UNIT-I

INTRODUCTION TO NUMBER SYSTEM & CODES

Number System:

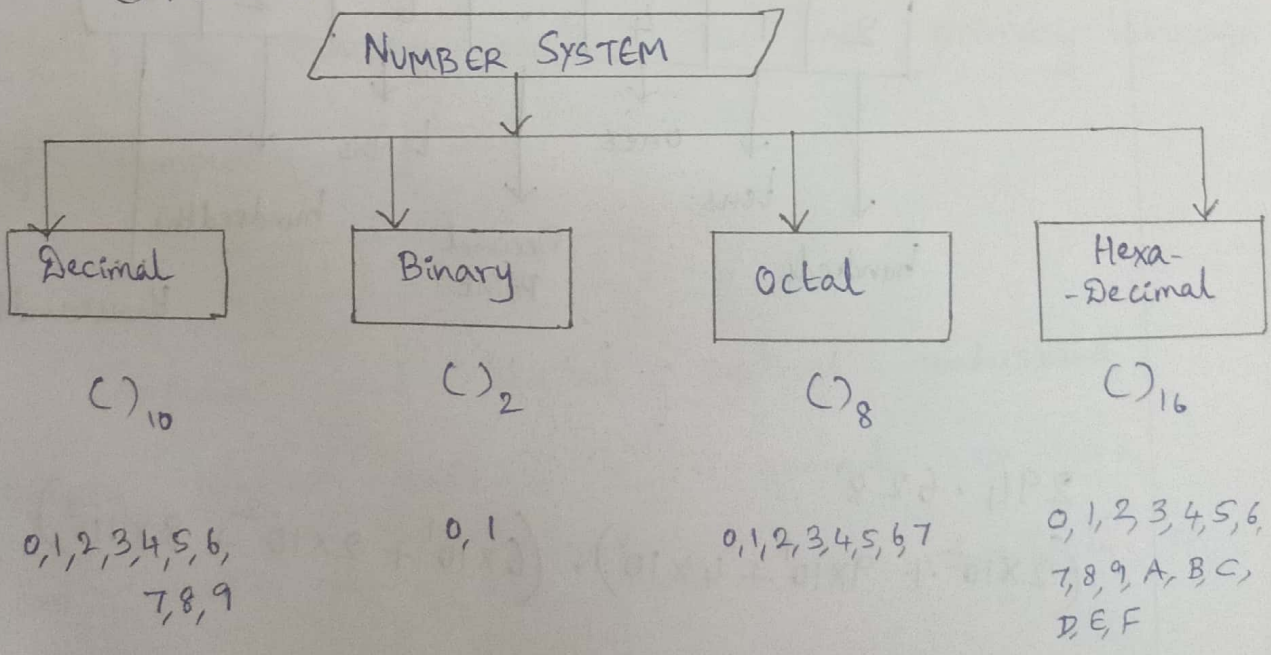
A Number system defines how a number can be represented using distinct symbols. A number can be represented differently in different systems.

Number system defines a set of values used to represent quantity.

In our daily life we use the Decimal Number system.

Types of Number Systems:

(1) Decimal Number System
(2) Binary Number system
(3) Octal Number system
(4) Hexadecimal Number System



(1) Decimal Number System:

Decimal Number system is commonly used to represent numbers. The base of the decimal number system is 10. Base is also called as Radix.

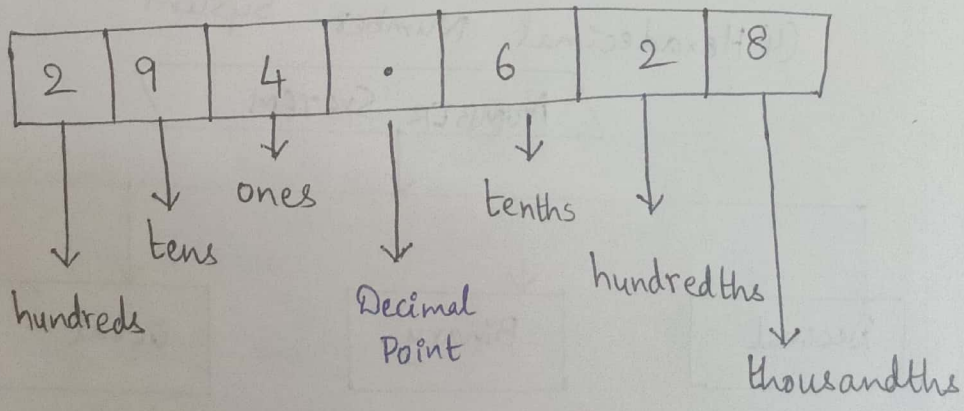
In decimal number system we use the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 to represent any number.

Ex: 294.628

Integral part is 294 (left of decimal point)

fractional part is 628 (right of decimal point)

Note: Each value in this number system has the place value of power 10.



294.628

$(2\times10^{2} + 9\times10^{1} + 4\times10^{0}) . (6\times10^{-1} + 2\times10^{-2} + 8\times10^{-3})$

Scanned with OKEN Scanner

$(294.628)_{10}$ → This is the representation of decimal number.

2. Binary Number System:

All digital circuits and systems use Binary number system. The base or radix of the binary number system is '2'. We use 0, 1 digits to represent the numbers in Binary Number System.

* Digits 0 and 1 are called Bits.
* 8 bits together make a Byte.

Ex: $(110011)_2$

Note: The Binary system is the primary language of Computing Systems.

0	1	1	0	1	1

Computers and Digital systems understand Binary number system. In general 0 and 1 are used as OFF and ON respectively if we consider digital circuits.

Scanned with OKEN Scanner

(3) Octal Number System:

A number system which has its base as "eight" is called an Octal number. It uses numbers as 0 to 7., i.e., digits 0, 1, 2, 3, 4, 5, 6, 7.

Base or Radix of octal number system is '8'.

Octal representation

0
1
2
3
4
5
6
7

Place value

4	2	1	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

General representation:

$$(abc)_8 = a \times 8^2 + b \times 8^1 + c \times 8^0$$

Note:

* It is a Positional Value system.
* Each digit has its value expressed in powers of 8.
* Decimal equivalent of any octal number is the sum of product of each digit with its positional value.

Scanned with OKEN Scanner

Ex: $(726)_8 = 7\times8^2 + 2\times8^1 + 6\times8^0$

$= 7\times64 + 16 + 6$

$= 448 + 22$

$= (470)_{10}$

$\therefore \boxed{(726)_8 = (470)_{10}}$

In the above example $(726)_8$

	7	2	6
Face Value	7	2	6
Position	3 (hundreds)	2 (tens)	1 (ones)
Positional Value	$64 / 8^2$	$8 / 8^1$	$1 / 8^0$
Place value	448	16	6

$448 + 16 + 6 = (470)_{10}$

* Used widely in Computing Systems.

(4) Hexadecimal Number System :

Hexadecimal Number system is a positional numeral system that represents numbers using base or radix of 16. It is represented by only 16 digits. Numbers ranging from 0 to 9

Scanned with OKEN Scanner

and alphabets from A to F.

i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
(A = 10, B = 11, C = 12, D = 13, E = 14, F = 15)

Ex:

$$(abc)_{16} = a \times 16^2 + b \times 16^1 + c \times 16^0$$

Hexadecimal Number system is widely used in Computing Systems and memory representations.

Decimal Numbers	4-bit Binary Number (8 4 2 1)	Hexadecimal Number
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B

Scanned with OKEN Scanner

	8	4	2	1	
12	1	1	0	0	C
13	1	1	0	1	D
14	1	1	1	0	E
15	1	1	1	1	F

Ex:

$$(2A5)_{16} = 2\times16^2 + A\times16^1 + 5\times16^0$$

$$= 2\times256 + 10\times16 + 5\times1$$

$$= 512 + 160 + 5$$

$$= 677$$

$$\boxed{(2A5)_{16} = (677)_{10}}$$

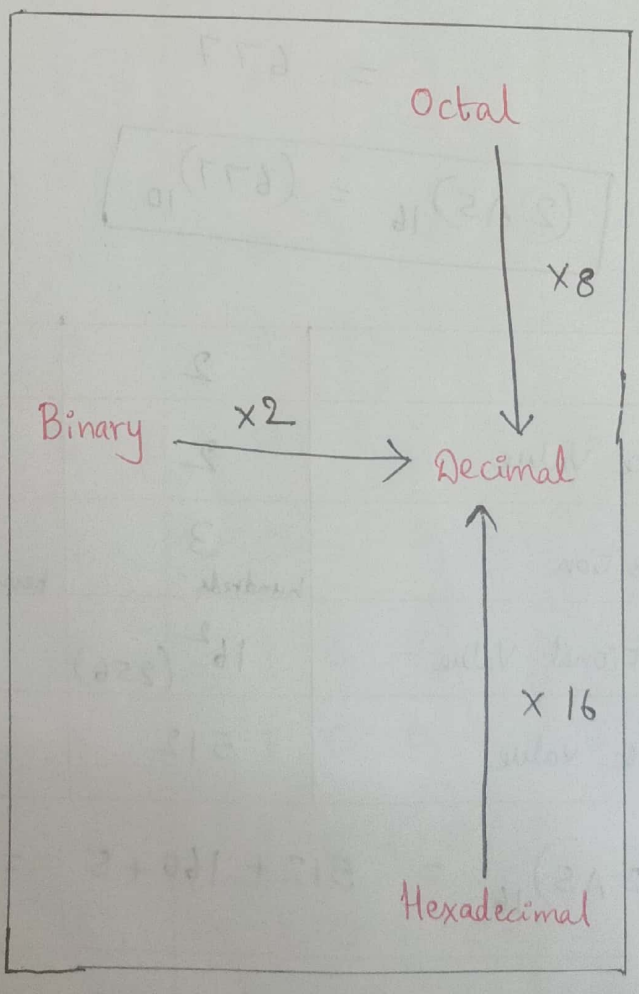
	2	A	5
Face Value	2	A	5
Position	3 hundreds	2 tens	1 ones
Positional Value	16^2 (256)	16^1 (16)	16^0 (1)
Place value	512	160	5

$$(2A5)_{16} = 512 + 160 + 5 = (677)_{10}$$

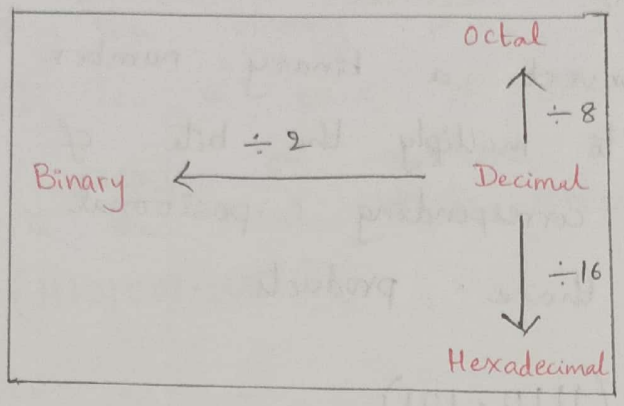
Scanned with OKEN Scanner

Name of the Number System	Base or Radix	No. of digits 0 to (r-1)	Digits
Binary	2	0 to (2-1)	0 and 1
Octal	8	0 to (8-1)	0 to 7 0 1 2 3 4 5 6 7
Decimal	10	0 to (10-1)	0 to 9 0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	0 to (16-1)	0 to 15 0 1 2 3 4 5 6 7 8 9 A B C D E F

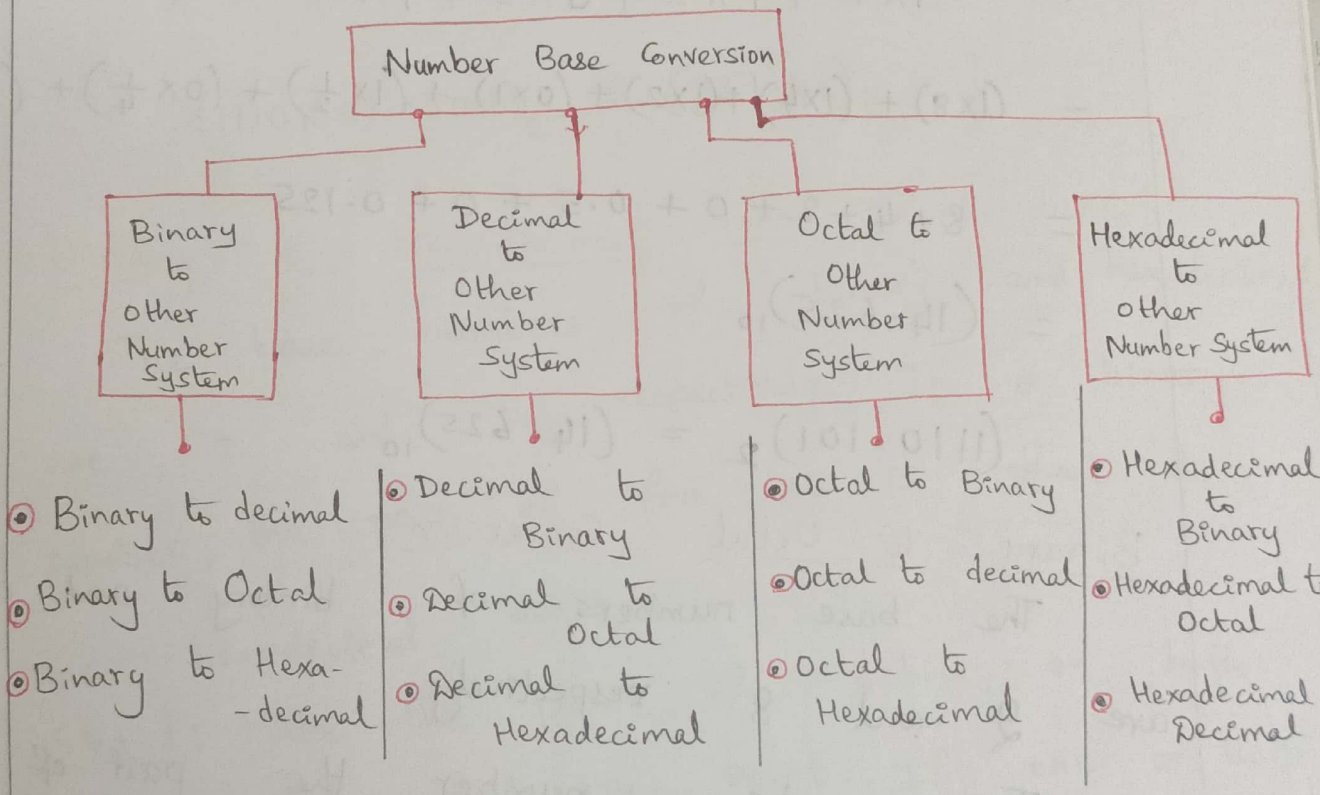
Number Base Conversions:



Scanned with OKEN Scanner



Number of Possible conversions 12.



Binary to Other number Systems:

There are three possible conversions.

① Binary to Decimal

② Binary to Octal

③ Binary to Hexadecimal

Binary to Decimal Conversion:

To convert a binary number into decimal we have to multiply the bits of binary number with its corresponding positional weights., and add all those products.

Ex: $(1110.101)_2$

$$1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 + 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3}$$

$$= (1\times 8) + (1\times 4) + (1\times 2) + (0\times 1) + \left(1\times \tfrac{1}{2}\right) + \left(0\times \tfrac{1}{4}\right) + \left(1\times \tfrac{1}{8}\right)$$

$$= 8 + 4 + 2 + 0 + 0.5 + 0 + 0.125$$

$$= (14.625)_{10}$$

$$(1110.101)_2 = (14.625)_{10}$$

Binary to Octal Conversion:

The base numbers of binary and octal are 2 and 8 respectively.

In a binary number, the pair of three bits is equal to one octal digit.

In two simple steps we can convert Binary to octal number.

Step1: In the first step we have to make the pairs of three bits on both

sides of the binary point. If one or two bits left in a pair of three bits pair, we add zeros on extreme sides.

Step 2: Write the octal digits corresponding to each pair of three digits.

Ex: $(11101001.10011)_2$

4 2 1 4 2 1 4 2 1 4 2 1
011 101 001 . 100 110

3 5 1 . 4 6

$$(11101001.10011)_2 = (351.46)_8$$

Binary to Hexadecimal Conversion:

The base numbers of binary and hexadecimal are 2 and 16 respectively. In a binary number, the pair of four bits is equal to one hexadecimal digit.

Step 1: Make the pairs of 4 bits on both the sides of binary point. If one or two bits left we add 0's on extreme sides.

Step 2: Write the hexadecimal digits corresponding to each pair of 4 digits.

Ex: $(111100011100.110110)_2$

8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1
0111 1000 1100 . 1101 1010

7 8 C . D A

Scanned with OKEN Scanner

$(1111000\,1100.1101101)_2 = (78C.DA)_{16}$

Decimal to Other Number System

* The decimal number can be an integer or floating-point integer.
* We convert decimal part and fractional part of the number individually.

Conversion of decimal to other number system can be done in two steps.

Step 1:

* We perform the division operation on the number and successive part with base 'r'.
* We will list down all the remainders till the quotient is zero.
* We will write the remainders in reverse order for getting the equivalent number of base 'r'.
* We denote Least and Most Significant digits are denoted by the first and last remainders.

Step 2:

* Multiplication operation is done with base 'r' of fractional and successive fraction.
* The carriers are noted until the result is zero or when the required number of equivalent digit is obtained.

Scanned with OKEN Scanner

* To get the fractional part of the equivalent number of base 'r', the normal sequence of carrying is considered.

Decimal to Binary Conversion:

To convert decimal to binary number we follow two simple steps.

Step 1:

We perform the division operation on the decimal number and the successive quotient with the base of binary (2).

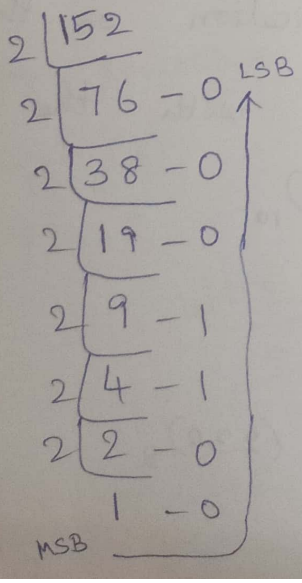
Step 2:

We perform the multiplication on the number and the successive quotient with the base of binary (2).

Ex: $(152.25)_{10}$

152 + 0.25

Step 1:



$(152)_{10} = (10011000)_2$

~~0.25~~

Step 2:

$0.25 \times 2 = 0.50$ — 0 ↓

$0.50 \times 2 = 1.00$ — 1

$(0.25)_{10} = (0.01)_2$

Finally

$$(152.25)_{10} = (10011000.01)_2$$

~~Binary~~ to

Decimal to Octal Conversion:

In two steps we can convert a decimal number to octal number.

Step 1:

We perform division operation on the decimal number and successive quotient with the base of octal (8).

Step 2:

We perform the multiplication on the integer and the successive quotient with the base of octal (8).

Ex: $(152.25)_{10}$

Step 1: 152

8 | 152

8 | 19 — 0 ↑ LSB

2 — 3

MSB

$(230)_8$

Scanned with OKEN Scanner

$(152)_{10} = (230)_8$

step2 : $(0.25)_{10}$

$0.25 \times 8 = 2.00$ — 2

$(0.25)_{10} = (0.2)_8$

$\boxed{(152.25)_{10} = (230.2)_8}$

Decimal to Hexadecimal Conversion:

Two steps to convert decimal number to hexadecimal number.

Step 1:

We perform the division operation on the number and successive quotient with the base of hexadecimal (16).

Step 2:

We perform the multiplication on the integer and the successive quotient with the base of hexadecimal (16).

Ex: 152.25

step1:

16 | 152

9 — 8

$\longrightarrow$

$(152)_{10} = (98)_{16}$

Scanned with OKEN Scanner

Step 2:

$0.25 \times 16 = 4.00 \quad - 4$

$(0.25)_{10} = (4)_{16}$

$\boxed{(152.25)_{10} = (98.4)_{16}}$

Octal to Other number Conversion

Like binary and decimal the octal number can also be converted into other number systems. The process of conversion differs from others.

Octal to Decimal Conversion:

The process of converting octal to decimal starts from multiplying the digits of octal numbers with its corresponding positional weights. At last we add all those products.

Ex: 243.21

We multiply each digit of 243.21 with its respective positional weight, and last we add the products of all the bits with its weight.

$(243.21) = 2 \times 8^2 + 4 \times 8^1 + 3 \times 8^0 + 2 \times 8^{-1} + 1 \times 8^{-2}$

$= 2 \times 64 + 4 \times 8 + 3 \times 1 + \frac{2}{8} \times \frac{1}{8} + \frac{1}{64}$

Scanned with OKEN Scanner

$$= 128 + 32 + 3 + 0.25 + 0.015$$

$$= 163.265$$

$$(243.21)_8 = (163.265)_{10}$$

Octal to Binary Conversion:

The process of Octal to Binary conversion is the reverse process of Binary to Octal.

We have to write the three bit binary code of each octal number digit.

Ex: $(732.16)_8$

We write the three bit binary digit for 7, 3, 2, 1 and 6.

8	7	3	2	1	6
	4 2 1	4 2 1	4 2 1	4 2 1	4 2 1
	1 1 1	0 1 1	0 1 0	0 0 1	1 1 0

So the binary equivalent of

$$(732.16)_8 = (111\ 011\ 010 \cdot 001\ 110)_2$$

Octal to Hexadecimal Conversion:

Octal to Hexadecimal conversion can be done in two steps.

Scanned with OKEN Scanner

Step 1:

In the first step we find the binary equivalent of the given number.

Step 2:

Next, we have to make the pairs of four bits on both the sides of the binary point. If there are one, two, or three bits left in a pair of four bits pair, we have to add the required number of zeros on extreme sides.

Write the hexadecimal digits corresponding to each pair.

Ex:

$(315.26)_8$

Step 1:

3	1	5		2	6
421	421	421		421	421
011	001	101	.	010	110

$315.26 = 011\ 001\ 101 \cdot 010\ 110$

Step 2:

8421	8421	8421		8421	8421
0000	1100	1101	.	0101	1000
0	C	D	.	5	8

$(315.26)_8 = (CD.58)_{16}$

Hexadecimal to Other Number System:

Hexadecimal to decimal number conversion differs from the remaining conversions.

Hexadecimal to Decimal Conversion:

The conversion process of hexadecimal to decimal starts from multiplying the digits of hexadecimal numbers with its corresponding positional weights.

Then, we add all those products.

Ex: $(32A.5C)_{16}$

We multiply each digit with its positional weight and last we add the products.

$$(32A.5C)_{16} = 3\times16^{2} + 2\times16^{1} + A\times16^{0} + 5\times16^{-1} + C\times16^{-2}$$

$$= (3\times256) + (2\times16) + (10\times1) + \left(5\times\frac{1}{16}\right) + \left(12\times\frac{1}{256}\right)$$

$$= 768 + 32 + 10 + (5\times0.0625) + (12\times0.0039)$$

$$= 810 + 0.3125 + 0.0468$$

$$= 810.3593$$

$$\boxed{(32A.5C)_{16} = (810.3593)_{10}}$$

Scanned with OKEN Scanner

Hexadecimal to Binary Conversion:

We write the four bit binary code of each hexadecimal digit.

Ex: $(2B3.1A)_{16}$

2	B	3	1	A
8421	8421	8421	8421	8421
0010	1011	0011	0001	1010

$$(2B3.1A)_{16} = (0010\ 1011\ 0011.\ 0001\ 1010)_2$$

Hexadecimal to Octal Conversion:

In two steps we can convert Hexadecimal to Octal number.

Step 1:

Write the binary equivalent of the hexadecimal number.

Step 2:

Make the pairs of three bits on both the sides of the binary point.

If two or one bit left in a pair of three bits, add the required number of zeros on the extreme sides.

Scanned with OKEN Scanner

Now, write the octal digits corresponding to each pair.

Ex:

$(D21 \cdot E3)_{16}$

Step1

Write the four bit binary digit for D, 2, 1, E and 3

D	2	1	E	3
8421	8421	8421	8421	8421
1101	0010	0001 .	1110	0011

Step 2:

Make three bit pairs on both sides of the binary point.

11 010 0100 001 . 111 000 11

1101 0010 0001 . 1110 0011

421	421	421	421 .	421	421	421
110	100	100	001 .	111	000	110
6	4	4	1 .	7	0	6

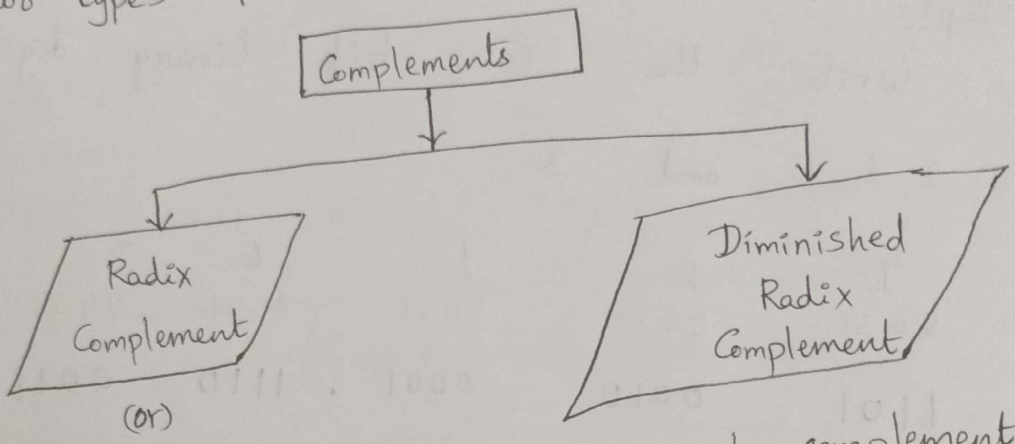
$$(D21 \cdot E3)_{16} = (6441 \cdot 706)_8$$

Scanned with OKEN Scanner

Complements:

Complements are used in digital computers for simplifying subtraction Operation.

For every system of radix-r there are two types of complements.



r's Complement:

For a positive number 'N' in base 'r' with an integer part of n digits, r's complement of N is

$$\begin{cases} r^n - N & \text{for } N \neq 0 \\ 0 & \text{for } N = 0 \end{cases}$$

Eg: 10's complement of $(52520)_{10}$ is ??

Here $N = 52520$, $r = 10$

n = no. of digits $= 5$

r's comp $= r^n - N$

$= 10^5 - 52520 = 100000 - 52520 = 47480$

10's complement of (52520) is (47480)

Scanned with OKEN Scanner

Find 10's complements of $(0.3267)_{10}$

Here $N = 0.3267$

$n = 0$

$r = 10$

10's comp $= r^n - N$

$= 10^0 - 0.3267$

$= 1 - 0.3267$

$= 0.6733$

10's complement of (0.3267) is (0.6733)

(r-1)'s complement :

For a positive number N in base r with an integer part of n digits and a fraction part of m digits,

(r-1)'s complement of N is $\begin{cases} r^n - r^{-m} - N & \text{for } N \neq 0 \\ 0 & \text{for } N = 0 \end{cases}$

Eg: 9's complement of $(23970)_{10}$

Here $N = 23970$

$n = 5$

$m = 0$

9's comp $= r^n - r^{-m} - N = 10^5 - 10^{-0} - 23970$

Scanned with OKEN Scanner

$$= 100000 - \frac{1}{10^0} - 23970$$

$$= 100000 - 1 - 23970$$

$$= 76029$$

9's complement of (23970) is (76029).

Base 2 → 1's complement
Base 2 → 2's complement

Base 8 → 7's complement
Base 8 → 8's complement

Base 10 → 9's complement
Base 10 → 10's complement

Base 16 → 15's complement
Base 16 → 16's complement

We get r's complement if we add 1 to (r-1)'s complement.

* r's complement = (r-1)'s complement + 1

(r-1)'s complement = r's complement − 1

∴ r's complement = $r^n - N$

(r-1)'s complement = $r^n - N - 1$ (or) $r^n - 1 - N$

Scanned with OKEN Scanner

Find the 2's complement of 1101

Method 1:

$r = 2$, $n = 4$, $N = 1101$

$2^4 - 1101 = 16 - 1101 = 10000 - 1101$

$(\because 16 = {}^{16\ 8\ 4\ 2\ 1}10000)$

$$\begin{array}{r} {}^{0\ 1\ 1\ 1\ 1}10000 \\ 1101 \\ \hline 00011 \end{array}$$

$10000 - 1101 = 00011 = 3$

2's complement of 1101 is 3.

Method 2: 1101

1's complement of 1101 is 0010

2's complement = 1's complement + 1

$= 0010 + 0001$

$= 0011 \rightarrow 3$

$$\begin{array}{r} 0010 \\ 0001 \\ \hline 0011 \end{array}$$

2's complement of 1101 is 3.

Find the 7's complement of an octal number 5674.

$(r-1)$'s complement $= r^n - N - 1$

Here $r = 8$, $n = 4$, $N = (5674)_8$

7's complement $= 8^4 - (5674)_8 - 1$

Scanned with OKEN Scanner

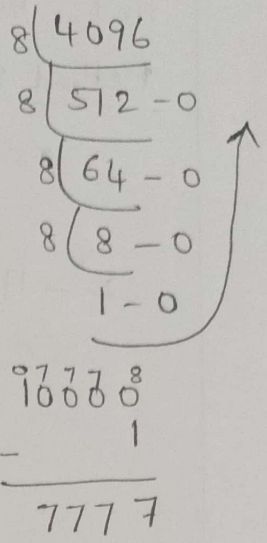
$= (4096)_{10} - (5674)_8 - 1$

Converting $(4096)_{10}$ to Octal

$= 10000 - 5674 - 1$

$= 7777 - 5674$

$= 2103.$



7's complement of 5674 is 2103.

Note:

Directly we can ~~directly~~ take the 4 digit largest number and subtract it from this. (given number)

$7777 - 5674 = 2103.$

Now 8's complement of 5674 = 7's complement + 1

$= 2103 + 1$

$= 2104.$

8's complement of 2103 is 2104.

Ex: 1's & 2's complement of 1101.

Take the largest 4 digit number i.e., 1111

$1111 - 1101$

```
  1111
- 1101
------
  0010   → 1's complement
```

2's complement of 1101 is 1's complement + 1.

i.e.,
```
  0010
+    1
------
  0011
```

1's complement of 1101 is 0010

2's complement of 1101 is 0011

Subtraction with r's complement:

The subtraction of two positive numbers M-N both of base r may be done as follows.

① Add the minuend M to the r's complement of subtrahend N.

② From the result obtained in first step fo an end carry

(i) If an end carry occurs, discard it.

(ii) If an end carry doesn't occur, take r's complement of the number obtained in

step 1, and place a negative sign infront.

Eg: Perform subtraction using 10's complement

5250 - 321

Here M = 5250

N = 321

finding 10's complement of N

N = 0321

n = 4

$r^n - N = 10^4 - 0321 = 10000 - 0321$

$= 9679$

Now perform M + 10's complement of N.

5250 + 9679 = 14929

↓

descard the carry.

∴ 5250 - 321 = 4929

Eg: Perform subtraction using 10's complement

753 - 864

Here M = 753

N = 864

10's complement of $N = r^n - N$

$N = 864$

$n = 3$

10's complement $= 10^3 - 864 = 1000 - 864 = 136$

10's complement of N (864) is 136

Perform M + 10's complement of N

$= 753 + 136 = 889$

↳ No carry.

So find 10's complement of 889.

$10^3 - 889 = 1000 - 889 = +111$

Placing negative sign for the result;

$\therefore$ **753 - 864 = -111**

Eg: Perform subtraction using 2's complement

11010 - 1101

Here M = 11010

N = 01101

2's complement of N = 10010

$+ \quad 1$

10011

Perform M + 2's complement of N

11010 + 10011 = $\begin{array}{r} 11010 \\ 10011 \\ \hline 101101 \end{array}$

→ discard the carry.

(1)01101

So $\boxed{11010 - 1101 = 01101}$

Eg: Perform subtraction using 2's complement.

10010 − 10011

Here M = 10010

N = 10011

2's complement of N = 10011 = 01100
1
01101

Perform M + 2's complement of N

10010 + 01101 = 10010
01101
11111 — No carry

Again find 2's complement of result 11111

i.e., 00000
1
00001

Place negative sign for the result (−00001)

∴ $\boxed{10010 - 10011 = (-00001)}$

Subtraction with (r-1)'s complement:

The subtraction of two positive numbers both of base r may be done as M-N follow:

① Add the minuend M to the (r-1)'s complement of subtrahend N.

② Find the result obtained in step 1 for an end carry.

(i) If an end carry occurs, add 1 to the least significant digit.

(ii) If an end carry doesnot occur take (r-1)'s complement of number obtained in step 1 and place a negative sign infront.

Eg: Perform subtraction using 9's complement

5250 - 321

Here M = 5250

N = 0321

9's complement of N $= r^n - r^{-m} - N$

$10^4 - 10^{-0} - 0321 = 10000 - 1 - 0321$

$= 9678$

Scanned with OKEN Scanner

Perform M + 9's complement of N

5250 + 9678 = (1)4928

↳ carry

If carry occurs add 1 to LSB.

~~4298~~	4928
1	1
429	4929

∴ **5250 − 321 = 4929**

Eg:

Perform subtraction using 9's complement 753-864

Here M = 753

N = 864

9's complement of N $= r^n - r^{-m} - N$

$= 10^3 - 10^{-0} - 864 = 1000 - 1 - 864$

$= 135$

Perform M + 9's complement of N = 753 + 135

= 888

↳ No carry

Finding 9's complement of 888

$r^n - r^{-m} - N = 1000 - 1 - 888 = 111$

Placing negative sign for 111 = (-111)

∴ **753 − 864 = (−111)**

Scanned with OKEN Scanner

Eg: Perform subtraction using 1's complement

1010100 − 1000100

Here M = 1010100

N = 1000100

1's complement of N = 0111011

Perform M + 1's complement of N

```
  1010100
  0111011
 ---------
 (1)0001111
```

↳ carry occurs

Add 1 to LSB

```
  0001111
        1
  ①①①①
 ---------
  0010000
```

∴ 1010100 − 1000100 = 0010000

Eg: Perform subtraction using 1's complement

1000100 − 1010100

Here M = 1000100

N = 1010100

Perform M + 1's complement of N

1's complement of N = 0101011

~~00~~ *

~~100~~

M+ 1's complement of N

```
  1000100
  0101011
---------
  1101111
```

No carry occurs

1's complement of 1101111 is 0010000

Add negative sign :, −10000

∴ **1000100 − 1010100 = (−10000)**

Binary Arithmetic in Computers:

Binary Arithmetic is an essential part of various digital systems. We can add, subtract, multiply and divide binary numbers.

Binary Addition:

Binary Addition is performed as same in Decimal additions. While performing Binary addition we will have two outputs sum (s) and carry (c).

Scanned with OKEN Scanner

Input A	Input B	Sum(S) (A+B)	Carry(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$0+0 = 0$

$0+1 = 1$

$1+0 = 1$

$1+1 = 0$ (with a carry to adjacent left bit)

When we perform the addition 1+1, we get the value 2. But in Binary world, two is represented as 10_2. Thus we write a 0, and the (one) 1 is overflown to the next bit.

Eg:

$(85)_{10}$

$+(181)_{10}$

$(266)_{10}$

```
  1 1 1   1   1
  0 1 0 1 0 1 0 1
  1 0 1 1 0 1 0 1
-----------------
1 0 0 0 0 1 0 1 0
```

Binary Subtraction:

Binary subtraction is performed as same in Decimal subtractions. While performing binary subtraction we will have two outputs., Difference and Borrow.

Input A	Input B	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$0 - 0 = 0$

$0 - 1 = 1$ (borrow / take 1 from the adjacent bit to the left)

$1 - 0 = 1$

$1 - 1 = 0$

In the second case, we see ~~$0 - 1 = 0$~~

$0 - 1 = 1$ with a borrow 1., as it becomes $(10)_2$

Scanned with OKEN Scanner

for binary 2. Thus 2-1 gives 1.

Eg:

$(26)_{10}$ 0 0 1 1 0 1 0

$-(12)_{10}$ 0 0 0 1 1 0 0

$(14)_{10}$ 0 0 0 1 1 1 0

Binary Multiplication:

The binary multiplication is the easiest one when compared to the other operations. It is pretty similar to decimal multiplication - any number multiplied with a 0 gives 0 as the product.

Input A	Input B	Product
0	0	0
0	1	0
1	0	0
1	1	1

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

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

Scanned with OKEN Scanner

Eg:

```
  1001
 x 101
------
  1001
 0000
1001
------
101101
------
```

1001 x 101 = 101101

Binary Division:

Under Binary division, we perform two main functions - Multiplication and subtraction. We also call this method as the long division method.

Eg:

```
      101
    -----
101 ) 11010
      -101
      ----
        11
       -00
       ---
        110
       -101
       ----
          1
```

```
      5
    -----
5 ) 26
    -25
    ---
     1
```

Input A	Input B	Divide (D) A/B
0	0	Not defined
0	1	0
1	0	Not defined
1	1	1

Binary Codes:

Digital Data is represented, stored and transmitted as group of bits. This group of bits are termed as Binary code. These weights are broadly categorized into five categories.

1. Weighted Codes
2. Non-weighted Codes
3. Alphanumeric Codes
4. Error Detecting codes
5. Error Correcting Codes
6. Binary Coded Decimal code

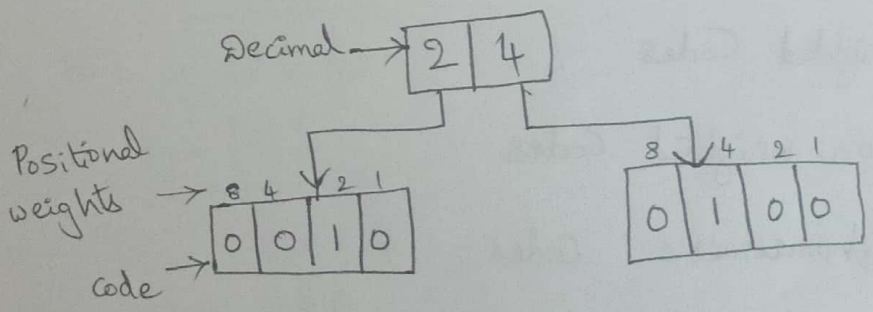
Scanned with OKEN Scanner

Advantages of Binary code:

* Binary codes are suitable for computer applications & 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.

① 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.



* A sequence of binary bits which represents a decimal digit is called codeword.

Scanned with OKEN Scanner

Weighted codes can be further classified as positively weighted codes and negatively weighted codes.

8421 Code:

* The weights of this code are 8, 4, 2 and 1.
* This code has all positive weights. It is positively weighted code.
* It is also called as BCD code (Binary coded decimal)

Adv:

It is very similar to decimal system.

We need to remember binary equivalent of decimal numbers 0 to 9 only.

Dis Adv:

Addition and subtraction of BCD have different rules.

BCD arithmetic is little more complicated.

BCD needs more number of bits than binary to represent decimal number. BCD is less efficient than binary.

Scanned with OKEN Scanner

Eg: 12

Binary 1100 (4 bits)

BCD 00001100 (8 bits)

2421 code

The weights of this code are 2, 4, 2 and 1. This code has all positive weights. So it is positively weighted code.

It is unnatural BCD code.

Sum of weights of unnatural BCD code is equal to 9.

It is a self-complementing code.

Self-complementing codes provide 9's complement of a decimal number just by interchanging 1's and 0's in its equivalent 2421 representation.

Decimal	0	1	2	3	4
2421	0000	0001	0010	0011	0100

It differs from '8'.

Scanned with OKEN Scanner

Decimal	2421 code			
	2	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	1	1	0
9	1	1	1	1

8 4 -2 -1 code:

The weights of this code are 8, 4, -2, -1. This code has negative weights along with positive weights. So it is negatively weighted code.

It is unnatural BCD and self complementing code.

Scanned with OKEN Scanner

Decimal Number	8 4 -2 -1 code 8 4 -2 -1
0	0 0 0 0
1	0 1 1 1
2	0 1 1 0
3	0 1 0 1
4	0 1 0 0
5	1 0 1 1
6	1 0 1 0
7	1 0 0 1
8	1 0 0 0
9	1 1 1 1

Ex:

Find 8421, 2421 and 8,4,-2,-1 equivalent of decimal number 786.

8421

7 8 6

0111 1000 0110

Scanned with OKEN Scanner

$(786)_{10} = (0111\ 1000\ 0110)_{BCD}$

2421

7	8	6
1101	1110	1100

2421 equivalent of 786 is 110111101100.

8 4 -2 -1

7	8	6
1001	1000	1010

8 4 -2 -1 equivalent of 786 is 100110001010.

② Non-weighted Codes:

In Non-weighted codes the positional weights are not assigned.

we have two Non-weighted codes.

① Excess-3 code

② Gray code.

Non-weighted Binary codes are used

@ To perform certain arithmetic operations

Scanned with OKEN Scanner

(b) Shift position encodes

(c) Error detecting purpose.

(1) Excess-3 code:

Its also called as XS-3 code.

Used to express decimal numbers.

Excess-3 codes are derived from 8421 Binary code words.

Adding $(0011)_2$ or $(3)_{10}$ to each code word in 8421 code we can get excess-3 code.

Decimal	8421 Code	Excess-3
	0000	0011
0	0001	0100
1	0010	0101
2	0011	0110
3		0111
4	0100	1000
5	0101	1001
6	0110	1010
7	0111	1011
8	1000	1100
9	1001	

Scanned with OKEN Scanner

② Gray Code:

It is not an arithmetic code.

There are no specific weights assigned to bit position.

Only one bit will change each time the decimal number is incremented.

As only 1 bit changes at a time Gray code is also called as Unit distance code.

Gray code is a cyclic code.

It is also called reflected code.

Applications:

Used in shaft position Encoders.

Shaft position enoder produces a code word which is represents angular position of shaft.

Binary code to Gray code conversion:

① Consider the given binary code and place a zero to left of MSB.

② Compare the successive two bits starting from zero. If 2 bits are same output is zero, Otherwise output is one.

③ Repeat the above step till LSB of gray code is obtained.

Ex:- Binary code 1000

① Place zero to left of MSB

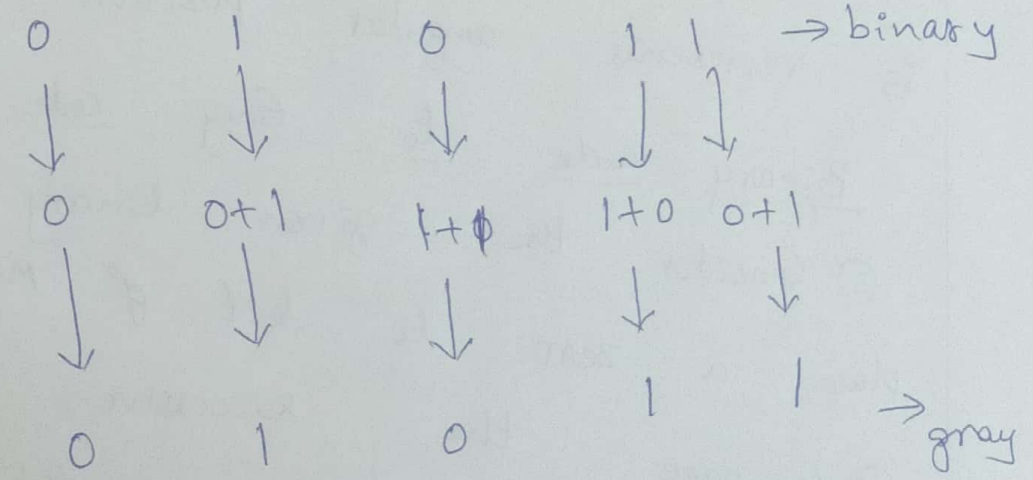
01000

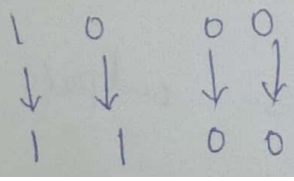
② Compare the successive two bits

01000

1100

In the gray code, the MSB will always be same as the 1st bit of given binary number.





Scanned with OKEN Scanner

Error Detection & Correction Codes:

During transmission of binary data from one system to the other noise may also be added. Due to this there may be errors in received data at other system.

Bit 0 may be changed to 1 or bit 1 may change to 0.

We can't avoid interference of noise we can get back original data first by detecting whether any error is present and then correcting those errors.

Error Detection code:

It is used to detect error present in received data bit stream.

These codes contain some bits which are appended to original bit stream.

Eg: Parity code.

Types of Error Detection codes

1. Parity Checking
2. Cyclic Redundancy check (CRC)

Scanned with OKEN Scanner

3. Longitudinal Redundancy Check (LRC)

4. Checksum

Parity Checking:

Parity bit means additional bit added to the data at the transmitter before transmitting data.

Before adding parity bit calculate no. of 1's and 0's.

Based on this calculation an extra bit is added to actual data.

Addition of parity bit to the data will result in change of data string size.

If we have an 8 bit data after adding a parity bit it will become a 9 bit binary data string.

It is also called Vertical Redundancy check (VRC).

Two types of parity bits.

① Even Parity

② Odd Parity

Scanned with OKEN Scanner

Even Parity:

If the data has even number of 1's parity bit is '0'.

If the data has odd number of 1's parity bit is '1'.

Eg: 10000101 11

Parity bit is '0'

Eg: 1101101

Parity bit is '1'

Odd Parity:

If data has odd number of 1's parity bit is zero.

Eg 10011101

Parity bit is '0'

If data has even number of 1's parity is one.

Eg: 10010101

Parity bit is '1'

The circuit which adds parity bit to the data at transmitter is called

Parity Generator.

Parity bits are transmitted and checked at the receiver.

If the parity bit at transmitter and receiver are not equal then an error is detected.

The circuit which checks parity at receiver are called Parity Checker.

3 bit data			Message with even parity		Message with odd parity	
			Msg	Parity	Msg	Parity
A	B	C				
0	0	0	000	0	000	1
0	0	1	001	1	001	0
0	1	0	010	1	010	0
0	1	1	011	0	011	1
1	0	0	100	1	100	0
1	0	1	101	0	101	1
1	1	0	110	0	110	1
1	1	1	111	1	111	0

Scanned with OKEN Scanner

Error Correcting Codes:

Error correction techniques are of two types.

1. Single bit Error correction
2. Burst error correction

1. Single bit error correction is the process of correcting single bit error.

2. Burst error correction is method of correcting and detecting burst errors in data sequence.
   → More than one bit change occurs.

Note:
If the bandwidth of a channel is 1Gbps then for how much duration the error should last?



10^9 bits – 1 sec

1 bit – $\frac{1}{10^9}$ sec = 1ns

Single Parity

→ m+1 bits
→ even parity
→ Can detect all single bit errors in codeword.
→ can detect all odd no. of errors also.

Scanned with OKEN Scanner

Hamming Code:

Hamming code is single error correcting code.

No. of parity bits can be obtained from this formula. $2^P \geq P+m+1$

where P = parity bits

m = no. of message bits.

For security purpose & easy reception we are adding parity bits.

Let us transmit 4 bit digital data.

Substitute $m = 4$

$$2^P \geq P+4+1$$

$$2^P \geq P+5$$

To make the above eqn equal.

$P=3$; $2^3 \geq 3+5$

$8 \geq 8$

Atleast 3 parity bits have to be considered.

So now the data becomes 7 bit (4+3).

We have to add parity bits in the place of $2^0, 2^1, 2^2 \ldots 2^n$ places.

(7, 4)

↓ → No. of message bits

No. of data bits being transmitted

Scanned with OKEN Scanner

2^0	2^1		2^2		2^3	
1	2	3	4	5	6	7
P_1	P_2	m_1	P_3	m_2	m_3	m_4

Eg: A (7,4) hamming code is received as 1110000. Determine the corrected code when even parity is there.

1	2	3	4	5	6	7
1	1	1	0	0	0	0

By using reference places

E_3

E_2

E_1

No. of parity bits = no. of errors combinations

(7, 4) → length of message code

↓

length of hamming code

Error places = 7-4 = 3

Even parity

E_3 - 4,5,6,7 = 0000 $E_3 = 0$

E_2 - 2,3,6,7 = 1100 $E_2 = 0$

E_1 - 1,3,5,7 = 1100 $E_1 = 0$

Scanned with OKEN Scanner

$E_3E_2E_1 = 000 - '0'$

As it is '0' there is no error.

Let us do the same example for odd parity.

1	2	3	4	5	6	7
1	1	1	0	0	0	0

odd parity

E_3 – 4,5,6,7 – 0000 → $E_3 - 1$

E_2 – 2,3,6,7 – 1100 → $E_2 - 1$

E_1 – 1,3,5,7 – 1100 → $E_1 - 1$

Error $= E_3E_2E_1 = 111 = 7$

In 7th place error is present.

So the corrected message is.,

1110001

Construct (7,4) hamming code for the message (1000), Consider even parity.

1	2	3	4	5	6	7
P_1	P_2	m_1	P_3	m_2	m_3	m_4
P_1	P_2	1	P_3	0	0	0

Scanned with OKEN Scanner

$P_1 \rightarrow 1, 3, 5, 7$ for even parity

$P_1 100 \Rightarrow P_1 = 1$

$P_2 \rightarrow 2, 3, 6, 7$ for even parity

$P_2 100 \Rightarrow P_2 = 1$

$P_3 \rightarrow 4, 5, 6, 7$ for even parity

$P_3 000 \Rightarrow P_3 = 0$

So the hamming code is

P_1	P_2	1	P_3	0	0	0
1	1	1	0	0	0	0

Construct (7,4) hamming code for the message 1010 such that odd parity exists.

1	2	3	4	5	6	7
P_1	P_2	1	P_3	0	1	0

$P_1 \rightarrow 1, 3, 5, 7$ for odd parity $P_1 = 0$

$P_1 100$

$P_2 \rightarrow 2, 3, 6, 7$ for odd parity $P_2 = 1$

$P_2 110$

$P_3 \rightarrow 4, 5, 6, 7$ for odd parity $P_3 = 0$

$P_3 010$

Hamming code is

P_1	P_2	1	P_3	0	1	0
0	1	1	0	0	1	0

Scanned with OKEN Scanner

Generate hamming code for the message 1110

$$2^P \geq P+m+1$$

$m = 4$

$$2^P \geq P+4+1$$

$$2^P \geq P+5$$

if $P = 3$

$$2^3 \geq 3+5$$

$$8 \geq 8$$

2^0	2^1		2^2			
1	2	3	4	5	6	7
P_1	P_2	1	P_3	1	1	0

$P_1 \rightarrow 1,3,5,7 \rightarrow P_1 110 \rightarrow$ for even parity $P_1 = 0$

$P_2 \rightarrow 2,3,6,7 \rightarrow P_2 110 \rightarrow$ for even parity $P_2 = 0$

$P_3 \rightarrow 4,5,6,7 \rightarrow P_3 110 \rightarrow$ for even parity $P_3 = 0$

P_1	P_2	1	P_3	1	1	0
0	0	1	0	1	1	0

Hamming code:

It is developed by R.W. Hamming.

It not only identifies error bit in whole data sequence and it also corrects it.

Scanned with OKEN Scanner

This code uses a number of parity bits located at certain positions in codeword.

No. of parity bits depends upon no. of information bits.

Hamming code uses the relation between redundancy bits and data bits and this code can be applied to any number of data bits.

Redundancy bit:

Redundancy means the difference between number of bits of actual data sequence and transmitted bits.

In Hamming code redundancy bits are placed at certain calculated positions in order to eliminate errors.

The distance between two redundancy bits is called Hamming distance.

No. of Parity Bits:

The no. of parity bits to be added to a data string depends upon the no. of information bits of data string which is to be transmitted.

No. of parity bits will be calculated by using data bits.

Scanned with OKEN Scanner

Where to place these Parity Bits:

We should know the appropriate positions to place them in information string to provide single bit error correction.

We have 4 data bits and 3 parity bits.

Total codeword to be transmitted is of 7 bits (4+3).

We generally represent data sequence from right to left.

bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0

Parity bits are located at positions of power of two. i.e., 1, 2, 4, 8 and 16.

0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

1, 3, 5, 7

2, 3, 6, 7

4, 5, 6, 7

Scanned with OKEN Scanner

Rules to determine the Parity sequence:

Rule 1:

The value of the parity bit is determined by the sequence of bits that is alternatively checked and skipped.

$P_{1,3,5,7}$

Rule 2:

To determine the value of parity bit P_2, the rule is to check two bits and skip two bits.

$P_{2,3,6,7}$.

Rule 3:

Here the rule is check four bits, skip four bits.

$P_{3,5,6,7}$.

Encode a binary word 11001 into the even parity hamming code.

Given no. of data bits, $m = 5$

Let us try $P = 4$

$$2^4 \geq 5 + 4 + 1$$

$$16 \geq 10$$

Equation is satisfied.

So the codeword will be of 9 bits.

$(9, 5) \rightarrow$ hamming code.

Scanned with OKEN Scanner

2^0	2^1		2^2				2^3	
1	2	3	4	5	6	7	8	9
P_1	P_2	m_1	P_3	m_2	m_3	m_4	P_4	m_5
		1		1	0	0		1

Parity bits:-

$P_1 = 1, 3, 5, 7, 9$ even parity

$P_1 1 1 0 - 1$

$P_2 = 2, 3, 6, 7$

$P_2 1 0 0 - 1$

$P_3 = 3, 5, 6, 7$

$1 1 0 0 - 0$

$P_4 = 8, 9$

$P_4 1 - 1$

1 1 1 0 1 0 0 1 1

0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1

Scanned with OKEN Scanner

Encode the data 1011001 using even parity Hamming code. Generate the code.

Given message = 1011001

No. of message bits $m = 7$

No. of parity bits $\Rightarrow 2^P \geq m + P + 1$

let $P = 3$

$2^3 \geq 7 + P + 1$

$8 \geq 8 + 3$

$8 \ngeq 11$

Not satisfied.

let $P = 4$

$2^4 \geq 7 + 4 + 1$

$16 \geq 12$ (satisfied)

True

$\therefore$ No. of parity bits $P = 4$.,

Total number of bits $= m + P = 7 + 4 = 11$.

			2^3				2^2		2^1	2^0
11	10	9	8	7	6	5	4	3	2	1
						m_2	P_3	m_1	P_2	P_1
m_7	m_6	m_5	P_4	m_4	m_3		P_3	1	P_2	P_1
1	0	1	P_4	1	0	0				

$P_1 = 1, 3, 5, 7, 9, 11 = P_1\ 10111$ even parity

0

$\therefore P_1 = 0$

$P_2 = 2, 3, 6, 7, 10, 11 = P_2\ 10101$ 1

$\therefore P_2 = 1$

Scanned with OKEN Scanner

$P_3 = 4,5,6,7 = P_3\,0\,0\,1$ even parity

1

$\therefore P_3 = 1$

0

$P_4 = 8,9,10,11 = P_4\,1\,0\,1$

$\therefore P_4 = 0$

Total message : 1 0 1 0 1 0 0 1 1 1 0

$\therefore P_4 P_3 P_2 1 = 0\,1\,1\,0 = 6$

Error is @ 6th place.

6	5	4	3	2	1
0	0	1	1	1	0

1 0 1 0 1

Corrected message is

1 0 1 0 1 1 0 1 1 1 0

7 bit Hamming code is received as 1011011. Assume even parity and state whether received code is correct or not. Locate the bit in error.

Received hamming code 1011011

7	6	5	4	3	2	1
1	0	1	1	0	1	1
			P_3		P_2	P_1

Detecting errors:

① Analyse P_1

$1,3,5,7$ — P_1 1011 even

1

Analyse P_2

2, 3, 6, 7	even
1 0 0 1	0

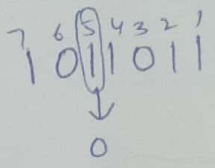
Analyse P_3

4, 5, 6, 7	even
1 1 0 1	1

Error : $P_3 P_2 P_1$

1 0 1 $\rightarrow$ 5

Error exists @ 5th place.



Corrected code is 1001011.

7 bit Hamming code is received as 1101100. Assume even parity and state whether received code is correct or wrong.

Received Hamming code

7	6	5	4	3	2	1
1	1	0	1	1	0	0
			P_3		P_2	P_1

Analyse P_1

1 3 5 7	even
P_1	0
0 1 0 1	

Analyse P_2 1

2 3 6 7

0 1 1 1

Analyse P_3

4	5	6	7	even
1	0	1	1	1

$E_3 E_2 E_1$

$P_3 P_2 P_1 = 110 \rightarrow 6^{th}$

Received code is wrong as error is detected.

1 (1) 0 1 1 0 0

↓

1 0 0 1 1 0 0 → Corrected code.

---

7 bit Hamming code is received as 1011011. Assume even parity and state whether received code is correct or wrong. If wrong locate the bit in error.

Received Hamming code 1011011

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

Detecting Errors:

1. Analyze P1:

Least Significant bit 1

P1	D3	D5	D7
1	0	1	1

Parity bit is one.

P1 = 1

2. Analyze P2:

1 in Second Position

P2	D3	D6	D7
1	0	0	1

P2 = 0

3. Analyze P3

1 in third Position

P3	D5	D6	D7
1	1	0	1

P4 = 1

P1 and P4 are not equal to zero

So Error Exists

Correcting Errors

Error Word P4 P2 P1

1 0 1

Decimal Value of 101 is 5.

So 5th bit is Error

We can write correct code word by simply inverting 5th bit

1001011.

7 bit Hamming code is received as 1101100. Assume even parity and state whether received code is correct or wrong. If wrong locate the error bit

Received Hamming Code.

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

Analyze P1:

P1	D3	D5	D7
0	1	0	1

P1 = 0.

Analyze P2

P2	D3	D6	D7
0	1	1	1

P2 = 1

Analyze P4

P4	D5	D6	D7
1	0	1	1

P4 = 1

P2 and P4 are not equal to zero

Received Wrong Code

Error is Detected.

Correcting Errors:

Error Word	P4	P2	P1
	1	1	0

Decimal Value of 110 is 6.

So 6th bit is error

We can write correct code word by simply inverting 6th bit

1001100.

Boolean Algebra

→ Boolean Algebra is used to analyze and simplify logic circuits. It uses binary numbers 0 and 1. Also called Binary Algebra or Logical Algebra.

1. Associative Law

$(A.B).C = A.(B.C)$

$(A+B)+C = A+(B+C)$

2. Commutative Law

$A.B = B.A$

$A+B = B+A$

3. Distributive Law

$A+(B.C) = (A+B).(A+C)$

$A.(B+C) = (A.B)+(A.C)$

4. Absorption Law

$B+(B.A) = B$

$B.(B+A) = B$

5. Demorgan Law

$(A.B)' = A'+B'$

$(A+B)' = A'.B'$

6. Complement Law

$B.B' = 0$

$B+B' = 1$

7. Double Negation Law

$(A')' = A$

8. Idempotent Law

$B.B = B$

$B+B = B$

9. Identity Law

$B.1 = B$

$B+0 = B$

10. Annulment Law

$B.0 = 0$

$B+1 = 1$

Basic Algebraic Properties:

1. A set is a collection of objects with a common property.
   If S is a set and x is a member of set S then $x \in S$.
2. A binary operator on a set S is a rule that assigns to each pair of elements in S another element that is in S.
3. Axioms are the assumption that are valid without proof.

Examples of Axioms:

1. Closure:
   A set S is closed with respect to binary operator.
   iff for all $x, y \in S$.
   $$(x \cdot y) \in S$$
2. Associativity:
   A binary operator defined on a set S is associative
   iff for all $x, y, z \in S$
   $$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

3. Identity Element

A set S has an identity element e iff for every $x \in S$

$$e \cdot x = x \cdot e = x$$

4. Commutativity

A binary operator . is commutative iff for all $x, y \in S$

$$x \cdot y = y \cdot x$$

5. Inverse Element:

A set S has an inverse, iff for every $x \in S$, there exists an element $y \in S$ such that

$$x \cdot y = e$$

6. Distributive.

If $*$ and $+$ are two binary operators on a set S . is said to be distributive over $*$ if for all $x, y, z \in S$.

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

Axiomatic Definitions of Boolean Algebra

Boolean Algebra is a set of elements B. with two binary operators + and . which satisfies following six axioms.

1. Closure:

B is closed with respect to operator +

B is closed with respect to operator .

2. Identity:

→ B has an identity element with respect to +, designated by 0.

→ B also has an identity element with respect to . designated by 1.

3. Commutativity:

B is commutative with respect to +, .

4. Distributive:

The operator . is distributed over +.

" " + " " " .

5. Complement:

For every $x \in B$ there exists an element $x' \in B$ such that

$$x + x' = 1$$

$$x \cdot x' = 0$$

6. Cardinality:

There are atleast two elements $x, y \in B$ such that

$$x \neq y$$

Difference between Boolean Algebra and Ordinary Algebra.

1. In ordinary Algebra, + is not distributed over .
2. Boolean algebra does not have inverse with respect to + and . these are no subtraction or division operations in boolean algebra.
3. Complements are available in BA but not in OA
4. BA applies to finite set of elements whereas OA would apply to infinite set of real numbers.
5. BA does not include associativity

Two Valued Boolean Algebra:

-> Set B has two elements: 0 and 1

-> Algebra has two operators: AND and OR.

x	y	x.y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Two valued boolean algebra satisfies Huntington axioms.

1. Closure:

Closure is evident in AND/OR tables since the result of each operation is an element of B.

2. Identity:

Identity elements in this algebra are:

0 for the operator +

1 for the operator .

$0+0=0$ $\quad 1\cdot 1=1$

$0+1=1$ $\quad 1\cdot 0=0\cdot 1=0$

$1+0=1$

3. Commutative:

4. Distributive:

$x\cdot(y+z)=(x\cdot y)+(x\cdot z)$

x	y	z	y+z	x·(y+z)	(x·y)	(x·z)	(x·y)+(x·z)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
1	1	0	1	1	0	1	1
1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1

$x+(y\cdot z) = (x+y)\cdot(x+z)$

x	y	z	$y\cdot z$	$x+(y\cdot z)$	$x+y$	$x+z$	$(x+y)\cdot(x+z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

5. Complement:

0 and 1 are complements of each other.

$0+0' = 0+1 = 1$ $\quad$ $0\cdot 0' = 0\cdot 1 = 0$

$1+1' = 1+0 = 1$ $\quad$ $1\cdot 1' = 1\cdot 0 = 0$

6. Cardinality:

Cardinality axiom is satisfied, two valued BA has two distinct elements 1 and 0 and $1 \neq 0$.

Boolean Operator Precedence:

1. Parentheses ( )
2. NOT '
3. AND .
4. OR +

Eg: Evaluate expression $(x+xy)'$ for $x=1$ and $y=0$

$(1+1\cdot0)'$

$(1+0)'$

$(1)'$

0

Duality Principle:

We simply interchange OR and AND operations and replace is by 0's and 0s by 1's

Theorems of Boolean Algebra:

Theorem 1 (Idempotency):

a. $x+x=x$

$x+x = (x+x)\cdot 1$ 2b

$=(x+x)\cdot(x+x')$ 5a

$= x+xx'$ 4b.

$= x+0$ 5b

$= x$ 2a.

b. $xx = x$

$xx+0$ 2a

$xx+xx'$ 5b

$x(x+x') = x\cdot 1 = x$

4a 5a 2b.

Theorem 2:

(a) $x+1 = 1$

$x+1 = 1 \cdot (x+1)$ 2b

$= (x+x')(x+1)$ 5a

$= x + x' \cdot 1$ 4b

$= x + x'$ 2b

$= 1$ 5a.

(b) $x \cdot 0 = 0$

By duality

Theorem 3 (Absorption):

a. $x + xy = x$

$x + xy = x \cdot 1 + xy$ 2b

$= x(1+y)$ 4a

$= x(1+y)$ 3a

$= x \cdot 1$ 2a

$= x$ 2b.

b. $(y+x)\,x = x$

$(y+x)\,x =$

Theorem 4 (Involution):

$$(x')' = x$$

Theorem 5 : Associative

a. $(x+y)+z = x+(y+z)$

b. $x(yz) = (xy)z$

Theorem 6 : Demorgans Law

a. $(x+y)' = x' \cdot y'$.

b. $(xy)' = x'+y'$.

a.

x	y	$x+y$	$(x+y)'$	x'	y'	$x' \cdot y'$
0	0	0	1			
0	1	1	0			
1	0	1	0			
1	1	1	0			

b.

x	y	xy	$(xy)'$	x'	y'	$x'+y'$
0	0	0	1			
0	1	0	1			
1	0	0	1			
1	1	1	0			

Venn Diagram:

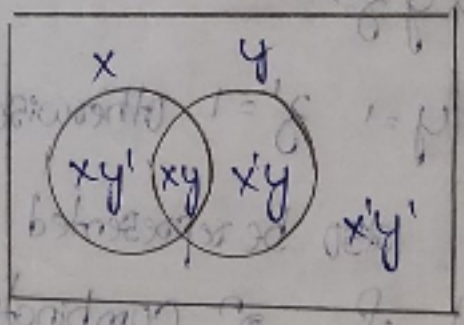
→ To visualize relationships among variables of boolean expression.

→ This diagram consists of rectangles inside of which are drawn overlapping circles. one for each variable.

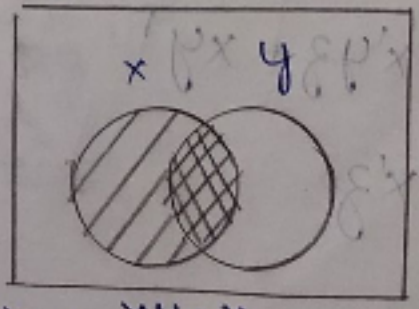
→ If we are inside circle say $x = 1$. When outside we say $x = 0$.

→ for two overlapping circles there are four distinct areas.

1. The area not belonging to either x or y $x'y'$
2. The area inside circle y but outside x. $x'y$
3. " " " x but " y xy'
4. Area inside both circles xy



→ Venn Diagrams may be used to illustrate the postulates of boolean algebra or to show validity of theorems.



$x = xy + x$

Boolean Functions:

→ A binary variable can take the value 0 or 1.

→ A boolean function is an expression formed with binary variables, two binary operators OR and AND unary operator NOT, parantheses and equal sign.

$F_1 = xyz'$.

$F_1 = 1$ if $x = 1$ $y = 1$ $z' = 1$ otherwise $F_1 = 0$.

→ Boolean Function also be represented in truth table

We need a list of 2^n combinations of 1s and 0's of n binary variables

Eg: $F_2 = x + y'z$

$F_3 = x'y'z + x'yz + xy'$

$F_4 = xy' + x'z$.

x	y	z	x'	y'	z'	xyz'	y'z	x+y'z
0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	1	1
0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	0	1	0	0	1	1
1	1	0	0	0	1	1	0	1
1	1	1	0	0	0	0	0	0

Simplify a boolean function to a minimum number of literals

1. $x+x'y$

$(x+x')(x+y)$

$1.(x+y)$

$(x+y)$

2. $x(x'+y)$

$xx'+xy$

$0+xy$

xy

3. $x'y'z+x'yz+xy'$

$x'z(y+y')+xy'$

$x'z(1)+xy'$

$x'z+xy'$

4. $xy+x'z+yz$

$xy+x'z+yz(x+x')$

$xy+x'z+xyz+x'yz$

$xy(1+z)+x'z(1+y)$

$xy+x'z$

5. $(x+y)(x'+z)(y+z)$

By duality from 4.

$(x+y).(x'+z)$

Complement of a function

-> Complement of a function F is F', and is obtained by interchanging 1's and 0's.

-> The complement of a function may be derived algebraically through Demorgans theorem.

$(A+B+C)'$

$(A+x)'$ $\quad x = B+C$

$A' \cdot x'$

$A' \cdot (B+C)'$

$A' \cdot (B' \cdot C')$

$A' \cdot B' \cdot C'$

Find the complement of the function

$F = x'y'z' + x'y'z$

$F_1' = [x'y'z' + x'y'z]'$

$= (x'y'z')' \cdot (x'y'z)'$

$(a+b)' = a' \cdot b'$

$(a \cdot b)' = a' + b'$

$= [(x')' + y' + (z')'] \cdot [(x')' + (y')' + z']$

$= (x + y' + z)(x + y + z')$

$f_2 = x(y'z' + yz)$.

$f_2' = [x(y'z' + yz)]'$

$= x' + (y'z' + yz)'$

$= x' + (y'z')' \cdot (yz)'$.

$= x' + (y')' + (z')' \cdot y' + z'$

$= x' + (y+z)(y'+z')$.

Canonical and Standard forms:

Canonical form:

→ In BA, boolean function can be expressed as canonical Disjunctive Normal form known as minterm and some expressed as canonical Conjunctive Normal form known as maxterm.

→ In minterm we looks for the functions where the output results in 1.

→ In maxterm we looks for function where output results in 0.

→ Sum of minterms also known as Sum of Products (SoP)

→ Product of maxterms also known as Product of Sum (PoS)

Boolean functions expressed as sum of minterms or product of maxterms are said to be in canonical form.

Standard form:

A boolean variable can be expressed in either true form or complemented form.

In standard form boolean function will contain all variables in either true form or complemented form.

In canonical no. of variables depends on output of SOP or POS.

x	y	z	Minterms		Maxterms
0	0	0	$x'y'z'$	m_0	$x+y+z$
0	0	1	$x'y'z$	m_1	$x+y+z'$
0	1	0	$x'yz'$	m_2	$x'+y'+z$
0	1	1	$x'yz$	m_3	$x+y'+z'$
1	0	0	$xy'z'$	m_4	$x'+y+z$
1	0	1	$xy'z$	m_5	$x'+y+z'$
1	1	0	xyz'	m_6	$x'+y'+z$
1	1	1	xyz	m_7	$x'+y'+z'$

Product format — Sum format

B'	BC	A+BC
1	0	0
1	1	1
0	0	0
0	0	0
1	0	1
1	1	1
0	0	1
0	0	1

Sum of Minterms:

Consider those inputs whose output is 1

Input zero with complement

Input one without complement.

Express the boolean function F = A+B'C in sum of minterms.

$F = A + B'C$

$= A(B+B') + B'C(A+A')$

$= AB + AB' + AB'C + A'B'C$

$= AB(C+C') + AB'(C+C') + AB'C + A'B'C$

$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$

$= ABC + ABC' + AB'C + AB'C' + A'B'C$

$F(A,B,C) = \Sigma(1,4,5,6,7)$.

Product of Maxterms:

Consider those inputs whose output is 0.

Input zero without Complement

Input one with Complement

Express the boolean function F = xy + x'z in POS form.

$F = xy + x'z$

$= (xy + x')(xy + z)$

$= (x+x')(y+x')(x+z)(y+z)$

$= (x'+y)(x+z)(y+z)$

$= (x'+y+zz')(x+z+yy')(y+z+xx')$

$= (x'+y+z)(x'+y+z')(x+y+z)(x+y'+z)$

$(x+y+z)(x'+y+z)$

$(x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$

$M_0 M_2 M_4 M_5$.

$F(x,y,z) = \Pi(0,2,4,5)$.

Conversion between Canonical Forms:

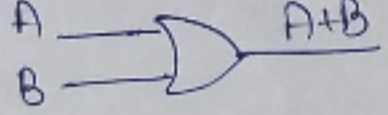
To convert from one canonical form to another interchange the symbols Σ and Π and list those numbers missing from original form.

Eg: $F(x,y,z) = \Pi(0,2,4,5)$

$F(x,y,z) = \Sigma(1,3,6,7)$.

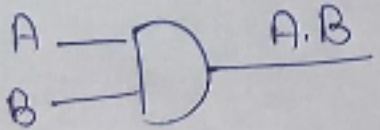
Digital Logic Gates:

1. OR gate



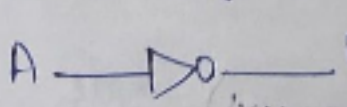
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

2. AND gate



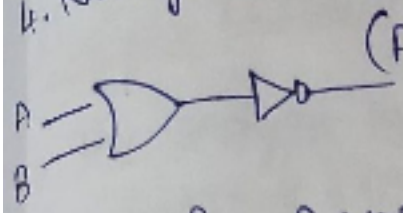
A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

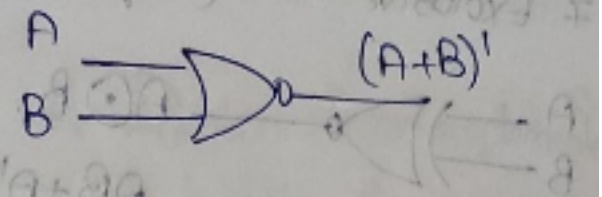
3. NOT gate



A	A'
0	1
1	0

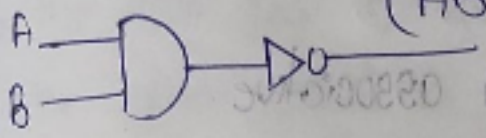
4. NOR gate:

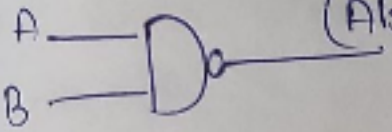




A	B	A NOR B (A ↓ B)
0	0	1
0	1	0
1	0	0
1	1	0

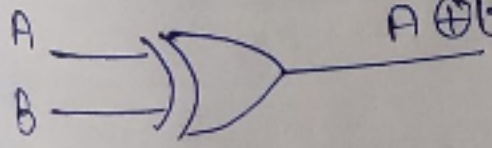
5. NAND gate:





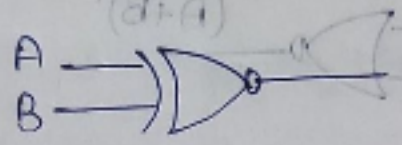
A	B	A NAND B (A ↑ B)
0	0	1
0	1	1
1	0	1
1	1	0

6. Exclusive OR (XOR)



A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

7 Exclusive NOR (equivalence)



$AB + A'B'$

A	B	A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

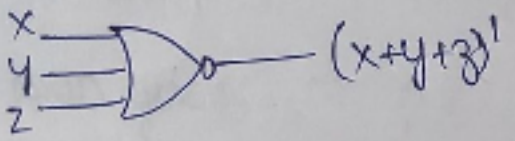
→ NAND and NOR functions are commutative

→ NAND and NOR operators are not associative

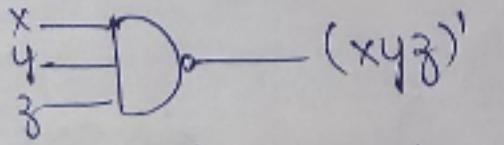
$$(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$$

$$x \downarrow y \downarrow z = (x+y+z)'$$

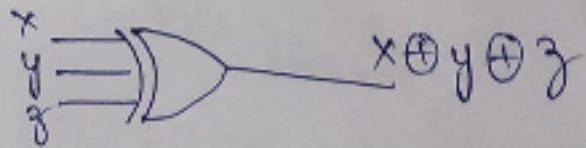
$$x \uparrow y \uparrow z = (xyz)'$$



Three input NOR gate



Three input NAND gate.



x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1