CHAPTER 1

Number System

LEARNING OBJECTIVES

After studying this unit, you should be able to

1. Represent the digital information in terms of various number systems.
2. Analyze and convert the information from one base to another base.
3. Implement the digital systems in terms of logic circuits.
4. Understand the concepts and techniques of code conversion.

1.1 DIGITAL ELECTRONICS

Digital electronics is a branch of electronics which deals with arithmetic of digits, design of digital and logic circuits. To control and process various systems, digital circuits are mostly used. The term digit is derived from the digits. So, the systems, whose operation is based on digits, are categorized as digital system.

1.1.1 INTRODUCTION

Digital electronics involves the passage of electronic pulses along logic circuits. The daily life example of digital electronics is switch. In digital system, the operation of switch is represented as logic HIGH and logic LOW, which is represented as '0' or '1'. Here, '0' depicts that bulb is OFF and '1' depicts that bulb is ON. From toy to satellite and pocket calculator to automobile industry, there are enormous applications of digital circuits. Digital world has made the life of human being easier, as digital circuits are less complex and more accurate than analog circuits.

Positive logic: A digital system where low-level logic is represented as digit (bit) 0 and high-level logic is represented as 1.

Negative logic: A digital system where low-level logic is represented as digit (bit) 1 and high-level logic is represented as level 0.

Positive logic	Negative logic
Low = 0	Low = 1
High = 1	High = 0

The digital system is based on devices having two states. For example, voltage 0 V is represented as level 0 and + 5 V is represented as level 1, in case of positive logic; whereas voltage 0 V is represented as logic 1 and + 5 V is represented as level 0 in case of negative logic. The positive logic and negative logic are shown as Fig. 1.1(a) and Fig. 1.1(b).

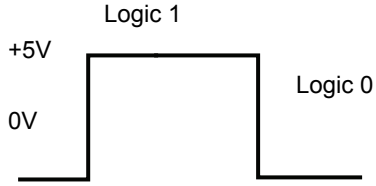


Fig. 1.1(a) Positive logic

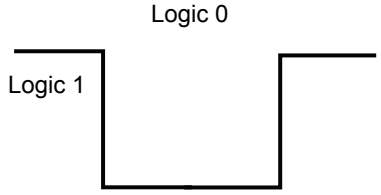


Fig. 1.1(b) Negative logic

1.1.2 ADVANTAGES OF DIGITAL SYSTEMS

- Digital systems are easier to design because only two bits are involved in designing 0 and 1.
- Accuracy level of digital system is higher than analog system.
- Noise effect is less on digital system.
- The data can be stored easily in latches or flip flops. The information can be stored for period of time.
- The operation of digital system is more reliable than analog system.
- The size of digital IC is small and compact. It is easier and cheaper to fabricate digital IC than analog IC.
- The power consumption for digital IC is less than analog IC.
- The same digital IC can be used for multiple times with high fidelity and reliability.

1.1.3 COMPARISON OF DIGITAL AND ANALOG SYSTEMS

The system which has capability to process continuous range of values and signals is known as analog system. In analog device, data is represented by binary system format whereas in digital device, the data is represented by binary system format. In daily life, we experience so many analog systems such as analog watches, voltmeter, speedometer, CRO readings etc. Analog systems have been used since so many decades and has various applications in electronic industry. The digital system deals with discrete range of values and signals. For example, a digital weighing machine indicates the weight 47.6 kg, as compare to analog weighing machine. Reading an analog device, involves human error and approximation.

Parameter	Analog System	Digital System
Nature of signal	Continuous signals	Discrete signals
Accuracy and Precision	Less accurate	More accurate
Noise Effect	Much effected	Less effected

Contd...

Parameter	Analog System	Digital System
Memory and storage	Memory is not available	It has memory
Fabrication of IC	Analog IC are difficult to fabricate	Digital IC are easy to fabricate
Size	Large in size	Small and compact
Reliable	Reliability is less	More reliable
Examples	Voltmeter, power supply, CRO	Counters, digital meter, computers

1.2 DATA REPRESENTATION AND CODING

The data is the set of values or information, which is required to process, control or design a system. The data representation is a technique to store, represent, process and transmit data. The computer operates on data and transmit it in form of electrical signals, which can be further stored in form of mechanical, optical or magnetic storage media. Computers do not understand human language. So, data and instructions are need to be converted in machine specific language before entering in computer. Due to this reason, it is necessary to understand the data representation types and techniques, which can be handled and processed by a computer directly.

1.2.1 DATA REPRESENTATION IN DIGITAL SYSTEM

Modern era is known as world of integration, where the main consideration of designers and manufacturers are minimum space, minimum cost and fast processing. In order to achieve these constraints, millions of electronic components are integrated on a single chip or circuits. In digital electronics, the logic is represented as high level and low level. These two levels are denoted by digits 0 and 1. For example, a vacant glass of water can be represented as 0 and filled glass of water can be represented as 1, in case of digital systems.

1.2.2 DATA REPRESENTATION IN CODING SCHEMES

When the letters, alphabets, numbers or words are represented by specific symbol or value then it is called that data is encoded. In digital system, the digital data is stored, represented and processed in form of binary bits. The number and value of bits per character are dependent on coding scheme. So, digital coding is representation of letters, words, symbols in digital format using binary digits.

The most common coding schemes are:

- Binary Coded Decimal (BCD)
- Extended Binary Coded Decimal Interchange Code (EBCDIC)

- American Standard Code for Information Interchange (ASCII).
- Gray code
- Hamming code

1.3 NUMBER REPRESENTATION

In real world, we deal with real numbers. The conversion and storing of real numbers play an important role in digital systems. In modern computation technology, two approaches are mainly used to store the real numbers. These are fixed point notation and floating-point notation.

1.3.1 FIXED POINT REPRESENTATION

The fixed-point representation is widely used in game application and digital signal processors. In fixed point number representation, number of digits (bits) are reserved for fractional part and number of digits (bits) are reserved for integral part. The decimal point is the main constraint of the representation. The portion which is to the right side of decimal point is known as fraction and portion which is to the left side of decimal point is known as integral. The same number of digits or bits are allocated for each portion, irrespective of the size of number.

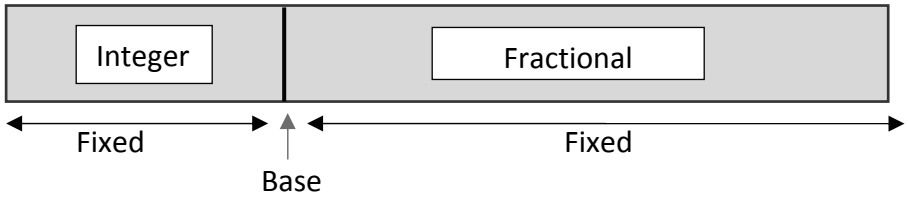


Fig. 1.2 Fixed point representation

The base or radix is repositioned when signed integers are stored in the memory. The base or radix is multiplied by a fixed scaling factor. The scaling factor in binary is always 2 raised to a fixed exponent. In that manner, the radix or base point repositioned, either right or left to its starting position. The conversion can move the base point in following three directions:

- **No change in base point position:** The case arises by a scaling factor whose exponent is 0, which indicates that stored integer value is exactly same as the integer value which is being represented.
- **Base point is shifted to the right:** This case represents the condition when scaling factor has exponent 1 or more. It indicates that represented number is larger than stored binary integer. Additional zeros are appended to right of LSB.

- **Base point is shifted to the left:** This case represents the condition when scaling factor has exponent negative. It indicates that represented number is smaller than stored binary integer. It further suggests that represented number has fractional part.

1.3.1.1 EXAMPLE OF FIXED-POINT NUMBERS

Assume an 8-bit signed number $(00011011)_2$, is stored in memory using 8 bits of storage.

In first case, if scale factor is 2^2. As the scale factor is more than 1, so the base point will be shifted two points to the right.

$$\text{Number} = (01101100)_2$$

In second case, if scale factor is 2^{-3}. As the scale factor is negative, so the base point will be shifted three points to the left.

$$\text{Number} = (00011.011)_2$$

1.3.1.2 ADVANTAGES AND DISADVANTAGES OF FIXED-POINT REPRESENTATION

The fixed-point notation is widely used in real time applications. It has following advantages, which make it preferable for users.

- Improved performance
- Optimized solution
- No need of additional hardware or software concepts or models
- Numbers are represented exactly and accurately.

The main disadvantage of the fixed-point representation is range of values which they can represent. A very limited range of values can be represented in fixed point notation of numbers.

1.3.1.3 RANGE OF VALUES STORED IN FIXED POINT REPRESENTATION

If we want to store b bits of number with scale factor n, then the minimum and maximum range of numbers that can be stored using fixed point representation can be calculated using following formula:

Minimum value: $-\frac{2^{b-1}}{2^n}$

Maximum value: $\frac{(2^{b-1})-1}{2^n}$

Due to limited range of values, floating point notation exists!

1.3.2 FLOATING POINT REPRESENTATION

An alternative to fixed point representation, we have floating point representation. The floating means that radix point can float. There is no fixed criterion of digits before or after the decimal point. When fractional numbers need to be stored in memory, modern technology uses floating point representation. The floating-point representation is based on scientific notation. Degrees of precision depends on the scale of number.

Fig. 1.3 Floating point representation

The signed bit, mantissa and exponent are the parameters of floating point representation.

$$\text{Floating point representation} = Sign\ bit\ \times Mantissa\ \times Base^{Exponent}$$

where binary has 2 base and decimal numbers have 10 base. The sign bit indicates whether the overall number is positive or negative.

1.3.2.1 IEEE 754 REPRESENTATIONS

Due to wide range of applications, the Institute of Electrical and Electronic Engineers, IEEE, has standardised the format of storage of floating point in memory, known as IEEE 754. This standard defines a number of representations, which consumes different bits of storage

Precision Level	Storage Bits
Half Precision	16
Single Precision	32
Double Precision	64
Quadruple Precision	128

In each of these cases, their basic structure is as follows:

$$\text{Floating point representation} = Sign\ bit\ \times Mantissa\ \times Base^{Exponent}$$

1.3.2.2 NOT A NUMBER (NAN)

The IEEE standard is having a concept of not a number (NAN), which is used to represent a number which is not real. When we divide something by zero, we usually get this result. This NaN is represented in memory by an exponent with all bit set and non-zero mantissa.

1.3.2.3 FLOATING POINT ADDITION

Assume you have two numbers 9.70×10^{-1} and 9.99×10^{1}. Add these two numbers.

Step-1	Rewrite the numbers, so that exponent of both numbers match.	$9.70 \times 10^{-1} = 0.097 \times 10^{1}$
Step-2	Add the mantissas of both numbers	0.097 + 9.99 = 10.087 So, the sum is 10.087×10^{1}
Step-3	Convert the result in normalised form	$10.087 \times 10^{1} = 1.0087 \times 10^{2}$
Step-4	Round the result if mantissa does not fit in the space provided.	1.0087×10^{2} can be rounded off as 1.009×10^{2}

1.3.2.4 ADVANTAGES AND DISADVANTAGES OF FLOATING-POINT REPRESENTATION

As computer memory is limited, numbers with limited precision are not stored in computer. So, floating point numbers are of great benefit. The floating-point notation has the advantage that it has wide range of numbers. But the downside of floating point notations is that it round off large numbers.

1.4 DIGITAL NUMBER SYSTEM

In digital electronics, the number system is used for representing the information. The number system has various bases. The radix or base is the total numbers of digits that a particular number system can accommodate. Suppose if the number system representing the digit from 0 – 9 then the base of the system is the 10.

A system based on digital logics and circuits understand the digits and their value based on their position in the designated number. The value of digit can be evaluated using

- Position of digit in number system
- The radix/base of the number system

The different number system that are used to implement and design various logic circuits and systems are:

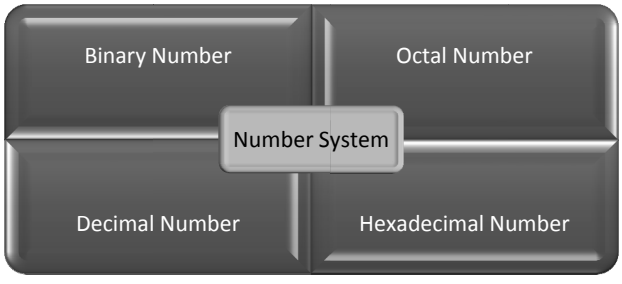


Fig. 1.4 Types of number system

1.4.1 BINARY NUMBER SYSTEM

Binary number system is also called base 2 system. In binary number system, any number can be represented using two digits 0 and 1. It has radix (base) 2. The binary number system is mostly used in computers, where programming language is based on two digits number system.

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

In binary number system, each position in a binary number represents a 0 power of the base (2).

Binary Number	1	1	0	.	1	1	1
Power of Base	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
Magnitude of each term	4	2	0		0.5	0.25	0.125

1.4.2 OCTAL NUMBER SYSTEM

In octal number system, any number can be represented using eight digits 0 to 7. It has radix (base) 8. Each position in an octal number represents a 0 power of the base (8). While working on computers, it is easier to write number in octal form rather than binary form.

Octal number	1	4	0	.	2	1	0
Power of Base	8^2	8^1	8^0		8^{-1}	8^{-2}	8^{-3}
Magnitude of each term	64	32	0		0.25	0.015	0

1.4.3 DECIMAL NUMBER SYSTEM

The number system that we use in our daily life applications is decimal number system. In decimal number system, any number can be represented using ten digits 0 to 9. It has radix (base) 10. Each position in a decimal number represents a X power of the base (10). In decimal number system, the successive positions to the left of the decimal point represents units, tens, hundreds, thousands and so on. The advantages in the decimal number system are that the mathematical terminology and concept use is identical to the base that is used for counting everyday numbers.

Decimal number	1	4	0	.	2	1	0
Power of Base	10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}
Magnitude of each term	100	40	0		0.2	0.01	0

1.4.4 HEXADECIMAL NUMBER SYSTEM

In hexadecimal number system, any number can be represented using digits and alphabets as 0 to 9 and A, B, C, D, E, F. It has radix (base) 16. Each position in a hexadecimal number represents a X power of the base (16). The hexadecimal system is commonly used by programmers to describe locations in memory because it can represent every byte (i.e., eight bits) as two consecutive hexadecimal digits instead of the eight digits.

Hexadecimal number	2	A	0	.	2	1	0
Power of Base	16^2	16^1	16^0		16^{-1}	16^{-2}	16^{-3}
Magnitude of each term	512	160	0		0.125	0.003	0

The value of digits and letters used in hexadecimal number system is:

Digit/Alphabet	Value	Digit/Alphabet	Value
0	0	8	8
1	1	9	9
2	2	A	10
3	3	B	11
4	4	C	12
5	5	D	13
6	6	E	14
7	7	F	15

The number system and digit associated with these number systems are as follow:

Number System	Base (Radix)	Digits & Alphabets
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0-9, A, B, C, D, E, F

1.5 NUMBER SYSTEM CONVERSION

In our daily life, we use decimal number system. But computers do not understand decimal number system. So, to process computations binary number system is needed. The representation of large numbers in binary is difficult to read. In context to understand

long sequence of binary digits, hexadecimal and octal number system are useful. The number system conversion is technique to convert one number in base r to another number system in equivalent base r.

1.5.1 CONVERSION FROM BINARY NUMBER SYSTEM TO ANY OTHER BASE

The binary number system is string of zeros and ones. The base of binary number system is two as it has only two values 0 and 1. The binary digit is known as bit. For example, the binary number 110101 has 6 bits and 0100 has 4 bits. The group of four bits is known as nibble and group of 8 bits is known as byte. A byte is basic unit in data processing. The range of n bit binary number is from 0 to 2^n-1. It means, 2-bit binary number has range from 0 to 3 and 4-bit binary number has range from 0 to 15.

1.5.1.1 BINARY TO DECIMAL NUMBER CONVERSION

Conversion from binary to decimal number system involves multiplication of each bit by its positional weight and summing up the values.

Rules for Conversion from Binary to Decimal

Rule 1: Note the positions of each bit and calculate its positional weight as 2^x; as binary system has radix 2.

Rule 2: Take the product of bit value and its positional weight.

Rule 3: Find the summation of all product values. This is required decimal number.

Example 1.1 Convert binary number $(1011.01)_2$ to decimal

Solution:

Bit value	1	0	1	1	.	0	1
Positional weight	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}
Product	8	0	2	1	.	0	0.25
Summation	8 + 0 + 2 + 1 + 0 + 0.25 = 11.25						

So, $(1011.01)_2 = (11.25)_{10}$

Example 1.2 Convert binary number $(0100.10)_2$ to decimal

Solution:

Bit value	0	1	0	0	.	1	0
Positional weight	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}
Product	0	4	0	0	.	0.5	0
Summation	0 + 4 + 0 + 0 + 0.5 + 0 = 4.5						

So, $(0100.10)_2 = (4.5)_{10}$

1.5.1.2 BINARY TO OCTAL NUMBER CONVERSION

The radix of octal number is 8 and binary number is 2. The radix of octal number can be mentioned as $2^3 = 8$, which indicates that grouping of 3 bits produces an octal number. Append zeros, in case bits are less than 3.

Rules for Conversion from Binary to Octal

Rule 1: Make a group of three binary digits.

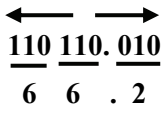
Rule 2: For integer, start grouping from right to left direction.

Rule 3: For fractional part, start grouping from left to right direction.

Rule 4: write the decimal equivalent of each group of bits.

Example 1.3 Convert binary number $(110110.010)_2$ to Octal

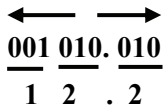
***Solution*:** The grouping of 3 bits can be done as:



So, $(110110.010)_2 = (66.2)_8$

Example 1.4 Convert binary number (1010.01)2 to Octal

***Solution*:** During grouping of 3 bits, zeros can be appended to complete the grouping



So, $(110110.010)_2 = (12.2)_8$

1.5.1.3 BINARY TO HEXADECIMAL NUMBER CONVERSION

The hexadecimal number system is mostly used in digital computation, because it deals with grouping of 4 bits. The radix of hexadecimal number is 16 and binary number is 2. The radix of hexadecimal number can be mentioned as $2^4 = 16$, which indicates that grouping of 4 bits produces a hexadecimal number. Append zeros, in case bits are less than 3.

Rules for Conversion from Binary to Hexadecimal

Rule1: Make a group of four binary digits.

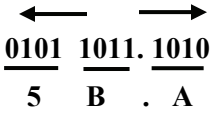
Rule 2: For integer, start grouping from right to left direction.

Rule 3: For fractional part, start grouping from left to right direction.

Rule 4: Write the decimal equivalent of each group of bits.

Example 1.5 Convert binary number $(01011011.1010)_2$ to Hexadecimal

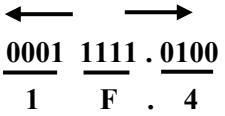
Solution: The grouping of 4 bits can be done as:



So, $(110110.010)_2 = (5B.A)_{16}$

Example 1.6 Convert binary number $(11111.01)_2$ to Hexadecimal

Solution: During grouping of 4 bits, zeros can be appended to complete the grouping



So, $(110110.010)_2 = (1F.4)_{16}$

1.5.2 CONVERSION FROM DECIMAL NUMBER SYSTEM TO ANY OTHER BASE

The decimal number system has digits from 0 to 9. It has radix 10. The daily life calculations and systems use decimal system. But the computer is unable to process the decimal inputs. So, the conversion from decimal number system to another number system is required.

1.5.2.1 DECIMAL TO BINARY NUMBER SYSTEM CONVERSION

The method used for converting a decimal integer number to binary number is known as double dabble. It involves successive division. The decimal number is divided by 2 and note down the remainders. Then use bottom to top approach for obtaining the binary number. The decimal fractional number is converted to binary number by successive multiplication.

Rules for conversion from Decimal to Binary

(A) For Decimal Integer:

Rule 1: Divide the quotient by 2, record the remainder.

Rule 2: Stop the division if quotient is less than base 2.

Rule 3: Adopt Bottom-Top approach and note the remainders. It is equivalent binary number.

(B) For Decimal Fraction:

Rule 1: Multiply the decimal fraction with 2. Separate the integer part from the product.

Rule 2: Repeat the multiplication of fractional part with 2, it will produce new product. Note the integer value and repeat multiplication of fractional part.

Rule 3: Adopt Top-Bottom approach for noted integer values. It is equivalent binary values. Discard the fractional part.

Example 1.7 Convert Decimal Number $(35.2)_{10}$ into Binary

Solution:

(A) Integer part: $(35)_{10}$

2	35	
2	17	1
2	8	1
2	4	0
2	2	0
	1	0

$(35)_{10} = (100011)_2$

(B) Fractional part: $(0.2)_{10}$

$0.2 \times 2 = 0.4$ integer: 0
$0.4 \times 2 = 0.8$ integer: 0
$0.8 \times 2 = 1.6$ integer: 1
$0.6 \times 2 = 1.2$ integer: 1

$(0.2)_{10} = (0.0011)_2$

So, $(35.2)_{10} = (100011.0011)_2$

Example 1.8 Convert Decimal Number $(105.75)_{10}$ into Binary

***Solution*:**

(A) Integer part: $(105)_{10}$

2	105	
2	52	1
2	26	0
2	13	1
2	6	1
2	3	0
2	1	1

$(105)_{10} = (1101001)_2$

(B) Fractional part: $(0.75)_{10}$

$0.75 \times 2 = 1.5$ integer: 1
$0.5 \times 2 = 1.0$ integer: 1
$0.0 \times 2 = 0.0$ integer: 0

$(0.75)_{10} = (0.110)_2$

So, $(105.75)_{10} = (1101001.110)_2$

1.5.2.2 DECIMAL TO OCTAL NUMBER SYSTEM CONVERSION

The decimal numbers can be converted into octal numbers by dividing by 8, in case of decimal integer and multiply by 8 for the case of decimal fraction. The rules for conversion are stated here below:

Rules for conversion from Decimal to Octal

(A) For Decimal Integer:

Rule 1: Divide the quotient by 8, record the remainder.

Rule 2: Stop the division if quotient is less than base 8.

Rule 3: Adopt Bottom-Top approach and note the remainders. It is an equivalent octal number.

(B) For Decimal Fraction:

Rule 1: Multiply the decimal fraction with 8. Separate the integer part from the product.

Rule 2: Repeat the multiplication of fractional part with 8, it will produce new product. Note the integer value and repeat multiplication of fractional part.

Rule 3: Adopt Top-Bottom approach for noted integer values. It is an equivalent octal values. Discard the fractional part.

Example 1.9 Convert Decimal Number (37.45)10 into Octal

(A) Integer part: $(37)_{10}$

8	37	
8	4	3 ↑

$(37)_{10} = (43)_8$

(B) Fractional part: $(0.45)_{10}$

$0.45 \times 8 = 3.60$ integer: 3
$0.60 \times 8 = 4.80$ integer: 4
$0.80 \times 8 = 6.40$ integer: 6
$0.40 \times 8 = 3.20$ integer: 3

↓

$(0.45)_{10} = (0.3463)_8$

So, $(37.45)_{10} = (43.3463)_8$

Example 1.10 Convert Decimal Number $(412.53)_{10}$ into Octal

(A) Integer part: $(412)_{10}$

8	412	
8	51	4
8	6	3 ↑

$(412)_{10} = (634)_8$

(B) Fractional part: $(0.53)_{10}$

$0.53 \times 8 = 4.24$ integer: 4
$0.24 \times 8 = 1.92$ integer: 1
$0.92 \times 8 = 7.36$ integer: 7
$0.36 \times 8 = 2.88$ integer: 2

$(0.53)_{10} = (0.4172)_8$

So, $(412.53)_{10} = (634.4172)_8$

1.5.2.3 DECIMAL TO HEXADECIMAL NUMBER SYSTEM CONVERSION

The decimal numbers can be converted into hexadecimal numbers by dividing by 16, in case of decimal integer and multiply by 16 for the case of decimal fraction. The rules for conversion are stated here below:

Rules for conversion from Decimal to Hexadecimal

(A) For Decimal Integer:

Rule 1: Divide the quotient by 16, record the remainder.

Rule 2: Stop the division if quotient is less than base 16.

Rule 3: Adopt Bottom-Top approach and note the remainders. It is an equivalent hexadecimal number.

(B) For Decimal Fraction:

Rule 1: Multiply the decimal fraction with 16. Separate the integer part from the product.

Rule 2: Repeat the multiplication of fractional part with 8, it will produce new product. Note the integer value and repeat multiplication of fractional part.

Rule 3: Adopt Top-Bottom approach for noted integer values. It is an equivalent hexadecimal values. Discard the fractional part.

Example 1.11 Convert Decimal Number $(107.35)_{10}$ into Hexadecimal

(A) Integer part: $(107)_{10}$

16	107
16	6

B

$(107)_{10} = (6B)_{16}$

(B) Fractional part: $(0.35)_{10}$

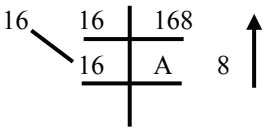
$0.35 \times 16 = 5.60$ integer: 5
$0.60 \times 16 = 9.60$ integer: 9
$0.60 \times 16 = 9.60$ integer: 9
$0.60 \times 16 = 9.60$ integer: 9

$(0.35)_{10} = (0.5999)_{16}$

So, $(107.35)_{10} = (43.3463)_8$

Example 1.12 Convert Decimal Number $(168.53)_{10}$ into Hexadecimal

(A) Integer part: $(168)_{10}$



$(168)_{10} = (A8)_{16}$

(B) Fractional part: $(0.53)_{10}$

$0.53 \times 16 = 8.48$ integer: 8

$0.48 \times 16 = 7.68$ integer: 7

$0.68 \times 16 = A.88$ integer: A

$0.88 \times 8 = E.08$ integer: E

$(0.53)_{10} = (0.87AE)_{16}$

So, $(168.53)_{10} = (A8.87AE)_8$

1.5.3 CONVERSION FROM OCTAL NUMBER SYSTEM TO ANY OTHER BASE

Octal number system has base 8. It is clubbing of 3 binary digits. The range of octal number is from 0 to 7. The conversion from octal to other number is important for optimization purpose.

1.5.3.1 OCTAL TO BINARY NUMBER SYSTEM CONVERSION

Octal to binary number system is reverse of binary to octal number system. The rules for conversion are as below:

Rules for Conversion from Octal to Binary

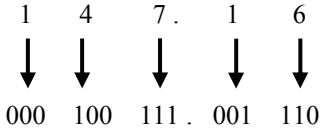
Rule1: Write the octal number: integer and fraction part.

Rule 2: Mention each octal digit in terms of 3 binary bits. It is an equivalent required binary digit.

Rule 3: For fractional part, start grouping from left to right direction.

Example 1.13 Convert Octal Number $(147.16)_8$ to Binary

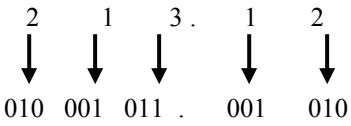
***Solution*:**



So, $(147.16)_8 = (000100111.001110)_2$

Example 1.14 Convert Octal Number $(213.12)_8$ to Binary

***Solution*:**



So, $(213.12)_8 = (010001011.001010)_2$

1.5.3.2 OCTAL TO DECIMAL NUMBER SYSTEM CONVERSION

As the decimal system is widely used number system in our day to day life. The conversion from octal to decimal number system is multiplication of octal bits by its positional weight and summing up the product. The rules for conversion are stated as below:

Rules for Conversion from Octal to Decimal

Rule 1: Note the positions of each bit and calculate its positional weight as 8^x; as octal system has radix 8.

Rule 2: Take the product of bit value and its positional weight.

Rule 3: Find the summation of all product values. This is required decimal number.

Example 1.15 Convert octal number $(6230.41)_8$ to decimal

***Solution*:**

Bit value	6	2	3	0	.	4	1
Positional weight	8^3	8^2	8^1	8^0	.	8^{-1}	8^{-2}
Product	3072	128	24	0	.	0.5	0.015
Summation	3072 + 128 + 24 + 0 + 0.5 + 0.015 = 3224.515						

So, $(6230.41)_8 = (3224.515)_{10}$

Example 1.16 Convert octal number $(1211.20)_8$ to decimal

***Solution*:**

Bit value	1	2	1	1	.	2	0
Positional weight	8^3	8^2	8^1	8^0	.	8^{-1}	8^{-2}
Product	512	128	8	1	.	0.25	0
Summation	512 + 128 + 8 + 1 + 0.25 + 0 = 649.25						

So, $(1211.20)_8 = (649.25)_{10}$

1.5.3.3 OCTAL TO HEXADECIMAL NUMBER SYSTEM CONVERSION

The octal number system is clubbing of 3 binary digits and hexadecimal number system is clubbing of 4 binary digits. Most of the computational techniques are based on hexadecimal system, because 4 binary digits are grouped in this number system. The conversion rules from octal to hexadecimal are as below:

Rules for Conversion from Octal to Hexadecimal

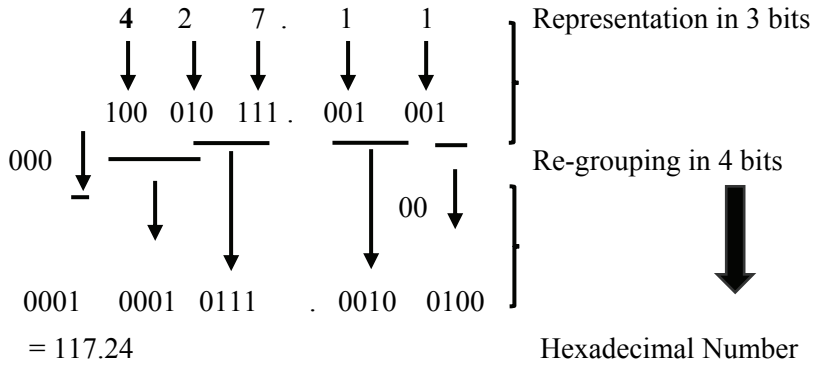
Rule1: Write the octal number: integer and fraction part.

Rule 2: Mention each octal digit in terms of 3 binary bits.

Rule 3: Regroup the bits in 4-bit. It is an equivalent required hexadecimal digit.

Example 1.17 Convert Octal Number $(427.11)_8$ to Hexadecimal

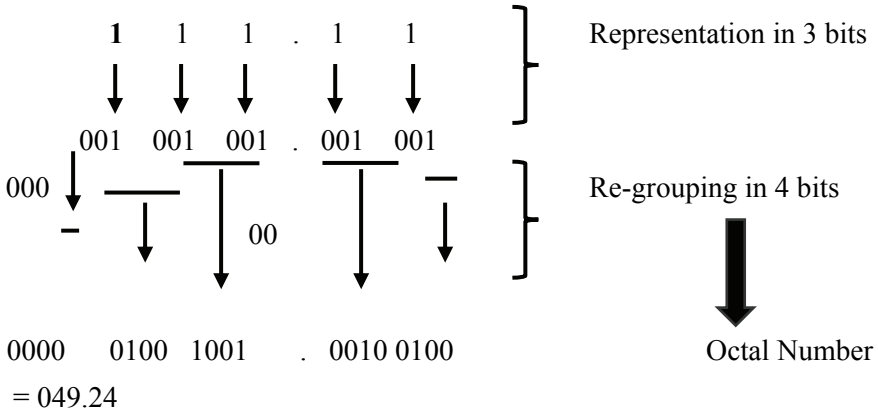
***Solution*:**



So, $(427.11)_8 = (117.24)_{16}$

Example 1.18 Convert Octal Number $(111.11)_8$ to Hexadecimal

***Solution*:**



So, $(111.11)_8 = (049.24)_{16}$

1.5.4 CONVERSION FROM HEXADECIMAL NUMBER SYSTEM TO ANY OTHER BASE

The hexadecimal number system has base 16. The range of hexadecimal numbers are from 0-9, A, B, C, D, E, F. due to wide range of applications, the conversion from hexadecimal number system to another number system is important.

1.5.4.1 HEXADECIMAL TO BINARY NUMBER SYSTEM CONVERSION

Hexadecimal to binary number system is reverse of binary to hexadecimal number system. The rules for conversion are as below:

Rules for Conversion from Hexadecimal to Binary

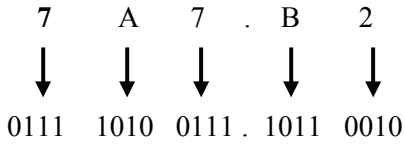
Rule 1: Write the hexadecimal number: integer and fraction part.

Rule 2: Mention each Hexadecimal digit in terms of 4 binary bits. It is an equivalent required binary digit.

Rule 3: For fractional part, start grouping from left to right direction.

Example 1.19 Convert Hexadecimal Number $(7A7.B2)_{16}$ to Binary

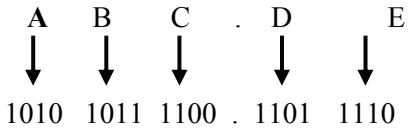
***Solution*:**



So, $(7A7.B2)_{16} = (011110100111.10110010)_2$

Example 1.20 Convert Hexadecimal Number $(ABC.DE)_{16}$ to Binary

***Solution*:**



So, $(ABC.DE)_{16} = (101010111100.11011110)_2$

1.5.4.2 HEXADECIMAL TO DECIMAL NUMBER SYSTEM CONVERSION

The decimal number system is the major requirement for real time applications. The conversion from hexadecimal to decimal number system is multiplication of bit value by its positional weight and summing up the product.

Rules for Conversion from Hexadecimal to Decimal

Rule 1: Note the positions of each bit and calculate its positional weight as 16^x; as octal system has radix 16.

Rule 2: Take the product of bit value and its positional weight.

Rule 3: Find the summation of all product values. This is required decimal number.

Example 1.21 Convert hexadecimal number $(6AAA.C1)_{16}$ to decimal.

***Solution*:**

Bit value	6	A	A	A	.	C	1
Positional weight	16^3	16^2	16^1	16^0	.	16^{-1}	16^{-2}
Product	40960	2560	160	10	.	0.75	0.003
Summation	40960 + 2560 + 160 + 10 + 0.06 + 0.003 = 43690.753						

So, $(6AAA.C1)_{16} = (43690.753)_{10}$

Example 1.22 Convert hexadecimal number $(2381.FF)_{16}$ to decimal.

***Solution*:**

Bit value	2	3	8	1	.	F	F
Positional weight	16^3	16^2	16^1	16^0	.	16^{-1}	16^{-2}
Product	8192	768	128	1	.	0.93	0.05
Summation	8192 + 768 + 128 + 1 + 0.93 + 0.05 = 9089.98						

So, $(2381.FF)_{16} = (9089.98)_{10}$

1.5.4.3 HEXADECIMAL TO OCTAL NUMBER SYSTEM CONVERSION

The hexadecimal number system is having radix 16 and it is group of 4 binary digits, whereas octal number system is clubbing of 3 binary digits. The conversion rules from hexadecimal to octal are as below:

Rules for Conversion from Hexadecimal to Octal

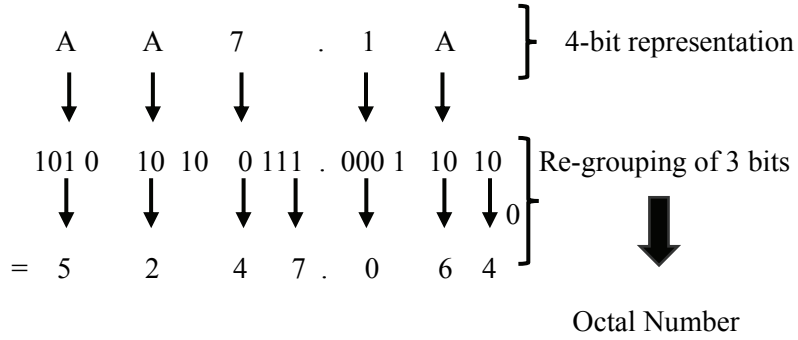
Rule 1: Write the hexadecimal number: integer and fraction part.

Rule 2: Mention each hexadecimal digit in terms of 4 binary bits.

Rule 3: Regroup the bits in 3-bit. It is an equivalent required octal digit.

Example 1.23 Convert Hexadecimal Number $(AA7.1A)_8$ to Octal

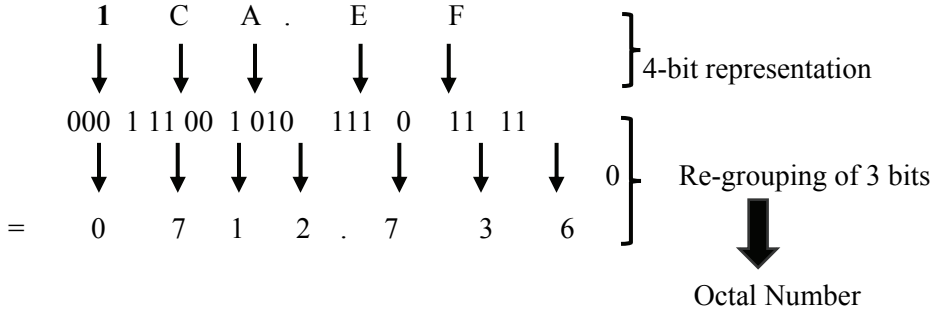
***Solution*:**



So, $(AA7.1A)_{16} = (5247.064)_8$

Example 1.24 Convert Hexadecimal Number $(1CA.EF)_8$ to Octal

***Solution*:**



So, $(1CA.EF)_{16} = (0712.736)_8$

1.6 BINARY ARITHMETIC

Binary number system is the foundation of all computer related operations. Arithmetic operations are possible on binary number system. The procedure of binary arithmetic is almost similar to mathematical calculation. Rules need to be followed for binary addition, binary subtraction, binary multiplication and binary division.

1.6.1 BINARY ADDITION

When we add two numbers the resultant is sum and if that sum is having an extra bit, that bit is known as carry bit. The numbers to be added are known as augend and addend. The binary addition deals with string of 0 and 1.

The rules for binary addition are as follow:

Rules for Binary Addition

0 + 0 = 0	Sum = 0; Carry = 0
0 + 1 = 1	Sum = 1; Carry = 0
1 + 0 = 1	Sum = 1; Carry = 0
1 + 1 = 10	Sum = 0; Carry = 1
1 + 1 + 1 = 11	Sum = 1; Carry = 1

Example 1.25 Add the following Binary Numbers

(a) 10 + 10 (b) 11 + 111 (c) 111 + 1111 (d) 101+101

***Solution*:**

(a) Mathematically, 2 + 2 = 4

Binary:

$$\begin{array}{r} 10 \\ +\ 10 \\ \hline 100 \end{array} \quad = \quad (4)_{10}$$

Answer: Sum = $(100)_2$; Carry = 1

(b) Mathematically, 3 + 7 = 10

Binary: The first number has 2 bits and second number has 3 bits. So, to equate number of bits, append zero to first number.

$$\begin{array}{r} 011 \\ +\ 111 \\ \hline 1010 \end{array} \quad = \quad (10)_{10}$$

Answer: Sum = $(1010)_2$; Carry = 1

(c) Mathematically, 7 + 15 = 22

Binary: The first number has 3 bits and second number has 4 bits. So, to equate number of bits, append zero to first number

$$\begin{array}{r} 0111 \\ +\ 1111 \\ \hline 10110 \end{array} \quad = \quad (22)_{10}$$

Answer: Sum = $(10110)_2$; Carry = 1

(d) Mathematically, 5 + 5 = 10

Binary:

$$\begin{array}{r} 101 \\ +\ 101 \\ \hline 1010 \end{array} \quad = \quad (10)_2$$

Answer: Sum = $(1010)_2$; Carry = 1

1.6.2 BINARY SUBTRACTION

In the subtraction y = a – b;

a is minuend, b is subtrahend and y is difference.

If the value of a is less than b, then a digit is taken from more significant position, is known as borrow bit. The binary subtraction is performed on binary bits.

Rules for Binary Subtraction	
0 – 0 = 0	Difference = 0; Borrow = 0
1 – 0 = 1	Difference = 1; Borrow = 0
1 – 1 = 0	Difference = 0; Borrow = 0
0 – 1 = 1	Difference = 1; Borrow = 1

Example 1.26 Subtract the following binary bits

(a) 1011-1001 (b) 10000-101

***Solution*:**

(a) Mathematically, 11 – 9 = 2

$$\begin{array}{r} 1011 \\ -1001 \\ \hline 0010 \end{array} \quad = \quad (2)_{10}$$

Answer: Difference = $(0010)_2$; Borrow = 0

(b) Mathematically, 16 – 5 = 11

$$\begin{array}{r} 1111 \quad \leftarrow \text{ Borrow from high order bit} \\ 10000 \\ -\ 101 \\ \hline 1011 \end{array} \quad = \quad (11)_{10}$$

Answer: Difference = $(1011)_2$; Borrow = 0

1.6.3 BINARY MULTIPLICATION

Binary multiplication is performed in same manner as decimal multiplication. Partial product is created and successive partial product is shifted to left. Then add all the partial products.

Rules for Binary Multiplication

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

Example 1.27 Perform the binary multiplication on following numbers

(a) 111×111 (b) 1001×1111

***Solution*:**

(a)

$$\begin{array}{r} 111 \\ \times\ 111 \\ \hline 1111 \\ 1111\ \times \\ 1111\ \times\times \\ \hline 110001 \end{array}$$

Answer: $(110001)_2$

(b)

$$\begin{array}{r} 1001 \\ \times\ 1111 \\ \hline 1001 \\ 1001\ \times \\ 1001\ \times\times \\ 1001\ \times\times\times \\ \hline 10000111 \end{array}$$

Answer: $(10000111)^2$

1.6.4 BINARY DIVISION

Decimal division and binary division, both are having same procedure and pattern. The rules for binary division are as below:

Rules for Binary Division

0 ÷ 1 = 0

1 ÷ 0 = 0

Division by 0 is not allowed

Example 1.28 Perform the following Binary Division Operation

(a) 110 ÷ 11 (b) 1111 ÷ 111

***Solution*:**

(a)

```
        10          Quotient
  11 ) 110
       11
       ---
       000          Remainder
```

Answer: $(10)_2$

(b)

```
        1010        Quotient
  101 ) 110110
        101
        ----
          111
          101
          ---
          100       Remainder
```

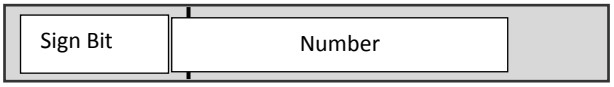
Answer: $(1010)_2$

1.7 REPRESENTATION OF SIGNED NUMBER

Unsigned numbers do not require any arithmetic sign to represent a number. They are positive numbers. A k bit unsigned number represents all the numbers from 0 to 2^k-1.

Range of 8-bit unsigned number		Range of 16-bit unsigned number	
Number system	Range	Number system	Range
Decimal	0-255	Decimal	0-65535
Hexadecimal	00-FF	Hexadecimal	0000-FFFF

In binary number systems, the negative numbers are encoded using signed numbers. The signed numbers require an arithmetic sign. The MSB, most significant bit of a binary number is used to represent the sign bit, which further indicates that whether number is positive or negative.



If sign bit = 0, number is positive; sign bit = 1, number is negative

For example,

(+2) = 0010 → (–2) = 1010

(+7) = 0111 → (–2) = 1111

(+1) = 0001 → (–2) = 1001

The signed numbers are represented by following representation

- Sign magnitude representation
- 1's complement representation
- 2's complement representation

For signed bit representation, the most significant bit MSB has been assigned as signed bit.

1.7.1 SIGN MAGNITUDE REPRESENTATION

The sign magnitude representation is also known as true magnitude representation. In this representation, magnitude of the number is specified as true binary value. The value of sign bit indicates whether the number is positive or negative.

Example 1.29 Represent + 12 and – 12 into 8-bit sign magnitude form.

Solution:

Sign bit

↑

+ 12 can be represented as 00001100

–12 can be represented as 10001100

↓

Sign bit

Example 1.30 Represent the following numbers in 8-bit sign magnitude form

(a) $(31)_{10}$ (b) $(-25)_{10}$ (c) $(44)_{10}$ (d) $(-18)_{10}$

***Solution*:** As it is required to represent the numbers in 8-bit magnitude form, one bit is reserved for sign bit and other 7-bit represent the magnitude of number. If the magnitude is less than 7-bit, then append zeros as required.

Number	Sign bit	Magnitude Bit						
31	0	0	0	1	1	1	1	1
-25	1	0	0	1	1	0	0	1
44	0	0	1	0	1	1	0	0
-18	1	0	0	1	0	0	1	0

1.7.2 1'S COMPLEMENT REPRESENTATION

The positive numbers are represented in similar manner for sign magnitude and 1's complement form. For negative numbers, complements have been taken for each bit, including the sign bit. The 1's complement can be calculated by complementing each bit (i.e., 0 for 1 and 1 for 0). The 1's complement of A is $\overline{A}$.

Example 1.31 Represent the following number in 8-bit 1's complement form

(a) $(44)_{10}$ (b) $(57)_{10}$ (c) $(37)_{10}$ (d) $(72)_{10}$

***Solution*:**

(a) The 8-bit binary representation of $(44)_{10} = (00101100)_2$

1's complement of $(44)_{10} = (11010011)_2$

Answer: $(11010011)_2$

(b) The 8-bit binary representation of $(57)_{10} = (00111001)_2$

1's complement of $(57)_{10} = (11000110)_2$

Answer: $(11000110)_2$

(c) The 8-bit binary representation of $(37)_{10} = (00100101)_2$

1's complement of $(37)_{10} = (11011010)_2$

Answer: $(11011010)_2$

(d) The 8-bit binary representation of $(72)_{10} = (01001000)_2$

1's complement of $(72)_{10} = (10110111)_2$

Answer: $(10110111)_2$

1.7.3 2'S COMPLEMENT REPRESENTATION

The positive numbers are represented in similar manner for sign magnitude and 2's complement form. For negative numbers, complements have been taken for each bit, including the sign bit.

2's complement = 1's complement + 1

Example 1.32 Represent the following number in 8-bit 2's complement form

(a) $(-44)_{10}$ (b) $(-57)_{10}$ (c) $(-37)_{10}$ (d) $(-72)_{10}$

***Solution*:**

(a) The 8-bit binary representation of $(44)_{10} = (00101100)_2$

1's complement of $(44)_{10} = (11010011)_2$

2's complement of $(44)_{10} = 11010011$

$+ \quad 1$

11010100

Answer: $(11010100)_2$

(b) The 8-bit binary representation of $(57)_{10} = (00111001)_2$

1's complement of $(57)_{10} = (11000110)_2$

2's complement of $(57)_{10} = 11000110$

$+ \quad 1$

11000111

Answer: $(11000111)_2$

(c) The 8-bit binary representation of $(37)_{10} = (00100101)_2$

1's complement of $(37)_{10} = (11011010)_2$

2's complement of $(37)_{10} = 11011010$

$+ \quad 1$

11011011

Answer: $(11011011)_2$

(d) The 8-bit binary representation of $(72)_{10} = (01001000)_2$

1's complement of $(72)_{10} = (10110111)_2$

2's complement of $(72)_{10} = 10110111$

$+ \quad 1$

10111000

Answer: $(10111000)_2$

1.8 COMPLEMENTS r's AND (r-1)'s

For logical manipulation and subtraction simplification, the complements are used in digital electronics. Generally, there are two types of complements: Radix complement r's complement and diminished radix complement (r-1)'s complement.

For binary number system (r = 2); diminished radix complement is 1's complement and radix complement is 2's complement.

For decimal number system (r = 10); diminished radix complement is 9's complement and radix complement is 10's complement.

1.8.1 DIMINISHED RADIX COMPLEMENT (r-1)'S COMPLEMENT

The diminished radix complements are radix-1 complement. If a number K is having n digits and radix r; the (r-1)'s complement of K can be calculated as:

$$(r-1)\text{'s complement of } K = (r^n-1)-K$$

where n = Number of digits; r = Radix or base

1's complement can be calculated by changing all 1's to 0's and all 0's to 1's

---

Example 1.33 Find 9's complement for given decimal number

(a) 532 (b) 4367 (c) 71654

***Solution*:**

(a) The number 532 is having 3 digits, n = 3

9's complement of 532 = $(10^3 - 1) - 532$

$= (1000-1)-532$

$= 999 - 532$

$= 467$

Answer: $(467)_{10}$

(b) The number 4367 is having 4 digits, n = 4

9's complement of 4367 = $(10^4) - 4367$

$= (10000 - 1) - 4367$

$= 9999-4367$

$= 5632$

Answer: $(5632)_{10}$

(c) The number 71654 is having 5 digits, n = 5

9's complement of 71654 = $(10^5 - 1) - 71654$

$= (100000 - 1) - 71654$

$= 99999 - 71654$

$= 28345$

Answer: $(28345)_{10}$

1.8.2 RADIX COMPLEMENT r'S COMPLEMENT

If a number K is having n digits and radix r; the (r)'s complement of K can be calculated as:

(r)'s complement of K = $(r^n - 1)$'s complement +1

Or, (r – 1)'s complement of K = $(r^n) - K$

where n = Number of digits; r = Radix or base

The 2's complement can be calculated as above discussed method as well as other method; where 2's complement = 1's complement +1

Example 1.34 Find the 10's complement of following decimal numbers

(a) 532 (b) 4367 (c) 71654

***Solution*:**

(a) The number 532 is having 3 digits, n=3

10's complement of 532 = $(10^3) - 532$

= (1000)-532

= 468

Answer: $(468)_{10}$

(b) The number 4367 is having 4 digits, n = 4

10's complement of 4367 = $(10^4) - 4367$

= (10000) – 4367

= 5633

Answer: $(5633)_{10}$

(c) The number 71654 is having 5 digits, n = 5

10's complement of 71654 = $(10^5) - 71654$

= (100000) – 71654

= 28346

Answer: $(28346)_{10}$

Example 1.35 Find the 2's complement of following binary numbers

(a) 111 (b) 10101 (c) 1111001

***Solution*:**

Method-1:

(a) Binary number 111; n = 3

2's complement of 111 = $(2^3) - 111$

$= (8)_{10} - 111$

$= 1000 - 111$

$= 001$

Answer: $(001)_2$

(b) Binary number 10101; n = 5

2's complement of 111 = $(2^5) - 10101$

$= (32)_{10} - 10101$

$= 100000 - 10101$

$= 01011$

Answer: $(01011)_2$

(c) Binary number 1111001; n = 7

2's complement of 1111001 = $(2^7) - 1111001$

$= (128)_{10} - 1111001$

$= 10000000 - 1111001$

$= 0000111$

Answer: $(0000111)_2$

Method-2:

2's complement = 1's complement + 1

(a) Binary number = 111

1's complement of 111 = 000

2's complement of 111 = 000

$+ \ 1$

001

Answer: $(001)_2$

(b) Binary number = 10101

1's complement of 10101 = 01010

2's complement of 10101 = 01010

$+ \ 1$

01011

Answer: $(01011)_2$

(c) Binary number = 1111001

1's complement of 1111001 = 0000110

2's complement of 1111001 = 0000110

$$\begin{array}{r} + \quad 1 \\ \hline 0000111 \end{array}$$

Answer: $(0000111)_2$

1.8.3 SUBTRACTION USING COMPLEMENTS

When we subtract two binary numbers X-Y; where X is minuend and Y is subtrahend, the role of complement becomes very important. There are few rules for subtraction of binary number using r's complement and (r-1)'s complement.

1.8.3.1 SUBTRACTION USING r's COMPLEMENT

Subtraction using r's complement: Let's take two binary numbers X and Y.

Rule 1: Check the number of bits in both binary numbers X and Y. Equate the digits by appending zeros.

Rule 2: Find the r's complement of Y. Add the result to X.

Rule 3: If carry is there, discard the carry. Remaining digits are the required resultant.

Example 1.36 Subtract 11011 – 1001 using 2's complement

***Solution*:**

Step 1: Equate the number of digits of both binary numbers. Equated digits:

11011 – 01001

Step 2: Take the 2's complement of 01001

1's complement of 01001 = 10110

2's complement of 01001 = 10111

Step 3: Add 2's complement of subtrahend to minuend

2's complement of subtrahend =10111

$$\begin{array}{rr} \text{Minuend} = & + 11011 \\ \hline & 110010 \end{array}$$

Step 4: Check whether carry is present or not

The addition produces a carry

Discard the carry. Remaining is the result ***Answer:*** $(10010)_2$

Example 1.37 Subtract 1011 – 10011 using 2's complement

***Solution*:**

Step 1: Equate the number of digits of both binary numbers. Equated digits:

01011 – 10011

Step 2: Take the 2's complement of 10011

1's complement of 10011 = 01100

2's complement of 10011 = 01101

Step 3: Add 2's complement of subtrahend to minuend

2's complement of subtrahend = 01101

$$\begin{array}{rl} \text{Minuend} & = +\ 01011 \\ & \underline{\quad 11000 \quad} \end{array}$$

Step 4: Check whether carry is present or not

The addition does not produce a carry

Take again the 2's complement

1's complement of 11000 = 00111

2's complement of 11000 = 01000

Answer: $-(01000)_2$

Example 1.38 Subtract 62516 – 4234 using 10's complement

***Solution*:**

Step 1: Equate the number of digits of both binary numbers. Equated digits:

62516 – 04234

Step 2: Take the 10's complement of 04234

10's complement of 04234 = $10^5 - 04234$

$= 100000 - 04234$

$= 95766$

Step 3: Add 10's complement of subtrahend to minuend

10's complement of subtrahend = 95766

$$\begin{array}{rl} \text{Minuend} & = +\ 62516 \\ & \underline{\quad 158282 \quad} \end{array}$$

Step 4: Check whether carry is present or not

The addition produces a carry

Discard the carry. Remaining is the result

Answer: $(58282)_{10}$

Example 1.39 Subtract 4234 - 62516 using 10's complement

***Solution*:**

Step 1: Equate the number of digits of both binary numbers. Equated digits:

04234 – 62516

Step 2: Take the 10's complement of 62516

10's complement of 62516 = 10^5 – 62516

= 100000 – 62516

= 37484

Step 3: Add 10's complement of subtrahend to minuend

10's complement of subtrahend = 37484

Minuend = + 04234

41718

Step 4: Check whether carry is present or not

The addition does not produce a carry

Step 5: Take again the 10's complement and put a negative sign

10's complement of 41718 = 10^5 – 41718

= 100000 – 41718

= 58282

Answer: $-(58282)_{10}$

1.8.3.2 SUBTRACTION USING (r-1)'S COMPLEMENT

Subtraction using (r-1)'s complement: Let's take two binary numbers X and Y.

Rule 1: Check the number of bits in both binary numbers X and Y. Equate the digits by appending zeros.

Rule 2: Find the (r-1)'s complement of Y. Add the result to X.

Rule 3: If carry is there, add the carry to least significant bit LSB of the sum. That sum is the required resultant.

Rule 4: If carry is not present, it indicates it is a negative number. Take again the r's complement of the number. Put negative sign ahead, this is the required resultant.

Example 1.40 Subtract 11011 – 1001 using 1's complement

***Solution*:**

Step 1: Equate the number of digits of both binary numbers. Equated digits:

11011 – 01001

Step 2: Take the 1's complement of 01001

1's complement of 01001 = 10110

Step 3: Add 1's complement of subtrahend to minuend

1's complement of subtrahend = 10110

$$\begin{array}{r} \text{Minuend} = + \ 11011 \\ \hline 110001 \end{array}$$

Step 4: Check whether carry is present or not

The addition produces a carry

Add the carry to LSB of the sum and that is the result

$$\begin{array}{r} = \ 10001 \\ + \quad 1 \\ \hline 10010 \end{array}$$

Answer: $(10010)_2$

Example 1.41 Subtract 1011 – 10011 using 2's complement

***Solution*:**

Step 1: Equate the number of digits of both binary numbers. Equated digits:

01011 – 10011

Step 2: Take the 1's complement of 10011

1's complement of 10011 = 01100

Step 3: Add 1's complement of subtrahend to minuend

1's complement of subtrahend = 01100

$$\begin{array}{r} \text{Minuend} = + \ 01011 \\ \hline 10111 \end{array}$$

Step 4: Check whether carry is present or not

The addition does not produce a carry

Take again the 1's complement

1's complement of 10111 = 01000 ***Answer:*** $-(01000)_2$

Example 1.42 Subtract 62516 – 4234 using 9's complement

***Solution*:**

Step 1: Equate the number of digits of both binary numbers. Equated digits:

62516 – 04234

Step 2: Take the 9's complement of 04234

9's complement of 04234 = $(10^5 - 1) - 04234$

= 99999 – 04234

= 95765

Step 3: Add 9's complement of subtrahend to minuend

9's complement of subtrahend = 95765

Minuend = + 62516

158281

Step 4: Check whether carry is present or not

The addition produces a carry

Add the carry to LSB of the number

= 58281

\+ 1

58282

Answer: $(58282)_{10}$

Example 1.43 Subtract 4234 – 62516 using 9's complement

Solution:

Step 1: Equate the number of digits of both binary numbers. Equated digits:

04234 - 62516

Step 2: Take the 9's complement of 62516

9's complement of 62516 = $(10^5 - 1) - 62516$

= 99999 – 62516

= 37483

Step 3: Add 9's complement of subtrahend to minuend

9's complement of subtrahend = 37483

Minuend = + 04234

41717

Step 4: Check whether carry is present or not

The addition does not produce a carry

Step 5: Take again the 9's complement and put a negative sign

9's complement of 41717 = $(10^5 - 1) - 41717$

= 99999 – 41717

= 58282

Answer: $-(58282)_{10}$

1.8.4 DIFFERENCE BETWEEN 1'S COMPLEMENT AND 2'S COMPLEMENT

- 1′ s complement has two representations of 0 (zero) – 00000000, which is positive zero (+0) and 11111111, which is negative zero (-0); whereas in 2′ s complement, there is only one representation for zero – 00000000 (+0) because if we add 1 to 11111111 (-1), we get 00000000 (+0) which is the same as positive zero. This is the reason why 2′ s complement is generally used.
- While adding numbers using 1′ s complement, we first do binary addition, then add in an end-around carry value. But, 2′ s complement has only one value for zero, and doesn't require carry values.
- 1's complement is much easier than 2's complement, because it involves lesser number of arithmetic operations.
- Mostly, 2's complement is used for arithmetic operation and 1's complement is used for logical manipulation.

1.9 HEXADECIMAL ARITHMETIC

Hexadecimal system is mostly used number system in memory systems and calculations. The grouping of 4 bits are represented as hexadecimal digit. The binary numbers having 8, 16, 32 bits can be represented by 2, 4 and 8 hexadecimal digits respectively.

1.9.1 HEXADECIMAL ADDITION

The hexadecimal digits are from 0 to 9, A to F. The addition follows the similar pattern and format as decimal number addition. Few rules are to be followed carefully, If the addition is more than 16, then again divide by 16 and convert it in hexadecimal number:

$$F + 1 = 10$$
$$F + F = 1E$$
$$F + F + 1 = 1F$$

The addition of two hexadecimal numbers will never produce a carry more than 1.

$$F + 1 = (15)_{10} + (1)_{10}$$
$$= (16)_{10} = (10000)_2$$
$$= (0001\ 0000)_2$$
$$= (10)_{16}$$

$$F + F = (15)_{10} + (15)_{10}$$
$$= (30)_{10} = (11110)_2$$
$$= (0001\ 1110)_2$$
$$= (1E)_{16}$$

$$F + F + 1 = (15)_{10} + (15)_{10} + 1$$
$$= (31)_{10} = (11111)_2$$
$$= (0001\ 1111)_2$$
$$= (1F)_{16}$$

Table 1.1 Hexadecimal Addition Table

Addition	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

By using rows and columns of above drawn table, the hexadecimal addition can be performed efficiently and accurately.

Example 1.44 Perform the hexadecimal addition 1A2 + ABC

***Solution*:** Hexadecimal addition

$$\begin{array}{r} 1\ A\ 2 \\ +\ A\ B\ C \\ \hline C\ 5\ E \end{array}$$

This can be validated using decimal addition

	1		
Decimal representation	1	10	2
	10	11	12
	C	5	E

Answer: $(C5E)_{16}$

Example 1.45 Perform the hexadecimal addition 2B8 + A9F

***Solution*:** Hexadecimal addition

$$\begin{array}{r} 2\ B\ 8 \\ +\ A\ 9\ F \\ \hline D\ 5\ 7 \end{array}$$

This can be validated using decimal addition

	1	1– → Carry	
Decimal representation	2	11	08
	10	09	15
	D	5	7

Answer: $(D57)_{16}$

1.9.2 HEXADECIMAL SUBTRACTION

The hexadecimal subtraction can be done using complement method. The subtraction can be done by both ways: difference and borrow traditional method and complement method. As the radix of hexadecimal number is 16, so, rules for subtraction using 15's complement (r-1)'s and 16's (r)'s complement is applicable here.

Example 1.46 Perform Hexadecimal subtraction 8A2 – 2B1

***Solution*:** Hexadecimal subtraction

$$\begin{array}{r} 8\ A\ 2 \\ -\ 2\ B\ 1 \\ \hline 5\ F\ 1 \end{array}$$

This can be validated using decimal addition

	16	→	Borrow
Decimal representation	8	10	2
	– 2	11	1
	5	F	1

In hexadecimal subtraction, for subtraction a bigger number from smaller number, we have to borrow number of values, 16 from successive digit.

Answer: $(5F1)_{16}$

Example 1.47 Calculate 15's complement and 16's complement of

(a) 10AB (b) 15AC

***Solution*:**

(a) Hexadecimal number = 10AB

15's complement of 10AB = FFFF – 10AB

= EF54

16's complement of 10AB = 15's complement + 1

= EF54 + 1 = EF55

Answer: $(EF55)_{16}$

(b) Hexadecimal number = 15AC

15's complement of 15AC = FFFF – 15AC

= EA53

16's complement of 15AC = 15's complement + 1

= EA53 + 1 = EA54

Answer: $(EA54)_{16}$

Example 1.48 Perform hexadecimal subtraction 45A2 – 23BC

(a) Using 15's complement (b) Using 16's complement

***Solution*:**

(a) Subtraction using 15's complement

15's complement of 23BC = FFFF – 23BC

= DC43

Add sum to minuend = DC43 + 45A2 = 121E5

As carry is there, so add the carry to LSB and this is the required result

= 21E5 + 1 = 21E6

Answer: $(21E6)_{16}$

(b) Subtraction using 16's complement

15's complement of 23BC = FFFF – 23BC

= DC43

16's complement of 23BC = DC43 + 1 = DC44

Add sum to minuend = DC44 + 45A2 = 21E6

As carry is there, so discard the carry and remaining is the required result

Answer: $(21E6)_{16}$

1.10 CODES

In digital number systems, for the fast processing of information, data is used in binary format. Various binary codes are used to represent data which may be numeric, alphabets or special characters. Some are weighted and few are non-weighted codes Although, the information used in every code is represented in binary form, yet the interpretation of this binary information is possible only if the code in which this information is available is known.

1.10.1 BINARY CODED DECIMAL (BCD)

For fast calculation and conversion, binary coded decimal numbers are used. It is used to display decimal number in binary form. In this, the decimal number is written by replacing decimal digit in integers and fractions with its four-bit binary equivalent value. The range of BCD numbers are 0-9. The most commonly used BCD code is 8421, where the digits 8, 4, 2 and 1 depicts the weight of different bits in combination of four-bits, starting from MSB to LSB. Other BCD codes are 4221 and 5421. These codes are weighted binary codes because each position of a number represents a specific weight.

Table 1.2 Weighted Codes

Decimal Digit	BCD Code		
	8421	4221	5421
0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	1000	0100
5	0101	1001	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

The easy and efficient conversion in between 8421 code numbers and familiar decimal numbers is the main advantage of this code. For example, using 8421 code, decimal digit can easily be converted to binary as 1001. On the similar manner, 4221 and 5421 code work.

Example 1.49 Convert BCD numbers to decimal numbers

(a) 10010001 (b) 00010001

***Solution*:** The equivalent decimal number of BCD is calculated by grouping of 4 digits and then converted to decimal number.

(a) Decimal conversion of BCD 1001 0001 is $(91)_{10}$

(b) Decimal conversion of BCD 0001 0001 is $(11)_{10}$

1.10.1.1 BCD ADDITION

The BCD code is commonly used in arithmetic operation. In BCD addition, if the sum exceeds than 9, it is invalid result, which can be avoided by adding 6 (0110). The six digits 10 to 15 are not allowed in BCD numbers, so 6 is added.

Addition of two BCD Numbers

Step 1: Add two BCD numbers.

Step 2: If the resultant sum is less than or equal to 9, it is valid result.

Step 3: If resultant sum is more than 9, it is an invalid result.

Step 4: Add 0110 to resultant sum, so that six invalid states 1010, 1011, 1100, 1101, 1110 and 1111 can be avoided. If carry is generated, represent the carry in four bits.

Example 1.50 Perform BCD addition on following numbers

(a) 1001 + 0001 (b) 00010001 + 01011101

***Solution*:**

(a) The two BCD numbers are 1001 and 0001

```
   1001
 + 0001
 ------
   1010        Invalid BCD, Sum is out of limits
 + 0110        Adding 0110
 ------
 1 0000        Carry is there, represent it in four bits
 ------
 = 0001 0000
```

Answer: $(0001\ 0000)_2$

(b) The two BCD numbers are 00010001 and 01010101

```
   00010001
 + 01010101
 ----------
   0110 0110   Valid BCD, Sum is within limits
 ----------
 = 0110  0110
```

Answer: $(1100\ 1100)_2$

1.10.1.2 BCD SUBTRACTION

The subtraction of BCD numbers follows the subtraction using r's complement and (r-1)'s complement rules.

Example 1.51 Perform BCD Subtraction 888 – 243 using

(a) 9's complement (b) 10's complement

***Solution*:**

(a) 9's complement of 243 = $(10^3 - 1) - 243$

$= 999 - 243$

$= 756$

Add to minuend 888 = 756 + 888 = 1644

Carry is there, so add the carry to LSB

$= 644 + 1 = 645$

Answer: $(645)_{10}$

(b) 10's complement of 243 = $(10^3) - 243$

$= 1000 - 243$

$= 757$

Add to minuend 888 = 757 + 888 = 1645

Carry is there, so discard the carry remaining is the required result.

$= 645$

Answer: $(645)_{10}$

Example 1.52 Perform BCD Subtraction 754 – 431 using 10's complement

***Solution*:**

10's complement of 431 = $(10^3) - 431$

$= 1000 - 431$

$= 569$

Add to minuend 754 = 569 + 754 = 1 323

Carry is there, so discard the carry remaining is the required result.

$= 323$

Answer: $(323)_{10}$

1.10.2 EXCESS-3 CODE

The excess-3 code is one type of non-weighted code, where 3 is added to each decimal digit and result is represented in four binary bits. This code is not positional weighted.

The excess-3 code is self-complementing code which is helpful in subtraction. The self-complementing is a code which represents 9's complement of a decimal code, when its digits are inverted.

Table 1.3 Excess-3 Code

Decimal Digit	8421 (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

The decimal digit 3 is added to each value to extract the excess-3 value.

1.10.2.1 EXCESS-3 ADDITION

The excess-3 addition follows few rules, which are discussed as below:

Rules for Excess-3 Addition

Step 1: Equate the number of bits of both numbers, which are to be added.

Step 2: Add the two numbers.

Step 3: If carry is there, add 0011 (3) to the resultant sum.

Step 4: If carry is not present, subtract 0011 (3) from the resultant sum.

Example 1.53 Perform excess-3 addition of following numbers:

(a) 1011 + 1001 (b) 0101 + 1011

***Solution*:**

(a) Both numbers are having equal number of digits i.e. 4

$$\begin{array}{r} 1011 + 1001 \\ 1011 \\ +\ 1001 \\ \hline 10100 \end{array} \quad \rightarrow \text{Carry is generated}$$

```
  0001  0100
+ 0011 0011        → Add 3 (0011) to the sum
-----------
  0100  0111       → Result
-----------
```

Answer: $(0100\ 0111)_2$

(b) Both numbers are having equal number of digits i.e. 4;

```
0101 + 1011
   0101
 + 1011
 ------
   1111        → No Carry is generated
 ------
   1111
 – 0011        Subtract 3 (0011) from the sum
 ------
   1100        → Result
 ------
```

Answer: $(1100)_2$

Example 1.54 Perform excess-3 addition of following decimal numbers

(a) 46 (b) 430

***Solution*:**

(a)

```
   4  6
 + 3  3
 -------
   7  9
 -------
   ↓  ↓
0111 1001
```

(b)

```
   4  3  0
 + 3  3  3
 ----------
   7  6  3
 ----------
   ↓  ↓  ↓
0111 0110 0011
```

Answer: (a) $(01111001)_2$ (b) $(011101100011)_2$

1.10.2.2 EXCESS-3 SUBTRACTION

The rules for subtraction using excess-3 are as follow:

Rules for Excess-3 Subtraction

Step 1: Equate the number of bits of both numbers, which are to be added. Calculate excess-3 of both numbers.

Step 2: 1's Complement the subtrahend.

Step 3: Add complemented subtrahend to minuend.

Step 4: If carry is there, add 0011 (3) to the resultant sum and end around the carry.

Step 5: If carry is not present, subtract 0011 (3) from the resultant sum. Take 1's complement of the result. It is a negative number.

Example 1.55 Perform excess-3 subtraction of following numbers:

(a) $(3)_{10} - (6)_{10}$ (b) $(6)_{10} - (3)_{10}$

***Solution*:**

(a) Excess-3 of minuend 3 = 3 + 3 = 6 = 0110

Excess-3 of subtrahend 6 = 6 + 3 = 9 = 1001

1's Complement of excess-3 of subtrahend = 0110

Add the complemented value to minuend = 0110

$$\begin{array}{r} +\ 0110 \\ \hline 1100 \end{array}$$

As carry is not present, subtract 0011 from resultant

$$\begin{array}{r} =\ 1100 \\ -\ 0011 \\ \hline 1101 \end{array}$$

It is a negative number. Take again the 1's complement of result = 0010

Answer: $-(0010)_2$

(b) Excess-3 of minuend 6 = 6 + 3 = 9 = 1001

Excess-3 of subtrahend 3 = 3 + 3 = 6 = 0110

1's Complement of excess 3 of subtrahend = 1001

Add the complemented value to minuend = 1001

$$\begin{array}{r} +1001 \\ \hline 10010 \end{array}$$

As carry present, add the carry to LSB = 0010

$$\begin{array}{r} +\ 1 \\ \hline 0011 \end{array}$$

Answer: $(0011)_2$

1.10.3 GRAY CODE

The Gray code is one type of non-weighted binary code. It is not a positional weight. It is a binary code that progresses in such a way that between two successive codes, only one

bit is changed. It is not an arithmetic code. The single bit change property is important in some applications, e.g., shaft position encoders. In these applications the chances of error increase if more than one-bit change occurs. Gray codes are also called reflected codes.

Decimal Digit	Binary	Gray
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

1.10.3.1 BINARY TO GRAY CODE CONVERSION

The rules for changing binary number into equivalent Gray code are:

1. The left most bit (most significant bit) in Gray code is the same as the left most bit in binary

   1 0 1 1 Binary
   ↓
   1 Gray

2. Add the left most bit to the adjacent bit

   1 + 0 1 1
   1 1

3. Add the next adjacent pair

   1 0 + 1 1
   1 1 1 0

4. Add the next adjacent pair and discard carry

   1 0 1 + 1
   1 1 1 0

5. Continue the above process till completion.

Example 1.56 Convert the following Binary numbers to Gray numbers:

(a) 10110 (b) 01010

***Solution*:**

(a) $(10110)_2 \rightarrow (11101)_{gray}$ (b) $(01010)_2 \rightarrow (01101)_{gray}$

1.10.3.2 GRAY TO BINARY CONVERSION

The method to convert from Gray code to binary is an under:

1. Left most bit in binary is the same as the left most bit in Gray code.

```
1   1   0   1   1   Gray
↓
1                   Binary
```

2. Add the binary MSB to the Gray digit in the adjacent position. Discard carry

```
1   1   0   1   1   Gray
↓↗
1   0               Binary
```

3. Add the binary digit generated in step 2 to the next Gray digit. Discard carry

```
1   1   0   1   1   Gray
↓  ↗
1   0   0           Binary
```

4. Continue the above process till all the digits are covered. Discard Carry in each case

```
1   1   0   1   1   Gray
↓    ↗
1   0   0   1   0   Binary
```

Example 1.57 Convert the following Gray numbers to Binary numbers:

(a) 11101 (b) 01101

***Solution*:**

(a) $(11101)_{gray} \rightarrow (10110)_2$ (b) $(01101)_2 \rightarrow (01010)_{gray}$

1.10.4 SEQUENTIAL CODE

A code is said to be sequential when two subsequent codes, seen as numbers in binary representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.

1.10.5 ALPHANUMERIC CODE

Alphanumeric codes are used to represent the letters of alphabets and numeric characters. For efficient, secure and fast communication, we need alphabets, numeric as well as digits. So, alphanumeric codes are widely used.

1.10.5.1 ASCII CODE

The most widely accepted code is called the American Standard Code for Information Interchange (ASCII). It is a 7-bit code for representing alphanumeric and control characters. The ASCII code associates an integer value for each symbol in the character set, such as letters, digits, punctuation marks, special characters, and control characters. For printers and small computation systems, ASCII codes are used. Usually, it is of 7-bit nature but sometimes 8^{th} bit is associated with it, which represents parity. The characters are assigned in ascending binary numbers.

Table 1.4 ASCII codes

Least Significant Bit (LSB)	Most Significant Bit (MSB)							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	\	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	‘	<	L	\	l	|
1101	CR	GS	-	=	M	]	M	}
1110	SO	RS	.	>	N		N	~
1111	SI	US	/	?	O	-	O	DEL

From ASCII Table 1.4, we can code any letter, numeric or special character in 7-bit format. For example, C can be coded as 0011100 and = can be coded as 1101011. The definitions of various control abbreviations, used in this Table 1.4 are elaborated in Table 1.5.

Table 1.5 Control Abbreviations

NUL	Null	DLE	Data Link Space
SOH	Start of heading	DC1	Direct Control 1
STX	Start Text	DC2	Direct Control 2
ETX	End Text	DC3	Direct Control 3
EOT	End of Transmission	DC4	Direct Control 4
ENQ	Enquiry	NAK	Negative Acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End of Transmission Block
BS	Back Space	CAN	Cancel
HT	Horizontal Tab	EM	End of Medium
LF	Line Feed	SUB	Substitute
VT	Vertical Tab	ESC	Escape
FF	Form Feed	FS	Form Separator
CR	Carriage Return	GS	Group Separator
SO	Shift Out	RS	Record Separator
SI	Shift In	US	Unit Separator

1.10.5.2 EBCDIC CODE

EBCDIC is Extended Binary Coded Decimal Interchange Code. It is 8-bit code and mostly used in large computers for communication and coding purpose. This code includes lowercase and capital alphabets, numeric from 0 to 9 and special characters. With 8-bit, 256 characters can be coded. The Table 1.6 shows EBCDIC code for alphabets and special characters.

Table 1.6 EBCDIC code

Alphabets	EBCDIC	Alphabets/numeric	EBCDIC	Special Character	EBCDIC
A	11000001	S	11100010	<	01001011
B	11000010	T	11100011	(	01001100
C	11000011	U	11100100	+	01001101
D	11000100	V	11100101	/	01000000
E	11000101	W	11100110	&	01001011
F	11000110	X	11100111	:	01111011
G	11000111	Y	11101000	#	01111011
H	11001000	Z	11101001	@	01111100
I	11001001	0	11110000	‘	01111101
J	11010011	1	11110001	=	01111110

Contd…

Alphabets	EBCDIC	Alphabets/numeric	EBCDIC	Special Character	EBCDIC
K	11010010	2	11110010	“	01111111
L	11010011	3	11110011	‘	01101011
M	11010100	4	11110100	%	01101100
N	11010101	5	11110101	-	01101101
O	11010110	6	11110110	>	01101110
P	11010111	7	11110111		
Q	11011000	8	11111000		
R	11011001	9	11111001		

We can easily write the code in 8-bit format by looking carefully the EBCDIC table.

1.11 ERROR DETECTION CODE

Each and every digit is important in digital system. When a binary string is transferred from one channel to another. Sometimes at receiver end, the receiving string is not same as starting string. That indicates that there is some error in the transmission channel. Due to this reason, an extra bit is associated with binary number which identifies the error. That extra bit is known as Parity bit. The value of parity may be zero or one.

1.11.1 PARITY BIT

A parity is a system that checks the errors in a binary string by counting the number of 1’s. it is said to be odd parity, when the number of 1’s is odd and it is said to be even parity, when the number of 1’s is even.

1100110011 even parity; because number of ones are even

1110001111 odd parity; because number of ones are odd

For error detection, we can use 7-bits for data and 8^{th} bit for parity. If the numbers of 1’s are even, then it is even parity. The even parity can be converted to odd parity by appending one more 1 in the binary string. In such a way, number of 1’s is odd.

1.11.2 CHECK SUM

The above parity check method is useful to detect one-bit error. But there may be the possibility that two errors are in the same word. The parity check method is not supportive in this case. So, to detect two errors in same word, check sum is used. As a word is transmitted, it is added to previous word and sum is retained at sending end. For example,

Word A: 11100110

Word B: 00011001

Sum: 11111111

The sum is also sent and at receiving end it is checked. In such manner, two errors can be detected in a word. This method is used in teleprocessing.

Example 1.58 Find the parity bit to make data of odd parity.

(a) 0011011 (b) 1100111

***Solution*:**

(a) Data = 0011011
Number of 1's = 4
Parity Bit = 1 (By adding parity bit, the number has odd number of 1's)

(b) Data = 1100111
Number of 1's = 5
Parity Bit = 0 (Number of 1's in data is already odd, so parity is 0)

1.11.3 PARITY DATA CODES

With each character, a parity bit can be added. The method for detecting error is 2 out of 5 code and bi-quinary 5043210 code, as shown in Table 1.7.

The 2 out of 5 code has five bits which represent decimal digits 0 to 9. Each code word has two 1's. After transmission, the number of 1's can be checked at receiving end. If it is other than 2, it indicates that error is there.

Table 1.7 Parity data codes

Decimal Digit	2 out of 5 Codes	Bi-Quinary 5043210 Code
0	00011	0100001
1	00101	0100010
2	00110	0100100
3	01001	0101000
4	01010	0110000
5	01100	1000001
6	10001	1000010
7	10010	1000100
8	10100	1001000
9	11000	1010000

The bi-quinary 5043210 code has positional weightage and used in counters. The two-bit group having weights 5 indicates whether number is < or = or > 5. The five-bit group indicates the count below or above 5.

1.12 ERROR CORRECTING CODE

A method is developed by R W Hammings which is known as Hamming Code. This code is used for error correction. This will identify as well as correct the error. The code uses a number of parity bits located at certain positions in code.

1.12.1 NUMBER OF PARITY BITS

The number of parity bits can be calculated by this method. If the number of data bits is m, then number of parity bits p, is determined by following relationship:

$$2^p \geq m + p + 1 \qquad .....(1.1)$$

For example, a data 1101 has 4 bits. The parity bit can be calculated by hit and trial method as:

$$p = 2$$

i.e., $2^2 \geq 4 + 2 + 1$

$= 4 < 7$

So, it does not satisfy the equation 1.1

We can try for p = 3;

i.e. $2^3 \geq 4 + 3 + 1$

$8 = 8$

This satisfies equation (1.1.). so, the number of parity bits are 3.

The parity bit always occupies 2^n position i.e. 1, 2, 4 etc. positions are reserved for parity bit. If P is parity bit and D is data bit then the format will be as:

D7 D6 D5 P4 D3 P2 P1

The bit P1 is set so that it establishes even parity over bits 1, 3, 5 and 7 (i.e., data bits D3, D5, D7 and itself P1). P2 is set for even parity over bits, 2, 3, 6 and 7 (i.e., D3, D6, D7 and itself P2). P4 is set for even parity over bits 4, 5, 6, 7 (i.e., D5, D6, D7 and itself P4).

At the receiver end, each group is checked for even parity. If an error is indicated, it is located by forming a p bit binary number formed by the p parity bits.

Example 1.59 A seven bit Hamming code as received is 11111201. Check if it is correct. If not find the correct code if even parity is used.

***Solution*:**

D7	D6	D5	P4	D3	P2	P1
1	1	1	1	1	0	1

As the bits 4, 5, 6, 7 have even number of 1s. Hence, no error

As the bits 2, 3, 6, 7 have odd number of 1s. Hence, error

As the bits 1, 3, 5, 7 have even number of 1s. Hence, no error

Evidently, the error is in bit 2 position.

The correct code is 1111111.

Example 1.60 The seven bit Hamming code as received is 0010001. Assuming that even parity has been used, check if it is correct. If not find the correct code.

***Solution*:**

D7	D6	D5	P4	D3	P2	P1
0	0	1	0	0	0	1

As the bits 4, 5, 6, 7 have odd number of 1s. Hence, error

As the bits 2, 3, 6, 7 have even number of 1s. Hence, no error

As the bits 1, 3, 5, 7 have even number of 1s. Hence, no error.

Evidently, the error in bit 4.

The correct code is 0011001.

Example 1.61 The seven bit Hamming code as received is 1110111. Assuming that even parity has been used, check if it is correct. If not find the correct code.

***Solution*:**

D7	D6	D5	P4	D3	P2	P1
1	1	1	0	1	1	1

As the bits 4, 5, 6, 7 have odd number of 1s. Hence, error

As the bits 2, 3, 6, 7 have even number of 1s. Hence, no error

As the bits 1, 3, 5, 7 have even number of 1s. Hence, no error.

Evidently, the error in bit 4.

The correct code is 1111101.

SHORT ANSWER QUESTIONS

Ques: 1.62 Define bit and byte in a digital system.

***Solution*: *Bit*:** It is a fundamental unit of binary number system. It can be represented as 0 or 1.

***Byte*:** The group of 8 bits is called a byte. It estimates the size of computer memory.

Ques: 1.63 What are the limitations of Digital circuits?

***Solution*:** As the real world is analog based. So, DAC (digital to analog converters) are used which operates on digital information and convert digital blocks to real-world analog form.

Ques: 1.64 The presence and absence of students in a class represent 1 or 0 respectively. Represent the binary form of attendance sheet.

***Solution*:**

P	P	A	A	P
A	P	P	A	A
A	A	A	P	P

Row 1: 11001; Row 2: 01100; Row 3: 00011

Ques: 1.65 Express generally, a number in any digital number system

***Solution*:** If a number is having n digits and d values per position, r represents the radix of a number and X is the number. Then it can be generally displayed as:

$$X = d_n r^n + d_{n-1} r^n - 1 + \ldots\ldots\ldots\ldots + d_1 r^1 + d_0 r^0$$

Ques: 1.66 Explain the terms (a) MSB (b) LSB

***Solution*:** The bit which is having the highest power of radix is known as most significant bit (MSB). It is at the extreme left.

The bit which is having the lowest power of radix is known as least significant bit (LSB). It is at the extreme right.

Ques: 1.67 Define radix point

***Solution*:** The point which separates integer and fraction part, when expressed in the form of positional radix. For example, in binary system, the number can be expressed as:

$$2^4\ 2^3\ 2^2\ 2^1 2^0\ .\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}$$

↑

Radix point

Ques: 1.68 Give the daily life example of analog system and digital system

***Solution*: *Analog system*:** Analog voltmeter, analog ammeter, analog clock, analog weighing scale, old radio, old telephone handset, record player etc.

***Digital system*:** Digital clock, digital multimeter, home security system, digital alarm clock, compact disc player, digital wrist watch, digital scale, counter etc.

Ques: 1.69 What is positional weight system?

***Solution*:** In positional weight system, position of digit determines the weight. It is associated with radix point. Any number system can be represented using positional weight system.

Ques: 1.70 In number 4150 and 1450, determine the weight of 4.

***Solution*:** In 4150, the weight of 4 is 1000 and in 1450, the weight of 4 is 100.

Ques: 1.71 Explain true complement of a binary number.

***Solution*:** The true complement of binary system is known as 2's complement. In this, subtract each digit of number from r-1 and then add 1 to LSB

Ques: 1.72 Which logic gate satisfies the operation of 1's complement?

***Solution*:** The NOT gate having inverted operation satisfies the 1's complement function.

Ques: 1.73 Explain two-part conversion method.

***Solution*:** While converting decimal to binary number, two-part method is used. Where integer part of the decimal number is divided by 2 again and again. Mention 0 if remainder is not present, otherwise write 1. The first division represents LSB and last division represents MSB.

Ques: 1.74 Explain 10 as a signed magnitude number

***Solution*:** The decimal number $10 = (1010)_2$

Positive number $+ 10 = (0\ 1010)_2$

Negative number $- 10 = (1\ 1010)_2$

Ques: 1.75 Why hexadecimal numbers are preferred mostly?

***Solution*:** Representing the number in binary system takes so much bits. It produces a long string. It is difficult to read and understand such long numbers. The hexadecimal numbers are formed by grouping 4 bits. The data becomes concise and simple. So, hexadecimal number system is preferred over other number systems.

Ques: 1.76 What is the minimum to maximum range for two-digit, three-digit and four digit hexadecimal number?

***Solution*:** The two-digit hexadecimal number is $(FF)_{16.}$ The range of number is from 0 to $(r^2 - 1)$ i.e. 0 to 255. The three-digit hexadecimal number is $(FFF)_{16.}$ The range of number is from 0 to $(r^2 - 1)$ i.e. 0 to 4095. The four-digit hexadecimal number is $(FFFF)_{16.}$ The range of number is from 0 to $(r^2 - 1)$ i.e. 0 to 65535.

Ques: 1.77 Find X, Y and radix of following equation:

$(XY)r = (25)_{10}$ and $(YX)r = (31)_{10}$.

***Solution*:**

Since the base or radix is r. So,

$$Xr^1 + Yr^0 = 25$$

or $$Xr + Y = 25 \quad .....(1)$$

and $$Yr^1 + Xr^0 = 31$$

or $$Yr + X = 31 \quad .....(2)$$

Also, $$Y = X + 1 \quad .....(3)$$

From Eqns. (1), (2) and (3)

$$X = 3, Y = 4 \text{ and } r = 7.$$

Ques: 1.78 Explain the different formats of binary floating-point numbers

***Solution*:** The different formats for floating point numbers are:

(i) single precision: it has 32 bits
(ii) double precision: it has 64 bits
(iii) extended precision: it has 80 bits

Ques: 1.79 What is the advantage of octal number system?

***Solution*:** For expressing long sequence of binary numbers, octal number system is attractive and concise one, where grouping of 3 binary bits are created.

Ques: 1.80 Define nibble and word.

***Solution*: *Nibble*:** The group of 4 bits is called nibble. It is half size of a byte.

***Word*:** The combination of two or more bits formulates a word. Word length can be 16, 32, or 64 bits etc.

Ques: 1.81 What is two-state operation in digital systems?

***Solution*:** In digital systems, two-state operation means Level Low and Level High. The voltage having value 5 V is represented by logic 1 and it is the part of Level High. Whereas, 0 V represents Logic 0 or Level Low.

PREVIOUS YEAR GATE QUESTIONS

Ques: 1.82 The smallest integer that can be represented by an 8-bit number in 2's complement form is: ***(GATE 2013)***

(a) –256 (b) –128 (c) –127 (d) 0

Answer: (b)

[Hint: For n bit 2's complement numbers, range of number is -(2(n-1)) to +(2(n-1)-1)]

Ques: 1.83 The number 43 in 2's complement representation is: ***GATE 2000)***

(a) 01010101 (b) 11010101 (c) 00101011 (d) 10101011

Answer: (c)

[Hint: 2's complement numbers is 1's complement +1, it is a positive number]

Ques: 1.84 The 2's complement representation of the decimal value -15 ***(GATE 2002)***

(a) 1111 (b) 11111 (c) 111111 (d) 10001

Answer: (d)

[Hint: 2's complement numbers is 1's complement +1, it is a negative number]

Ques: 1.85 The decimal value 0.25 is: ***(GATE 2002)***

(a) is equivalent to binary 0.1
(b) is equivalent to binary 0.01
(c) is equivalent to binary 0.00111
(d) cannot be represented precisely in binary

Answer: (b)

[Hint: convert decimal into binary number, by multiplying the fraction by radix 2]

Ques: 1.86 Given $\sqrt{(224)r} = (13)r$; The value of the radix r is: ***(GATE 1997)***

(a) 10 (b) 8 (c) 5 (d) 6

Answer: (c)

[Hint: $2r^2 + 2r^1 + 4r^0 = (1r^1 + 3r^0)^2$; $2r^2 + 2r + 4 = (r + 3)^2$; $r^2 - 4r - 5 = 0 \Rightarrow r=5$]

Ques: 1.87 Let A = 1111 1010 and B = 0000 1010 be two 8-bit 2's complement numbers. Their product in 2's complement is: ***(GATE 2004)***

(a) 1100 0100 (b) 1001 1100 (c) 1010 0101 (d) 1101 0101

Answer: (a)

[Hint: Since MSB of a is 1, 2's complement of A = 0000 0110. Therefore A = –6 B = 10. Since product is a negative number we have to find the 2's complement of -60]

Ques: 1.88 The 2's complement representation of $(-539)_{10}$ in hexadecimal is: ***(GATE 2001)***

(a) ABE (b) DBC (c) DE5 (d) 9E7

Answer: (c)

[Hint: The 2's complement of -539 is 1101 1110 0101. Represent it in hexadecimal]

Ques: 1.89 P is a 16-bit signed integer. The 2's complement representation of P is $(F87B)_{16}$. The 2's complement representation of 8 x P is: ***(GATE 2010)***

(a) $(C3D8)_{16}$ (b) $(187B)_{16}$ (c) $(F878)_{16}$ (d) $(987B)_{16}$

Answer: (a)

[Hint: Convert hexadecimal number into binary: 1111 1000 0111 1011; MSB is 1; it is a negative number. Take 2's complement. Represents in binary by clubbing 4 digits]

Ques: 1.90 The decimal value 0.5 in IEEE single precision floating point representation has: ***(GATE 2012)***

(a) fraction value of 0000….000 and exponential value 0
(b) fraction value of 0000….000 and exponential value -1
(c) fraction value of 1000….000 and exponential value 0
(d) no exact representation

Answer: (b)

[Hint: 0.5 in base 10 means 1 X 2-1 in base 2]

Ques: 1.91 $(1217)_8$ is equivalent to: ***(GATE 2009)***

(a) $(1217)_{16}$ (b) $(028F)_{16}$ (c) $(2297)_{10}$ (d) $(0B17)_{16}$

Answer: (b)

[Hint: use the octal number system conversion rules]

Ques: 1.92 In the IEEE floating point representation, the hexadecimal value 0 × 00000000 corresponds to: ***(GATE 2008)***

(a) the normalized value 2-127 (b) the normalized value 2-126
(c) the normalized value +0 (d) the special value +0

Answer: (d)

[Hint: Floating point rules see section 1.5.2.]

Ques: 1.93 The range of integers that can be represented by an n bit 2's complement number system is: ***(GATE 2005)***

(a) -2^{n-1} to $(2^{n-1} - 1)$ (b) $-(2^{n-1} - 1)$ to $(2^{n-1} - 1)$
(c) -2^{n-1} to 2^{n-1} (d) $-(2^{n-1} + 1)$ to $(2^{n-1} + 1)$

Answer: (a)

[Hint: Sign character is of 8 bits; So, -128 to 127 characters can be stored]

Ques: 1.94 If $(73)_x = (54)_y$; then find the values of x & y. ***(GATE 2004)***

(a) 8, 16 (b) 10, 12 (c) 9, 13 (d) 8, 11

Answer: (d)

[Hint: Convert both numbers in decimal and find the value of x and y)

Ques: 1.95 2's complement representation of 16-bit number is FFFF. Its magnitude in decimal representation is: ***(GATE 1997)***

(a) 0 (b) 1 (c) 32,767 (d) 65,535

Answer: (b)

[Hint: use 2's complement rule of hexadecimal numbers]

Ques: 1.96 2's complement representation of a 2's complemented number 1101 is: ***(GATE 1998)***

(a) 110100 (b) 001101 (c) 110111 (d) 111101

Answer: (d)

[Hint: use the rules of 2's complement]

Ques: 1.97 What is the 4bit 2's complement representation a decimal number 1000. ***(GATE 2002)***

(a) + 8 (b) 0 (c) –7 (d) –8

Answer: (d)

[Hint: use the rules of 2's complement]

Ques: 1.98 The range of signed decimal number which can be represented by 6-bit 1's complement form is: ***(GATE 2004)***

(a) –31 to +31 (b) –63 to +64 (c) –64 to +63 (d) –32 to +31

Answer: (a)

[Hint: use the rules of signed bit representation]

Ques: 1.99 Decimal 43 in hexadecimal and BCD number system is: ***(GATE 2005)***

(a) B2, 0100 0011 (b) 2B, 0100 0011 (c) 2B, 0011 0100 (d) B2, 0100 0011

Answer: (b)

[Hint: use decimal to hexadecimal and decimal to BCD conversion rules]

Ques: 1.100 The two numbers represented in signed 2's complement form is P = 11101101 and Q = 11100110. If Q is subtracted from P, the value obtained in 2's complement form is: ***(GATE 2008)***

(a) 00000111 (b) 00000111 (c) 1111001 (d) 111111001

Answer: (b)

[Hint: use the rules of 2's complement]

Ques: 1.101 If X = 01110 & Y = 11001 are two 5-bit binary numbers represented in 2's complement form. Using 6 bits, represent X+Y in 2's complement form. ***(GATE 2007)***

(a) 100111 (b) 001000 (c) 000111 (d) 101001

Answer: (c)

[Hint: use the rules of 2's complement]

PRACTICE QUESTIONS

Ques: 1.102 Determine the decimal numbers represented by the following binary numbers:

(a) $(10111.1001)_2$ (b) $(10.1001)_2$ (c) $(11111110)_2$

Ques: 1.103 Determine the binary numbers represented by the following decimal numbers:

(a) $(242.14)_{10}$ (b) $(177.77)_{10}$ (c) $(446.664)_{10}$

Ques: 1.104 Perform the binary addition:

(a) $(1011.1101)_2 + (101.001)_2$ (b) $(10101)_2 + (01011)_2$

Ques: 1.105 Perform the following subtractions using 2's complement method.

(a) $(0100)_2 - (01001)_2$ (b) $(01100)_2 - (0001)_2$

Ques: 1.106 Convert the following numbers from decimal to octal and then to binary.

(a) $(646.13)_{10}$ (b) $(7456)_{10}$ (c) $(131.311)_{10}$

Ques: 1.107 Convert the following binary numbers to octal and then to decimal.

(a) $(01011100.1001010)_2$ (b) $(11110011.010101)_2$

Ques: 1.108 Encode the following decimal numbers in BCD code:

(a) $(461.767)_{10}$ (b) $(327.89)_{10}$

Ques: 1.109 Mention your name in ASCII code and your university name in EBCDIC code.

Ques: 1.110 Convert following number in excess-3 code:

(a) $(743)_{10}$ (b) $(10011001)_2$ (c) $(457.754)_{10}$

Ques: 1.111 Convert following binary number into gray numbers:

(a) $(10010011)_2$ (b) $(00110011)_2$

Ques: 1.112 Convert following gray number into binary numbers:

(a) $(11011011)_{gray}$ (b) $(11110011)_{gray}$

Ques: 1.113 Perform the binary multiplication:

(a) $(1101.101)_2 \times (1101)_2$ (b) $(101.\ 101)_2 \times (110.110)_2$

Ques: 1.114 Carry out the following binary division:

(a) $(11111111)_2 \div (110011)_2$ (b) $(110000)_2 \div (100)_2$

Ques: 1.115 Perform 1's complement operation on following binary numbers:

(a) $(110011001.101)_2$ (b) $(101.00101)_2$

Ques: 1.116 Perform 2's complement operation on following binary numbers:

(a) $(1101101.101)_2$ (b) $(010101.10101)_2$