

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

NUMBER SYSTEM

A Number system is a method of showing numbers by writing, which is a mathematical way of representing the numbers of a given set, by using the numbers or symbols in a mathematical manner. The writing system for denoting numbers using digits or symbols in a logical manner is defined as a **Number system**.

Types of Number Systems

Based on the base value and the number of allowed digits, number systems are of many types. The four common types of Number System are:

- **Binary Number System**
- **Decimal Number System**
- **Octal Number System**
- **Hexadecimal Number System**

Binary Number System

Number System with base value 2 is termed as Binary number system. It uses 2 digits i.e. 0 and 1 for the creation of numbers. The numbers formed using these two digits are termed Binary Numbers. The binary number system is very useful in electronic devices and computer systems because it can be easily performed using just two states ON and OFF i.e. 0 and 1.

Decimal Numbers 0-9 are represented in binary as: 0, 1, 10, 11, 100, 101, 110, 111, 1000, and 1001

For example, 14 can be written as 1110, 19 can be written as 10011, 50 can be written as 110010.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Example of 19 in the binary system

2	19	
2	9	1
2	4	1
2	2	0
2	1	0
2	0	1

Here 19 can be written as 10011

advantages

Logic operations are the backbone of any digital computer, although solving a problem on computer could involve an arithmetic operation too. The introduction of the mathematics of logic by George Boole laid the foundation for the modern digital computer. He reduced the mathematics of logic to a binary notation of '0' and '1'.

Another advantage of this number system was that all kind of data could be conveniently represented in terms of 0s and 1s.

Also basic electronic devices used for hardware implementation could be conveniently and efficiently operated in two distinctly different modes.

Decimal Number System

Number system with a base value of 10 is termed a Decimal number system. It uses 10 digits i.e. 0-9 for the creation of numbers. Here, each digit in the number is at a specific place with place value a product of different powers of 10. Here, the place value is **termed from right to left as first place value called units, second to the left as Tens, so on Hundreds, Thousands, etc.** Here, units have the place value as 10⁰, tens have the place value as 10¹, hundreds as 10², thousands as 10³, and so on.

For example, 10264 has place values as,

$$\begin{aligned} & (1 \times 10^4) + (0 \times 10^3) + (2 \times 10^2) + (6 \times 10^1) + (4 \times 10^0) \\ & = 1 \times 10000 + 0 \times 1000 + 2 \times 100 + 6 \times 10 + 4 \times 1 \\ & = 10000 + 0 + 200 + 60 + 4 \\ & = 10264 \end{aligned}$$

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Octal Number System

Octal Number System is one in which the base value is 8. It uses 8 digits i.e. 0-7 for the creation of Octal Numbers. Octal Numbers can be converted to Decimal values by multiplying each digit with the place value and then adding the result. Here the place values are 8^0 , 8^1 , and 8^2 . Octal Numbers are useful for the representation of UTF8 Numbers. Example,

$(135)_{10}$ can be written as $(207)_8$

$(215)_{10}$ can be written as $(327)_8$

Hexadecimal Number System

Number System with base value 16 is termed as Hexadecimal Number System. It uses 16 digits for the creation of its numbers. **Digits from 0-9 are taken like the digits in the decimal number system but the digits from 10-15 are represented as A-F i.e. 10 is represented as A, 11 as B, 12 as C, 13 as D, 14 as E, and 15 as F.** Hexadecimal Numbers are useful for handling memory address locations. The hexadecimal number system provides a condensed way of representing large binary numbers stored and processed. Examples,

$(255)_{10}$ can be written as $(FF)_{16}$

$(1096)_{10}$ can be written as $(448)_{16}$

$(4090)_{10}$ can be written as $(FFA)_{16}$

HEXADECIMAL	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DECIMAL	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Sample Problems

Question 1: Convert $(19)_{10}$ as a binary number?

Solution:

2	19	1
2	9	1
2	4	0
2	2	0
	1	

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Collect remainders from bottom to top we will get 10011 which is binary representation of decimal number 10.

Therefore $(19)_{10} = (10011)_2$

Question 2: Convert 325_8 into a decimal?

Solution:

$$325_8 = 3 \times 8^2 + 2 \times 8^1 + 5 \times 8^0$$

$$= 3 \times 64 + 2 \times 8 + 5 \times 1$$

$$= 192 + 16 + 5$$

$$= 213_{10}$$

Question 3: Convert $(2056)_{16}$ into an octal number?

Solution:

Here $(2056)_{16}$ is in hexadecimal form

First we will convert into decimal form from hexadecimal.

$$(2056)_{16} = 2 \times 16^3 + 0 \times 16^2 + 5 \times 16^1 + 6 \times 16^0$$


$$= 2 \times 4096 + 0 + 80 + 6$$

$$= 8192 + 0 + 80 + 6$$

$$= (8278)_{10}$$

Now convert this decimal number into octal number by dividing it by 8

8	8278	6
8	1034	2
8	129	1
8	16	0
	2	



So will take the value of remainder from 20126

$$(8278)_{10} = (20126)_8$$

$$\text{Therefore, } (2056)_{16} = (20126)_8$$

Question 4: Convert $(101110)_2$ into octal number.

Solution:

Given $(101110)_2$ a binary number, to convert it into octal number

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

OCTAL NUMBER	BINARY NUMBER
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Using the above table we can write given number as,

101 110 i.e.

101 = 5

110 = 6

So (101110)₂ in octal number is (56)₈

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Binary Operations:

Binary Addition is a base-2 number system that uses two states 0 and 1 to represent a number. We can also call it to be a true state and a false state. A binary number is built the same way as we build a normal decimal number.

Binary arithmetic is an essential part of various digital systems. You can add, subtract, multiply, and divide binary numbers using various methods. These operations are much easier than decimal number arithmetic operations because the binary system has only two digits: 0 and 1.

Binary additions and subtractions are performed as same in decimal additions and subtractions. When we perform binary additions, there will be two outputs: Sum (S) and Carry (C).

1. There are four rules for binary addition:

Input A	Input B	Sum (S) A+B	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2. There are four rules for binary subtraction:

Input A	Input B	Subtract (S) A-B	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

3. There are four rules for binary multiplication:

Multiplication is always 0, whenever at least one input is 0.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Input A	Input B	Multiply (M) AxB
0	0	0
0	1	0
1	0	0
1	1	1

There are four parts in any division: Dividend, Divisor, quotient, and remainder.

4. There are four parts in any division: Dividend, Divisor, quotient, and remainder.

Input A	Input B	Divide (D) A/B
0	0	Not defined
0	1	0
1	0	Not defined
1	1	1

The result is always not defined, whenever the divisor is 0

Binary arithmetic is essential part of all the digital computers and many other digital system.

Addition Binary Numbers

Rule to follow when binary numbers are added:

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$ (with a carry of 1)

Examples - Adding Binary and Decimal Numbers

	Decimal	Binary
--	---------	--------

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

	Decimal	Binary
1	$\begin{array}{r} 5 \\ + 6 \\ \hline = 11 \end{array}$	$\begin{array}{r} 101 \\ + 110 \\ \hline = 1011 \end{array}$
2	$\begin{array}{r} 6 \\ + 6 \\ \hline = 12 \end{array}$	$\begin{array}{r} 110 \\ + 110 \\ \hline = 1100 \end{array}$
3	$\begin{array}{r} 15 \\ + 20 \\ \hline = 35 \end{array}$	$\begin{array}{r} 1111 \\ + 10100 \\ \hline = 100011 \end{array}$
4	$\begin{array}{r} 3.25 \\ + 5.75 \\ \hline = 9.00 \end{array}$	$\begin{array}{r} 11.01 \\ + 101.11 \\ \hline = 1001.00 \end{array}$

Subtracting Binary Numbers

Rule to follow when binary numbers are subtracted:

- $0 - 0 = 0$
- $1 - 0 = 1$
- $1 - 1 = 0$
- $0 - 1 = 1$ (with a borrow of 1)

Examples - Subtracting Binary and Decimal Numbers

	Decimal	Binary
1	$\begin{array}{r} 9 \\ - 5 \\ \hline = 4 \end{array}$	$\begin{array}{r} 1001 \\ - 101 \\ \hline = 0100 \end{array}$

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

	Decimal	Binary
2	$\begin{array}{r} 16 \\ - 3 \\ \hline = 13 \end{array}$	$\begin{array}{r} 10000 \\ - 11 \\ \hline = 01101 \end{array}$
3	$\begin{array}{r} 6.25 \\ - 4.50 \\ \hline = 1.75 \end{array}$	$\begin{array}{r} 110.01 \\ - 100.10 \\ \hline = 001.11 \end{array}$

Multiplication of Binary Numbers

Rule to follow when binary numbers are multiplied:

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$

Examples of Binary Multiplication

Some binary multiplication examples are given below for a better understanding of this concept.

Example 1: Solve 1010×101

Solution:

1010×101

$$\begin{array}{r} 1010 \\ (\times) 101 \\ \hline 1010 \\ 0000 \\ \hline \end{array}$$

01010 First Intermediate Sum

1010

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

110010 ← -----result

Comparison with Decimal values:

$$1010_2 = 10_{10}$$

$$1010_2 = 5_{10}$$

$$10 \times 5 = 50_{10}$$

$$(110010)_2 = 50_{10}$$

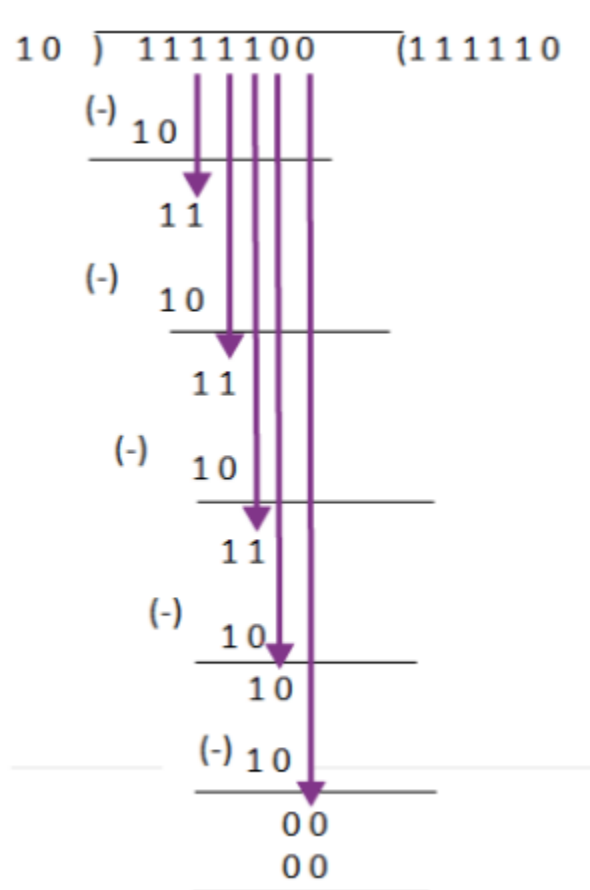
Division of Binary Numbers

Rule to follow when binary numbers are divided:

- $0 / 1 = 0$
- $1 / 1 = 1$

Example divide 1111100 by 10

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.



GRAY CODE Gray code has a property that **two successive numbers differ in only one bit** because of this property gray code does the cycling through various states with minimal effort and is used in K-maps, error correction, communication, etc.

For example, the sequence of Gray codes for 3-bit numbers is: 000, 001, 011, 010, 110, 111, 101, 100, so $G(4)=6$.

How to generate n bit Gray Codes?

Following is 2-bit sequence ($n = 2$)

00 01 11 10

Following is 3-bit sequence ($n = 3$)

000 001 011 010 110 111 101 100

And Following is 4-bit sequence ($n = 4$)

0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111
1110 1010 1011 1001 1000

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

n-bit Gray Codes can be generated from a list of (n-1)-bit Gray codes using the following steps.

1. Let the list of (n-1)-bit Gray codes be L1. Create another list L2 which is the reverse of L1.
2. Modify the list L1 by prefixing a '0' in all codes of L1.
3. Modify the list L2 by prefixing a '1' in all codes of L2.
4. Concatenate L1 and L2. The concatenated list is the required list of n-bit Gray codes.

How to Convert Binary To Gray and Vice Versa?

Binary : 0011

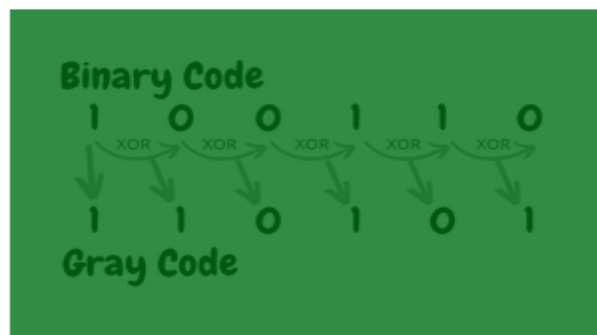
Gray : 0010

Binary : 01001

Gray : 01101

BINARY TO GRAY CONVERSION :

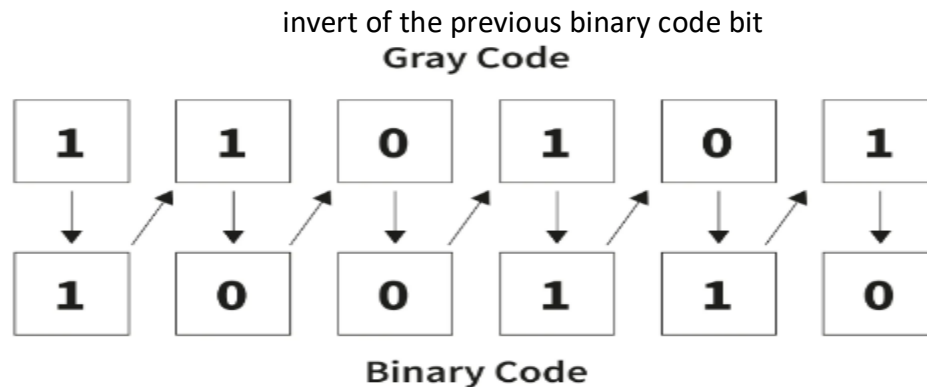
1. The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
2. Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.



GRAY TO BINARY CONVERSION :

1. The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given gray code.
2. Other bits of the output binary code can be obtained by checking the gray code bit at that index. If the current gray code bit is 0, then copy the previous binary code bit, else copy the

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.



BCD CODE:

The BCD equivalent of a decimal number is written by replacing each decimal digit in the integer and fractional parts with its four bit binary equivalent. The BCD code is more precisely known as 8421 BCD code, with 8, 4, 2 and 1 representing the weights of different bits in the four-bit groups, Starting from MSB and proceeding towards LSB. This feature makes it a weighted code, which means that each bit in the four bit group representing a given decimal digit has an assigned weight.

Many decimal values, have an infinite place-value representation in binary but have a finite place-value in binary-coded decimal. For example, 0.2 in binary is .001100... and in BCD is 0.0010. It avoids fractional errors and is also used in huge financial calculations.

Consider the following truth table and focus on how are these represented.

Truth Table for Binary Coded Decimal

DECIMAL NUMBER	BCD
0	0000
1	0001
2	0010
3	0011

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

DECIMAL NUMBER	BCD
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

1. Convert (123)₁₀ in BCD

From the truth table above,

1 -> 0001

2 -> 0010

3 -> 0011

thus, BCD becomes -> 0001 0010 0011

2. Convert (324)₁₀ in BCD

(324)₁₀ -> 0011 0010 0100 (BCD)

Again from the truth table above,

3 -> 0011

2 -> 0010

4 -> 0100

thus, BCD becomes -> 0011 0010 0100

This is how decimal numbers are converted to their equivalent BCDs.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

EXCESS-3 CODE

The excess-3 code (or XS3) is a non-weighted code used to express code used to express decimal numbers. It is a self-complementary binary coded decimal (BCD) code and numerical system which has biased representation. It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9. Excess-3 arithmetic uses different algorithm than normal non-biased BCD or binary positional number system.

Representation of Excess-3 Code

Excess-3 codes are unweighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit. An Excess-3 equivalent of a given binary number is obtained using the following steps:

- Find the decimal equivalent of the given binary number.
- Add +3 to each digit of decimal number.
- Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.

You can add 0011 to each four-bit group in binary coded decimal number (BCD) to get desired excess-3 equivalent.

These are following excess-3 codes for decimal digits –

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

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Decimal Digit	BCD Code	Excess-3 Code
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

The codes 0000 and 1111 are not used for any digit.

Example – Convert decimal number 23 to Excess-3 code.

So, according to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

= $23+33=56$ = 0101 0110 which is required excess-3 code for given decimal number 23.

Example – Convert decimal number 15.46 into Excess-3 code.

According to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

= $15.46+33.33=48.79$ = 0100 1000.0111 1001 which is required excess-3 code for given decimal number 15.46.

Converting into Binary Coded Decimal (BCD) codes

One should note that to given Excess-3 code, the equivalent decimal number can be determined by splitting number into 4-bit group starting from least significant for integer part and from leftmost digit for fractional part. Then subtract 0011 (=3) from each four-bit group that will be binary decimal digit (BCD) form of that number. Now you can also convert this BCD code into decimal number by converting each 4-bit group into decimal digit.

Example – Convert Excess-3 code 1001001 into BCD and decimal number.

So, grouping 4-bit for each group, i.e., 0100 1001 and subtract 0011 0011 from given number. Therefore,

= $0100\ 1001 - 0011\ 0011 = 0001\ 0110$

So, binary coded decimal number is 0001 0110 and decimal number will be 16.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Self-complementary property

Excess-3 code is non-weighted and self complementary code. A self complementary binary codes are always complement themselves. The complement of a binary number can be obtained from that number by replacing 0's with 1's and 1's with 0's. The sum of binary number and its complement is always equal to decimal 9. In other words, **the 1's complement of an excess-3 code is the excess-3 code for the 9's complement of the corresponding decimal number.**

For example, the excess-3 code for decimal number 5 is 1000 and 1's complement of 1000 is 0111, which is excess-3 code for decimal number 4, and it is 9's complement of number 5.

Advantages of Excess-3 Codes

These are following advantages of Excess-3 codes,

- These are unweighted binary decimal codes.
- These are self-complementary codes.
- These use biased representation.
- The codes 0000 and 1111 are not used for any digit which is an advantage for memory organization as these codes can cause fault in transmission line.
- It has no limitation, and it considerably simplifies arithmetic operations.
- It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds

ASCII CODE.

ASCII (American Standard Code for Information Interchange) is the most common character encoding format for text data in computers and on the internet. In standard ASCII-encoded data, there are unique values for 128 alphabetic, numeric or special additional characters and control codes.

ASCII characters may be represented in the following ways:

- as pairs of hexadecimal digits -- base-16 numbers, represented as 0 through 9 and A through F for the decimal values of 10-15;
- as three-digit octal (base 8) numbers;
- as decimal numbers from 0 to 127; or

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

- as 7-bit or 8-bit binary

For example, the ASCII encoding for the lowercase letter "m" is represented in the following ways:

Character	Hexadecimal	Octal	Decimal	Binary (7 bit)	Binary (8 bit)
m	0x6D	/155	109	110 1101	0110 1101

ASCII characters were initially encoded into 7 bits and stored as 8-bit characters with the most significant bit -- usually, the left-most bit -- set to 0.

Why is ASCII important?

ASCII was the first major character encoding standard for data processing. Most modern computer systems use Unicode, also known as the Unicode Worldwide Character Standard. It's a character encoding standard that includes ASCII encodings.

The Internet Engineering Task Force (IETF) adopted ASCII as a standard for internet data when it published "ASCII format for Network Interchange" as RFC 20 in 1969. That request for comments (RFC) document standardized the use of ASCII for internet data and was accepted as a full standard in 2015.

ASCII encoding is technically obsolete, having been replaced by Unicode. Yet, ASCII characters use the same encoding as the first 128 characters of the Unicode Transformation Format 8, so ASCII text is compatible with UTF-8.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

PARITY CODE

The parity code is used for the purpose of detecting errors during the transmission of binary information. The parity code is a bit that is included with the binary data to be transmitted.

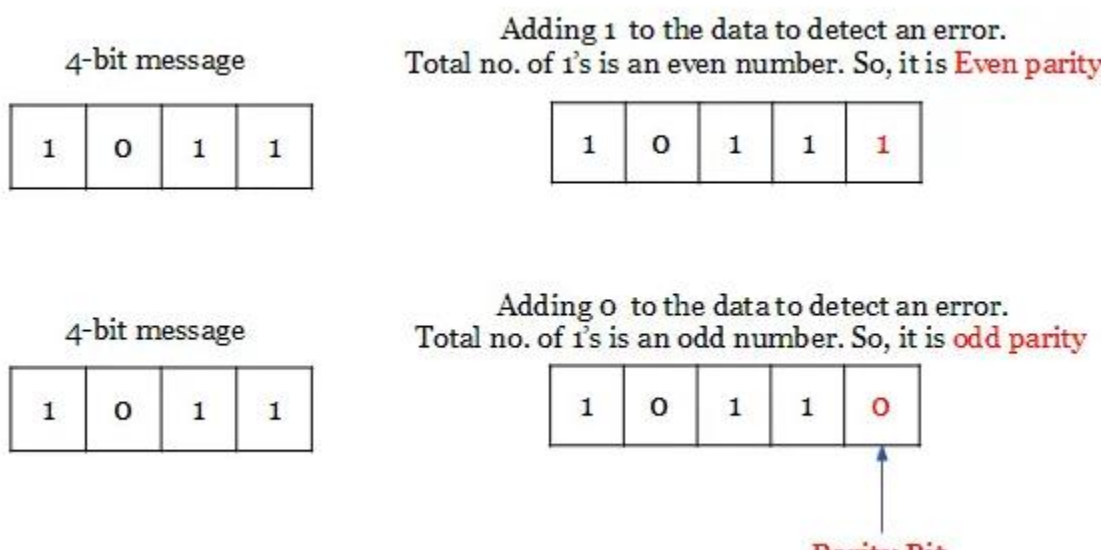
The inclusion of a parity bit will make the number of 1's either odd or even. Based on the number of 1's in the transmitted data, the parity code is of two types.

- Even parity code
- Odd parity code

In even parity, the added parity bit will make the total number of 1's an even number. If the added parity bit make the total number of 1's as odd number, such parity code is said to be odd parity code.

Let us consider the 4-bit message(1011) to be transmitted. Adding 1 to the message will make the total number of 1's in the message to be an even number. Hence it is called as even parity.

For the same message, adding 0 with the transmitted message will make the total number of 1's to be an odd number. Hence it is called as odd parity. It is shown in the example below.



4-bit message	Message with Odd parity	Message with Even Parity
---------------	-------------------------	--------------------------

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

4-bit message	Message with Odd parity	Message with Even Parity
0000	00001	00000
0001	00010	00011
0010	00100	00101
0011	00111	00110
0100	01000	01001
0101	01011	01010
0110	01101	01100
0111	01110	01111

SIGN MAGNITUDE REPRESENTATION OF BINARY NUMBER

- Using Sign magnitude: The MSB is used for representing sign of the number and the remaining bits represent the magnitude of the number.

Range : $-2^{(n-1)} + 1$ to $+2^{(n-1)} - 1$

if $n=5$ range will be from -15 to +15

- positive signed binary numbers
- Negative Signed Binary Numbers



SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

SIGN MAGNITUDE ADDITION

SIGN MAGNITUDE ADDITION ALGORITHM(P+Q)

- ▶ When the signs of P and Q are equal, add the two magnitudes and connect the sign of P to the output.
- ▶ When the signs of P and Q are different, compare the magnitudes and subtract the smaller number from the greater number.
 - ▶ The signs of the output have to be equal as P in case $P > Q$ or the complement of the sign of Q in case $P < Q$.
 - ▶ When the two magnitudes are equal, subtract Q from P and modify the sign of the output to positive.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

SIGN MAGNITUDE SUBTRACTION

SIGN MAGNITUDE SUBTRACTION ALGORITHM(P-Q)

- ▶ When the signs of P and Q are different, add the two magnitudes and connect the signs of P to the output.
- ▶ When the signs of P and Q are the same, compare the magnitudes and subtract the smaller number from the greater number.
 - ▶ The signs of the output have to be equal as P in case $P > Q$ or the complement of the sign of Q in case $P < Q$.
 - ▶ When the two magnitudes are equal, subtract Q from P and modify the sign of the output to positive.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

BCD ARITHMETIC

BCD SUBTRACTION USING 9'S COMPLEMENT:

► For A-B

1. Take 9's complement for B
2. Add it to A using BCD addition
3. If addition is invalid BCD then add 6
4. If carry then add it to the next bits
5. In final result, if carry is occurred then add it the remaining result and if there is no any carry over, then take 9's complement of the result and it is negative.

- Example: subtract 216 from 541 using 9's complement method.

BCD ARITHMETIC

BCD SUBTRACTION USING 10'S COMPLEMENT:

For A-B

1. Take 10's complement for B
2. Add it to A using BCD addition
3. If addition is invalid BCD then add 6
4. If carry then add it to the next bits
5. In final result, if carry is occurred then it is ignored and if there is no any carry over, then take 10's complement of the result and it is negative.

- Example: subtract 216 from 541 using 10's complement method.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

1'S COMPLEMENT OF A BINARY NUMBER:

There is a simple algorithm to convert a binary number into 1's complement. To get 1's complement of a binary number, simply invert the given number. You can simply implement logic circuit using only NOT gate for each bit of Binary number input. Implementation of logic circuit of 4-bit 1's complement is given as following below.

Example-1: Find 1's complement of binary number 10101110.

Simply invert each bit of given binary number, so 1's complement of given number will be 01010001.

Example-2: Find 1's complement of binary number 10001.001.

Simply invert each bit of given binary number, so 1's complement of given number will be 01110.110.

Example-3: Find 1's complement of each 3 bit binary number.

Simply invert each bit of given binary number, so 1's complement of each 3 bit binary number will be,

Binary number	1's complement
000	111
001	110
010	101
011	100
100	011
101	010
110	001

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Binary number	1's complement
111	000

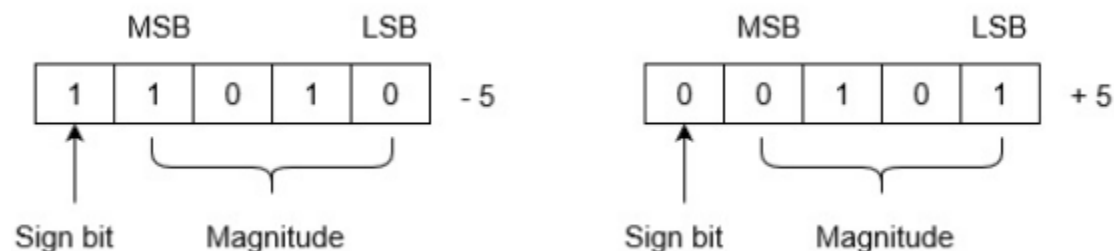
Uses of 1's Complement Binary Numbers:

There are various uses of 1's complement of Binary numbers, mainly in signed Binary number representation and various arithmetic operations for Binary numbers, e.g., additions, subtractions, etc.

1's Complement in Signed Binary number Representation:

1's complement binary numbers are very useful in Signed number representation. Positive numbers are simply represented as Binary number. There is nothing to do for positive binary number. But in case of negative binary number representation, we represent in 1's complement. If the number is negative then it is represented using 1's complement. First represent the number with positive sign and then take 1's complement of that number.

Example: Let we are using 5 bits register. The representation of -5 and +5 will be as follows:



+5 is represented as it is represented in sign magnitude method. -5 is represented using the following steps:

(i) $+5 = 0\ 0101$

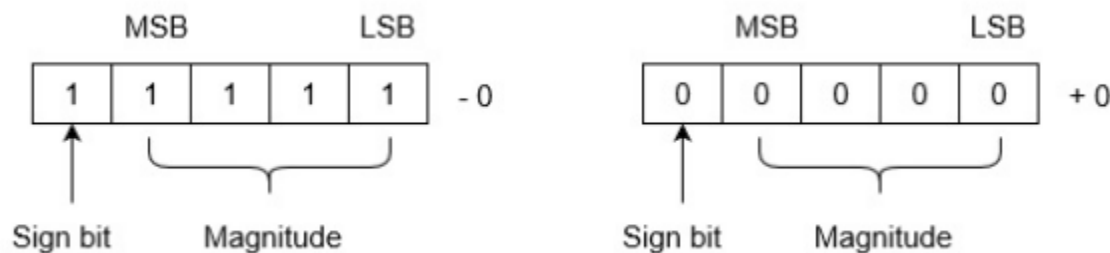
(ii) Take 1's complement of 0 0101 and that is 1 1010. MSB is 1 which indicates that number is negative.

MSB is always 1 in case of negative numbers.

Range of Numbers: For k bits register, positive largest number that can be stored is $(2^{(k-1)} - 1)$ and negative lowest number that can be stored is $-(2^{(k-1)} - 1)$.

Note that drawback of this system is that 0 has two different representation one is -0 (e.g., 1 1111 in five bit register) and second is +0 (e.g., 0 0000 in five bit register).

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.



Lets see arithmetic operations: Subtractions and Additions in 1's complement binary numbers.

Subtractions by 1's Complement:

The algorithm to subtract two binary number using 1's complement is explained as following below:

- Take 1's complement of the subtrahend
- Add with minuend
- If the result of above addition has carry bit 1, then add it to the least significant bit (LSB) of given result
- If there is no carry bit 1, then take 1's complement of the result which will be negative

Note that subtrahend is number that to be subtracted from the another number, i.e., minuend.

Example (Case-1: When Carry bit 1): Evaluate $10101 - 00101$

According to above algorithm, take 1's complement of subtrahend 00101, which will be 11010, then add both of these. So, $10101 + 11010 = 1\ 01111$. Since, there is carry bit 1, so add this to the LSB of given result, i.e., $01111 + 1 = 10000$ which is the answer.

Example (Case-2: When no Carry bit): Evaluate 11110 with 1110

According to above algorithm, take 1's complement of subtrahend 11110, which will be 00011. Then add both of these, So, $11001 + 00011 = 11100$. Since there is no carry bit 1, so take 1's complement of above result, which will be 00011, and this is negative number, i.e., 00011, which is the answer.

Similarly, you can subtract two mixed (with fractional part) binary numbers. Note that you always add Carry bit the the least significant bit (LSB) of the result, whenever you get carry bit 1. LSB of fractional binary number is last (rightmost) bit of mixed or fractional binary numbers.

Additions by 1's Complement:

There are difference scenario for addition of two binary numbers using 1's complement. These are explained as following below.

Case-1: Addition of positive and negative number when positive number has greater magnitude:

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

When positive number has greater magnitude, then take simply 1's complement of negative number and the end-around carry of the sum is added to the least significant bit (LSB).

Example: Add 1110 and -1101.

So, take 1's complement of 1101, which will be 0010, then add with given number. So, $1110 + 0010 = 1\ 0000$, then add this carry bit to the LSB, $0000 + 1 = 0001$, which is the answer.

Note that if the register size is big then fill the same value of MSB to preserve sign magnitude for inputs and output.

Case-2: Addition of positive and negative number when negative number has greater magnitude:

When the negative number has greater magnitude, then take 1's complement of negative number and add with given positive number. Since there will not be any end-around carry bit, so take 1's complement of the result and this result will be negative.

Example: Add 1010 and -1100 in five-bit registers.

Note that there are five-bit registers, so these new numbers will be 01010 and -01100. Now take 1's complement of 01100 which will be 10011 and add $01010 + 10011 = 11101$. Then take 1's complement of this result, which will be 00010 and this will be negative number, i.e., -00010, which is the answer.

Case-3: Addition of two negative numbers:

You need to take 1's complement for both numbers, then add these 1's complement of numbers. Since there will always be end-around carry bit, so add this again to the MSB of result. Now, take 1's complement also of previous result, so this will be negative number.

Alternatively, you can add both negative number directly, and get this result which will be negative only.

Example: add -1010 and -0101 in five bit-register.

These five bit numbers are -01010 and -00101. Add complements of these numbers, $10101 + 11010 = 1\ 01111$. Since, there is a carry bit 1, so add this to the LSB of result, i.e., $01111 + 1 = 10000$. Now take the 1's complement of this result, which will be 01111 and this number is negative, i.e., -01111, which is answer.

Note that *end-around-carry-bit* addition occurs only in 1's complement arithmetic operations but not in 2's complement arithmetic operations.

TWO'S COMPLEMENT 2's complement of binary number is 1's complement of given number plus 1 to the least significant bit (LSB). For example 2's complement of binary number 10010 is $(01101) + 1 = 01110$.

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

2's Complement of a Binary Number

There is a simple algorithm to convert a binary number into 2's complement. To get 2's complement of a binary number, simply invert the given number and add 1 to the least significant bit (LSB) of given result. Implementation of 4-bit 2's complementation number is given as following below.

Example-1 – Find 2's complement of binary number 10101110.

Simply invert each bit of given binary number, which will be 01010001. Then add 1 to the LSB of this result, i.e., $01010001 + 1 = 01010010$ which is answer.

Example-2 – Find 2's complement of binary number 10001.001.

Simply invert each bit of given binary number, which will be 01110.110 Then add 1 to the LSB of this result, i.e., $01110.110 + 1 = 01110.111$ which is answer.

Example-3 – Find 2's complement of each 3 bit binary number.

Simply invert each bit of given binary number, then add 1 to LSB of these inverted numbers,

Binary number	1's complement	2's complement
000	000 (+0) and 111 (-0)	000
001	110	111
010	101	110
011	100	101
100	011	100
101	010	011
110	001	010
111	000	001

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Uses of 2's Complement Binary Numbers

Characteristics of 2's complement no.s:

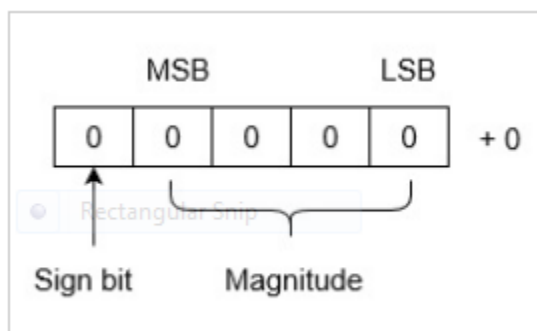
1. There is one unique zero
2. 2's comp of 0 is 0
3. The leftmost bit can't be used to express a quantity . it is a 0 no. is +ve.
4. For an n-bit word which includes the sign bit there are $(2^{n-1}-1)$ +ve integers, 2^{n-1} -ve integers & one 0 , for a total of 2^n unique states.
5. Significant information is contained in the 1's of the +ve no.s & 0's of the -ve no.s
6. A -ve no. may be converted into a +ve no. by finding its 2's comp.

2's Complementation in Signed Binary number Representation

Positive numbers are simply represented as simple Binary representation. But if the number is negative then it is represented using 2's complement. First represent the number with positive sign and then take 2's complement of that number.

Range of Numbers –For k bits register, positive largest number that can be stored is $(2^{(k-1)}-1)$ and negative lowest number that can be stored is $-(2^{(k-1)})$.

The advantage of this system is that 0 has only one representation for -0 and +0. Zero (0) is considered as always positive (sign bit is 0) in 2's complement representation. Therefore, it is unique or unambiguous representation.



Lets see arithmetic operations: Subtractions and Additions in 2's complement binary numbers.

Subtractions by 2's Complement

The algorithm to subtract two binary number using 2's complement is explained as following below –

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

- Take 2's complement of the subtrahend
- Add with minuend
- If the result of above addition has carry bit 1, then it is dropped and this result will be positive number.
- If there is no carry bit 1, then take 2's complement of the result which will be negative

Note that subtrahend is number that to be subtracted from the another number, i.e., minuend.

Also, note that adding *end-around carry-bit* occurs only in 1's complement arithmetic operations but not 2's complement arithmetic operations.

Example (Case-1: When Carry bit 1) –Evaluate $10101 - 00101$

According to above algorithm, take 2's complement of subtrahend 00101, which will be 11011, then add both of these. So, $10101 + 11011 = 1\ 10000$. Since, there is carry bit 1, so dropped this carry bit 1, and take this result will be 10000 will be positive number.

Example (Case-2: When no Carry bit) –Evaluate $11001 - 11100$

According to above algorithm, take 2's complement of subtrahend 11100, which will be 00100. Then add both of these, So, $11001 + 00100 = 11101$. Since there is no carry bit 1, so take 2's complement of above result, which will be 00011, and this is negative number, i.e., 00011, which is the answer.

similarly, you can subtract two mixed (with fractional part) binary numbers.

ADDITIONS BY 2'S COMPLEMENT –

There are different scenario for addition of two binary numbers using 2's complement. These are explained as following below.

Case-1 – Addition of positive and negative number when positive number has greater magnitude:

When positive number has greater magnitude, then take simply 2's complement of negative number and carry bit 1 is dropped and this result will be positive number.

Example –Add 1110 and -1101.

So, take 2's complement of 1101, which will be 0011, then add with given number. So, $1110 + 0011 = 1\ 0001$, and carry bit 1 is dropped and this result will be positive number, i.e., +0001.

Note that if the register size is big then use sign extension method of MSB bit to preserve sign of number.

Case-2 – Addition of positive and negative number when negative number has greater magnitude –

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

When the negative number has greater magnitude, then take 2's complement of negative number and add with given positive number. Since there will not be any end-around carry bit, so take 2's complement of the result and this result will be negative.

Example – Add 1010 and -1100 in five-bit registers.

Note that there are five-bit registers, so these new numbers will have 01010 and -01100. Now take 2's complement of 01100 which will be 10100 and add $01010 + 10100 = 11110$. Then take 2's complement of this result, which will be 00010 and this will be negative number, i.e., -00010, which is the answer.

Case-3 – Addition of two negative numbers –

You need to take 2's complement for both numbers, then add these 2's complement of numbers. Since there will always be end-around carry bit, so it is dropped. Now, take 2's complement also of previous result, so this will be negative number.

Alternatively, you can add both of these Binary numbers and take result which will be negative only.

Example – add -1010 and -0101 in five bit-register.

These five bit numbers are -01010 and -00101. Add 2's complements of these numbers, $10110 + 11011 = 1\ 10001$. Since, there is a carry bit 1, so it is dropped. Now take the 2's complement of this result, which will be 01111 and this number is negative, i.e., -01111, which is answer.

Note that 2's complement arithmetic operations are much easier than 1's complement because of there is no addition of *end-around-carry-bit*.

BINARY CODED DECIMAL ADDITION (BCD ADDITION)

The **addition of BCD numbers** is slightly different from **binary addition**. Here, the rules of binary addition are partially applicable only to the individual 4-bit groups. The **BCD addition**, is thus carried out by individually adding the corresponding 4-bit groups starting from the LSB side and if there is a carry to the next group, or if the result belongs to any of the 6 illegal states then we add **6₁₀(0110)** to the sum term of that group and resulting carry is added in the next group.

Example: Perform BCD Addition of 6 and 7

Solution

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

BCD representation of **6** is given as **0110** and for **7** it is **0111**.

When we add **6** and **7** in **BCD**, we get **1101** which is an invalid state therefore, we add **0110 (6)** to the sum to get correct result which is **0001 0011 (13)**.

The diagram illustrates the BCD addition of 6 and 7. It shows the initial sum 1101, which is labeled as an 'Illegal Form'. To correct this, 0110 (the BCD representation of 6) is added to the sum. The final result is 0001 0011, which represents the decimal number 13. A bracket under the first four digits of the final result is labeled 'Carry' with an upward arrow.

$$\begin{array}{r}
 \overset{1}{0}110 \\
 + 0111 \\
 \hline
 1101 \quad \text{(Illegal Form)} \\
 + 0110 \quad \text{(Adding 6)} \\
 \hline
 0001\,0011 \\
 \underbrace{}_{\uparrow \text{Carry}}
 \end{array}$$

wex

Example 2: Perform BCD Addition of 8765 and 3943

Solution

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

	¹¹¹ 1000	¹¹¹ 0111	¹ 0110	¹¹¹ 0101
	+ 0011	1001	0100	0011
	<u>1100</u>	<u>0001</u>	<u>1010</u>	<u>1000</u>
Correction →	0110	0110	0110	0000
	<u>0001</u>	<u>0010</u>	<u>0111</u>	<u>0000</u>
	1	2	7	0
				8

BCD representation of **8765** is given as **1000 0111 0110 0011** and for **3943** it is **0011 1001 0100 0011**.

Firstly, we will perform a normal **binary addition** of two numbers now we see **1100** and **1010** which are illegal states also the third group of 4-bits from LSB side i.e., **0000** has a carry **1** to the next group. So, for correction, we have to add **0110** to all three groups. Thus, we get the correct result as **0001 0010 0111 0000 1000** which is equivalent to **(12708)₁₀** in decimal number system and this is what we get on adding **(8765)₁₀ + (3943)₁₀ = (12708)₁₀**. Hence, our result is also verified.

LOGIC GATE

A logic gate is an idealized or physical device implementing a Boolean function, that take inputs and produces a single logic output.

Computers are made up of logic gates. Logic gates take information coming in and output different information depending on what type of gate they are.

TRUTH TABLE AND LOGIC GATES

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Truth Table

A truth table is a representation of all possible combinations of Input values and their result in a tabular format. If the result of a logical expression or logical statement is always true or 1, it is known as Tautology. If the result expression or logical statement is always false or 0, it is known as Fallacy.

A	B	C	Decimal Equivalent
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	0	1	7

The number of rows in a truth table is calculated as 2^n where n is the total number of variables. Therefore if there are three variables A, B and C, then the possible values will be $2^3 = 8$ combinations. It can be represented as follows:

Thus, for n variable, the number of possible combinations of inputs will be 2^n , which are binary representations of the Integer from 0 to $(2^n - 1)$.

The logical operators are:

i) **NOT or Negation:** The operator is unary operator because it operates on single variable. This operator is also called complement and the symbol for this is '~' or '-' or "'". Thus, if the letter x stands for a logical statement then the statement $\sim x$ or x' means complement of x . The table clearly shows that if x is True, x' is False, x' is True.

X	X'
0	1
1	0

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

ii) **AND (Conjunction) Operator:** This operator operates on two or more operands on two or more operands. AND operator is used to perform logical multiplication and the symbol for AND operator is dot (' . ').

X	X'	F
0	0	0
0	1	0
1	0	0
1	1	1

iii) **OR (Disjunction) Operator:** OR operator is used to perform logical Addition and the symbol for OR operator is '+' plus.

X	X'	F
0	0	0
0	1	0
1	0	0
1	1	1

iv) **NAND**



$$Q = A \text{ NAND } B$$

Truth Table

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

v)NOR

vi)EX-OR

vii)EX-NOR

Logic Gates

A gate is a circuit which takes one or more inputs and generate only one output. A gate is a digital circuit because it can take only two values, i.e. either high or low. Logic gates use the binary operators AND, OR and NOT There are three fundamental logic gates which are as follows:

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

YES



INPUT		OUTPUT
A		
0		0
1		1

NOT



INPUT		OUTPUT
A		
0		1
1		0

AND



INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR



INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR



INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NAND



INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

XNOR



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

AND Gate

AND Gate works on two or more inputs which result in a single output. When all the inputs are 1 or high, then the output is 1 otherwise the output is 0.

OR Gate

OR Gate works on two or more inputs which result in a single output. When any of the input signals is 1, the output signal is 1 and if all input signal is 0, the output signal will be 0.

NOT Gate

NOT gate works on a single input signal and generates a single output signal. The output state is a negation or complement of an input signal which is denoted by (') **NAND Gate, Logic NOT**
NAND Gate

This gate is the combination of AND or NOT gate. The NAND gate has two or more input but only one output. It produces output 1 when any of the input is 0.

NOR Gate

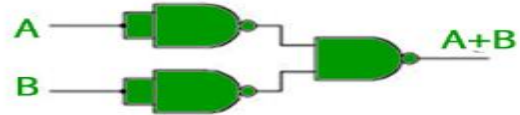
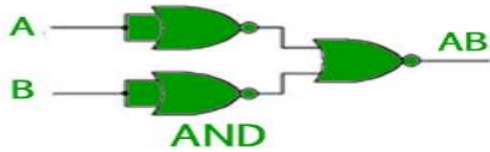
The NOR gate (negated OR) gives an output of 1 only if both inputs are 0, it gives 0 otherwise. For n-input gate if all inputs are 0 then it gives 1 otherwise 0.

XOR Gate– The XOR gate gives an output of 1 if either both inputs are different, it gives 0 if they are same. For n-input gate if the number of input 1 are odd then it gives 1 otherwise 0.

XNOR Gate The XNOR gate (negated XOR) gives an output of 1 if both inputs are same and 0 if they are different. For n-input gate if the number of input 1 are even then it gives 1 otherwise odd.

Universal Logic Gates – Out of the eight logic gates discussed above, NAND and NOR are also known as **universal gates** since they can be used to implement any digital circuit without using any other gate. This means that every gate can be created by NAND or NOR gates only. Implementation of three basic gates using NAND and NOR gates is shown below –

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.



SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

Hamming code is a set of error-correction codes that can be used to **detect and correct the errors** that can occur when the data is moved or stored from the sender to the receiver.

Redundant bits – Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of redundant bits can be calculated using the following formula:

$$2^r \geq m + r + 1$$

where, r = redundant bit, m = data bit

Suppose the number of data bits is 7, then the number of redundant bits can be calculated using: $2^4 \geq 7 + 4 + 1$ Thus, the number of redundant bits = 4

Parity bits. A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd. Parity bits are used for error detection. There are two types of parity bits:

1. **Even parity bit:** In the case of even parity, for a given set of bits, the number of 1's are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1's an even number. If the total number of 1's in a given set of bits is already even, the parity bit's value is 0.
2. **Odd Parity bit** – In the case of odd parity, for a given set of bits, the number of 1's are counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1's an odd number. If the total number of 1's in a given set of bits is already odd, the parity bit's value is 0.

Algorithm of Hamming code: Hamming Code is simply the use of extra parity bits to allow the identification of an error.

1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
2. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
3. All the other bit positions are marked as data bits.
4. Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.
 - a. Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
 - b. Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

- c. Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).
 - d. Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).
 - e. In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.
5. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.
 6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Position	R8	R4	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

R1 -> 1,3,5,7,9,11
 R2 -> 2,3,6,7,10,11
 R3 -> 4,5,6,7
 R4 -> 8,9,10,11

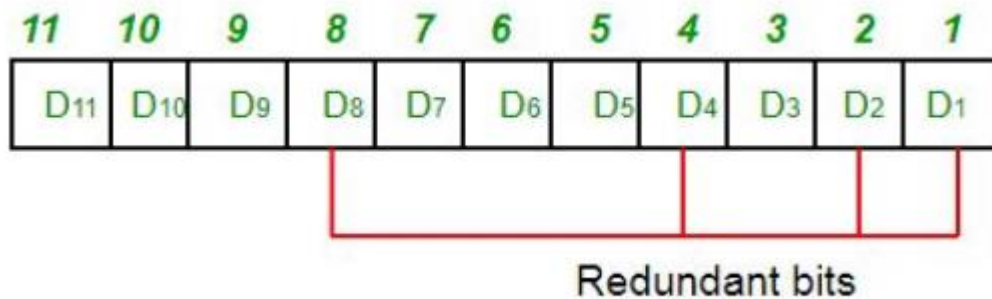
Determining the position of redundant bits – These redundancy bits are placed at positions that correspond to the power of 2.

As in the above example:

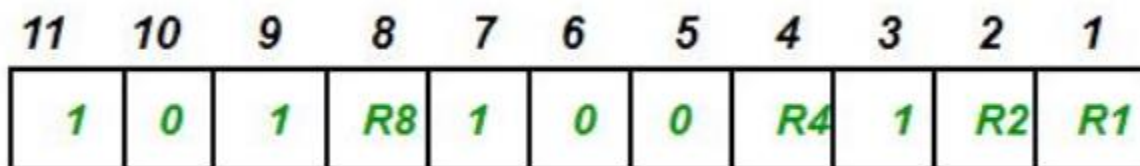
- The number of data bits = 7

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

- The number of redundant bits = 4
- The total number of bits = 11
- The redundant bits are placed at positions corresponding to power of 2 that is 1, 2, 4, and 8



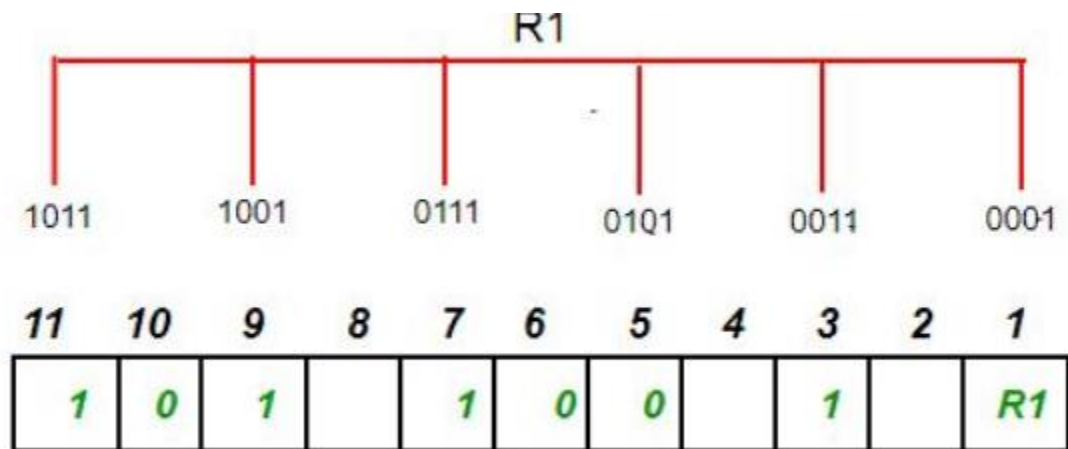
- Suppose the data to be transmitted is 1011001, the bits will be placed as follows:



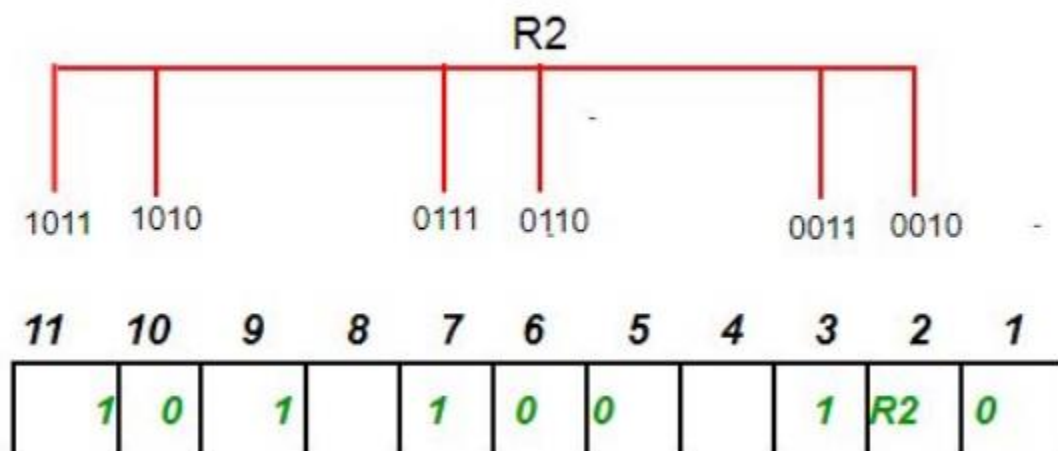
Determining the Parity bits:

- R₁ bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the least significant position. R₁: bits 1, 3, 5, 7, 9, 11

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.



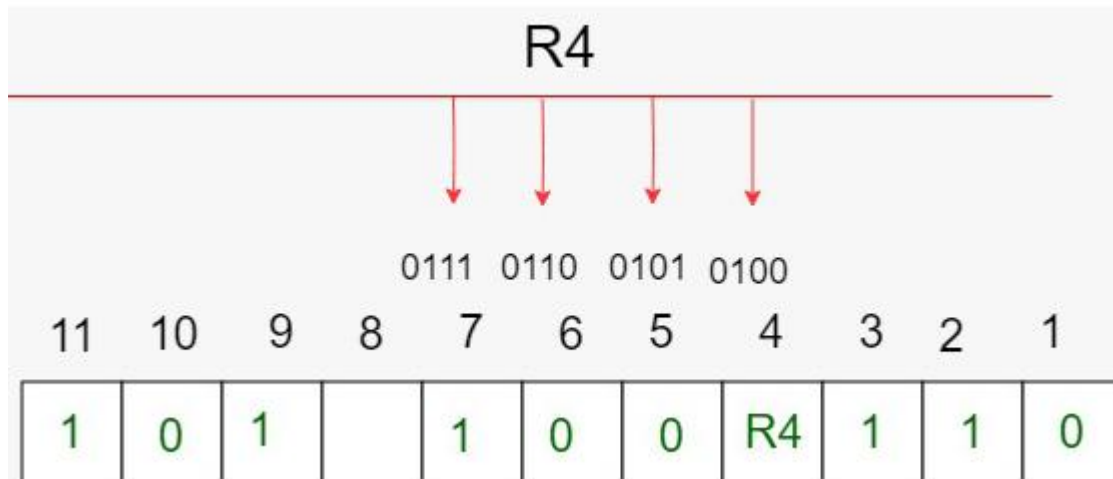
- To find the redundant bit R1, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R1 is an even number the value of R1 (parity bit's value) = 0
- R2 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the second position from the least significant bit. R2: bits 2,3,6,7,10,11



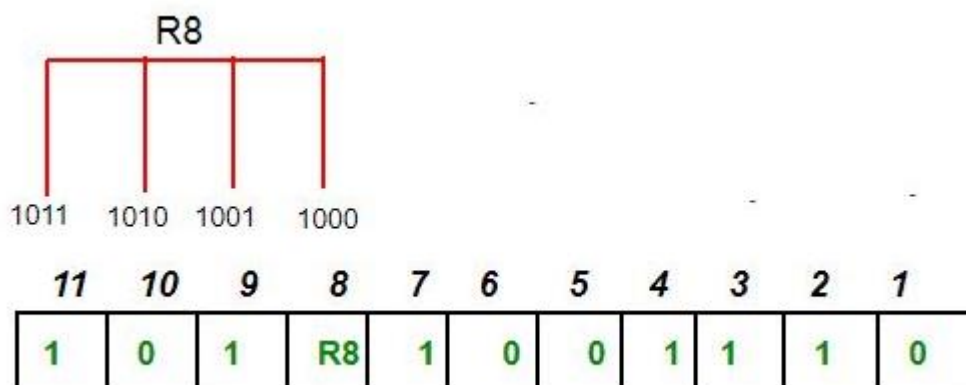
- To find the redundant bit R2, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R2 is odd the value of R2 (parity bit's value) = 1

SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

- R4 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the third position from the least significant bit. R4: bits 4, 5, 6, 7



- To find the redundant bit R4, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R4 is odd the value of R4 (parity bit's value) = 1
- R8 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit. R8: bit 8, 9, 10, 11



SYLLABUS Unit 1: Introduction to Number System and Logic Gates: Number Systems- Binary, Decimal, Octal, Hexadecimal; Codes- Grey, BCD, Excess-3, ASCII, Parity; Binary Arithmetic- Addition, Subtraction, Multiplication, Division using Sign Magnitude, 1's complement, 2's complement, BCD Arithmetic; Logic Gates-AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.

- To find the redundant bit R8, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R8 is an even number the value of R8 (parity bit's value)=0. Thus, the data transferred is:

11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	1	0	0	1	1	1	0

Error detection and correction: Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number:

For all the parity bits we will check the number of 1's in their respective bit positions.

For R1: bits 1, 3, 5, 7, 9, 11. We can see that the number of 1's in these bit positions are 4 and that's even so we get a 0 for this.

For R2: bits 2,3,6,7,10,11 . We can see that the number of 1's in these bit positions are 5 and that's odd so we get a 1 for this.

For R4: bits 4, 5, 6, 7 . We can see that the number of 1's in these bit positions are 3 and that's odd so we get a 1 for this.

For R8: bit 8,9,10,11 . We can see that the number of 1's in these bit positions are 2 and that's even so we get a 0 for this.

The bits give the binary number 0110 whose decimal representation is 6. Thus, bit 6 contains an error. To correct the error the 6th bit is changed from 1 to 0.