Module 1

Number Systems and codes

Motivation:

Digital Logic Design deals with binary numbers. Hence the concept of number systems and codes becomes the basis for learning this course. By learning number system, the students get to know about different number systems and the conversion between different number systems.

Syllabus:

Lecture no	Content	Duration (Hr)	Self-Study (Hrs)
1	**Introduction to number system and conversions:** Binary, Octal	1	1
2	Decimal and Hexadecimal number Systems	1	1
3	Binary arithmetic: addition, subtraction (1's and 2's complement), multiplication and division	1	2
4	Octal arithmetic: Addition and Subtraction (7's and 8's complement method for octal)	1	2
5	Hexadecimal arithmetic: Addition and Subtraction (15's and 16's complement method for Hexadecimal)	1	2
6	**Codes:** Gray Code, BCD Code	1	2
7	Excess-3 code, ASCII Code	1	2
8	**Error Detection and Correction:** Hamming codes	1	2

Learning Objectives:

Learners shall be able to:
1. To explain the different number system formats
2. To compute the conversion between different number systems.
3. To compute binary, octal and hexadecimal arithmetic.
4. To identify different codes and perform operations on them

Theoretical Background:

Most number systems follow a common pattern for writing down the value of a number:
A fixed number of values can be written with a single numerical character, then a new column is used to count how many times the highest value in the counting system has been reached. The number of numerical values the system uses is called the base of the system. For example, the decimal system has 10 numerical characters and so has a base of 10:
0 1 2 3 4 5 6 7 8 9

For writing numbers greater than 9 a second column is added to the left, and this column has 10 times the value of the column immediately to its right.
Because number systems commonly used in digital electronics have different base values to the decimal system, they look less familiar, but work in essentially the same way.

Key Definitions:

Number System: Number systems are the technique to represent numbers in the computer system architecture, every value that you are saving or getting into/from computer memory has a defined number system.

Computer architecture supports following number systems.

- Binary number system
- Octal number system
- Decimal number system
- Hexadecimal (hex) number system

Binary Number System: A Binary number system has only two digits that are **0 and 1**. Every number (value) represents with 0 and 1 in this number system. The base of binary number system is 2, because it has only two digits.

Octal number system: Octal number system has only eight (8) digits from **0 to 7**. Every number (value) represents with 0,1,2,3,4,5,6 and 7 in this number system. The base of octal number system is 8, because it has only 8 digits.

Decimal number system: Decimal number system has only ten (10) digits from **0 to 9**. Every number (value) represents with 0,1,2,3,4,5,6, 7,8 and 9 in this number system. The base of decimal number system is 10, because it has only 10 digits.

Hexadecimal number system: A Hexadecimal number system has sixteen (16) alphanumeric values from **0 to 9** and **A to F**. Every number (value) represents with 0,1,2,3,4,5,6, 7,8,9,A,B,C,D,E and F in this number system. The base of hexadecimal number system is 16, because it has 16 alphanumeric values. Here **A is 10**, **B is 11**, **C is 12**, **D is 13**, **E is 14** and **F is 15**.

Gray Code: Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit). It is not weighted that means it does not depends on positional value of digit. This cyclic variable code that means every transition from one value to the next value involves only one bit change.

BCD Code: BCD is simply the 4-bit binary code representation of a decimal digit with each decimal digit replaced in the integer and fractional parts with its binary equivalent. BCD Code uses four bits to represent the 10 decimal digits of 0 to 9.

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.

Course Content

Lecture 1

1.1 Introduction to number system and conversions:

A digital system can understand positional number system only where there are a few symbols called digits and these symbols represent different values depending on the position they occupy in the number.
A value of each digit in a number can be determined using

- The digit
- The position of the digit in the number
- The base of the number system (where base is defined as the total number of digits available in the number system).

Decimal Number System

The number system that we use in our day-to-day life is the decimal number system. Decimal number system has base 10 as it uses 10 digits from 0 to 9. In decimal number system, the successive positions to the left of the decimal point represents units, tens, hundreds, thousands and so on.

Each position represents a specific power of the base (10). For example, the decimal number 1234 consists of the digit 4 in the units position, 3 in the tens position, 2 in the hundreds position, and 1 in the thousands position, and its value can be written as

$(1\times1000) + (2\times100) + (3\times10) + (4\times l)$
$(1\times10^3) + (2\times10^2) + (3\times10^1) + (4\times l0^0)$
$1000 + 200 + 30 + 1$
1234

Binary Number System

Characteristics

- Uses two digits, 0 and 1.
- Also called base 2 number system
- Each position in a binary number represents a 0 power of the base (2). Example: 2^0
- Last position in a binary number represents an x power of the base (2). Example: 2^x where x represents the last position - 1.

Example

Binary Number: 10101_2

Calculating Decimal Equivalent −

Step	Binary Number	Decimal Number
Step 1	10101_2	$((1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	10101_2	$(16 + 0 + 4 + 0 + 1)_{10}$
Step 3	10101_2	21_{10}

Let's check the take away from this lecture

1) If the decimal number is a fraction then its binary equivalent is obtained by _____ the number continuously by 2.

a) Dividing

b) Multiplying

c) Adding

d) Subtracting

2) The decimal equivalent of the binary number $(1011.011)_2$ is _____

a) $(11.375)_{10}$

b) $(10.123)_{10}$

c) $(11.175)_{10}$

d) $(9.23)_{10}$

3) How can you represent a decimal point?

a) By a series of coefficients

b) By weight decided by its position

c) By location as well as base

d) None of the above

Exercise

Q.1 Discuss the characteristics of a decimal number system.

Q.2 Convert $(1010.01)_2$ to its decimal equivalent.

Q.3 Convert $(152.25)_{10}$ to its binary equivalent.

Learning from this lecture: Learners will be able to understand decimal and binary number systems.

Lecture 2

Octal Number System

Characteristics

- Uses eight digits, 0,1,2,3,4,5,6,7.
- Also called base 8 number system
- Each position in an octal number represents a 0 power of the base (8). Example: 8^0
- Last position in an octal number represents an x power of the base (8). Example: 8^x where x represents the last position - 1.

Example

Octal Number − 12570_8

Calculating Decimal Equivalent −

Step	Octal Number	Decimal Number
Step 1	12570_8	$((1 \times 8^4) + (2 \times 8^3) + (5 \times 8^2) + (7 \times 8^1) + (0 \times 8^0))_{10}$
Step 2	12570_8	$(4096 + 1024 + 320 + 56 + 0)_{10}$
Step 3	12570_8	5496_{10}

Hexadecimal Number System

Characteristics

- Uses 10 digits and 6 letters, 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
- Letters represents numbers starting from 10. A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
- Also called base 16 number system.
- Each position in a hexadecimal number represents a 0 power of the base (16). Example 16^0.

- Last position in a hexadecimal number represents an x power of the base (16). Example 16^x where x represents the last position - 1.

Example −

Hexadecimal Number: $19FDE_{16}$

Calculating Decimal Equivalent −

Step	Hexadecimal Number	Decimal Number
Step 1	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (F \times 16^2) + (D \times 16^1) + (E \times 16^0))_{10}$
Step 2	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (15 \times 16^2) + (13 \times 16^1) + (14 \times 16^0))_{10}$
Step 3	$19FDE_{16}$	$(65536 + 36864 + 3840 + 208 + 14)_{10}$
Step 4	$19FDE_{16}$	106462_{10}

Decimal to Other Base System

Steps

- **Step 1** − Divide the decimal number to be converted by the value of the new base.
- **Step 2** − Get the remainder from Step 1 as the rightmost digit (least significant digit) of new base number.
- **Step 3** − Divide the quotient of the previous divide by the new base.
- **Step 4** − Record the remainder from Step 3 as the next digit (to the left) of the new base number.

Repeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.

The last remainder thus obtained will be the Most Significant Digit (MSD) of the new base number.

Example −

Decimal Number: 29_{10}

Calculating Binary Equivalent −

Step	Operation	Result	Remainder

Step 1	29 / 2	14	1
Step 2	14 / 2	7	0
Step 3	7 / 2	3	1
Step 4	3 / 2	1	1
Step 5	1 / 2	0	1

As mentioned in Steps 2 and 4, the remainders have to be arranged in the reverse order so that the first remainder becomes the Least Significant Digit (LSD) and the last remainder becomes the Most Significant Digit (MSD).

Decimal Number − 29_{10} = Binary Number − 11101_2.

Other Base System to Decimal System

Steps

- **Step 1** − Determine the column (positional) value of each digit (this depends on the position of the digit and the base of the number system).
- **Step 2** − Multiply the obtained column values (in Step 1) by the digits in the corresponding columns.
- **Step 3** − Sum the products calculated in Step 2. The total is the equivalent value in decimal.

Example

Binary Number − 11101_2

Calculating Decimal Equivalent −

Step	Binary Number	Decimal Number
Step 1	11101_2	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	11101_2	$(16 + 8 + 4 + 0 + 1)_{10}$
Step 3	11101_2	29_{10}

Binary Number − 11101_2 = Decimal Number − 29_{10}

Other Base System to Non-Decimal System

Steps

- **Step 1** − Convert the original number to a decimal number (base 10).
- **Step 2** − Convert the decimal number so obtained to the new base number.

Example

Octal Number − 25_8

Calculating Binary Equivalent −

Step 1 − Convert to Decimal

Step	Octal Number	Decimal Number
Step 1	25_8	$((2 \times 8^1) + (5 \times 8^0))_{10}$
Step 2	25_8	$(16 + 5)_{10}$
Step 3	25_8	21_{10}

Octal Number − 25_8 = Decimal Number − 21_{10}

Step 2 − Convert Decimal to Binary

Step	Operation	Result	Remainder
Step 1	21 / 2	10	1
Step 2	10 / 2	5	0
Step 3	5 / 2	2	1
Step 4	2 / 2	1	0

Step 5	1 / 2	0	1

Decimal Number − 21_{10} = Binary Number − 10101_2

Octal Number − 25_8 = Binary Number − 10101_2

Shortcut method - Binary to Octal

Steps

- **Step 1** − Divide the binary digits into groups of three (starting from the right).
- **Step 2** − Convert each group of three binary digits to one octal digit.

Example

Binary Number − 10101_2

Calculating Octal Equivalent −

Step	Binary Number	Octal Number
Step 1	10101_2	010 101
Step 2	10101_2	2_8 5_8
Step 3	10101_2	25_8

Binary Number − 10101_2 = Octal Number − 25_8

Shortcut method - Octal to Binary

Steps

- **Step 1** − Convert each octal digit to a 3-digit binary number (the octal digits may be treated as decimal for this conversion).
- **Step 2** − Combine all the resulting binary groups (of 3 digits each) into a single binary number.

Example

Octal Number − 25_8

Calculating Binary Equivalent −

Step	Octal Number	Binary Number

Step 1	25_8	$2_{10}\ 5_{10}$
Step 2	25_8	$010_2\ 101_2$
Step 3	25_8	010101_2

Octal Number − 25_8 = Binary Number − 10101_2

Shortcut method - Binary to Hexadecimal

Steps

- **Step 1** − Divide the binary digits into groups of four (starting from the right).
- **Step 2** − Convert each group of four binary digits to one hexadecimal symbol.

Example

Binary Number − 10101_2

Calculating hexadecimal Equivalent −

Step	Binary Number	Hexadecimal Number
Step 1	10101_2	0001 0101
Step 2	10101_2	$1_{10}\ 5_{10}$
Step 3	10101_2	15_{16}

Binary Number − 10101_2 = Hexadecimal Number − 15_{16}

Shortcut method - Hexadecimal to Binary

Steps

- **Step 1** − Convert each hexadecimal digit to a 4 digit binary number (the hexadecimal digits may be treated as decimal for this conversion).
- **Step 2** − Combine all the resulting binary groups (of 4 digits each) into a single binary number.

Example

Hexadecimal Number − 15_{16}

Calculating Binary Equivalent −

Step	Hexadecimal Number	Binary Number
Step 1	15_{16}	$1_{10}\ 5_{10}$
Step 2	15_{16}	$0001_2\ 0101_2$
Step 3	15_{16}	00010101_2

Hexadecimal Number − 15_{16} = Binary Number − 10101_2

Let's check the take away from this lecture

1) Convert $(0.345)_{10}$ into an octal number:

a) $(0.16050)_8$

b) $(0.26050)_8$

c) $(0.19450)_8$

d) $(0.24040)_8$

2) The representation of octal number $(532.2)_8$ in decimal is _____

a) $(346.25)_{10}$

b) $(532.864)_{10}$

c) $(340.67)_{10}$

d) $(531.668)_{10}$

3) The largest two digit hexadecimal number is _____

a) $(FE)_{16}$

b) $(FD)_{16}$

c) $(FF)_{16}$

d) $(EF)_{16}$

Exercise

Q.1 Discuss the characteristics of a octal number system.

Q.2 Convert $(152.25)_8$ to its decimal equivqlent.

Q.3 Convert $(152.25)_{10}$ to its hexadecimal equivalent.

Learning from this lecture: Learners will be able to understand octal and hexadecimal number systems.

Lecture 3

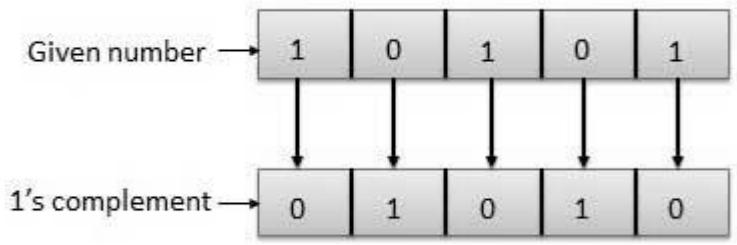
Binary Arithmetic

Binary system complements

As the binary system has base r = 2. So the two types of complements for the binary system are 2's complement and 1's complement.

1's complement

The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement. Example of 1's Complement is as follows.

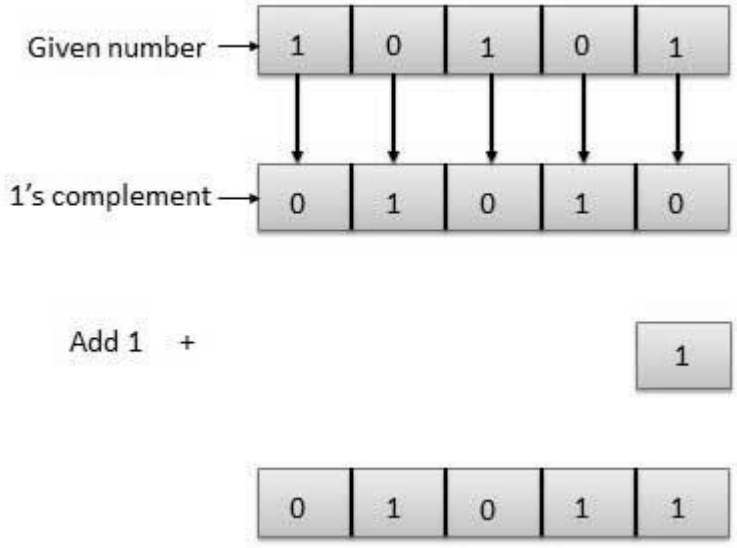


2's complement

The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number.

2's complement = 1's complement + 1

Example of 2's Complement is as follows.



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

Binary Addition

It is a key for binary subtraction, multiplication, division. There are four rules of binary addition.

Case	A + B	Sum	Carry
1	0 + 0	0	0
2	0 + 1	1	0
3	1 + 0	1	0
4	1 + 1	0	1

In fourth case, a binary addition is creating a sum of (1 + 1 = 10) i.e. 0 is written in the given column and a carry of 1 over to the next column.

Example − Addition

0011010 + 001100 = 00100110

```
    1 1          carry
  0 0 1 1 0 1 0  = 26(10)
+ 0 0 0 1 1 0 0  = 12(10)
-----------------
  0 1 0 0 1 1 0  = 38(10)
```

Binary Subtraction

Subtraction and Borrow, these two words will be used very frequently for the binary subtraction. There are four rules of binary subtraction.

Case	A - B	Subtract	Borrow
1	0 - 0	0	0
2	1 - 0	1	0
3	1 - 1	0	0
4	0 - 1	0	1

Example − Subtraction

0011010 - 001100 = 00001110

```
        1 1      borrow
  0 0 1 1 0 1 0  = 26(10)
- 0 0 0 1 1 0 0  = 12(10)
-----------------
  0 0 0 1 1 1 0  = 14(10)
```

Binary Multiplication

Binary multiplication is similar to decimal multiplication. It is simpler than decimal multiplication because only 0s and 1s are involved. There are four rules of binary multiplication.

Case	A x B	Multiplication
1	0 x 0	0
2	0 x 1	0
3	1 x 0	0
4	1 x 1	1

Example − Multiplication

Example:

0011010 x 001100 = 100111000

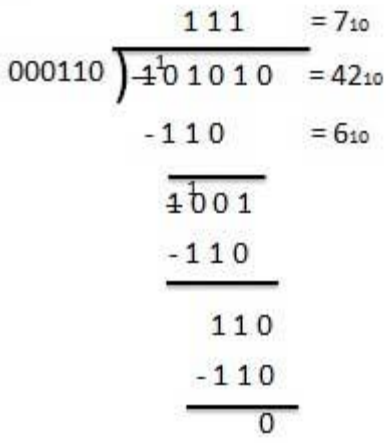
```
      0 0 1 1 0 1 0  = 26(10)
    x 0 0 0 1 1 0 0  = 12(10)
---------------------
      0 0 0 0 0 0 0
    0 0 0 0 0 0 0
  0 0 1 1 0 1 0
0 0 1 1 0 1 0
---------------------
1 0 0 1 1 1 0 0 0  = 312(10)
```

Binary Division

Binary division is similar to decimal division. It is called as the long division procedure.

Example − Division

101010 / 000110 = 000111



Let's check the take away from this lecture

1) Perform binary addition: 101101 + 011011 = ?

a) 011010

b) 1010100

c) 101110

d) 1001000

2) Perform binary subtraction: 101111 – 010101 = ?

a) 100100

b) 010101

c) 011010

d) 011001

3) 100101 × 0110 = ?

a) 1011001111

b) 0100110011

c) 101111110

d) 0110100101

Exercise

Q.1 Write down the rules for binary addition.

Q.2 Write down the rules for binary subtraction.

Q.3 Perform the following using 2's complement subtraction: i) 15-9 ii) 27-17

Q.4 Divide the binary numbers: 111101 ÷ 1001 and find the remainder.

Learning from this lecture: Learners will be able to understand binary arithmetic operations such as addition, subtraction, multiplication and division.

Lecture 4

Octal Arithmetic

Octal Addition

Following octal addition table will help you to handle octal addition.

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

A

Sum

B

To use this table, simply follow the directions used in this example: Add 6_8 and 5_8. Locate 6 in the A column then locate the 5 in the B column. The point in 'sum' area where these two columns intersect is the 'sum' of two numbers.

$6_8 + 5_8 = 13_8$.

Example − Addition

$456_8 + 123_8 = 601_8$

1 1	carry
4 5 6	$= 302_{10}$
+1 2 3	$= 83_{10}$
6 0 1	$= 385_{10}$

Octal Subtraction

The subtraction of octal numbers follows the same rules as the subtraction of numbers in any other number system. The only variation is in borrowed number. In the decimal system, you borrow a group of 10_{10}. In the binary system, you borrow a group of 2_{10}. In the octal system you borrow a group of 8_{10}.

Example − Subtraction

Example:

$456_8 - 173_8 = 333_8$

8	borrow
$^{3}4\ 5\ 6$	$= 302_{10}$
-1 7 3	$= 123_{10}$
2 6 3	$= 179_{10}$

Let's check the take away from this lecture

1) Octal subtraction of $(232)_8$ from $(417)_8$ will give _____

a) 165

b) 185

c) 815

d) 516

2) $(123)_8 + (7651)_8 = ?$

a) $(7771)_8$

b) $(7774)_8$

c) $(1154)_8$

d) $(6254)_8$

3) 7's complement subtraction of 342 from 614 will give _____

a) 352

b) -352

c) -252

d) 252

Exercise

Q.1 Write down the rules for octal addition.
Q.2 Write down the rules for octal subtraction.
Q.3 Find 7's complement of 402
Q.4 Find Subtraction of 342 and 614 using 8's complement method.

Learning from this lecture: Learners will be able to understand octal arithmetic operations such as addition, subtraction (7's and 8's complement method).

Lecture 5

Hexadecimal Arithmetic

Hexadecimal Addition

Following hexadecimal addition table will help you greatly to handle Hexadecimal 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

X

Sum

Y

To use this table, simply follow the directions used in this example − Add A_{16} and 5_{16}. Locate A in the X column then locate the 5 in the Y column. The point in 'sum' area where these two columns intersect is the sum of two numbers.

$A_{16} + 5_{16} = F_{16}$.

Example − Addition

$4A6_{16} + 1B3_{16} = 659_{16}$

1	carry
4 A 6	$= 1190_{10}$
+ 1 B 3	$= 435_{10}$
6 5 9	$= 1625_{10}$

Hexadecimal Subtraction

The subtraction of hexadecimal numbers follow the same rules as the subtraction of numbers in any other number system. The only variation is in borrowed number. In the decimal system, you borrow a group of 10_{10}. In the binary system, you borrow a group of 2_{10}. In the hexadecimal system you borrow a group of 16_{10}.

Example - Subtraction

$4A6_{16} - 1B3_{16} = 2F3_{16}$

16	borrow
$^{3}4$ A 6	$= 1190_{10}$
- 1 B 3	$= 435_{10}$
2 F 3	$= 755_{10}$

Let's check the take away from this lecture

1) Evaluate $(BA3)_{16} + (5DE)_{16}$

a) 1181

b) 1254

c) 1555

d) 1165

2) $(AF.C1)_{16} + (78.989)_{16} = ?$

a) $(126.599)_{16}$

b) $(128.599)_{16}$

c) $(138.599)_{16}$

d) $(136.588)_{16}$

3) Find 15's complement of 1B06

a) F4E8

b) E4F8

c) E4F9

d) F4E9

Exercise
Q.1 Write down the rules for hexadecimal addition.
Q.2 Write down the rules for hexadecimal subtraction.
Q.3 Find Subtraction of B06 and C7C using 15's complement method.
Q.4 Find Subtraction of B06 and C7C using 16's complement method.

Learning from this lecture: Learners will be able to understand hexadecimal arithmetic operations such as addition, subtraction (15's and 16's complement method).

Lecture 6

1.2 Codes

In the coding, when numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called as a code. The digital data is represented, stored and transmitted as group of binary bits. This group is also called as **binary code**. The binary code is represented by the number as well as alphanumeric letter.

Advantages of Binary Code

Following is the list of advantages that binary code offers.

- Binary codes are suitable for the computer applications.
- Binary codes are suitable for the digital communications.
- Binary codes make the analysis and designing of digital circuits if we use the binary codes.
- Since only 0 & 1 are being used, implementation becomes easy.

Classification of binary codes

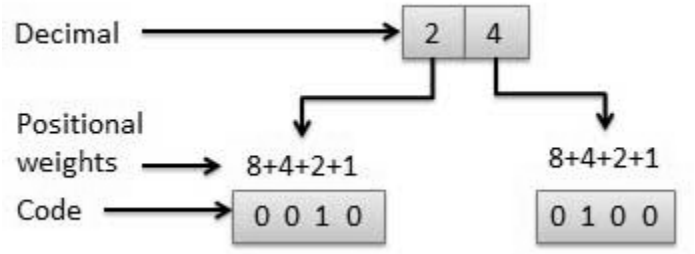
The codes are broadly categorized into following four categories.

- Weighted Codes
- Non-Weighted Codes
- Binary Coded Decimal Code
- Alphanumeric Codes

- Error Detecting Codes
- Error Correcting Codes

Weighted Codes

Weighted binary codes are those binary codes which obey the positional weight principle. Each position of the number represents a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9. In these codes each decimal digit is represented by a group of four bits.



Non-Weighted Codes

In this type of binary codes, the positional weights are not assigned. The examples of non-weighted codes are Excess-3 code and Gray code.

Gray Code

It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position. It has a very special feature that, only one bit will change each time the decimal number is incremented as shown in fig. As only one bit changes at a time, the gray code is called as a unit distance code. The gray code is a cyclic code. Gray code cannot be used for arithmetic operation.

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

Application of Gray code

- Gray code is popularly used in the shaft position encoders.

- A shaft position encoder produces a code word which represents the angular position of the shaft.

Binary Coded Decimal (BCD) code

In this code each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. In the BCD, with four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Advantages of BCD Codes

- It is very similar to decimal system.
- We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes

- The addition and subtraction of BCD have different rules.
- The BCD arithmetic is little more complicated.
- BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

Let's check the take away from this lecture

1) Binary coded decimal is a combination of _____

a) Two binary digits

b) Three binary digits

c) Four binary digits

d) Five binary digits

2) Code is a symbolic representation of _____ information.

a) Continuous

b) Discrete

c) Analog

d) Both continuous and discrete

3) Convert the binary number 1100 to Gray code

a) 0011

b) 1010

c) 1100

d) 1001

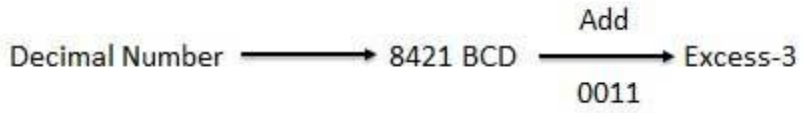
Exercise
Q.1 Explain the features of Gray code.
Q.2 Convert the Gray code 1011 to binary.
Q.3 State the difference between binary coding and binary-coded decimal.

Learning from this lecture: Learners will be able to understand Gray code and BCD code.

Lecture 7

Excess-3 Code

The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimal numbers. The Excess-3 code words are derived from the 8421 BCD code words adding $(0011)_2$ or (3)10 to each code word in 8421. The excess-3 codes are obtained as follows –



Example

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Alphanumeric codes

A binary digit or bit can represent only two symbols as it has only two states '0' or '1'. But this is not enough for communication between two computers because there we need many more symbols for communication. These symbols are required to represent 26 alphabets with capital and small letters, numbers from 0 to 9, punctuation marks and other symbols.

The alphanumeric codes are the codes that represent numbers and alphabetic characters. Mostly such codes also represent other characters such as symbol and various instructions necessary for conveying information. An alphanumeric code should at least represent 10 digits and 26 letters of alphabet i.e. total 36 items. The following three alphanumeric codes are very commonly used for the data representation.

- American Standard Code for Information Interchange (ASCII).
- Extended Binary Coded Decimal Interchange Code (EBCDIC).
- Five bit Baudot Code.

ASCII code is a 7-bit code whereas EBCDIC is an 8-bit code. ASCII code is more commonly used worldwide while EBCDIC is used primarily in large IBM computers.

There are many methods or techniques which can be used to convert code from one format to another. This is demonstrated below:

- Binary to BCD Conversion
- BCD to Binary Conversion
- BCD to Excess-3
- Excess-3 to BCD

Binary to BCD Conversion

Steps

- **Step 1** -- Convert the binary number to decimal.
- **Step 2** -- Convert decimal number to BCD.

Example − convert $(11101)_2$ to BCD.

Step 1 − Convert to Decimal

Binary Number − 11101_2

Calculating Decimal Equivalent −

Step	Binary Number	Decimal Number
Step 1	11101_2	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	11101_2	$(16 + 8 + 4 + 0 + 1)_{10}$

Step 3	11101_2	29_{10}

Binary Number − 11101_2 = Decimal Number − 29_{10}

Step 2 − Convert to BCD

Decimal Number − 29_{10}

Calculating BCD Equivalent. Convert each digit into groups of four binary digits equivalent.

Step	Decimal Number	Conversion
Step 1	29_{10}	0010_2 1001_2
Step 2	29_{10}	00101001_{BCD}

Result

$(11101)_2$= $(00101001)_{BCD}$

BCD to Binary Conversion

Steps

- **Step 1** -- Convert the BCD number to decimal.
- **Step 2** -- Convert decimal to binary.

Example − convert $(00101001)_{BCD}$ to Binary.

Step 1 - Convert to BCD

BCD Number − $(00101001)_{BCD}$

Calculating Decimal Equivalent. Convert each four digit into a group and get decimal equivalent for each group.

Step	BCD Number	Conversion
Step 1	$(00101001)_{BCD}$	0010_2 1001_2
Step 2	$(00101001)_{BCD}$	2_{10} 9_{10}
Step 3	$(00101001)_{BCD}$	29_{10}

BCD Number − $(00101001)_{BCD}$ = Decimal Number − 29_{10}

Step 2 - Convert to Binary

Used long division method for decimal to binary conversion.

Decimal Number − 29_{10}

Calculating Binary Equivalent –

Step	Operation	Result	Remainder
Step 1	29 / 2	14	1
Step 2	14 / 2	7	0
Step 3	7 / 2	3	1
Step 4	3 / 2	1	1
Step 5	1 / 2	0	1

As mentioned in Steps 2 and 4, the remainders have to be arranged in the reverse order so that the first remainder becomes the least significant digit (LSD) and the last remainder becomes the most significant digit (MSD).

Decimal Number − 29_{10} = Binary Number − 11101_2

Result

$(00101001)_{BCD} = (11101)_2$

BCD to Excess-3

Steps

- **Step 1** -- Convert BCD to decimal.
- **Step 2** -- Add $(3)_{10}$ to this decimal number.
- **Step 3** -- Convert into binary to get excess-3 code.

Example − convert $(0110)_{BCD}$ to Excess-3.

Step 1 − Convert to decimal

$(0110)_{BCD} = 6_{10}$

Step 2 − Add 3 to decimal

$(6)_{10} + (3)_{10} = (9)_{10}$

Step 3 − Convert to Excess-3

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

Result

$(0110)_{BCD} = (1001)_{XS-3}$

Excess-3 to BCD Conversion

Steps

- **Step 1** -- Subtract $(0011)_2$ from each 4 bit of excess-3 digit to obtain the corresponding BCD code.

Example − convert $(10011010)_{XS-3}$ to BCD.

Given XS-3 number = 1 0 0 1 1 0 1 0
Subtract $(0011)_2$ = 1 0 0 1 0 1 1 1

BCD = 0 1 1 0 0 1 1 1

Result

$(10011010)_{XS-3} = (01100111)_{BCD}$

Let's check the take away from this lecture

1) The excess-3 code for 597 is given by _____

a) 100011001010

b) 100010100111

c) 010110010111

d) 010110101101

2) The decimal equivalent of the excess-3 number 110010100011.01110101 is _____

a) 970.42

b) 1253.75

c) 861.75

d) 1132.87

3) ASCII Code is used for representing more than _____ characters

a) 16

b) 8

c) 64

d) 32

Exercise
Q.1 Add the two BCD numbers: 1001 + 0100.
Q.2 Find the Excess-3 code for the decimal number 31.
Q.3 Convert Excess-3 code 1001001 into BCD and decimal number.

Learning from this lecture: Learners will be able to understand Excess-3 and ASCII Codes.

Lecture 8

1.3 Error Detection and Correction:

We know that the bits 0 and 1 corresponding to two different range of analog voltages. So, during transmission of binary data from one system to the other, the noise may also be added. Due to this, there may be errors in the received data at other system.

That means a bit 0 may change to 1 or a bit 1 may change to 0. We can't avoid the interference of noise. But, we can get back the original data first by detecting whether any errors present and then correcting those errors. For this purpose, we can use the following codes.

- Error detection codes
- Error correction codes

Hamming Code

Hamming code is useful for both detection and correction of error present in the received data. This code uses multiple parity bits and we have to place these parity bits in the positions of powers of 2.

The **minimum value of 'k'** for which the following relation is correct validvalid is nothing but the required number of parity bits.

$$2^k \geq n+k+1$$

Where,

'n' is the number of bits in the binary code information

'k' is the number of parity bits

Therefore, the number of bits in the Hamming code is equal to n + k.

Let the **Hamming code** is $b_{n+k}b_{n+k-1}.....b_3b_2b_1$ & parity bits $p_k,p_{k-1},....p_1$. We can place the 'k' parity bits in powers of 2 positions only. In remaining bit positions, we can place the 'n' bits of binary code.

Based on requirement, we can use either even parity or odd parity while forming a Hamming code. But, the same parity technique should be used in order to find whether any error present in the received data.

Follow this procedure for finding **parity bits**.

- Find the value of $\mathbf{p_1}$, based on the number of ones present in bit positions b_3, b_5, b_7 and so on. All these bit positions suffixesin their equivalent binary have '1' in the place value of 2^0.
- Find the value of $\mathbf{p_2}$, based on the number of ones present in bit positions b_3, b_6, b_7 and so on. All these bit positions suffixesin their equivalent binary have '1' in the place value of 2^1.
- Find the value of $\mathbf{p_3}$, based on the number of ones present in bit positions b_5, b_6, b_7 and so on. All these bit positions suffixesin their equivalent binary have '1' in the place value of 2^2.
- Similarly, find other values of parity bits.

Follow this procedure for finding **check bits**.

- Find the value of c_1, based on the number of ones present in bit positions b_1, b_3, b_5, b_7 and so on. All these bit positions suffixes in their equivalent binary have '1' in the place value of 2^0.
- Find the value of c_2, based on the number of ones present in bit positions b_2, b_3, b_6, b_7 and so on. All these bit positions suffixes in their equivalent binary have '1' in the place value of 2^1.
- Find the value of c_3, based on the number of ones present in bit positions b_4, b_5, b_6, b_7 and so on. All these bit positions suffixes in their equivalent binary have '1' in the place value of 2^2.
- Similarly, find other values of check bits.

The decimal equivalent of the check bits in the received data gives the value of bit position, where the error is present. Just complement the value present in that bit position. Therefore, we will get the original binary code after removing parity bits.

Example 1

Let us find the Hamming code for binary code, $d_4d_3d_2d_1$ = 1000. Consider even parity bits.

The number of bits in the given binary code is n=4.

We can find the required number of parity bits by using the following mathematical relation.

$$2^k \geq n+k+1$$

Substitute, n=4 in the above mathematical relation.

$$\Rightarrow 2^k \geq 4+k+1$$

$$\Rightarrow 2^k \geq 5+k$$

The minimum value of k that satisfied the above relation is 3. Hence, we require 3 parity bits p_1, p_2, and p_3. Therefore, the number of bits in Hamming code will be 7, since there are 4 bits in binary code and 3 parity bits. We have to place the parity bits and bits of binary code in the Hamming code as shown below.

The **7-bit Hammingcode** is $b_7b_6b_5b_4b_3b_2b_1=d_4d_3d_2p_3d_1p_2bp_1$
By substituting the bits of binary code, the Hamming code will be $b_7b_6b_5b_4b_3b_2b_1=100p_3Op_2p_1$. Now, let us find the parity bits.

$$p_1=b_7\oplus b_5\oplus b_3=1\oplus 0\oplus 0=1$$

$$p_2=b_7\oplus b_6\oplus b_3=1\oplus 0\oplus 0=1$$

$$p_3=b_7\oplus b_6\oplus b_5=1\oplus 0\oplus 0=1$$

By substituting these parity bits, the **Hamming code** will be $b_7b_6b_5b_4b_3b_2b_1=1001011$.

Example 2

In the above example, we got the Hamming code as $b_7b_6b_5b_4b_3b_2b_1=1001011$. Now, let us find the error position when the code received is $b_7b_6b_5b_4b_3b_2b_1=1001111$.

Now, let us find the check bits.

$$c_1=b_7\oplus b_5\oplus b_3\oplus b_1=1\oplus 0\oplus 1\oplus 1=1$$

$$c_2=b_7\oplus b_6\oplus b_3\oplus b_2=1\oplus 0\oplus 1\oplus 1=1$$

$$c_3=b_7\oplus b_6\oplus b_5\oplus b_4=1\oplus 0\oplus 0\oplus 1=0$$

The decimal value of check bits gives the position of error in received Hamming code.

$$c_3c_2c_1=(011)_2=(3)_{10}$$

Therefore, the error present in third bit (b_3) of Hamming code. Just complement the value present in that bit and remove parity bits in order to get the original binary code.

Let's check the take away from this lecture

1) Why do we require hamming codes?

a) Error correction

b) Encryption only

c) Decryption

d) Bit stuffing

2) Hamming codes can be used for both single-bit error and burst error detection and correction.

a) True

b) False

3) Who invented Hamming codes?

a) Richard Hamming

b) Ross Hamming

c) Shannon

d) Huffman

Exercise
Q.1 What is Error Detection and Correction?
Q.2 List the applications of Hamming code
Q.3 Show that Hamming code actually achieves the theoretical limit for minimum number of check bits to do 1-bit error-correction**.**

Learning from this lecture: Learners will be able to understand Error detection and correction with the help of Hamming code.

Conclusion

The study of Number System and Codes explains the fundamentals for digital logic design and analysis. Since binary numbers are used in digital electronics, the understanding of number system is required. The transmission of data from one end to other end requires the understanding of Error Detection and Correction techniques and thereby the study of different Codes.

Short Answer Questions:

1. What is Number System?

Ans) Number systems are the technique to represent numbers in the computer system architecture, every value that you are saving or getting into/from computer memory has a defined number system.

2. What is binary number system?

Ans) A Binary number system has only two digits that are 0 and 1. Every number (value) represents with 0 and 1 in this number system. The base of binary number system is 2, because it has only two digits.

3. What is octal number system?

Ans) Octal number system has only eight (8) digits from 0 to 7. Every number (value) represents with 0,1,2,3,4,5,6 and 7 in this number system. The base of octal number system is 8, because it has only 8 digits.

4. What is hexadecimal number system?

Ans) A Hexadecimal number system has sixteen (16) alphanumeric values from 0 to 9 and A to F. Every number (value) represents with 0,1,2,3,4,5,6, 7,8,9,A,B,C,D,E and F in this number system. The base of hexadecimal number system is 16, because it has 16 alphanumeric values. Here A is 10, B is 11, C is 12, D is 13, E is 14 and F is 15.

5. What is decimal number system?

Ans) Decimal number system has only ten (10) digits from 0 to 9. Every number (value) represents with 0,1,2,3,4,5,6, 7,8 and 9 in this number system. The base of decimal number system is 10, because it has only 10 digits.

6. What is Gray Code?

Ans) Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit). It is not weighted that means it does not depends on positional value of digit. This cyclic variable code that means every transition from one value to the next value involves only one bit change.

7. What is BCD Code?

Ans) BCD is simply the 4-bit binary code representation of a decimal digit with each decimal digit replaced in the integer and fractional parts with its binary equivalent. BCD Code uses four bits to represent the 10 decimal digits of 0 to 9.

8. What is Excess-3 Code?

Ans) 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.

9. What is Error Detection and Correction?

Ans) Error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data in many cases.

10. What is Hamming Code?

Ans) Hamming code is a set of error-correction code s that can be used to detect and correct bit errors that can occur when computer data is moved or stored. Like other error-correction code, Hamming code makes use of the concept of parity and parity bits, which are bits that are added to data so that the validity of the data can be checked when it is read or after it has been received in a data transmission. Using more than one parity bit, an error-correction code can not only identify a single bit error in the data unit, but also its location in the data unit.

Long Answer Questions:

1. Evaluate the following using 2's complement method:

(i) 110110 - 10110

Solution:

The numbers of bits in the subtrahend is 5 while that of minuend is 6. We make the number of bits in the subtrahend equal to that of minuend by taking a `0' in the sixth place of the subtrahend.

Now, 2's complement of 010110 is (101101 + 1) i.e.101010. Adding this with the minuend.

1 1 0 1 1 0 Minuend

1 0 1 0 1 0 2's complement of subtrahend

Carry over 1 1 0 0 0 0 0 Result of addition

After dropping the carry over we get the result of subtraction to be 100000.

(ii) 10110 – 11010

Solution:

2's complement of 11010 is (00101 + 1) i.e. 00110. Hence

Minued - 1 0 1 1 0

2's complement of subtrahend - 0 0 1 1 0

Result of addition - 1 1 1 0 0

As there is no carry over, the result of subtraction is negative and is obtained by writing the 2's complement of 11100 i.e.(00011 + 1) or 00100.
Hence the difference is – 100.

(iii) 1010.11 – 1001.01

Solution:

2's complement of 1001.01 is 0110.11. Hence

Minued - 1 0 1 0 . 1 1

2's complement of subtrahend - 0 1 1 0 . 1 1

Carry over 1 0 0 0 1 . 1 0

After dropping the carry over we get the result of subtraction as 1.10.

(iv) 10100.01 – 11011.10

Solution:

2's complement of 11011.10 is 00100.10. Hence

Minued - 1 0 1 0 0 . 0 1

2's complement of subtrahend - 0 1 1 0 0 . 1 0

Result of addition - 1 1 0 0 0 . 1 1

As there is no carry over the result of subtraction is negative and is obtained by writing the 2's complement of 11000.11.
Hence the required result is – 00111.01.

2. Compute the following for Octal numbers. (Refer the section Octal Arithmetic in 1.1)

i) 130-110 ii) 130+102 iii) 130.5 + 125.5 iv) 150.5-125.5

3. Compute the following for Hexadecimal numbers. (Refer the section Hexadecimal Arithmetic in 1.1)

i) 15646-C89A ii) 9DAD+6E5E iii) B881+FFBD iv) C326-56C2

4. Explain Hamming Code with example. (Refer the section Hamming Code in 1.3)

5. Convert $(00101001)_{BCD}$ to binary by explaining the steps. (Refer 1.2)

6. Calculate the following:

(i) Convert $(1056)_{16}$ to (?)$_8$

(ii) Convert $(11672)_8$ to (?)$_{16}$

(iii) Convert $(2724)_8$ to (?)$_5$ (Refer the sections decimal, octal and hexadecimal number system in 1.1)

Set of Questions for FA/IA/ESE

Q. 1) Problems based on conversion of bases.

Q. 2) Problems on Binary addition and subtraction (1's and 2's complement method).

Q. 3) Problems on Octal addition and subtraction (7's and 8's complement method).

Q.4) Problems on Hexadecimal addition and subtraction (15's and 16's complement method

Q. 5) Problems on BCD addition and subtraction (9's complement and 10's complement).

Q. 6) Convert a given binary number to Gray Code and Vice versa.

Q. 7) Convert a given binary number to Excess-3 Code and Vice versa.

Q. 8) Explain Hamming code with example.

References:

1) Modern Digital Electronics By R. P. Jain.
2) Digital Logic and Computer Design By M. Morris Mano.
3) Digital Principles and Applications By Donald p Leach, Albert Paul Malvino

Practice for Module-01

Q.1) Convert the following numbers from base 10 to base 16- (10 marks)

1. $(2020)_{10}$
2. $(2020.65625)_{10}$
3. $(172)_{10}$
4. $(172.983)_{10}$

Q.2) Convert the following numbers from base 10 to base 8- (10 marks)

1. $(1032)_{10}$
2. $(1032.6875)_{10}$
3. $(172)_{10}$

4. $(172.878)_{10}$

Q.3) Convert the following numbers from base 10 to base 2- (10 marks)

1. $(18)_{10}$
2. $(18.625)_{10}$
3. $(172)_{10}$
4. $(172.878)_{10}$

Q.4) Convert the following numbers to base 10- (10 marks)

1. $(10010)_{2}$
2. $(254)_{8}$
3. $(AC)_{16}$
4. $(10010.101)_{2}$
5. $(254.7014)_{8}$
6. $(AC.FBA5)_{16}$
7. $(0.1402)_{8}$
8. $(0.ABDF)_{16}$

Q. 5) a) Perform binary addition for 11011 & 10101. (5 marks)

b) Compute 10001-11101 using 2's complement binary subtraction. (5 marks)

Q.6) a) Perform BCD Addition of 8765 and 3943. (5 marks)

b) Compute 325-641using 10's complement subtraction. (5 marks)

Q.7) Calculate hamming code for data 1011 (5 marks).

Q.8) Detect and correct error for a received 7-bit Hamming code: 1011011. (10 marks)

Self-assessment

Q.1) What is Number System? Define different number systems like binary, octal and hexadecimal.

Q. 2) What is the role of different Codes in Digital Logic Design.

Q. 3) What is Error Detection and Correction? Explain how Hamming code can be used in error detection.

Self-evaluation

Name of Student		
Class		
Roll No.		
Subject		
Module No.		
S.No		Tick Your choice
1.	Do you understand the different number systems?	○ Yes ○ No
2.	Do you understand the conversion of bases?	○ Yes ○ No
3.	Do you understand the binary, octal and hexadecimal arithmetic operations such as addition, subtraction etc.?	○ Yes ○ No
4.	Do you understand the features of different codes like Gray code, BCD code, Excess-3 code ?	○ Yes ○ No
5.	Do you understand how Hamming code can be generated from the given data ?	○ Yes ○ No
6.	Do you understand module 1 ?	○ Yes, Completely. ○ Partialy. ○ No, Not at all.