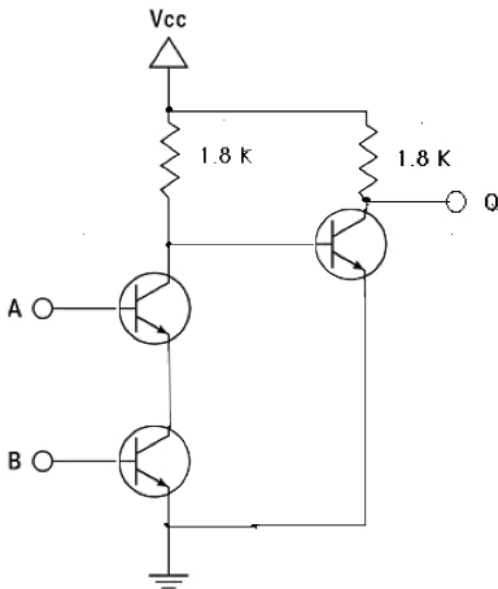
**NOT** – 1 transistor



**NAND** - 2 transistors

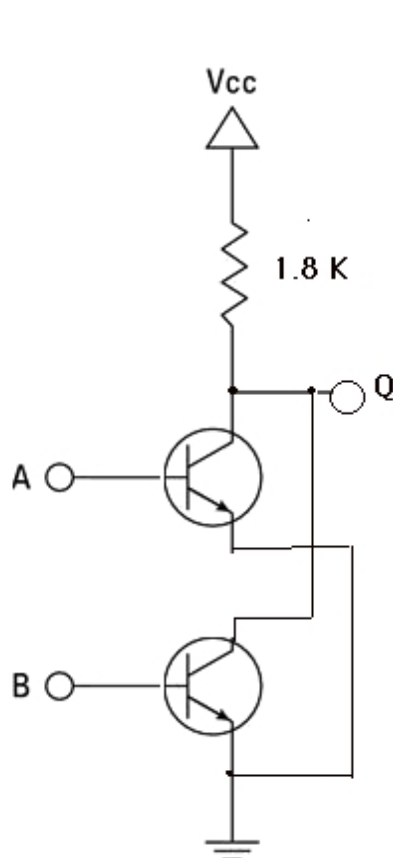


**AND** – uses 3 transistors (send the output of a NAND through another transistor acting as a NOT gate to complement the result):

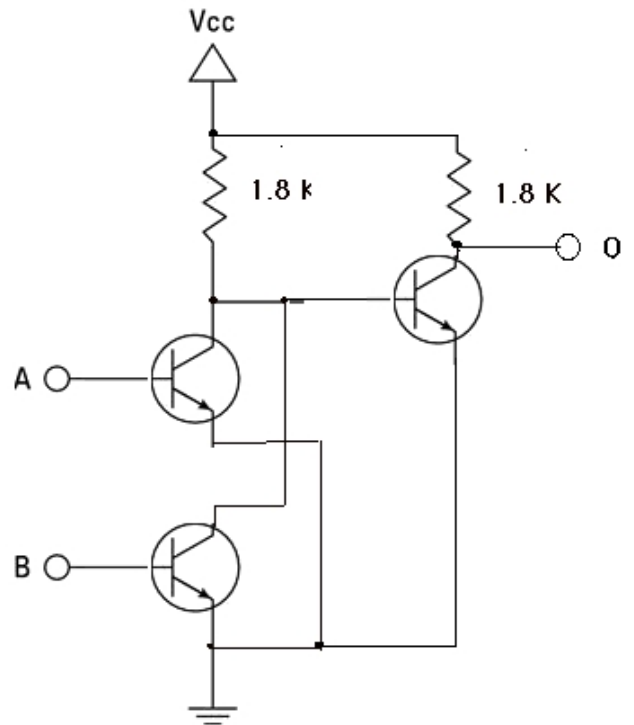


Similarly, these are the transistor circuits for a NOR and OR gate:

**NOR – 2**  
transistors



**OR – 3** transistors





## Truth Tables and Sum-of-Products

**Truth tables** specify the output for all the given input combinations of a function.

An input combination can be expressed by ANDing together the inputs (each input or its' complement is used in the expression, depending upon which combination is being expressed)

A function can then be expressed as a **sum-of-products** by ORing together the input combinations which make the function true.

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

$$F = A'B' + A'B$$

$$Q = A' + A'B + A'B'$$

F and Q are equivalent (produce the same function) when they have the same truth table.

When there is an equivalent circuit that uses fewer gates, transistors, or chips, it is preferable to use that circuit in the design

