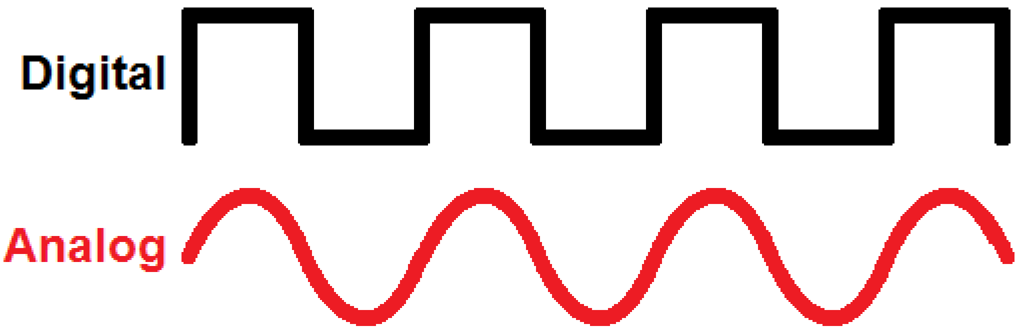
1.0 Introduction

What is Digital?

Electronic signals can be divided into two categories: analog and digital. Analog signals can take any shape and represent an infinite number of possible values. Digital signals have a very defined, discrete set of possible values -- usually only *two*.



Many electronic systems will use a combination of analog and digital circuits, but at the heart of most computers and other every day electronics are discrete, digital circuits.

What is Digital Logic?

We humans are (mostly) logical beings. When we make decisions and act upon them, logic is working in the background. Logic is the process of evaluating one or more inputs, weighing them against any number of outcomes, and deciding a path to follow. Just like we apply logic to make all of our decisions, computers use digital logic circuits to make theirs. They use a set of standard **logic gates** to help propagate a decision.

Inputs and Outputs

A digital logic circuit uses digital inputs to make logical decisions and produce digital outputs. Every logic circuit requires at least one input, before it can produce any kind of output.

Digital logic inputs and outputs are usually binary. In other words they can only be one of two possible values.

There are a number of ways to **represent binary values**: 1/0 is the least verbose and most common way. However, you may also see them in a boolean representation like TRUE/FALSE or HIGH/LOW. When the values are represented at a hardware level, they might be given actual voltage levels: 0V [Volts] is a 0, while a higher voltage - usually 3V or 5V - is used to represent a 1.

1	0
HIGH	LOW
True	False
5V (Volts)	0V

1.1 Number System:

A number is a mathematical object used to count, measure, and label. Numbers are represented by a string of digital symbols. A number system of base r is a system that uses distinct symbols for r digits. That is in a positional base$r$ numeral system r basic symbols (or digits) corresponding to the first r natural numbers including zero are used. To generate the rest of the numerals, the position of

the symbol in the figure is used. The symbol in the last position has its own value, and as it moves to the left its value is multiplied by r. There are
four systems of arithmetic used in digital system. These systems are Decimal, Binary, Hexadecimal and Octal.

System	Base	Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

.

1.2 Decimal Number System:

The Decimal number system has a base ten. This system uses ten distinct digits 0 1 2 3 4 5 6 78 9 to form any number. Each digit can be used individually or they can be grouped to form anumeric value. Each of decimal digits, 0 through 9, has a place value or weight depending on its position. The weights are units, tens, hundreds, thousands and so on. The same can be expressed as the powers of its base as 10^0, 10^1, 10^2, 10^3 $\cdots$ $etc$for the integer part and 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4} $\cdots$ etc for the fractional part. 10^0, 10^1, 10^2, 10^3 $\cdots$ etcrepresents the units, tens, hundreds, thousands etc. and the quantities 10^{-1}, 10^{-2}, 10^{-3}, $\cdots$ etc represents one tenth, one hundredth, one thousandth etc. The integer part and fractional parts areseparated by a decimal point. The position weights in decimal system is given as

$\cdots$	10^3	10^2	10^1	10^0	$\cdot$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	$\cdots$

Example:

(i) $7693 = 7 \times 10^3 + 6 \times 10^2 + 9 \times 10^1 + 3 \times 10^0$

$= 7 \times 1000 + 6 \times 100 + 9 \times 10 + 3 \times 1$

$= 7000 + 600 + 90 + 3$

(ii) $1936.46 = 1 \times 10^3 + 9 \times 10^2 + 3 \times 10^1 + 6 \times 10^0 + 4 \times 10^{-1} + 6 \times 10^{-2}$

$= 1000 + 900 + 30 + 6 + 0.4 + 0.06$

1.3 Binary Number System:

The base of the binary number system is two. It uses the digits0 and 1 only. The two digits 0 and 1 are called a bit. The place value of each position can be expressed in terms of powers of 2 like 2^0, 2^1, 2^2, etc for integer part and 2^{-1}, 2^{-2}, 2^{-3}, etc for the fractional part. A binary point separates the integer and fractional part. The position weights in the binary is given as

$\cdots$	2^3	2^2	2^1	2^0	$\cdot$	2^{-1}	2^{-2}	2^{-3}	2^{-4}	$\cdots$

Some common terms in Binary number system:

- Bit is an abbreviation of the term ‘binary digit’ and is the smallest unit of information. It is either ‘0’ or ‘1’.
- A byte is a string of eight bits. The byte is the basic unit of data operated upon as a single unit in computers.
- A computer word is again a string of bits whose size, called the ‘word length’ or ‘word size’, is fixed for a specified computer, although it may vary from computer to computer. The word length may equal one byte, two bytes, four bytes or be even larger.

1.4 Octal Number System:

The base of the octal number system is eight. It uses eight digits 0 1 2 3 4 5 6 and 7 to form a number. The place value of each position can be expressed in terms of powers of 8 like8^0, 8^1, 8^2, etc for integer part and 8^{-1}, 8^{-2}, 8^{-3}, etc for the fractional part. An octal point separates the integer and fractional part. Sets of 3-bit binary numbers can be represented by octal numbers (000, 001, 010,011, 100, 101,110,111) and this can be conveniently be used for entering data in the computer. The position weights in the octal system is given as

$\cdots$	8^3	8^2	8^1	8^0	$\cdot$	8^{-1}	8^{-2}	8^{-3}	8^{-4}	$\cdots$

1.5 Hexadecimal Number System:

The Hexadecimal number system has a base of 16. It has 16 distinct digit symbols. It uses the digits 0 1 2 3 4 5 6 7 8 9 plus the letters $A\ B\ C\ D\ E$ *and* F. Any hexadecimal digit can be represented by a group of four-bit binary sequence. That is the Hexadecimal numbers are represented by sets of 4-bit binary sequence (0000, 0001,0010, 0011, 0100,0101,0110, 0111,1000,1001,1010,1011,1100,1101,1110,1111). The position weight in the hexadecimal number system is given as

⋯	16^3	16^2	16^1	16^0	·	16^{-1}	16^{-2}	16^{-3}	16^{-4}	⋯

Number System

Decimal (Base 10)	**Binary (Base 2)**	**Octal (Base 8)**	**Hexadecimal (Base 16)**
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	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

1.6 Decimal to Binary Conversion:

Decimal number can be converted to binary by repeatedly dividing the number by 2 for integer part and collecting the reminders. The remainders can be written in the reverse order(from bottom to top) to get binary result. For fractional part, it has to be multiplied by 2 successively and collecting the carries, to write from top to bottom. The multiplication is repeated till the fractional part becomes zero or the required number of significant bit is obtained.

1. Convert $(19)_{10}$ into its Binary equivalent

```
2 | 19
2 | 9   —— 1 LSB (lowest Significant
2 | 4   —— Bit)1
2 | 2   —— 0
    1   —— 0
MSB (Most Significant Bit)
```

1 0 0 1 1
MSB LSB

$(19)_{10} = (10011)2$

2. Convert$(0.625)_{10}$ into its Binary equivalent

$0.625 \times 2 = 1.250$ *carry is* 1 (*MSB*)

$0.250 \times 2 = 0.500$ *carry is* 0

$0.500 \times 2 = 1.000$ *carry is* 1 (*LSB*)

1 0 1
MSB LSB

$(0.625)_{10} = (0.101)2$

3. Convert $(107.6875)_{10}$ to its equivalent Binary

```
2 | 107
2 | 53  —— 1 LSB (lowest Significant
Bit)2 | 26  —— 1
2 | 13  —— 0
2 | 6   —— 1
2 | 3   —— 0
    1   —— 1
MSB (Most Significant Bit)
```

1 1 0 1 0 1 1
MSB LSB

$0.6875 \times 2 = 1.3750$ *carry is* 1 (*MSB*)

$0.3750 \times 2 = 0.7500$ *carry is* 0

$0.7500 \times 2 = 1.500$ *carry is* 1

$0.5000 \times 2 = 1.000$ *carry is* 1 (*LSB*)

1 0 1 1
MSB LSB

$$(107.6875)_{10} = (1101011.1011)_2$$

1.7 Binary to Decimal Conversion:

A binary number can be converted into decimal number by adding the products ofeach bit and its corresponding weight.

Example:

(i) $(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$$= 4 + 0 + 1$$

$$= (5)_{10}$$

(ii) $(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$$= 16 + 0 + 0 + 2 + 1$$

$$= (19)_{10}$$

(iii) $(0.101)_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

$$= 1 \times 0.5 + 0 + 1 \times 0.125$$

$$= 0.5 + 0 + 0.125$$

$$= (0.625)_{10}$$

(iv) $(1101011.1011)_2$

$(1101011)_2 = 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^1$

$$= 64 + 32 + 0 + 8 + 0 + 2 + 1$$

$$= (107)_{10}$$

$(0.1011) \quad = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$

$$= 0.5 + 0 + 0.125 + 0.0625$$

$$= (0.6875)_{10}$$

$$\therefore (1101011.1011)_2 = (107.6875)_{10}$$

1.8 Hexadecimal to Decimal:

A hexadecimal number can be converted into decimal number by adding the products ofeach digit and its corresponding weight. The weights are power of 16.

Example:

1. Convert hexadecimal $(D5)16$ to decimal

$$(D5)16 = (13 \times 16^1 + 5 \times 16^0) \Longrightarrow (D)16 = (13)10$$

$$= 13 \times 16 + 5 \times 1$$

$$= 208 + 5$$

$$= (213)10$$

$$ie., (D5)16 = (213)10$$

2. Convert hexadecimal $(3FC.8)16$ to decimal

$$(3FC.8)16 = (3 \times 16^2 + 15 \times 16^1 + 12 \times 16^0 + 8 \times 16^{-1})$$

$$\Longrightarrow (F)16 = (15)10\&(C)16 = (12)10$$

$$= 3 \times 256 + 15 \times 16 + 12 \times 1 + 8 \times \frac{1}{16}$$

$$= 768 + 240 + 12 + 0.5$$

$$= (1020.5)10$$

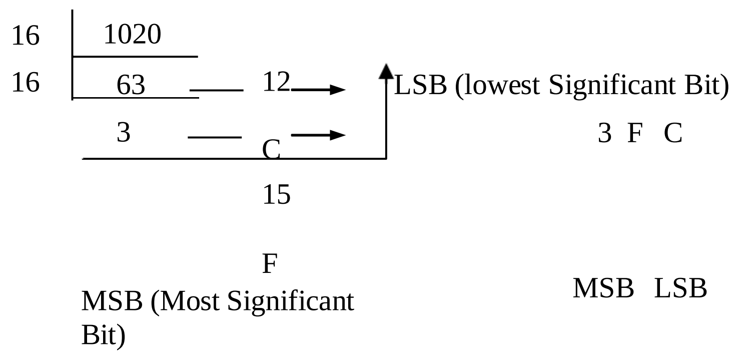
$$ie., (3FC.8)16 = (1020.5)10$$

1.9 Decimal to Hexadecimal:

Decimal number can be converted to hexadecimal by repeatedly dividing the number by 16 for integer part and collecting the reminders. The remainders can be written in the reverse order (from bottom to top) to get hex result. For fractional part, it has to be multiplied by 16successively and collecting the carries, to write from top to bottom. The multiplication is repeated till the fractional part becomes zero or the required numbers of significant digits are obtained.

Example:

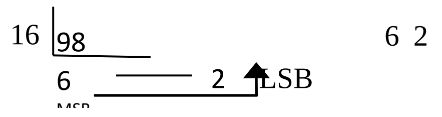
1. Convert $(1020)_{10}$ to hexadecimal



16 | 1020
16 | 63 — 12 → LSB (lowest Significant Bit)
3 — C →
15
F
MSB (Most Significant Bit)
3 F C
MSB LSB

$$(1020)10 = (3FC)16$$

2. Convert (98.625)10 to hexadecimal



$$0.625 \times 16 = 10.000 \ \ carry\ is\ \ 10 \ \Longrightarrow A$$

$$(98.625)10 = (62.A)_{16}$$

1.10 Hexadecimal to Binary:

Hexadecimal numbers can be converted into binary numbers by converting each hexadecimal digit to its 4-bit binary equivalent

1. Convert $(25)H$ to Binary

   $(25) = (\underbrace{0\,0\,1\,0}_{2}\;\underbrace{0\,1\,0\,1}_{5})_2$

2. Convert $(3A.7)H$ to Binary

   $(3A) = (\underbrace{0\,0\,1\,1}_{3}\;\underbrace{1\,0\,1\,0}_{A})_2$

   $(.7) = (.\underbrace{0\,1\,1\,1}_{7})_2$

   $(3A.7) = (\underbrace{0\,0\,1\,1}_{3}\;\underbrace{1\,0\,1\,0}_{A} \cdot \underbrace{0\,1\,1\,1}_{7})_2$

3. Convert $(CD.E8)H$ to Binary

   $(CD) = (\underbrace{1\,1\,0\,0}_{C}\;\underbrace{1\,1\,0\,1}_{D})_2$

   $(.E8) = (.\underbrace{1\,1\,1\,0}_{E}\;\underbrace{1\,0\,0\,0}_{8})_2$

   $(CD.E8) = (\underbrace{1\,1\,0\,0}_{C}\;\underbrace{1\,1\,0\,1}_{D} \cdot \underbrace{1\,1\,1\,0}_{E}\;\underbrace{1\,0\,0\,0}_{8})_2$

1.11 Binary to Hexadecimal:

Conversion from binary to hexadecimal is easily accomplished by partitioning the binary number into groups of four binary digits, starting from the binary point to the left and to the right. It may be necessary to add zeros to the last group, if it does not end in exactly four bits. Each group of 4-bits binary must be represented by its hexadecimal equivalent.

1. $(1010 \cdot 1101)2 = (A \cdot D)$
2. $(110 \cdot 101)2 = (0110 \cdot 1010)2 = (6 \cdot A)$
3. $(1110 \cdot 11)2 = (1110 \cdot 1100)2 = (E \cdot C)$

1.12 Octal to Decimal:

An octal number can be converted into decimal number by adding the products of each digitand its corresponding weight. The weights are power of 8.

1. $(75)8 = 7 \times 8^1 + 5 \times 8^0$

   $= 56 + 5$

   $= (61)10$

2. $(45 \cdot 6)8 = 4 \times 8^1 + 5 \times 8^0 + 6 \times 8^{-1}$

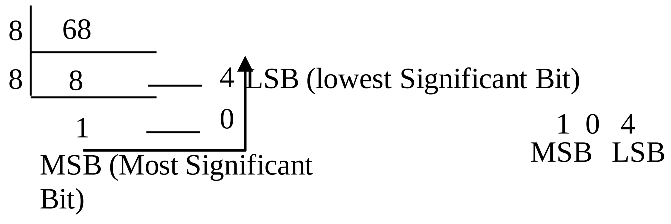
   $= 32 + 5 + 0.75$

   $= (37.75)10$

1.13 Decimal to octal:

Decimal number can be converted to octal by repeatedly dividing the number by 8 for integer part and collecting the reminders. The remainders can be written in the reverse order (from bottom to top) to get octal result. For fractional part, it has to be multiplied by 8 successively and collecting the carries, to write from top to bottom. The multiplication is repeated till the fractional part becomes zero or the required numbers of significant digits are obtained.

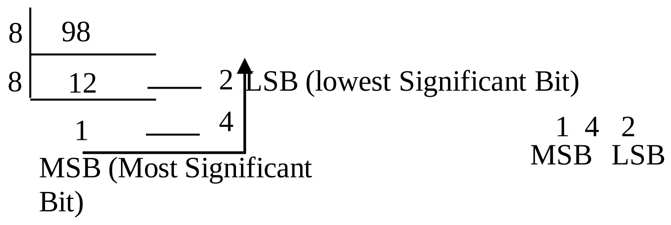
1. Convert (68)10 to octal



8 | 68
8 | 8 —— 4 LSB (lowest Significant Bit)
1 —— 0
MSB (Most Significant Bit)
1 0 4
MSB LSB

$$(68)_{10} = (104)_8$$

2. Convert $(98.625)_{10}$ to Octal



8 | 98
8 | 12 —— 2 LSB (lowest Significant Bit)
1 —— 4
MSB (Most Significant Bit)
1 4 2
MSB LSB

$$0.625 \times 8 = 5.0000 \; carry\ is \; 5$$

$$(98.625)_{10} = (142.5)_8$$

1.14 Octal to Binary:

Octal numbers can be converted into binary numbers by converting each octal digit to its 3-bit binary equivalent

1. $(27)8 = (\underbrace{0\,10}_{2}\underbrace{1\,11}_{7})_2$
2. $(135)8 = (\underbrace{0\,01}_{1}\underbrace{0\,11}_{3}\underbrace{1\,01}_{5})_2$
3. $(45.5)8 = (\underbrace{1\,00}_{4}\underbrace{1\,01}_{5}\cdot\underbrace{1\,01}_{5})_2$

1.15 Binary to Octal:

Conversion from binary to octal is the simplest procedure by grouping the binary number into groups of three binary digits, starting from the binary point to the left and to the right. It may be necessary to add zeros to the last group, if it does not end in exactly three bits. Each group of 3-bits binary must be represented by its octal equivalent.

1. (10 101)2 = (010 101)2 = (25)8
2. (101011)2 = (101 011)2 = (53)8
3. (11110.11)2 = (011 110 . 110)2 = (36.6)8
4. (11011011. 1111)2 = (011 011 011. 111 100)2 = (333.74)8

1.16 Hexadecimal to Octal:

This can be achieved by first writing down the four-bit binary equivalent of hexadecimaldigit and then partitioning it into group of 3 bits each. Finally, the three-bit octal equivalent is written down.

Example:

1. Convert $(2AB.9)H$ to octal

$Hexa\ decimal\ Number\ \rightarrow$ 2 A B . 9

↓ ↓ ↓ . ↓

$4\ bit\ \ Binary\ Number$ → 0010 1010 1011 . 1001

$3\ bi\ \ artition$ → $\underline{0\ 01\ 0\ 10\ 1\ 01\ 0\ 11}$. $\underline{1\ 00\ 1\ 00}$

↓ ↓ ↓ ↓ . ↓ ↓

$Octal\ Number$ → 1 2 5 3 . 4 4

$\therefore (2AB.9)16 = (1253.44)8$

2. Convert $(3FC.82)H$ to octal

$Hexa\ decimal\ Number\ \rightarrow$ 3 F C . 8 2

↓ ↓ ↓ . ↓ ↓

$4\ bit\ \ Binary\ Number$ → 0011 1111 1100 . 1000 0010

$3\ bi\ \ artition$ → $\underline{0\ 01\ 1\ 11\ 1\ 11}$. $\underline{1\ 00\ 0\ 00\ 1\ 00}$

$\underline{1\ 00}$

↓ ↓ ↓ ↓ . ↓ ↓ ↓

$Octal\ Number$ → 1 7 7 4 . 4 0 4

$\therefore (3FC.82)16 = (1253.44)8$

1.17 Octal to Hexadecimal:

This can be achieved by first writing down the three-bit binary equivalent of octal digitand then partitioning it into group of 4 bits each. Finally, the four-bit hexadecimal equivalent is written down.

1. Convert (16.2)8 to Hexadecimal

$$\begin{array}{lcccccc}
\textit{octal Number} & \rightarrow & 1 & & 6 & . & 2 \\
 & & \downarrow & & \downarrow & . & \downarrow \\
3\ \textit{bit Binary} & \rightarrow & 001 & & 110 & \cdot & 010 \\
 & & \downarrow & & \downarrow & . & \downarrow \\
4\ \textit{bi artition} & \rightarrow & \underbrace{0\,0\,0\,0} & & \underbrace{1\,1\,1\,0} & . & \underbrace{0\,1\,0\,0} \\
 & & \downarrow & & \downarrow & . & \downarrow \\
\textit{Hex Number} & \rightarrow & 0 & & E & . & 4
\end{array}$$

$$\therefore (16.2)8 = (0E.4)16 = (E.4)$$

A zero is added to the right most group to make it a group of 4 bits and left most zerosare dropped

2. Convert (764.352)8 to Hexadecimal

$$\begin{array}{lcccccccc}
\textit{octal Number} & \rightarrow & 7 & 6 & 4 & . & 3 & 5 & 2 \\
 & & \downarrow & \downarrow & \downarrow & . & \downarrow & \downarrow & \downarrow \\
3\ \textit{bit Binary} & \rightarrow & 111 & 110 & 100 & . & 011 & 101 & 010 \\
 & & \downarrow & \downarrow & \downarrow & . & \downarrow & \downarrow & \downarrow \\
4\ \textit{bi artition} & \rightarrow & \underbrace{0001} & \underbrace{1111} & \underbrace{0100} & . & \underbrace{0111} & \underbrace{0101} & \underbrace{0000} \\
 & & \downarrow & \downarrow & \downarrow & . & \downarrow & \downarrow & \downarrow \\
\textit{Hex Number} & \rightarrow & 1 & F & 4 & . & 7 & 5 & 0
\end{array}$$

$$\therefore (764.352)8 = (1F4.750)16 = (1F4.75)$$

1.18 Binary Arithmetic:

Arithmetic operations such as addition, subtraction, multiplication and division can be performed on binary numbers.

1.18.1 Binary addition:

The addition of two Binary numbers is very similar to addition of two decimal numbers. It is key to binary subtraction, multiplication and division. The following rules are followed while adding two binary numbers.

Augend + Addend (A) (B)	Carry (A) (B)	Sum	Result
0 + 0	0	0	0
0 + 1	0	1	1
1 + 0	01		1
1 + 1	1	0	10 ; read as 0 with a carry 1
1 + 1 +1	1	1	11 ; read as 1 with a carry 1

Example:

1. Add the binary numbers (i) 1011 and 1110 (ii) 10.001 and 11.110

(i)

Binary Number	Equivalent Decimal
1 1 1 ← Carry	
1 0 1 1	11
+ 1 1 1 0	14
Sum= 1 1 0 0 1	25

(ii)

Binary Number	Equivalent Decimal
1 ← Carry	
1 0 . 0 0 1	2 . 1 2 5
+ 1 1 . 1 1 0	+ 3 . 7 5 0
Sum= 1 0 1 . 1 1 1	5 . 8 7 5

1.18.2 Binary subtraction:

The subtraction of two Binary numbers is very similar to subtraction of two decimal numbers. Subtraction is the inverse operation of addition. The following rules are used in subtracting two binary numbers.

Minuend - Subtrahend	Difference	Borrow
0 - 0	0	0
1 - 0	1	0
1 - 1	0	0
0 - 1	1	1 read as difference 1 with borrow 1

Example:

1. Subtract the binary numbers (i) 101 from 1001 (ii) 11and 10000

(i) Binary Number Equivalent Decimal1 0 0 1 9

	Binary Number	Equivalent Decimal
	1 0 0 1	9
-	1 0 1	- 5
Difference =	1 0 0	4

(ii)

	Binary Number	Equivalent Decimal1
	0 0 0 0	16
-	1 1	- 3
Difference =	1 1 0 1	13

1.18.3 Binary Multiplication:

The multiplication of two Binary numbers is very similar to multiplication oftwo decimal numbers. The following rules are used to multiply two binary numbers.

(i) $0 \times 0 = 0$

(ii) $0 \times 1 = 0$

(iii) $1 \times 0 = 0$

(iv) $1 \times 1 = 1$

Example:

Multiply the following binary numbers (i) 1011 and 1101 and (ii) 1.01 and 10.1

(i)

	Binary Multiplication	Equivalent decimal
Multiplicand	1011	11
Multiplier	× 1101	× 13
	1011	33
	0000	11
	1011	143
	1011	
	10001111	

(ii)

```
Binary Multiplication            Equivalent decimal
Multiplicand    1.01             1.25
Multiplier    × 10.1             ×2.5
              ------             ----------
                 101                625
                000                240
                                 ----------
              101                     3025
              ------
11001                                ------
11.001                           3.025
```

1.18.4 Binary Division:

The division in Binary is exactly same as in decimal. The division by 0 is not allowed. The binary division has only two rules.

(i) 0 ÷ 1 = 0 and (ii) 1 ÷ 1 = 1

Example:

Divide the binary numbers (i) 11001 by 101 (ii) 1010 by 100

(i)

```
Binary Division          Equivalent decimal
                             5
       101                 5| 25
101 | 11001                 |  25
    | 101                       0
     00101
       101
     00000
```

(ii)

```
Binary Division          Equivalent decimal
       10.1                 2.5
100 | 1010               4| 10
    | 100                 |  8
      00100                 20
        100                  20
      00000                   0
```

1.18.5

1.19 Complements:

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. There are two types of complements, one is$r$'s complement and another is (r − 1)'s complement, where r is base of the system. For binary system the base r is 2, therefore 2's complement and 1's complement is possible.

The 1's complement of a binary number is obtained by complementing all its bits, i.e. by replacing 0s with 1s and 1s with 0s. For example, the 1's complement of $(10010110)_2$ is $(01101001)_2$. The 2's complement of a binary number is obtained by adding '1' to its 1's complement. The 2's complement of $(10010110)_2$ is $(01101010)_2$.

Number Representation in Binary

Different formats used for binary representation of both positive and negative decimal numbers include the sign-bit magnitude method, the 1's complement method and the 2's complement method.

1.8.5.1 Sign-Bit Magnitude

In the sign-bit magnitude representation of positive and negative decimal numbers, the MSB represents the 'sign', with a '0' denoting a plus sign and a '1' denoting a minus sign. The remaining bits represent the magnitude. In eight-bit representation, while MSB represents the sign, the remaining seven bits represent the magnitude. For example, the eight-bit representation of +9 would be 00001001, and that for −9 would be 10001001. An n−bit binary representation can be used to represent decimal numbers in the range of $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$. That is, eight-bit representation can be used to represent decimal numbers in the range from −127 to +127 using the sign-bit magnitude format.

1.8.5.2 1's Complement

In the 1's complement format, the positive numbers remain unchanged. The negative numbers are obtained by taking the 1's complement of the positive counterparts. For example, +9 will be represented as 00001001 in eight-bit notation, and −9 will be represented as 11110110, which is the 1's complement of 00001001. Again, n-bit notation can be used to represent numbers in the range from $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$ using the 1's complement format. The eight-bit representation of the 1's complement format can be used to represent decimal numbers in the range from −127 to +127.

1.8.5. 3 2's Complement

In the 2's complement representation of binary numbers, the MSB represents the sign, with a '0' used for a plus sign and a '1' used for a minus sign. The remaining bits are used for representing magnitude. Positive magnitudes are represented in the same way as in the case of sign-bit or 1's complement representation. Negative magnitudes are represented by the 2's complement of their positive counterparts. For example, +9 would be represented as 00001001, and −9 would be written as 11110111. Please note that, if the 2's complement of the magnitude of +9 gives a magnitude of −9, then the reverse process will also be true, i.e. the 2's complement of the magnitude of −9 will give a magnitude of +9. The n-bit notation of the 2's complement format can be used to represent all decimal numbers in the range from $+(2^{n-1}-1)$ to $-(2^{n-1}-1)$. The 2's complement format is very popular as it is very easy to generate the 2's complement of a binary number and also because arithmetic operations are relatively easier to perform when the numbers are represented in the 2's complement format.

1.19.1 's complement:

As mentioned above the 1's complement of a binary number is formed by simply changing each 1in thenumber to 0 and each 0 in the number to 1.

Example:

Binary Number Equivalent 1's complement

1011	⟹	0100
110001	⟹	001110
100100	⟹	011011
11001110	⟹	00110001
1010110	⟹	01010010

1.19.2 1's complement Subtraction:

(i) To subtract a smaller number from a larger number, the procedure is as follows:

1. Determine the 1's complement of the smaller number.
2. Add the 1's complement to the larger number
3. Remove the carry and add to the sum. The carry is called end-around carry.The number of bits in the minuend and subtrahend must be equal.

Example:

1. Subtract $(01110)_2$ from $(10001)_2$ using 1's complement

Direct Method of binary subtraction	1's complement method of subtraction	Equivalent Decimal subtraction
10001 - 01110 00011	10001 + 10000 (1's complement of 10001) 100010(end-around carry1) +1 (add carry to sum) 00011	17 - 14 3

2. Subtract $(101101)_2$ from $(110011)_2$using 1's complement

Direct Method of binary subtraction	1's complement method of subtraction	Equivalent Decimal subtraction
110011 - 101101 000110	110011 + 010010(1's complement of 101101**)** 1000101 (end-around carry1) +1 (add carry to sum) 000110	51 - 45 6

(ii) To subtract a larger number from a smaller binary number , the procedure is as follows

1. Determine the 1's complement of the larger number.
2. Add the 1's complement to the smaller number
3. The answer has an opposite sign and is the result. There is no carry

Example:

1. Subtract$(10010)_2$ from $(1100)_2$using 1's complement

Direct Method of binary subtraction	1's complement method of subtraction	Equivalent Decimal subtraction
01100 - 10010 - 00110	01100 + 01101(1's complement of 10010**)** 11001 put – sign and take 1's complement for the sum we get , -00110	51 - 45 -6

2. Subtract $(1101)_2$ from $(1010)_2$using 1's complement

Direct Method of binary subtraction	1's complement method of subtraction	Equivalent Decimal subtraction
1010 - 1101 - 0011	1010 + 0010(1's complement of 1101**)** 1100 put – sign and take 1's complement for the sum we get , -0011	10 - 13 - 3

1.19.2 2’s Complement:

The 2’s complement of a binary number is formed by taking 1’s complement of the number and adding 1 to the least significant bit (LSB) position

$$\boxed{2's\ complement = 1's\ complement\ +\ 1}$$

Example:

Write 2’s complement of a binary number (1010)2

```
1’s complement of 1010  =  0101
                 Add 1  =   + 1
   2’s complement of 1010  =  0110
```

Some more Example:

Binary Number		1’s complement	2’s complement
1011	⟹	0100⟹	0101
110001	⟹	001110⟹	001111
100100	⟹	011011⟹	011100
11001110	⟹	00110001⟹	00110010
1010110	⟹	01010010⟹	01010011

Addition Using the 2’s Complement Method

The 2’s complement is the most commonly used code for processing positive and negative binary numbers. It forms the basis of arithmetic circuits in modern computers. When the decimal numbers to be added are expressed in 2’s complement form, the addition of these numbers, following the basic laws of binary addition, gives correct results. Final carry obtained, if any, while adding MSBs should be disregarded. To illustrate this, we will consider the following four different cases:

1. Both the numbers are positive.
2. Larger of the two numbers is positive.
3. The larger of the two numbers is negative.
4. Both the numbers are negative.

Case 1

• Consider the decimal numbers +37 and +18.
• The 2’s complement of +37 in eight-bit representation = 00100101.
• The 2’s complement of +18 in eight-bit representation = 00010010.
• The addition of the two numbers, that is, +37 and +18, is performed as follows

```
  00100101
+ 00010010
  00110111
```

• The decimal equivalent of $(00110111)_2$ is (+55), which is the correct answer.

Case 2

• Consider the two decimal numbers +37 and -18.
• The 2’s complement representation of +37 in eight-bit representation = 00100101.
• The 2’s complement representation of −18 in eight-bit representation = 11101110.
• The addition of the two numbers, that is, +37 and −18, is performed as follows:

```
00100101
+ 11101110
  00010011
```

• The final carry has been disregarded.
• The decimal equivalent of $(00010011)_2$ is +19, which is the correct answer.
Case 3
• Consider the two decimal numbers +18 and −37.
• −37 in 2's complement form in eight−bit representation = 11011011.
• +18 in 2's complement form in eight−bit representation = 00010010.
• The addition of the two numbers, that is, −37 and +18, is performed as follows:
11011011
+ 00010010
11101101
• The decimal equivalent of $(11101101)_2$, which is in 2's complement form, is −19, which is the correct answer. 2's complement representation was discussed in detail in Chapter 1 on number systems.
Case 4
• Consider the two decimal numbers −18 and −37.
• −18 in 2's complement form is 11101110.
• −37 in 2's complement form is 11011011.
• The addition of the two numbers, that is, −37 and −18, is performed as follows:
11011011
+ 11101110
11001001
• The final carry in the ninth bit position is disregarded.
• The decimal equivalent of (11001001)2, which is in 2's complement form, is −55, which is the correct answer.
It may also be mentioned here that, in general, 2's complement notation can be used to perform addition when the expected result of addition lies in the range from -2^{n-1} to $+(^{2n-1}- 1)$, n being the number of bits used to represent the numbers. As an example, eight-bit 2's complement arithmetic cannot be used to perform addition if the result of addition lies outside the range from −128 to +127.
Different steps to be followed to do addition in 2's complement arithmetic are summarized as follows:
1. Represent the two numbers to be added in 2's complement form.
2. Do the addition using basic rules of binary addition.
3. Disregard the final carry, if any.
4. The result of addition is in 2's complement form.
Example 3.1
Perform the following addition operations:
1. $(275.75)_{10}$
+ $(37.875)_{10}$
2. $(AF1.B3)_{16}$
+ $(FFF.E)_{16}$
Solution
1. As a first step, the two given decimal numbers will be converted into their equivalent binary numbers (decimal-to-binary conversion has been covered at length in Chapter 1, and therefore the decimal-to-binary conversion details will not be given here):
$(275.75)_{10} = (100010011.11)_2$ and (37.875)10 = $(100101.111)_2$
The two binary numbers can be rewritten as $(100010011.110)_2$ and $(000100101.111)_2$ to have the same number of bits in their integer and fractional parts. The addition of two numbers is performed as follows:
100010011.110
000100101.111
100111001.101
The decimal equivalent of (100111001.101)2 is $(313.625)_{10.}$
2. $(AF1.B3)_{16}$
$=(101011110001.10110011)_2$ and $(FFF.E)_{16}$
$=(111111111111.1110)_2$. (1111111111
11.1110)$_2$ can also be written as $(111111111111.11100000)_2$ to have the same number of bits in the integer and fractional parts. The two numbers can now be added as follows:
0101011110001.10110011

0111111111111.11100000
1101011110001.10010011
The hexadecimal equivalent of $(1101011110001.10010011)_2$ is $(1AF1.93)_{16}$, which is equal to the hex addition of $(AF1.B3)_{16}$ and $(FFF.E)_{16}$.

Example 3.2
Find out whether 16-bit 2's complement arithmetic can be used to add 14 276 and 18 490.
Solution
The addition of decimal numbers 14 276 and 18 490 would yield 32 766. 16-bit 2's complement arithmetic has a range of −215 to +(215− 1), i.e. −32 768 to +32 767. The expected result is inside the allowable range. Therefore, 16-bit arithmetic can be used to add the given numbers.
Example 3.3
Add −118 and −32 firstly using eight-bit 2's complement arithmetic and then using 16-bit 2's complement arithmetic. Comment on the results.
Solution
• −118 in eight-bit 2's complement representation = 10001010.
• −32 in eight-bit 2's complement representation = 11100000.
• The addition of the two numbers, after disregarding the final carry in the ninth bit position, is 01101010. Now, the decimal equivalent of $(01101010)_2$, which is in 2's complement form, is +106. The reason for the wrong result is that the expected result, i.e. −150, lies outside the range of eight-bit 2's complement arithmetic. Eight-bit 2's complement arithmetic can be used when the expected result lies in the range from −27 to + (27 − 1), i.e. −128 to +127. −118 in 16-bit 2's complement representation = 1111111110001010.
• −32 in 16-bit 2's complement representation = 1111111111100000.
• The addition of the two numbers, after disregarding the final carry in the 17th position, produces 1111111101101010. The decimal equivalent of $(1111111101101010)_2$, which is in 2's complement form, is −150, which is the correct answer. 16-bit 2's complement arithmetic has produced the correct result, as the expected result lies within the range of 16-bit 2's complement notation.

Subtraction Using 2's Complement Arithmetic
Subtraction is similar to addition. Adding 2's complement of the subtrahend to the minuend and disregarding the carry, if any, achieves subtraction. The process is illustrated by considering six different cases:
1. Both minuend and subtrahend are positive. The subtrahend is the smaller of the two.
2. Both minuend and subtrahend are positive. The subtrahend is the larger of the two.
3. The minuend is positive. The subtrahend is negative and smaller in magnitude.
4. The minuend is positive. The subtrahend is negative and greater in magnitude.
5. Both minuend and subtrahend are negative. The minuend is the smaller of the two.
6. Both minuend and subtrahend are negative. The minuend is the larger of the two.
Case 1
• Let us subtract +14 from +24.
• The 2's complement representation of +24 = 00011000.
• The 2's complement representation of +14 = 00001110.
• Now, the 2's complement of the subtrahend (i.e. +14) is 11110010.
• Therefore, +24 − (+14) is given by

$$\begin{array}{r} 00011000 \\ +\ \underline{11110010} \\ \mathbf{00001010} \end{array}$$

with the final carry disregarded.
• The decimal equivalent of $(00001010)_2$ is +10, which is the correct answer.

Case 2
• Let us subtract +24 from +14.
• The 2's complement representation of +14 = 00001110.
• The 2's complement representation of +24 = 00011000.

• The 2’s complement of the subtrahend (i.e. +24) = 11101000.
• Therefore, +14 − (+24) is given by
  00001110
+ <u>11101000</u>
  11110110
• The decimal equivalent of (11110110)2, which is of course in 2’s complement form, is −10 which is the correct answer.
Case 3
• Let us subtract −14 from +24.
• The 2’s complement representation of +24 = 00011000 = minuend.
• The 2’s complement representation of −14 = 11110010 = subtrahend.
• The 2’s complement of the subtrahend (i.e. −14) = 00001110.
• Therefore, +24 − (−14) is performed as follows:
  00011000
+ <u>00001110</u>
  00100110
• The decimal equivalent of (00100110)2 is +38, which is the correct answer.
Case 4
• Let us subtract −24 from +14.
• The 2’s complement representation of +14 = 00001110 = minuend.
• The 2’s complement representation of −24 = 11101000 = subtrahend.
• The 2’s complement of the subtrahend (i.e. −24) = 00011000.
• Therefore, +14 − (−24) is performed as follows:
  00001110
+ <u>00011000</u>
  00100110
• The decimal equivalent of (00100110)2 is +38, which is the correct answer.
Case 5
• Let us subtract −14 from −24.
• The 2’s complement representation of −24 = 11101000 = minuend.

• The 2’s complement representation of −14=11110010 = subtrahend.
• The 2’s complement of the subtrahend = 00001110.
• Therefore, −24 − (−14) is given as follows:
  11101000
+ <u>00001110</u>
  11110110
• The decimal equivalent of (11110110)2, which is in 2’s complement form, is −10, which is the correct answer.
Case 6
• Let us subtract −24 from −14.
• The 2’s complement representation of −14 = 11110010 = minuend.
• The 2’s complement representation of −24=11101000 = subtrahend.
• The 2’s complement of the subtrahend = 00011000.
• Therefore, −14 − (−24) is given as follows:
  11110010
+ <u>00011000</u>
  00001010
with the final carry disregarded.
• The decimal equivalent of $(00001010)_2$, which is in 2’s complement form, is +10, which is the correct answer.
It may be mentioned that, in 2’s complement arithmetic, the answer is also in 2’s complement notation, only with the MSB indicating the sign and the remaining bits indicating the magnitude. In 2’s complement notation, positive magnitudes are represented in the same way as the straight binary numbers, while the negative magnitudes are represented as the 2’s complement of their straight binary counterparts. A ‘0’ in the MSB position indicates a positive sign, while a ‘1’ in the MSB position indicates a negative sign.

The different steps to be followed to do subtraction in 2's complement arithmetic are summarized as follows:

1. Represent the minuend and subtrahend in 2's complement form.
2. Find the 2's complement of the subtrahend.
3. Add the 2's complement of the subtrahend to the minuend.
4. Disregard the final carry, if any.
5. The result is in 2's complement form.
6. 2's complement notation can be used to perform subtraction when the expected result of subtraction lies in the range from -2^{n-1} to $+(2^{n-1}-1)$, n being the number of bits used to represent the numbers.

Example 3.4

Subtract $(1110.011)_2$ from $(11011.11)_2$ using basic rules of binary subtraction and verify the result by showing equivalent decimal subtraction.

Solution

The minuend and subtrahend are first modified to have the same number of bits in the integer and fractional parts. The modified minuend and subtrahend are $(11011.110)_2$ and $(01110.011)_2$ respectively:

$$\begin{array}{r} 11011.110 \\ -\ \underline{01110.011} \\ \mathbf{01101.011} \end{array}$$

The decimal equivalents of $(11011.110)_2$ and $(01110.011)_2$ are 27.75 and 14.375 respectively. Their difference is 13.375, which is the decimal equivalent of $(01101.011)_2$.

Example 3.5

Subtract (a) $(-64)_{10}$ from $(+32)_{10}$ and (b) $(29.A)_{16}$ from $(4F.B)_{16}$. Use 2's complement arithmetic.

Solution:

(a) $(+32)_{10}$ in 2's complement notation = $(00100000)_2$.

$(-64)_{10}$ in 2's complement notation = $(11000000)_2$.

The 2's complement of $(-64)_{10}$

= $(01000000)_2$.

$(+32)_{10} - (-64)_{10}$ is determined by adding the 2's complement of $(-64)_{10}$ to $(+32)_{10}$.

Therefore, the addition of $(00100000)_2$ to $(01000000)_2$ should give the result. The operation is shown as follows:

$$\begin{array}{r} 00100000 \\ \underline{+01000000} \\ \mathbf{01100000} \end{array}$$

The decimal equivalent of $(01100000)_2$ is +96, which is the correct answer as $+32 - (-64) = +96$.

(b) The minuend = $(4F.B)_{16}$

= (01001111.1011)2.

The minuend in 2's complement notation = $(01001111.1011)_2$.

The subtrahend = $(29.A)_{16}$

= $(00101001.1010)_2$.

The subtrahend in 2's complement notation = $(00101001.1010)_2$.

The 2's complement of the subtrahend = $(11010110.0110)_2$.

$(4F.B)_{16} - (29.A)_{16}$ is given by the addition of the 2's complement of the subtrahend to the minuend.

$$\begin{array}{r} 01001111\_1011 \\ \underline{+\ 11010110\_0110} \\ \mathbf{00100110\_0001} \end{array}$$

with the final carry disregarded. The result is also in 2's complement form. Since the result is a positive number, 2's complement notation is the same as it would be in the case of the straight binary code.

The hex equivalent of the resulting binary number = $(26.1)_{16}$, which is the correct answer.