Sumedh

• Decimal Conversions

1. To binary

eg $(25.625)_{10}$

divide by 2 — Multiply by 2

2	25	
2	12	– 1
2	6	– 0
2	3	– 0
	1	– 1

carry

(11001)

$0.625 \times 2 = 1.25$ — 1

$0.25 \times 2 = 0.50$ — 0

$0.5 \times 2 = 1.0$ — 1

$.101$

$\therefore (11001.101)_2$

2. To Octal

eg $(175)_{10}$

divide by 8

8	175	
8	21	– 7
8	2	– 5
	0	– 2

$(257)_8$

* for fraction :
same as binary, just multiply by 8

3. To Hexadecimal

eg $(1983)_{10}$

divide by 16

16	1983	
16	123	– 15
16	7	– 11
	0	– 7

$(15 \quad 11 \quad 7)$

↓

$(F\,B\,7)$

By 8421

Same here as well (multiply by 16)

- Binary Conversion

1. To decimal

eg $(1101.10)_2$

$$\begin{matrix} 2^3 & 2^2 & 2^1 & 2^0 & . & 2^{-1} & 2^{-2} \\ 1 & 1 & 0 & 1 & . & 1 & 0 \end{matrix}$$

$$= \begin{array}{lr} 1 \times 2^0 & 1 \\ 0 \times 2^1 & 0 \\ 1 \times 2^2 & 4 \\ 1 \times 2^3 & + \; 8 \\ \hline & 13 \end{array} \qquad \begin{array}{lr} 1 \times 2^{-1} & = 0.5 \\ 0 \times 2^{-2} & = 0 \\ \hline & 0.5 \end{array}$$

$$= (13.05)_{10}$$

2. To Octal

By (4 2 1) code, just do pairs of 3

eg 0110011.101

↓

000 110 011 101

↓

$(063.5)_8$

3. To Hexadecimal

By (8421) code, just do pairs of 4

eg 10110 . 01

↓

0001 0110 0100

↓

$(16.4)_{16}$

- Octal Conversion

1. To decimal

eg $(12.3)_8$

$\overset{8^1}{1}\ \overset{8^0}{2}\ .\ \overset{8^{-1}}{3}$

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

$=$

2. To binary

Write each digit's (421) code

eg $(12.5)_8$

001 010 101

$(001010.101)_2$

3. To Hexadecimal

Step - 1 : Convert to binary
Step - 2 : " binary to hexa

eg $(26.2)_8$

S-1 : To binary

(26.2)

010 110 010

= 010110.010

S.2 : To Hex

(8421)

010110.010

$= (16.4)_{16}$

• Hexadecimal Conversions

1. To decimal

eg $(1AB)_{16}$

$$\overset{16^2}{1}\ \overset{16^1}{A}\ \overset{16^0}{B}$$

$= 1 \times 16^2 + 10 \times 16^1 + 11 \times 16^0$

$= (427)_{10}$

2. To binary

write each digit's (8421) code

eg $(1AB)_{16}$

0001 1010 1011

$= (110101011)_2$

3. To Octal

Step -1 : Convert to binary
Step - 2 : Then binary to Octal

eg $(1AB)_{16}$

S- 1 : To binary

1 A B

0001 1010 1011

$(110101011)_2$

S- 2 To Octal (421) code

$\underline{110}\ \underline{101}\ \underline{011}$

$(653)_8$

- Binary Calculations

1. Addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

eg

$$\begin{array}{r} {}^{1\,1} \\ 111101 \\ +\,111000 \\ \hline 1110101 \end{array}$$

* Note : 1 + 1 = 1 0 (carry)
1 + 1 + 1 = 1 1

2. Subtraction

A	B	borrow	diff
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

eg

$$\begin{array}{r} 11\cancel{1}0 \\ -\,1101 \\ \hline 0001 \end{array}$$

* Note 10 - 1 = 1

3. Multiplication

0 × 1 = 0
1 × 1 = 1

eg

$$\begin{array}{r} 11.01 \\ \times\,10.10 \\ \hline 0000 \\ 1101 \\ 0000 \\ 1101 \\ \hline 1000.0010 \end{array}$$

= 1000 · 0010

3. division

eg $(11011011)_2$ by $(110)_2$

$$\begin{array}{r|l} & 100100.1 \\ \hline 110 & 11011011 \\ & -110 \\ \hline & 0110 \\ & -110 \\ \hline & 0110 \\ & 110 \\ \hline & 0 \end{array}$$

= 100100.1

- **Subtraction by 2's complement**

Case - 1 : $m > n$ $(m - n)$

Step - 1 : Take n 2's Complement
2 : Add m + n 2's
3 : If carry at last digit, discard it

eg $(68)_{10} - (27)_{10}$

2	68	
2	34	0
2	17	0
2	8	1
2	4	0
2	2	0
	1	0

2	27	
2	13	1
2	6	1
2	3	0
	1	1

$\therefore \underset{m}{(1000100)} - \underset{n}{(0011011)}$

$n \; 2's = 1100101$

$$
\begin{array}{rl}
m + n\;2's = & 1000100 \\
 & 1100101 \\
\hline
 & \cancel{1}0101001
\end{array}
$$

$= 0101001$

Case - 2 : $m < n$ $(m - n)$

Step - 1 : Take n 2's Compliment
2 : add m + n 2's
3 : Take 2's Compliment of the answer

eg $(43)_{10} - (89)_{10}$

2	43	
2	21	1
2	10	1
2	5	0
2	2	1
	1	0

2	89	
2	44	1
2	22	0
2	11	0
2	5	1
2	2	1
	1	0

$\underset{m}{(101011)} - \underset{n}{(1011001)}$

$n \; 2's = 0100111$

$$
\begin{array}{rl}
m + n\;2's = & 0101011 \\
 & 0100111 \\
\hline
 & 1010010
\end{array}
$$

2s

$= 0101110$

BCD Arithmetic

→ They follow (8 4 2 1) code

eg
9 = 1001
3 = 0011
23 — write seperate for each digit
0010 0011 = 0010 0011

64 = 0110 0100

• BCD Addition

Case - 1 : No Carry (Normal Addⁿ)

eg

25	0010	0101
13	0001	0011
38	0011	1000

Case -2 : If Carry, add 6/(0110) to that group

└ i.e the val shld not be more than 8/(1000)

eg

679	0110	0111	1001
536	0101	0011	0110
1215	1100	1011	1111
	+ 0110	+ 0110	+ 0110
	10010	0001	0101

carry

carry

: 1 0010 0001 0101

= 1215

- BCD Subtraction

Case - 1 : No borrow, normal difference

eg

```
  3 8      0011  1000
  1 5      0001  0101
  ---      ----------
  2 3      0010  0011
```

Case - 2 : If borrow, then subtract 6/(0110) frm that grp

└ i.e borrow from other group

eg

```
206.7      0010   0000   0110 . 0111
147.8      0001   0100   0111 . 1000
           ----------------------------
           0000   1011   1110   1111
                - 0110 - 0110 - 0110
                  ----   ----   ----
                  0101   1000   1001
```

= 0101 1000.1001

= 58.9

- Gray Code

Remember X-OR gate for gray code conversion

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

- Binary to gray code

eg 0 1 0 0 1

XOR

→ 0 $0 \oplus 1$ $1 \oplus 0$ $0 \oplus 0$ $0 \oplus 1$

01101

eg 10110

→ 1 1⊕0 0⊕1 1⊕1 1⊕0

11101

- Gray code to binary conversion

eg 01101

→ 0 1 1 0 1

0 1 0 0 1

eg 10110

→ 11011

- Excess - 3 code

- Decimal to Excess -3

Step -1 : Convert decimal to BCD code
Step- 2 : Add 3/(0011) to each group

eg $(26)_{10}$ 2 6

BCD	0010	0110
+3	0011	0011
	0101	1001

(0101 1001)

= 59

*Note : To cross-check, add 33 to decimal no

i.e 26+ 33 = 59

- Binary to Excess - 3

Step - 1 : Convert to decimal
Step - 2 : add 33
Step - 3 : write in BCD

eg $(11110)_2$

$\rightarrow$

$0 \times 2^0 = 0$
$1 \times 2^1 = 2$
$1 \times 2^2 = 4$
$1 \times 2^3 = 8$
$1 \times 2^4 = 16$
$= 30$

$= (30)_{10}$

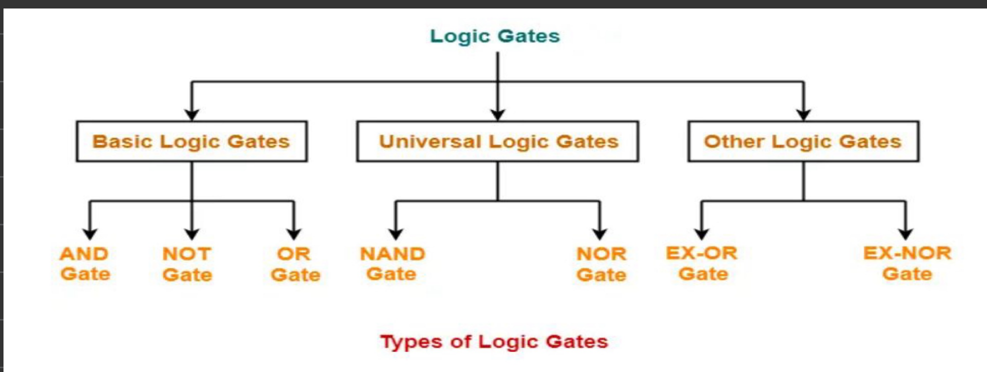
$30 + 33 = (63)_{10}$

$(0110 \quad 0011)$

- ASCII

- Control Characters-0 to 31 and 127
- Special Characters- 32 to 47, 58 to 64, 91 to 96, and 123 to 126
- Numbers Characters- 0 to 9
- Letters Characters - 65 to 90 and 97 to 122

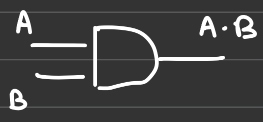
- Logic Gates



Types of Logic Gates

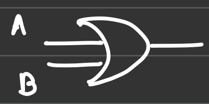
1. AND Gate

A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1



2. OR Gate

A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1



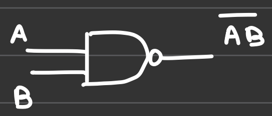
3. Not Gate

$0 \rightarrow 1$

$1 \rightarrow 0$

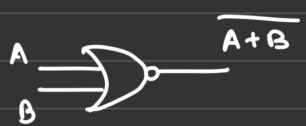
4. NAND Gate

A	B	$Y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0



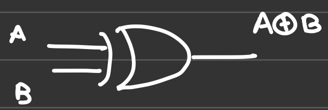
5. NOR Gate

A	B	$Y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



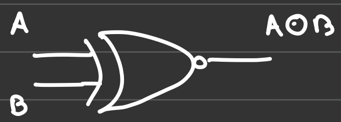
6. EX-OR Gate

A	B	Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0



7. EX-NOR Gate

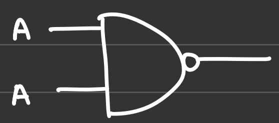
A	B	Y = A⊙B
0	0	1
0	1	0
1	0	0
1	1	1



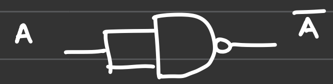
- NAND gate to all gate

1. NOT

Take A = B

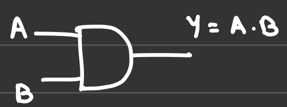


$Y = \overline{A \cdot A} = \bar{A} =$

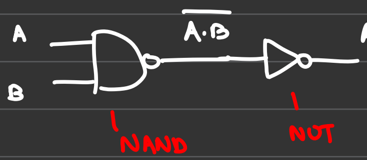


NOT as NAND

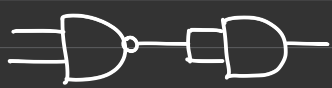
2. AND



So,

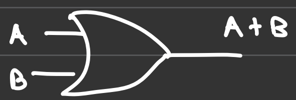


=

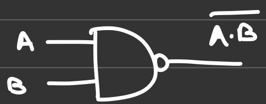


AND as NAND

3. OR



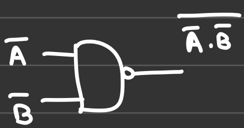
So



de-morgan law

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

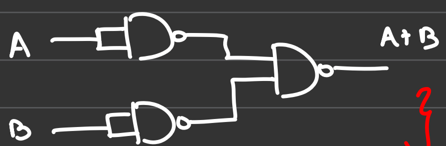
=



$= A + B$

NOT as NAND

∴



OR as NAND

4. NOR

Just add NOT gate to OR as NAND

∴