Digital Numbering Systems

- Decimal
- Binary
- Octal
- Hexadecimal

We will focus mainly on Binary and Hex.

Decimal System

The decimal system is composed of 10 numerals or symbols. These 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Using these symbols as digits of a number, we can express any quantity. The decimal system is also called the base-10 system because it has 10 digits.

10^3	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}
=1000	=100	=10	=1	.	=0.1	=0.01	=0.001
Most Significant Digit				Decimal point			Least Significant Digit

Even though the decimal system has only 10 symbols, any number of any magnitude can be expressed by using our system of positional weighting.

The concept of positional weighting is important to understand because it is applicable to evaluating values in all number systems

Binary System

In the binary system, there are only two symbols or possible digit values, 0 and 1. This base-2 system can be used to represent any quantity that can be represented in decimal or other base system.

2^3	2^2	2^1	2^0
=8	=4	=2	=1
Most Significant Digit			

We will not concern ourselves with binary fractions and decimal points.

Binary Counting

The Binary counting sequence is shown in the table:

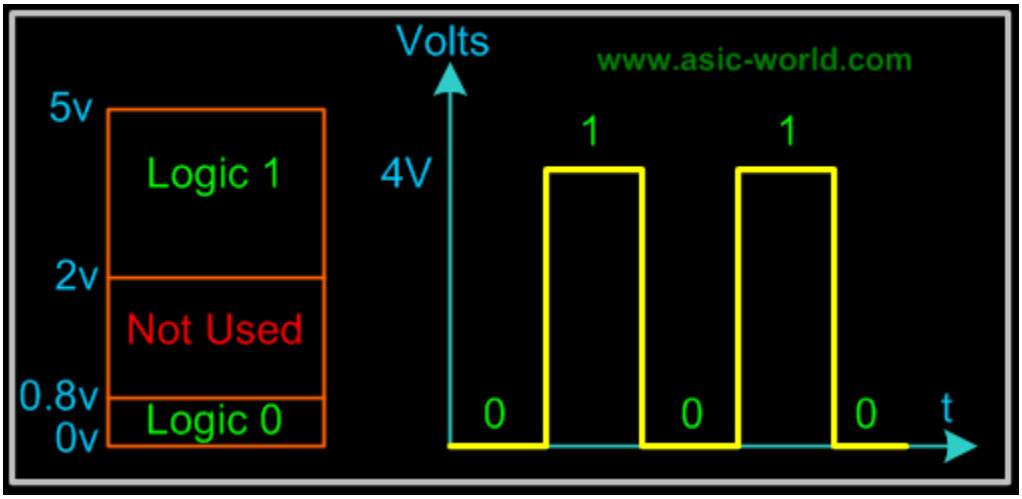
2^3	2^2	2^1	2^0	Decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Representing Binary Quantities

In digital systems the information that is being processed is usually presented in binary form. Binary quantities can be represented by any device that has only two operating states or possible conditions. E.g.. a switch is only open or closed. We arbitrarily (as we define them) let an open switch represent binary 0 and a closed switch represent binary 1. Thus we can represent any binary number by using series of switches.

Typical Voltage Assignment

Binary 1: Any voltage between 2V to 5V
Binary 0: Any voltage between 0V to 0.8V
Not used: Voltage between 0.8V to 2V in 5 Volt CMOS and TTL Logic, this may cause error in a digital circuit. Today's digital circuits works at 1.8 volts, so this statement may not hold true for all logic circuits.



Octal System

The octal number system has a base of eight, meaning that it has eight possible digits: 0,1,2,3,4,5,6,7.

8^3	8^2	8^1	8^0
=512	=64	=8	=1
Most Significant Digit			

Octal to Decimal Conversion

- $237_8 = 2 \times (8^2) + 3 \times (8^1) + 7 \times (8^0) = 159_{10}$
- $24.6_8 = 2 \times (8^1) + 4 \times (8^0) + 6 \times (8^{-1}) = 20.75_{10}$
- $11.1_8 = 1 \times (8^1) + 1 \times (8^0) + 1 \times (8^{-1}) = 9.125_{10}$
- $12.3_8 = 1 \times (8^1) + 2 \times (8^0) + 3 \times (8^{-1}) = 10.375_{10}$

Hexadecimal System

The hexadecimal system uses base 16. Thus, it has 16 possible digit symbols. It uses the digits 0 through 9 plus the letters A, B, C, D, E, and F as the 16 digit symbols.

16^3	16^2	16^1	16^0
=4096	=256	=16	=1
Most Significant Digit			

Hexadecimal to Decimal Conversion

- $24.6_{16} = 2 \times (16^1) + 4 \times (16^0) + 6 \times (16^{-1}) = 36.375_{10}$
- $11.1_{16} = 1 \times (16^1) + 1 \times (16^0) + 1 \times (16^{-1}) = 17.0625_{10}$
- $12.3_{16} = 1 \times (16^1) + 2 \times (16^0) + 3 \times (16^{-1}) = 18.1875_{10}$

Code Conversion

Converting from one code form to another code form is called code conversion, like converting from binary to decimal or converting from hexadecimal to decimal.

Binary-To-Decimal Conversion

Any binary number can be converted to its decimal equivalent simply by summing together the weights of the various positions in the binary number which contain a 1.

Binary	Decimal
11011_2	
$2^4+2^3+0^1+2^1+2^0$	$=16+8+0+2+1$
Result	27_{10}

and

Binary	Decimal
10110101_2	
$2^7+0^6+2^5+2^4+0^3+2^2+0^1+2^0$	$=128+0+32+16+0+4+0+1$
Result	181_{10}

You should have noticed that the method is to find the weights (i.e., powers of 2) for each bit position that contains a 1, and then to add them up.

Decimal-To-Binary Conversion

There are 2 methods:

- Reverse of Binary-To-Decimal Method
- Repeat Division

Reverse of Binary-To-Decimal Method

Decimal	Binary
45_{10}	$=32 + 0 + 8 + 4 + 0 + 1$
	$=2^5+0+2^3+2^2+0+2^0$
Result	$=101101_2$

Repeat Division-Convert decimal to binary

This method uses repeated division by 2.

Convert 25_{10} to binary

Division	Remainder	Binary
25/2	= 12+ remainder of 1	1 (Least Significant Bit)
12/2	= 6 + remainder of 0	0
6/2	= 3 + remainder of 0	0
3/2	= 1 + remainder of 1	1
1/2	= 0 + remainder of 1	1 (Most Significant Bit)
Result	25_{10}	= 11001_2

Binary-To-Octal / Octal-To-Binary Conversion

Octal Digit	0	1	2	3	4	5	6	7
Binary Equivalent	000	001	010	011	100	101	110	111

Each Octal digit is represented by three binary digits.

Example:
$100\ 111\ 010_2 = (100)\ (111)\ (010)_2 = 4\ 7\ 2_8$

Repeat Division-Convert decimal to octal

This method uses repeated division by 8.

Example: Convert 177_{10} to octal and binary

Division	Result	Binary
177/8	= 22+ remainder of 1	1 (Least Significant Bit)
22/ 8	= 2 + remainder of 6	6
2 / 8	= 0 + remainder of 2	2 (Most Significant Bit)
Result	177_{10}	= 261_8
Binary		= 010110001_2

Hexadecimal to Decimal/Decimal to Hexadecimal Conversion

Example:
$2AF_{16} = 2 \times (16^2) + 10 \times (16^1) + 15 \times (16^0) = 687_{10}$

Repeat Division- Convert decimal to hexadecimal
This method uses repeated division by 16.

Example: convert 378_{10} to hexadecimal and binary:

Division	Result	Hexadecimal
378/16	= 23+ remainder of 10	A (Least Significant Bit)23
23/16	= 1 + remainder of 7	7
1/16	= 0 + remainder of 1	1 (Most Significant Bit)
Result	378_{10}	= $17A_{16}$
Binary		= 0001 0111 1010_2

Binary-To-Hexadecimal /Hexadecimal-To-Binary Conversion

Hexadecimal Digit	0	1	2	3	4	5	6	7
Binary Equivalent	0000	0001	0010	0011	0100	0101	0110	0111

Hexadecimal Digit	8	9	A	B	C	D	E	F
Binary Equivalent	1000	1001	1010	1011	1100	1101	1110	1111

Each Hexadecimal digit is represented by **four** bits of binary digit.

Example:

$1011\ 0010\ 1111_2 = (1011)\ (0010)\ (1111)_2 = B\ 2\ F_{16}$

Boolean Algebra

Symbolic Logic

Boolean algebra derives its name from the mathematician George Boole. Symbolic Logic uses values, variables and operations :

- **True** is represented by the value **1**.
- **False** is represented by the value **0**.

Variables are represented by letters and can have one of two values, either 0 or 1. Operations are functions of one or more variables.

- **AND** is represented by X.Y
- **OR** is represented by X + Y
- **NOT** is represented by X' . Throughout this tutorial the X' form will be used and sometime !X will be used.

These basic operations can be combined to give expressions.

Example :

- X
- X.Y
- W.X.Y + Z

Precedence

As with any other branch of mathematics, these operators have an order of precedence. NOT operations have the highest precedence, followed by AND operations, followed by OR operations. Brackets can be used as with other forms of algebra. e.g.

X.Y + Z and **X.(Y + Z)** are not the same function.

Function Definitions

The logic operations given previously are defined as follows :

Define f(X,Y) to be some function of the variables X and Y.

f(X,Y) = X.Y

- 1 if X = 1 and Y = 1
- 0 Otherwise

f(X,Y) = X + Y

- 1 if X = 1 or Y = 1
- 0 Otherwise

f(X) = X'

- 1 if X = 0
- 0 Otherwise

Truth Tables:

Truth tables are a means of representing the results of a logic function using a table. They are constructed by defining all possible combinations of the inputs to a function, and then calculating the output for each combination in turn. For the three functions we have just defined, the truth tables are as follows.

AND

X	Y	F(X,Y)
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	F(X,Y)
0	0	0
0	1	1
1	0	1
1	1	1

NOT

X	F(X)
0	1
1	0

Truth tables may contain as many input variables as desired

F(X,Y,Z) = X.Y + Z

X	Y	Z	F(X,Y,Z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1