## Course Contents

**Unit-05:Data Representation (6 Hrs.)**

- **Introduction; Number System;**
- **Conversion from Decimal to Binary, Octal, Hexadecimal;**
- **Conversion of Binary, Octal, Hexadecimal to Decimal;**
- **Conversion of Binary to Octal, Hexadecimal;**
- **Conversion of Octal, Hexadecimal to Binary;**
- **Binary Arithmetic;**
- **Signed and Unsigned Numbers;**
- **Binary Data Representation;**
- **Binary Coding Schemes;**
- **Logic Gates**

## Course Contents

**Unit-05:Data Representation**

- **Introduction; Number System;**
- **Conversion from Decimal To Binary, Octal, Hexadecimal;**
- **Conversion of Binary, Octal, Hexadecimal To Decimal;**
- **Conversion of Binary To Octal, Hexadecimal;**
- **Conversion of Octal, Hexadecimal To Binary;**
- **Binary Arithmetic;**
- **Signed and Unsigned Numbers;**
- **Binary Data Representation;**
- **Binary Coding Schemes;**
- **Logic Gates**

## Course Contents

**Introduction:**

- **We use computer to process data and get desire output.**
- **The data input can be in the form of alphabets, digits, symbols, audio, video etc but computer can only understand 0 and 1 so data must be represented in the computer in 0's and 1's.**

**We will study data representation in this chapter.**

- **Numeric data(0,1,2..9)**
- **Alphabetic data(A,B,C,…..Z)**
- **Alphanumeric data – Combination of any symbols (A,BC,…Z), (0,1,2..9), or special characters (+,-, blank etc)**

## Course Contents

**Number System:**

- **There are infinite ways to represent a number.**
- **The four commonly associated with modern computers and digital electronics are: Decimal, Binary, Octal, and Hexadecimal.**

- Decimal (base 10) is the way most human beings represent numbers. Decimal is sometimes abbreviated as dec.
- Decimal counting goes:
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, and so on.

## Course Contents

**Number System:**

- **Binary** (base 2) is the natural way most digital circuits represent and manipulate numbers.

- Binary is sometimes abbreviated as bin.
- Binary counting goes:
  0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001, and so on.

## Course Contents

**Number System:**

- **Octal** (base 8) was previously a popular choice for representing digital circuit numbers in a form that is more compact than binary.
- Octal is sometimes abbreviated as oct.
- Octal counting goes:
  0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, and so on.

## Course Contents

**Number System:**

- **Hexadecimal** (base 16) is currently the most popular choice for representing digital circuit numbers in a form that is more compact than binary.
- Hexadecimal is sometimes abbreviated as hex.
- Hexadecimal counting goes:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, and so on.

## Course Contents

**Number System:**

- Base number: The base of the number decides the valid digits that are used to make a number. Decimal number has a base 10, Octal is 8 and Hexadecimal is 16
- Face Value : The face value of a digit is the digit located at that position. E.g. in decimal 52, face value at position 0 is 2 and face value at position 1 is 5
- Position Value: The position value of a digit is ($base^{position}$). E.g. in decimal number 52, the position value of digit 2 is $10^0$ and position value of digit 5 is $10^1$.

## Course Contents

**Number System:**

We are concerned with 4 kinds of number.

1. Decimal Number System – Base 10
2. Binary Number System – Base 2
3. Octal Number System – Base 8
4. Hexadecimal Number System – Base 16

- Number given as input to computer and output from computer are generally in decimal number system and are most understood by humans.
- But computer understand binary number system (0's and 1's). Binary data also represented internally as in Octal and Hexa due to their ease of use.

## Course Contents

**Number System:**

- All four number systems are equally capable of representing any number.
- A number can be perfectly converted between the various number systems without any loss of numeric value.
- Since the job of electrical and software engineers is to work with digital circuits, engineers require number systems that can best transfer information between the human world and the digital circuit world.

## Course Contents

**Number System:**

- An octal number (base 8) can be up to 1/3 the length of a binary number (base 2). 8 is a whole power of 2 ($2^3$=8). That means three binary digits convert neatly into one octal digit.
- A hexadecimal number (base 16) can be up to 1/4 the length of a binary number. 16 is a whole power of 2 ($2^4$=16). That means four binary digits convert neatly into one hexadecimal digit.
- Unfortunately, decimal (base 10) is not a whole power of 2. So, it is not possible to simply chunk groups of binary digits to convert the raw state of a digital circuit into the human-centric format.

## Course Contents

| DECIMAL | BINARY | OCTAL | HEXADECIMAL |
|---|---|---|---|
| 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 |

## Course Contents

**Conversion from Decimal to Binary: Example of 1792**

| Decimal Number | Operation | Quotient | Remainder | Binary Result |
|---|---|---|---|---|
| 1792 | ÷ 2 = | 896 | 0 | 0 |
| 896 | ÷ 2 = | 448 | 0 | 00 |
| 448 | ÷ 2 = | 224 | 0 | 000 |
| 224 | ÷ 2 = | 112 | 0 | 0000 |
| 112 | ÷ 2 = | 56 | 0 | 00000 |
| 56 | ÷ 2 = | 28 | 0 | 000000 |
| 28 | ÷ 2 = | 14 | 0 | 0000000 |
| 14 | ÷ 2 = | 7 | 0 | 00000000 |
| 7 | ÷ 2 = | 3 | 1 | 100000000 |
| 3 | ÷ 2 = | 1 | 1 | 1100000000 |
| 1 | ÷ 2 = | 0 | 1 | 11100000000 |
| 0 | done. | | | |

## Course Contents

**Conversion from Decimal to Octal : Example to convert 1792 decimal to octal:**

| Decimal Number | Operation | Quotient | Remainder | Octal Result |
|---|---|---|---|---|
| 1792 | ÷ 8 = | 224 | 0 | 0 |
| 224 | ÷ 8 = | 28 | 0 | 00 |
| 28 | ÷ 8 = | 3 | 4 | 400 |
| 3 | ÷ 8 = | 0 | 3 | 3400 |
| 0 | done. | | | |

## Course Contents

**Conversion from Decimal to Hexadecimal:**

| Decimal Number | Operation | Quotient | Remainder | Hexadecimal Result |
|---|---|---|---|---|
| 1792 | ÷ 16 = | 112 | 0 | 0 |
| 112 | ÷ 16 = | 7 | 0 | 00 |
| 7 | ÷ 16 = | 0 | 7 | 700 |
| 0 | done. | | | |

## Course Contents

**Conversion from Decimal to Hexadecimal:**

| Decimal Number | Operation | Quotient | Remainder | Hexadecimal Result |
|---|---|---|---|---|
| 48879 | ÷ 16 = | 3054 | 15 | F |
| 3054 | ÷ 16 = | 190 | 14 | EF |
| 190 | ÷ 16 = | 11 | 14 | EEF |
| 11 | ÷ 16 = | 0 | 11 | BEEF |
| 0 | done. | | | |

## Course Contents

**Conversion of Binary to Decimal:**

Convert binary number $1010_{(\text{base } 2)}$ into decimal form

$1010_{(\text{base } 2)} = 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 = 8 + 0 + 2 + 0 = 10$

So, the required decimal number is
$110101_{(\text{base } 2)} = 53_{(\text{base } 10)}$

Alternatively, $(110101)_2 = (53)_{10}$
Where, (base 10) means the number is in decimal number system and (base 2) means the number is in binary number system.

## Course Contents

**Conversion of Octal to Decimal:**

Convert octal number $156_{(\text{base } 8)}$ into decimal form

$156 = 1x8^2 + 5x8^1 + 6x8^0 = 64 + 40 + 6 = 110$

## Course Contents

**Conversion of Hexadecimal to Decimal:**

Convert hexadecimal number A1$_{(base\ 16)}$ into decimal form

A1$_{(base\ 16)}$ = Ax$16^1$ + 1x$16^0$ = 10x$16^1$ + 1x$16^0$ = 160 + 1 = 161

So, the required decimal number is
A1$_{(base\ 16)}$ = 161$_{(base\ 10)}$

Alternatively, $(A1)_{16} = (161)_{10}$
Where, (base 10) means the number is in decimal number system and (base 16) means the number is in hexadecimal number system.

## Course Contents

**Conversion of Binary to Octal:**

Steps:

1. Divide the binary digits into groups of 3 digits, starting from the right.
2. Convert each group of 3 binary digits into 1 octal digit.

Convert Binary number $100101_2$ into Octal form

Step 1. Make groups of 3 digits from right $100101_2$
Groups: $100_2$ $101_2$
Step 2. Convert each 3 digits group into 1 octal digit
$101_{2\,=}$ = $5_8$ $100_2$ = $4_8$ so, $100101_2$ = $45_8$

## Course Contents

**Conversion of Binary to Hexadecimal:**

Steps:

1. Divide the binary digits into groups of 4 digits, starting from the right
2. Convert each group of 4 binary digits into 1 hexadecimal digit

Convert Binary number 101001012 into Hexadecimal form

Step 1. Make groups of 4 digits from right $10100101_2$
Groups: $1010_2$ $0101_2$
$0101_2 = 5_{16}$
$1010_2 = 4_{16}$
Step 2. Combine the groups so, $10100101_2 = 45_{16}$

## Course Contents

**Conversion of Octal to Binary:**

Steps
1. Convert each octal digits into 3 digits binary group
2. Combine the groups

Convert Octal number $45_8$ into Binary form

Step 1. Convert each octal digit into 3 digits binary group
$45_8$ Groups: $4_8$ $5_8$
$5_8 = 101_2$
$4_8 = 100_2$
Step 2. Combine the groups
so, $45_8 = 100101_2$

## Course Contents

**Conversion of Hexadecimal to Binary:**

Steps

1. Convert each hexadecimal digit into group of 4 digits binary
2. Combine the groups

Convert Hexadecimal number $A5_{16}$ into Binary form

Step 1. Convert each hexadecimal digit into group of 4 digits binary
$A5_{16}$
Groups: $A_{16}$ $5_{16}$
$5_{16} = 0101_2$
$A_{16} = 1010_2$
Step 2. Combine the groups
so, $A5_{16} = 10100101_2$

## Course Contents

**Octal to Hexadecimal conversion:**

Steps

1. Convert each octal digit into groups of 3 digits binary
2. Combine the groups from step 1
3. Divide the binary digits from step 2 into groups of 4 digits, starting from the right
4. Convert each group of 4 binary digits into 1 hexadecimal digit

## Course Contents

**Octal to Hexadecimal conversion:**

Convert Octal number $25_8$ into Hexadecimal form

Step 1. Convert each octal digit into groups of 3 digits binary $25_8$
Groups: $2_8$ $5_8$
$5_8 = 101_2$
$2_8 = 010_2$
Step 2. Combine the groups
so, $25_8 = 010101_2$
Step 3. Divide the binary digits from step 2 into groups of 4 digits, starting from the right
Groups: $0001_2$ $0101_2$
Step 4. Convert each group of 4 binary digits into 1 hexadecimal digit
$0101_2 = 5_{16}$
$0001_2 = 1_{16}$
so, $25_8 = 15_{16}$

## Course Contents

**Hexadecimal to Octal conversion:**

Steps:

1. Convert each hexadecimal digit into groups of 4 digits binary
2. Combine the groups from step 1
3. Divide the binary digits from step 2 into groups of 3 digits, starting from the right
4. Convert each group of 3 binary digits into 1 octal digit

## Course Contents

**Hexadecimal to Octal conversion:**

Convert Hexadecimal number $15_{16}$ into Octal form

Step 1. Convert each hexadecimal digit into groups of 4 digits binary $15_{16}$
Groups: $1_{16}$ $5_{16}$
$5_{16} = 0101_2$
$1_{16} = 0001_2$
Step 2. Combine the groups
so, $15_{16} = 00010101_2$
Step 3. Divide the binary digits from step 2 into groups of 3 digits, starting from the right
Groups: $000_2$ $010_2$ $101_2$
Step 4. Convert each group of 3 binary digits into 1 octal digit
$101_2 = 5_8$
$010_2 = 2_8$
$000_2 = 0_8$
so, $15_{16} = 025_8 = 25_8$

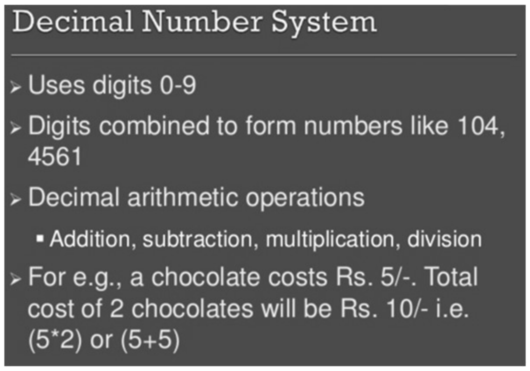
## Course Contents

**Unit-05:Data Representation (6 Hrs.)**

- **Binary Arithmetic;**
- **Signed and Unsigned Numbers;**
- **Binary Data Representation;**
- **Binary Coding Schemes;**
- **Logic Gates**

## Course Contents

**Binary Arithmetic:**

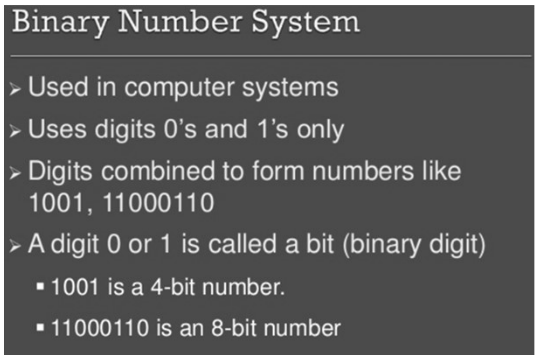


Decimal Number System

- Uses digits 0-9
- Digits combined to form numbers like 104, 4561
- Decimal arithmetic operations
  - Addition, subtraction, multiplication, division
- For e.g., a chocolate costs Rs. 5/-. Total cost of 2 chocolates will be Rs. 10/- i.e. (5*2) or (5+5)

## Course Contents

**Binary Arithmetic:**

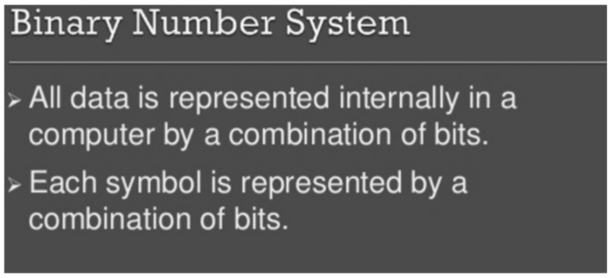


Binary Number System

- Used in computer systems
- Uses digits 0's and 1's only
- Digits combined to form numbers like 1001, 11000110
- A digit 0 or 1 is called a bit (binary digit)
  - 1001 is a 4-bit number.
  - 11000110 is an 8-bit number

## Course Contents

**Binary Arithmetic:**

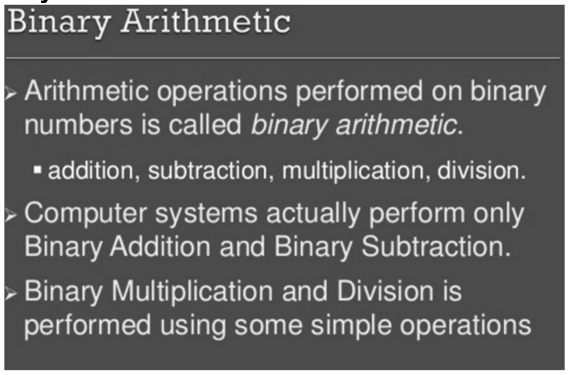


### Binary Number System

- All data is represented internally in a computer by a combination of bits.
- Each symbol is represented by a combination of bits.

## Course Contents

**Binary Arithmetic:**

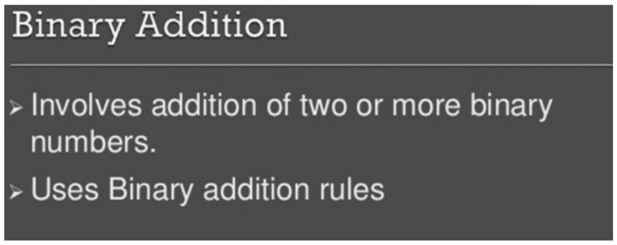


### Binary Arithmetic

- Arithmetic operations performed on binary numbers is called *binary arithmetic*.
  - addition, subtraction, multiplication, division.
- Computer systems actually perform only Binary Addition and Binary Subtraction.
- Binary Multiplication and Division is performed using some simple operations

## Course Contents

**Binary Arithmetic:**

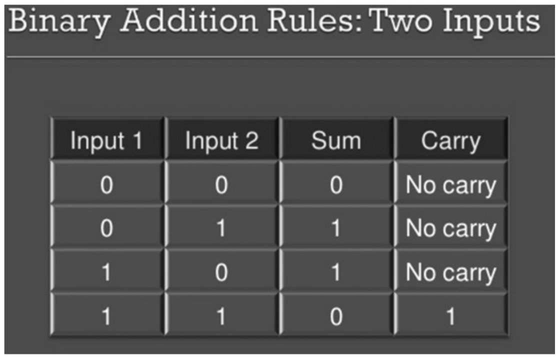


### Binary Addition

- Involves addition of two or more binary numbers.
- Uses Binary addition rules

## Course Contents

**Binary Arithmetic:**

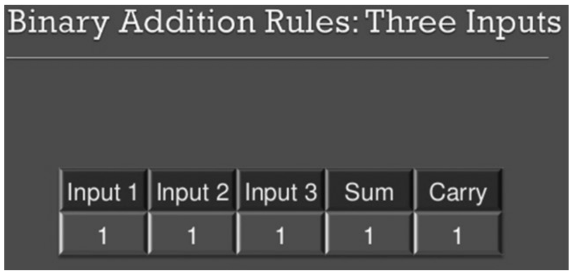


### Binary Addition Rules: Two Inputs

| Input 1 | Input 2 | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | No carry |
| 0 | 1 | 1 | No carry |
| 1 | 0 | 1 | No carry |
| 1 | 1 | 0 | 1 |

## Course Contents

**Binary Arithmetic:**

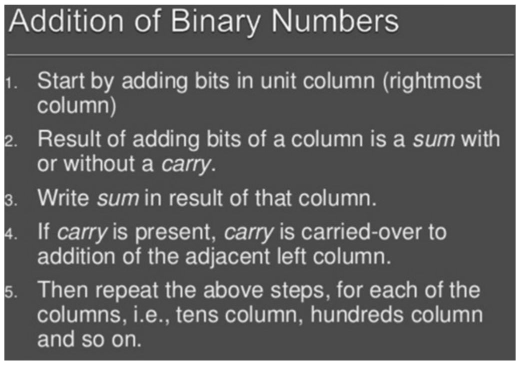


### Binary Addition Rules: Three Inputs

| Input 1 | Input 2 | Input 3 | Sum | Carry |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |

## Course Contents

**Binary Arithmetic:**

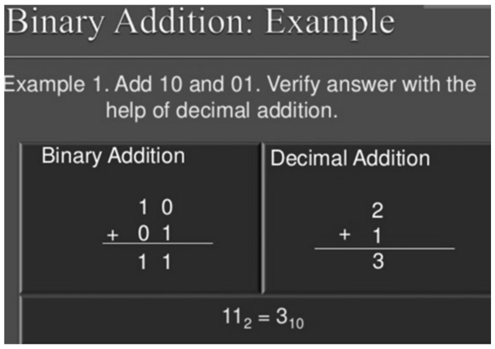


### Addition of Binary Numbers

1. Start by adding bits in unit column (rightmost column)
2. Result of adding bits of a column is a *sum* with or without a *carry*.
3. Write *sum* in result of that column.
4. If *carry* is present, *carry* is carried-over to addition of the adjacent left column.
5. Then repeat the above steps, for each of the columns, i.e., tens column, hundreds column and so on.

## Course Contents

**Binary Arithmetic:**



Binary Addition: Example

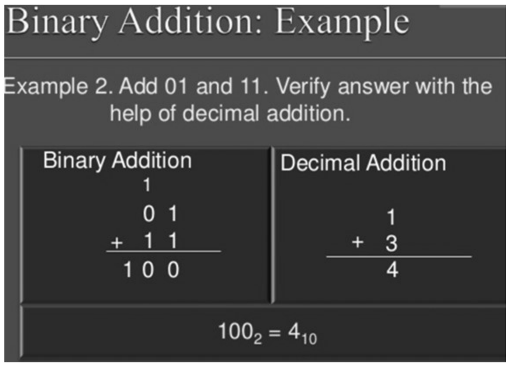
Example 1. Add 10 and 01. Verify answer with the help of decimal addition.

| Binary Addition | Decimal Addition |
|---|---|
| 1 0 + 0 1 = 1 1 | 2 + 1 = 3 |

$11_2 = 3_{10}$

## Course Contents

**Binary Arithmetic:**



Binary Addition: Example

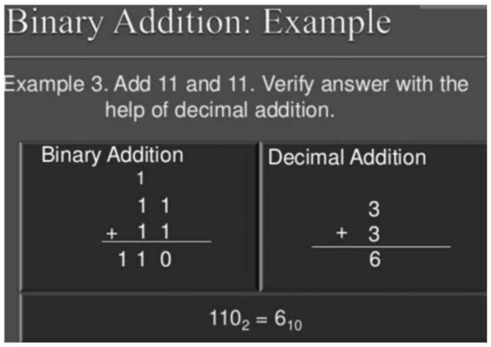
Example 2. Add 01 and 11. Verify answer with the help of decimal addition.

```
Binary Addition      Decimal Addition
     1
     0 1                  1
   + 1 1                + 3
   -----                ---
   1 0 0                  4
```

$100_2 = 4_{10}$

## Course Contents

**Binary Arithmetic:**



Binary Addition: Example

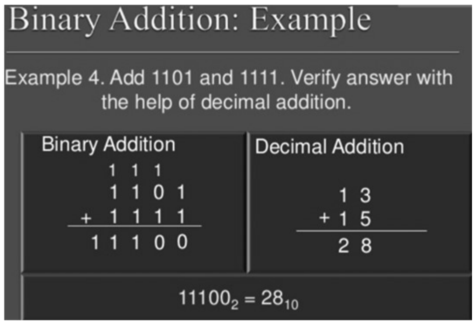
Example 3. Add 11 and 11. Verify answer with the help of decimal addition.

```
Binary Addition      Decimal Addition
     1
     1 1                  3
   + 1 1                + 3
   -----                ---
   1 1 0                  6
```

$110_2 = 6_{10}$

## Course Contents

**Binary Arithmetic:**



Binary Addition: Example

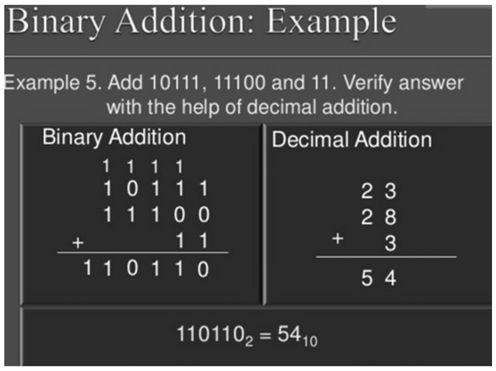
Example 4. Add 1101 and 1111. Verify answer with the help of decimal addition.

```
Binary Addition      Decimal Addition
    1 1 1
    1 1 0 1               1 3
  + 1 1 1 1             + 1 5
  ---------             -----
  1 1 1 0 0               2 8
```

$11100_2 = 28_{10}$

## Course Contents

**Binary Arithmetic:**



Binary Addition: Example

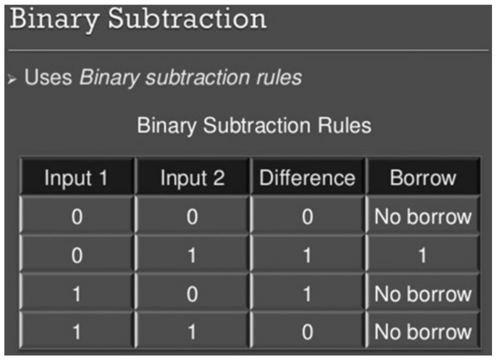
Example 5. Add 10111, 11100 and 11. Verify answer with the help of decimal addition.

```
Binary Addition      Decimal Addition
      1 1 1 1
      1 0 1 1 1            2 3
      1 1 1 0 0            2 8
  +         1 1          +   3
  -------------          -----
    1 1 0 1 1 0            5 4
```

$110110_2 = 54_{10}$

## Course Contents

**Binary Arithmetic:**



Binary Subtraction

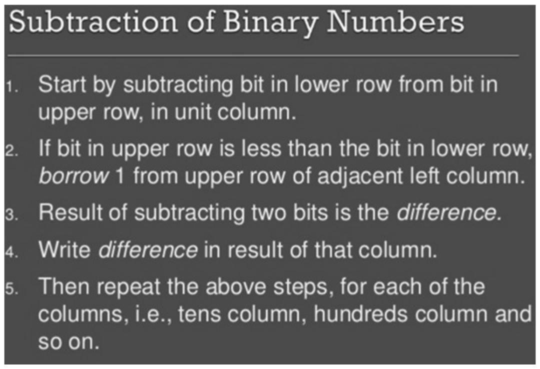
- Uses *Binary subtraction rules*

Binary Subtraction Rules

| Input 1 | Input 2 | Difference | Borrow |
|---|---|---|---|
| 0 | 0 | 0 | No borrow |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | No borrow |
| 1 | 1 | 0 | No borrow |

## Course Contents

**Binary Arithmetic:**

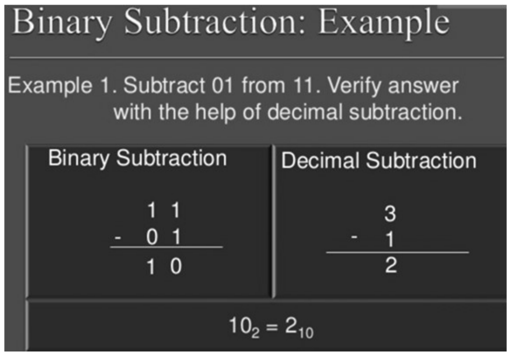


### Subtraction of Binary Numbers

1. Start by subtracting bit in lower row from bit in upper row, in unit column.
2. If bit in upper row is less than the bit in lower row, *borrow* 1 from upper row of adjacent left column.
3. Result of subtracting two bits is the *difference*.
4. Write *difference* in result of that column.
5. Then repeat the above steps, for each of the columns, i.e., tens column, hundreds column and so on.

## Course Contents

**Binary Arithmetic:**



### Binary Subtraction: Example

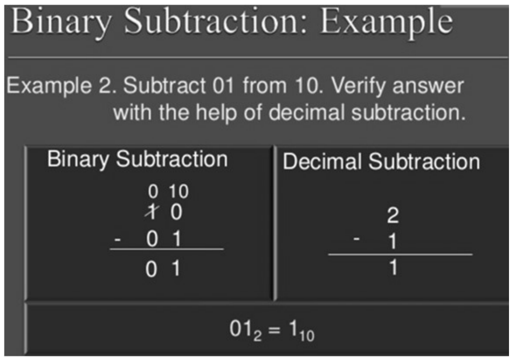
Example 1. Subtract 01 from 11. Verify answer with the help of decimal subtraction.

| Binary Subtraction | Decimal Subtraction |
|---|---|
| $\begin{array}{ccc} & 1 & 1 \\ - & 0 & 1 \\ \hline & 1 & 0 \end{array}$ | $\begin{array}{cc} & 3 \\ - & 1 \\ \hline & 2 \end{array}$ |

$10_2 = 2_{10}$

## Course Contents

**Binary Arithmetic:**



### Binary Subtraction: Example

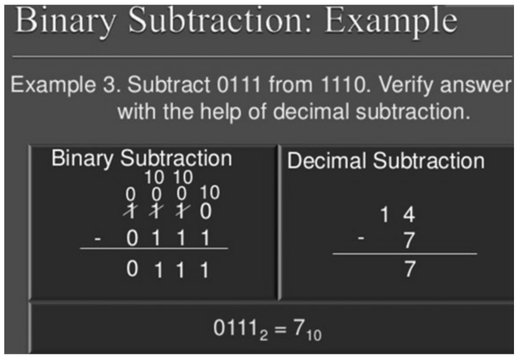
Example 2. Subtract 01 from 10. Verify answer with the help of decimal subtraction.

| Binary Subtraction | Decimal Subtraction |
|---|---|
| $\begin{array}{ccc} & 0 & 10 \\ & \cancel{1} & 0 \\ - & 0 & 1 \\ \hline & 0 & 1 \end{array}$ | $\begin{array}{cc} & 2 \\ - & 1 \\ \hline & 1 \end{array}$ |

$01_2 = 1_{10}$

## Course Contents

**Binary Arithmetic:**



### Binary Subtraction: Example

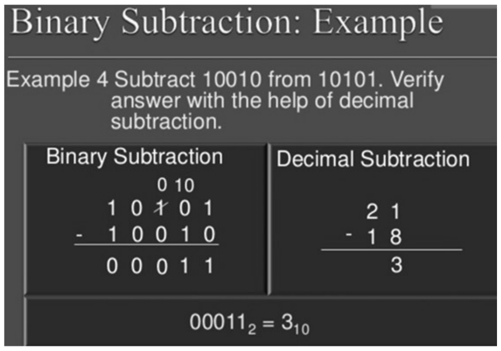
Example 3. Subtract 0111 from 1110. Verify answer with the help of decimal subtraction.

| Binary Subtraction | Decimal Subtraction |
|---|---|
| $\begin{array}{ccccc} & & 10 & 10 & \\ & 0 & 0 & 0 & 10 \\ & \cancel{1} & \cancel{1} & \cancel{1} & 0 \\ - & 0 & 1 & 1 & 1 \\ \hline & 0 & 1 & 1 & 1 \end{array}$ | $\begin{array}{ccc} & 1 & 4 \\ - & & 7 \\ \hline & & 7 \end{array}$ |

$0111_2 = 7_{10}$

## Course Contents

**Binary Arithmetic:**



### Binary Subtraction: Example

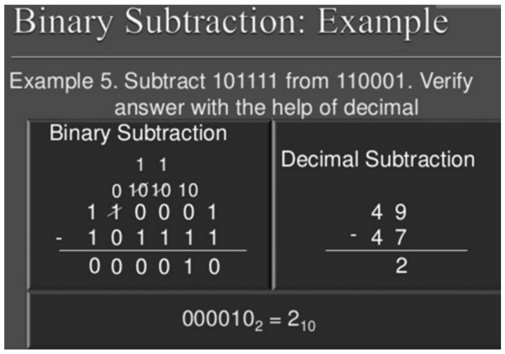
Example 4 Subtract 10010 from 10101. Verify answer with the help of decimal subtraction.

| Binary Subtraction | Decimal Subtraction |
|---|---|
| $\begin{array}{cccccc} & & & 0 & 10 & \\ & 1 & 0 & \cancel{1} & 0 & 1 \\ - & 1 & 0 & 0 & 1 & 0 \\ \hline & 0 & 0 & 0 & 1 & 1 \end{array}$ | $\begin{array}{ccc} & 2 & 1 \\ - & 1 & 8 \\ \hline & & 3 \end{array}$ |

$00011_2 = 3_{10}$

## Course Contents

**Binary Arithmetic:**



### Binary Subtraction: Example

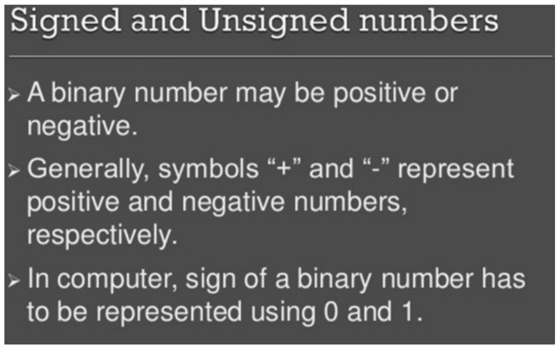
Example 5. Subtract 101111 from 110001. Verify answer with the help of decimal

| Binary Subtraction | Decimal Subtraction |
|---|---|
| $\begin{array}{ccccccc} & & & 1 & 1 & & \\ & & 0 & \cancel{10} & \cancel{10} & 10 & \\ & 1 & \cancel{1} & 0 & 0 & 0 & 1 \\ - & 1 & 0 & 1 & 1 & 1 & 1 \\ \hline & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$ | $\begin{array}{ccc} & 4 & 9 \\ - & 4 & 7 \\ \hline & & 2 \end{array}$ |

$000010_2 = 2_{10}$

## Course Contents

**Binary Arithmetic:**

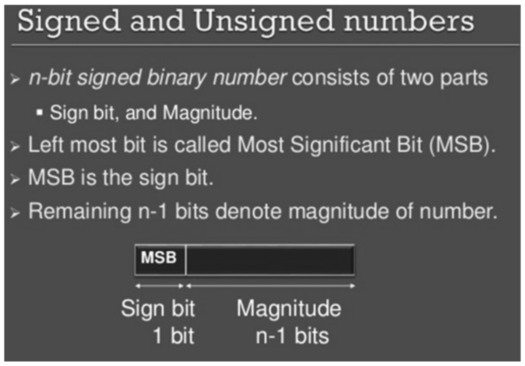


### Signed and Unsigned numbers

- A binary number may be positive or negative.
- Generally, symbols "+" and "-" represent positive and negative numbers, respectively.
- In computer, sign of a binary number has to be represented using 0 and 1.

## Course Contents

**Binary Arithmetic:**



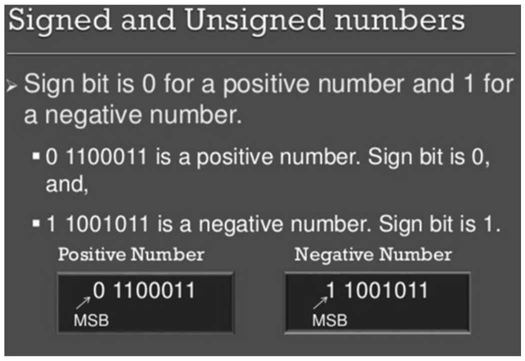
### Signed and Unsigned numbers

- *n-bit signed binary number* consists of two parts
  - Sign bit, and Magnitude.
- Left most bit is called Most Significant Bit (MSB).
- MSB is the sign bit.
- Remaining n-1 bits denote magnitude of number.

MSB
Sign bit 1 bit
Magnitude n-1 bits

## Course Contents

**Binary Arithmetic:**



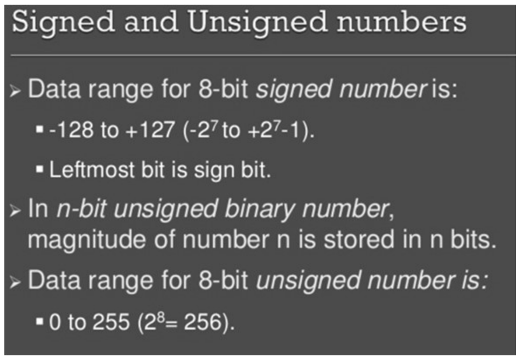
### Signed and Unsigned numbers

- Sign bit is 0 for a positive number and 1 for a negative number.
  - 0 1100011 is a positive number. Sign bit is 0, and,
  - 1 1001011 is a negative number. Sign bit is 1.

Positive Number
0 1100011
MSB
Negative Number
1 1001011
MSB

## Course Contents

**Binary Arithmetic:**

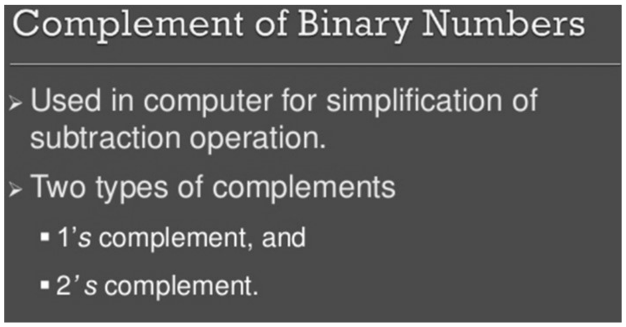


### Signed and Unsigned numbers

- Data range for 8-bit *signed number* is:
  - -128 to +127 ($-2^7$ to $+2^7-1$).
  - Leftmost bit is sign bit.
- In *n-bit unsigned binary number*, magnitude of number n is stored in n bits.
- Data range for 8-bit *unsigned number is:*
  - 0 to 255 ($2^8$= 256).

## Course Contents

**Binary Arithmetic:**

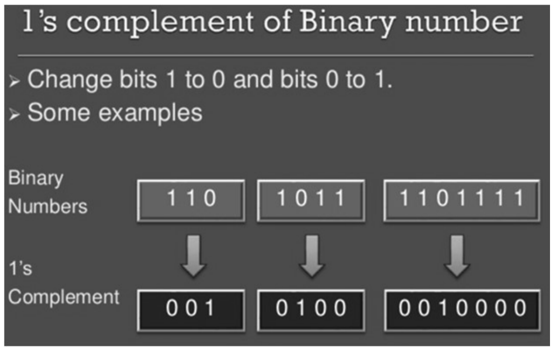


### Complement of Binary Numbers

- Used in computer for simplification of subtraction operation.
- Two types of complements
  - 1's complement, and
  - 2's complement.

## Course Contents

**Binary Arithmetic:**



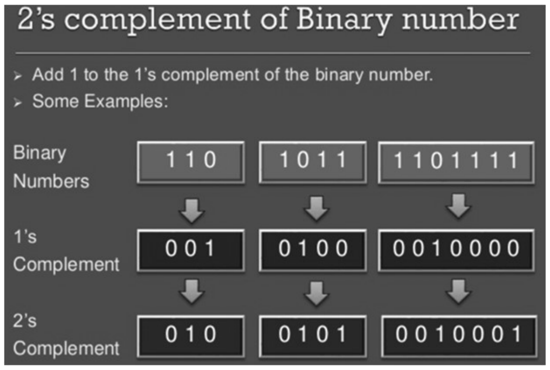
### 1's complement of Binary number

- Change bits 1 to 0 and bits 0 to 1.
- Some examples

Binary Numbers
1 1 0
1 0 1 1
1 1 0 1 1 1 1
1's Complement
0 0 1
0 1 0 0
0 0 1 0 0 0 0

## Course Contents

**Binary Arithmetic:**



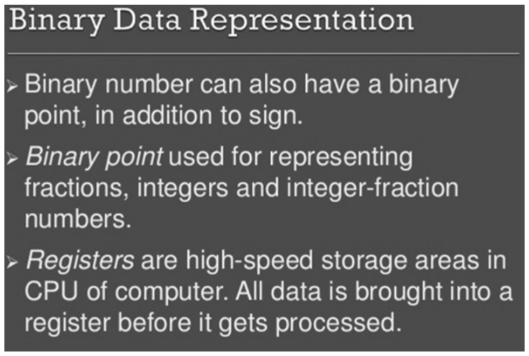
### 2's complement of Binary number

- Add 1 to the 1's complement of the binary number.
- Some Examples:

| | | | |
|---|---|---|---|
| Binary Numbers | 1 1 0 | 1 0 1 1 | 1 1 0 1 1 1 1 |
| 1's Complement | 0 0 1 | 0 1 0 0 | 0 0 1 0 0 0 0 |
| 2's Complement | 0 1 0 | 0 1 0 1 | 0 0 1 0 0 0 1 |

## Course Contents

**Binary Arithmetic:**

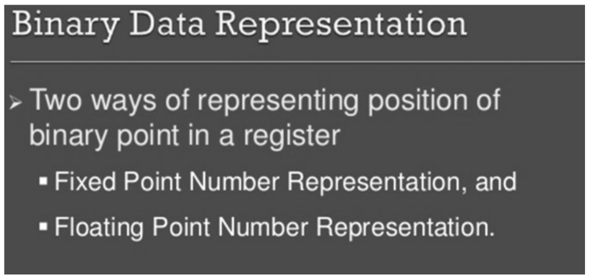


### Binary Data Representation

- Binary number can also have a binary point, in addition to sign.
- *Binary point* used for representing fractions, integers and integer-fraction numbers.
- *Registers* are high-speed storage areas in CPU of computer. All data is brought into a register before it gets processed.

## Course Contents

**Binary Arithmetic:**

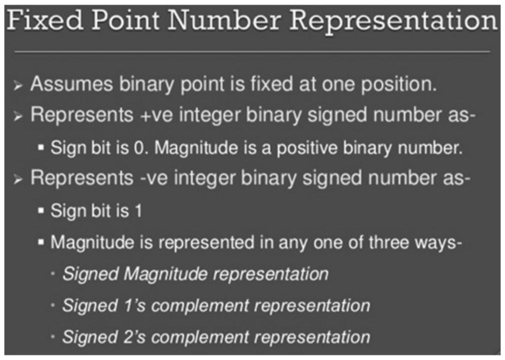


### Binary Data Representation

- Two ways of representing position of binary point in a register
  - Fixed Point Number Representation, and
  - Floating Point Number Representation.

## Course Contents

**Binary Arithmetic:**

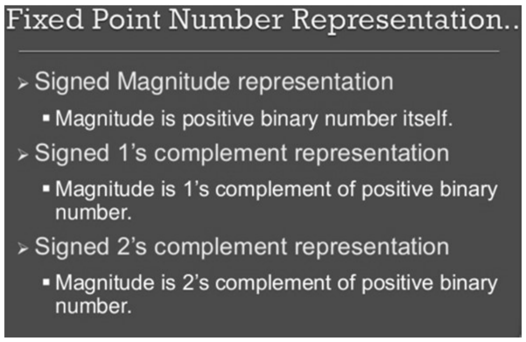


### Fixed Point Number Representation

- Assumes binary point is fixed at one position.
- Represents +ve integer binary signed number as-
  - Sign bit is 0. Magnitude is a positive binary number.
- Represents -ve integer binary signed number as-
  - Sign bit is 1
  - Magnitude is represented in any one of three ways-
    - *Signed Magnitude representation*
    - *Signed 1's complement representation*
    - *Signed 2's complement representation*

## Course Contents

**Binary Arithmetic:**

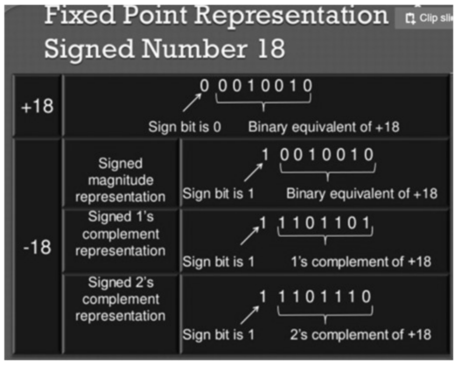


### Fixed Point Number Representation..

- Signed Magnitude representation
  - Magnitude is positive binary number itself.
- Signed 1's complement representation
  - Magnitude is 1's complement of positive binary number.
- Signed 2's complement representation
  - Magnitude is 2's complement of positive binary number.

## Course Contents

**Binary Arithmetic:**

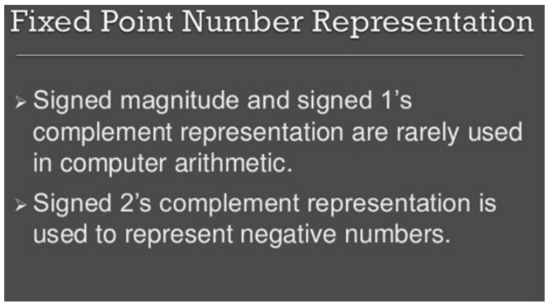


### Fixed Point Representation Signed Number 18

| | | |
|---|---|---|
| +18 | | $\underset{\text{Sign bit is 0}}{0}\ \underbrace{0010010}_{\text{Binary equivalent of +18}}$ |
| -18 | Signed magnitude representation | $\underset{\text{Sign bit is 1}}{1}\ \underbrace{0010010}_{\text{Binary equivalent of +18}}$ |
| | Signed 1's complement representation | $\underset{\text{Sign bit is 1}}{1}\ \underbrace{1101101}_{\text{1's complement of +18}}$ |
| | Signed 2's complement representation | $\underset{\text{Sign bit is 1}}{1}\ \underbrace{1101110}_{\text{2's complement of +18}}$ |

## Course Contents

**Binary Arithmetic:**

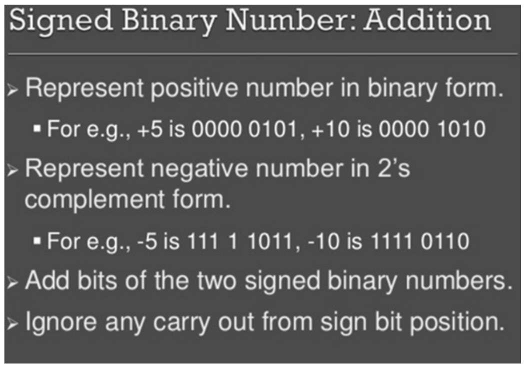


### Fixed Point Number Representation

- Signed magnitude and signed 1's complement representation are rarely used in computer arithmetic.
- Signed 2's complement representation is used to represent negative numbers.

## Course Contents

**Binary Arithmetic:**

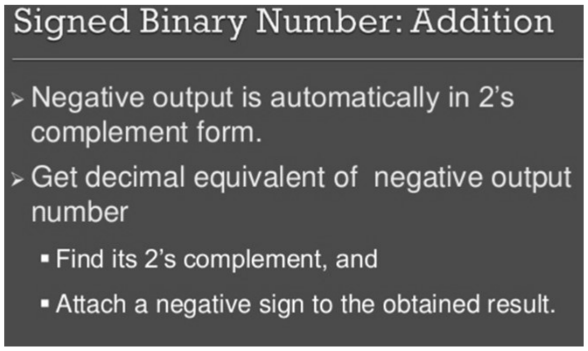


### Signed Binary Number: Addition

- Represent positive number in binary form.
  - For e.g., +5 is 0000 0101, +10 is 0000 1010
- Represent negative number in 2's complement form.
  - For e.g., -5 is 111 1 1011, -10 is 1111 0110
- Add bits of the two signed binary numbers.
- Ignore any carry out from sign bit position.

## Course Contents

**Binary Arithmetic:**

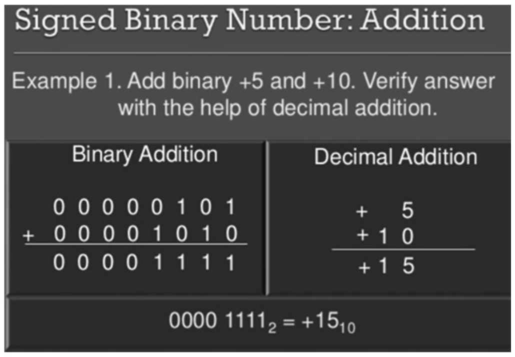


### Signed Binary Number: Addition

- Negative output is automatically in 2's complement form.
- Get decimal equivalent of negative output number
  - Find its 2's complement, and
  - Attach a negative sign to the obtained result.

## Course Contents

**Binary Arithmetic:**



### Signed Binary Number: Addition

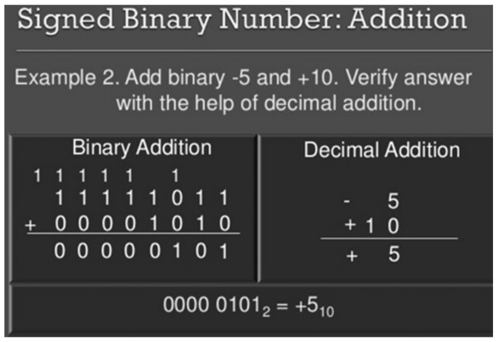
Example 1. Add binary +5 and +10. Verify answer with the help of decimal addition.

| Binary Addition | Decimal Addition |
|---|---|
| 0 0 0 0 0 1 0 1 | + 5 |
| + 0 0 0 0 1 0 1 0 | + 1 0 |
| 0 0 0 0 1 1 1 1 | + 1 5 |

$0000\ 1111_2 = +15_{10}$

## Course Contents

**Binary Arithmetic:**



### Signed Binary Number: Addition

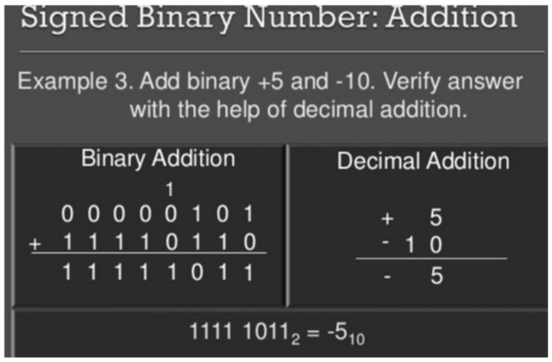
Example 2. Add binary -5 and +10. Verify answer with the help of decimal addition.

| Binary Addition | Decimal Addition |
|---|---|
| 1 1 1 1 1   1 | |
| 1 1 1 1 1 0 1 1 | - 5 |
| + 0 0 0 0 1 0 1 0 | + 1 0 |
| 0 0 0 0 0 1 0 1 | + 5 |

$0000\ 0101_2 = +5_{10}$

## Course Contents

**Binary Arithmetic:**



### Signed Binary Number: Addition

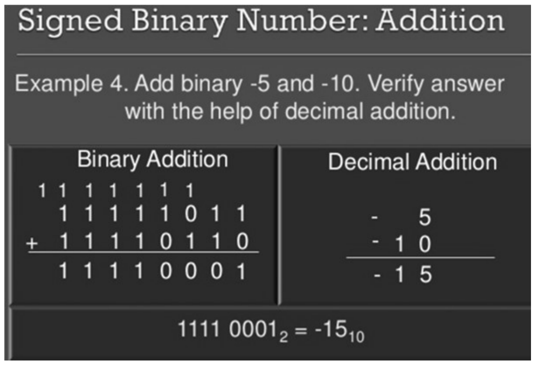
Example 3. Add binary +5 and -10. Verify answer with the help of decimal addition.

| Binary Addition | Decimal Addition |
|---|---|
| 1 | |
| 0 0 0 0 0 1 0 1 | + 5 |
| + 1 1 1 1 0 1 1 0 | - 1 0 |
| 1 1 1 1 1 0 1 1 | - 5 |

$1111\ 1011_2 = -5_{10}$

## Course Contents

**Binary Arithmetic:**



### Signed Binary Number: Addition

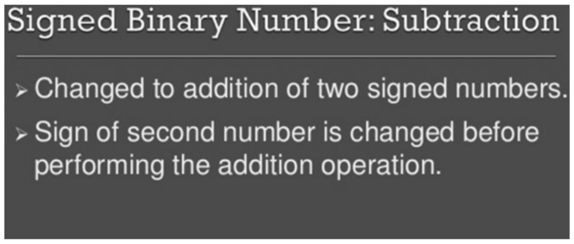
Example 4. Add binary -5 and -10. Verify answer with the help of decimal addition.

| Binary Addition | Decimal Addition |
|---|---|
| 1 1 1 1 1 1 1 (carry)<br>1 1 1 1 1 0 1 1<br>+ 1 1 1 1 0 1 1 0<br>1 1 1 1 0 0 0 1 | - 5<br>- 1 0<br>- 1 5 |

$1111\ 0001_2 = -15_{10}$

## Course Contents

**Binary Arithmetic:**

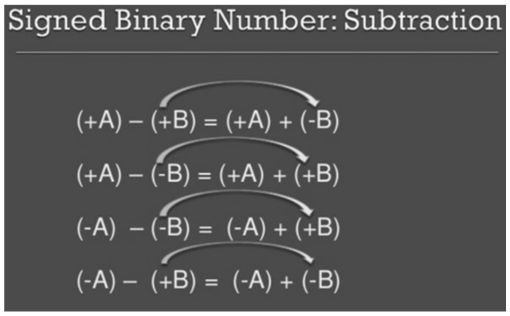


### Signed Binary Number: Subtraction

- Changed to addition of two signed numbers.
- Sign of second number is changed before performing the addition operation.

## Course Contents

**Binary Arithmetic:**



### Signed Binary Number: Subtraction

(+A) – (+B) = (+A) + (-B)

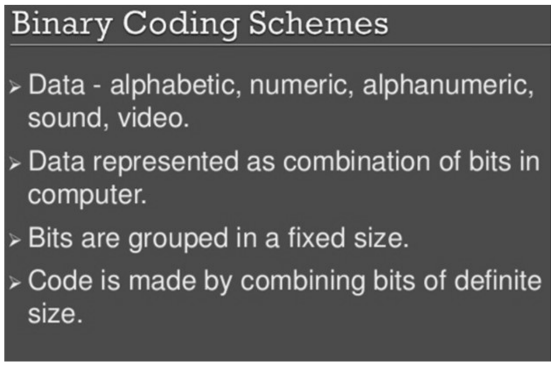
(+A) – (-B) = (+A) + (+B)

(-A) – (-B) = (-A) + (+B)

(-A) – (+B) = (-A) + (-B)

## Course Contents

**Binary Coding Schemes:**

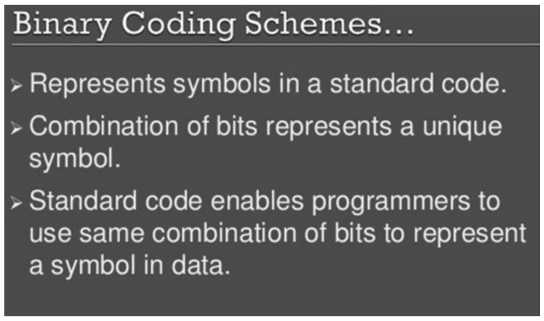


### Binary Coding Schemes

- Data - alphabetic, numeric, alphanumeric, sound, video.
- Data represented as combination of bits in computer.
- Bits are grouped in a fixed size.
- Code is made by combining bits of definite size.

## Course Contents

**Binary Coding Schemes:**

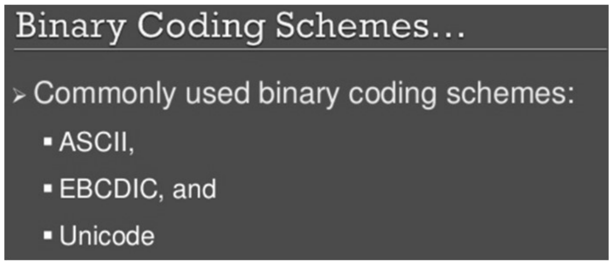


### Binary Coding Schemes…

- Represents symbols in a standard code.
- Combination of bits represents a unique symbol.
- Standard code enables programmers to use same combination of bits to represent a symbol in data.

## Course Contents

**Binary Coding Schemes:**

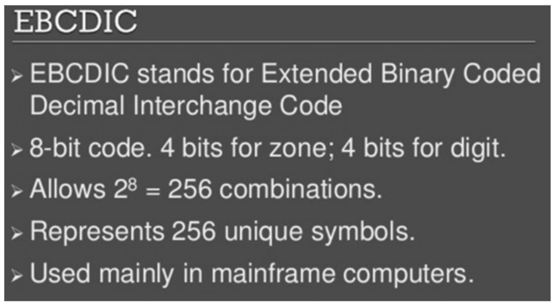


### Binary Coding Schemes…

- Commonly used binary coding schemes:
  - ASCII,
  - EBCDIC, and
  - Unicode

## Course Contents

**Binary Coding Schemes:**

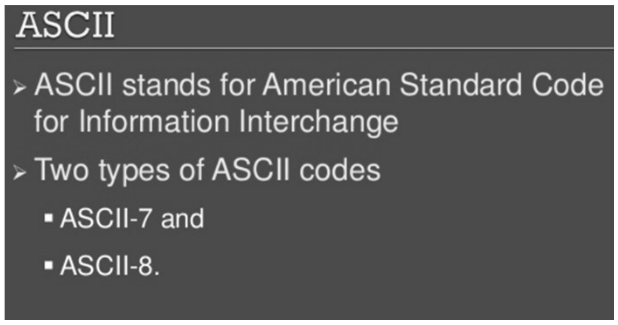


### EBCDIC

- EBCDIC stands for Extended Binary Coded Decimal Interchange Code
- 8-bit code. 4 bits for zone; 4 bits for digit.
- Allows $2^8$ = 256 combinations.
- Represents 256 unique symbols.
- Used mainly in mainframe computers.

## Course Contents

**Binary Coding Schemes:**

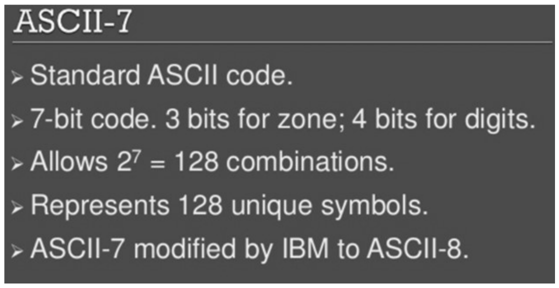


### ASCII

- ASCII stands for American Standard Code for Information Interchange
- Two types of ASCII codes
  - ASCII-7 and
  - ASCII-8.

## Course Contents

**Binary Coding Schemes:**



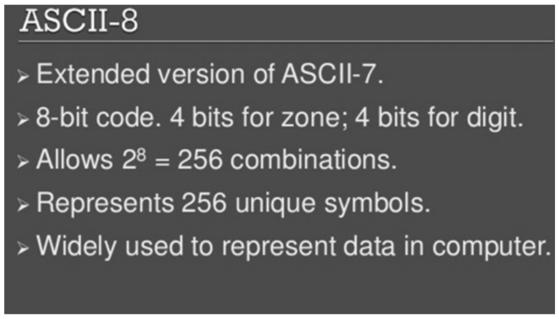
### ASCII-7

- Standard ASCII code.
- 7-bit code. 3 bits for zone; 4 bits for digits.
- Allows $2^7$ = 128 combinations.
- Represents 128 unique symbols.
- ASCII-7 modified by IBM to ASCII-8.

The **zone** used by **ASCII** for alphabets is 0100. For e.g. A is represented as 0100(**zone**)0001(**digit**). The hex equivalent is 41 for A.
Character A = 0100 0001; Character B = 0100 0010

## Course Contents

**Binary Coding Schemes:**

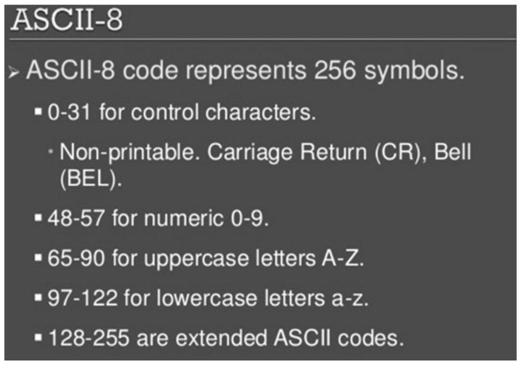


### ASCII-8

- Extended version of ASCII-7.
- 8-bit code. 4 bits for zone; 4 bits for digit.
- Allows $2^8$ = 256 combinations.
- Represents 256 unique symbols.
- Widely used to represent data in computer.

## Course Contents

**Binary Coding Schemes:**

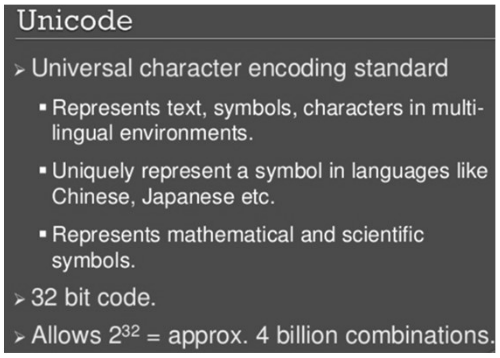


### ASCII-8

- ASCII-8 code represents 256 symbols.
  - 0-31 for control characters.
    - Non-printable. Carriage Return (CR), Bell (BEL).
  - 48-57 for numeric 0-9.
  - 65-90 for uppercase letters A-Z.
  - 97-122 for lowercase letters a-z.
  - 128-255 are extended ASCII codes.

## Course Contents

**Binary Coding Schemes:**

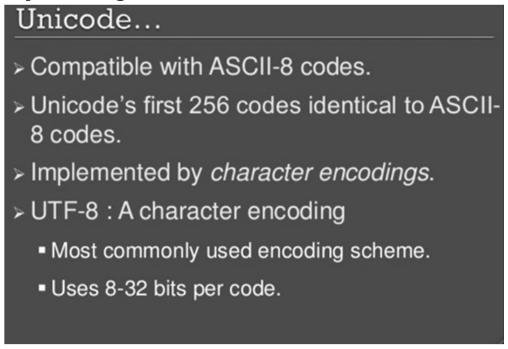


### Unicode

- Universal character encoding standard
  - Represents text, symbols, characters in multi-lingual environments.
  - Uniquely represent a symbol in languages like Chinese, Japanese etc.
  - Represents mathematical and scientific symbols.
- 32 bit code.
- Allows $2^{32}$ = approx. 4 billion combinations.

## Course Contents

**Binary Coding Schemes:**



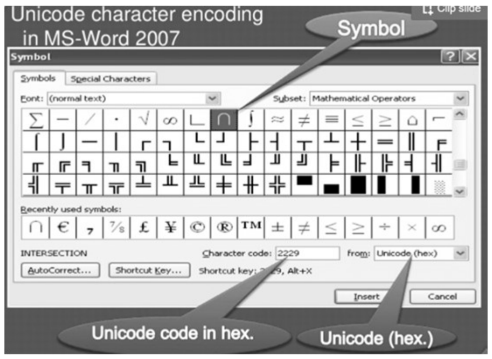
Unicode…

- Compatible with ASCII-8 codes.
- Unicode's first 256 codes identical to ASCII-8 codes.
- Implemented by *character encodings*.
- UTF-8 : A character encoding
  - Most commonly used encoding scheme.
  - Uses 8-32 bits per code.

UTF: Unicode Transformation Format

## Course Contents

**Binary Coding Schemes:**



Font: (normal text)
Subset: Mathematical Operators
INTERSECTION
Character code: 2229
from: Unicode (hex)
Shortcut key: 2229, Alt+X

## Course Contents

**Binary Coding Schemes:**

**UTF encodings include:**

- UTF-1, a retired predecessor of UTF-8, maximizes compatibility with ISO 2022, no longer part of The Unicode Standard;
- UTF-7, a 7-bit encoding sometimes used in e-mail, often considered obsolete (not part of The Unicode Standard, but only documented as an informational RFC, i.e., not on the Internet Standards Track);
- UTF-8, an 8-bit variable-width encoding which maximizes compatibility with ASCII;
- UTF-EBCDIC, an 8-bit variable-width encoding similar to UTF-8, but designed for compatibility with EBCDIC (not part of The Unicode Standard);
- UTF-16, a 16-bit, variable-width encoding;
- UTF-32, a 32-bit, fixed-width encoding.

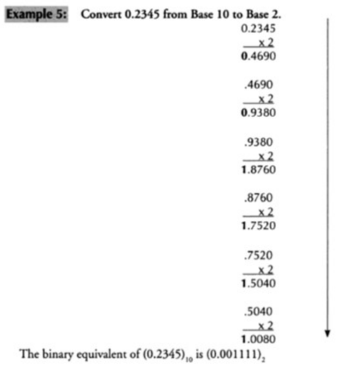
## Course Contents

**Unit-05:Data Representation (6 Hrs.)**

- **Signed and Unsigned Numbers;**
- **Binary Data Representation;**
- **Binary Coding Schemes;**
- **Logic Gates**

## Course Contents

**Converting Decimal Fraction to Binary, Octal and Hexa:**



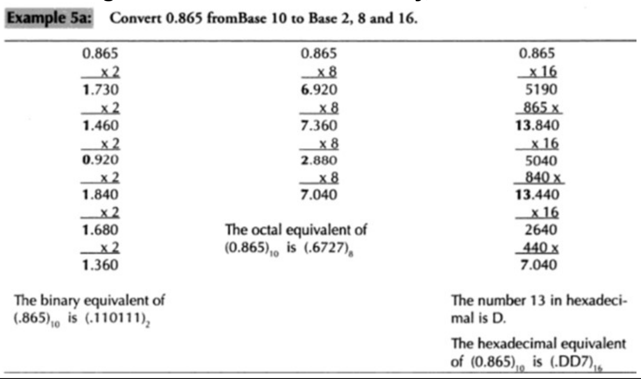
**Example 5:** Convert 0.2345 from Base 10 to Base 2.

```
 0.2345
    x 2
 0.4690

  .4690
    x 2
 0.9380

  .9380
    x 2
 1.8760

  .8760
    x 2
 1.7520

  .7520
    x 2
 1.5040

  .5040
    x 2
 1.0080
```

The binary equivalent of $(0.2345)_{10}$ is $(0.001111)_2$

## Course Contents

**Converting Decimal Fraction to Binary, Octal and Hexa:**



**Example 5a:** Convert 0.865 fromBase 10 to Base 2, 8 and 16.

```
0.865        0.865        0.865
  x 2          x 8         x 16
1.730        6.920         5190
  x 2          x 8         865 x
1.460        7.360       13.840
  x 2          x 8         x 16
0.920        2.880         5040
  x 2          x 8         840 x
1.840        7.040       13.440
  x 2                      x 16
1.680                      2640
  x 2                      440 x
1.360                      7.040
```

The binary equivalent of $(.865)_{10}$ is $(.110111)_2$

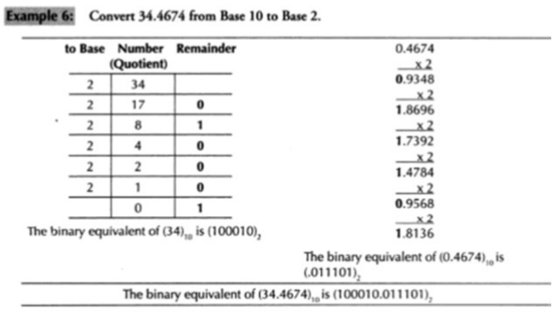
The octal equivalent of $(0.865)_{10}$ is $(.6727)_8$

The number 13 in hexadecimal is D.

The hexadecimal equivalent of $(0.865)_{10}$ is $(.DD7)_{16}$

## Course Contents

**Converting Decimal Integer, Fraction to Binary, Octal and Hexa:**



**Example 6:** Convert 34.4674 from Base 10 to Base 2.

| to Base | Number (Quotient) | Remainder |
|---|---|---|
| 2 | 34 | |
| 2 | 17 | **0** |
| 2 | 8 | **1** |
| 2 | 4 | **0** |
| 2 | 2 | **0** |
| 2 | 1 | **0** |
| | 0 | **1** |

The binary equivalent of $(34)_{10}$ is $(100010)_2$

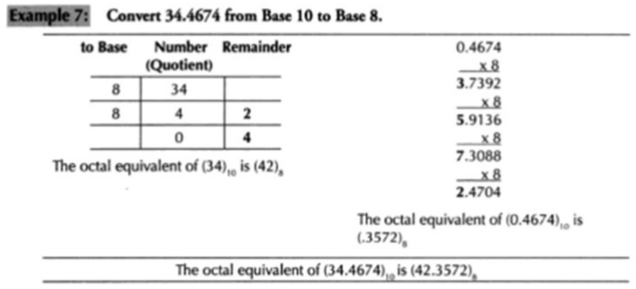
```
 0.4674
    x 2
 0.9348
    x 2
 1.8696
    x 2
 1.7392
    x 2
 1.4784
    x 2
 0.9568
    x 2
 1.8136
```

The binary equivalent of $(0.4674)_{10}$ is $(.011101)_2$

The binary equivalent of $(34.4674)_{10}$ is $(100010.011101)_2$

## Course Contents

**Converting Decimal Integer, Fraction to Binary, Octal and Hexa:**



**Example 7:** Convert 34.4674 from Base 10 to Base 8.

| to Base | Number (Quotient) | Remainder |
|---|---|---|
| 8 | 34 | |
| 8 | 4 | **2** |
| | 0 | **4** |

The octal equivalent of $(34)_{10}$ is $(42)_8$

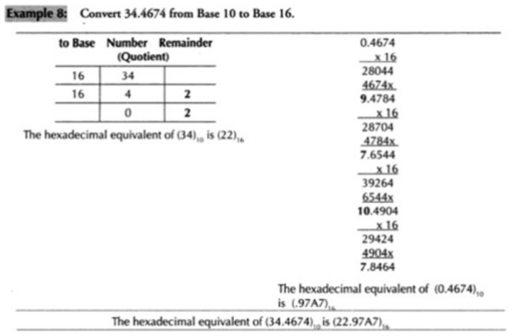
```
 0.4674
    x 8
 3.7392
    x 8
 5.9136
    x 8
 7.3088
    x 8
 2.4704
```

The octal equivalent of $(0.4674)_{10}$ is $(.3572)_8$

The octal equivalent of $(34.4674)_{10}$ is $(42.3572)_8$

## Course Contents

**Converting Decimal Integer, Fraction to Binary, Octal and Hexa:**



**Example 8:** Convert 34.4674 from Base 10 to Base 16.

| to Base | Number (Quotient) | Remainder |
|---|---|---|
| 16 | 34 | |
| 16 | 4 | **2** |
| | 0 | **2** |

The hexadecimal equivalent of $(34)_{10}$ is $(22)_{16}$

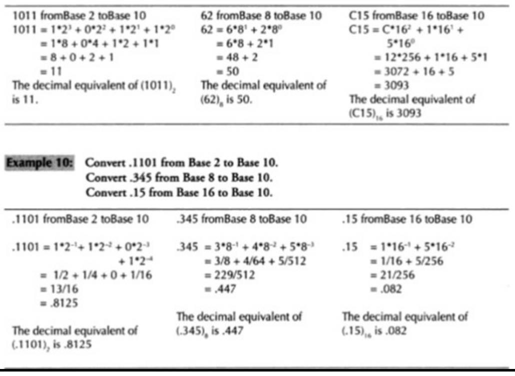
```
  0.4674
    x 16
   28044
   4674x
  9.4784
    x 16
   28704
   4784x
  7.6544
    x 16
   39264
   6544x
 10.4904
    x 16
   29424
   4904x
  7.8464
```

The hexadecimal equivalent of $(0.4674)_{10}$ is $(.97A7)_{16}$

The hexadecimal equivalent of $(34.4674)_{10}$ is $(22.97A7)_{16}$

## Course Contents

**Converting Fraction Binary, Octal and Hexa to Decimal**



1011 fromBase 2 toBase 10

$1011 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$
$= 1*8 + 0*4 + 1*2 + 1*1$
$= 8 + 0 + 2 + 1$
$= 11$

The decimal equivalent of $(1011)_2$ is 11.

62 fromBase 8 toBase 10

$62 = 6*8^1 + 2*8^0$
$= 6*8 + 2*1$
$= 48 + 2$
$= 50$

The decimal equivalent of $(62)_8$ is 50.

C15 fromBase 16 toBase 10

$C15 = C*16^2 + 1*16^1 + 5*16^0$
$= 12*256 + 1*16 + 5*1$
$= 3072 + 16 + 5$
$= 3093$

The decimal equivalent of $(C15)_{16}$ is 3093

**Example 10:** Convert .1101 from Base 2 to Base 10.
Convert .345 from Base 8 to Base 10.
Convert .15 from Base 16 to Base 10.

.1101 fromBase 2 toBase 10

$.1101 = 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4}$
$= 1/2 + 1/4 + 0 + 1/16$
$= 13/16$
$= .8125$

The decimal equivalent of $(.1101)_2$ is .8125

.345 fromBase 8 toBase 10

$.345 = 3*8^{-1} + 4*8^{-2} + 5*8^{-3}$
$= 3/8 + 4/64 + 5/512$
$= 229/512$
$= .447$

The decimal equivalent of $(.345)_8$ is .447

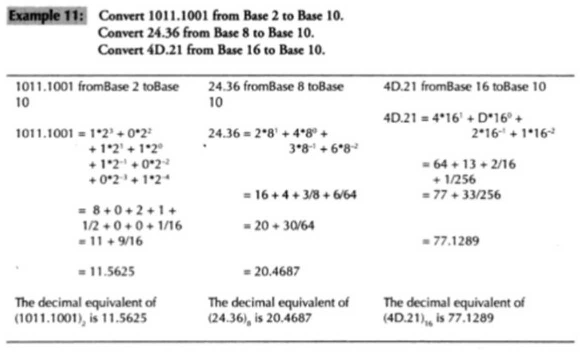
.15 fromBase 16 toBase 10

$.15 = 1*16^{-1} + 5*16^{-2}$
$= 1/16 + 5/256$
$= 21/256$
$= .082$

The decimal equivalent of $(.15)_{16}$ is .082

## Course Contents

**Converting Fraction Binary, Octal and Hexa to Decimal**



**Example 11:** Convert 1011.1001 from Base 2 to Base 10.
Convert 24.36 from Base 8 to Base 10.
Convert 4D.21 from Base 16 to Base 10.

1011.1001 fromBase 2 toBase 10

$1011.1001 = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 0*2^{-2} + 0*2^{-3} + 1*2^{-4}$
$= 8 + 0 + 2 + 1 + 1/2 + 0 + 0 + 1/16$
$= 11 + 9/16$
$= 11.5625$

The decimal equivalent of $(1011.1001)_2$ is 11.5625

24.36 fromBase 8 toBase 10

$24.36 = 2*8^1 + 4*8^0 + 3*8^{-1} + 6*8^{-2}$
$= 16 + 4 + 3/8 + 6/64$
$= 20 + 30/64$
$= 20.4687$

The decimal equivalent of $(24.36)_8$ is 20.4687

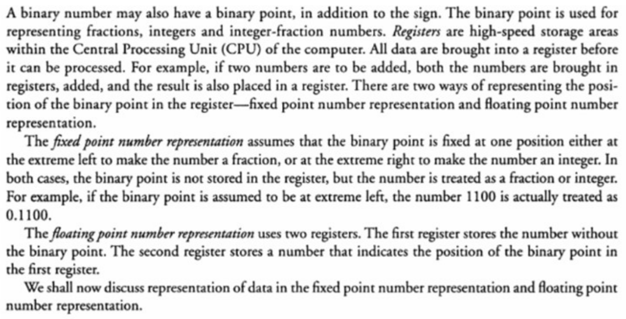
4D.21 fromBase 16 toBase 10

$4D.21 = 4*16^1 + D*16^0 + 2*16^{-1} + 1*16^{-2}$
$= 64 + 13 + 2/16 + 1/256$
$= 77 + 33/256$
$= 77.1289$

The decimal equivalent of $(4D.21)_{16}$ is 77.1289

## Course Contents

- **Binary Data Representation;**



A binary number may also have a binary point, in addition to the sign. The binary point is used for representing fractions, integers and integer-fraction numbers. *Registers* are high-speed storage areas within the Central Processing Unit (CPU) of the computer. All data are brought into a register before it can be processed. For example, if two numbers are to be added, both the numbers are brought in registers, added, and the result is also placed in a register. There are two ways of representing the position of the binary point in the register—fixed point number representation and floating point number representation.

The *fixed point number representation* assumes that the binary point is fixed at one position either at the extreme left to make the number a fraction, or at the extreme right to make the number an integer. In both cases, the binary point is not stored in the register, but the number is treated as a fraction or integer. For example, if the binary point is assumed to be at extreme left, the number 1100 is actually treated as 0.1100.

The *floating point number representation* uses two registers. The first register stores the number without the binary point. The second register stores a number that indicates the position of the binary point in the first register.

We shall now discuss representation of data in the fixed point number representation and floating point number representation.