UNIT – I

Number System and Codes

- **Binary number system**
- **Binary to Decimal**
- **Decimal to Binary**
- **Hexadecimal**
- **Ascii code**
- **Excess-3 Code**
- **Gray Code**

Digital Logic

- **Basic Gates- AND, OR, NOT**
- **Universal Logic Gates- NOR, NAND**

NUMBER SYSTEM AND CODES

In a digital system, the system can understand only the optional number system. In these systems, digits symbols are used to represent different values, depending on the index from which it settled in the number system.

In simple terms, for representing the information, we use the number system in the digital system.

The digit value in the number system is calculated using:

1. The digit
2. The index, where the digit is present in the number.
3. Finally, the base numbers, the total number of digits available in the number system.

Types of Number System

In the digital computer, there are various types of number systems used for representing information.

1. Binary Number System
2. Decimal Number System
3. Hexadecimal Number System
4. Octal Number System

Binary Number System

Generally, a binary number system is used in the digital computers. In this number system, it carries only two digits, either 0 or 1. There are two types of electronic pulses present in a binary number system. The first one is the absence of an electronic pulse representing '0'and second one is the presence of electronic pulse representing '1'. Each digit is known as a bit. A four-bit collection (1101) is known as a nibble, and a collection of eight

bits (11001010) is known as a byte. The location of a digit in a binary number represents a specific power of the base (2) of the number system.

Characteristics:

1. It holds only two values, i.e., either 0 or 1.
2. It is also known as the base 2 number system.
3. The position of a digit represents the 0 power of the base(2). Example: 2^0
4. The position of the last digit represents the x power of the base(2). Example: 2^x, where x represents the last position, i.e., 1

Examples:

$(10100)_2$, $(11011)_2$, $(11001)_2$, $(000101)_2$, $(011010)_2$.

Decimal Number System

The decimal numbers are used in our day to day life. The decimal number system contains ten digits from 0 to 9(base 10). Here, the successive place value or position, left to the decimal point holds units, tens, hundreds, thousands, and so on.

The position in the decimal number system specifies the power of the base (10). The 0 is the minimum value of the digit, and 9 is the maximum value of the digit. For example, the decimal number 2541 consist of the digit 1 in the unit position, 4 in the tens position, 5 in the hundreds position, and 2 in the thousand positions and the value will be written as:

$(2\times1000) + (5\times100) + (4\times10) + (1\times1)$
$(2\times10^3) + (5\times10^2) + (4\times10^1) + (1\times10^0)$
$2000 + 500 + 40 + 1$
2541

Octal Number System

The octal number system has base 8(means it has only eight digits from 0 to 7). There are only eight possible digit values to represent a number. With the help of only three bits, an octal number is represented. Each set of bits has a distinct value between 0 and 7.

Below, we have described certain characteristics of the octal number system:

Characteristics:

1. An octal number system carries eight digits starting from 0, 1, 2, 3, 4, 5, 6, and 7.
2. It is also known as the base 8 number system.
3. The position of a digit represents the 0 power of the base(8). Example: 8^0
4. The position of the last digit represents the x power of the base(8). Example: 8^x, where x represents the last position, i.e., 1

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

Examples:

$(273)_8$, $(5644)_8$, $(0.5365)_8$, $(1123)_8$, $(1223)_8$.

Hexadecimal Number System

It is another technique to represent the number in the digital system called the **hexadecimal number system**. The number system has a base of 16 means there are total 16 symbols(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) used for representing a number. The single-bit representation of decimal values10, 11, 12, 13, 14, and 15 are represented by A, B, C, D, E, and F. Only 4 bits are required for representing a number in a hexadecimal number. Each set of bits has a distinct value between 0 and 15. There are the following characteristics of the octal number system:

Characteristics:

1. It has ten digits from 0 to 9 and 6 letters from A to F.
2. The letters from A to F defines numbers from 10 to 15.
3. It is also known as the base 16number system.
4. In hexadecimal number, the position of a digit represents the 0 power of the base(16). Example: 16^0
5. In hexadecimal number, the position of the last digit represents the x power of the base(16). Example: 16^x, where x represents the last position, i.e., 1

Binary Number	Hexadecimal Number
0000	0
0001	1
0010	2

0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Examples:

$(FAC2)_{16}$, $(564)_{16}$, $(0ABD5)_{16}$, $(1123)_{16}$, $(11F3)_{16}$.

Number Base Conversion

Types of number systems are binary, decimal, octal, and hexadecimal. In this part, we will learn how we can change a number from one number system to another number system.

As, we have four types of number systems so each one can be converted into the remaining three systems. There are the following conversions possible in Number System

1. Binary to other Number Systems.
2. Decimal to other Number Systems.
3. Octal to other Number Systems.
4. Hexadecimal to other Number Systems.

Binary to other Number Systems

There are three conversions possible for binary number, i.e., binary to decimal, binary to octal, and binary to hexadecimal. The conversion process of a binary number to decimal differs from the remaining others. Let's take a detailed discussion on Binary Number System conversion.

Binary to Decimal Conversion

The process of converting binary to decimal is quite simple. The process starts from multiplying the bits of binary number with its corresponding positional weights. And lastly, we add all those products.

Let's take an example to understand how the conversion is done from binary to decimal.

Example: $(10110.001)_2$

We multiplied each bit of $(10110.001)_2$ with its respective positional weight, and last we add the products of all the bits with its weight.

$$(10110.001)_2=(1\times24)+(0\times23)+(1\times22)+(1\times21)+(0\times20)+(0\times2\text{-}1)+(0\times2\text{-}2)+(1\times2\text{-}3)$$
$$(10110.001)_2=(1\times16)+(0\times8)+(1\times4)+(1\times2)+(0\times1)+(0\times1/2)+(0\times1/4)+(1\times1/8)$$
$$(10110.001)_2=16+0+4+2+0+0+0+0.125$$
$$(10110.001)_2=(22.125)_{10}$$

Binary to Octal Conversion

The base numbers of binary and octal are 2 and 8, respectively. In a binary number, the pair of three bits is equal to one octal digit. There are only two steps to convert a binary number into an octal number which are as follows:

1. In the first step, we have to make the pairs of three bits on both sides of the binary point. If there will be one or two bits left in a pair of three bits pair, we add the required number of zeros on extreme sides.
2. In the second step, we write the octal digits corresponding to each pair.

Example: $(111110101011.0011)_2$

1. Firstly, we make pairs of three bits on both sides of the binary point.

111 110 101 011.001 1

On the right side of the binary point, the last pair has only one bit. To make it a complete pair of three bits, we added two zeros on the extreme side.

111 110 101 011.001 100

2. Then, we wrote the octal digits, which correspond to each pair.

$(111110101011.0011)_2=(7653.14)_8$

Binary to Hexadecimal Conversion

The base numbers of binary and hexadecimal are 2 and 16, respectively. In a binary number, the pair of four bits is equal to one hexadecimal digit. There are also only two steps to convert a binary number into a hexadecimal number which are as follows:

1. In the first step, we have to make the pairs of four bits on both sides of the binary point. If there will be one, two, or three bits left in a pair of four bits pair, we add the required number of zeros on extreme sides.
2. In the second step, we write the hexadecimal digits corresponding to each pair.

Example: $(10110101011.0011)_2$

1. Firstly, we make pairs of four bits on both sides of the binary point.

111 1010 1011.0011

On the left side of the binary point, the first pair has three bits. To make it a complete pair of four bits, add one zero on the extreme side.

0111 1010 1011.0011

2. Then, we write the hexadecimal digits, which correspond to each pair.

$(011110101011.0011)_2=(7AB.3)_{16}$

Decimal to other Number System

The decimal number can be an integer or floating-point integer. When the decimal number is a floating-point integer, then we convert both part (integer and fractional) of the decimal number in the isolated form(individually). There are the following steps that are used to convert the decimal number into a similar number of any base **'r'**.

1. In the first step, we perform the division operation on integer and successive part with base **'r'**. We will list down all the remainders till the quotient is zero. Then we find out the remainders in reverse order for getting the integer part of the equivalent number of base **'r'**. In this, the least and most significant digits are denoted by the first and the last remainders.
2. In the next step, the multiplication operation is done with base **'r'** of the fractional and successive fraction. The carries are noted until the result is zero or when the required number of the equivalent digit is obtained. For getting the fractional part of the equivalent number of base **'r'**, the normal sequence of carrying is considered.

Decimal to Binary Conversion

For converting decimal to binary, there are two steps required to perform, which are as follows:

1. In the first step, we perform the division operation on the integer and the successive quotient with the base of binary(2).

2. Next, we perform the multiplication on the integer and the successive quotient with the base of binary(2).

Example: $(152.25)_{10}$

Step 1:

Divide the number 152 and its successive quotients with base 2.

Operation	Quotient	Remainder
152/2	76	0 (LSB)
76/2	38	0
38/2	19	0
19/2	9	1
9/2	4	1
4/2	2	0
2/2	1	0
½	0	1(MSB)

$(152)_{10}=(10011000)_2$

Step 2:

Now, perform the multiplication of 0.27 and successive fraction with base 2.

Operation	Result	carry
0.25×2	0.50	0
0.50×2	0	1

$(0.25)_{10}=(.01)_2$

Decimal to Octal Conversion

For converting decimal to octal, there are two steps required to perform, which are as follows:

1. In the first step, we perform the division operation on the integer and the successive quotient with the base of octal(8).
2. Next, we perform the multiplication on the integer and the successive quotient with the base of octal(8).

Example: $(152.25)_{10}$

Step 1:

Divide the number 152 and its successive quotients with base 8.

Operation	Quotient	Remainder
152/8	19	0
19/8	2	3
2/8	0	2

$(152)_{10}=(230)_8$

Step 2:

Now perform the multiplication of 0.25 and successive fraction with base 8.

Operation	Result	carry
0.25×8	**0**	**2**

$(0.25)_{10}=(2)_8$

So, the octal number of the decimal number 152.25 is **230.2**

Decimal to hexadecimal conversion

For converting decimal to hexadecimal, there are two steps required to perform, which are as follows:

1. In the first step, we perform the division operation on the integer and the successive quotient with the base of hexadecimal (16).
2. Next, we perform the multiplication on the integer and the successive quotient with the base of hexadecimal (16).

Example: $(152.25)_{10}$

Step 1:

Divide the number 152 and its successive quotients with base 8.

Operation	Quotient	Remainder
152/16	9	8

9/16	0	9

$(152)_{10}=(98)_{16}$

Step 2:

Now perform the multiplication of 0.25 and successive fraction with base 16.

Operation	Result	carry
0.25×16	0	4

$(0.25)_{10}=(4)_{16}$

So, the hexadecimal number of the decimal number 152.25 is **230.4.**

Octal to other Number System

Like binary and decimal, the octal number can also be converted into other number systems. The process of converting octal to decimal differs from the remaining one. Let's start understanding how conversion is done.

Octal to Decimal Conversion

The process of converting octal to decimal is the same as binary to decimal. The process starts from multiplying the digits of octal numbers with its corresponding positional weights. And lastly, we add all those products.

Let's take an example to understand how the conversion is done from octal to decimal.

Example: $(152.25)_8$

Step 1:

We multiply each digit of **152.25** with its respective positional weight, and last we add the products of all the bits with its weight.

$(152.25)_8=(1\times8^2)+(5\times8^1)+(2\times8^0)+(2\times8^{-1})+(5\times8^{-2})$
$(152.25)_8=64+40+2+(2\times1/8)+(5\times1/64)$
$(152.25)_8=64+40+2+0.25+0.078125$
$(152.25)_8=106.328125$

So, the decimal number of the octal number 152.25 is **106.328125**

Octal to Binary Conversion

The process of converting octal to binary is the reverse process of binary to octal. We write the three bits binary code of each octal number digit.

Example : $(152.25)_8$

We write the three-bit binary digit for 1, 5, 2, and 5.

$(152.25)_8$=$(001101010.010101)_2$

So, the binary number of the octal number 152.25 is **$(001101010.010101)_2$**

Octal to hexadecimal conversion

For converting octal to hexadecimal, there are two steps required to perform, which are as follows:

1. In the first step, we will find the binary equivalent of number **25**.
2. Next, we have to make the pairs of four bits on both sides of the binary point. If there will be one, two, or three bits left in a pair of four bits pair, we add the required number of zeros on extreme sides and write the hexadecimal digits corresponding to each pair.

Example: $(152.25)_8$

Step 1:

We write the three-bit binary digit for 1, 5, 2, and 5.

$(152.25)_8$=$(001101010.010101)_2$

So, the binary number of the octal number 152.25 is **$(001101010.010101)_2$**

Step 2:

1. Now, we make pairs of four bits on both sides of the binary point.

0 0110 1010.0101 01

On the left side of the binary point, the first pair has only one digit, and on the right side, the last pair has only two-digit. To make them complete pairs of four bits, add zeros on extreme sides.

0000 0110 1010.0101 0100

2. Now, we write the hexadecimal digits, which correspond to each pair.

$(0000\ 0110\ 1010.0101\ 0100)_2$=$(6A.54)_{16}$

Hexa-decimal to other Number System

Like binary, decimal, and octal, hexadecimal numbers can also be converted into other number systems. The process of converting hexadecimal to decimal differs from the remaining one. Let's start understanding how conversion is done.

Hexa-decimal to Decimal Conversion

The process of converting hexadecimal to decimal is the same as binary to decimal. The process starts from multiplying the digits of hexadecimal numbers with its corresponding positional weights. And lastly, we add all those products.

Let's take an example to understand how the conversion is done from hexadecimal to decimal.

Example: $(152A.25)_{16}$

Step 1:

We multiply each digit of **152A.25** with its respective positional weight, and last we add the products of all the bits with its weight.

$\mathbf{(152A.25)_{16}}=(1\times16^3)+(5\times16^2)+(2\times16^1)+(A\times16^0)+(2\times16^{-1})+(5\times16^{-2})$
$\mathbf{(152A.25)_{16}}=(1\times4096)+(5\times256)+(2\times16)+(10\times1)+(2\times16^{-1})+(5\times16^{-2})$
$\mathbf{(152A.25)_{16}}=4096+1280+32+10+(2\times1/16)+(5\times1/256)$
$\mathbf{(152A.25)_{16}}=5418+0.125+0.125$
$\mathbf{(152A.25)_{16}}=5418.14453125$

So, the decimal number of the hexadecimal number 152A.25 is **5418.14453125**

Hexadecimal to Binary Conversion

The process of converting hexadecimal to binary is the reverse process of binary to hexadecimal. We write the four bits binary code of each hexadecimal number digit.

Example : $(152A.25)_{16}$

We write the four-bit binary digit for 1, 5, A, 2, and 5.

$\mathbf{(152A.25)_{16}=(0001\ 0101\ 0010\ 1010.0010\ 0101)_2}$

So, the binary number of the hexadecimal number 152.25 is $\mathbf{(1010100101010.00100101)_2}$

Hexadecimal to Octal Conversion

For converting hexadecimal to octal, there are two steps required to perform, which are as follows:

1. In the first step, we will find the binary equivalent of the hexadecimal number.

2. Next, we have to make the pairs of three bits on both sides of the binary point. If there will be one or two bits left in a pair of three bits pair, we add the required number of zeros on extreme sides and write the octal digits corresponding to each pair.

Example: $(152A.25)_{16}$

Step 1:

We write the four-bit binary digit for 1, 5, 2, A, and 5.

$(152A.25)_{16}=(0001\ 0101\ 0010\ 1010.0010\ 0101)_2$

So, the binary number of hexadecimal number 152A.25 is **$(0011010101010.010101)_2$**

Step 2:

3. Then, we make pairs of three bits on both sides of the binary point.

001 010 100 101 010.001 001 010

4. Then, we write the octal digit, which corresponds to each pair.

$(001010100101010.001001010)_2=(12452.112)_8$

So, the octal number of the hexadecimal number 152A.25 is **12452.112**

Gray Code

The **Gray Code** is a sequence of binary number systems, which is also known as **reflected binary code**. The reason for calling this code as reflected binary code is the first N/2 values compared with those of the last N/2 values in reverse order. In this code, two consecutive values are differed by one bit of binary digits. Gray codes are used in the general sequence of hardware-generated binary numbers. These numbers cause ambiguities or errors when the transition from one number to its successive is done. This code simply solves this problem by changing only one bit when the transition is between numbers is done.

The gray code is a very light weighted code because it doesn't depend on the value of the digit specified by the position. This code is also called a cyclic variable code as the transition of one value to its successive value carries a change of one bit only.

How to generate Gray code?

The prefix and reflect method are recursively used to generate the Gray code of a number. For generating gray code:

1. We find the number of bits required to represent a number.
2. Next, we find the code for 0, i.e., 0000, which is the same as binary.
3. Now, we take the previous code, i.e., 0000, and change the most significant bit of it.
4. We perform this process reclusively until all the codes are not uniquely identified.

5. If by changing the most significant bit, we find the same code obtained previously, then the second most significant bit will be changed, and so on.

Gray Code Table

Decimal Number	Binary Number	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Excess-3 Code

The excess-3 code is also treated as **XS-3 code**. The excess-3 code is a non-weighted and self-complementary BCD code used to represent the decimal numbers. This code has a biased representation. This code plays an important role in arithmetic operations because it resolves deficiencies encountered when we use the 8421 BCD code for adding two decimal digits whose sum is greater than 9. The Excess-3 code uses a special type of algorithm, which differs from the binary positional number system or normal non-biased BCD.

We can easily get an excess-3 code of a decimal number by simply adding 3 to each decimal digit. And then we write the 4-bit binary number for each digit of the decimal number. We can find the excess-3 code of the given binary number by using the following steps:

1. We find the decimal number of the given binary number.
2. Then we add 3 in each digit of the decimal number.
3. Now, we find the binary code of each digit of the newly generated decimal number.

We can also add 0011 in each 4-bit BCD code of the decimal number for getting excess-3 code.

The Excess-3 code for the decimal number is as follows:

Decimal Digit	BCD Code	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

In excess-3 code, the codes 1111 and 0000 are never used for any decimal digit. Let's take some examples of Excess-3 code.

Example: Decimal number 31

1. We find the BCD code of each digit of the decimal number.

Digit	BCD
3	0011
1	0001

2) Then, we add 0011 in both of the BCD code.

Decimal	BCD	Excess-3
3	0011+0011	0110

1	0001+0011	0100

3. So, the excess-3 code of the decimal number 31 is **0110 0100**

Example: Decimal number 81.61

1. We find the BCD code of each digit of the decimal number.

Digit	BCD
8	1000
1	0001
6	0110
1	0001

2) Then, we add 0011 in both of the BCD code.

Decimal	BCD	Excess-3
8	1000+0011	1011
1	0001+0011	0100
6	0110+0011	1001

3) So, the excess-3 code of the decimal number 81.61 is **1011 0100.1001 0100**

ASCII CODE

The ASCII stands for American Standard Code for Information Interchange. The ASCII code is an alphanumeric code used for data communication in digital computers. The ASCII is a 7-bit code capable of representing 2^7 or 128 number of different characters. The ASCII code is made up of a three-bit group, which is followed by a four-bit code.

- The ASCII Code is a 7 or 8-bit alphanumeric code.
- This code can represent 127 unique characters.
- The ASCII code starts from 00h to 7Fh. In this, the code from 00h to 1Fh is used for control characters, and the code from 20h to 7Fh is used for graphic symbols.
- The 8-bit code holds ASCII, which supports 256 symbols where math and graphic symbols are added.
- The range of the extended ASCII is 80h to FFh.

Basic Gates

The basic gates are AND, OR & NOT gates.

AND gate

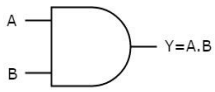
An AND gate is a digital circuit that has two or more inputs and produces an output, which is the **logical AND** of all those inputs. It is optional to represent the **Logical AND** with the symbol '.'.

The following table shows the **truth table** of 2-input AND gate.

A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Here A, B are the inputs and Y is the output of two input AND gate. If both inputs are '1', then only the output, Y is '1'. For remaining combinations of inputs, the output, Y is '0'.

The following figure shows the **symbol** of an AND gate, which is having two inputs A, B and one output, Y.



This AND gate produces an output (Y), which is the **logical AND** of two inputs A, B. Similarly, if there are 'n' inputs, then the AND gate produces an output, which is the logical AND of all those inputs. That means, the output of AND gate will be '1', when all the inputs are '1'.

OR gate

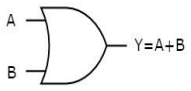
An OR gate is a digital circuit that has two or more inputs and produces an output, which is the logical OR of all those inputs. This **logical OR** is represented with the symbol '+'.

The following table shows the **truth table** of 2-input OR gate.

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Here A, B are the inputs and Y is the output of two input OR gate. If both inputs are '0', then only the output, Y is '0'. For remaining combinations of inputs, the output, Y is '1'.

The following figure shows the **symbol** of an OR gate, which is having two inputs A, B and one output, Y.



This OR gate produces an output (Y), which is the **logical OR** of two inputs A, B. Similarly, if there are 'n' inputs, then the OR gate produces an output, which is the logical OR of all those inputs. That means, the output of an OR gate will be '1', when at least one of those inputs is '1'.

NOT gate

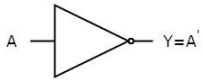
A NOT gate is a digital circuit that has single input and single output. The output of NOT gate is the **logical inversion** of input. Hence, the NOT gate is also called as inverter.

The following table shows the **truth table** of NOT gate.

A	Y = A'
0	1
1	0

Here A and Y are the input and output of NOT gate respectively. If the input, A is '0', then the output, Y is '1'. Similarly, if the input, A is '1', then the output, Y is '0'.

The following figure shows the **symbol** of NOT gate, which is having one input, A and one output, Y.



This NOT gate produces an output (Y), which is the **complement** of input, A.

Universal gates

NAND & NOR gates are called as **universal gates**. Because we can implement any Boolean function, which is in sum of products form by using NAND gates alone. Similarly, we can implement any Boolean function, which is in product of sums form by using NOR gates alone.

NAND gate

NAND gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical AND** of all those inputs.

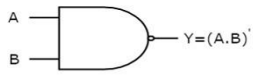
The following table shows the **truth table** of 2-input NAND gate.

A	B	Y = (A.B)'
0	0	1

0	1	1
1	0	1
1	1	0

Here A, B are the inputs and Y is the output of two input NAND gate. When both inputs are '1', the output, Y is '0'. If at least one of the input is zero, then the output, Y is '1'. This is just opposite to that of two input AND gate operation.

The following image shows the **symbol** of NAND gate, which is having two inputs A, B and one output, Y.



NAND gate operation is same as that of AND gate followed by an inverter. That's why the NAND gate symbol is represented like that.

NOR gate

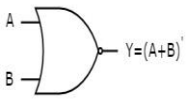
NOR gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical OR** of all those inputs.

The following table shows the **truth table** of 2-input NOR gate

A	B	Y = (A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

Here A, B are the inputs and Y is the output. If both inputs are '0', then the output, Y is '1'. If at least one of the input is '1', then the output, Y is '0'. This is just opposite to that of two input OR gate operation.

The following figure shows the **symbol** of NOR gate, which is having two inputs A, B and one output, Y.



NOR gate operation is same as that of OR gate followed by an inverter. That's why the NOR gate symbol is represented like that.

REVIEW QUESTIONS

1) Define Ascii code.

2) Define gray code.
3) Explain binary to decimal conversion.
4) Define hexadecimal.
5) Explain about universal logic gates.
6) Define excess-3 code.
7) Define AND gate.
8) Explain binary to octal conversion.