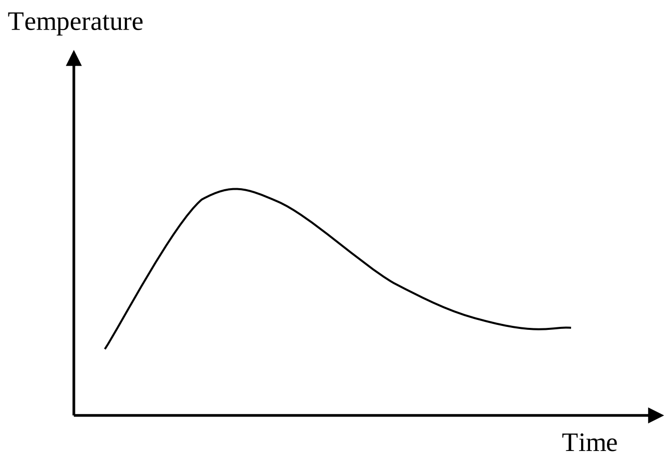
ANALOG AND DIGITAL ELECTRONIC CONCEPTS, NUMBER SYSTEMS

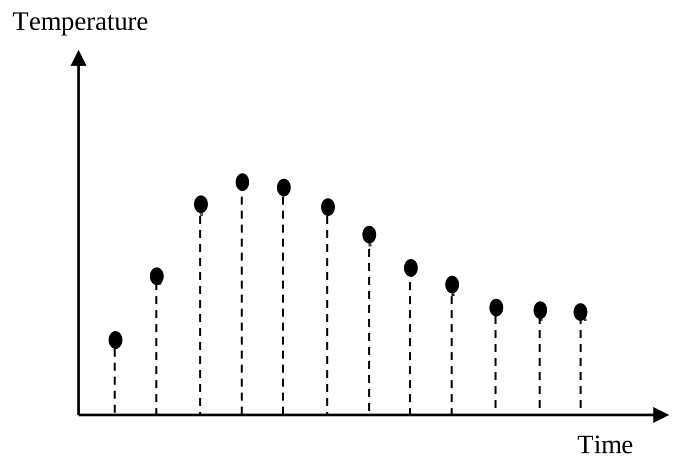
- **Analog vs. Digital**
- **Number Systems**
- **Arithmetic Operations on Binary Numbers**
- **One’s Complement and Two’s Complement**
- **Signed Numbers**

Analog vs. Digital

There are two types of electronic circuits: Analog and digital circuits.

- Analog signal is continuous. It means that, analog circuits process continuous values.
- Digital signal is discrete (discontinuous). Digital circuits process discrete values.





Digital representation of data has advantages over analog representation, in terms of data processing, interpreting, storing, and transmission.

Analog vs. Digital

A digital value is represented by a combination of ON and OFF voltage values. Generally, +5V is considered ON, and 0V is considered OFF.

For instance, the temperature value 25ºC can be input to a digital circuit as a string that consists of ON and OFF values.

These voltage values are interpreted as logic 1's and 0's, and can be represented as "11001" in binary system.

Number Systems

Decimal System (Base 10)

This is the system we use in our daily life. The digits from 0 to 9 is used to represent 10 different values.

123.456

Weights of corresponding digits are: 10^2 10^1 10^0 . 10^{-1} 10^{-2} 10^{-3}

The decimal number 123.456 denotes

$$(1\times10^2)+(2\times10^1)+(3\times10^0)+(4\times10^{-1})+(5\times10^{-2})+(6\times 10^{-3})$$

Number Systems

Binary System (Base 2): In binary system, only the symbols 0 and 1 are used to represent the numbers. 0's and 1's are called bits.

In binary numbers, the rightmost bit is called LSB (Least Significant Bit), and the leftmost bit is called MSB (Most Significant Bit).

11001001_2

MSB LSB

An n-bit binary number can be between 0 and $(2^n - 1)$.

110.011_2

Weights of corresponding bits are: $2^2 \quad 2^1 \quad 2^0 \;.\; 2^{-1} \quad 2^{-2} \quad 2^{-3}$

Number Systems

Conversion of binary numbers to decimal:

Binary numbers are converted to decimal by adding up the weights of 1's.

The binary number 10110_2 corresponds to decimal number $2^4+2^2+2^1=22$

In other words;

$(10110)_2 = (22)_{10}$

The binary number 10.101_2 corresponds to decimal $2^1+2^{-1}+2^{-3} = 2.625$

Number Systems

Conversion of decimal numbers to binary:

This conversion can be done in two ways.

1st method

Write down the binary weights as a sequence as below. (The number of bits depends on the number to convert).

… 256 128 64 32 16 8 4 2 1

Then, find out which weights have a sum of the number to convert. And then, mark those bits as 1, and the rest as 0.

Example: Let's convert 98 to binary.

98 = 64 + 32 + 2

So, we find out that the binary number is;

64 32 16 8 4 2 1

$(1 \ \ 1 \ \ 0 \ \ 0 \ 0 \ 1 \ 0)_2$

This method is practical especially when converting smaller numbers.

Conversion of decimal numbers to binary:

2^{nd} method

To convert a decimal number to binary, we divide the decimal number by 2 consecutively until the quotient is 0, and take the remainders.

The remainder of first division is the LSB, and the remainder of last division is MSB.

Example: Let's convert $(13)_{10}$ to binary.

<u>Division</u>	<u>Remainder</u>
13/2 = 6	1 (LSB)
6/2= 3	0
3/2= 1	1
1/2= 0	1 (MSB)

$(13)_{10} = (1101)_2$

Conversion of decimal floating point numbers to binary:

This conversion can be done in two ways.

1st method

Write down the weights of fractional part of the floating point number as a sequence below.

0.5 0.25 0.125 0.0625 …

Then, find out which weights have a sum of the fractional part to convert. And then, mark those bits as 1, and the rest as 0.

Example: Let's convert $(0.625)_{10}$ to binary.

0.5 0.25 0.125 …

$(1 \quad 0 \quad\quad 1)_2$

So, $(0.625)_{10} = (0.101)_2$

Conversion of decimal floating point numbers to binary:
2nd method

Multiply the fractional part by 2 consecutively until the fractional part of the result is 0, or until we decide that we have enough bits.

Then, we get the whole parts of the results.

Example: Let's convert $(0.625)_{10}$ to binary.

$$\left.\begin{array}{r} 0.625\times2 = \mathbf{1}.25 \text{ (MSB)} \\ 0.25\times2 = \mathbf{0}.50 \\ 0.5\times2 = \mathbf{1}.00 \text{ (LSB)} \end{array}\right\} (0.625)_{10}=(0.101)_2$$

The fractional part is 0.
It means that the conversion is done.

Arithmetic Operations on Binary Numbers

Binary Addition

Binary addition has 4 basic rules:

0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = **1**0   (0 carry **1**)

If we also have a carry bit to add, there are 4 different probabilities.

Carry Sum

Carry

1 + 0 + 0 = 01

1 + 0 + 1 = 10

1 + 1 + 0 = 10

1 + 1 + 1 = 11

Example:

$$\begin{array}{r} 101_2 \\ +\ 001_2 \\ \hline 110_2 \end{array}$$

Arithmetic Operations on Binary Numbers

Binary Subtraction

Binary subtraction has 4 basic rules.

0 - 0 = 0, 1 - 0 = 1, 1 - 1 = 0, **1**0 - 1 = 1 (We borrow 1 from the left bit)

Example:

$$\begin{array}{r} 110 \\ -\ 001 \\ \hline 101 \end{array}$$

Binary Multiplication

Binary multiplication has 4 basic rules

$0 \times 0 = 0$, $0 \times 1 = 0$, $1 \times 0 = 0$, $1 \times 1 = 1$

It is similar to decimal multiplication.

Arithmetic Operations on Binary Numbers

Example:

$$
\begin{array}{r}
10 \\
\times\ 11 \\
\hline
\left.\begin{array}{r} 10 \\ +10 \end{array}\right\} \text{Partial products} \\
\hline
110
\end{array}
$$

Binary Division

This operation is also similar to decimal division.

Example:

$$
\begin{array}{r|l}
\overline{1001} & 11 \\
\cline{2-2}
-\ 11 & 11 \\
\cline{1-1}
011 & \\
-\ 11 & \\
\cline{1-1}
000 &
\end{array}
$$

One's Complement and Two's Complement

One's complement and two's complement are especially used in representing negative numbers and performing arithmetic operations on negative numbers.

One's complement

We calculate one's complement by simply converting 1's to 0's and 0's to 1's.

Example:

Binary Number 110110

$\downarrow$ $\downarrow$

One's Complement 001001

We can also calculate one's complement by the formula 2^n-N-1 where n is the number of bits, and N is the binary number.

Example:

One's complement of 110110_2 is 2^6-110110-1=1000000-110110-1

=1000000-(110111)

= 001001

One's Complement and Two's Complement

Two's complement

We calculate two's complement by simply adding 1 to one's complement.

Binary Number	$\longrightarrow$	110110
One's Complement	$\longrightarrow$	001001
		+ 1
Two's Complement	$\longrightarrow$	001010

We can also calculate two's complement by the formula 2^n-N where n is the number of bits, and N is the binary number.

Example:

Two's complement of 110110_2 is 2^6-110110 = 1000000-110110

= 001010

Signed Numbers

Signed numbers contain both the sign and the magnitude. Sign is whether the number is positive or negative, and quantity is the value of the number. In binary system, signed numbers can be represented in three forms: Sign-Magnitude, One's Complement, and Two's Complement.

In all these three methods, if the MSB is 1, then the number is negative.

Sign-Magnitude Form

In sign-magnitude form, the MSB represents the sign, and the rest bits represent the magnitude.

The magnitude part is the value of the number, no matter what the sign is.

Example: Let's convert $(-19)_{10}$ and $(19)_{10}$ to 8-bit binary.

$$-19 = (\underset{\text{Sign bit}}{\underset{\downarrow}{1}}\underbrace{0010011}_{\text{Magnitude bits}})_2 \qquad 19 = (\underset{\text{Sign bit}}{\underset{\downarrow}{0}}\underbrace{0010011}_{\text{Magnitude bits}})_2$$

Signed Numbers

One's Complement Form

Positive numbers are represented just the same as sign-magnitude form. Negative numbers are represented as one's complement of the magnitude.

Example: Assuming that we store the numbers in 8-bits, we represent the number -19 as one's complement of +19 $(00010011)_2$, which is $(11101100)_2$.

Two's Complement Form

Positive numbers are represented just the same as sign-magnitude form. Negative numbers are represented as two's complement of the magnitude.

Example: Assuming that we store the numbers in 8-bits, we represent the number -19 as two's complement of +19 $(00010011)_2$, which is $(11101101)_2$.

Converting Signed Numbers to Decimal

If the number is in sign-magnitude form, we convert *the magnitude part* (excluding the sign bit) just like an unsigned number. And then, check the sign bit to detect whether the number is positive or negative.

Example: Let's convert signed binary number 10011000_2 to decimal.

$-(2^3+2^4) = -24$

If the number is in one's complement form, we convert *positive numbers* just like an unsigned number. To convert *negative numbers*, we add negative weight of the sign bit to the sum of other bits' weights, and then, add 1.

Example: 00011000_2 is a positive number because the sign bit is 0. So, the decimal value is $2^4+2^3=24$.

Example: 11100111_2 is a negative number because the sign bit is 1. So, the decimal value is $-2^7+2^6+2^5+2^2+2^1+2^0 + 1 = -24$.

Converting Signed Numbers to Decimal

If the number is in one's complement form, we convert positive and negative numbers by adding negative weight of the sign bit to the sum of other bits' weights.

With N bits, we can represent numbers between -2^{N-1} and $(2^{N-1} -1)$ in two's complement form.

Example: The number 01000101_2 in two's complement form is $2^6+2^2+2^0=69$ in decimal.

Example: The number 10111011_2 in two's complement form is $-2^7+2^5+2^4+2^3+2^1+2^0 = -69$ in decimal.

Two's complement form is used extensively because both negative and positive numbers can be converted the same way. However, in one's complement form, positive and negative numbers are converted in different ways. Another reason is that the number 0 causes an ambiguity in one's complement form because it can be represented in two different forms: 00000000_2 and 11111111_2.