# GUDLAVALLERU ENGINEERING COLLEGE

**(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)**

**Seshadri Rao Knowledge Village, Gudlavalleru – 521 356.**

## Department of Computer Science and Engineering

# HANDOUT

## on

# DIGITAL LOGIC DESIGN

## Vision

To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society

## Mission

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behavior & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

## Program Educational Objectives

- Identify, analyze, formulate and solve Computer Science and Engineering problems both independently and in a team environment by using the appropriate modern tools.
- Manage software projects with significant technical, legal, ethical, social, environmental and economic considerations
- Demonstrate commitment and progress in lifelong learning, professional development, leadership and Communicate effectively with professional clients and the public.

# HANDOUT ON DIGITAL LOGIC DESIGN

Class & Sem. :II B.Tech – I Semester Year: 2018-19

Branch : CSE Credits: 3

==========

## 1. Brief History and Scope of the Subject

In 1947, Bardeen, Braittain and Shockley invented the transistor at Bell Labs. This resulted in solid state switching, that is much faster and more reliable than relays. This enabled the creation of powerful computers. In 1958, Jack Kilby and Robert Noyce invented integrated circuits that enabled more and less expensive digital circuits in a smaller package. These were used in the space program in which weight is an important factor. In 1969, Dick Morley invented the first Programmable Logic Controller (PLC), the MODICOM Model 84. The PLC is designed for the more rugged applications and more power that is required for manufacturing. These devices have replaced by control relays in many manufacturing areas. In 1971, Robert Noyce and Gordon Moore introduced a “Computer on a chip”. It executed 60,000 operations per second which is substantially more than the 5 operations per second for Shannon’s relay logic device. Improvements in integrated circuits and microprocessors have enhanced the functionality of Programmable Logic Controllers. In mid 1970’s through 1980’s, Allen Bradley produced PLC1 through PLC5 series of Programmable Logic Controllers using integrated circuits and microprocessors. In 1980, IBM started production of the IBM Personal Computer (PC) which made computing available to all. The PC is useful for both programming Programmable Logic Controllers and for the analysis and design of digital logic circuits. In addition to computers and the PLC, digital circuits are used in cell phones and other mobile devices, automobiles, medical devices, security systems, household appliances, energy management systems and High Definition Television (HDTV).

### *Recent developments*

In 2009, researchers discovered that memristors can implement a boolean state storage similar to a flip flop, implication and logical inversion, providing a complete logic family with very small amounts of space and power, using familiar CMOS semiconductor processes.The discovery of superconductivity has enabled the

development of rapid single flux quantum (RSFQ) circuit technology, which uses Josephson junctions instead of transistors. Most recently, attempts are being made to construct purely optical computing systems capable of processing digital information using nonlinear optical elements.

## 2. Pre-Requisites

- Programming Language such as C/C++

## 3. Course Objectives:

- To familiarize with the design of digital logic circuits.

## 4. Course Outcomes:

CO1: Translate nymber given in one number system to another number system.
CO2: Apply complements to perform addition and subtraction of signed numbers.
CO3: reduce Boolean function using Boolean laws, theorems and K-Maps.
CO4: design combinational logic circuits such as adders, subtractors, decoders, encoders, Mux and De-Mux.
CO5: prepare characteristic equation and excitation tables of SR, JK, T and D flip-flops.
CO6: design counters and registers using flip-flops.

## 5. Program Outcomes:

Graduates of the Computer Science and Engineering Program will have an ability to

a. apply knowledge of computing, mathematics, science and engineering fundamentals to solve complex engineering problems.
b. formulate and analyze a problem, and define the computing requirements appropriate to its solution using basic principles of mathematics, science and computer engineering.
c. design, implement, and evaluate a computer based system, process, component, or software to meet the desired needs.
d. design and conduct experiments, perform analysis and interpretation of data and provide valid conclusions.
e. use current techniques, skills, and tools necessary for computing practice.
f. understand legal, health, security and social issues in Professional Engineering practice.

g. understand the impact of professional engineering solutions on environmental context and the need for sustainable development.
h. understand the professional and ethical responsibilities of an engineer.
i. function effectively as an individual, and as a team member/ leader in accomplishing a common goal.
j. communicate effectively, make effective presentations and write and comprehend technical reports and publications.
k. learn and adopt new technologies, and use them effectively towards continued professional development throughout the life.
l. understand engineering and management principles and their application to manage projects in the software industry.

## 6. Mapping of Course Outcomes with Program Outcomes:

| | a | b | c | d | e | f | g | h | i | j | k | l |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO1 | 2 | | | | | | | | | | | |
| CO2 | | 2 | 2 | | | | | | | | | |
| CO3 | | 2 | 2 | 2 | 2 | | | | | | 2 | |
| CO4 | | 2 | 2 | 2 | 2 | | | | | | 2 | |
| CO5 | | 2 | 2 | 2 | 2 | | | | | | 2 | |

## 7. Prescribed Text Books

a. M. Morris Mano, Michael D Ciletti, Digital Design, PEA, 5th edition.

## 8. Reference Text Books

a. Kohavi, Jha, Switching and Finite Automata Theory, Cambridge, 3rd edition.
b. Leach, Malvino, Saha, Digital Logic Design, TMH.
c. Roth, Fundamentals of Logic Design, Cengage, 5th edition.

## 9. URLs and Other E-Learning Resources

**URLs:**

IEEE Xplore: IEE proceedings -Computers and Digital Techniques: http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=4641395
IET Digital Library: http://digital-library.theiet.org/content/journals/iet-cdt

IEEE Xplore: IEEE Design & Test of Computers
http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=54

**E-Learning Materials:**

Journals:

INTERNATIONAL JOURNALS:

- IEEE trans on electronic devices.
- IEEE journal of solid state circuits

NATIONAL JOURNALS:

- ELECTRONICS today
- IETE technical review

## 10. Digital Learning Materials:

a. SONET CDs - Switching Theory and Logic design -34
b. IIT CDs - Principles and Design of Digital Systems 28, Madras.

## 11. Lecture Schedule / Lesson Plan

| Topic | No. of Periods | |
|---|---|---|
| | Theory | Tutorial |
| **UNIT –1: Number Systems** | | |
| Binary,octal,decimal,&hexadecimalnumbersystems | 1 | 1 |
| Number base conversions | 2 | |
| Problems | 2 | |
| R's and (r-1)'s COMPLEMENTS | 1 | 1 |
| Sutraction of unsigned binary numbers | 2 | |
| Signed binary numbers | 1 | |
| Weighted and non weighted codes | 2 | |
| **UNIT – 2: Logic gates and Boolean Algebra** | | |
| NOT, AND,OR, universal gates,Ex-Or and Ex-Nor gates | 2 | 1 |
| Boolean theorems, complement and dual | 2 | |
| SOP,POS, two level realization of logic functions using universal gates | 2 | 1 |
| Minimization of logic functions(pos,sop) using boolean theorems | 2 | |
| K-map(upto 4-variables), don't care conditions | 2 | |
| **UNIT – 3: Combinational Logic Circuits-1** | | |
| Design of half-adder, full adder | 2 | 1 |
| Half Subtractor, full subtractor | 2 | |
| Ripple adders and subtractors using 1's and 2's complement method | 2 | 1 |

| | | |
|---|---|---|
| **UNIT – 4: Combinational Logic Circuits-2** | | |
| Design of decoders, encoders | 2 | 1 |
| Priority Encoder | 1 | |
| Multiplexers and De-multiplexers | 2 | |
| Higher order Decoders | 2 | 1 |
| Higher order multiplexers and De-multiplexers | 2 | |
| Realization of Boolean functions using decoders, Multiplexers | 2 | |
| **UNIT – 5: Sequential Logic Circuits** | | |
| Classification of sequential circuits, latch and flip-flop | 2 | 1 |
| RS-latch using NAND and NOR gates, truth tables | 2 | |
| RS, JK, T & D flip-flops, truth and excitation tables | 2 | |
| Conversion of flip flops | 2 | 1 |
| Flip-flops with asynchronous inputs(Preset and clear) | 2 | |
| **UNIT – 6: Registers and counters** | | |
| Design of Registers, Bi-directional shift registers | 2 | 1 |
| Universal Shift registers | 2 | |
| Design of ripple counters, synchronous counters and variable Modulus counters | 2 | 1 |
| Ring counter and Johnson counter | 2 | |
| **Total No.of Periods:** | **56** | **12** |

## UNIT – I

**Objective:**

- To familiarize with the concepts and techniques associated with number systems and codes.

**Syllabus:**

Number Systems: Binary, Octal, Decimal, Hexadecimal Number Systems. Conversion of Numbers from One Radix to another Radix, r's and (r-1)'s Complement, Subtraction of Unsigned Numbers, Signed Binary Numbers, Weighted and Non weighted codes.

**Learning Outcomes:**

Students will be able to

- To convert a number in one radix to another radix.
- To determine r's and (r-1)'s complement of a number.
- To perform addition and subtraction of signed and unsigned numbers.
- To code the numbers in weighted code and non weighted code.

# Learning Material

## UNIT - I
## NUMBER SYSTEMS

**Definition**: A **number system** (or **system** of **numeration**) is a writing **system** for expressing **numbers**; that is, a mathematical notation for representing **numbers** of a given set, using digits or other symbols in a consistent manner.

**BASE or Radix:** A base of a number system or radix defines the range of values that a digit may have.

**Different types of number systems are:**

- Binary Number System
- Octal Number System
- Decimal Number System
- Hexa Decimal Number System

## BINARY (BASE 2) NUMBER SYSTEM

- Digital and computer technology is based on the binary number system, since the foundation is based on a transistor, which only has two states: **ON** or **OFF**.
- Each bit must be less than 2 which mean it has to be either 0 or 1.
- Each bit is weighted based on its position in the sequence (powers of 2) from the Least Significant Bit (LSB) to the Most Significant Bit (MSB).
- Weighing factors (or weights) are in powers of 2:

  **… $2^3$ $2^2$ $2^1$ $2^0$ . $2^{-1}$ $2^{-2}$ $2^{-3}$ …**
- Example

  **$1101.101_2 = 1\times2^3 + 1\times2^2 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-3}$**

## OCTAL (BASE 8) NUMBER SYSTEM

- This number system is used by humans as a representation of long strings of bits since they are:
  Easier to read and write, for example $(347)_8$ is easier to read and write than $(011100111)_2$.
- Octal digits: **0, 1, 2, 3, 4, 5, 6, 7**.
- Weighing factors (or weights) are in powers of 8:

  **… $8^3$ $8^2$ $8^1$ $8^0$ . $8^{-1}$ $8^{-2}$ $8^{-3}$ …**
- Example

  $56.2_8 = (5 \times 8^1) + (6 \times 8^0) + (2 \times 8^{-1})$

## DECIMAL (BASE 10) SYSTEM

- Base or radix is 10 (the base or radix of a number system is the total number of symbols/digits allowed in the system).
- Symbols/digits = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
- Each digit is weighted based on its position in the sequence (power of 10) from the Least Significant Digit (LSD, power of 0) to the Most Significant Digit (MSD, highest power).
- Weighing factors (or weights) are in powers of 10:

  **… $10^3$ $10^2$ $10^1$ $10^0$ . $10^{-1}$ $10^{-2}$ $10^{-3}$ …**
- Example, the 9 in the two numbers below has different values:

  $(75\underline{\mathbf{9}}4)_{10} = (7 \times 10^3) + (5 \times 10^2) + (9 \times 10^1) + (4 \times 10^0)$

  $(\underline{\mathbf{9}}12)_{10} = (9 \times 10^2) + (1 \times 10^1) + (2 \times 10^0)$

## HEXADECIMAL (BASE 16) NUMBER SYSTEM

- Hexadecimal has sixteen values 0 to 15, but to keep all these values in a single column, the 16 values (0 to 15) are written as 0 to F, using the letters A to F to represent numbers 10 to 15.
- Hexadecimal digits: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**.
- Weighing factors (or weights) are in powers of 8:

  **… $16^3$ $16^2$ $16^1$ $16^0$ . $16^{-1}$ $16^{-2}$ $16^{-3}$ …**
- Example

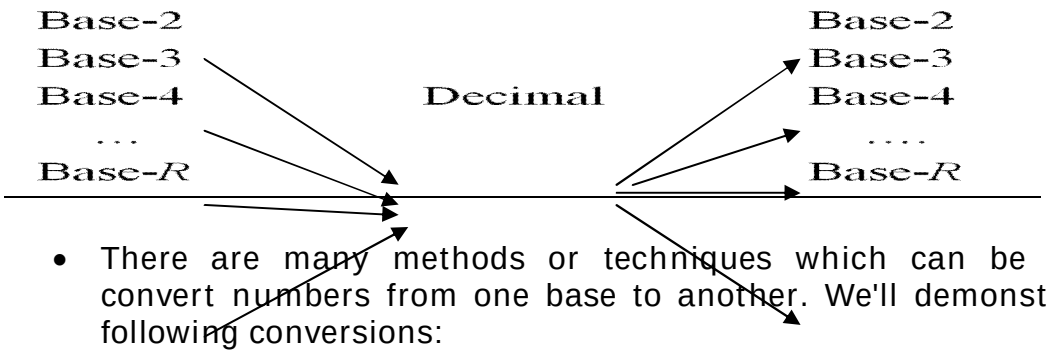
  $98.2_{16} = (9 \times 16^1) + (8 \times 16^0) + (2 \times 16^{-1})$

## NUMBER SYSTEMS CONVERSION TABLE

| Decimal Base-10 | Binary Base-2 | Octal Base-8 | Hexadecimal Base-16 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 |
| 3 | 11 | 3 | 3 |
| 4 | 100 | 4 | 4 |
| 5 | 101 | 5 | 5 |
| 6 | 110 | 6 | 6 |
| 7 | 111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |
| 16 | 10000 | 20 | 10 |
| 17 | 10001 | 21 | 11 |
| 18 | 10010 | 22 | 12 |
| 19 | 10011 | 23 | 13 |
| 20 | 10100 | 24 | 14 |
| 21 | 10101 | 25 | 15 |
| 22 | 10110 | 26 | 16 |
| 23 | 10111 | 27 | 17 |
| 24 | 11000 | 30 | 18 |
| 25 | 11001 | 31 | 19 |
| 26 | 11010 | 32 | 1A |
| 27 | 11011 | 33 | 1B |

| 28 | 11100 | 34 | 1C |
|---|---|---|---|
| 29 | 11101 | 35 | 1D |
| 30 | 11110 | 36 | 1E |
| 31 | 11111 | 37 | 1F |
| 32 | 100000 | 40 | 20 |

## CONVERSION OF NUMBERS FROM ONE RADIX TO ANOTHER RADIX




- There are many methods or techniques which can be used to convert numbers from one base to another. We'll demonstrate the following conversions:
  - A. Decimal to Any base (non- decimal)
  - B. Any base (non-decimal) to Decimal
  - C. Non-Decimal to Non-Decimal
  - D. Shortcut method − Binary to Octal
  - E. Shortcut method − Octal to Binary
  - F. Shortcut method − Binary to Hexadecimal
  - G. Shortcut method − Hexadecimal to Binary

### *A. CONVERSION FROM DECIMAL TO NON-DECIMAL*

- Repeated Division-by-base Method (for integer numbers)
- Repeated Multiplication-by-base Method (for fractions)

Example: To convert a **integer number** to binary, use successive division by 2 until the quotient is 0. The remainders form the answer, with the first remainder as the *least significant bit (LSB)* and the last as the *most significant bit (MSB)*.

| | | |
|---|---|---|
| 2 | 43 | |
| 2 | 21 rem 1 | ← LSB |
| 2 | 10 rem 1 | |
| 2 | 5 rem 0 | |
| 2 | 2 rem 1 | |
| 2 | 1 rem 0 | |
| | 0 rem 1 | ← MSB |

$(43)_{10} = (101011)_2$

- To convert decimal **fractions** to binary, repeated multiplication by 2 is used, until the fractional product is 0 (or until the desired number of decimal places). The carried digits, or *carries*, produce the answer, with the first carry as the MSB, and the last as the LSB.

| | Carry | |
|---|---|---|
| $0.3125\times2=0.625$ | 0 | ←MSB |
| $0.625\times2=1.25$ | 1 | |
| $0.25\times2=0.50$ | 0 | |
| $0.5\times2=1.00$ | 1 | ←LSB |

$(0.3125)_{10} = (0.0101)_2$

**Steps**

- **Step 1** – Divide the decimal number to be converted by the value of the new base.
- **Step 2** – Get the remainder from Step1 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, get the remainders from right to left, until the quotient becomes zero in Step 3.

### i. CONVERSION FROM DECIMAL to BINARY

Decimal Number: $29_{10}$

| 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 – $11101_2$.

## ii. CONVERSION FROM DECIMAL to OCTAL

Decimal Number: 4321

| Step | Operation | Result | Remainder |
|---|---|---|---|
| Step 1 | 4321/8 | 540 | 1 |
| Step 2 | 540/8 | 67 | 4 |
| Step 3 | 67 / 8 | 8 | 3 |
| Step 4 | 8 / 8 | 1 | 0 |
| Step 5 | 1 / 8 | 0 | 1 |

Decimal Number: 4321 = Octal Number: 10341

## iii. CONVERSION FROM DECIMAL to HEXA DECIMAL

| Step | Operation | Result | Remainder |
|---|---|---|---|
| Step 1 | 1234/16 | 77 | 2 |
| Step 2 | 77/16 | 4 | 13 = D |
| Step 3 | 4/16 | 0 | 4 |

Decimal Number: 1234 = Hexa Decimal Number: 4D2

What is the hexadecimal equivalent of the decimal number 3315.3?

```
Quotient      Remainder
------ ----------------------------
3315/16 =   207       3 ------+
 207/16  =    12      15 ----+ |
12/16    =     0      12 --+  | |  (Stop when the quotient is 0)
                           |  | |
                          C F 3  (HEX; Base 16)
                           (HEX; Base 16)

 Product            Integer Part   0.4 C C C ...
-------------------------------        | | | |
0.3*16   =   4.8       4            ----+ | | | |
 0.8*16   =  12.8      12            ------+ | | |
0.8*16   =  12.8       12            --------+ | |
0.8*16   =  12.8       12            ----------+ |
       :                                        ---------------------+
       :    Thus, 3315.3 (DEC) --> CF3.4CCC... (HEX)
```

### *B.CONVERSION FROM ANY BASE (NON-DECIMAL) TO DECIMAL*

**Step 1** – Determine the column (positional) number of each digit from right to left, beginning with column 0 (this depends on the position of the digit and the base of the number system).

**Step 2** – Multiply the digits with "base power column number" in the corresponding columns.

**Step 3** – Sum the products calculated in Step 2. The total is the equivalent value in decimal.

#### i. CONVERSION FROM BINARY TO DECIMAL

Calculating Decimal Equivalent of Binary Number – $11101_2$

| 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 – $29_{10}$

#### ii. CONVERSION FROM OCTAL TO DECIMAL

Octal Number – $25_8$

| 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 – $21_{10}$

#### iii. CONVERSION FROM HEXADECIMAL TO DECIMAL

Hexa Decimal Number $21_{16}$

| Step | HEXADECIMAL Number | Decimal Number |
|---|---|---|
| Step 1 | $21_{16}$ | $((2 \times 16^1) + (1 \times 16^0))_{10}$ |
| Step 2 | $21_{16}$ | $(32 + 1)_{10}$ |
| Step 3 | $21_{16}$ | $33_{10}$ |

Hexa Decimal Number $21_{16}$ = Decimal Number $33_{10}$

***c.NON-DECIMAL SYSTEM TO NON-DECIMAL SYSTEM***

**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.

**i) Octal to Binary**

Octal Number – $25_8$
Calculating Binary Equivalent –

Step 1 – Convert the given number into Decimal number

| 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 number into Binary Number

| 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 − $10101_2$

### *C. SHORTCUT METHOD - BINARY TO OCTAL*

**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 − $25_8$

## Relationship between Binary - Octal and Binary-hexadecimal

As demonstrated by the table below, there is a direct correspondence between the binary system and the octal system, with three binary digits corresponding to one octal digit. Likewise, four binary digits translate directly into one hexadecimal digit.

| BIN | OCT | HEX | DEC |
|---|---|---|---|
| 0000 | 00 | 0 | 0 |
| 0001 | 01 | 1 | 1 |
| 0010 | 02 | 2 | 2 |
| 0011 | 03 | 3 | 3 |
| 0100 | 04 | 4 | 4 |
| 0101 | 05 | 5 | 5 |
| 0110 | 06 | 6 | 6 |
| 0111 | 07 | 7 | 7 |
| | | | |
| 1000 | 10 | 8 | 8 |
| 1001 | 11 | 9 | 9 |
| 1010 | 12 | A | 10 |
| 1011 | 13 | B | 11 |
| 1100 | 14 | C | 12 |
| 1101 | 15 | D | 13 |
| 1110 | 16 | E | 14 |
| 1111 | 17 | F | 15 |

### *D. SHORTCUT METHOD - OCTAL TO BINARY*

**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**

Convert Octal Number $25_8$ into Binary number

| 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 − $10101_2$

### *E. SHORTCUT METHOD - BINARY TO HEXADECIMAL*

**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**

Convert Binary Number $10101_2$ into Hexadecimal number.

| 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 − $15_{16}$

### *F. SHORTCUT METHOD - HEXADECIMAL TO BINARY*

**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:** Convert Hexadecimal Number $15_{16}$ into Binary number.

| 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 − $10101_2$

## COMPLEMENTS

- Complements are used in the digital computers in order to simplify the subtraction operation and for the logical manipulations.
- For each radix-r system (radix r represents base of number system) there are two types of complements.

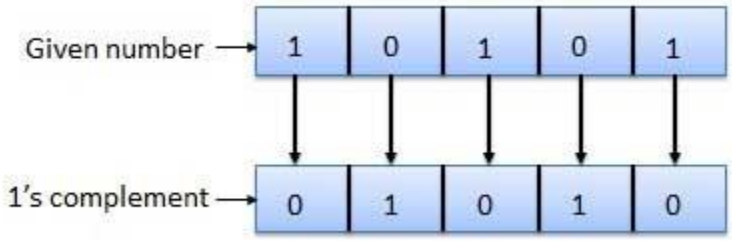
| S.N. | Complement | Description |
|---|---|---|
| 1 | Radix Complement | The radix complement is referred to as the r's complement |
| 2 | Diminished Radix Complement | The diminished radix complement is referred to as the (r-1)'s complement |

### *A. 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.

#### (i) 1's complement

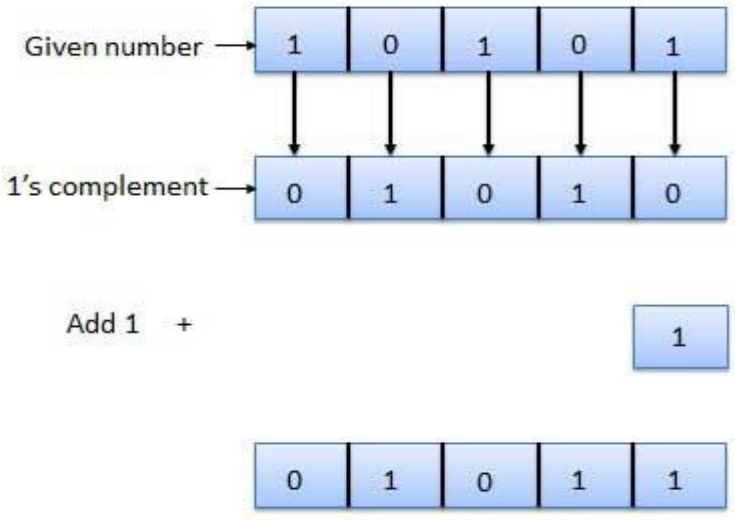
- The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's.
- This is called as taking complement or 1's complement.
- Example of 1's Complement is as follows.




**(ii) 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 of N = 1's complement of N + 1

- Example of 2's Complement is as follows.




***B. Decimal System Complements***

**(i) 9's Complement**

- To obtain the 9's complement of any number we have to subtract the number with $(10^n - 1)$ where n = number of digits in the number

(or)

- In a simpler manner we have to subtract each digit of the given decimal number from 9.
- The table given below will explain the 9's complement more easily.

| Decimal digit | 9s complement |
|---|---|
| 0 | 9 |
| 1 | 8 |

| | |
|---|---|
| 2 | 7 |
| 3 | 6 |
| 4 | 5 |
| 5 | 4 |
| 6 | 3 |
| 7 | 2 |
| 8 | 1 |
| 9 | 0 |

**(ii) 10's complement**

- It is relatively easy to find out the 10's complement after finding out the 9's complement of that number.
- We have to add 1 with the 9's complement of any number to obtain the desired 10's complement of that number.

(or)

- If we want to find out the 10's complement directly, we can do it by following the following formula, ($10^n$ - number),
  where n = number of digits in the number.

**Example:** Let us take a decimal number 456, 9's complement of this number will be

```
  999
(-) 456
-------
  543
```

10's complement of this no:

```
  543
(+)1
-------
  544
```

## Unsigned numbers

H. Unsigned binary numbers are, by definition, positive numbers and thus do not require an arithmetic sign.

I. An *m*-bit unsigned number represents all numbers in the range 0 to $2^m - 1$.

J. For example, the range of 8-bit unsigned binary numbers is from 0 to $255_{10}$ in decimal and from 00 to $FF_{16}$ in hexadecimal.

K. Similarly, the range of 16-bit unsigned binary numbers is from 0 to $65,535_{10}$ in decimal and from 0000 to $FFFF_{16}$ in hexadecimal.

## Subtraction of Unsigned Numbers

### 1. Subtraction of unsigned decimal numbers

#### a) Subtraction with 9's complement

We will understand this method of subtraction via an example
Example 1: A = 215 B = 155
We want to find out A-B by using 9's complement method.

Step 1: First we have to find out 9's complement of B

999

155 (-)

844

Step 2: Now we have to add 9's complement of B to A.

844

215(+)

1059

Step 3: The left most bit of the result is called carry and is added back to the part of the result without it. If there is no carry the answer will be – (9's complement of the answer)

059

1(+)

60

Example 2: Another different type of example is given A = 4567 B = 1234 We need to find out A – B.
9's complement of B is 8765
Adding 9's complement of B with A.

8765

4567

13332

Adding the carry with the result we get 3333 Now the answer is 3333.

**b) 10's complement Subtraction**

Again we will show the procedure by an example taking the same data A = 215 B = 155. 10's complement of B = 845, Adding 10's complement of B to A.

845

215 (+)

1060

In this case the carry is omitted the answer is 60

Taking the other example A = 4567 B = 1234
10's complement of B = 8766.
Adding 10's complement of B with A

8766

4567(+)

13333

To get the answer the carry is ignored So, the answer is 3333.

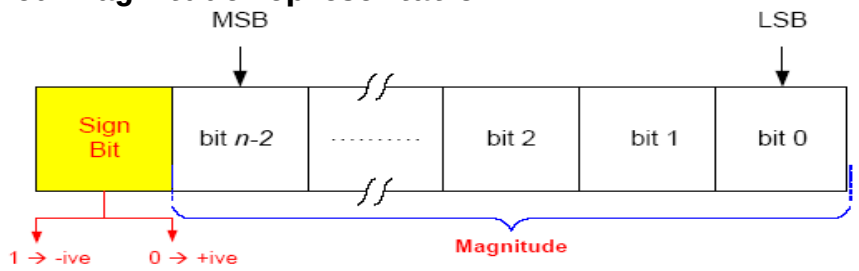
## Signed Numbers and their Representations

- *Signed numbers*, on the other hand, require an arithmetic sign.
- The most significant bit of a binary number is used to represent the sign bit. If the sign bit is equal to zero, the signed binary number is positive; otherwise, it is negative. The remaining bits represent the actual number.
- In digital circuits there is no provision made to put a plus or even a minus sign to a number, since digital systems operate with binary numbers that are represented in terms of “0’s” and “1’s”.

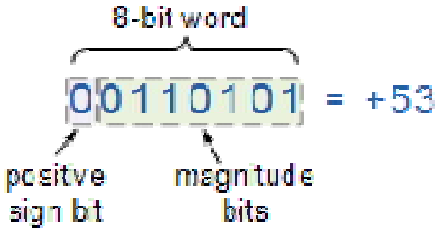
- Total possible numbers representable by an n-bit register using signed magnitude is equal to: $2^{n-1}$, as the MSB is always reserved for the sign
- Largest representable magnitude, in this method, is $(2^{n-1}-1)$
- Range of numbers: $-(2^{n-1}-1)$, ... , -1, -0, +0, +1, ..., $+(2^{n-1}-1)$
- For an 8-bit register, the leftmost bit is a sign bit, which leaves 7 bits to represent the magnitude.
- The largest magnitude representable in 7-bits =127 $=2^{(8-1)}-1$
- A signed number has:
  - Magnitude (or absolute value)
  - Sign (positive or negative)
  - Everything has to be in binary (0s and 1s)

  - Signed-magnitude representation
  - Complement representation

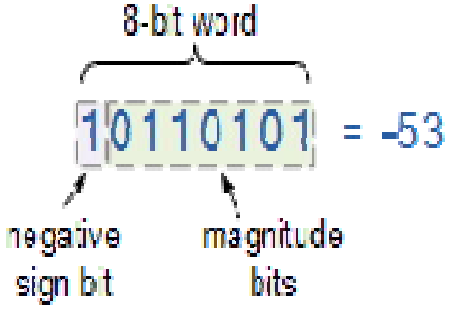
**1) Signed magnitude representation**




**Positive Signed Binary Numbers**




**Negative Signed Binary Numbers**



## 2) Complement representation

- Subtraction: A – B = A + (-B) = A + B'
  - A negative number is represented with its complement
  - Note that B = (B')'
- Two types of complements for each base-r
  - (r-1)'s complement
    - 1's complement
  - r's complement
    - 2's complement

### a) 1's Complement representation

- A positive number is as in signed-magnitude
- For a negative number, flip all bits !
- Negative numbers always have a 1 in the MSB
- 4 bits represent the range: -7 to +7

| Number | 1's Complement |
|---|---|
| +1 | 0001 |
| -1 | 1110 |
| +5 | 0101 |
| -5 | 1010 |
| +0 | 0000 |
| -0 | 1111 |

### b) 2's Complement representation

- A positive number is as in signed-magnitude
- Negative numbers always have a 1 in the MSB
- 4 bits represent the range: -8 to +7
- For a negative number, two way to get the 2's complement:
  - add 1 to 1's complement, or
  - Starting from right find first '1', invert all bit to the left of that one

| Number | 2's Complement |
|---|---|
| +1 | 0001 |
| -1 | 1111 |
| +5 | 0101 |
| -5 | 1011 |
| +0 | 0000 |
| -0 | none |

## 1's Complement vs 2's Complement

**1's Complement**

- Advantages
  - Easy to generate the complement
  - Only one addition process
- Disadvantages
  - Handling last carry out
  - Two different 0s!

**2's Complement**

- Disadvantages
  - Harder to generate the complement
- Advantages
  - Only One zero !
  - Only one addition process
  - No end around carry

*Usually, the 2's complement is the preferred way*

## 4-bit Signed Binary Number Comparison

1. Positive numbers are identical in all representations and have 0 in the MSB
2. 2's complement has only one 0; the others have 2 0s
3. All negative numbers have 1 in the MSB
4. 2's complement has one more negative number!

| Decimal | Signed Magnitude | Signed One's Complement | Signed Two's Complement |
|---|---|---|---|
| +7 | 0111 | 0111 | 0111 |
| +6 | 0110 | 0110 | 0110 |
| +5 | 0101 | 0101 | 0101 |
| +4 | 0100 | 0100 | 0100 |
| +3 | 0011 | 0011 | 0011 |
| +2 | 0010 | 0010 | 0010 |
| +1 | 0001 | 0001 | 0001 |
| +0 | 0000 | 0000 | 0000 |
| -0 | 1000 | 1111 | – |

| | | | |
|---|---|---|---|
| **-1** | **1001** | **1110** | **1111** |
| **-2** | **1010** | **1101** | **1110** |
| **-3** | **1011** | **1100** | **1101** |
| **-4** | **1100** | **1011** | **1100** |
| **-5** | **1101** | **1010** | **1011** |
| **-6** | **1110** | **1001** | **1010** |
| **-7** | **1111** | **1000** | **1001** |

## Signed-Magnitude Arithmetic

Example 1

Signs are different - determine which is larger magnitude

```
  0 0101   (+5)
+ 1 0011   (-3)
```

Put larger magnitude number on top

Subtract

Result has sign of larger magnitude number…

```
  0 0101   (+5)
- 1 0011   (-3)
  0 0010   (+2)
```

---

Example 2

Signs are different - determine which is larger magnitude

```
  0 0010   (+2)
+ 1 0101   (-5)
```

Put larger magnitude number on top

Subtract

Result has sign of larger magnitude number…

```
  1 0101   (-5)
- 0 0010   (+2)
  1 0011   (-3)
```

---

## Subtraction using 1's complement

### Example: Binary subtraction using 1's complement

Regular Approach:

M - N

M = 0 1 0 1 0 1 0 0

N = 0 1 0 0 0 1 0 0 -

---

0 0 0 1 0 0 0 0

1's complement:

M – N = M + N'

M = 0 1 0 1 0 1 0 0

N = 1 0 1 1 1 0 1 1 +

---

End Around Carry 1 0 0 0 0 1 1 1 1

1 +

0 0 0 1 0 0 0 0

Regular Approach:

N - M

N = 0 1 0 0 0 1 0 0

M = 0 1 0 1 0 1 0 0 -

---

- 0 0 0 1 0 0 0 0

1's complement:

N + M'

N = 0 1 0 0 0 1 0 0

M = 1 0 1 0 1 0 1 1 +

---

No End Carry

1 1 1 0 1 1 1 1

**Correction Step Required:**

-(1's complement of 11101111) =

-(**00010000**)

## Subtraction using 2's complement

### Example: Binary subtraction using 2's complement

Regular Approach:

M - N

M = 0 1 0 1 0 1 0 0

N = 0 1 0 0 0 1 0 0 -

---

0 0 0 1 0 0 0 0

2's complement:

M – N = M + N'

M = 0 1 0 1 0 1 0 0

N = 1 0 1 1 1 1 0 0 +

---

Discard End Carry

1 0 0 0 1 0 0 0 0

Regular Approach:

N - M

N = 0 1 0 0 0 1 0 0

M = 0 1 0 1 0 1 0 0 -

---

- 0 0 0 1 0 0 0 0

2's complement:

N + M'

N = 0 1 0 0 0 1 0 0

M = 1 0 1 0 1 1 0 0 +

---

No End Carry

1 1 1 1 0 0 0 0

**Correction Step Required:**

-(2's complement of 11110000) =

-(**00010000**)

## 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 CODE

- 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.

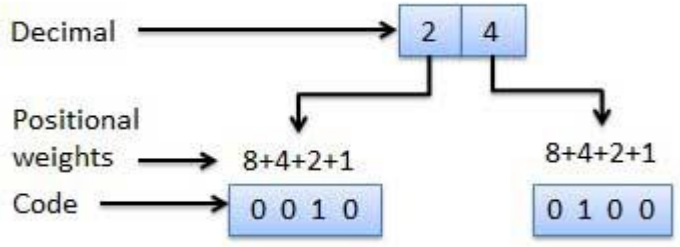
**Classification of binary codes**

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.



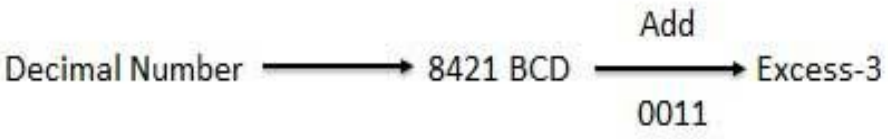
| DECIMAL DIGIT | 8421 | 84-2-1 | 7421 | 5421 | 2421 | BIQUINARY |
|---|---|---|---|---|---|---|
| | 8 4 2 1 | 8 4 -2 -1 | 7 4 2 1 | 5 4 2 1 | 2 4 2 1 | 5 0 4 3 2 1 0 |
| 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 1 0 0 0 0 1 |
| 1 | 0 0 0 1 | 0 1 1 1 | 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | 0 1 0 0 0 1 0 |
| 2 | 0 0 1 0 | 0 1 1 0 | 0 0 1 0 | 0 0 1 0 | 0 0 1 0 | 0 1 0 0 1 0 0 |
| 3 | 0 0 1 1 | 0 1 0 1 | 0 0 1 1 | 0 0 1 1 | 0 0 1 1 | 0 1 0 1 0 0 0 |
| 4 | 0 1 0 0 | 0 1 0 0 | 0 1 0 0 | 0 1 0 0 | 0 1 0 0 | 0 1 1 0 0 0 0 |
| 5 | 0 1 0 1 | 1 0 1 1 | 0 1 0 1 | 0 1 0 1 | 1 0 1 1 | 1 0 0 0 0 0 1 |
| 6 | 0 1 1 0 | 1 0 1 0 | 0 1 1 0 | 0 1 1 0 | 1 1 0 0 | 1 0 0 0 0 1 0 |
| 7 | 0 1 1 1 | 1 0 0 1 | 1 0 0 0 | 0 1 1 1 | 1 1 0 1 | 1 0 0 0 1 0 0 |
| 8 | 1 0 0 0 | 1 0 0 0 | 1 0 0 1 | 1 0 1 1 | 1 1 1 0 | 1 0 0 1 0 0 0 |
| 9 | 1 0 0 1 | 1 1 1 1 | 1 0 1 0 | 1 1 0 0 | 1 1 1 1 | 1 0 1 0 0 0 0 |

***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.

**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 –



| 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 |

## 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

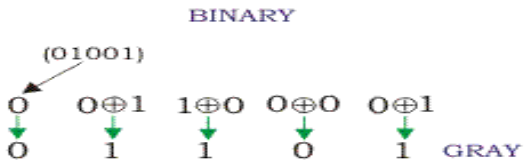
- Gray code is popularly used in the shaft position encoders.
- A shaft position encoder produces a code word which represents the angular position of the shaft.

### *Binary to gray code conversion*

- Binary to gray code conversion is a very simple process.
- There are several steps to do this type of conversions.

Steps given below elaborate the type of conversion.

- The M.S.B. of the gray code will be exactly equal to the first bit of the given binary number.
- Now the second bit of the code will be exclusive-or of the first and second bit of the given binary number, i.e if both the bits are same the result will be 0 and if they are different the result will be 1.
- The third bit of gray code will be equal to the exclusive-or of the second and third bit of the given binary number. Thus the **Binary to gray code conversion** goes on. One example given below can make your idea clear on this type of conversion.




- Thus the equivalent gray code is 01101.
- Now concentrate on the example where the M.S.B. of the binary is 0 so for it will be 0 for the most significant gray bit.
- Next, the XOR of the first and the second bit is done.
- The bits are different so the resultant gray bit will be 1.
- Again move to the next step, XOR of second and third bit is again 1 as they are different.
- Next, XOR of third and fourth bit is 0 as both the bits are same.
- Lastly the XOR of fourth and fifth bit is 1 as they are different.
- That is how the result of binary to gray code conversion of 01001 is done whose equivalent gray code is 01101.

### *Gray code to binary conversion*

Gray code to binary conversion is again very simple and easy process. Following steps can make your idea clear on this type of conversions.

- The M.S.B of the binary number will be equal to the M.S.B of the given gray code.

- Now if the second gray bit is 0 the second binary bit will be same as the previous or the first bit. If the gray bit is 1 the second binary bit will alter. If it was 1 it will be 0 and if it was 0 it will be 1.
- This step is continued for all the bits to do **Gray code to binary conversion**. One example given below will make your idea clear.


- The M.S.B of the binary will be 0 as the M.S.B of gray is 0. Now move to the next gray bit.
- As it is 1 the previous binary bit will alter i.e it will be 1, thus the second binary bit will be 1.
- Next look at the third bit of the gray code. It is again 1 thus the previous bit i.e the second binary bit will again alter and the third bit of the binary number will be 0.
- Now, 4th bit of the given gray is 0 so the previous binary bit will be unchanged, i.e 4th binary bit will be 0.
- Now again the 5th grey bit is 1 thus the previous binary bit will alter, it will be 1 from 0.
- Therefore the equivalent Binary number in case of gray code to binary conversion will be (01001)

***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.

**Conversions**

There are many methods or techniques which can be used to convert code from one format to another. We'll demonstrate here the following

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

***Binary to BCD Conversion***

- **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 − $11101_2$

| 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***

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

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

**Step 1 - Convert to Decimal**

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***

- **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 $(1001)_{BCD}$ to Excess-3.

Step 1 − Convert to decimal

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

Step 2 − Add 3 to decimal

$(9)_{10} + (3)_{10} = (12)_{10}$

Step 3 − Convert to Excess-3

$(12)_{10} = (1100)_2$

Result

$(1001)_{BCD} = (1100)_{XS-3}$

***Excess-3 to BCD Conversion***

- **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   = 0 0 1 1   0 0 1 1
                     ----------------------
               BCD = 0 1 1 0   0 1 1 1
```

Result

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

***BCD Addition***

| Decimal number | Binary number | Binary Coded Decimal(BCD) |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0010 |
| 3 | 0011 | 0011 |
| 4 | 0100 | 0100 |
| 5 | 0101 | 0101 |
| 6 | 0110 | 0110 |
| 7 | 0111 | 0111 |
| 8 | 1000 | 1000 |
| 9 | 1001 | 1001 |
| 10 | 1010 | 0001 0000 |
| 11 | 1011 | 0001 0001 |

| | | |
|---|---|---|
| 12 | 1100 | 0001 0010 |
| 13 | 1101 | 0001 0011 |
| 14 | 1110 | 0001 0100 |
| 15 | 1111 | 0001 0101 |

Like other number system in BCD arithmetical operation may be required. BCD is a numerical code which has several rules for addition. The rules are given below in three steps with an example to make the idea of BCD Addition clear.

1. At first the given number are to be added using the rule of binary.

For example,

Case 1:

$$\begin{array}{r} 1010 \\ +\ 0101 \\ \hline 1111 \end{array}$$

Case 2:

$$\begin{array}{r} 0001 \\ +\ 0101 \\ \hline 0110 \end{array}$$

2. In second step we have to judge the result of addition. Here two cases are shown to describe the rules of BCD Addition. In case 1 the result of addition of two binary number is greater than 9, which is not valid for BCD number. But the result of addition in case 2 is less than 9, which is valid for BCD numbers.
3. If the four bit result of addition is greater than 9 and if a carry bit is present in the result then it is invalid and we have to add 6 whose binary equivalent is $(0110)_2$ to the result of addition. Then the resultant that we would get will be a valid binary coded number. In case 1 the result was $(1111)_2$, which is greater than 9 so we have to add 6 or $(0110)_2$ to it

$$(1111)_2 + (0110)_2 = 0001\ 0101 = 15$$

- As you can see the result is valid in BCD.
- But in case 2 the result was already valid BCD, so there is no need to add 6. This is how BCD Addition could be.

- Now a question may arrive that why 6 is being added to the addition result in case BCD Addition instead of any other numbers.
- It is done to skip the six invalid states of binary coded decimal i.e from 10 to 15 and again return to the BCD codes.
- Now the idea of BCD Addition can be cleared from example.

Example: Let, 0101 is added with 0110.

```
   0101
+  0110
  ------
   1011   → Invalid BCD number
+  0110   → Add 6
  ------
0001 0001 → Valid BCD number
```

*Check your self.* $(0101)_2 \rightarrow (5)_{10}$ & $(0110)_2 \rightarrow (6)_{10}$ $(5)_{10} + (6)_{10} = (11)_{10}$

## ***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).

- 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.

**ASCII Code (Decimal)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 nul | 16 dle | 32 sp | 48 0 | 64 @ | 80 P | 96 ` | 112 p |
| 1 soh | 17 dc1 | 33 ! | 49 1 | 65 A | 81 Q | 97 a | 113 q |
| 2 stx | 18 dc2 | 34 " | 50 2 | 66 B | 82 R | 98 b | 114 r |
| 3 etx | 19 dc3 | 35 # | 51 3 | 67 C | 83 S | 99 c | 115 s |
| 4 eot | 20 dc4 | 36 $ | 52 4 | 68 D | 84 T | 100 d | 116 t |
| 5 enq | 21 nak | 37 % | 53 5 | 69 E | 85 U | 101 e | 117 u |
| 6 ack | 22 syn | 38 & | 54 6 | 70 F | 86 V | 102 f | 118 v |
| 7 bel | 23 etb | 39 ' | 55 7 | 71 G | 87 W | 103 g | 119 w |
| 8 bs | 24 can | 40 ( | 56 8 | 72 H | 88 X | 104 h | 120 x |
| 9 ht | 25 em | 41 ) | 57 9 | 73 I | 89 Y | 105 i | 121 y |
| 10 nl | 26 sub | 42 * | 58 : | 74 J | 90 Z | 106 j | 122 z |
| 11 vt | 27 esc | 43 + | 59 ; | 75 K | 91 [ | 107 k | 123 { |
| 12 np | 28 fs | 44 , | 60 < | 76 L | 92 \ | 108 l | 124 \| |
| 13 cr | 29 gs | 45 - | 61 = | 77 M | 93 ] | 109 m | 125 } |
| 14 so | 30 rs | 46 . | 62 > | 78 N | 94 ^ | 110 n | 126 ~ |
| 15 si | 31 us | 47 / | 63 ? | 79 O | 95 _ | 111 o | 127 del |

- The characters between 0 and 31 are generally not printable (control characters, etc). 32 is the space character.
- Also note that there are only 128 ASCII characters.
- This means only 7 bits are required to represent an ASCII character.
- However, since the smallest size representation on most computers is a byte, a byte is used to store an ASCII character.
- The Most Significant bit(MSB) of an ASCII character is 0.

## UNIT-I
## Assignment-Cum-Tutorial Questions
## SECTION-A

### Objective Questions

1. $(254)_{10}$ =---------- in Octal system.
2. $(11001001)_2$=---------- in Decimal. [ ]
   a) 201 b) 20 c) 2001 d) 210
3. $(1217)_8$ is equivalent to ---------- [ ]
   a)$(1217)_{16}$ b)$(028F)_{16}$ c)$(2297)_{10}$ d)$(0B17)_{16}$
4. $(734)_8 = (\ )_{16}$ [ ]
   a) C 1 D b) D C 1 c) 1 C D d) 1 D C
5. Convert binary 111111110010 to hexadecimal. [ ]
   a) $(EE2)_{16}$ b)$(FF2)_{16}$ c) $(2FE)_{16}$ d)$(FD2)_{16}$
6. 11001, 1001 and 111001 correspond to the 2's complement representation of which one of the following sets of number?[ ]
   a) 25, 9 and 57 respectively b) -6, -6 and -6 respectively
   c) -7, -7 and -7 respectively d) -25, -9 and -57 respectively
7. Which of the following Twos Complement binary numbers is equivalent to $-75_{10}$? [ ]
   a) 11001011 b) 01001100 c) 11001100 d) 10110101
8. -8 is equal to signed magnitude binary number [ ]
   a) 10001000 b) 00001000 c)10000000 d)11000000
9. 9's complement of 546700 is [ ]
   a)453299 b)453399 c)543399 d)543299
10. 2's complement representation of a 16 bit number (one sign bit and 15 magnitude bits) is FFFF. Its magnitude in decimal representation is
    a)0 b)1 c)32,767 d) 65,535
11. No of Characters specified in 6-bit code are [ ]
    a) 61 b) 62 c) 63 d) 64
12. Binary code that distinguishes ten elements must contain at least
    a)Two bits b)Three bits c)Four bits d)Five bits [ ]
13. End around carry is used to correct the result of additions in which of the following number systems? [ ]
    a) 8 bit Signed Binary. b) 8 bit Ones Complement.
    c) 8 bit Twos Complement. d) Excess 3

14. The two numbers represented in signed 2's compliment form are P = 11101101 and Q = 11100110. If Q is subtracted from P, the value obtained in signed 2's compliment form is [ ]
    a) 100000111 b) 00000111 c) 11111001 d) 111111001
15. The number of bits used to store a BCD digit is: [ ]
    a) 2 b) 1 c) 4 d) 8
16. 842 + 537 = [ ]
    a)0001 0011 0111 0101 b)0111 0011 0111 0101
    c)0001 0111 0111 0101 d)0001 0011 0111 0111
17. 3428 is the decimal value for which of the following binary coded decimal (BCD) groupings? [ ]
    a) 110100001101010 b) 11010001001000
    c) 1010000101000 d) 011010010000010
18. 8723 in BCD [ ]
    a)1000_0111_0010_0011 b)1000_0001_0010_0011
    c)1000_0101_0010_0011 d)1000_0111_0110_0011
19. The excess-3 code of decimal 7 is represented by [ ]
    a)1100 b)1001 c)1011 d)1010
20. ASCII stands for [ ]
    a) African standard code for information interchange
    b) American standard code for integer interchange
    c) American standard code for information interchange
    d) African standard code for integer interchange

## SECTION-B

## SUBJECTIVE QUESTIONS

1) Convert $(27.315)_{10}$ into the number with base 2, 5, 8 and 16
2) Convert the following number with indicated bases to decimal
   i. $(1\ 0\ 1\ 1\ 1\ 1)_2$ ii. $(A\ 3\ B.3C)_{16}$ iii. $(2\ 3\ 7)_8$ iv. $(4\ 3.02)_5$
3) Convert $(2AC5.D)_H$ to octal and binary.
4) A) Determine the value of base X, If
   i. $(193)_x = (623)_8$ ii $(225)_x = (341)_8$

B) Determine the base of the numbers in each case for the following operations to be correct:
   i. 14/2=5, ii. 24+17=40.

C) The solutions to the quadratic equation $X^2$-11X+22=0 are X=3 and X=6. What is the base of the number?

5) Devise a scheme for converting base 3 numbers directly to base 9. Use your method to convert the following number to base 9: $(1110212.20211)_3$

6) Perform the following using 1's and 2's complement

(i)add -20 to +26 (ii) add +25 to -15

7) perform the following using signed 9's and 10's complement.

(i)(+9286)+(+801) (ii) (+9286)+(-801)

(iii) (-9286)+(+801) (iv)(-9286)+(-801)

8) Represent 54 using (i) BCD (ii) 3321 (iii) 84-2-1 (iv) 6311 codes.

9) Represent the unsigned decimal numbers 842 and 535 in BCD and then show the steps necessary to form their sum.

10) A) Give the binary, excess-3 and gray code for 0-9 numbers.

B) (i) Convert gray code 101011 into binary

(ii) Convert binary code 10111011 into gray code.

## SECTION-C

## QUESTIONS AT THE LEVEL OF GATE

1. The 16 bit 2's complement representation of an integer is 1111 1111 1111 0101; its decimal representation is----------
   **[Gate-2016]**
2. Consider the equation $(43)_x=(y3)_8$ where x and y as unknown. The number of possible solutions is----------
   **[Gate-2015]**
3. 7. Decimal 43 in Hexadecimal and BCD number system is respectively
   a)B2, 0100 0011 b)2B, 0100 0011 **[Gate-2005]**
   c)2B, 0011 0100 d)B2, 0100 0100 [ ]
4. 4– bit 2's complement representation of a decimal number is 1000. The number is **[Gate-2002]**
   a)+8 b) 0 c)-7 d)-8 [ ]
5. The 2's complement representation of -17 is **[Gate-2001]**
   a)101110 b)101111 c)111110 d)1100001 [ ]