MATH 8B [2024 NOV 3]
HANDOUT 7 : LOGIC 2 : GATES AND CIRCUITS

REMINDER: BASIC LOGIC OPERATIONS AND LAWS

- **NOT** (for example, $\text{NOT } A$): opposite of A: true if A is false, and false if A is true. Commonly denoted by $\neg A$ or (in computer science) $!A$.
- **AND** (for example $A \text{ AND } B$): true if both A, B are true, and false if at least one false. Commonly denoted by $A \wedge B$
- **OR** (for example $A \text{ OR } B$): true if at least one of A, B is true, and false otherwise. Sometimes also called "inclusive or" to distinguish it from the "exclusive or" described in problem 4 below.
  Commonly denoted by $A \vee B$.
- **XOR** (for example, $A \text{ XOR } B$): exclusive or; true if exactly one of A, B is true and false otherwise
- **NAND**: not and: $A \text{ NAND } B = \neg(A \text{ AND } B)$. True if at least one of A, B is false; false if both A, B are true.
- **NOR**: not or:$A \text{ NOR } B = \neg(A \text{ OR } B)$. True if both of A, B are false; false if at least one of A, B is true.

Some logic laws:

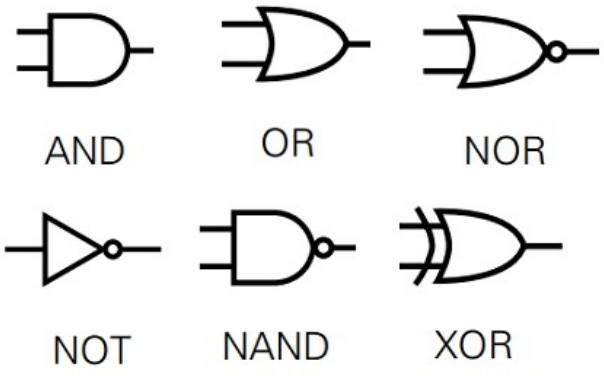
- **Double negation:**
$$\neg(\neg A) \iff A$$
- **De Morgan's law:**
$$\neg(A \wedge B) \iff (\neg A) \vee (\neg B)$$
$$\neg(A \vee B) \iff (\neg A) \wedge (\neg B)$$

LOGIC GATES IN ELECTRONICS

In electronics, it is common to have digital signals which take two values: LOW and HIGH (e.g., in certain microcontrollers, voltage can range between 0–5V, and anything over 2.5V is considered HIGH, and anything below, LOW). These two values can also be considered as two binary digits: LOW$= 0$, HIGH$= 1$, or as boolean values: LOW$= 0 =$false, HIGH$= 1 =$true.

The basis of all modern computers are "logic gates", chips that take two (or more) such inputs and produce an output described by some truth table. (These chips contain transistors and diodes, but this is irrelevant for us). For example, below is a typical such chip, containing four AND gates, each with two inputs:

In electronics, there are standard notations for different kinds of gates (AND, OR, NAND...):



Combining such simple gates, one can create more complicated ones — and use that to create circuits which take as input a collection of binary digits and produce as output some function such as sum or product of inputs (interpreting n binary inputs as an n-digit binary number).

There is a number of online simulators that allow you to create and test such circuits virtually; in particular, we will use `http://logic.ly/demo` (Demo version is enough).

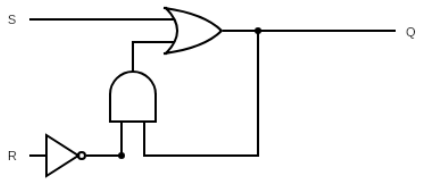
PROBLEMS

1. Similarly to what we did with NAND, show that any logic operation can be expressed using NOR, that is defined as: $A \text{ NOR } B = \text{NOT}(A \text{ OR } B)$. Use the online circuit simulator to show that your formulas indeed work.
2. **Half adder:** Adding two one-digit binary numbers is an operation which has two inputs, A and B, and produces two outputs, X_0, X_1, which are just two digits of the sum $A + B$ written in binary. E.g., $1_b + 1_b = 10_b$ (we use subscript b to indicate that this is a binary number, not a decimal one); thus, when $A = 1, B = 1$, we have $X_0 = 0$ (last digit of sum) and $X_1 = 1$.
   (a) Write a table of values for this operation, listing for each possible combination of A, B the values of outputs X_0, X_1.
   (b) Can you get X_0 from A, B by using one of the standard logic gates listed above?
   (c) Same question for X_1
   (d) Construct a circuit with two inputs and two outputs, consisting of the basic logic gates, which implements this addition operation. [This is commonly known as half-adder.] Test your circuit in `http://logic.ly`.

*3. **Full adder:** Can you construct a circuit which does binary addition of three inputs A, B, C (each input being a 1-digit binary number)? This circuit is known as full adder.
   Hint: use the half-adder you constructed in the previous problem as one of the building blocks — you can use it to first add A and B. You only need one half-adder!
   Test your final circuit in `http://logic.ly`.
4. **Ripple-carry adder:** Using the full adder constructed in the previous problem as a building block, construct a circuit which adds two 3-digit binary numbers. Your

circuit must have six inputs A_0, A_1, A_2, B_0, B_1, B_2 (representing digits of A, B respectively) and four outputs $X_0, \ldots, X_3$, representing digits of the sum.
This is known as ripple-carry adder (carry because it implements usual addition with carry digits; word "ripple" can be ignored for now).

5. Consider the circuit shown below.
Can you make a table of values, describing for each combination of values of the inputs S, R the value of the output? [Hint: there is a catch there. . .]
Test it on `http://logic.ly`



6. Show that for any value of A, expression $A \vee (\neg A)$ is true (such expressions, which are true for all values of variables involved, are called *tautologies*). This particular tautology is sometimes called **"law of excluded middle"** (meaning there is no middle ground — A must be true or false).
Similarly, show that $A \wedge (\neg A)$ is always false.

Boolean algebra practice

7. Remember the logical expression $A \Rightarrow B$ which is equivalent to $\neg A \vee B$. Show that if $A \vee B$ and $\neg B$ are true, we can conclude A. In other words, show the following expression is true:

$$(A \vee B) \wedge (\neg B) \Rightarrow A.$$

8. Show that if $A \Rightarrow B$ and $\neg B$ are true, we can conclude $\neg A$.

9. Show that if $A \Rightarrow B$ and A are both true, we can conclude B.

10. Just as we concluded that $A \Rightarrow B$ had an expression in simple logic gates as $\neg A \vee B$, write down the truth table for the expression $A \Leftrightarrow B$, and see if you can express it with $\neg$, $\wedge$, and $\vee$ gates. Which familiar gate/operation is $\neg(A \Leftrightarrow B)$?

11. Show that $(A \Rightarrow B) \wedge (B \Rightarrow A)$ is the same as $A \Leftrightarrow B$.