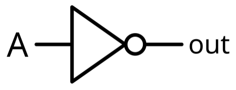
# Boolean Logic and Digital Circuits

When working with binary, a single bit is either TRUE (1), or FALSE (0). We can operate on the value of binary bits with logical operations. Three fundamental logical operations are AND, OR, and NOT.

## NOT

The simplest operation is NOT. Given a boolean variable A, where A is either 0 or 1, the NOT operation negates the current value of A. The digital circuit diagram for NOT looks like this:



If A is 0, then NOT A is 1

If A is 1, then NOT A is 0

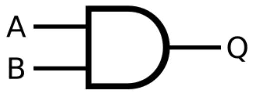
A nice way to represent this is with a truth table. The left-hand side is the input value, and the right-hand side is the output value.

| $A$ | $\neg A$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Notice the syntax for representing not. We use the expression $\neg A$ to say "NOT A".

## AND

Given to input bits A and B, the AND operation returns TRUE when A *and* B are both true. The digital circuit diagram for AND looks like this:

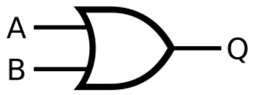


The truth table for AND is:

| $A$ | $B$ | $A \wedge B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## OR

Given input bits A and B, the OR operation returns TRUE when either A *or* B is true. The digital circuit diagram for OR looks like this:



The truth table for OR is:

| $A$ | $B$ | $A \vee B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

For each of the following Boolean expressions, write the truth table and draw the corresponding circuit diagram.

1. $A \wedge (A \vee B)$
2. $A \vee (A \wedge B)$
3. $(A \vee B) \wedge (\neg A \vee B)$
4. $A \vee \neg A$
5. $\neg(A \wedge B)$
6. $\neg A \vee \neg B$
7. $\neg(A \vee B)$
8. $A \wedge \neg B$
9. $\neg(A \wedge B)$
10. $\neg(A \wedge B \vee C)$
11. $\neg\neg(A \vee B)$
12. $A \wedge B \wedge C$
13. $(A \vee B) \wedge C$
14. $(A \vee B) \wedge \neg C$
15. $\neg(A \wedge B) \vee (B \wedge \neg C)$
16. $\neg(A \vee B) \vee (B \wedge \neg C)$
17. $(A \wedge \neg B) \vee (\neg A \wedge B)$
18. $A \wedge B$