330en02

2

BINARY LOGIC

In the previous lesson, you have learnt about computer fundamentals. In this lesson you will learn about binary logic. In digital computers, everything (including letters, words and whole texts) is represented in binary numbers (zero and one) i.e., it stores data in terms of bits. Binary logic is the basis of electronic systems, such as computers and cell phones. It works on 0's and 1's. It involves addition, subtraction, multiplication, division of zeros and ones. It includes logic gate functions, AND, OR and NOT which translates input signals into specific output. It facilitates computing, robotics and other electronic applications. In this lesson you will learn about binary numbers, octal numbers, hexadecimal numbers, conversion from one number system to another, character coding system and logic gates.

OBJECTIVES

After reading this lesson, you will be able to:

- define binary numbers, octal numbers and hexadecimal numbers;
- convert from one number system to another;
- use character coding system;
- identify logic gates.

2.1 NUMBER SYSTEM

You know that computer stores and processes two basic types of data viz.,character and number. The character type data includes alphabets and some special symbols, while the number data type includes numerical data on which arithmetic calculations can be done. For example, student name contains character data, roll number will have numeric data.

Digital computers internally use the binary (base 2) number system to represent data and perform arithmetic calculations. The binary number system is very

Notes

efficient for computers, but not for humans. Representing even relatively small numbers with the binary system requires working with long strings of ones and zeroes.

2.1.1 Binary numbers

Binary numbers are combination of two basic numbers (zero and one) while decimal numbers are combinations of 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.All the numbers which are generated by decimal numbers can also be generated by Binary numbers. Binary number has base 2 because all numbers are generated by combination of only two numbers (0 and 1) and decimal number has base 10 because all numbers are generated by combination of only ten numbers (0 to 9).

2.1.2 Octal Number and Hexadecimal numbers

Octal (base 8) was previously a popular choice for representing digital circuit numbers in a form that is more compact than binary. Octal is sometimes abbreviated as oct.

Octal counting starts at 0 and goes: 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13 14 15 and so on.

Hexadecimal (base 16) is currently the most popular choice for representing digital circuit numbers in a form that is more compact than binary. Hexadecimal numbers are sometimes represented by preceding the value with '0x', as in 0x1B84. Hexadecimal is sometimes abbreviated as hex.

Hexadecimal counting goes: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, and so on.

The following table shows the decimal number 0 to 15 and its corresponding binary, octal and hexadecimal values.

Decimal, Binary, Octal and Hexadecimal values

Decimal	Binary	Octal	Hexadecimal
0	0000	000	0
1	0001	001	1
2	0010	002	2
3	0011	003	3
4	0100	004	4
5	0101	005	5

6	0110	006	6
7	0111	007	7
8	1000	010	8
9	1001	011	9
10	1010	012	A
11	1011	013	B
12	1100	014	C
13	1101	015	D
14	1110	016	E
15	1111	017	F

All four number systems are equally capable of representing any number. Furthermore, a number can be perfectly converted between the various number systems without any loss of numeric value.

2.1.3 Conversion from one system to another

1. Decimal to Binary Conversion

There are two methods you can use: successive division and subtracting values using a table. Successive division requires dividing continuously by the base we are converting to until the quotient equals 0. The remainders compose the answer.

Steps for conversion from Decimal to Binary

1. Divide the decimal number by 2.
2. Take the remainder and record it on the side.
3. Repeat until the decimal number can not be divided into any more.
4. With the bits, record them in order from right to left as that will be the number in base 2.

For example, convert 9 into binary as follows;

9/2 = 4	and remainder = 1 first remainder is called LSB (least significant bit)
4/2 = 2	and remainder = 0
2/2 = 1	and remainder = 0
1/2 = 0	and remainder = 1 last remainder is called MSB (Most significant bit)

Now, decimal 9 is equivalent to 1001 binary number

$$(9)_{10} = (1001)_2$$

MSB (Most significant bit) **LSB (least significant bit)**

2. Decimal to Octal Conversion

We can convert decimal to octal and hexadecimal by using the similar conversion method of decimal to binary. i.e., successive division, which requires dividing continuously by the base we are converting to until the quotient equals 0. The remainders compose the answer.

Steps for conversion from Decimal to octal

1. Divide the decimal number by 8.
2. Take the remainder and record it on the side.
3. Repeat until the decimal number can not be divided any more.
4. With the bits, record them in order from right to left as that will be the number in base 8.

For example, to convert 19 into Octal follow the steps:

19/8=2 and remainder = 3 first remainder is called LSB (least significant bit)

2/8=0 and remainder = 2 last remainder is called MSB (Most significant bit)

Now decimal number **19** is equivalent to octal number **23**

$$(19)_{10} = (23)_8$$

3. Decimal to Hexadecimal Conversion

Division method can be used for conversion from decimal to hexadecimal.

Steps for conversion from Decimal to Hexadecimal

1. Divide the decimal number by 16
2. Take the remainder and record it on the side.
3. Repeat until the decimal number can not be divided into any more.
4. With the bits, record them in order from right to left as that will be the number in base 16.

For example, to convert decimal number 229 into hexadecimal number:

229/16 = 14 and remainder = 5 first remainder is called LSB (least significant bit)

14/16 = 0 and remainder = 14 last remainder is called MSB (Most significant bit)

In hexadecimal 10,11,12,13,14,15 are equivalent to A,B,C,D,E,F respectively

So $(229)_{10} = (E5)_{16}$

4. Binary to decimal

To find the decimal representation of a binary number simply take the sum of products of binary digits and the powers of 2 which they represent.

For example,

Conversion of $(1000)_2$ into decimal is as follows:

$(1000)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

$= 1 \times 8 + 0 \times 4 + 0 \times 2 + 0 \times 1$

$(1000)_2 = 8 + 0 + 0 + 0 = 8$

5. Octal to decimal

To find the decimal representation of an octal number simply take the sum of products of octal digits and the powers of 8 which they represent.

For example,

Conversion of $(1000)_8$ into decimal is as follows:

$(1000)_8 = 1 \times 8^3 + 0 \times 8^2 + 0 \times 8^1 + 0 \times 8^0$

$= 1 \times 512 + 0 \times 64 + 0 \times 8 + 0 \times 1$

$(1000)_8 = 512 + 0 + 0 + 0 = 512$

6. Hexadecimal to decimal

To find the decimal representation of a Hexadecimal number simply take the sum of products of hex number and the powers of 16 which they represent.

For example,

Conversion of $(1000)_{16}$ into decimal is as follows:

$(1000)_{16} = 1 \times 16^3 + 0 \times 16^2 + 0 \times 16^1 + 0 \times 16^0$

$= 1 \times 4096 + 0 \times 256 + 0 \times 16 + 0 \times 1$

$(1000)_8 = 4096 + 0 + 0 + 0 = 4096$

Now let us learn about one's complement and two's complement of a binary number.

2.2 ONE'S COMPLEMENT AND TWO'S COMPLEMENT

One's complement of a binary number can be achieved by changing all 0's to 1's and 1's to 0's.

Example:

There is a binary number 11001 and its 1's complement would be 00110 after all 0's to 1's and 1's to 0's.

Two's complement of binary number can be obtained by adding 1 to the LSB (Least Significant Bit) of 1's complement of binary number.

2's complement= 1's complement +1

Example:

Binary number: 11001

1's complement of number: 00110 (i.e. convert all 1's to 0's and all 0's to 1's)

2's complement of number= 00110+1=00111

2.3 CHARACTER CODING SYSTEM (ASCII, ISCII & UNICODE)

Computer works on character data and this data is not only alphabet but numeric values, punctuation; spaces, etc., are also character data. In common language whatever is on keyboard (except shift, caps lock key) are character data. As you know computer operates on binary values so these character are also represented in binary values. Most common character coding used in India are ASCII, ISCII and UNICODE.

ASCII

ASCII stands for *American Standard Code for Information Interchange*. Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'A' has numerical value as 65. Some other characters and its equivalent ASCII values are following:

Table 2.1: ASCII codes table

ASCII	Hex	Symbol
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D

69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

From the above table you can see, characters have equivalent ASCII values. Similarly, each character has ASCII value.

ASCII value for a to z is from 97 to 122 i.e., a = 97, z = 122.

Indian Script Code for Information Interchange (ISCII)

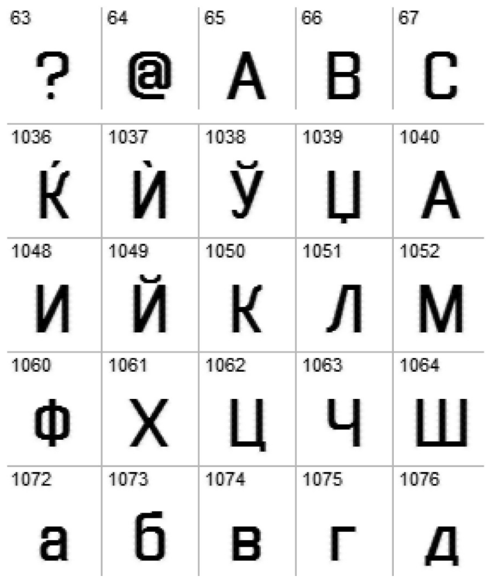
This coding scheme is used for Indian script and its symbols. Using this, it is possible to define a syllable boundary for Indian script word. It was developed by Bureau of Indian Standards in 1986 and revised to a more compact form in 1988. The table 2.2 shows few of the character set for Devanagari script -

Table 2.2: "first vowel of Hindi has numeric value 164"

अ 0905 164	आ 0906 165	इ 0907 166	ई 0908 167
उ 0909 168	ऊ 090A 169	ऋ 090B 170	ऐ 090E 171
क 0915 179	ख 0916 180	ग 0917 181	घ 0918 182
0 0966 241	1 0967 242	2 0968 243	3 0969 244

Unicode

Developed in 1987, Unicode provides a unique number for every character of all languages. It was developed to represent characters from a wide range of languages and can thus represent characters from English, French, Greek, Korean etc., all at the same time. The first 256 places of Unicode are same as ASCII and it expands the character set for new and unique characters that may arise in future.



63 ?	64 @	65 A	66 B	67 C
1036 Ќ	1037 Ѝ	1038 Ў	1039 Џ	1040 А
1048 И	1049 Й	1050 К	1051 Л	1052 М
1060 Ф	1061 Х	1062 Ц	1063 Ч	1064 Ш
1072 а	1073 б	1074 в	1075 г	1076 д

Fig. 2.1: Unicode Representation

2.4 LOGIC GATES

Logic gates are elementary building block of a digital circuit. Most logic gates have two inputs and only one output. The basic logic gates are AND, OR and NOT, NAND, NOR, XOR and XNOR. In this lesson, let us learn about AND, OR, NOT, NAND and NOR logic gates. Binary logic deals with true and false. You know that 0 is usually associated with false and 1 is associated with true.

Example

(a) $1 + 0 = 1$ (True)

(b) $0 + 1 = 1$ (True)

(c) $0 + 0 = 0$ (False)

(d) $1 + 1 = 1$ (False)

2.4.1 AND gate

This gate acts in the same way as the logical operator 'and' works. An AND operation produces an output of logic-0 if at least one of the input is zero. Output will be logic-1 if all the inputs are logic-1. Truth table of AND operation is shown below. A truth table is a list of all possible combination of input logic levels along with corresponding output level (refer to Table 2.3).

Table 2.3: Truth Table for AND operation

Input		**Output**
A	**B**	**C = A.B**
0	0	0
0	1	0
1	0	0
1	1	1

The equation for logic AND operation

$$C = AB$$

AND gate representation

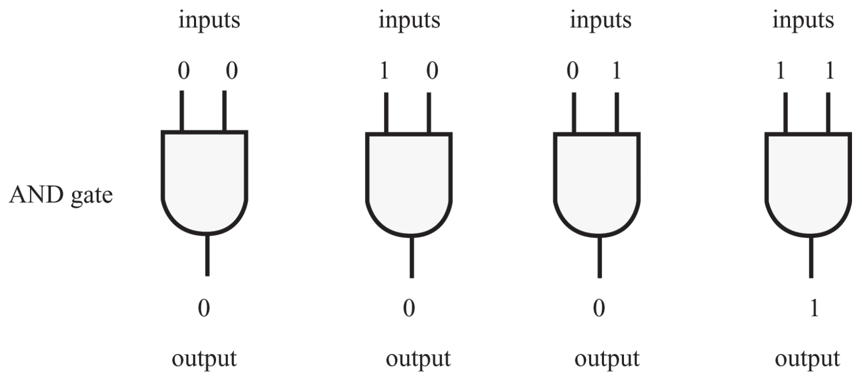


Fig. 2.2: AND gate

2.4.2 OR gate

This operation is represented by plus (+) sign. The OR gate produces an output of logic-1 if at least one of the inputs is logic-1.

Truth table of OR operation is given in Table 2.4

Table 2.4: Truth Table for OR operation

Input		**Output**
A	**B**	**C = A + B**
0	0	0
0	1	1
1	0	1
1	1	1

Equation of OR operation: $C = A + B$

OR gate representation

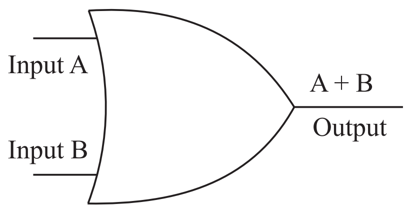


Fig. 2.3: OR Gate representation

In the above representation A,B are inputs and the output is A+B.

2.4.3 NOT operation

The NOT operation represents complement or inverse of inputs. A prime or bar represents this operation. The truth table for NOT operation is given in Table 2.5.

Table 2.5: Truth table for NOT operation

(Input)	(Output)
A	$\bar{A}$
0	1
1	0

NOT gate representation

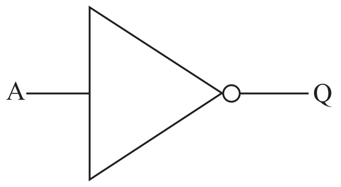


Fig. 2.4: NOT Gate representation

Equation for NOT operation

$$Q = \bar{A}$$

Other operations are combination of AND, OR and NOT. They are NOR and NAND.

- NOR operation is combination of OR and NOT. NAND operation is combination of AND and NOT.

- NAND and NOR operations are also known as universal operations because you can generate any other operation through it. These gates are called **Universal Gates**.

2.4.4 NOR Operation

It is the NOT-OR gate. The output is true when neither A nor B is true. If any one of the input variables is true then output is false. NOR gate represents complement of the OR operation. The graphic symbol for the NOR gate consists of an OR symbol with a bubble on the output, denoting that a complement operation is performed on the output of the OR gate.

Table 2.6: Truth table of NOR operation

Input		**Output**
A	**B**	**$C = \overline{A + B}$**
0	0	1
0	1	0
1	0	0
1	1	0

NOR gate representation

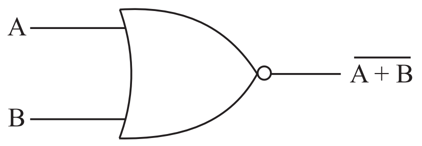


Fig. 2.5: NOR Gate representation

Equation for NOR gate

$$C = (A + B)^1 \text{ or } \overline{A + B}$$

2.4.5 NAND Operation

The NAND gate represents the complement of the AND operation. The graphic symbol for the NAND gate consists of an AND symbol with a bubble on the output, denoting that a complement operation is performed on the output of the AND gate.

Truth table of NAND operation is given in Table 2.7.

Table 2.7: Truth table of NAND operation

Input		**Output**
A	**B**	**C = $\overline{A \cdot B}$**
0	0	1
0	1	1
1	0	1
1	1	0

NAND gate representation

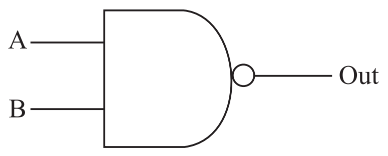


Fig. 2.6: NAND Gate representation

Equation for NAND gate

$$C = \overline{A \cdot B} \text{ or } (A \cdot B)'$$

NAND and NOR gates are called universal gates as they can implement any type of boolean expression.

Other two primary logic gates are XOR - Exclusive OR gate and XNOR - Exclusive NOR gate. In case of XOR gate the ouput is 1 only when one of inputs is 1. In case of XNOR gate the output is 1 when the both the inputs are the same

INTEXT QUESTIONS 2.1

State whether the following statements are true or false:

1. Binary logic works on two states.
2. "ON" state is represented by 0.
3. The NOR gate represents the complement of the OR operation.
4. The binary number system is a system of two digits-0 and 1.
5. 1 = 0 AND 1.

WHAT YOU HAVE LEARNT

- A computer stores and processes two basic types of data viz.,character and number.
- Binary numbers are combination of zeroes and ones.
- Octal or oct number uses digits 0 to 7.
- ASCII stands for *American Standard Code for Information Interchange*.
- ISCII stands for Indian Script Code for Information Interchange (ISCII)
- Basic binary operations are AND, OR and NOT that can be described in the form of GATEs. These gates are known as logic gates.
- NAND and NOR are universal gates because all operations can be performed through it.

TERMINAL EXERCISE

1. Define AND gate and its operation.
2. Differentiate between AND & OR gates.
3. What is a universal gate?
4. How can you create NOR gate using OR and NOT gate?
5. Convert $(990)_{10}$ into binary.

ANSWERS TO INTEXT QUESTIONS

2.1

1. True
2. False
3. True
4. True
5. False