DIGITAL DEVICES AND LOGIC CIRDUITS

Chapter 5

INTENDED LEARNING OUTCOMES

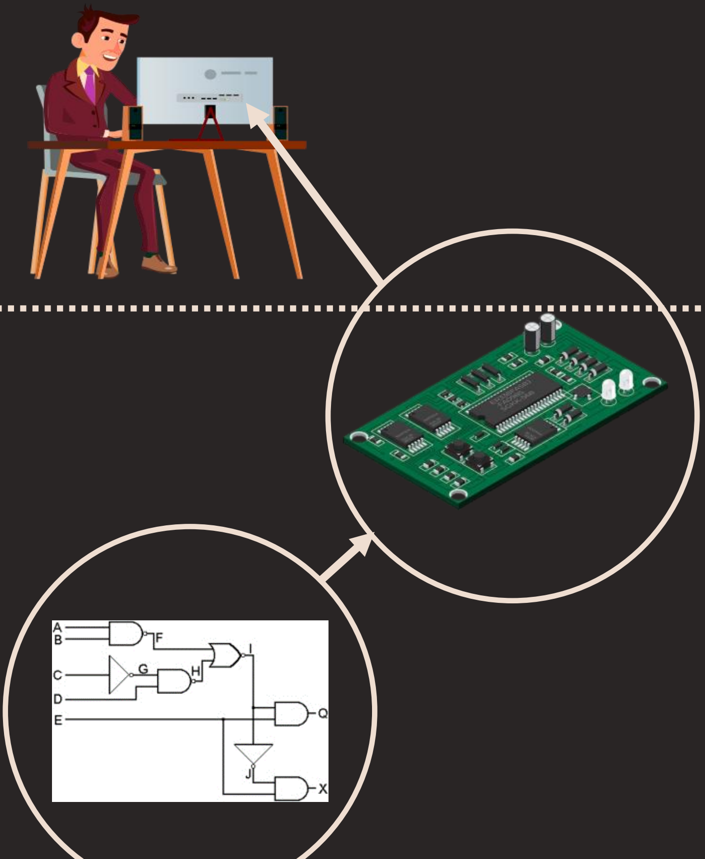
1	Explain the fundamental principles of digital logic systems
2	Understand the basic principles of Boolean algebra and its significance in digital logic design.
3	Apply Boolean algebra operations to simplify and analyze logical expressions.
4	Create and interpret truth tables for various logic gates and combinations.
5	Analyze and design simple combinational and sequential logic systems.
6	Explore the role of digital electronics in the design and function of embedded systems.

01

INTRODUCTION TO DIGITAL ELECTRONIC DEVICES

Chapter 5: Digital Devices and Logic Circuits

COMPUTER SYSTEM: LAYERS OF ABSTRACTION



SOFTWARE

Application Program

Language

Operating System

HARDWARE

Instruction Set Architecture (and I/O Interfaces)

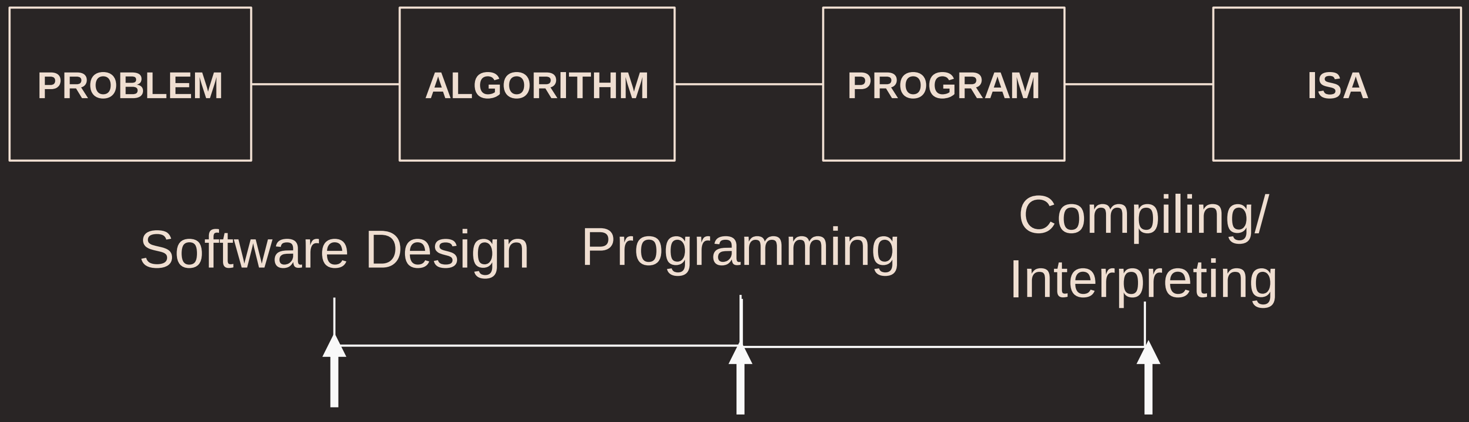
Microarchitecture

Logic Gates

Circuits

Devices

TRANSFORMATIONS BETWEEN LAYERS

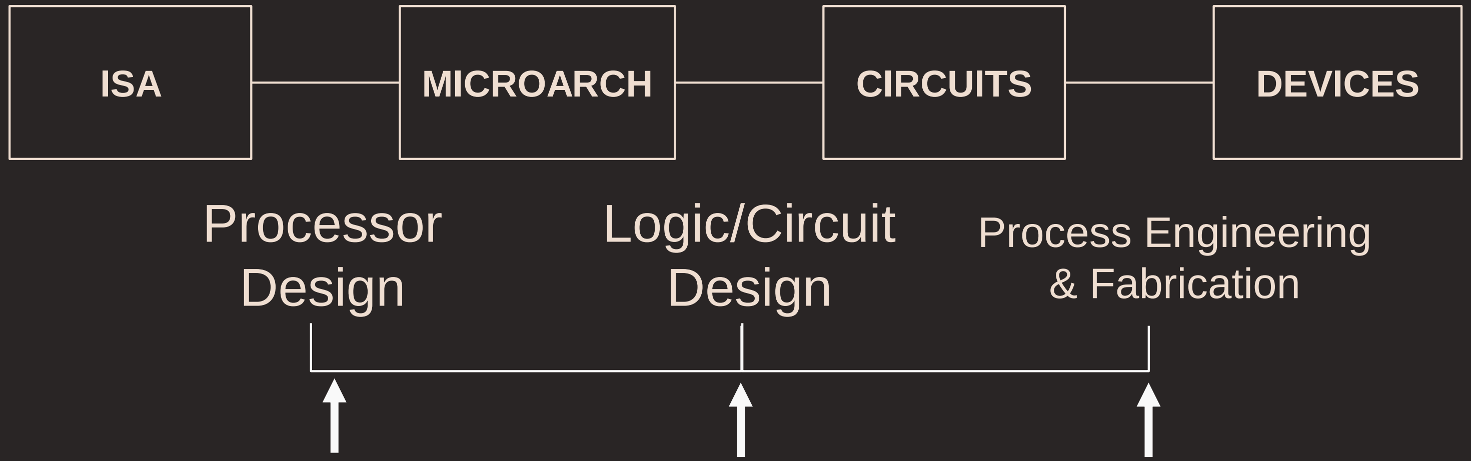


choose algorithms and data structures

use language to express design

convert language to machine instruction

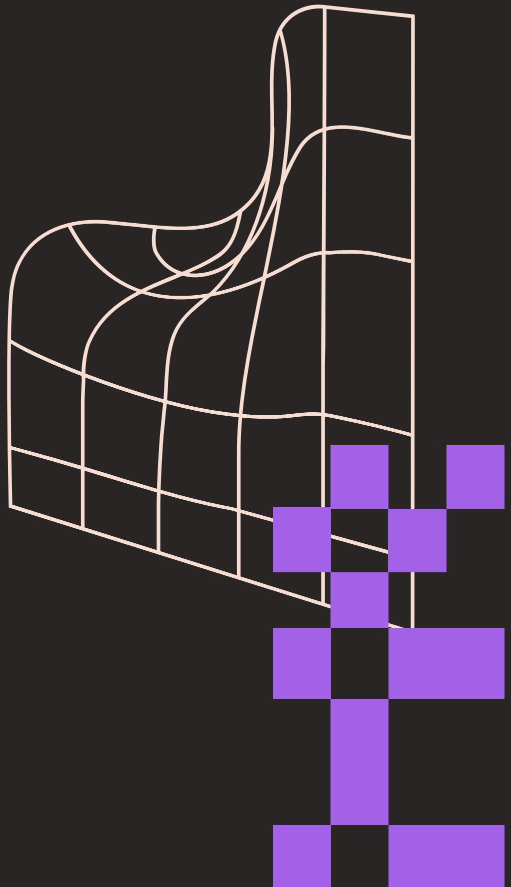
TRANSFORMATIONS BETWEEN LAYERS



Processor Design: choose structures to implement ISA

Logic/Circuit Design: gates and low-level circuits to implement components

Process Engineering & Fabrication: develop and manufacture lowest-level components

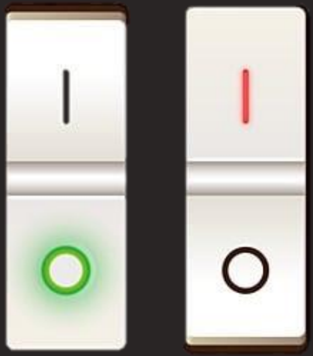


How do we represent data in a computer?

At the lowest level, a computer is an electronic machine.

- works by controlling the flow of electrons

How do we represent data in a computer?



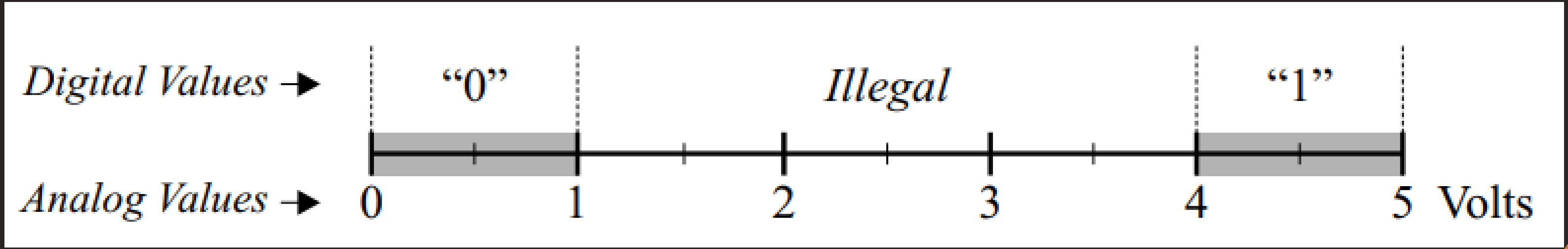
Easy to recognize two conditions:

1. presence of a voltage – the state is “1”
2. absence of a voltage – the state is “0”

COMPUTER IS A BINARY DIGITAL SYSTEM

Digital system
- finite number of symbols

Binary (base two) system
- has two states: 0 and 1



Basic unit of information is the binary digit, or **bit**

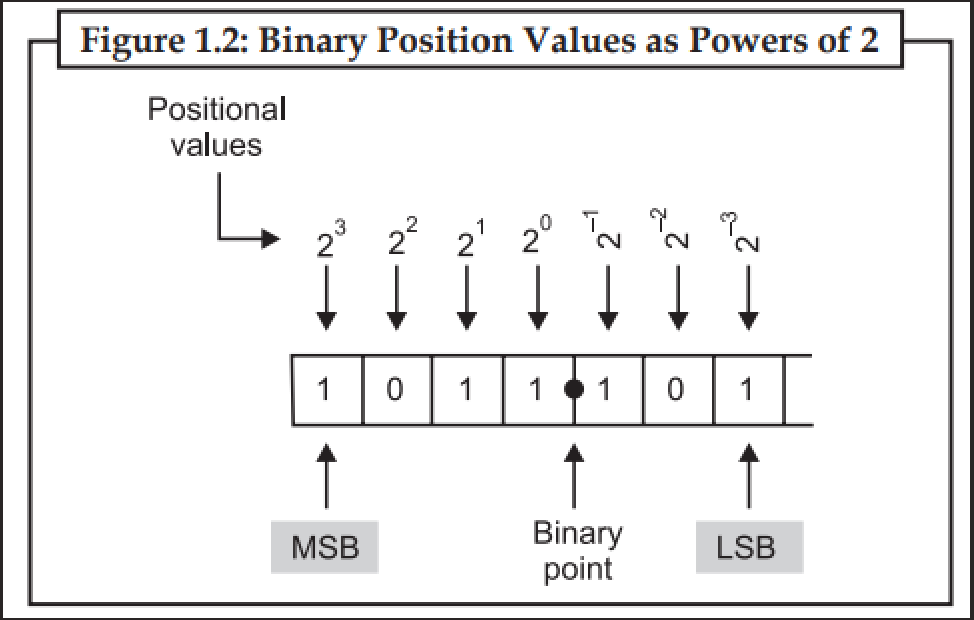
Values with more than two states require multiple bits.

– A collection of two bits has four possible states:
  00, 01, 10, 11

– A collection of three bits has eight possible states:

– A collection of **n** bits has $\mathbf{2}^{n}$ possible states.

REVIEW:



In the binary number shown in the figure, what is its equivalent decimal number?

LOGIC GATES

Logic gates are the basic components in digital electronics. They are used to create digital circuits and even complex integrated circuits. For example, complex integrated circuits may bring already a complete circuit ready to be used – microprocessors and microcontrollers are the best example – but inside them they were projected using several logic gates.

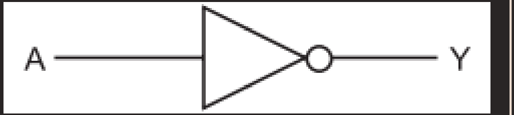
As discussed on chapter 1, digital electronics accept only two numbers, “0” and “1”. **Zero means a 0 V voltage**, while **“1” means 5 V voltages or 3.3 V voltages** on newer integrated circuits. *You can think “0” and “1” as a light bulb turned off or on or as a switch turned off or on*.

LOGIC GATES

A letter, also known as variable, represents a binary number. So “A” can be “0” or “1”. So, if A Notes
is connected to a switch, A will be “0” when the switch is turned off and “1” when the switch
is turned on. A line drawn right above the variable name means that the variable is inverted.

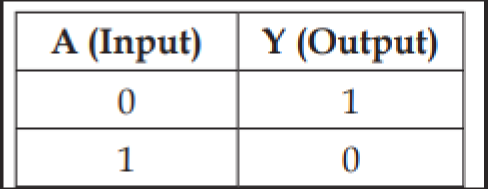
TYPES OF LOGIC GATES

The NOT (Inverter) Gate



On logic circuits, this symbol is short for **inverter.**

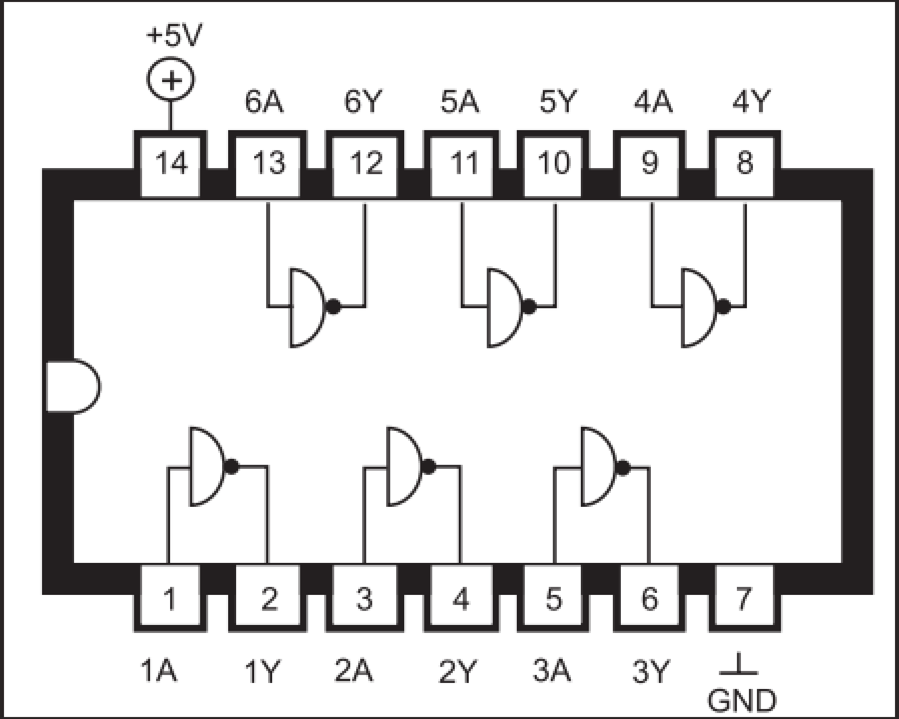
As the name implies, inverter will invert the number entered. If you enter ”0“, you will get a ”1“on its output, and if you enter a ”1“, you will get a ”0“ on its output. Inverter gate is also known as NOT



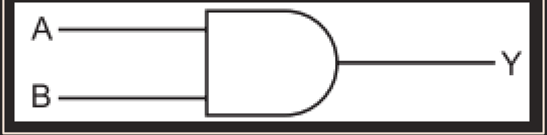
A (Input)	Y (Output)
0	1
1	0

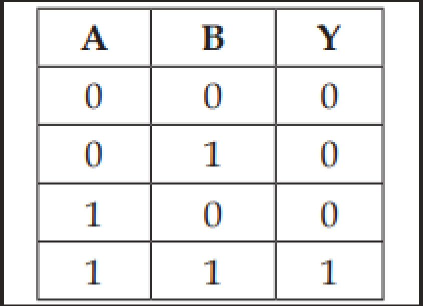
Truth Table

The Inverter: 74LS04 Integrated Circuit Provides Six Inverters



The AND Gate





A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

As its name implies, an AND logic gate performs an ”AND“ logic operation, which is a multiplication. It has at least two inputs. So, if A and B are its inputs, at the output we will find A * B (also represented as A • B). So, AND logic gate can be summarized by the formula Y = A* B (or Y = A * B).

Another way to understand AND logic gate: its output will only be at ”1“ when all its inputs are also at ”1“. Otherwise its output will be ”0“.

The AND Gate

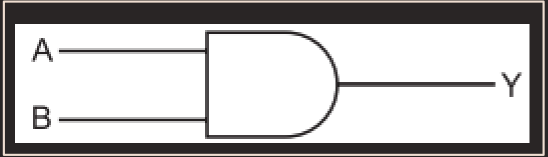


Figure a.

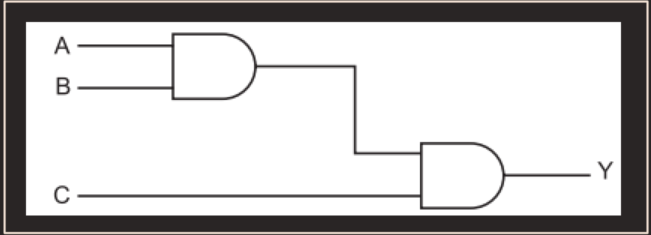
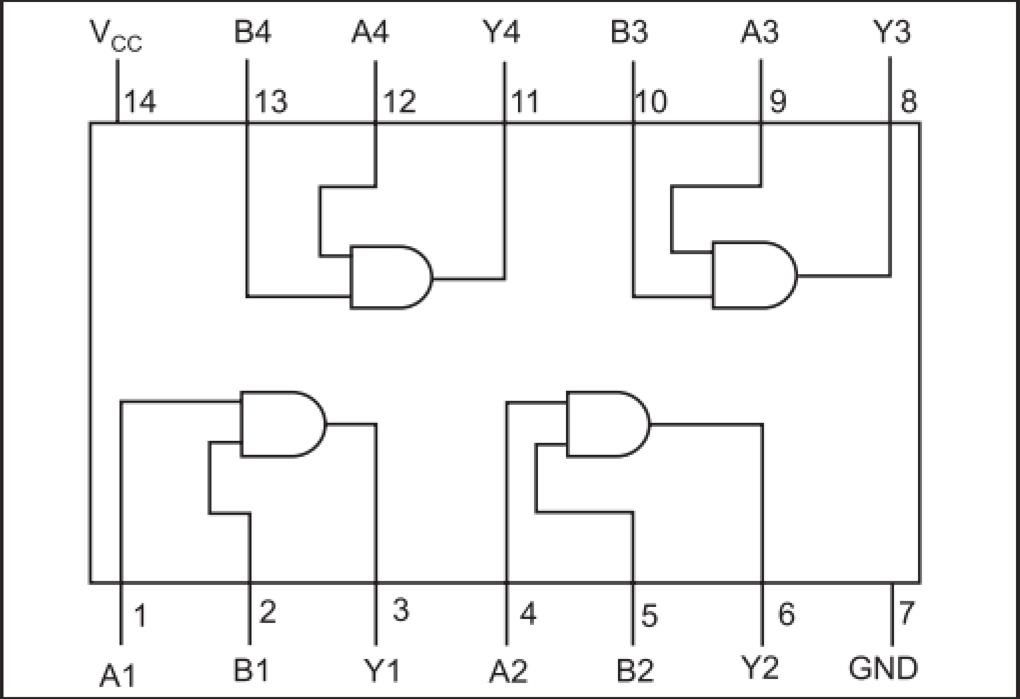
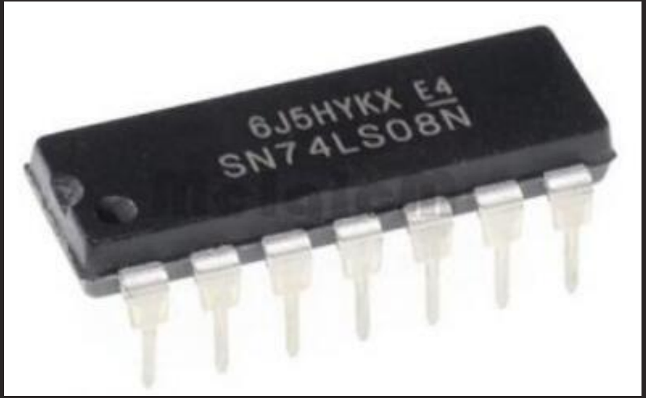


Figure b.

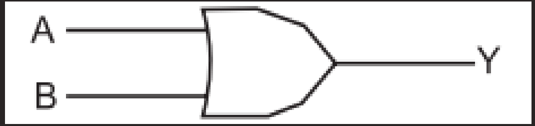
If you are projecting a circuit and need an AND logic gate with more inputs, you can go ahead and simple draw an AND logic gate like the one in Figure a. and put more inputs on it. But if you are working with integrated circuits with AND logic gates with fewer inputs that you need, you can expand the number of inputs by connecting them like shown in Figure b.

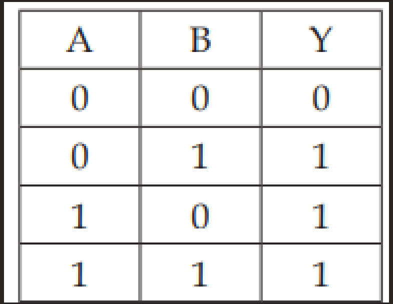
The AND Gate: 74LS08 Integrated Circuit Provides Four Two-Inputs





The OR Gate





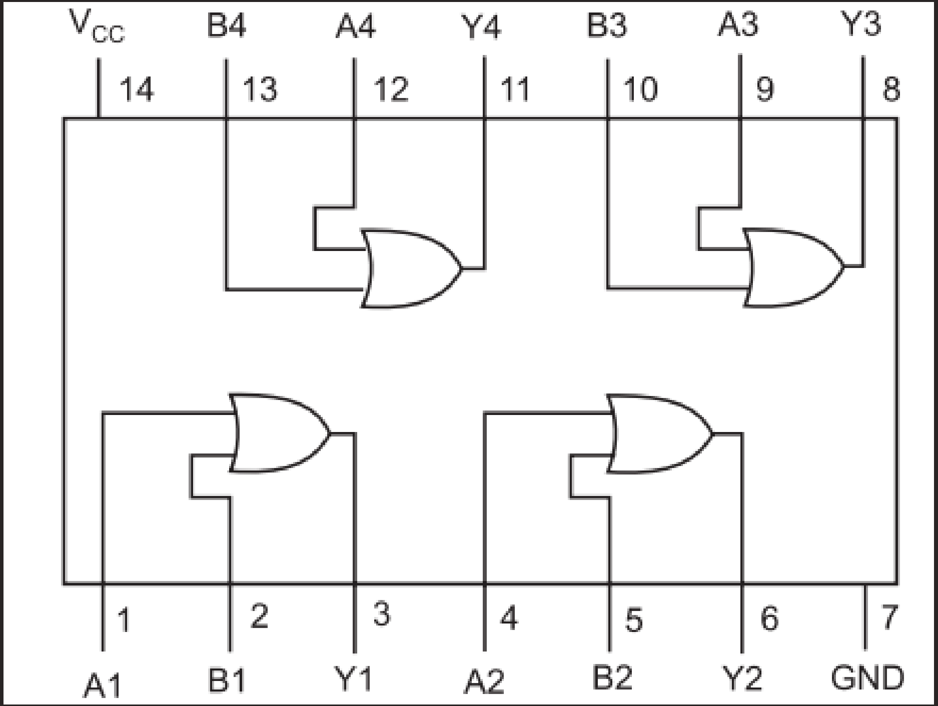
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

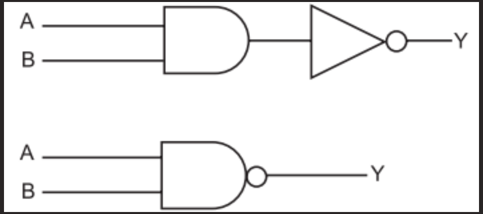
As its name implies, an OR logic gate performs an ”OR“ logic operation, which is an addition. It has at least two inputs. So, if A and B are its inputs, at the output we will find A + B. So, OR logic gate can be summarized by the formula Y = A + B.

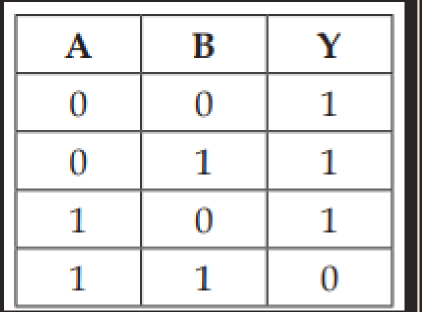
Another way to understand OR logic gate: its output will only be at ”0“ when all its inputs are also at ”0“. Otherwise its output will be ”1“

The OR Gate: 74LS32 Integrated Circuit Provides Four Two-Inputs



The NAND Gate





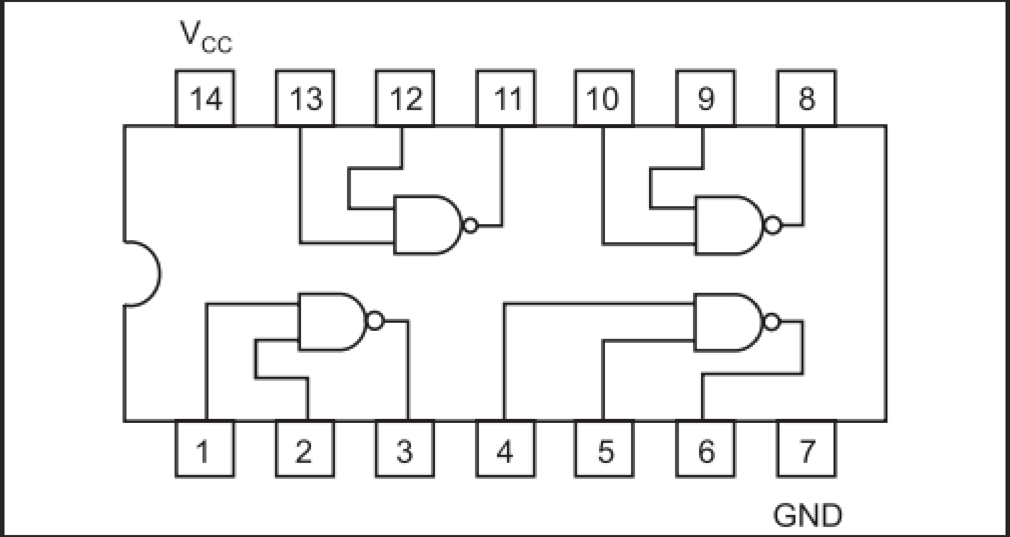
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

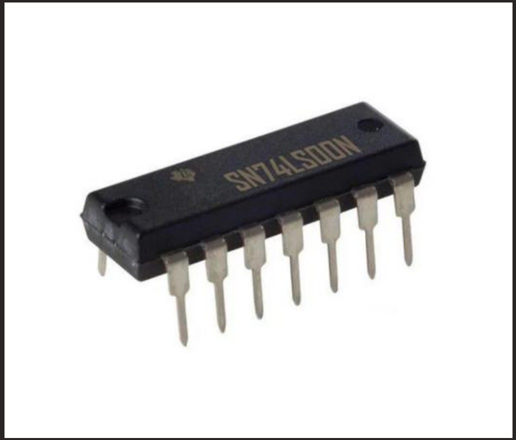
Truth Table

The ”N“ letter on NAND stands for NOT, meaning that NAND logic gate is an AND gate with an inverter attached. So, its output is the opposite from AND. Its symbol is the same of AND but with a ”o“ on its output, meaning that the output is inverted. You can build yourself a NAND gate by connecting an AND gate to an inverter.

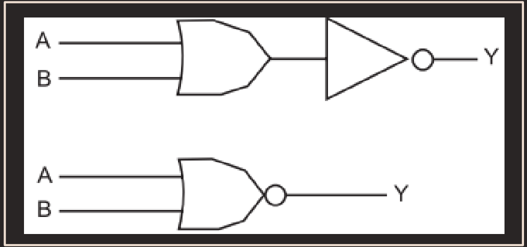
Another way to understand NAND logic gate: its output will only be at”0“ when all its inputs are also at ”1“. Otherwise its output will be ”1“

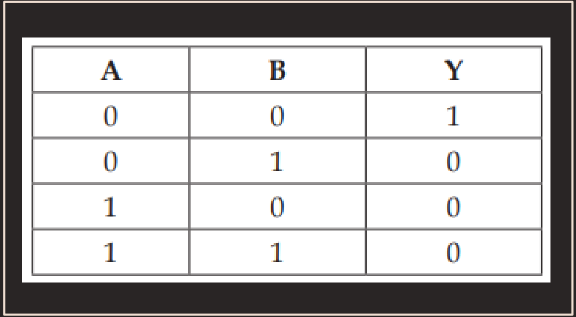
The NAND Gate: 74LS00 Integrated Circuit Provides Four Two-Inputs





The NOR Gate





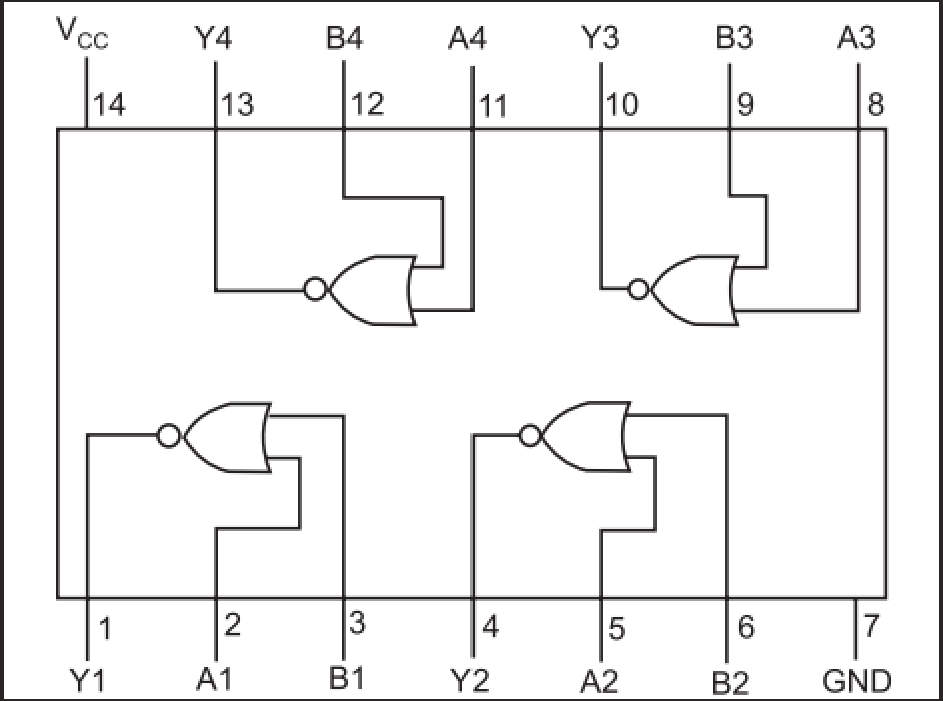
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table

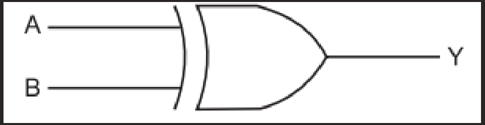
The ”N“ letter on NOR stands for NOT, meaning that NOR logic gate is an OR gate with an inverter attached. So, its output is the opposite from OR. Its symbol is the same of OR but with a ”o“ on its output, meaning that the output is inverted. You can build yourself a NOR gate by connecting an OR gate to an inverter.

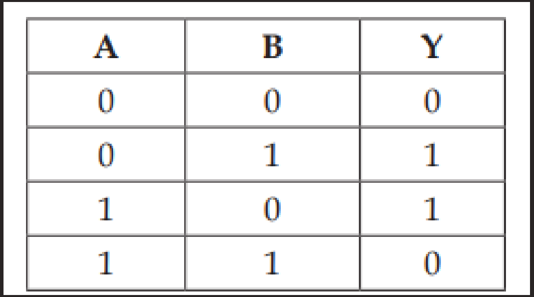
Another way to understand NOR logic gate: its output will only be at ”1“ when all its inputs are at ”0“. Otherwise its output will be ”0“.

The NOR Gate: 74LS02 Integrated Circuit Provides Four Two-Inputs



The XOR Gate





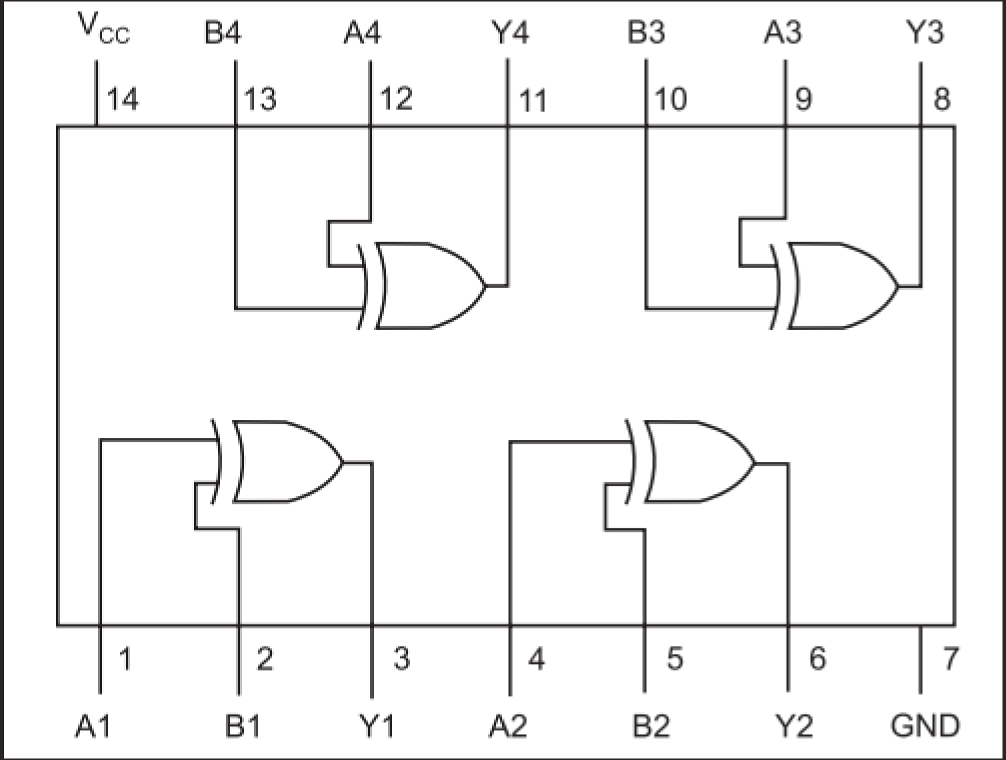
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

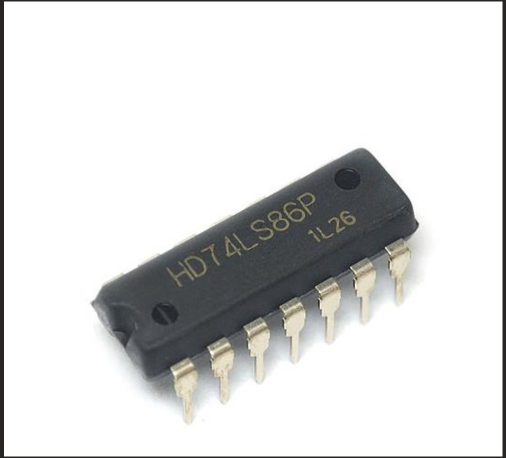
Truth Table

XOR stands for exclusive OR. XOR gate compares two values and if they are different its output will be “1.” XOR operation is represented by the symbol ⊕. So Y = A ⊕ B. You can see XOR logic gate symbol in Figure 2.6 and its truth table right below it.

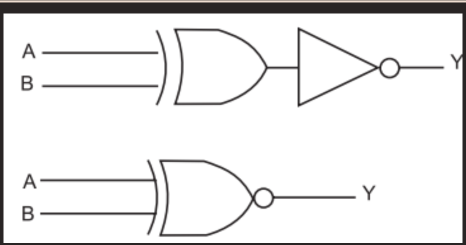
So its output will only be at “0” when all its inputs have the same value. Otherwise its output will be “1.”

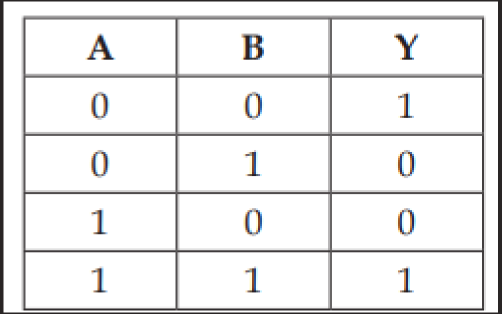
The XOR Gate: 74LS86 Integrated Circuit Provides Four Two-Inputs





The XNOR Gate





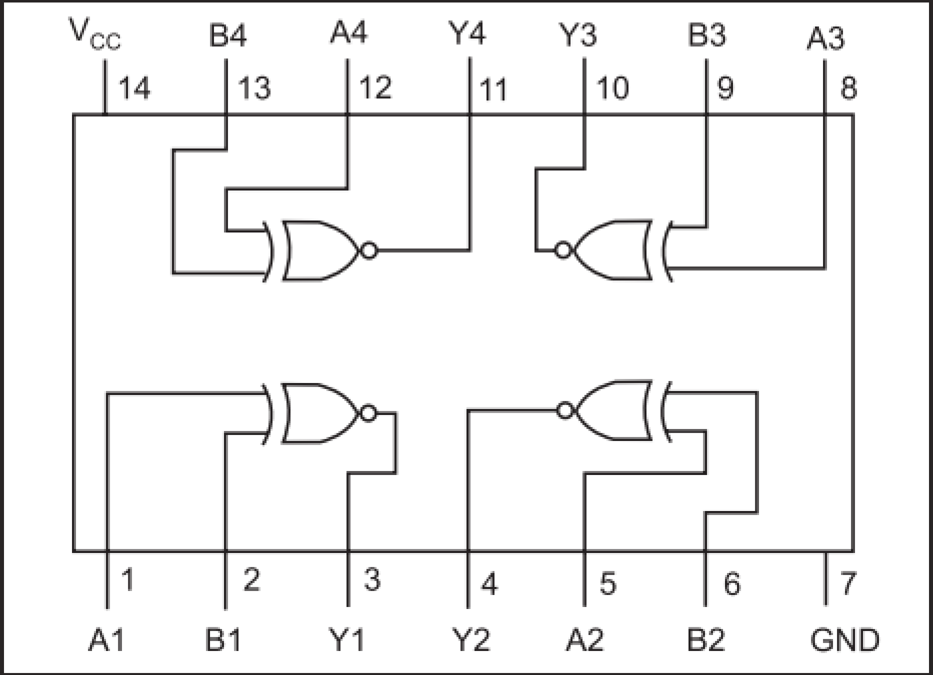
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

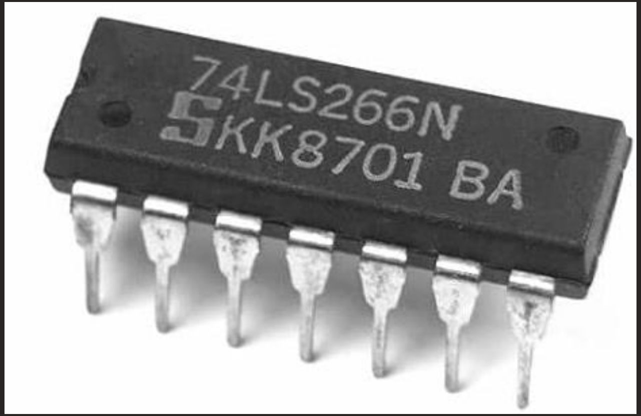
Truth Table

XNOR stands for exclusive NOR and is an XOR gate with its output inverted. So, its output is at ”1“ when the inputs have the same value and ”0“ when they are different. XNOR operation is represented by the symbol (•). So Y = A (•) B.

So its output will only be at ”1“ when all its inputs have the same value. Otherwise its output will be ”0“.

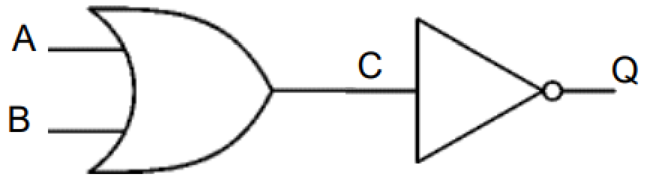
The XNOR Gate: 74LS66 Integrated Circuit Provides Four Two-Inputs





PRACTICE EXERCISES:

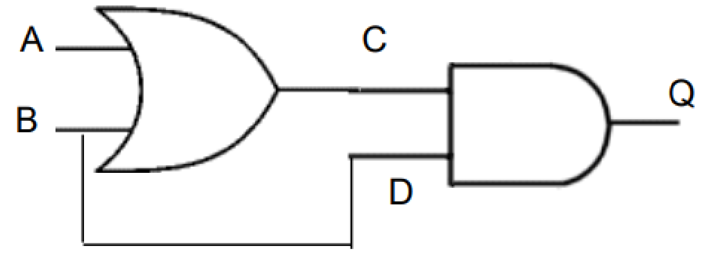
Complete the truth table



A	B	C	Q
0	0		
0	1		
1	0		
1	1		

PRACTICE EXERCISES:

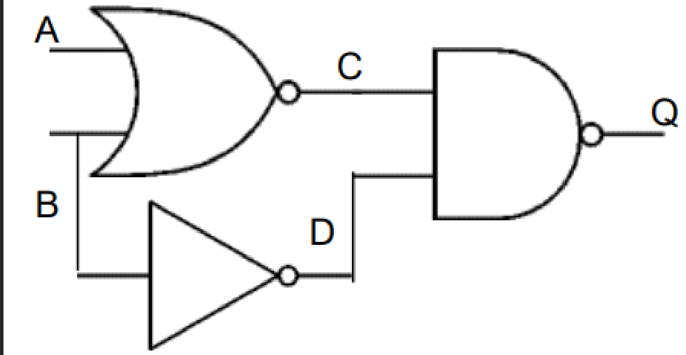
Complete the truth table



A	B	C	D	Q
0	0			
0	1			
1	0			
1	1			

PRACTICE EXERCISES:

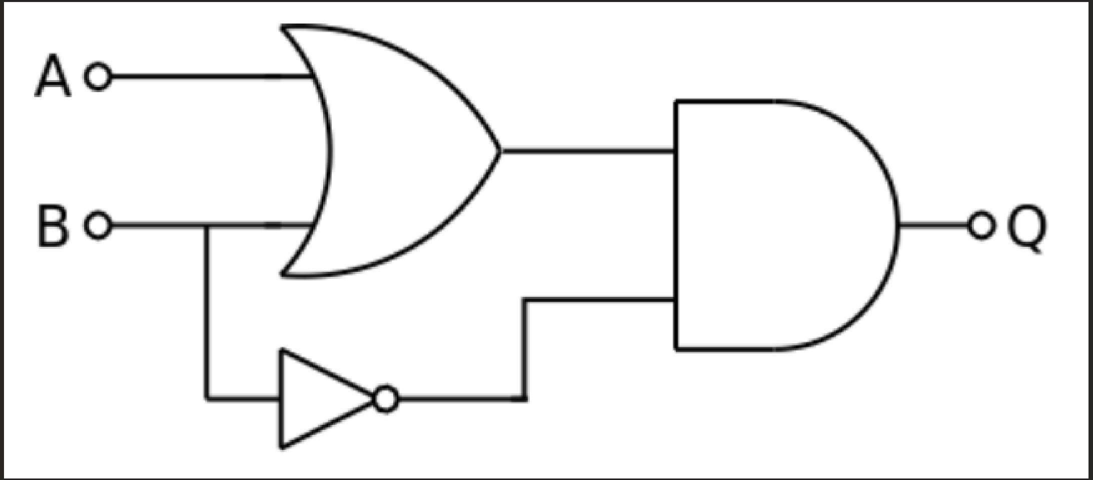
Complete the truth table



A	B	C	D	Q
	0			
	1			
	0			
	1			

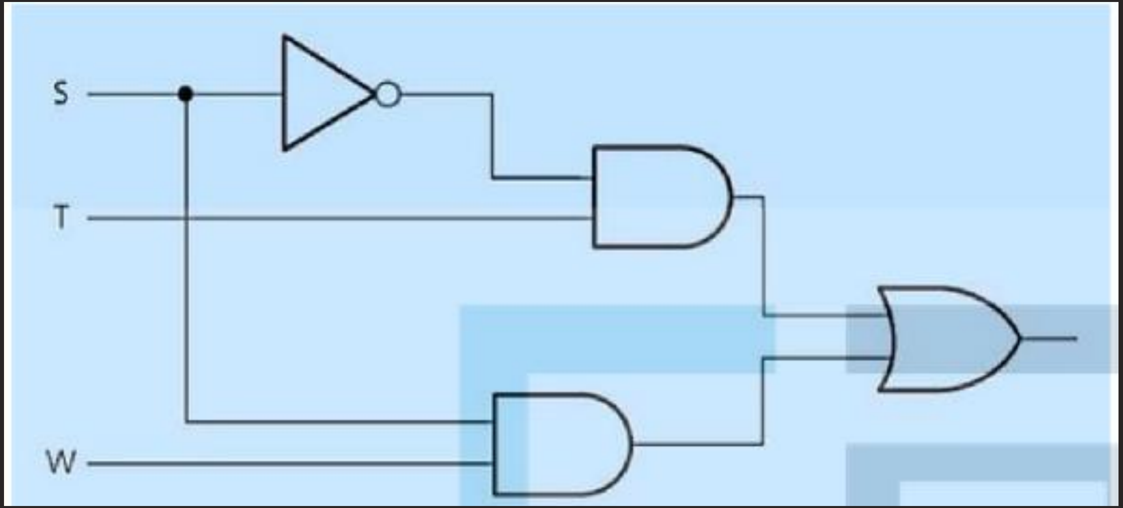
PRACTICE EXERCISES:

Produce a truth table for the below circuit.



PRACTICE EXERCISES:

Produce a truth table for the below circuit.



BOOLEAN ALGEBRA

Boolean Algebra

Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.

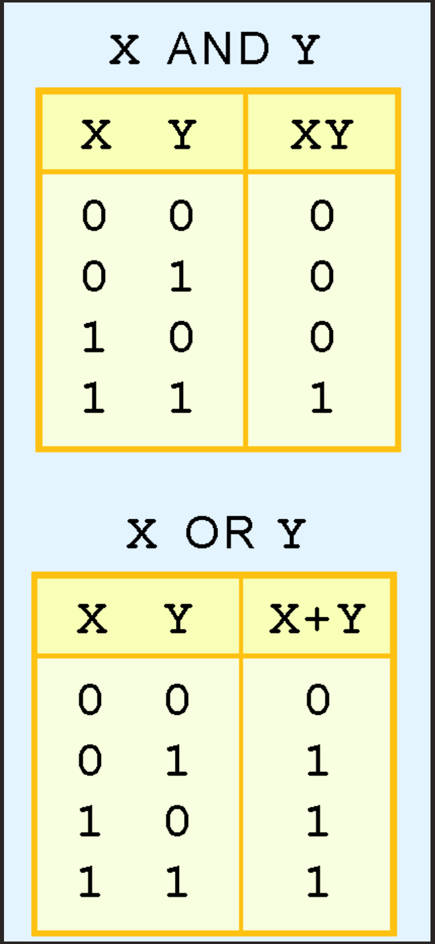
- In formal logic, these values are “true” and “false.”
- In digital systems, these values are “on” and “off,” 1 and 0, or “high” and “low.”

Boolean expressions are created by performing operations on Boolean variables.

- Common Boolean operators include AND, OR, and NOT.

Boolean Algebra

- A Boolean operator can be completely described using a truth table.
- The truth table for the Boolean operators AND and OR are shown at the right.
- The AND operator is also known as a Boolean product. The OR operator is the Boolean sum.



X AND Y

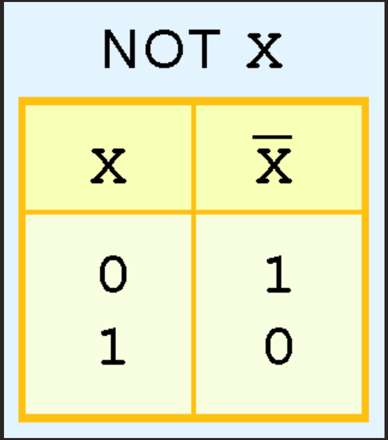
X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Algebra

- The truth table for the Boolean NOT operator is shown at the right.
- The NOT operation is most often designated by an overbar. It is sometimes indicated by a prime mark (‘) or an “elbow” (¬).


NOT x

x	$\overline{x}$
0	1
1	0

Boolean Algebra

- A Boolean function has:
  - At least one Boolean variable,
  - At least one Boolean operator, and
  - At least one input from the set {0,1}.
- It produces an output that is also a member of the set {0,1}.

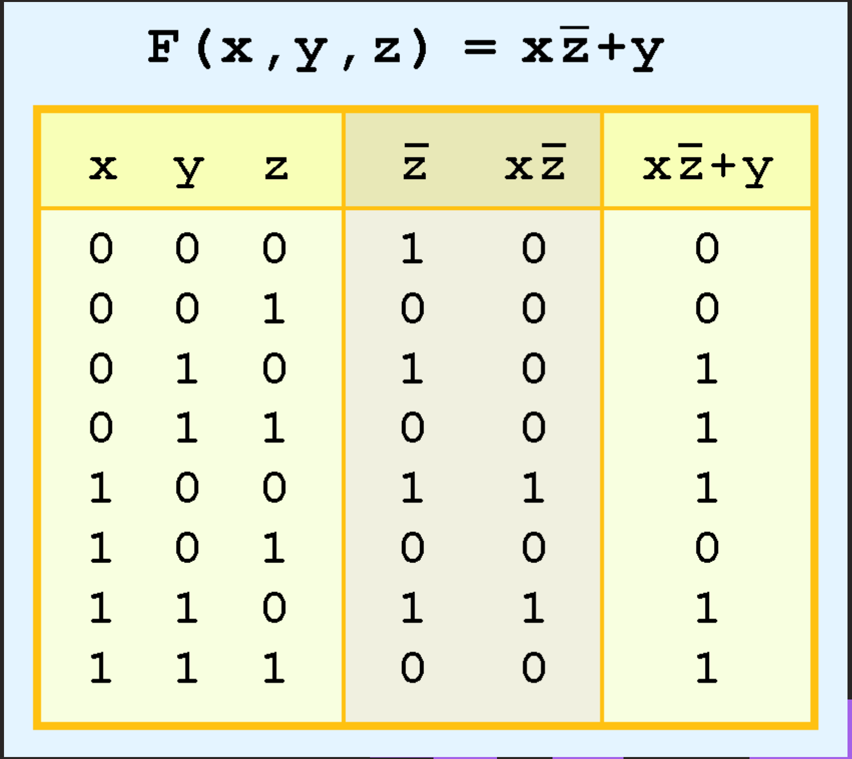
Boolean Algebra

- The truth table for the Boolean function:

$$F(x,y,z) = x\bar{z}+y$$

is shown at the right.

- To make evaluation of the Boolean function easier, the truth table contains extra (shaded) columns to hold evaluations of subparts of the function.

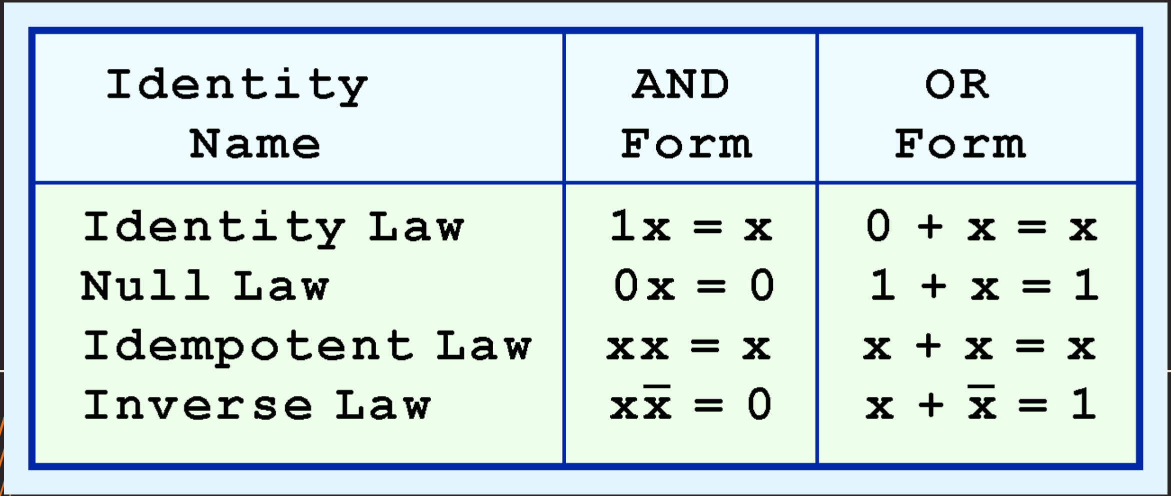


$$F(x,y,z) = x\bar{z}+y$$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Boolean Algebra

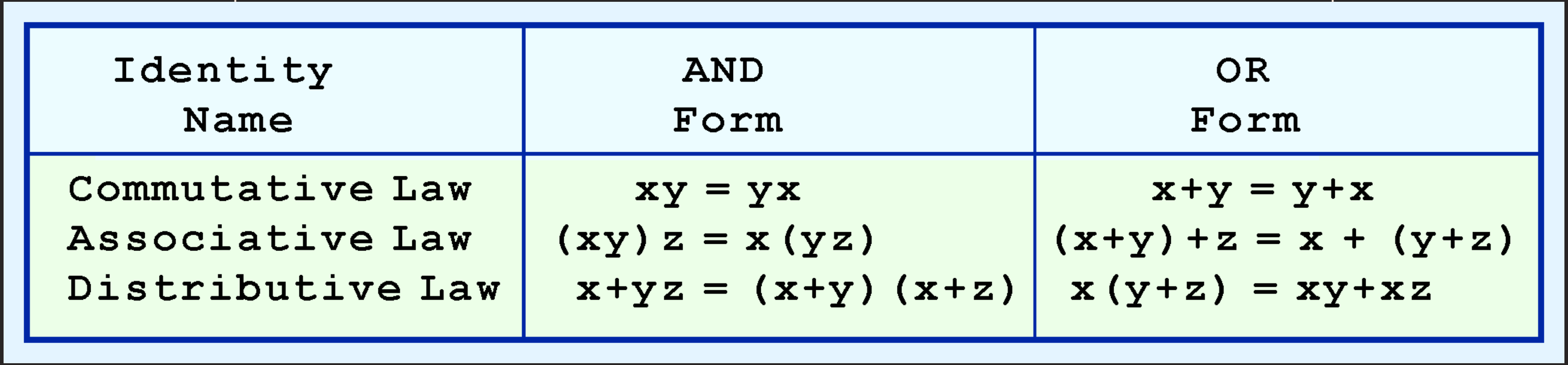
Most Boolean identities have an AND (product) form as well as an OR (sum) form. We give our identities using both forms. Our first group is rather intuitive:



Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0 + x = x$
Null Law	$0x = 0$	$1 + x = 1$
Idempotent Law	$xx = x$	$x + x = x$
Inverse Law	$x\overline{x} = 0$	$x + \overline{x} = 1$

Boolean Algebra

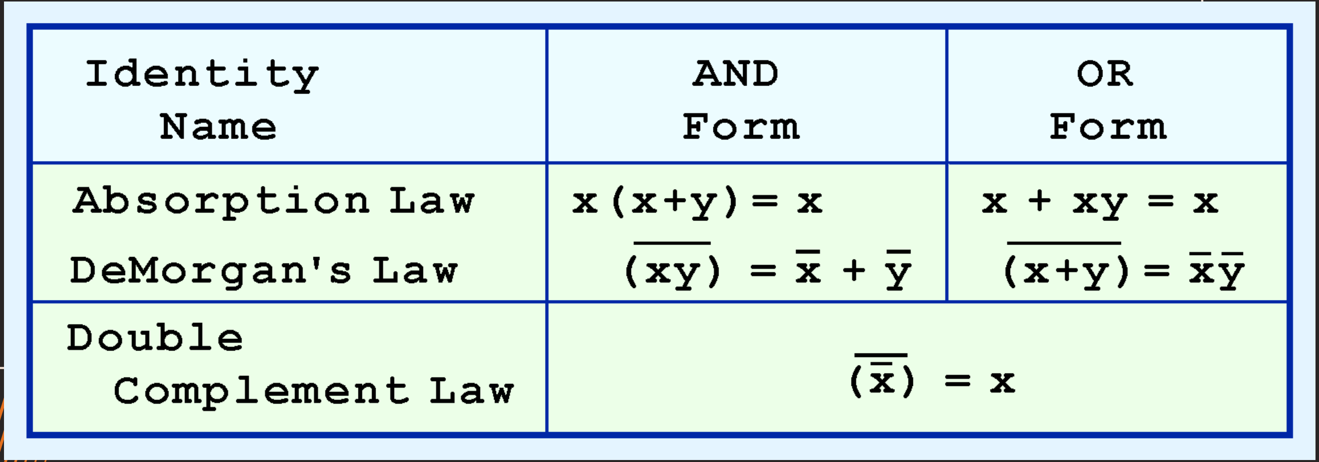
Our second group of Boolean identities should be familiar to you from your study of algebra:



Identity Name	AND Form	OR Form
Commutative Law	xy = yx	x+y = y+x
Associative Law	(xy)z = x(yz)	(x+y)+z = x + (y+z)
Distributive Law	x+yz = (x+y)(x+z)	x(y+z) = xy+xz

Boolean Algebra

- Our last group of Boolean identities are perhaps the most useful.
- If you have studied set theory or formal logic, these laws are also familiar to you.



Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$\overline{(xy)} = \bar{x} + \bar{y}$	$\overline{(x+y)} = \bar{x}\bar{y}$
Double Complement Law	$\overline{(\bar{x})} = x$	

Boolean Algebra

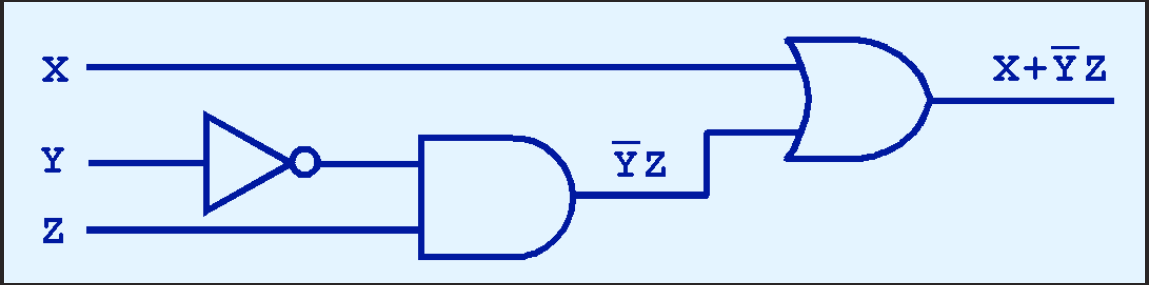
- Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly.
- DeMorgan's law provides an easy way of finding the complement of a Boolean function.
- Recall DeMorgan's law states:

$$\overline{(xy)} = \overline{x} + \overline{y} \quad \text{and} \quad \overline{(x+y)} = \overline{x}\,\overline{y}$$

Digital Components

- The main thing to remember is that combinations of gates implement Boolean functions.
- The circuit below implements the Boolean function

$$F(X,Y,Z) = X+\overline{Y}Z$$



We simplify our Boolean expressions so that we can create simpler circuits.

Practice Exercises:

Using Boolean algebra techniques, simplify the following expression

1. $AB + A(B + C) + B(B + C)$
2. $[A\bar{B}(C + BD) + \bar{A}\bar{B}]C$
3. $\bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + A\bar{B}C + ABC$

Practice Exercises:

Apply DeMorgan's theorems to each of the following expressions

1. $\overline{(A + B + C)D}$

2. $\overline{ABC + DEF}$

3. $\overline{A\bar{B} + \bar{C}D + EF}$