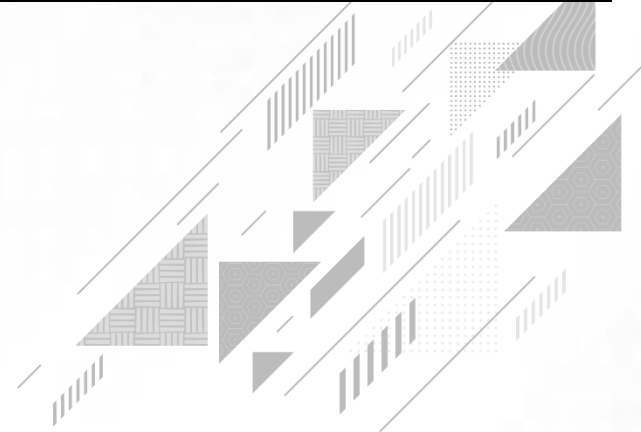
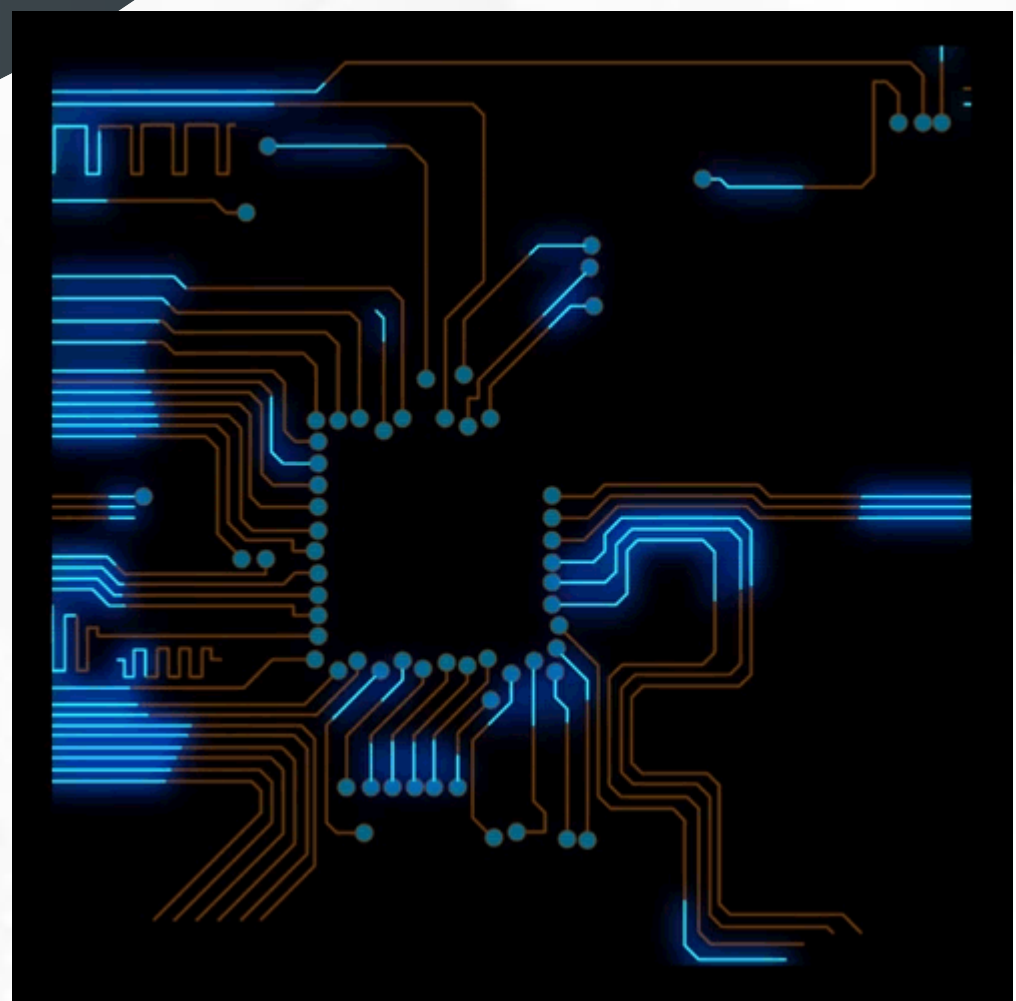


Digital Electronics (DE)
Code: 01EC0102 3rd sem

Unit-1

Number Systems and Codes





Outline

- 1. Analogue Versus Digital**
- 2. Introduction to Number Systems and its Conversions**
- 3. Floating-Point Numbers**
- 4. Various binary code**



Marwadi
University

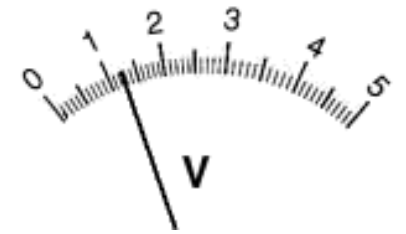
UNIT -1

Signal

- Signal: **Physical quantity** which contains some information.
- It is a function of one or more independent variables.
- Types: Analog and Digital



Types of signal: A analog and Digital signal



- An ***analog signal*** is an electric signal whose value **varies** in analogy with a physical quantity such as **temperature, force, or acceleration and any natural signal**.

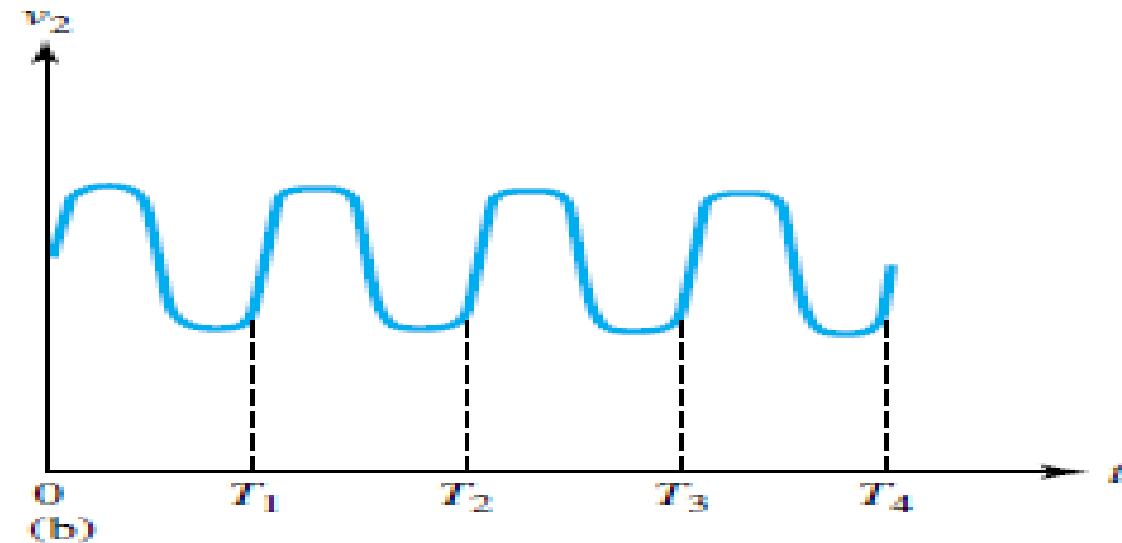
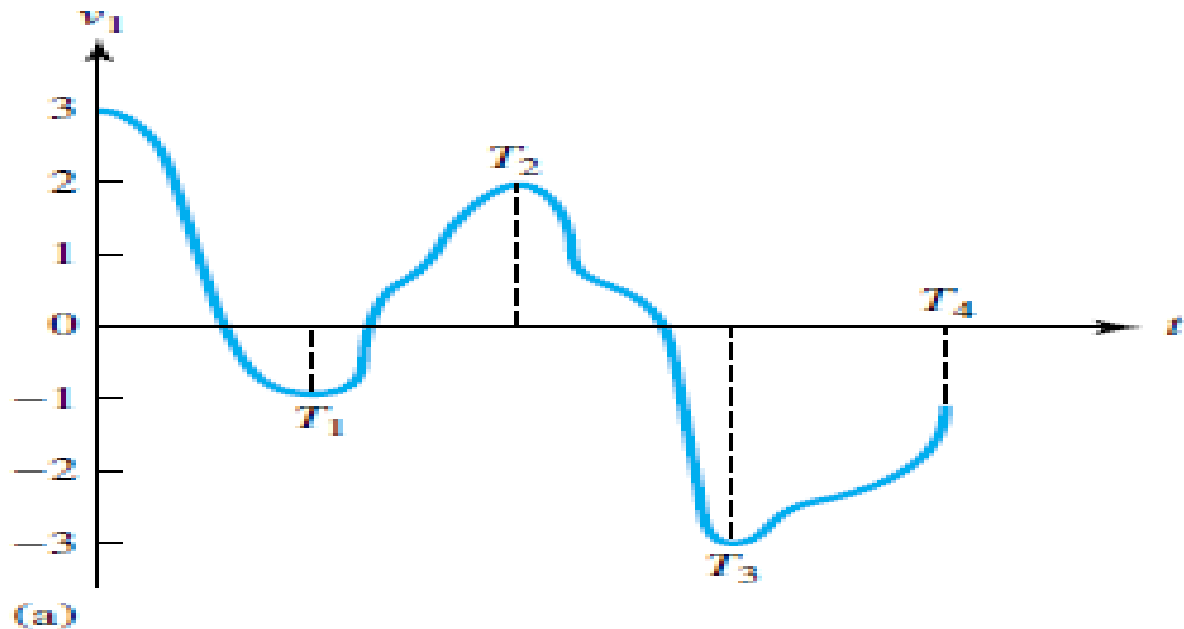


Figure 1 Continuous signals.

Types of signal: A analog and Digital signal

- A **digital signal** can only have a finite number of discrete amplitudes at any given time.
- Digital devices works only digital signal not analog signal.
- Greater accuracy

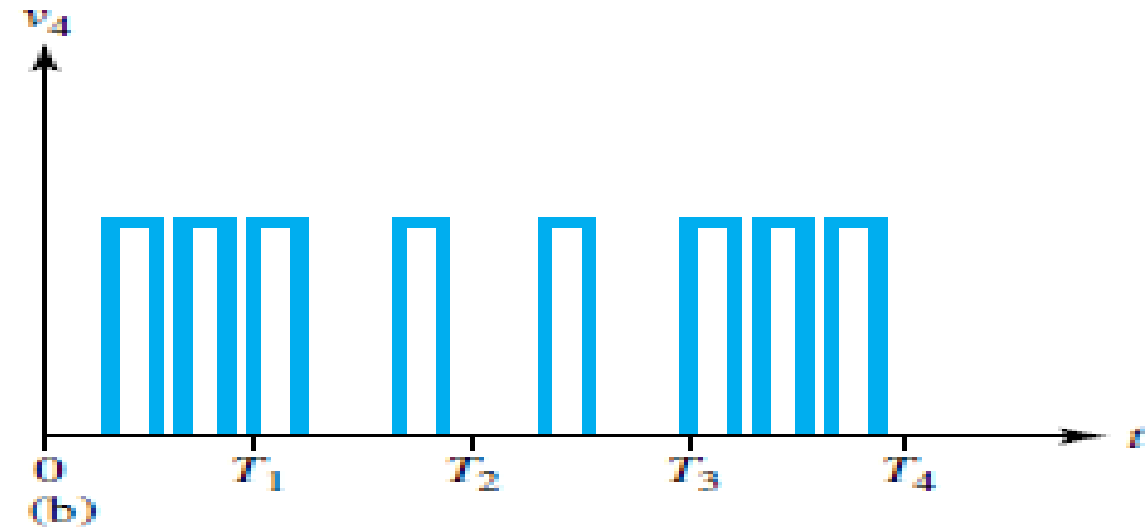
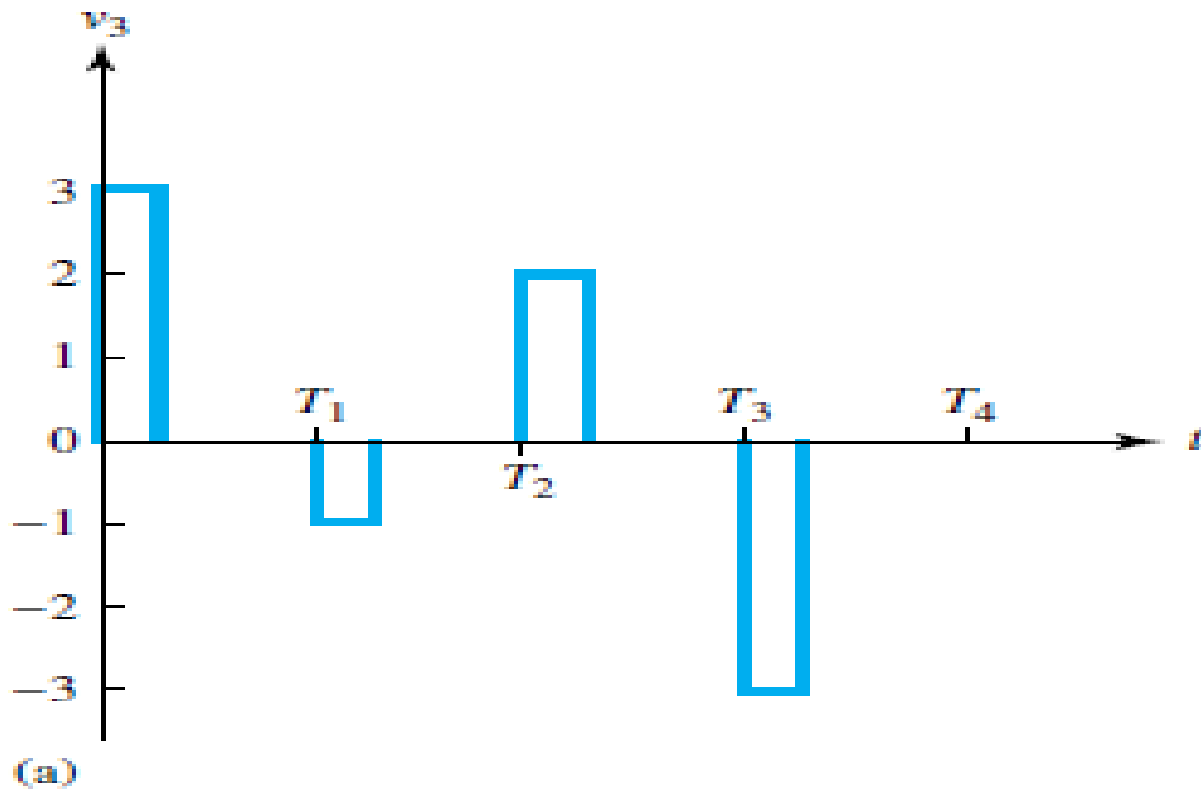
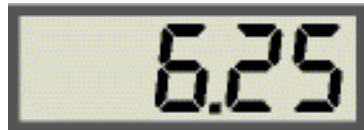


Figure Discrete signals.

Types of signal: Aanalogue and Digital signal

- The most common digital signals are *binary signals*.
- A binary signal is a signal that can take only one of two discrete values and is therefore characterized by transitions **between two states**.



Types of signal: Digital signal

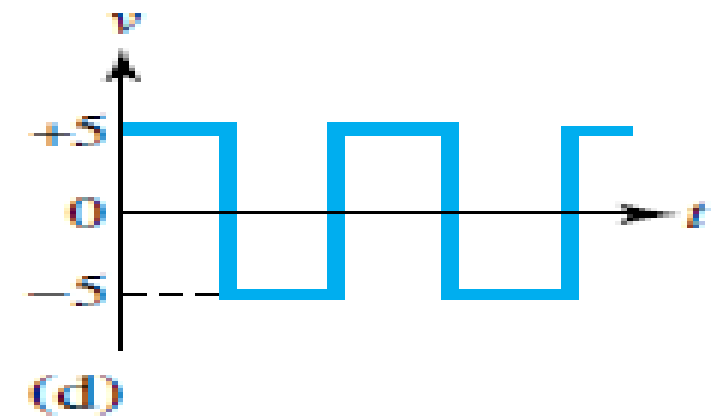
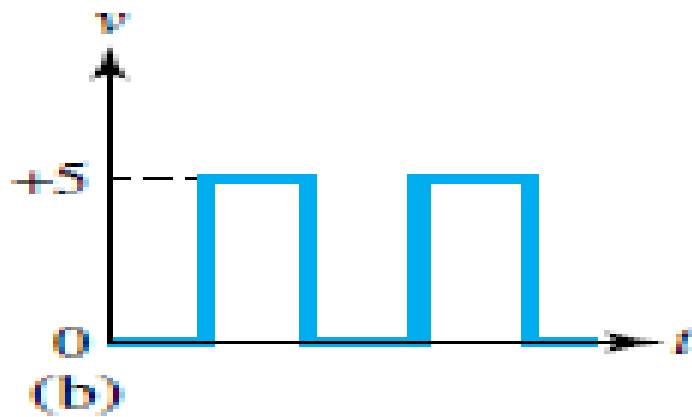
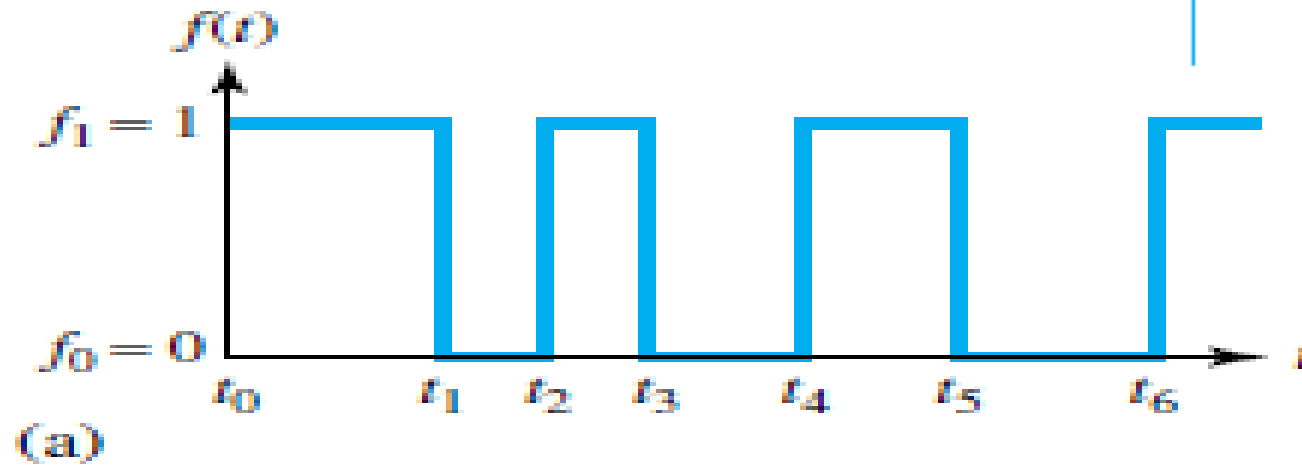
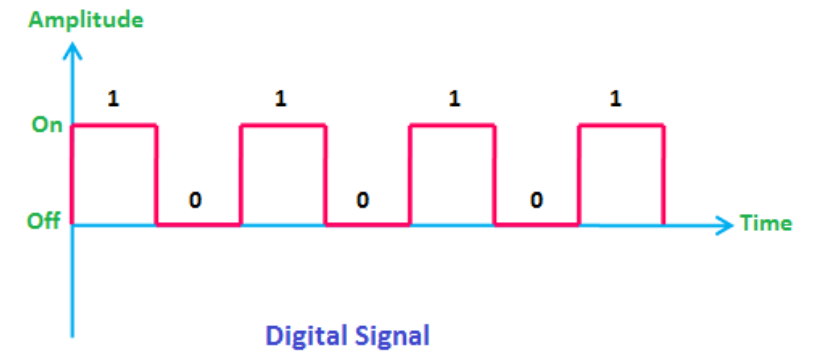


Figure Typical binary signals.

Difference between Analog and Digital signal

Comparison of Analog and Digital Signal

S.N.	Analog Signal	Digital Signal
1	Analog signal has a infinite values.	Digital signal has a finite number of the values.
2	Analog signal has a continuous nature.	Digital signal has a discrete nature.
3	Analog signal is generate by transducers and signal generators.	Digital signal is generate by A to D converter.
4	Example of analog signal: sine wave, triangular waves.	Example of digital signal: binary signal.

Example of Analog and Digital Devices

- 1) **Voltmeter** → analog or digital (digital voltmeter, or DVM)
- 2) **Speedometer of a car** → analog device,
- 3) **Odometer** (which records kilometres) → digital
- 4) **Toggle switch** → digital,
- 5) **Dimmer switch** → analog

Number System

Section - 1

Different number system with different number bases which plays **very important role** in **today's computer world**.



Common Number Systems

System	Base/ Radix	Symbols	Used by Humans?	Used in Computers and digital circuit?
Decimal	10	0, 1, ... 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, ... 7	No	No
Hexa- decimal	16	0, 1, ... 9, A, B, ... F	No	No

- The base (**radix**) of the number system is the total number of digits in the system.
- Octal and Hexa decimal : Used to represent long binary pattern into compact form.



DECIMAL	HEXADEC	OCTAL	BINARY
0	0	000	00000000
1	1	001	00000001
2	2	002	00000010
3	3	003	00000011
4	4	004	00000100
5	5	005	00000101
6	6	006	00000110
7	7	007	00000111
8	8	010	00001000
9	9	011	00001001
10	A	012	00001010
11	B	013	00001011
12	C	014	00001100
13	D	015	00001101
14	E	016	00001110
15	F	017	00001111
16	10	020	00010000
17	11	021	00010001
18	12	022	00010010
19	13	023	00010011
20	14	024	00010100
21	15	025	00010101
22	16	026	00010110
23	17	027	00010111
24	18	030	00011000
25	19	031	00011001

3 places
8 places

Some definition:

- Decimal → Decimal point
- Binary → Binary point
- Bit (Binary Digits) → 0 or 1
- Nibble → Group of 4 bits → 0010
- Byte → Group of 8 bits → 10101100
- word → 16 bits

Count from 0 to 8 in radix 6 system.

Count from 0 to 8 in radix 3 system.

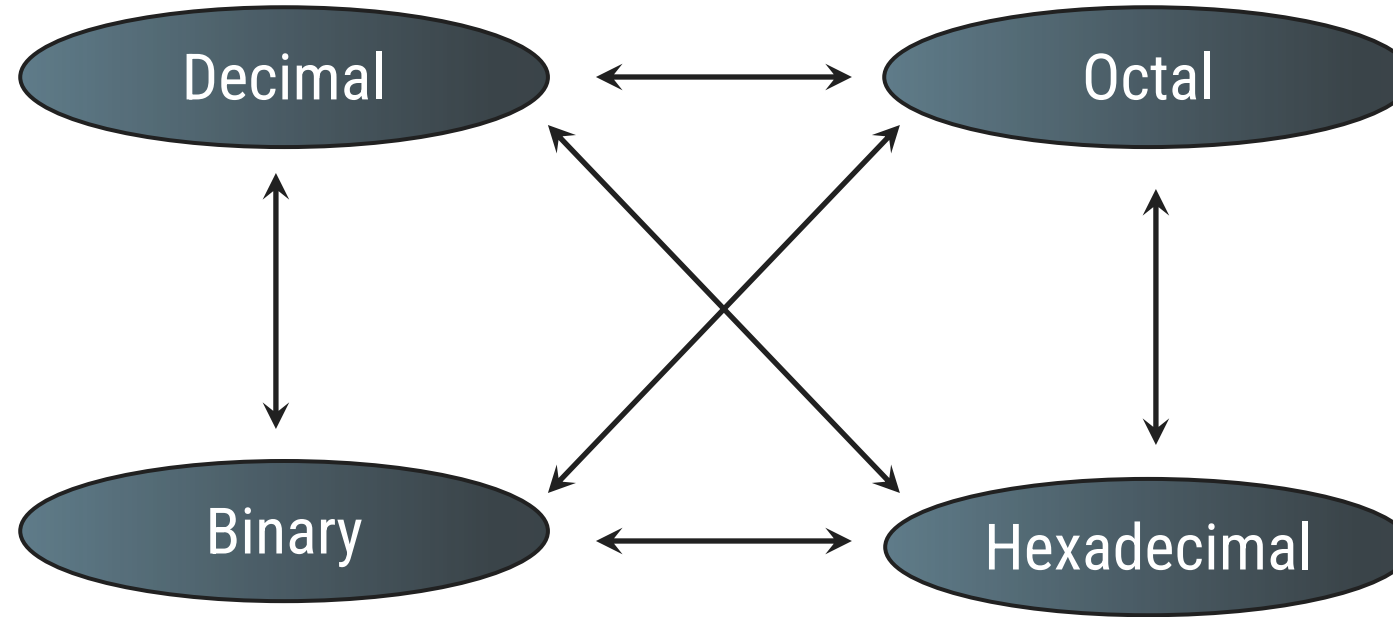
In decimal there are 0 to 25 numbers and in Hexa decimal there are 0 to 19 but in hexa between 9 and 10 there will be alphabets A to F.



Marwadi
University

Conversion among Bases

► Possibilities



There are total 12 possibilities among conversion bases

► Example

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$

Base



Marwadi
University

1) Decimal to Binary

► Technique

- ➔ Divide by **two**, keep track of the remainder
- ➔ The remainders read from bottom to top give the equivalent binary integer number.

► Example - 1

$$125_{10} = ?_2$$

$$62 \times 2 = 124 + 1 = 125$$

2	125	1
2	62	0
2	31	1
2	15	1
2	7	1
2	3	1
2	1	1
	0	

$$125_{10} = 1111101_2$$

► Example - 2

$$0.6875_{10} = ?_2$$

$$0.6875 \times 2 = 1.3750$$

$$0.3750 \times 2 = 0.7500$$

$$0.7500 \times 2 = 1.5000$$

$$0.5000 \times 2 = 1.0000$$


<u>integer</u>		<u>fraction</u>
1	+	0.3750
0	+	0.7500
1	+	0.5000
1	+	0.0000

$$0.6875_{10} = 0.1011_2$$



1) Decimal to Binary

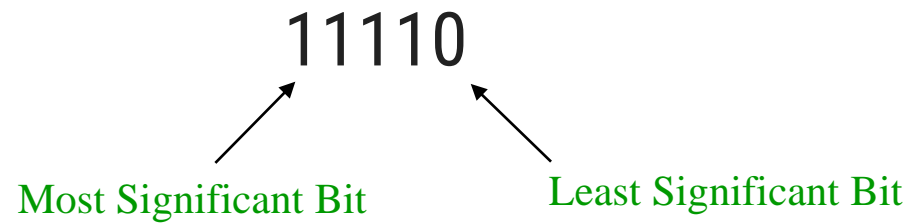
❖ The decimal number $(75)_{10}$ is converted into its binary equivalent:

			Quotient	Remainder		<div>MSB : Most Significant Bit LSB: Least Significant Bit</div>
75	÷	2 =	37	1		
37	÷	2 =	18	1		
18	÷	2 =	9	0		
9	÷	2 =	4	1		
4	÷	2 =	2	0		
2	÷	2 =	1	0		
1	÷	2 =	0	1		
			Stop			

Thus, the binary number for $(75)_{10}$ is given by:

$(1001011)_2$
 ↑ ↑
 MSB LSB

- The left most bit in binary is called the **Most Significant Bit (MSB)**
- The rightmost bit in binary is called the **Least Significant Bit (LSB)**



1)Decimal to Binary

❖ Conversion of fractional decimal number $(0.4375)_{10}$ into its binary equivalent

$$\begin{array}{lcl} 0.4375 \times 2 = & 0.8750 & \\ 0.8750 \times 2 = & 1.7500 & \\ 0.7500 \times 2 = & 1.5000 & \\ 0.5000 \times 2 = & 1.0000 & \\ & \text{Stop} & \end{array}$$



$$0.252_{10} = ?_2$$

$$\text{Thus, } (0.4375)_{10} = (.0111)_2.$$



Accuracy in Binary Number Conversion

Example

- ▶ Convert $(0.252)_{10}$ to binary with an error less than 1%.

Solution

- ▶ Absolute value of allowable error is found by calculating 1% of the number

$$E_{allow} = 0.01 \times 0.252 = 0.00252_{10}$$

- ▶ Maximum error due to truncation is set to be less than allowable error by solving from $E_{10} = 2^{-n}$. This equation is written as

$$2^{-n} < 0.00252$$

- ▶ Inverting both sides of the inequality

$$2^n > 397$$

Accuracy in Binary Number Conversion

- ▶ Taking log of both sides and solving for n

$$n \log 2 = \log 397$$

$$n = \frac{\log 397}{\log 2} = 8.63 \approx 9 \text{ (next largest integer)}$$

- ▶ This indicates that the use of **9 bits** in the binary number will guarantee an error less than 1%.
- ▶ So the conversion is carried out to **9 places** which results in

$$0.252_{10} = 0.010000001_2$$



2) Binary to Decimal

► Technique

- Multiply each bit by 2^n , where n is the “weight” of the bit
- The weight is the position of the bit, starting from 0 on the right (**Weighted form**). Finally, Add the results.

► Example - 1

$$101011_2 = ?_{10}$$

$$\begin{array}{cccccc} & 1 & 0 & 1 & 0 & 1 & 1 \\ & \swarrow & \swarrow & \swarrow & \swarrow & \swarrow & \swarrow \\ 1 \times 2^5 & + & 0 \times 2^4 & + & 1 \times 2^3 & + & 0 \times 2^2 & + & 1 \times 2^1 & + & 1 \times 2^0 \\ 32 & + & 0 & + & 8 & + & 0 & + & 2 & + & 1 \end{array}$$

$$101011_2 = 43_{10}$$

► Example - 2

$$11.11_2 = ?_{10}$$

In this time of system assume number system like integer and the decimal represents the addition of two numbers

$$\begin{array}{cccc} & 1 & 1 & . & 1 & 1 \\ & \swarrow & \swarrow & & \swarrow & \swarrow \\ 1 \times 2^1 & + & 1 \times 2^0 & + & 1 \times 2^{-1} & + & 1 \times 2^{-2} \\ 2 & + & 1 & + & 0.5 & + & 0.25 \end{array}$$

$$11.11_2 = 3.75_{10}$$



Marwadi
University

3)Decimal to Octal

► Technique

- Divide by **eight**, keep track of the remainder
- The remainders read from **bottom to top** give the equivalent octal integer number.

USE like euclid division lemma.

► Example - 1

$$125_{10} = ?_8$$

8	125	5
8	15	7
8	1	1
	0	

USE DUAL FACTOR .

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

► Example - 2

$$0.6875_{10} = ?_8$$

	<u>integer</u>		<u>fraction</u>
$0.6875 \times 8 = 5.5000$	5	+	0.5000
$0.5000 \times 8 = 4.0000$	4	+	0.0000

$$0.6875_{10} = 0.54_8$$



Marwadi
University

4) Octal to Decimal

► Technique

- ➔ Multiply each digit by 8^n , where n is the “weight” of the digit
- ➔ The weight is the position of the digit, starting from 0 on the right. Finally, Add the results.

► Example - 1

$$724_8 = ?_{10}$$

$$\begin{array}{ccc} 7 & 2 & 4 \\ \swarrow & \downarrow & \searrow \\ 7 \times 8^2 & + & 2 \times 8^1 & + & 4 \times 8^0 \\ 448 & + & 16 & + & 4 \end{array}$$

$$724_8 = 468_{10}$$

► Example - 2

$$43.25_8 = ?_{10}$$

In this type the conversion is about octal to decimal so we have to first multiply by 8 and further we have to reduce power and then we have to addition.

$$\begin{array}{ccccccc} 4 & 3 & . & 2 & 5 \\ \swarrow & \downarrow & & \downarrow & \searrow \\ 4 \times 8^1 & + & 3 \times 8^0 & + & 2 \times 8^{-1} & + & 5 \times 8^{-2} \\ 32 & + & 3 & + & 0.25 & + & 0.0781 \end{array}$$

$$43.25_8 = 35.3281_{10}$$



Marwadi
University

5) Decimal to Hexa-Decimal


► Technique

- ➔ Divide by **sixteen**, keep track of the remainder
- ➔ The remainders read from bottom to top give the equivalent hexadecimal integer number.

► Example - 1

$$1234_{10} = ?_{16}$$

16	1234	2
16	77	13=D
16	4	4
	0	




$$1234_{10} = 4D2_{16}$$

in this the conversion is about decimal to hexadecimal so we have to use dual factor also

► Example - 2

$$0.03125_{10} = ?_{16}$$

	<u>integer</u>		<u>fraction</u>
$0.03125 \times 16 = 0.5000$	0	+	0.5000
$0.5000 \times 16 = 8.0000$	8	+	0.0000



$$0.03125_{10} = 0.08_{16}$$



6) Hexa-Decimal to Decimal

► Technique

- ➔ Multiply each digit by 16^n , where n is the “weight” of the digit
- ➔ The weight is the position of the digit, starting from 0 on the right. Finally, Add the results.

► Example - 1

$$ABC_{16} = ?_{10}$$

A	B	C
↙	↓	↘
$A \times 16^2$	$+ B \times 16^1$	$+ C \times 16^0$
10×16^2	$+ 11 \times 16^1$	$+ 12 \times 16^0$
2560	$+ 176$	$+ 12$

$$ABC_{16} = 2748_{10}$$

► Example - 2

$$43.25_{16} = ?_{10}$$

in this we have to do hexadecimal to decimal so we have to multiply with 16 and do further process which we have done in previous one

4	3	.	2	5
↙	↓		↓	↘
4×16^1	$+ 3 \times 16^0$		$+ 2 \times 16^{-1}$	$+ 5 \times 16^{-2}$
64	$+ 3$		$+ 0.125$	$+ 0.0195$

$$43.25_{16} = 67.1445_{10}$$



Marwadi
University

6) Hexa-Decimal to Decimal

$$\begin{aligned} 356_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\ &= 768 + 80 + 6 \\ &= 854_{10} \end{aligned}$$

$$\begin{aligned} 2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= 687_{10} \end{aligned}$$



7) Octal to Binary

► Technique

→ Convert each octal digit to a 3-bit equivalent binary representation

► Example

$$705_8 = ?_2$$

7	0	5
↓	↓	↓
111	000	101

$$705_8 = 111000101_2$$

In this we have to use binary conversion table

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111



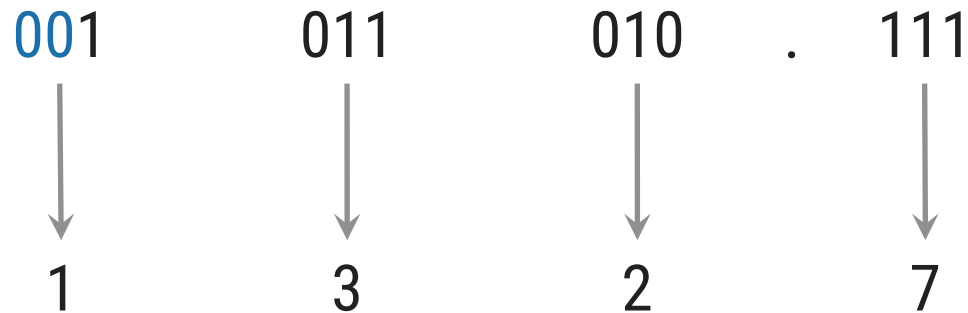
8) Binary to Octal

► Technique

- ➔ From given fractional **point**, group bits in **threes to right** and group bits in **threes to left**
- ➔ If, left with less than 3 bits at the end then **stuff 0s** to make it group of three
- ➔ Convert to octal digits

► Example

$$1011010.111_2 = ?_8$$



$$1011010.111_2 = 132.7_8$$



9) Hexa-Decimal to Binary

► Technique

→ Convert each hexadecimal digit to a 4-bit equivalent binary representation

► Example

$$10AF_{16} = ?_2$$

1	0	A	F
↓	↓	↓	↓
0001	0000	1010	1111

$$10AF_{16} = 1000010101111_2$$

Hexa-Decimal	Binary	Hexa-Decimal	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

same procedure we have to use which we have used in octal binary



Marwadi
University

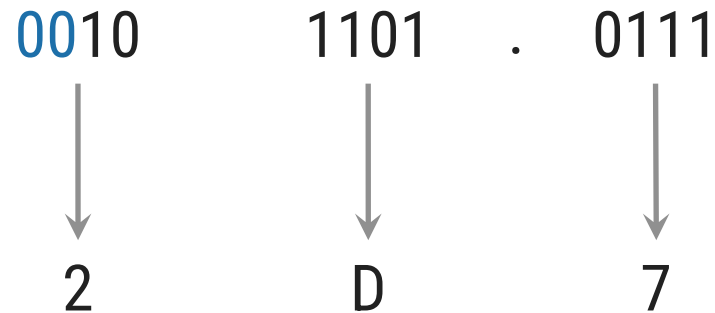
10) Binary to Hexa-Decimal

► Technique

- ➔ From given fractional **point**, group bits in **fours to right** and group bits in **fours to left**
- ➔ If, left with less than 4 bits at the end then **stuff 0s** to make it group of four
- ➔ Convert to hexadecimal digits

► Example

$$101101.0111_2 = ?_{16}$$



$$1011010111_2 = 2D.7_{16}$$

same procedure we have to use which we have used in octal binary

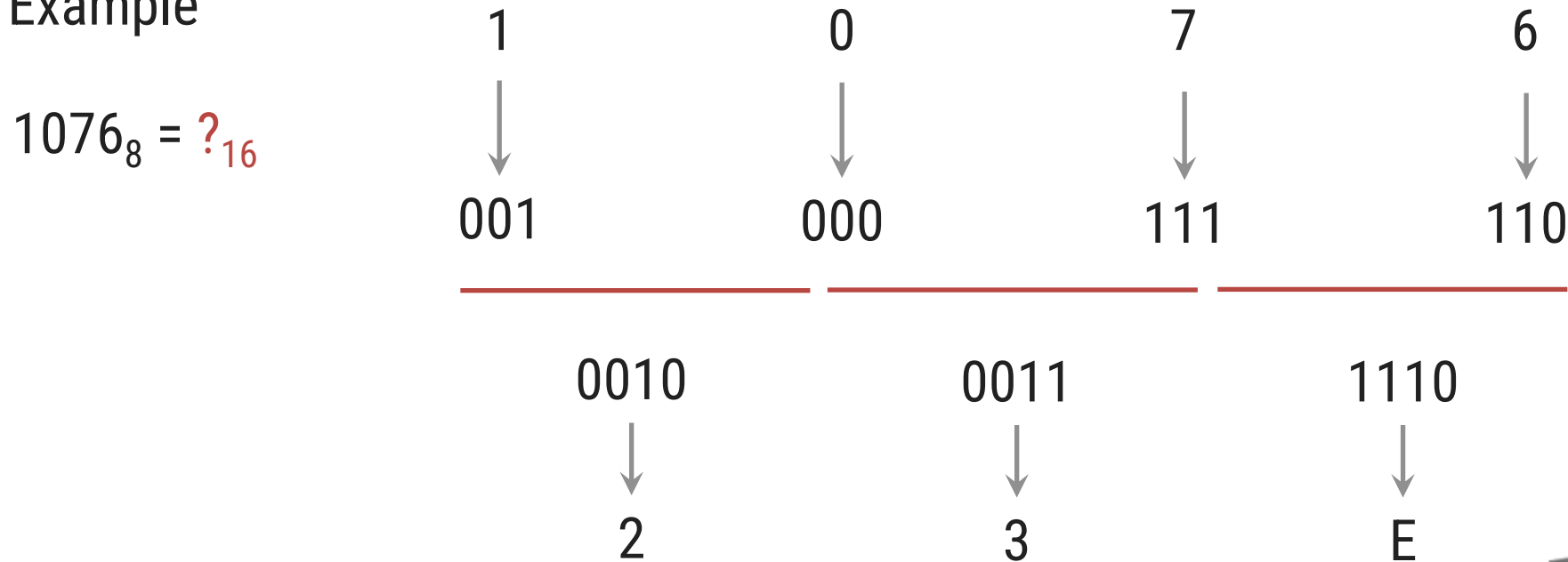


Marwadi
University

11) Octal to Hexa-Decimal

- ▶ Technique
- ▶ Direct conversion method is not available for it.
 - ➔ Convert Octal to Binary
 - ➔ these bits are grouped in **four bits**.
 - ➔ Convert Binary to Hexa-Decimal

▶ Example



$$1076_8 = 23E_{16}$$



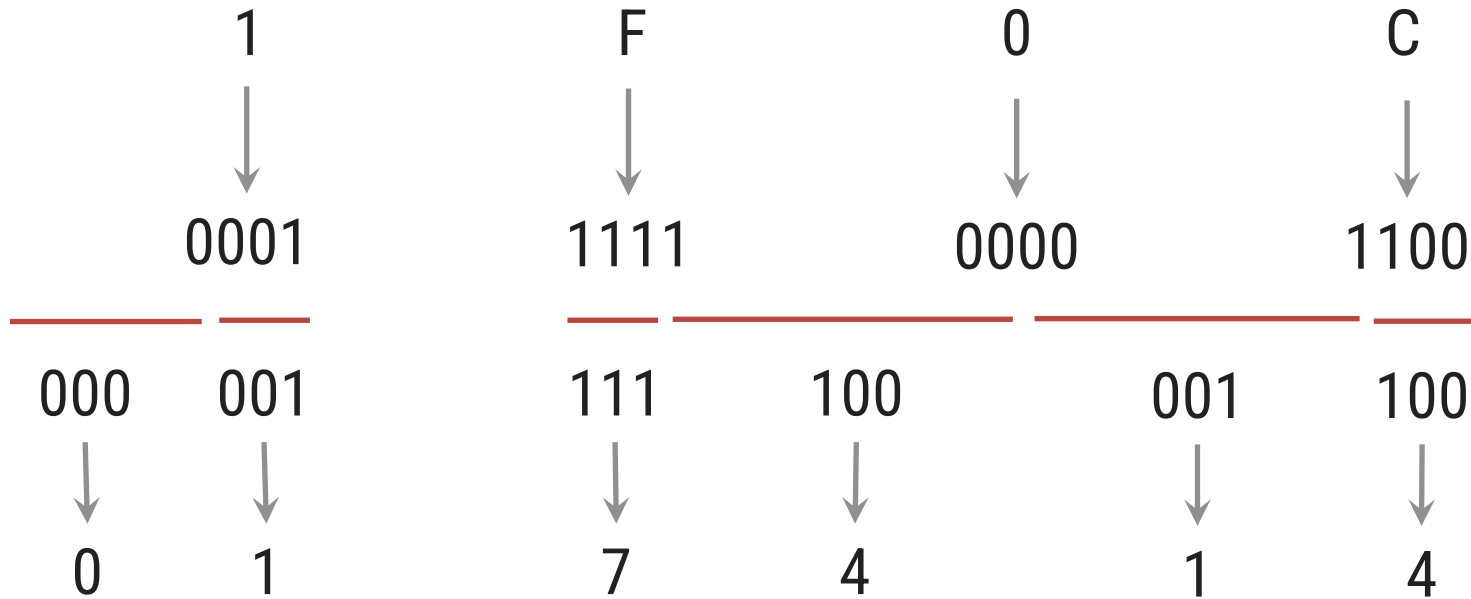
12)Hexa-Decimal to Octal

► Technique

- Convert Hexa-Decimal to Binary
- these bits are grouped in **three bits**.
- Convert Binary to Octal

► Example

$$1F0C_{16} = ?_8$$



$$1F0C_{16} = 17414_8$$



Exercise – Convert ...

Decimal	Binary	Octal	Hexa-decimal
33			
	1110101		
		703	
			1AF



Exe

Answer

Decimal	Binary	Octal	Hexa-decimal
33	100001	41	21
117	1110101	165	75
451	111000011	703	1C3
431	110101111	657	1AF



Marwadi
University

Exercise – Convert ...

Decimal	Binary	Octal	Hexa- decimal
29.8			
	101.1101		
		3.07	
			C.82

Don't use a calculator!

Exe

Answer

Decimal	Binary	Octal	Hexa- decimal
29.8	11101.110011...	35.63...	1D.CC...
5.8125	101.1101	5.64	5.D
3.109375	11.000111	3.07	3.1C
12.5078125	1100.10000010	14.404	C.82



Marwadi
University

Floating Point Representation

- Scientific notation for Decimal number system to represent very large or small number is ,
 - 1) $976,000,000,000,000 = 9.76 * 10^{(+14)}$
 - 2) $0.000000000000000976 = 9.76 * 10^{(-14)}$.
- **Decimal point** is put on a convenient location and use the **exponent of 10 to keep track of that decimal point**
- This allows a range of very large and very small numbers to be represented with only a few digits.
- **Normalized Floating point** number: if most significant digit of the number is non-zero. i.e. **350 is normalized but 00035 is not normalized**

Floating Point Representation

- Scientific notation of binary number is Floating point representation.

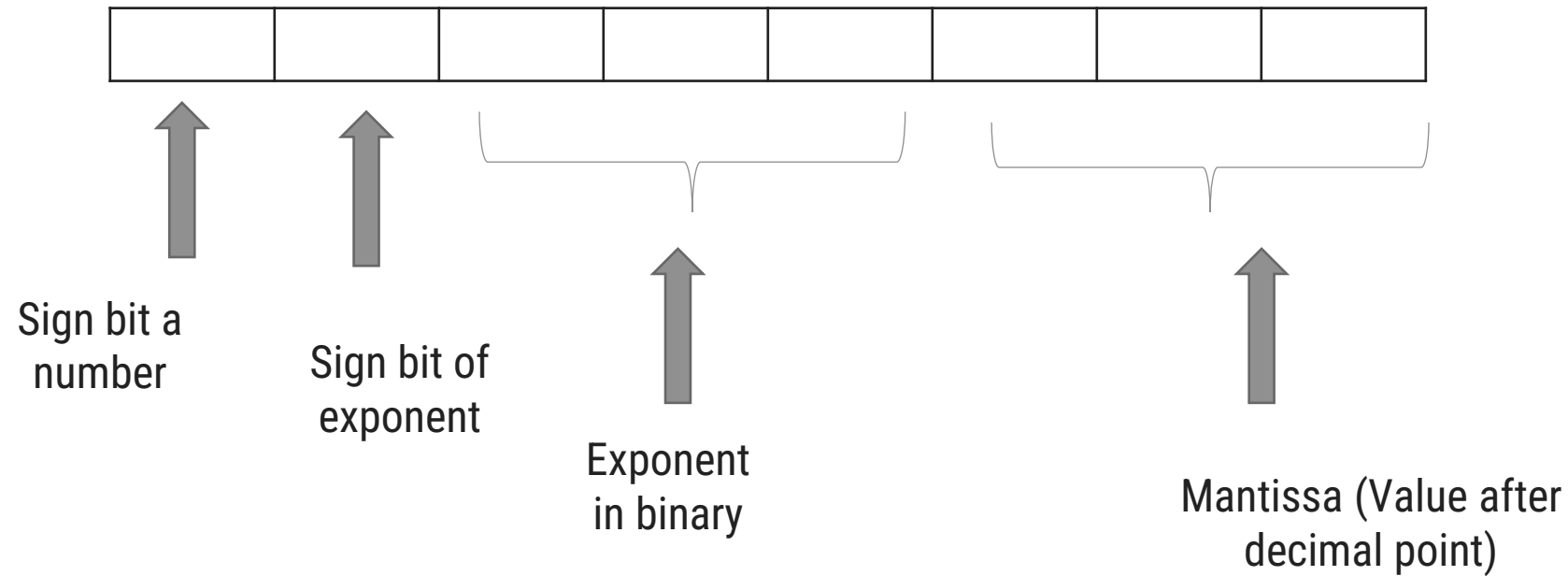
- A floating-point (FP) number :

$$\pm m \cdot b^e$$

1. **m** = mantissa - represents the fraction part of the number,
2. **e** = **exponent**, :- the position of the decimal(binary) point
3. **b** = base (radix) of the exponent.

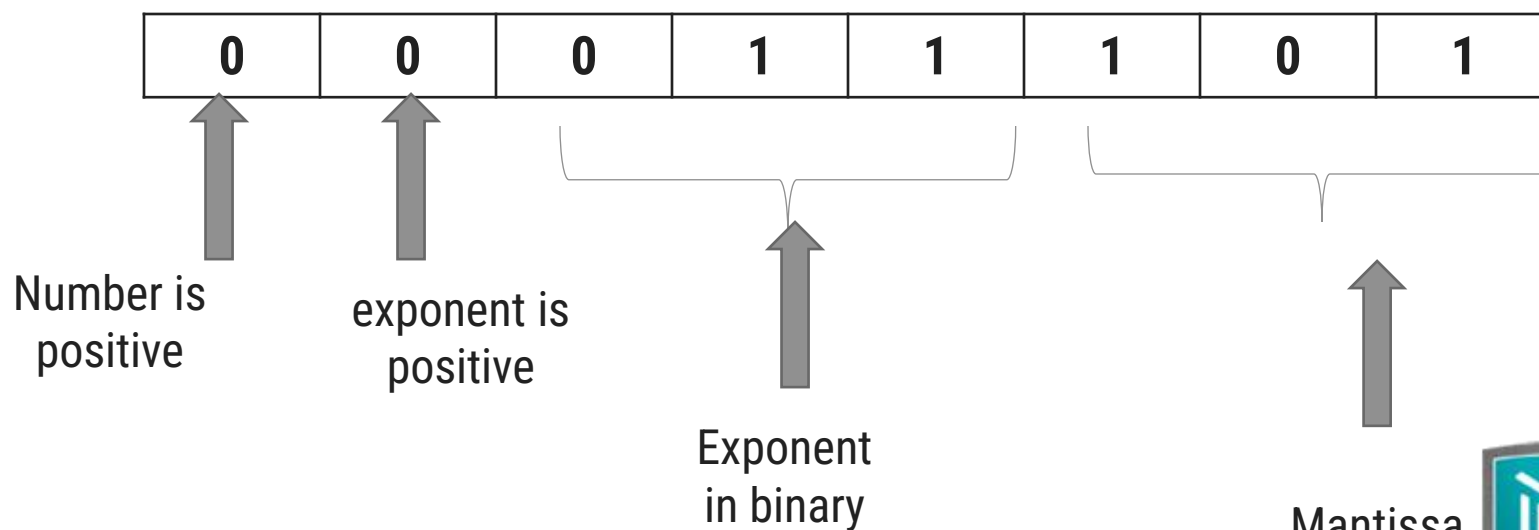


Floating point Representation



Representation of number in floating point method

- $(13.8)_{10} = (1101.11001...)_{2}$
 $= (1.10111001..) * 2^3$
 $= (1.1011) * 2^{(011)_2}$



Example -2

- $(-13.9)_{10} = - (1101.11100\dots)_2$
 $= - (1.10111100) * 2^3$
- $= - (1.10111100) * 2^{(011)_2}$
- $= - (1.1011) * 2^{(011)_2}$

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



Floating Point Representation

- Two main Standard:
 - 1) ANSI(American National Standard Institute)
 - 2) IEEE(Institute of Electrical and Electronics Engineers)

32 bit floating point number in ANSI(American National Standard Institute)

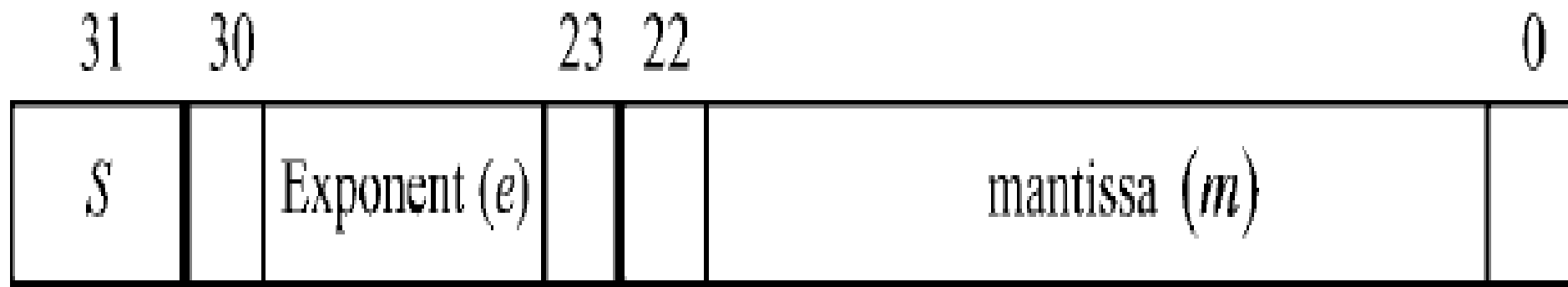


Figure Representation of a floating-point number

Floating Point Representation

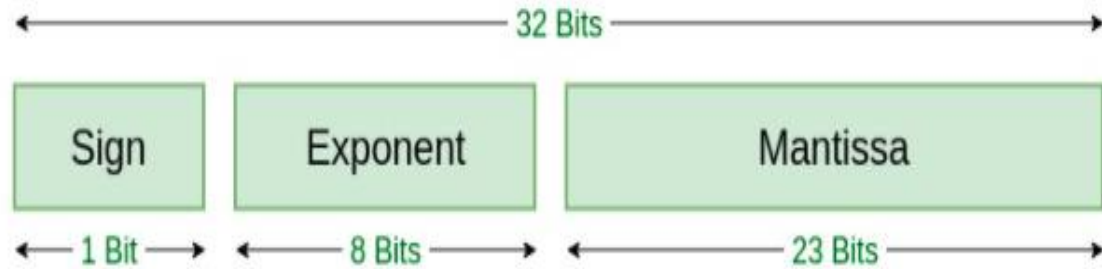
- The most commonly used format for representing floating-point numbers is the IEEE-754 standard.
- Established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE).

Figure 1.1 Characteristic parameters of IEEE-754 formats.

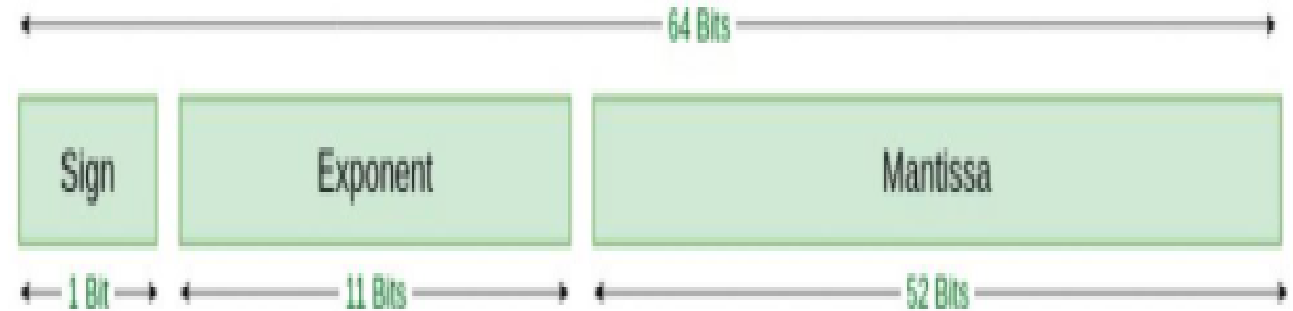
Precision	Sign (bits)	Exponent (bits)	Mantissa (bits)	Total length (bits)
Single	1	8	23	32
Single-extended	1	≥ 11	≥ 32	≥ 44
Double	1	11	52	64
Double-extended	1	≥ 15	≥ 64	≥ 80



Floating Point Representation- IEEE-754



Single Precision
IEEE 754 Floating-Point Standard



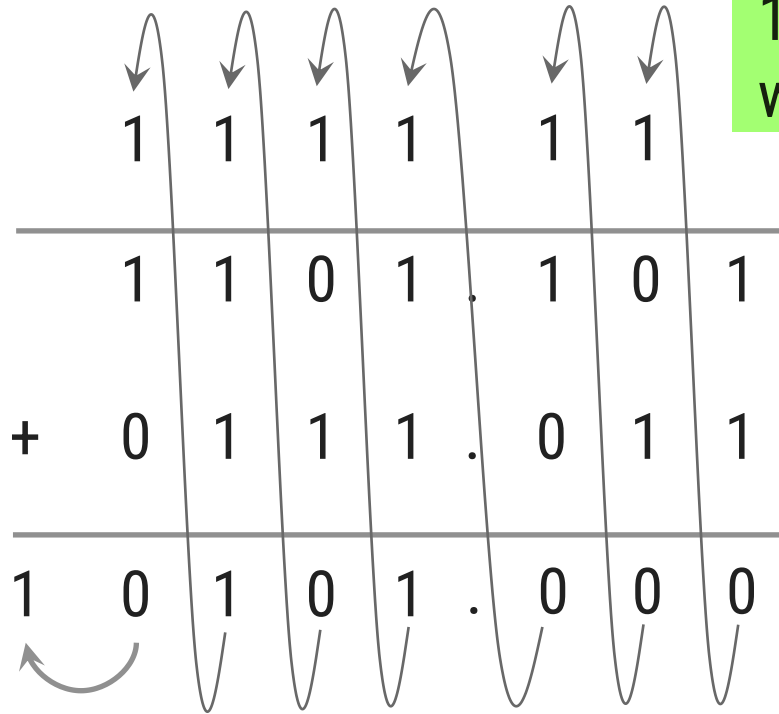
Double Precision
IEEE 754 Floating-Point Standard



Binary Addition & Subtraction

► Rules for binary addition

$0 + 0 = 0$
 $0 + 1 = 1$
 $1 + 0 = 1$
 $1 + 1 = 10$ i.e. 0
with a carry of 1

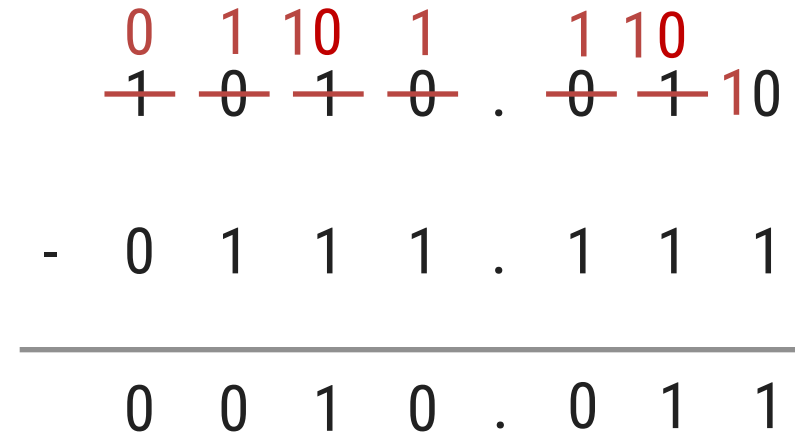


A binary addition problem: $1101.101 + 0111.011$. The result is 10101.000 . Curved arrows show the carry chain: from the first column (1+0=1), to the second (1+1=10, carry 1), to the third (0+1+1=10, carry 1), to the fourth (1+1+1=11, carry 1), to the fifth (1+0+1=10, carry 1), to the sixth (1+0=1), and finally to the seventh column where the final carry 1 is added to 0 to get 1.

$$\begin{array}{r} 1101.101 \\ + 0111.011 \\ \hline 10101.000 \end{array}$$

► Rules for binary subtraction

$0 - 0 = 0$
 $1 - 1 = 0$
 $1 - 0 = 1$
 $0 - 1 = 1$, with
a borrow 1



A binary subtraction problem: $1101.101 - 0111.111$. The result is 0010.011 . Red annotations show borrowing: the first column (1-1=0), the second (0-1 requires borrowing, becomes 10-1=1), the third (10-1=1), the fourth (1-0=1), the fifth (1-1=0), the sixth (0-1 requires borrowing, becomes 10-1=1), and the seventh (1-1=0). The final result is 0010.011.

$$\begin{array}{r} 01101.101 \\ - 00111.111 \\ \hline 00100.011 \end{array}$$



Binary Multiplication & Division

► Multiplication

$$\begin{array}{r} 10111 \\ x 10011 \\ \hline 10111 \\ 10111 \\ 00000 \\ 00000 \\ 10111 \\ \hline 110110101 \end{array}$$

► Division

$$\begin{array}{r|l} 110 & 101101 \mid 0111.1 \\ & \underline{000} \downarrow \\ & 1011 \downarrow \\ & 110 \downarrow \\ & 1010 \downarrow \\ & 110 \downarrow \\ & 1001 \\ & 110 \\ & \underline{110} \\ & 110 \\ & \underline{110} \\ & 000 \end{array}$$



Signed Binary Numbers

- ▶ Two ways of representing signed numbers:
 - 1) Sign-magnitude form, 2) Complement form.
- ▶ Most of computers use complement form for negative number notation.
- ▶ 1's complement and 2's complement are two different methods in this type.

1's Complement

- ▶ 1's complement of a binary number is obtained by subtracting each digit of that binary number from 1.
- ▶ Example

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \\ - \ 1 \ 1 \ 0 \ 1 \\ \hline 0 \ 0 \ 1 \ 0 \end{array}$$

(1's complement of 1101)

$$\begin{array}{r} 1 \ 1 \ 1 \ . \ 1 \ 1 \\ - \ 1 \ 0 \ 1 \ . \ 0 \ 1 \\ \hline 0 \ 1 \ 0 \ . \ 1 \ 0 \end{array}$$

(1's complement of 101.01)

Shortcut: Invert the numbers from 0 to 1 and 1 to 0



Marwadi
University

2's Complement

- ▶ 2's complement of a binary number is obtained by adding 1 to its 1's complement.
- ▶ Example

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \\ - \ 1 \ 1 \ 0 \ 0 \\ \hline 0 \ 0 \ 1 \ 1 \\ + 1 \\ \hline 0 \ 1 \ 0 \ 0 \end{array}$$

(2's complement of 1100)

$$\begin{array}{r} 1 \ 1 \ 1 \ . \ 1 \ 1 \\ - \ 1 \ 0 \ 1 \ . \ 0 \ 1 \\ \hline 0 \ 1 \ 0 \ . \ 1 \ 0 \\ + 1 \\ \hline 0 \ 1 \ 0 \ . \ 1 \ 1 \end{array}$$

(2's complement of 101.01)

Shortcut: Starting from right side, all bits are same till first 1 occurs and then invert rest of the bits



Marwadi
University

Representation of negative number in 2's complement form

- Express -65.5 in 12 bit 2's complement form.

2	65	1
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
2	1	1
	0	

$$0.5 \times 2 = 1.0$$

So, result in 12-bit binary is as follows:

$$65.5_{10} = 01000001.1000_2$$

For negative number, we have to convert this into 2's complement form

$$-65.5_{10} = 10111110.1000_2$$

Accuracy in Binary Number Conversion

Example

- ▶ Convert $(0.252)_{10}$ to binary with an error less than 1%.

Solution

- ▶ Absolute value of allowable error is found by calculating 1% of the number

$$E_{allow} = 0.01 \times 0.252 = 0.00252_{10}$$

- ▶ Maximum error due to truncation is set to be less than allowable error by solving from $E_{10} = 2^{-n}$. This equation is written as

$$2^{-n} < 0.00252$$

- ▶ Inverting both sides of the inequality

$$2^n > 397$$

Accuracy in Binary Number Conversion

- ▶ Taking log of both sides and solving for n

$$n \log 2 = \log 397$$

$$n = \frac{\log 397}{\log 2} = 8.63 \approx 9 \text{ (next largest integer)}$$

- ▶ This indicates that the use of **9 bits** in the binary number will guarantee an error less than 1%.
- ▶ So the conversion is carried out to **9 places** which results in

$$0.252_{10} = 0.010000001_2$$

9's Complement

- ▶ 9's complement of a decimal number is obtained by subtracting each digit of that decimal number from 9.
- ▶ Example

$$\begin{array}{r} 9 \quad 9 \quad 9 \quad 9 \\ - \quad 3 \quad 4 \quad 6 \quad 5 \\ \hline 6 \quad 5 \quad 3 \quad 4 \end{array}$$

(9's complement of 3465)

$$\begin{array}{r} 9 \quad 9 \quad 9 \quad . \quad 9 \quad 9 \\ - \quad 7 \quad 8 \quad 2 \quad . \quad 5 \quad 4 \\ \hline 2 \quad 1 \quad 7 \quad . \quad 4 \quad 5 \end{array}$$

(9's complement of 782.54)



10's Complement

- ▶ 10's complement of a decimal number is obtained by adding 1 to its 9's complement.
- ▶ Example

$$\begin{array}{r} 9 \quad 9 \quad 9 \quad 9 \\ - \quad 3 \quad 4 \quad 6 \quad 5 \\ \hline 6 \quad 5 \quad 3 \quad 4 \\ + \quad \quad \quad 1 \\ \hline 6 \quad 5 \quad 3 \quad 5 \\ \text{(10's complement of 3465)} \end{array}$$

$$\begin{array}{r} 9 \quad 9 \quad 9 \quad . \quad 9 \quad 9 \\ - \quad 7 \quad 8 \quad 2 \quad . \quad 5 \quad 4 \\ \hline 2 \quad 1 \quad 7 \quad . \quad 4 \quad 5 \\ + \quad \quad \quad 1 \\ \hline 2 \quad 1 \quad 7 \quad . \quad 4 \quad 6 \\ \text{(10's complement of 782.54)} \end{array}$$

Subtraction using 9's complement & 10's complement

► Using 9's complement

- ➔ Obtain 9's complement of subtrahend
- ➔ Add the result to minuend and call it intermediate result
- ➔ If **carry is generated** then answer is **positive** and add the carry to Least Significant Digit (LSD)
- ➔ If there is **no carry** then answer is **negative** and take 9's complement of intermediate result and place negative sign to the result.

► Using 10's complement

- ➔ Obtain 10's complement of subtrahend
- ➔ Add the result to minuend
- ➔ If **carry is generated** then answer is **positive**, ignore carry and result itself is answer
- ➔ If there is **no carry** then answer is **negative** and take 10's complement of intermediate result and place negative sign to the result.



Subtraction using 9's complement (Examples)

► Example - 1

$$745.81 - 436.62$$

7	4	5	.	8	1		7	4	5	.	8	1		
-	4	3	6	.	6	2	$\xrightarrow{\text{9's complement}}$	+	5	6	3	.	3	7
<hr/>								<hr/>						
3	0	9	.	1	9			1	3	0	9	.	1	8
								$\xrightarrow{+}$						
								<hr/>						
								3	0	9	.	1	9	



Subtraction using 9's complement (Examples)

► Example - 2

436.62 - 745.81

$$\begin{array}{r}
 436.62 \\
 - 745.81 \xrightarrow{\text{9's complement}} + 254.18 \\
 \hline
 - 309.19
 \end{array}
 \qquad
 \begin{array}{r}
 436.62 \\
 + 254.18 \\
 \hline
 690.80 \\
 \text{9's complement} \swarrow \\
 - 309.19
 \end{array}$$

As carry is not generated, so take 9's complement of the intermediate result and add ' - ' sign to the result

Subtraction using 10's complement (Examples)

► Example - 1

$$745.81 - 436.62$$

7 4 5 . 8 1		7 4 5 . 8 1
- 4 3 6 . 6 2	10's complement →	+ 5 6 3 . 3 8
3 0 9 . 1 9		1 3 0 9 . 1 9
	↑ Ignore the carry	

Subtraction using 10's complement (Examples)

► Example - 2
436.62 - 745.81

$$\begin{array}{r} 436.62 \\ - 745.81 \\ \hline -309.19 \end{array} \xrightarrow{\text{10's complement}} \begin{array}{r} 436.62 \\ + 254.19 \\ \hline 690.81 \\ - 309.19 \\ \hline \end{array}$$

10's complement

As carry is not generated, so take 10's complement of the intermediate result and add ' - ' sign to the result

Subtraction using 1's complement & 2's complement

► Using 1's complement

- ➔ Obtain 1's complement of subtrahend
- ➔ Add the result to minuend and call it intermediate result
- ➔ If **carry is generated** then answer is **positive** and add the carry to Least Significant Digit (LSD)
- ➔ If there is **no carry** then answer is **negative** and take 1's complement of intermediate result and place negative sign to the result.

► Using 2's complement

- ➔ Obtain 2's complement of subtrahend
- ➔ Add the result to minuend
- ➔ If **carry is generated** then answer is **positive**, ignore carry and result itself is answer
- ➔ If there is **no carry** then answer is **negative** and take 2's complement of intermediate result and place negative sign to the result.



Subtraction using 1's complement (Examples)

► Example - 1

$$68.75 - 27.50$$

68.75		01000100.1100
- 27.50	$\xrightarrow{\text{1's complement}}$	+ 11100100.0111
<hr/>		<hr/>
+ 41.25		100101001.0011
		$\xrightarrow{\quad\quad\quad} + 1$
		<hr/>
		00101001.0100



Subtraction using 1's complement (Examples)

- Example - 2
43.25 - 89.75

$$\begin{array}{r} 43.25 \\ - 89.75 \\ \hline \end{array} \xrightarrow{\text{1's complement}} \begin{array}{r} 00101011.0100 \\ + 10100110.0011 \\ \hline 11010001.0111 \\ \hline 00101110.1000 \end{array}$$

1's complement

As carry is not generated, so take 1's complement of the intermediate result and add ' - ' sign to the result



Subtraction using 2's complement (Examples)

- Example - 1
68.75 – 27.50

$$\begin{array}{r} 68.75 \\ - 27.50 \\ \hline + 41.25 \end{array}$$

2's complement

$$\begin{array}{r} 01000100.1100 \\ + 11100100.1000 \\ \hline 10010100.0100 \\ 00101001.0100 \end{array}$$

Ignore Carry bit



Subtraction using 2's complement (Examples)

- Example - 2
43.25 - 89.75

$$\begin{array}{r} 43.25 \\ - 89.75 \\ \hline -46.50 \end{array} \xrightarrow{\text{2's complement}} \begin{array}{r} 00101011.0100 \\ + 10100110.0100 \\ \hline 11010001.1000 \\ \xrightarrow{\text{2's complement}} 00101110.1000 \end{array}$$

As carry is not generated, so take 2's complement of the intermediate result and add ' - ' sign to the result





Binary Codes

Section - 2



Marwadi
University

DECIMAL	HEXADEC	OCTAL	BINARY
0	0	000	00000000
1	1	001	00000001
2	2	002	00000010
3	3	003	00000011
4	4	004	00000100
5	5	005	00000101
6	6	006	00000110
7	7	007	00000111
8	8	010	00001000
9	9	011	00001001
10	A	012	00001010
11	B	013	00001011
12	C	014	00001100
13	D	015	00001101
14	E	016	00001110
15	F	017	00001111
16	10	020	00010000
17	11	021	00010001
18	12	022	00010010
19	13	023	00010011
20	14	024	00010100
21	15	025	00010101
22	16	026	00010110
23	17	027	00010111
24	18	030	00011000
25	19	031	00011001



Marwadi
University

8421 BCD Code (Natural BCD Code)

- ▶ Each decimal digit, 0 through 9, is coded by 4-bit binary number
- ▶ 8, 4, 2 and 1 weights are attached to each bit
- ▶ BCD code is weighted code
- ▶ 1010, 1011, 1100, 1101, 1110 and 1111 are illegal codes
- ▶ Less efficient than pure binary
- ▶ Arithmetic operations are more complex than in pure binary
- ▶ Example

Decimal	1	4
	↓	↓
BCD	0001	0100
Binary	1110	



Binary Codes

Decimal	Binary	BCD
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111

Decimal	Binary	BCD
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

BCD Addition

► Example - 1

						1	1	1				
	2	5		0	0	1	0	0	1	0	1	
+	1	3		+	0	0	0	1	0	0	1	1
<hr/>												
	3	8			0	0	1	1	1	0	0	0

No carry, no illegal code. So, this is the correct sum.

Rule: If there is an illegal code or carry is generated as a result of addition, then add 0110 to particular that 4 bits of result.



BCD Addition

► Example - 2

			1	111		
	679.6	0110	0111	1001	.0110	
+	536.8	+ 0101	0011	0110	.1000	
<hr/>						
	1216.4	1011	1010	1111	.1110	All are illegal codes
		+0110	+0110	+0110	+.0110	Add 0110 to each
<hr/>						
		10001	10000	10101	1.0100	Propagate carry
		+ 1 ↙	+ 1 ↙	+ 1 ↙	+ 1 ↙	
		0001	0010	0001	0110	.0100
		<hr/>				Corrected sum

BCD Subtraction

► Example - 1

$$\begin{array}{r} 38 \\ - 15 \\ \hline 23 \end{array}$$

$$\begin{array}{r} 00111000 \\ - 00010101 \\ \hline 00100011 \end{array}$$

No borrow. So, this is the correct difference.

Rule: If one 4-bit group needs to take borrow from neighbor, then subtract 0110 from the group which is receiving borrow.



BCD Subtraction

► Example - 2

206.7	0010	0000	0110	.0111	
- 147.8	- 0001	0100	0111	.1000	
<hr/>	<hr/>				
58.9	0000	1011	1110	.1111	Borrows are present
		-0110	-0110	-.0110	Subtract 0110
	<hr/>				
		0101	1000	.1001	Corrected difference

Excess Three (XS-3) Code

- ▶ Excess Three Code = 8421 BCD + 0011(3)
- ▶ XS-3 code is non-weighted BCD code
- ▶ Also known as self complementing code
- ▶ 0000, 0001, 0010, 1101, 1110 and 1111 are illegal codes
- ▶ Example

Decimal	1	4
	↓	↓
BCD	0001	0100
XS-3	0100	0111



XS-3 Addition

► Example

247.6	0101	0111	1010	.1001
+ 359.4	+ 0110	1000	1100	.0111
<hr/> 607.0	<hr/> 1011	1111	10110	1.0000
		+ 1 ↙	+ 1 ↙	
	<hr/> 1011	10000	0111	.0000
	+ 1 ↙			
	<hr/> 1100	0000	0111	.0000
	- 0011	+0011	+0011	+.0011
	<hr/> 1001	0011	1010	.0011

Carry generated

Propagate carry

Rule: Add 0011 to group which generated carry and Subtract 0011 to group which do not generated carry

Corrected Sum in XS-3



Marwadi
University

XS-3 Subtraction

► Example

57.6	1000	1010	.1001
- 27.8	- 0101	1010	.1011
<hr/> 29.8	<hr/> 0010	1111	.1110
	+0011	-0011	-.0011
	<hr/> 0101	1100	.1011

Rule: Subtract 0011 to group which generated borrow and Add 0011 to group which do not generated borrow

Gray Code

- ▶ Only one bit changes between each pair of successive code words (**Unit distance code**).
- ▶ Gray code is a reflected code.
- ▶ Gray codes are designed recursively using following rules:
 - 1-bit Gray code has two code words, **0 and 1**.
 - The **first 2^n** code words of an $(n+1)$ -bit Gray code equal the code words of n -bit gray code, written **in order** with a leading **0 appended**.
 - The **last 2^n** code words of an $(n+1)$ -bit Gray code equal the code words of n -bit gray code, but written **in reverse order** with a leading **1 appended**.

Gray Code				Decimal	4-bit Binary
1-bit	2-bit	3-bit	4-bit		
0	0 0	0 0 0	0 0 0 0	0	0000
1	0 1	0 0 1	0 0 0 1	1	0001
	1 1	0 1 1	0 0 1 1	2	0010
	1 0	0 1 0	0 0 1 0	3	0011
		1 1 0	0 1 1 0	4	0100
		1 1 1	0 1 1 1	5	0101
		1 0 1	0 1 0 1	6	0110
		1 0 0	0 1 0 0	7	0111
			1 1 0 0	8	1000
			1 1 0 1	9	1001
			1 1 1 1	10	1010
			1 1 1 0	11	1011
			1 0 1 0	12	1100
			1 0 1 1	13	1101
			1 0 0 1	14	1110
			1 0 0 0	15	1111

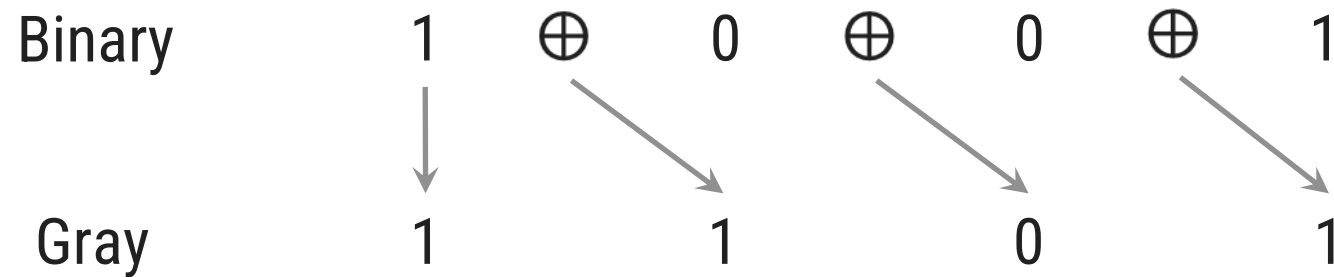


Binary to Gray and Gray to Binary Conversion

- Conversion of n-bit Binary number (B) to Gray Code (G) is as follows:

$G_n = B_n$	$G_{n-1} = B_n \oplus B_{n-1}$	$G_{n-2} = B_{n-1} \oplus B_{n-2} \dots$	$G_1 = B_2 \oplus B_1$
-------------	--------------------------------	--	------------------------

- Example: Convert $(1001)_2$ to Gray Code.

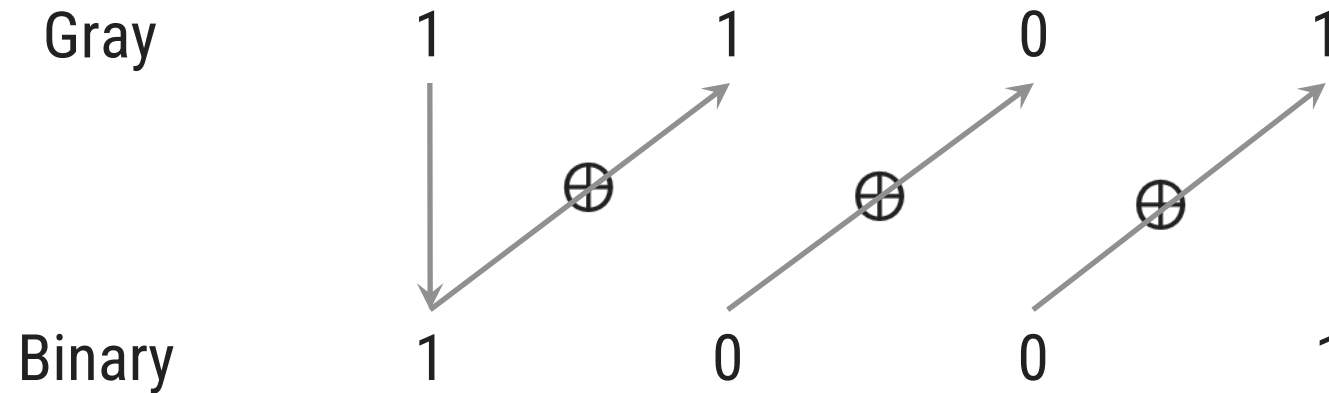


Gray to Binary Conversion

- Conversion of n-bit Gray Code (G) to Binary Number (B) is as follows:

$B_n = G_n$	$B_{n-1} = B_n \oplus G_{n-1}$	$B_{n-2} = B_{n-1} \oplus G_{n-2} \dots$	$B_1 = B_2 \oplus G_1$
-------------	--------------------------------	--	------------------------

- Example: Convert Gray code 1101 to Binary.



Error-Detecting Codes

- ▶ Noise can alter or distort the data in transmission.
- ▶ The 1s may get changed to 0s and 0s to 1s.
- ▶ Because digital systems must be accurate to the digit, errors can pose a serious problem.
- ▶ Single bit error should be detect & correct by different schemes.
- ▶ **Parity**, **Check Sums** and **Block Parity** are few examples of error detecting code.

Parity

- ▶ Parity bit is the simplest technique.
- ▶ There are two types of parity – **Odd parity** and **Even parity**.
- ▶ For odd parity, the parity is set to a 0 or a 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an odd number.
- ▶ For even parity, the parity is set to a 0 or a 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an even number.
- ▶ For example, 0110 binary number has “1” as Odd parity and “0” as Even parity.
- ▶ Detect a single-bit error but can not detect two or more errors within the same word.
- ▶ In any practical system, there is always a finite probability of the occurrence of single error.
- ▶ E.g. In an even-parity scheme, code 10111001 is erroneous because number of 1s is odd(5), while code 11110110 is error free because number of 1s is even(6).



Check Sums

- ▶ Simple parity can not detect two errors within the same word.
- ▶ Added to the sum of the previously transmitted words
- ▶ At the transmission, the **check sum** up to that time is sent to the receiver.
- ▶ The receiver can check its sum with the transmitted sum.
- ▶ If the **two sums are the same**, then **no errors** were detected at the receiver end.
- ▶ If there is an error, the receiving location can ask for retransmission of the entire data.
- ▶ This type of transmission is used in teleprocessing system.

Block Parity

	01011011	0		01011011	0		01011011	0
	10010101	1		10010101	1		10010101	1
	01101110	0		01101110	0	←	01101110	0
	11010011	0		11010011	0		10000011	0
	10001101	1		10001101	1		10001101	1
	01110111	1		01110111	1		01110111	1
Parity Row →	01110110	0		01110110	0		01110110	0
		↑		↑			↑	↑
		Parity Column						



Error Correcting Code

- ▶ 7-bit Hamming Code is widely used error correcting code, containing 4 bits of data and 3 bits of even parity.

- ▶ Pattern: $P_1 P_2 D_3 P_4 D_5 D_6 D_7$

- ▶ Group - 1: $P_1 D_3 D_5 D_7$

- ▶ Group - 2: $P_2 D_3 D_6 D_7$

- ▶ Group - 3: $P_4 D_5 D_6 D_7$

- ▶ Example: Data = 1101

$$P_1 P_2 D_3 P_4 D_5 D_6 D_7 = P_1 P_2 1 P_4 1 0 1$$

$$P_1 D_3 D_5 D_7 = 1 1 1$$

$$P_2 D_3 D_6 D_7 = 1 0 1$$

$$P_4 D_5 D_6 D_7 = 1 0 1$$

- ▶ 7-bit Hamming Code is 1 0 1 0 1 0 1

- ▶ How to detect error?

- ▶ Example: Received data = 1001001

$$P_1 P_2 D_3 P_4 D_5 D_6 D_7 = 1 0 0 1 0 0 1$$

$$P_1 D_3 D_5 D_7 = 1 0 0 1 \text{ (No Error)}$$

$$P_2 D_3 D_6 D_7 = 0 0 0 1 \text{ (Error)}$$

$$P_4 D_5 D_6 D_7 = 1 0 0 1 \text{ (No Error)}$$

- ▶ The error word is $0 1 0 = 2_{10}$.

- ▶ Complement the 2nd bit (from left).

- ▶ Correct code is 1 1 0 1 0 0 1



UNIT - 3

Boolean Algebra

Section - 3



Marwadi
University

Boolean Algebra Laws

▶ AND laws

1. $A \cdot 0 = 0$ (*Null Law*)
2. $A \cdot 1 = A$ (*Identity Law*)
3. $A \cdot A = A$
4. $A \cdot \bar{A} = 0$

▶ Commutative laws

1. $A + B = B + A$
2. $A \cdot B = B \cdot A$

▶ OR laws

1. $A + 0 = A$ (*Null Law*)
2. $A + 1 = 1$ (*Identity Law*)
3. $A + A = A$
4. $A + \bar{A} = 1$

▶ Associative laws

1. $(A + B) + C = A + (B + C)$
2. $(A \cdot B)C = A(B \cdot C)$



Boolean Algebra Laws

▶ Distributive laws

1. $A(B + C) = AB + AC$

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

▶ Idempotent laws

1. $A \cdot A = A$

2. $A + A = A$

▶ Redundant Literal Rule

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

2. $A(\bar{A} + B) = AB$

▶ Absorption laws

1. $A + AB = A$

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

▶ De Morgan's Theorem

1. $\overline{A + B} = \bar{A}\bar{B}$

2. $\overline{AB} = \bar{A} + \bar{B}$

Break the line change the sign



Marwadi
University

Proof of $\overline{A + B + C} = \bar{A} \bar{B} \bar{C}$

L.H.S.

R.H.S.

A	B	C	A+B+C	$\overline{A+B+C}$	\bar{A}	\bar{B}	\bar{C}	$\bar{A} \bar{B} \bar{C}$
0	0	0	0	1	1	1	1	1
0	0	1	1	0	1	1	0	0
0	1	0	1	0	1	0	1	0
0	1	1	1	0	1	0	0	0
1	0	0	1	0	0	1	1	0
1	0	1	1	0	0	1	0	0
1	1	0	1	0	0	0	1	0
1	1	1	1	0	0	0	0	0

From truth table, it is clearly visible that L.H.S. = R.H.S. Hence, **the complement of a sum of variables is equal to the product of their individual complements.**



Marwadi
University

Proof of $\overline{ABC} = \bar{A} + \bar{B} + \bar{C}$

L.H.S.					R.H.S.			
A	B	C	ABC	\overline{ABC}	\bar{A}	\bar{B}	\bar{C}	$\bar{A} + \bar{B} + \bar{C}$
0	0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0	1
0	1	0	0	1	1	0	1	1
0	1	1	0	1	1	0	0	1
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	0	1	1
1	1	1	1	0	0	0	0	0

From truth table, it is clearly visible that L.H.S. = R.H.S. Hence, **the complement of a product of variables is equal to the sum of their individual complements.**



Reducing Boolean Expression (Example – 1)

► Reduce the expression $f = A + B[AC + (B + \bar{C})D]$

$$f = A + B[AC + (B + \bar{C})D]$$

$$f = A + B[AC + BD + \bar{C}D]$$

(Distributive law)

$$f = A + BAC + BBD + B\bar{C}D$$

(Distributive law)

$$f = A + ABC + BD + B\bar{C}D$$

($A.A = A$)

$$f = A(1 + BC) + BD(1 + \bar{C})$$

$$f = A + BD$$

($1 + A = 1$)



Reducing Boolean Expression (Example – 2)

► Reduce the expression $f = A[B + \bar{C}(\overline{AB + AC})]$

$$f = A[B + \bar{C}(\overline{AB + AC})]$$

$$f = A[B + \bar{C}(\overline{AB} \overline{AC})]$$

(De-Morgan's law)

$$f = A[B + \bar{C}(\bar{A} + \bar{B})(\bar{A} + \bar{C})]$$

(De-Morgan's law)

$$f = A[B + \bar{C}(\bar{A}\bar{A} + \bar{A}\bar{C} + \bar{B}\bar{A} + \bar{B}\bar{C})]$$

(Distributive law)

$$f = A[B + \bar{C}\bar{A} + \bar{C}\bar{A}\bar{C} + \bar{C}\bar{B}\bar{A} + \bar{C}\bar{B}\bar{C}]$$

(Distributive law)

$$f = A[B + \bar{C}\bar{A} + 0 + \bar{C}\bar{B}\bar{A} + 0]$$

(A.A' = 0)

$$f = AB + A\bar{C}\bar{A} + A\bar{C}\bar{B}\bar{A}$$

(Distributive law)

$$f = AB + 0 + 0$$

(A.A' = 0)

$$f = AB$$





Logic Gates

Section - 4



Marwadi
University

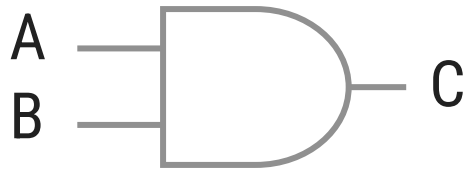
Logic Gates

- ▶ Most basic logical unit of the digital system is gate circuit.
- ▶ Types of gate circuits are as follows
 1. AND Gate
 2. OR Gate
 3. NOT Gate (Inverter)
 4. NOR Gate
 5. NAND Gate
 6. XOR Gate
 7. XNOR Gate

1. AND Gate

- ▶ AND Gate has an output which is normally at logic level “0” and only goes “HIGH” to a logic level “1” when ALL of its inputs are at logic level “1”

2-input AND Gate



Truth Table

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

Logic Notation

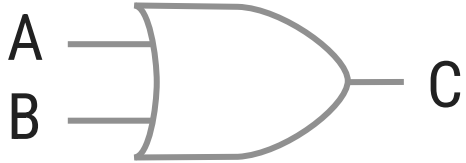
$$C = A \cdot B$$



2. OR Gate

- ▶ OR Gate or Inclusive-OR gate has an output which is normally at logic level “0” and only goes “HIGH” to a logic level “1” when one or more of its inputs are at logic level “1”.

2-input OR Gate



Truth Table

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

Logic Notation

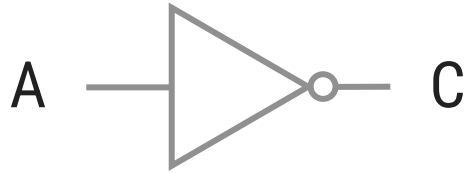
$$C = A + B$$



3. NOT (Inverter) Gate

- ▶ NOT gate has an output which is always opposite to input level.

Inverter Gate



Truth Table

A	C
0	1
1	0

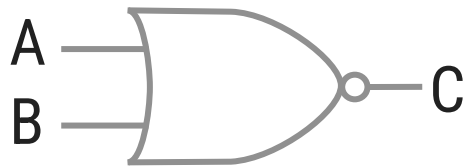
Logic Notation

$$C = \bar{A} \text{ or } C = A'$$

4. NOR Gate

- ▶ NOR Gate is an OR gate followed by an inverter.
- ▶ NOR Gate has an output which is normally at logic level “1” and only goes “LOW” to a logic level “0” when one or more of its inputs are at logic level “1”.

2-input NOR Gate



Truth Table

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

Logic Notation

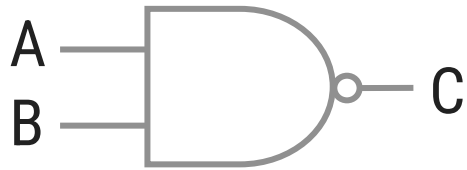
$$C = (A + B)'$$



5. NAND Gate

- ▶ NAND Gate is an AND gate followed by an inverter.
- ▶ NAND Gate has an output which is normally at logic level “1” and only goes “LOW” to a logic level “0” when ALL inputs are at logic level “1”.

2-input NAND Gate



Truth Table

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

Logic Notation

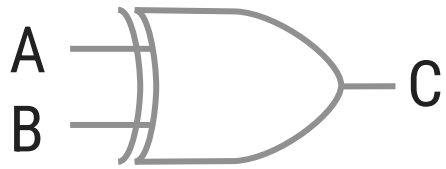
$$C = (A \cdot B)'$$



6. Exclusive-OR (X-OR) Gate

- ▶ X-OR gate that has 1 state when one and only one of its two inputs assumes a logic 1 state and has 0 state when all of its input are same.
- ▶ Also known as **anti-coincidence gate** or **inequality detector**.

2-input XOR Gate



Truth Table

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

Logic Notation

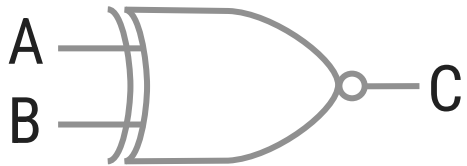
$$C = A \oplus B$$



7. Exclusive-NOR (X-NOR) Gate

- ▶ X-NOR gate that has 1 state when all of its input are same and has 0 state when one of its input has 0 state and other input is 1 state.
- ▶ Also known as **coincidence gate** or **equality detector**.

2-input XNOR Gate



Truth Table

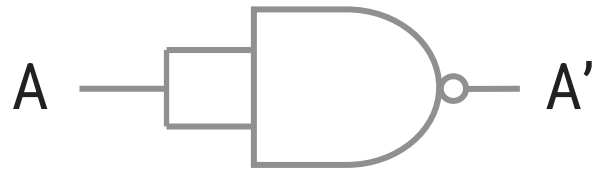
A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

Logic Notation

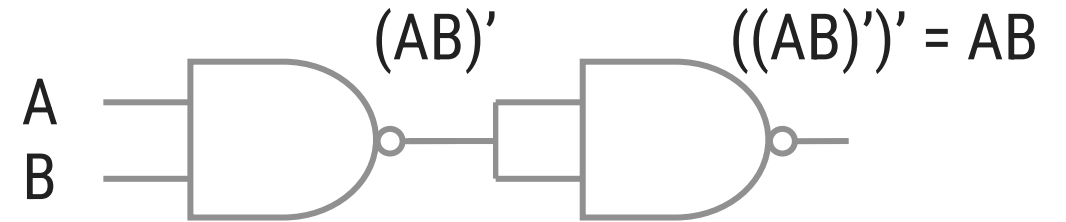
$$C = A \odot B$$



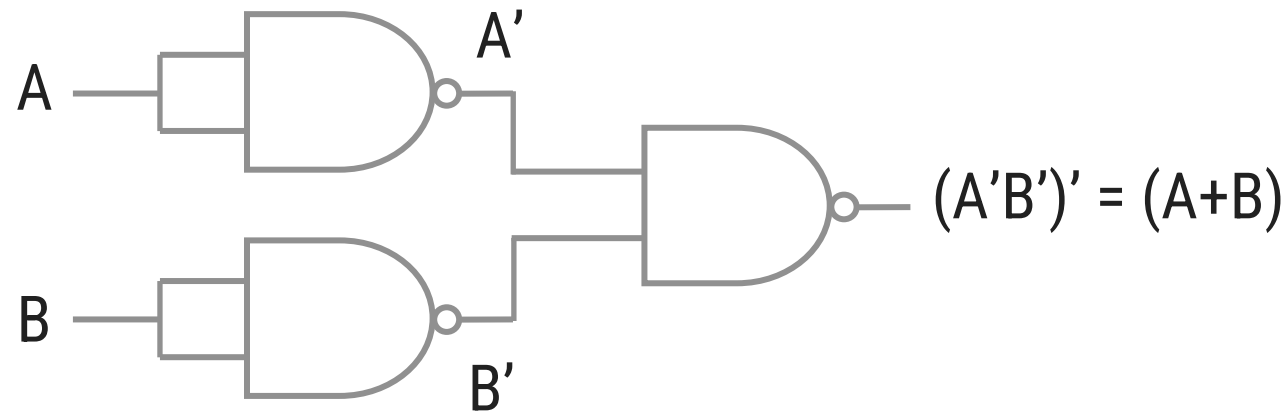
NAND as Universal Gate



NOT using NAND



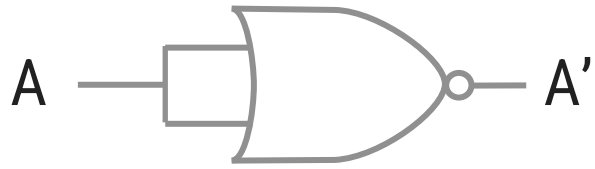
AND using NAND



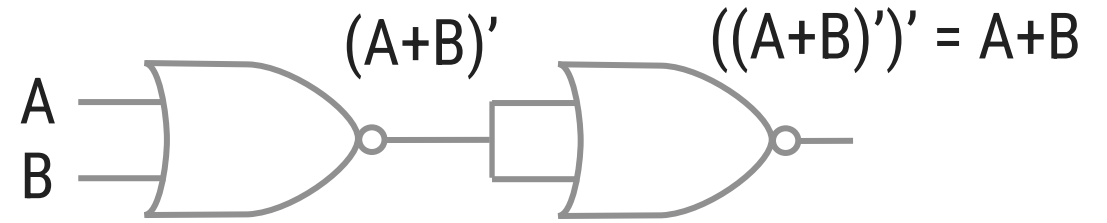
OR using NAND



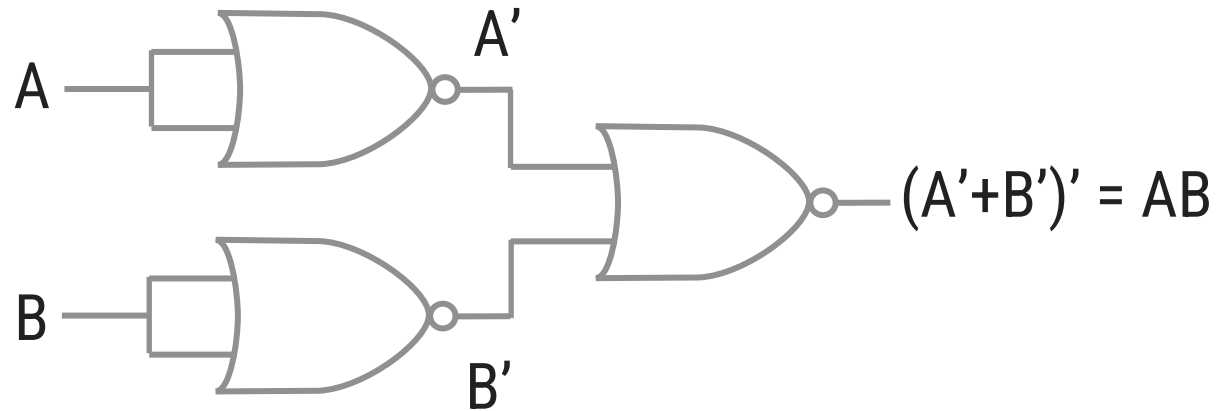
NOR as Universal Gate



NOT using NOR



OR using NOR



AND using NOR



