UNIT-1

Review of Number systems & codes (1)

There are fourtypes of number systems in digital electronics

① Decimal number system (Base 10)
② Binary number system (Base 2)
③ Octal number system (Base 8)
④ Hexa-decimal number system (Base 16)

Base is also known as radix

If the base of any number system describes the no. of distinct symbols and the highest digit/number in that number system.

Ex: In hexadecimal number system, there are 16 symbols they are

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

* List the first 20 numbers in hexadecimal number system

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13

* List the first 15 numbers in Base 12 system

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, 10, 11, 12

* List the first 10 numbers in octal number system

0, 1, 2, 3, 4, 5, 6, 7, 10, 11

Base Conversions:

To Convert $(\ )_{10}$ to $(\ )_r$ perform the successive division of the given decimal number with required base upto the coefficient is less than the base.

Ex: Convert $(48)_{10} = (?)_2$

(2)

$$\begin{array}{r|l} 2 & 48 \\ & 24-0 \\ & 12-0 \\ & 6-0 \\ & 3-0 \\ & 1-1 \end{array}$$

→ $(48)_{10} = (110000)_2$

* Convert $(71)_{10} = (?)_2$

$$\begin{array}{r|l} 2 & 71 \\ & 35-1 \\ & 17-1 \\ & 8-1 \\ & 4-0 \\ & 2-0 \\ & 1-0 \end{array}$$

$(71)_{10} = (1000111)_2$

* Convert $(524)_{10} = (?)_2$

$$\begin{array}{r|l} 2 & 524 \\ & 262-0 \\ & 131-0 \\ & 65-1 \\ & 32-1 \\ & 16-0 \\ & 8-0 \\ & 4-0 \\ & 2-0 \\ & 1-0 \end{array}$$

$(524)_{10} = (1000001100)_2$

* Convert $(461)_{10} = (?)_2$

$$\begin{array}{|l} 461 \\ 230-1 \\ 115-0 \\ 57-1 \\ 28-1 \\ 14-0 \\ 7-0 \\ 3-1 \\ 1-1 \end{array}$$

$(461)_{10} = (111001101)_2$

* Convert $(10.125)_{10} = (?)_2$

$$\begin{array}{r|l} 2 & 10 \\ 2 & 5-0 \\ 2 & 2-1 \\ & 1-0 \end{array}$$

$.125 \times 2 = 0.250$

$.250 \times 2 = 0.500$

$.500 \times 2 = 1.000$

$(10.125)_{10} = (1010.001)_2$

* Convert $(39.225)_{10} = (?)_2$

$$\begin{array}{r|l} 2 & 39 \\ 2 & 19 - 1 \\ 2 & 9 - 1 \\ 2 & 4 - 1 \\ 2 & 2 - 0 \\ & 1 - 0 \end{array}$$

$.225 \times 2 = 0.450$
$.450 \times 2 = 0.900$
$.900 \times 2 = 1.800$
$.800 \times 2 = 1.600$

$(39.225)_{10} = (100111.0011)_2$

* Convert $(63.525)_{10} = (?)_2$

$$\begin{array}{r|l} 2 & 63 \\ 2 & 31 - 1 \\ 2 & 15 - 1 \\ 2 & 7 - 1 \\ 2 & 3 - 1 \\ & 1 - 1 \end{array}$$

$.525 \times 2 = 1.050$
$.050 \times 2 = 0.100$
$.100 \times 2 = 0.200$
$.200 \times 2 = 0.400$
$.400 \times 2 = 0.800$

$(63.525)_{10} = (111111.1000)_2$

⇒ Conversion of decimal to Octal

* Convert $(25)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 25 \\ & 3 - 1 \end{array}$$

$(25)_{10} = (31)_8$

* Convert $(125)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 125 \\ 8 & 15 - 5 \\ & 1 - 7 \end{array}$$

$(125)_{10} = (175)_8$

* Convert $(93)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 93 \\ 8 & 11 - 5 \\ & 1 - 3 \end{array}$$

$(93)_{10} = (135)_8$

* Convert $(154.125)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 154 \\ 8 & 19 - 2 \\ & 2 - 3 \end{array}$$

$.125 \times 8 = 1.000$

$(154.125)_{10} = (232.1)_8$

* Convert $(23.225)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 23 \\ & 2 - 7 \end{array}$$

$.225 \times 8 = 1.800$
$.800 \times 8 = 6.400$
$.400 \times 8 = 3.200$
$.200 \times 8 = 1.600$

$(23.225)_{10} = (27.1631)_8$

⇒ Conversion from decimal to hexa decimal

4

* Convert $(31)_{10} = (?)_{16}$

$$\begin{array}{r|l} 16 & 31 \\ \hline & 1-F \end{array}$$

$(31)_{10} = (1F)_{16}$

* convert $(49)_{10} = (?)_{16}$

$$\begin{array}{r|l} 16 & 49 \\ \hline & 3-1 \end{array}$$

$(49)_{10} = (31)_{16}$

* convert $(62)_{10} = (?)_{16}$

$$\begin{array}{r|l} 16 & 62 \\ \hline & 3-E \end{array}$$

$(62)_{10} = (3E)_{16}$

* Convert $(74)_{10} = (?)_{16}$

$$\begin{array}{r|l} 16 & 74 \\ \hline & 4-A \end{array}$$

$(74)_{10} = (4A)_{16}$

* Convert $(70.250)_{10} = (?)_{16}$

$$\begin{array}{r|l} 16 & 70 \\ \hline & 4-6 \end{array}$$

$.250 \times 16 = 4.000$

$(70.250)_{10} = (46.4)_{16}$

* $(92.150)_{10} = (?)_{16}$

$$\begin{array}{r|l} 16 & 92 \\ \hline & 5-6 \end{array}$$

$.150 \times 16 = 2.400$

$.400 \times 16 = 6.400$

$.400 \times 16 = 6.400$

$.400 \times 16 = 6.400$

$(92.150)_{10} = (5C.2666)_{16}$

* convert $(10.125)_{10} = (?)_{4}$

$$\begin{array}{r|l} 4 & 10 \\ \hline & 2-2 \end{array}$$

$.125 \times 4 = 0.500$

$.500 \times 4 = 2.000$

$(10.125)_{10} = (22.02)_{4}$

r to 10 conversions

Binary to decimal conversion

5

→ $(110000)_2 = (?)_{10}$

$1\times2^5+1\times2^4+0\times2^3+0\times2^2+0\times2^1+0\times2^0$

$\Rightarrow 32+16$

$\rightarrow \boxed{(48)_{10}}$

→ $(1000001100)_2$

$1\times2^9+0+1\times2^3+1\times2^2$

$512+8+4$

$\boxed{(524)_{10}}$

→ $(1000111)_2$

$1\times2^6+0+1\times2^2+1\times2^1+1\times2^0$

$64+4+2+1$

$68+3$

$\boxed{(71)_{10}}$

→ $(111001101)_2$

$1\times2^8+1\times2^7+1\times2^6+1\times2^3+1\times2^2+1\times2^0$

$256+128+64+8+4+2+1$

$\boxed{(461)_{10}}$

→ $(1010.001)_2 = (?)_{10}$

$1\times2^3+1\times2^1+0\times2^{-1}+1\times2^{-3}$

$8+2+\frac{1}{2^3} \Rightarrow 10+0.125$

$\boxed{(10.125)_{10}}$

→ $(100111.0011)_2 = (?)_{10}$

$1\times2^5+1\times2^2+1\times2^1+1\times2^0+1\times2^{-3}+1\times2^{-4}$

$\Rightarrow 32+4+2+1+\frac{1}{2^3}+\frac{1}{2^4}$

$\Rightarrow 39+0.125+0.0625 \Rightarrow (39.1875)_{10} \simeq \boxed{(39.225)}$

→ $(111111.1000)_2 = (?)_{10}$

$1\times2^5+1\times2^4+1\times2^3+1\times2^2+1\times2^1+1\times2^0+1\times2^{-1}$

$32+16+8+4+2+1+0.5 \Rightarrow \boxed{(63.5)_{10}}$

Octal to decimal conversion:

* $(31)_8 = (?)_{10}$

$8^1 \times 3 + 1 \times 8^0$

$24 + 1$

$\boxed{(25)_{10}}$

* $(175)_8 = (?)_{10}$

$1 \times 8^2 + 7 \times 8^1 + 5 \times 8^0$

$64 + 56 + 5$

$\boxed{(125)_{10}}$

* $(135)_8 = (?)_{10}$

$1 \times 8^2 + 3 \times 8^1 + 5 \times 8^0$

$64 + 24 + 5$

$\boxed{(93)_{10}}$

* $(27.1631)_8 = (?)_{10}$

$2 \times 8^1 + 7 \times 8^0 + 1 \times 8^{-1} + 6 \times 8^{-2} + 3 \times 8^{-3} + 8^{-4} \times 1$

$16 + 7 + 0.125 + 0.09375 + 0.00585 + 0.000024$

$\boxed{(23.2248)_{10}}$

* $(232.1)_8 = (?)_{10}$

$2 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 1 \times 8^{-1}$

$128 + 24 + 2 \times \frac{1}{8}$

$154 + 0.125$

$\boxed{154.125}$

* hexadecimal to decimal

→ $(1F)_{16} = (?)_{10}$

$1 \times 16^1 + 15 \times 16^0$

$16 + 15 \Rightarrow \boxed{(31)_{10}}$

→ $(3E)_{16} = (?)_{10}$

$3 \times 16^1 + E \times 16^0$

$48 + 14 \times 1$

$\boxed{(62)_{10}}$

→ $(4A)_{16} = (?)_{10}$

$4 \times 16^1 + A \times 16^0$

$64 + 10 \times 1$

$\boxed{(74)_{10}}$

→ $(31)_{16} = (?)_{10}$

$16^1 \times 3 + 16^0 \times 1$

$48 + 1 = \boxed{(49)_{10}}$

→ $(46.4)_{16}$

$4 \times 16^1 + 6 \times 16^0 + 4 \times \frac{1}{16}$

$64 + 6 + 0.250 \Rightarrow \boxed{(70.250)_{10}}$

* $(5C.2666)_{16} = (?)_{10}$

$5 \times 16^1 + 12 \times 16^0 + 2 \times 16^{-1} + 6 \times 16^{-2} + 6 \times 16^{-3} + 6 \times 16^{-4}$

$80 + 12 + 0.125 + 0.0234 + 0.0014 + 0.00009$

$\Rightarrow (92.1498)_{10} \approx \boxed{(92.150)_{10}}$

Base 4 to Base 10

* $(22.02)_4$

$2\times4^1 + 2\times4^0 + 0 + 2\times4^{-2}$

$8+2+0+2\times\frac{1}{16}$

$10+0.125$

$\boxed{(10.125)_{10}}$

r-r conversions:

In binary to octal conversion segment the given binary number as group and each group should contain 3 bit. Replace each group with its octal equivalent.

3-bit binary code

2^2	2^1	2^0	
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

$2^3 = 8$

* Convert $(1011010)_2$ to $(?)_8$

sol

001 | 011 | 010

1 3 2

$\boxed{(132)_8}$

* $(1110101101101)_2$ to $(?)_8$

001 | 110 | 101 | 101 | 101

1 6 5 5 5

$\boxed{(16555)_8}$

Scanned with CamScanner

* $(1110111111110110.011011)_2 = (?)_8$. (8)

111	011	111	110	110	.	011	011	100
7	3	7	6	6	.	3	3	4

$\boxed{(73766.334)_8}$

Binary to hexadecimal:-

4 bit binary code

2^3	2^2	2^1	2^0	
8	4	2	1	
0	0	0	0	- 0
0	0	0	1	- 1
0	0	1	0	- 2
0	0	1	1	- 3
0	1	0	0	- 4
0	1	0	1	- 5
0	1	1	0	- 6
0	1	1	1	- 7
1	0	0	0	- 8
1	0	0	1	- 9
1	0	1	0	- 10 - A
1	0	1	1	- 11 - B
1	1	0	0	- 12 - C
1	1	0	1	- 13 - D
1	1	1	0	- 14 - E
1	1	1	1	- 15 - F

$2^4 = 16$

* $(11001010)_2 = (?)_{16}$ * (10 (9)

Sol

1100 | 1010
C | A

$\boxed{(CA)_{16}}$

* $(10101111010101011)_2 = (?)_{16}$

0010 | 1011 | 1010 | 1011
2 B A B

$\boxed{(2BAB)_{16}}$

* $(1011010\,10.10101011)_2 = (?)_{16}$

0001 | 0110 | 1010 . 1010 | 1011 | 1000
1 6 A A B 8

$\boxed{(16A.AB8)_{16}}$

0 0010000 00010001

Octal to Binary conversion:

* In octal to Binary conversion each digit of octal number should be replaced with its 3-bit binary equivalent

* $(132)_8$

1 | 3 | 2
001 | 011 | 010

$(001011010)_2$.

* $(16555)_8$

1 | 6 | 5 | 5 | 5
001 | 110 | 101

* $(73766.334)_8$

7 | 3 | 7 | 6 | 6 . 3 | 3 | 4
111 | 011 | 111 | 110 | 110 . 011 | 011 | 100

$(111011111110110.011011100)_2$.

Scanned with CamScanner

Hexadecimal to binary:-

(10)

* $(1011)_{16} = (?)_2$

1 | 0 | 1 | 1

0001 | 0000 | 0001 | 0001

$(0001000000010001)_2$

* $(CA)_{16} = (?)_2$

C | A

1100 | 1010

$(11001010)_2$

Octal to hexadecimal Conversion:-

* to convert Octal to hexadecimal convert the octal number to binary and then to hexadecimal.

→ convert $(145)_8 = (?)_{16}$

(a) $(145)_8 = (?)_2$

1 | 4 | 5

001 | 100 | 101

$(001100101)_2$

(b) 0000 | 0110 | 0101

0 6 5

$(065)_{16} = (65)_{16}$

→ $(1372)_8 = (?)_{16}$

(a) $(1372)_8 = (?)_2$.

1	3	7	2
001	011	111	010

$(001011111010)_2$

(b) $(001011111010)_2 = (?)_{16}$.

0010	1111	1010
2	F	A

$(2FA)_{16}$

Hexadecimal to octal

→ $(DAC)_{16} = (?)_8$

(a) $(DAC)_{16} = (?)_2$.

D	A	C
1101	1010	1100

$(110110101100)_2$.

(b) $(110|110|101|100)_2 = (?)_8$

110	110	101	100
6	6	5	4

$(6654)_8$

→ $(1A8)_{16} = (?)_8$

(a) $(1A8)_{16} = (?)_2$

1	A	8
0001	1010	1000

$(000110101000)_2$

(b)

000	110	101	000
0	6	5	0

$(650)_8$

*Note

10 − r → successive division

r − 10 → powers multiplication

r − r → if powers relation exists then grouping and use bit code
else
use intermediate (decimal | binary) to convert

Binary addition:-

→ 8 → 1 0 0 0
7 → 0 1 1 1
15 → 1 1 1 1

Here 2 → 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

→ (carry: 1 1 1)
7 0 1 1 1
7 0 1 1 1
14 1 1 1 0

→ (carry: 1 1 1)
15 1 1 1 1
15 1 1 1 1
1 1 1 1 0

→ (carry: 1 1)
5 0 1 0 1
5 0 1 0 1
10 1 0 1 0

→ (carry: 10 10 10 1)
1 0 1 0
1 1 1 1
0 1 1 1
1 0 0 0 0 0

→ Binary subtraction:

8
-4
4

10
3
07

25
7
18

6 0 1 1 0
2 0 0 1 0
4 0 1 0 0

8 1 0 0 0
7 0 1 1 1
0 0 0 1

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

$$\begin{array}{r} 0\ 1\ 1\ 2 \\ 0\ 1\ 2\ 2 \\ 0\ 2 \\ *\ 16\ \ 1\ 0\ 0\ 0\ 0 \\ 14\ \ 0\ 1\ 1\ 1\ 0 \\ \hline 2\ \ 0\ 0\ 0\ 1\ 0 \end{array} \qquad \begin{array}{r} 0\ 2\ 1\ 2\ 0\ 2 \\ *\ 18\ \ 1\ 0\ 0\ 1\ 0 \\ 13\ \ 0\ 1\ 1\ 0\ 1 \\ \hline 5\ \ 0\ 0\ 1\ 0\ 1 \end{array}$$

Compliments:

For any base "r" system there exists two compliments 1. r's compliment
2. (r-1)'s compliment

* To find r's compliment the following formula is used i.e., $r^n - N$

where r – Base of number system
n – no. of integer digits in the given number
N – The number for which r's compliment to be calculated.

* To find (r-1)'s compliment the following formula is used i.e., $(r^n - 1) - N$

Compliments in binary number system.

1's compliment

→ Find 1's compliment of 101

Sol

$N = 101$

$n = 3,\ r = 2$

Formula: $(r^n - 1) - N$

$(2^3 - 1) - N$

$(8-1) - 101$

$7(111) - 101$

$111 - 101$

$\boxed{010}$

$$\begin{array}{r} 111 \\ 101 \\ \hline 010 \end{array}$$

1's compliment of 101 is 010

→ Find 1's compliment of 1010

simply replace 1's with 0 & 0 with 1.

1's compliment of 1010 is 0101

14

→ 100011

1's compliment is 011100

→ 2's Compliment:

2's compliment can be calculated by adding 1 to 1's compliment of number.

*→ Find the 2's compliment of following number

→ 1101

1's compliment = 0010
Add 1 = 1
2's compliment 0011

→ 1000

1's compliment = 0111
Add 1 = 1
2's compliment 1000

→ 10000

1's compliment = 01111
Add 1 = 1
10000

→ 10110

1's compliment = 01001
1
01010

By directly 01010

→ 10000

1's compliment = 01111
1
10000

use not in sem exams

* To find 2's compliment directly there are two cases.

(i) if units place have zero then from right to left do not change the number until 1's reached then next numbers are interchanged by 0's as 1 & 1's as 0.

(ii) if units place have 1 then not exchange 1 kept it as 1 & then next numbers are interchanged by 0 as 1 & 1 as 0

(i) 1010
2's comp Ans 01010

(ii) 111101
2's comp 000011

Scanned with CamScanner

1's compliment subtraction

If "A-B" is the required operation take the 1's compliment of B & add it to A. After adding there are two cases

Case(i): Carry generated.

If carry generated after the addition of A and 1's compliment of B End-around the carry i.e., add carry to the LSB(least significant bit)

Case(ii): Carry not generated

If carry not generated after the addition, take the 1's compliment of result again and put a '-' sign before the result.

Ex:

```
  9  →  1 0 0 1
(-)4    0 1 0 0
 ---    -------
  5     0 1 0 1
```

case(i)

```
          9 →  1 0 0 1
1's comp of 4 → 1 0 1 1
          (+)   1 1
              ---------
      carry (1) 0 1 0 0
            +         1
              ---------
          5 →   0 1 0 1
```

case(ii).

```
   4   0 1 0 0
(+)9   0 1 1 0
     ---------
       1 0 1 0
     ---------
     - 0 1 0 1 → 1st comp of
```

```
*  8 → 1 0 0 0
   3 → 0 0 1 1
(-)
   5 → 0 1 0 1
```

case(i):

```
 8        1 0 0 0
-3    (+) 1 1 0 0
---      ---------
 5      (1) 0 1 0 0
 carry +          1
         ---------
          0 1 0 1
```

case(ii):

```
3                0 0 1 1
8(1's comp)      0 1 1 1
                ---------
                 1 0 1 0 → carry not generated
                ---------
-5              -0 1 0 1
```

(16)

* 10 → 1010
(−) 7 → 0111
3   0011

10 → 1010
1's comp of 7 (+) 1000
10010
(+) 1
0011

7   0111
10   0101
−3   1100
−0011

* 15 → 1111
(−) 11 → 1011
4   0100

15   1111
1's comp of 11 (+) 0100
10011
+1
4   00100

11   1011
1's of 15   0000
−4   1011
−0100

* 101101
010110

Case (i)

101101
(+) 101001
1010110
+1
010111

010110
(+) 010010
101000
(−010111)

2's compliment subtraction:

If "A-B" is the required operation, takes the Two's Compliment of "B" and Add it to "A". After addition there are two cases.

Case(1):

Carry generated. After adding "A" & 2's Compliment of "B" if Carry is generated neglect discard the Carry

Case(2):

Carry is not generated. After adding "A" & 2's compliment of "B" if Carry is not generated take the 2's Compliment of result again and Put a (-) (minus) sign before it.

Ex:

* 8 → 1000 → 1000
(-) 5 → 0101 → (+) 1011
= 3

(1) 0011 — discard the carry

0011 Ans. 3

5 → 0101 → 0101
(-) 8 → 1000 → (+) 1000
= -3

1101

Ag 2's (-0011) → -3.

* 9 → 1001 → 1001
(-) 4 → 0100 → (+) 1100
= 5.

(1) 0101 — discard

0101 → 5

4 → 0100 → 0100
(-) 9 → 1001 → 0111
= -5

1 (carry)

1011

2's (-0101) → -5

* 23 1 0 1 1 1 → 1 0 1 1 1

10 (−) 0 1 0 1 0 → (+) 1 0 1 1 0

(1) 0 1 1 0 1

discard $\boxed{01101}$

10 0 1 0 1 0 → 0 1 0 1 0

23 (−) 1 0 1 1 1 → (+) 0 1 0 0 1

1 0 0 1 1

(− 0 1 1 0 1)

* 32 16 8 4 2 1

13 0 0 1 1 0 1 → 0 0 1 1 0 1

1 1 1 1 0 1 → (+) 0 0 0 0 1 1

0 1 0 0 0 0

− 1 1 0 0 0 0

1 1 1 1 0 1 → 1 1 1 1 0 1

0 0 1 1 0 1 → (+) 1 1 0 0 1 1

(1) 1 1 0 0 0 0

discard $\boxed{110000}$

9's Compliment Subtraction:-

If 'A-B' is The required operation Take The 9's compliment of B & add it to A. After adding there are two cases

Case 1: Carry generated.

If Carry generated after the addition of A and 9's Compliment of B End around the carry i.e., add carry to LSB

Case 2: Carry not generated

If Carry not generated after the addition take the 9's Compliment of result again and put a '−' sign before the result.

(19)

* $8 \rightarrow 8$, $-5 \xrightarrow{9\text{'s com}} 4$; $8 + 4 = 12$, $+1$ → end around carry; $\boxed{3}$. ($9 - 5 = 4$)

$5 \rightarrow 5$, $-8 \xrightarrow{9\text{'s com}} 1$; $5 + 1 = 6$; 9's of 6 $\boxed{-3}$. ($9 - 8 = 1$, $9 - 6 = 3$)

* $9 \rightarrow 9$, $-4 \xrightarrow{9\text{'s com}} (+)5$; 14, $+1$; $\boxed{5}$. ($9 - 4 = 5$)

$4 \rightarrow 4$, $-9 \xrightarrow{9\text{'s}} (+)0$; 4; 9's of 4 $\boxed{-5}$. ($9 - 4 = 5$)

* $25 \rightarrow 25$, $-19 \xrightarrow{9\text{'s com}} (+)80$; 105, carry 1; $\boxed{6}$. ($99 - 19 = 80$)

$19 \rightarrow 19$, $-25 \xrightarrow{9\text{'s comp}} (+)74$; 93; $\boxed{-6}$. ($99 - 25 = 74$, $99 - 93 = 6$)

* $84 \rightarrow 84$, $-27 \xrightarrow{9\text{'s com}} (+)72$; 156, $+1$; $\boxed{57}$. ($99 - 27 = 72$)

$27 \rightarrow 27$, $(-)84 \rightarrow (+)15$; 42; 9's of 42 $\boxed{-57}$. ($99 - 84 = 15$, $99 - 42 = 57$)

1.

* $459 \rightarrow 459$, $(-)362 \xrightarrow{9\text{'s of}} (+)637$; 1096, $+1$; $\boxed{097}$. ($999 - 362 = 637$)

$362 \rightarrow 362$, $-(459) \rightarrow (+)540$; 902; 9's of 902 $\boxed{-97}$. ($999 - 459 = 540$, $999 - 902 = 97$)

Scanned with CamScanner

10's Compliment subtraction

If "A-B" is the required operation, takes 10's compliment of B and add it to A. After addition there are two cases:

Case(i): Carry generated

If carry generated after adding A & 10's compliment of B, if carry generated then discard the carry

Case(ii): Carry not generated.

After adding A & 10's compliment of B if carry not generated take 10's compliment of result again and put a (-) minus sign before it.

*

$$\begin{array}{r} 9 \\ (-)\,3 \end{array} \xrightarrow{10\text{'s of }3} \begin{array}{r} 9 \\ (+)\,7 \\ \hline \textcircled{1}\,6 \end{array} \quad \text{discard.}$$

10's comp
$$\begin{array}{r} 9 \\ (-)\,3 \\ \hline 6 \\ +1 \\ \hline \boxed{7} \end{array}$$

$$\begin{array}{r} 3 \\ -\,9 \\ \hline \end{array} \xrightarrow{10\text{'s of }9} \begin{array}{r} 3 \\ 1 \\ \hline 4 \end{array}$$

10's of 4 $\boxed{-6}$

10's comp
$$\begin{array}{r} 9 \\ -\,9 \\ \hline 0 \\ +1 \\ \hline 1 \end{array}$$

*

$$\begin{array}{r} 8 \\ (-)\,4 \end{array} \xrightarrow{10\text{'s of }4} \begin{array}{r} 8 \\ (+)\,6 \\ \hline \textcircled{1}\,\boxed{4} \end{array}$$

10's comp
$$\begin{array}{r} 9 \\ (-)\,4 \\ \hline 5 \\ +1 \\ \hline 6 \end{array}$$

$$\begin{array}{r} 4 \\ -\,8 \\ \hline \end{array} \xrightarrow{10\text{'s of }8} \begin{array}{r} 4 \\ (+)\,2 \\ \hline 6 \end{array}$$

10's of 6 $\boxed{-4}$

10's compl.
$$\begin{array}{r} 9 \\ (-)\,8 \\ \hline 1 \\ +1 \\ \hline 2 \end{array} \quad \begin{array}{r} 9 \\ (-)\,6 \\ \hline 3 \\ +1 \\ \hline 4 \end{array}$$

*

$$\begin{array}{r} 56 \\ -\,43 \end{array} \xrightarrow{10\text{'s of }43} \begin{array}{r} 56 \\ (+)\,57 \\ \hline \textcircled{1}\,13 \end{array}$$

10's comp
$$\begin{array}{r} 99 \\ -43 \\ \hline 56 \\ +1 \\ \hline 57 \end{array}$$

$$\begin{array}{r} 43 \\ -(56) \end{array} \xrightarrow{10\text{'s of }56} \begin{array}{r} 43 \\ (+)\,44 \\ \hline 87 \end{array}$$

10's of 87 $\rightarrow \boxed{-13}$

10's comp
$$\begin{array}{r} 99 \\ 56 \\ \hline 43 \\ +1 \\ \hline 44 \end{array}$$

$$\begin{array}{r} 99 \\ -\,87 \\ \hline 12 \\ +1 \\ \hline 13 \end{array}$$

$$
\begin{array}{l}
* \quad 546 \rightarrow 546 \\
\;\; -307 \xrightarrow{10\text{'s of}} \underline{693} \\
\qquad\qquad\;\; \circledast 239
\end{array}
\qquad
\begin{array}{r}
999 \\
\underline{307} \\
692 \\
\underline{+1} \\
693
\end{array}
\qquad\Bigg|\qquad
\begin{array}{l}
307 \rightarrow 307 \\
-546 \rightarrow \underline{454} \\
\qquad\quad\;\; 761 \\
10\text{'s of } 761 \rightarrow \boxed{-239}
\end{array}
\qquad
\begin{array}{r}
999 \\
(-)\underline{546} \\
453 \\
\underline{+1} \\
454
\end{array}
\quad
\begin{array}{r}
999 \\
\underline{761} \\
238 \\
\underline{+1} \\
239
\end{array}
$$

Binary codes:

Binary codes are classified into 6 types

1. Weighted Codes
2. Non-Weighted Codes
3. Sequential codes
4. Reflective codes
5. Alphanumeric Codes
6. Error Correcting & detecting codes

Weighted Codes:

In weighted codes each place value is having a specific weight. Based on the weights, the codes are derived.

Ex: 8421, 2421, 5411 etc....

Non-weighted codes.

These are the codes derived from weighted codes and no specific weight for place value

Ex: Gray code, Parity, excess 3 code etc...

Sequential codes:

(4 2 1 1) → not sequential

In Sequential codes the weights are continously increasing from LSB to MSB

Ex: 8421.

Reflective codes.

In these codes zero is the 1's compliment of 9, 1 is the 1's compliment of 8

(P.T.O)

Scanned with CamScanner

2 is the is compliment of 8^x and so on.
Ex: 2421

22

→ Write the 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	1	1	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

Alphanumeric codes:

In Alphanumeric codes each alphabet (Both uppercase & lowercase) numbers and symbols are having specific binary representation

Ex: ASCII code

American Standard Code for Information Interchange

Error Correcting & Error detecting codes

(23)

By using some of the binary codes errors can be detected but not corrected they are called error detecting codes.

By using some of the binary codes errors can be detected & corrected they are called error correcting code

Ex: Hamming code.

BCD Code: (Binary coded decimal)

In BCD code all the decimal digits from 0 to 9 are coded with binary numbers as shown below

	8	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	0	0	0
9 –	1	0	0	1

BCD addition:

In BCD addition, after adding two BCD numbers, the result should be valid BCD number. If it is Invalid add Correction factor 6 (0110) to the Invalid BCD number.

* 5 → 0 1 0 1
3 → (+) 0 0 1 1
8 → 1 0 0 0

* 5 → 0 1 0 1
5 → 0 1 0 1
10 (+)
1 0 1 0 → Invalid BCD.
+6 → + 0 1 1 0
0001 0000
1 0

* 8 → 1 0 0 0
+6 → + 0 1 1 0
14
1 1 1 0 → Invalid BCD
+6 → + 0 1 1 0
0001 0 1 0 0
1 4

* 15 → 0001 0101
12 → + 0001 0010
27
0010 0111
2 7

* 39 → 0011 1001
(+) 23 → + 0010 0011
62
0101 1100 Invalid BCD.
+6 0110
0110 0010
6 2

0110 0010
6 2

(25)

* 28 → 0010 1000
+ 26 → (+) 0010 0110

54 → 0100 1110 → Invalid BCD

0101 0100 +6 → 0110
5 4

0101 0100
5 4

* 45 → 0100 0101
+ 38 → 0011 1000

83 → 0111 1101 → Invalid BCD

1000 0011 +6 → 0110
8 3

1000 0011
8 3

* 157 → 0001 0101 0111
+ 138 → 0001 0011 1000

295 → 0010 1000 1111

0010 1001 0101 +6 → 0110
2 9 5

0010 1001 0101
2 9 5

* 867 → 1000 0110 0111
+ 574 → + 0101 0111 0100

1441 → 1101 1101 1011

0001 0100 0100 0001 +6 → 0110

1101 1110 0001

+6 0110

1110 0100 0001

+ 0110

0001 0100 0100 0001
1 4 4 1

BCD Subtraction :-

If "(A-B)" is required operation, take the 9's compliment of "B" and add it to A. After adding there are two cases.

Case 1: Invalid BCD

If the sum is invalid BCD number validate it by adding 6 (0110). After addition of 6, the carry has to be end arounded.

Case 2: Valid BCD.

After addition, if the result is valid BCD, take the 9's compliment of result once again & put (-) minus sign before it.

Ex:

① 8 → 1000
(-) 4 → (9's of 4) 0101
4 = 0100

(+)
1101 → Invalid BCD
+6 0110
1 0011
+ 1
0100

4 → 0100
-8 → (9's of 8) 0001
-4 = -0100

0101 → valid BCD
9's of result → $\boxed{-0100}$

② 9 → 1001
-3 → (9's of 3) 0110
6 = 0110

1111 → Invalid BCD
+6 0110
1 0101
+ 1
0110

9
-3
6

3 → 0011
-9 → (9's of 9) (+) 0000
-6

0011
9's of result is $\boxed{-0110}$

9
-3
6

* 25 → 0010 0101
-13 →(9's of 13) 1000 0110
12
0001 0010

(+) 1010 1011
+6 0110
1011 0001
+6 0110
1000 0001
→ +1
$\boxed{0001\ 0010}$

99
-13
86
27

99
(-) 87
12

99
25
74

13 → 0001 0011 + 0001 0011
-25 → 0010 0101 + 0111 0100
-12
1000 0111
9's of result $\boxed{-0001\ 0010}$

* 257 → 0010 0101 0111
-143 → (+) 0001 0100 0011
114 → 1010
0001 0001 0100

999
143
856

0010 0101 0111
+ 1000 0101 0110
1010 1010 1101
+6 0110
1010 1011 0011
+6 0110
1011 0001 0011
+6 0110
1 0001 0001 0011
+1
0001 0001 0100

(28)

143	→	0001	0100	0011	
-257	→	0111	0100	0010	+
-114		1000	1000	0101	

999
257
742

999
885
114

9's of 885 is $\boxed{-0001\ 0001\ 0100}$

Gray Code:

Gray code is a non weighted code, derived from binary code. In gray code there is only one bit change from one number to the immediate next number.

Q: Generate 4-bit gray code from single bit gray code.

Ans

0 - 0   0 0
1 - 1   0 1
         1 1
         1 0

0 - 0	0 - 0 0	0 - 0 0 0	0 - 0 0 0 0
1 - 1	1 - 0 1	1 - 0 0 1	1 - 0 0 0 1
	2 - 1 1	2 - 0 1 1	2 - 0 0 1 1
	3 - 1 0	3 - 0 1 0	3 - 0 0 1 0
		4 - 1 1 0	4 - 0 1 1 0
		5 - 1 1 1	5 - 0 1 1 1
		6 - 1 0 1	6 - 0 1 0 1
		7 - 1 0 0	7 - 0 1 0 0
			8 - 1 1 0 0
			9 - 1 1 0 1
			10 - 1 1 1 1
			11 - 1 1 1 0
			12 - 1 0 1 0
			13 - 1 0 1 1
			14 - 1 0 0 1
			15 - 1 0 0 0

Binary to Gray Code:

(29)

To convert given 4-bit binary to its equivalent 4-bit gray code, the following formula is used.

Let the given 4 bit binary number is $(B_3 B_2 B_1 B_0)$ and its equivalent 4 bit gray code is $(G_3 G_2 G_1 G_0)$ then

$G_3 = B_3$

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$

$G_0 = B_1 \oplus B_0$

EXOR ~~truth~~ table

$\oplus$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

→ Convert $(1111)_B = (?)_G$

Ans.

B_3 1 $\oplus$ 1 B_2 $\oplus$ 1 B_1 $\oplus$ 1 B_0

↓ ↓ ↓ ↓

$(1\ 0\ 0\ 0)_G$

→ $(10111 0)_B = (?)_G$

1 0 1 1 1 0

↓ ↓ ↓ ↓ ↓ ↓

1 1 1 0 0 1

$(111001)_G$

→ $(0111)_B = (?)_G$

Ans.

0 1 1 1

↓ ↓ ↓ ↓

$(0\ 1\ 0\ 0)_G$

→ $(1001)_B = (?)_G$

1 0 0 1

↓ ↓ ↓ ↓

$(1\ 1\ 0\ 1)_G$

→ $(1110)_B = (?)_G$

1 1 1 0

↓ ↓ ↓ ↓

$(1\ 0\ 0\ 1)_G$

Gray code to Binary code Conversion:

Let the given 4-bit gray code is $(G_3G_2G_1G_0)$ and its equivalent binary is $(B_3B_2B_1B_0)$ Then

$$B_3 = G_3$$

$$B_2 = \cancel{G_3 \oplus B_2}\ B_3 \oplus G_2.$$

$$B_1 = B_2 \oplus G_1$$

$$B_0 = B_1 \oplus G_0$$

→ Convert $(1000)_G = (?)_B$

G_3	G_2	G_1	G_0
1	0	0	0
1	1	1	1
B_3	B_2	B_1	B_0

$(1111)_B$

→ $(1011)_G = (?)_B$

1 0 1 1

1 1 0 1

$(1101)_B$

→ $(0111)_G = (?)_B$

0 1 1 1

0 1 0 1

$(0101)_B$

Excess -3 code:

Excess -3 code is a non weighted code delivered from BCD code for every BCD number 3 is added to get excess 3-code

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

Excess-3 addition:

In excess-3 addition after adding two excess-3 numbers there are two cases

Case(i): No carry

If there is no carry after excess-3 addition subtract (0011) 3 from the result to get final result

Case(ii): Carry generated:

After excess-3 addition if carry generated add 3(0011) to both sum & carry.

Ex:

```
    3 →     0 1 1 0
(+) 4 → (+) 0 1 1 1
    7       1 1 0 1
(1010)      0 0 1 1  (sub-3)
            1 0 1 0
```

(32)

```
 2 →  0101
+3 → +0110
 5     1011
1000   0011 (Sub 3)
       1000
```

```
  6 → 1001
+ 7 → 1010
 13   0001 0011
      0011 0011 Add 3
      0100 0110
        1     3
```

```
 13 →   0100 0110
 12     0100 0101
+25    +1000 1011
       -0011 sub(3)
        1000 1000
  Sub 3 0011
        0101 1000
          2    5
0101 1000
  2    5
```

```
  38  →   0110 1011
 +25      0101 1000
  63   (+)1100 0011
          0011 0011 sub 3
       (-)1001 (+)0110
1001 0110
```

```
 45  →  0111 1000
 36  → +0110 1001
+81     1110 0001
             0011 Add 3
        1110 0100
Sub 3   0011
        1011 0100
1011 0100
  8    1
```

```
  93  →    1100 0110
 +48    (+)0111 1011
 141  0001 0100 0001
      0011 0011 0011 Add(3)
      0100 0111 0100
0100 0111 0100
```

Excess-3 Subtraction:

In excess-3 subtraction if A-B is the required operation then take the 9's (or 1's compliment for excess 3 only) compliment of B and add it to A. After addition there are two cases.

Case 1: Carry generated.

After addition if carry generated, end-around the carry and add 3 (0011) to the sum to get the final answer.

Case 2: Carry not generated.

After addition if carry is not generated subtract 3 from the sum and once again take the 9's (or 1's compliment for excess 3 only) compliment of the result and put (-) sign before the result.

* Excess-3 is a self compliment code, because 9's compliment of 1's compliment is same.

Ex:

```
*   6 -->          1 0 0 1        9
  - 3 9's of 3 --> 1 0 0 1        3
  ---             ---------      ---
    3            1 0 0 1 0        6
  0110           |_______↑
                     +   1   (end-around)
                 ---------
                   0 0 1 1
                   0 0 1 1   Add 3 (0011)
                 ---------
                   0 1 1 0
                 ---------

    3 -->            0 1 1 0
  - 6 9's of 6 --> + 0 1 1 0
  ---               ---------
   -3                1 1 0 0
  -0110              0 0 1 1   Sub 3 (0011)
                    ---------
  9's of result      1 0 0 1
                  └> -0 1 1 0
                    ---------
```

* 5 → 1 0 0 0   9
-4 9's of 4 → (+) 1 0 0 0   4
1   5

1 0 0 0 0

0100   end around carry.

0 0 0 1
0 0 1 1 Add 3(0011)

0 1 0 0

4 → 0 1 1 1   9
-5 → 0 1 1 1   5
-1   4

1 ~~1~~0 ~~1~~02 ~~0~~2

-0100   0 0 1 1 Sub 3(0011)

1 0 1 1

9's of result (0 1 0 0)

* 2 6 → 0101 1001 → 0101 1001
-1 4 9's → 0100 0111 1's comp → 1011 1000
1 2   (+)

0001 0001 0001

0100 0101

0001 0010
(+) 0011 (0011) Add 3
0001 0101
0011 Add 3
0100 0101
1 2

* 1 4 → 0100 0111
-2 6 9's → (+) 1010 0110
-12   1110 1101

99
26
73

0011 (sub 3)

1 ~~1~~~~1~~0²~~0~~2 1010
0011   Sub 3

1 0 1 1 1 0 1 0.

9's of result (-0100 0101)
1 2

* 255 → 0101 1000 1000 → 0101 1000 1000

(−) 163 → 0100 1001 0110 → 1011 0110 1001

092

(+) 1 0000 1111 0001

1

0000 1111 0010

0011 1100 0101

(0011) Add3

0011 0011 0011

(+) 0010 0101

0000 1111 0010

Add3

0011

0000 1111 0101

(Add3) 0011 0011

(−)

0011 1100 0101

0 9 2.

→ 163 → 0100 1001 0110

(−) 255 1's → 1010 0111 0111

(+)

−092

1111 0000 1101

(−) 0011 + 0011 (+) 0011

1100 0011 1010

1's of result

(−0011 1100 0101)

* 335 → 0110 1011 1000

− 196 → 1011 0011 0110

189

(+) 1 0001 1110 1110

1

0100 1011 1100

0001 1110 1111

(+) 0011 (−) 0011 (−) 0011 Add3

0100 1011 1100

1 8 9

999
196
803

36

196 → 0100 1100 1001
−385 1001 0100 0111
−189 1110 0001 0000
0100 1011 1100 (−)0011 (+)0011 (+)0011

1011 0100 0011

⇓

9's of result −(0100 1011 1100)
1 8 9

Scanned with CamScanner

 www.Jntufastupdates.com Scanned with CamScanner

Signed Binary numbers:

In signed binary numbers, one sign bit is allocated for representation of +ve numbers and -ve numbers. For +ve numbers sign bit is 0. For -ve numbers the sign bit is 1.

For Ex:

→ sign bit

$\boxed{0}$ 0001 = +1

$\boxed{1}$ 0001 = -1

* The signed binary numbers can be represented in three ways.
  (1) sign magnitude representation
  (2) 1's compliment "
  (3) 2's compliment "

Note:
*** In all three representations, +ve numbers are having unique representation, where as -ve numbers having different representations

Ex:

	+5	-5
sign magnitude	0 0000101	1 0000101
1's compliment	0 0000101	1 1111010
2's compliment	0 0000101	1 1111011

Ex: using 2's compliment

+7 = 00000111

+5 = 00000101

-5 = 11111011

-7 = 11111001

+2 = 00000010

-2 = 11111110

+12 = 00001100

-12 = 11110100

+7 → 00000111
+5 → 00000101 (+)
carry: 111
+12 → 00001100

+7 → 00000111
-5 → 11111011 (+)
carry: 11111111
+2 → discard (1) 00000010

-7 → 11111001
+5 → 00000101 (+)
-2 → 11111110

$-7 \rightarrow 11111001$

$-5 \rightarrow 11111011$

-12 (+)

discard 1 11110100

same above question using 1's compliment

* Using 1's compliment.

$+7 \rightarrow 00000111$

$+5 \rightarrow 00000101$

$-7 \rightarrow 11111000$

$-5 \rightarrow 11111010$

$+2 \rightarrow 00000010$

$-2 \rightarrow 11111101$

$+12 \rightarrow 00001100$

$-12 \rightarrow 11110011$

$+7 \rightarrow 00000111$

$+5 \rightarrow 00000101$ (+)

$+12 \quad 00001100$

$+7 \rightarrow 00000111$

$-5 \rightarrow 11111010$ (+)

$+2 \quad 1\,00000001$

$+1$

00000010

$-7 \rightarrow 11111000$

$+5 \rightarrow 00000101$ (+)

$-2 \quad 11111101$

$-7 \rightarrow 11111000$

$-5 \rightarrow 11111010$

$-12 \quad 1\,11110010$

$+1$

11110011

⇒ ERROR DETECTING CODES:

Parity bit is an extra bit which is added to the original bit. Parity is of Two types 1. Even Parity
2. odd Parity.

Even Parity: For even parity the no. of 1's in the information should be even including parity bit.

Odd Parity: For odd parity the no. of 1's in information is odd including parity bit.

Ex: Generate Even & odd parity for given message.

a) Ans → 0 0 1 0 1 1 0

0 0 1 0 1 1 0 1 → even parity

0 0 1 0 1 1 0 0 → odd parity.

Block Parity: In block parity the parity is taken for both rows & columns for group of messages.

Ex:

Even parity

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

Scanned with CamScanner

Error detection & correction codes: (40)

In the error detection & correction codes Hamming code is the best example. By using hamming code, error can be detected and it can be corrected.

Hamming code generation:

To generate hamming code for 'm' no. of message bits the following formula is used

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

where p = no. of parity bits

Generate Hamming code for 1011.

Ans:

$$\underset{m_1}{1}\ \underset{m_2}{0}\ \underset{m_3}{1}\ \underset{m_4}{1}$$

$m = 4$

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

$2^P \geq 4+p+1 \quad [\because m=4]$

if $p=0$	if $p=1$	if $p=2$
$2^0 \geq 4+0+1$	$2^1 \geq 4+1+1$	$2^2 \geq 4+2+1$
$1 \geq 5$	$2 \geq 6$	$4 \geq 7$

if $p=3$

$2^3 \geq 4+3+1$

$8 \geq 8$

$m = 4$

$p = 3$

Total $= 7$

Scanned with CamScanner

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

001 – 1
010 – 2
011 – 3
100 – 4
101 – 5
110 – 6
111 – 7

$\boxed{(0110011)}$

0010:

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

0010
1001
1010
1110

(0101010)

1001:

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

(0011001)

Scanned with CamScanner

1010:

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

(1011010)

1110:

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

(0010110)

Error detection and correction:

→ original: 0101010 $\overset{err}{\Rightarrow}$ 0100010

	1	2	3	4	5	6	7	
	0	1	0	0	0	1	0	
$C_1 \to$ 1,3,5,7	0		0		0		0	→ 0
$C_2 \to$ 2,3,6,7		1	0			1	0	→ 0
$C_3 \to$ 4,5,6,7				0	0	1	0	→ 1

C_3	C_2	C_1	
1	0	0	⇒ 4

∴ There is an error in 4^{th} position.
the correct information is 0101010

(43)

⇒ 0101010 $\xrightarrow{\text{error}}$ 0101000

	1	2	3	4	5	6	7	
	0	1	0	1	0	0	0	
$c_1 \to 1,3,5,7 \to$	0		0		0		0	→ 0
$c_2 \to 2,3,6,7 \to$		1	0			0	0	→ 1
$c_3 \to 4,5,6,7 \to$				1	0	0	0	→ 1

c_3 c_2 c_1

1 1 0 → 6

∴ There is an error in 6th position.
The correct information is 0101010

⇒ 0101010 $\xrightarrow{\text{error}}$ 0001010

	1	2	3	4	5	6	7	
	0	0	0	1	0	1	0	
$c_1 \to 1,3,5,7 \to$	0		0		0		0	→ 0
$c_2 \to 2,3,6,7 \to$		0	0			1	0	→ 1
$c_3 \to 4,5,6,7 \to$				1	0	1	0	→ 0

c_3 c_2 c_1

0 1 0 → 2

∴ There is an error in 2nd position, the Correct information is 0101010.

⇒ 0101010

	1	2	3	4	5	6	7	
	0	1	0	1	0	1	0	
$c_1 \to 1,3,5,7$	0		0		0		0	→ 0
$c_2 \to 2,3,6,7$		1	0			1	0	→ 0
$c_3 \to 4,5,6,7$				1	0	1	0	→ 0

c_3 c_2 c_1

0 0 0

No error.

Logic Gates:

(1) Basic gates → AND, OR, NOT
(2) Universal gates → NAND, NOR
(3) Special gates → EX-OR, EX-NOR

AND Gate

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

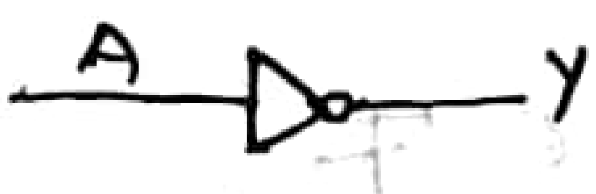
$Y = A \cdot B$
$= AB$

OR Gate:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

$Y = A + B$

NOT Gate:



A	Y
0	1
1	0

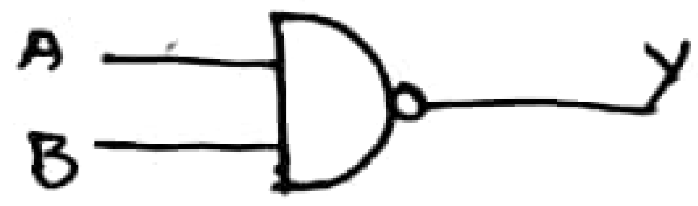
$Y = \overline{A} = A'$

(44)

Scanned with CamScanner

NAND Gate :

NAND Gate is AND gate followed by NOT Gate



$Y = \overline{A \cdot B}$

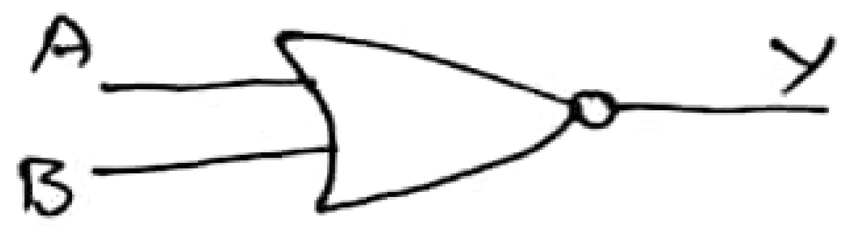
$= \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Any boolean function can be purely implemented with this two gates (NAND, NOR)

NOR Gate :

NOR Gate is OR Gate followed by NOT gate.



$Y = \overline{(A+B)}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Special gates:

→ EX-OR: (inequality gate)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = A \oplus B$

$= A\bar{B} + \bar{A}B$

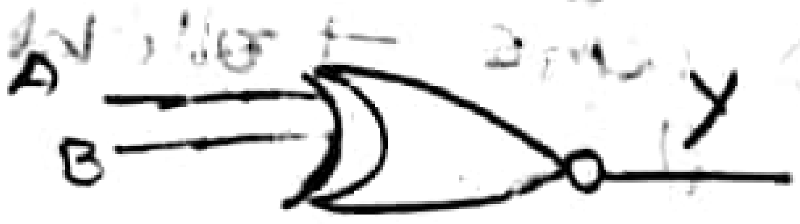
Important:

Property: $A \oplus 0 = A$

$A \oplus 1 = \bar{A}$

Scanned with CamScanner

EX-NOR (equallity gate)



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

$$Y = A \odot B$$
$$= AB + \overline{A}\overline{B}$$
$$= AB + \overline{A}\,\overline{B}$$