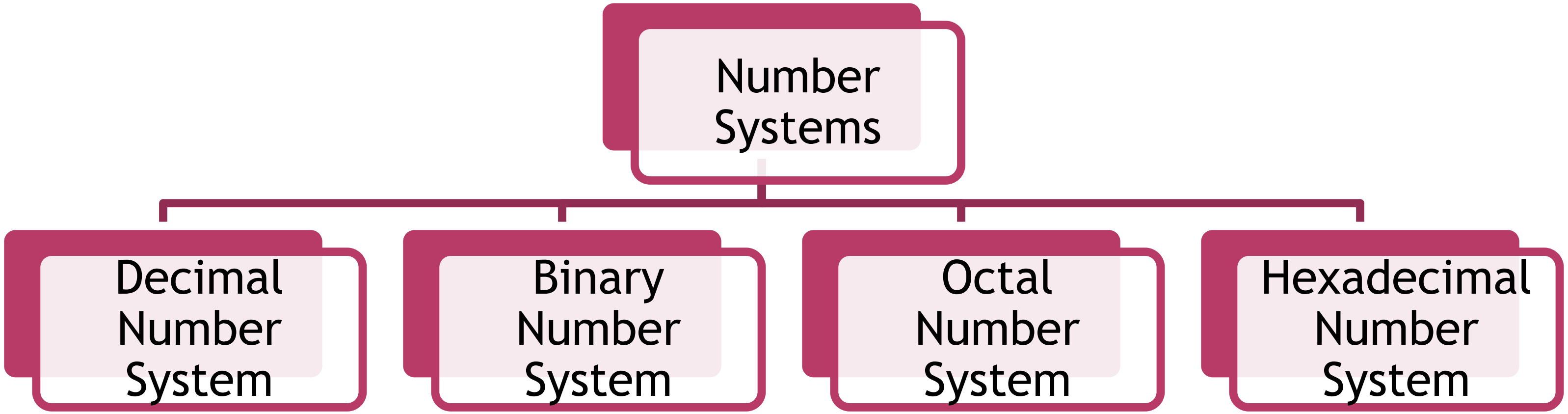
NUMBER SYSTEMS

- The technique to represent and work with numbers is called number system.
- The number systems are used to quantify the magnitude of something.
- The one way of representation of any quantity is numerical (Digital).

TYPES OF NUMBER SYSTEMS



DECIMAL NUMBER SYSTEM

- The decimal system contains ten unique symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
- In decimal number system involves ten symbols, we say that its base or radix is ten.
- The number system in which the weight of each digit depends on its relative position within the number, is called positional number system.

- Any positional number system can be expressed as sum of products of place value and the digit value .
- $d_n\ d_{n-1}\ d_{n-2} \ldots\ldots\ d_1\ d_0 .\ d_{-1}\ d_{-2}$ the decimal equivalent is

$(d_n \times 10^n) + (d_{n-1} \times 10^{n-1}) + \ldots\ldots\ldots + (d_1 \times 10^1) + (d_0 \times 10^0)$
$+ (d_{-1} \times 10^{-1}) + (d_{-2} \times 10^{-2}) \ldots\ldots$

BINARY NUMBER SYSTEM

- The binary system contains two unique symbols 0 and 1 .
- In binary number system involves two symbols, we say that its base or radix is two.
- The number system in which the weight of each digit depends on its relative position within the number is called positional number system.

- Any positional number system can be expressed as sum of products of place value and the digit value .
- $d_n\ d_{n-1}\ d_{n-2}\ .\ .\ .\ .\ .\ d_1\ d_0\ .\ d_{-1}\ d_{-2}$ the decimal equivalent is

$$(d_n \times 2^n) + (d_{n-1} \times 2^{n-1}) + \ldots\ldots\ldots +(d_1 \times 2^1) + (d_0 \times 2^0) + (d_{-1} \times 2^{-1}) + (d_{-2} \times 2^{-2}) \ldots\ldots$$

- The binary number system is used in digital computers because the switching circuits used in these computers use two-state devices such as transistors, diodes, etc.

OCTAL NUMBER SYSTEM

- The octal system contains eight unique symbols 0, 1, 2, 3, 4, 5, 6 and 7.
- In decimal number system involves eight symbols, we say that its base or radix is eight.
- The number system in which the weight of each digit depends on its relative position within the number, is called positional number system.

- Any positional number system can be expressed as sum of products of place value and the digit value .
- $d_n\ d_{n-1}\ d_{n-2}\ \ldots\ldots\ d_1\ d_0\ .\ d_{-1}\ d_{-2}$ the decimal equivalent is

  $(d_n \times 8^n) + (d_{n-1} \times 8^{n-1}) + \ldots\ldots\ldots + (d_1 \times 8^1) + (d_0 \times 8^0) + (d_{-1} \times 8^{-1}) + (d_{-2} \times 8^{-2}) \ldots\ldots$

HEXADECIMAL NUMBER SYSTEM

- The hexadecimal system contains sixteen unique symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,A,B,C,D,E,F
- In hexadecimal number system involves sixteen symbols, we say that its base or radix is sixteen.
- The number system in which the weight of each digit depends on its relative position within the number, is called positional number system.

- Any positional number system can be expressed as sum of products of place value and the digit value .
- $d_n\, d_{n-1}\, d_{n-2}\; \ldots\ldots\; d_1\; d_0 .\, d_{-1}\, d_{-2}$ the decimal equivalent is

$$(d_n \times 16^n) + (d_{n-1} \times 16^{n-1}) + \ldots\ldots\ldots + (d_1 \times 16^1) + (d_0 \times 16^0) + (d_{-1} \times 16^{-1}) + (d_{-2} \times 16^{-2}) \ldots\ldots$$

Decimal to any ‘base-r’

Integer numbers: Divide the given decimal number repeatedly by ‘r’ and collect the remainders. This must continue until the integer quotient becomes zero this method is called successive division.

EXAMPLES:

Decimal to Binary

2	24	
2	12	0
2	6	0
2	3	0
2	1	1
2	0	1

Write the Ramainders from bottom to top

$24_{10} = 11000_2$

Decimal to Octal

8	24	
8	3	0
8	0	3

Write the Ramainders from bottom to top

$24_{10} = 30_8$

Decimal to Hexa Decimal

16	24	
16	1	8
16	0	1

Write the Ramainders from bottom to top

$24_{10} = 18_{16}$

Decimal to any 'base-r'

EXAMPLES:

$(20)_{10} = (10100)_2$

2	20	0
2	10	0
2	5	1
2	2	0
	1	

		Remainder
8	123	3
8	15	7
	1	

$123_{10} = 173_8$

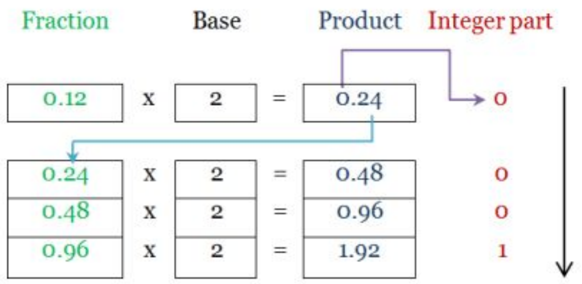
		Remainder Decimal	Remainder Hex
16	2598		
16	162	6	6
16	10	2	2
	0	10	A

$2598_{10} = A26_{16}$.

Fractional Numbers: First the given fraction number multiplied by ‘r’ to give an integer and fraction, the new fraction is multiplied by ‘2’ to give a new integer and a new fraction. This process continued until the fraction becomes zero (or) until the number of digits has sufficient accuracy.

EXAMPLES:

Decimal to Binary



For Fractional part, we get, (0.12) = (0.0001)

Convert $(105.15)_{10}$ to binary

2	105	
2	52	1
2	26	0
2	13	0
2	6	1
2	3	0
2	1	1
	0	1

Conversion of fraction 0.15_{10}

Given fraction	0.15
Multiply 0.15 by 2	0.30
Multiply 0.30 by 2	0.60
Multiply 0.60 by 2	1.20
Multiply 0.20 by 2	0.40
Multiply 0.40 by 2	0.80
Multiply 0.80 by 2	1.60

Reading the integers from top to bottom, $0.15_{10} = 0.001001_2$.
Therefore, the final result is, $105.15_{10} = 1101001.001001_2$.

Convert $(378.93)_{10}$ to octal

Conversion of 378_{10} to octal

Successive division	*Remainders*
$8 \mid 378$	
$8 \mid 47$	↑ 2
$8 \mid 5$	7
0	5

Read the remainders from bottom to top. Therefore, $378_{10} = 572_8$.

Conversion of 0.93_{10} to octal

0.93×8	7.44
0.44×8	3.52
0.52×8	↓ 4.16
0.16×8	1.28

Read the integers to the left of the octal point downwards. Therefore, $0.93_{10} = 0.7341_8$. Hence $378.93_{10} = 572.7341_8$.

Convert $(2598.675)_{10}$ to hexadecimal

Conversion of 2598_{10}

Successive division	Remainder Decimal	Remainder Hex
16 | 2598		
16 | 162	6	↑ 6
16 | 10	2	2
0	10	A

Reading the remainders upwards, $2598_{10} = A26_{16}$.

Therefore, $2598.675_{10} = A26.ACCC_{16}$.

Conversion of 0.675_{10}

Given fraction is 0.675

0.675×16	10.8
0.800×16	12.8
0.800×16	12.8
0.800×16	↓ 12.8

$0.675_{10} = 0.ACCC_{16}$.

Any ‘base-r’ to Decimal

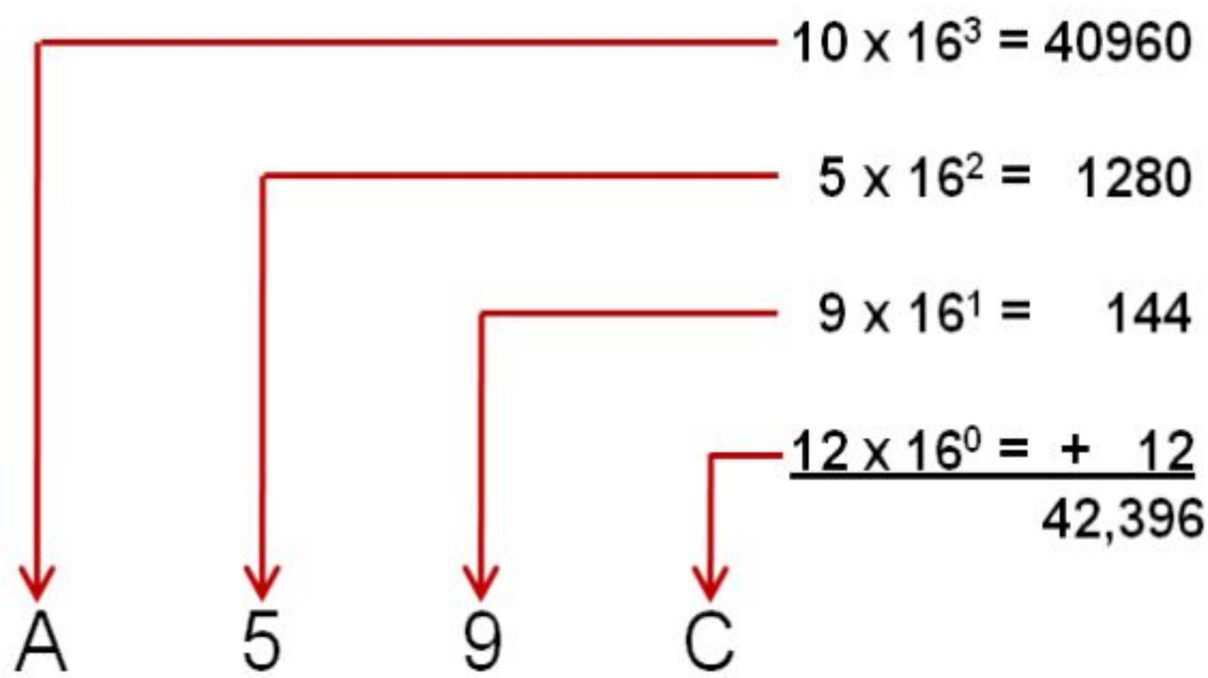
- The conversion from any base ‘r’ system to decimal system is by the positional weights methods.

Example 1:Convert $(10101)_2$ to decimal

$(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = (21)_{10}$

Example 2:

Convert $(A59C)_{16}$ to decimal = $(42396)_{10}$



Example 4: Convert $A0F9.0EB_{16}$ to decimal.

$$A0F9.0EB_{16} = (10 \times 16^3) + (0 \times 16^2) + (15 \times 16^1) + (9 \times 16^0) + (0 \times 16^{-1}) + (14 \times 16^{-2}) + (11 \times 16^{-3})$$

$$= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026$$
$$= 41209.0572_{10}$$

Example 5: Convert 4057.06_8 to decimal.

$$4057.06_8 = 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 0 \times 8^{-1} + 6 \times 8^{-2}$$

$$= 2048 + 0 + 40 + 7 + 0 + 0.0937$$

$$= 2095.0937_{10}$$

Example 6: Convert 11011.101_2 to decimal.

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

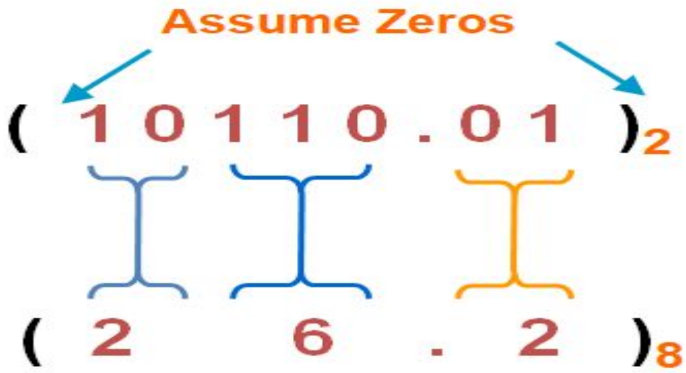
$$= (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$$
$$= 16 + 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125$$
$$= 27.625_{10}$$

BINARY – OCTAL CONVERSION

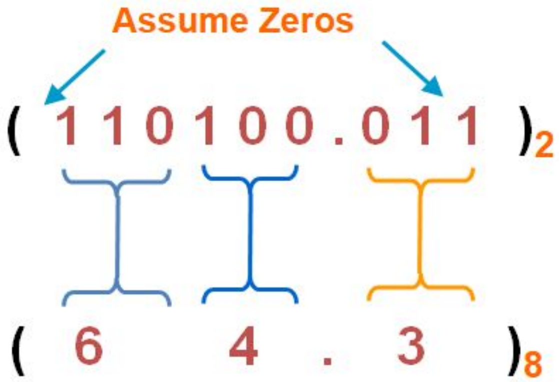
- $8 = 2^3$
- Each group of 3 bits represents an octal digit

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

EXAMPLE 1:



EXAMPLE 2:

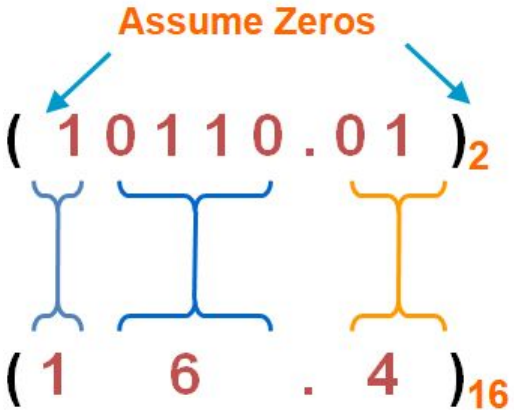


BINARY – HEXADECIMAL CONVERSION

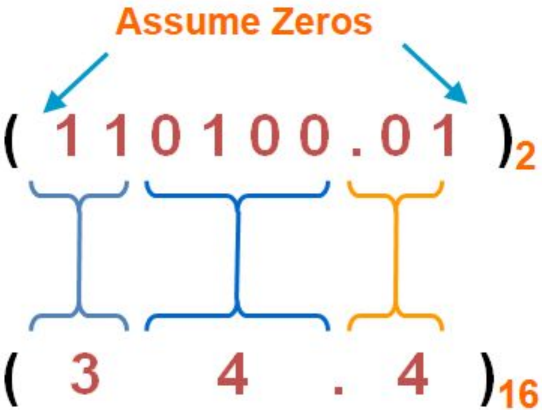
- $16 = 2^4$
- Each group of 4 bits represents a hexadecimal digit

Hex	Binary
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
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

EXAMPLE 1:

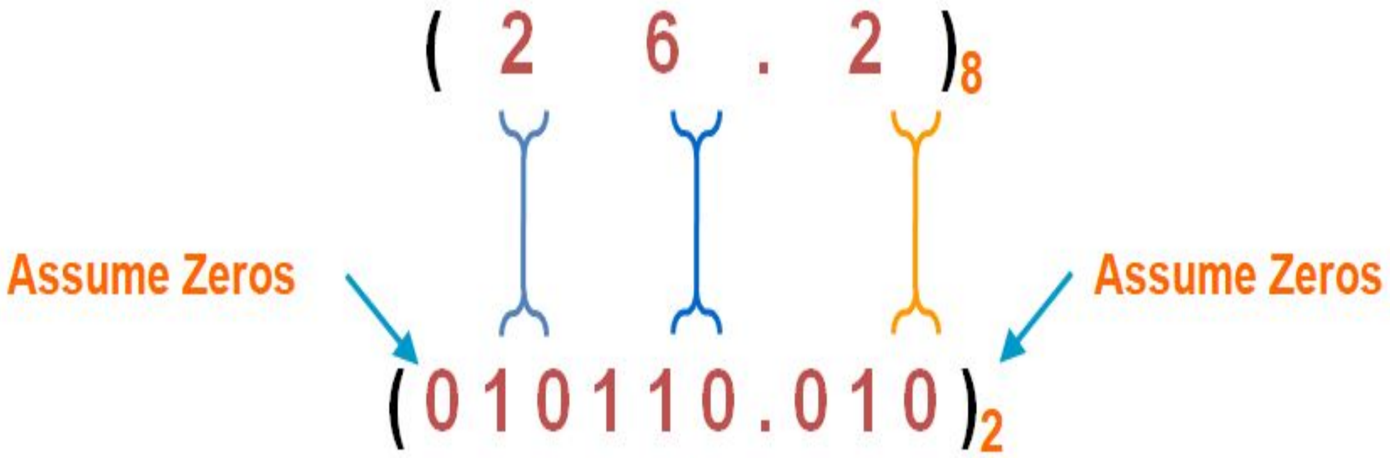


EXAMPLE 2:



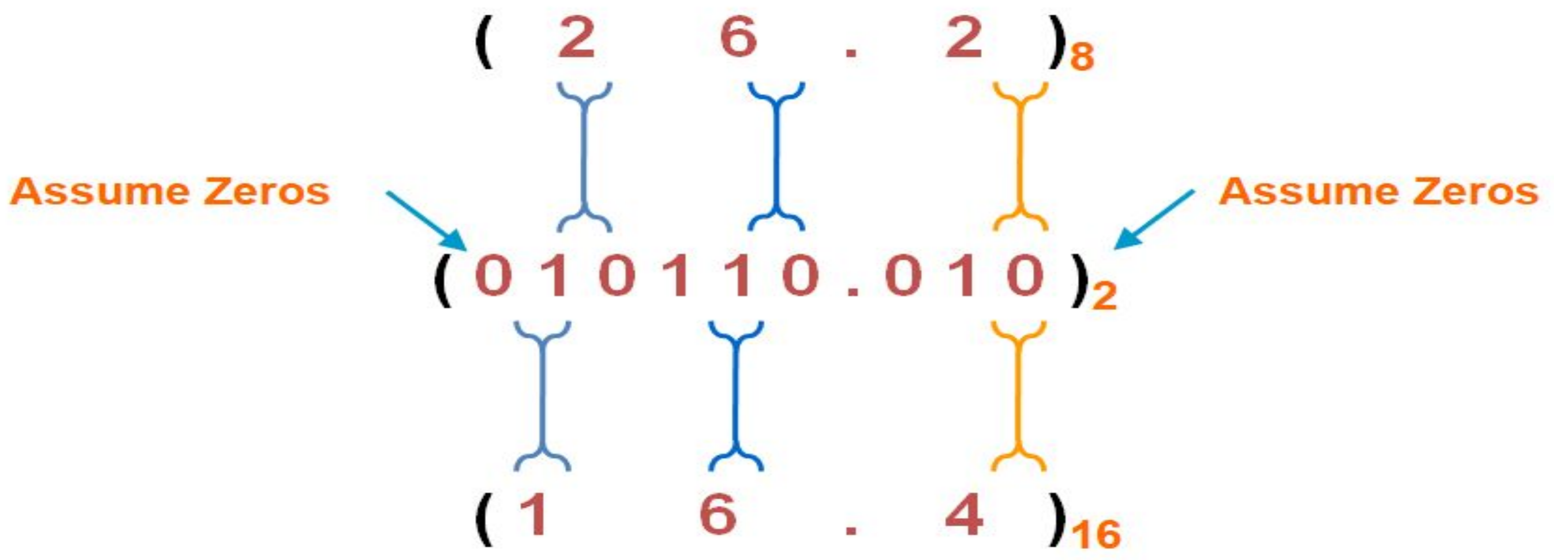
OCTAL - BINARY CONVERSION

EXAMPLE:



OCTAL – HEXADECIMAL CONVERSION

EXAMPLE:



HEXADECIMAL- BINARY CONVERSION

EXAMPLE:

$(3 \quad 4 \quad . \quad 4)_{16}$

Assume Zeros

$10100.01)_{2}$

BINARY-OCTAL CONVERSION

EXAMPLE: Convert $(101)_2$ to octal $= (5)_8$

Convert $(1101)_2$ to octal $= (15)_8$

$001101 = (15)_8$

Convert 110101.101010_2 to octal.

Groups of three bits are	110	101	.	101	010
Convert each group to octal	6	5	.	5	2
The result is			65.52_8		

BINARY-HEXADECIMAL CONVERSION

EXAMPLE: Convert $(1010)_2$ to hexadecimal $= (A)_{16}$

Convert $(10001)_2$ to hexadecimal $= (11)_{16}$

$00010001 = (11)_{16}$

Convert 01011111011.011111_2 to hexadecimal.

Given binary number is	01011111011.011111					
Groups of four bits are	0010	1111	1011	.	0111	1100
Convert each group to hex	2	F	B	.	7	C
The result is	$2FB.7C_{16}$					

HEXADECIMAL-OCTAL CONVERSION

EXAMPLE: Convert $(A2)_{16}$ to Octal $= (242)_8$

$(A2)_{16}$ is first converted to binary and then to octal

$(A2)_{16} = (1010\ 0010)_2 = (242)_8$

HEXADECIMAL-OCTAL CONVERSION

EXAMPLE:

$$(1\ 6\ .\ 4)_{16}$$

Assume Zeros

$$(0\,1\,0\,1\,1\,0\,.\,0\,1\,0)_{2}$$

Assume Zeros

$$(2\ 6\ .\ 2)_{8}$$

BINARY ADDITION

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

Add the following number $(1010)_2 + (0001)_2 = (1011)_2$

$$\begin{array}{r} 1010 \\ \underline{0001} \\ 1011 \end{array}$$

BINARY ADDITION

Add the following number $(1101.101)_2 + (111.011)_2 = (?)_2$

8	4	2	1		2^{-1}	2^{-2}	2^{-3}	(Column numbers)
	1	1	0	1	·	1	0	1
+		1	1	1	·	0	1	1
1	0	1	0	1	·	0	0	0

In the 2^{-3}'s column	$1 + 1 = 0$, with a carry of 1 to the 2^{-2} column
In the 2^{-2}'s column	$0 + 1 + 1 = 0$, with a carry of 1 to the 2^{-1} column
In the 2^{-1}'s column	$1 + 0 + 1 = 0$, with a carry of 1 to the 1's column
In the 1's column	$1 + 1 + 1 = 1$, with a carry of 1 to the 2's column
In the 2's column	$0 + 1 + 1 = 0$, with a carry of 1 to the 4's column
In the 4's column	$1 + 1 + 1 = 1$, with a carry of 1 to the 8's column
In the 8's column	$1 + 1 = 0$, with a carry of 1 to the 16's column

BINARY SUBTRACTION

A	B	Borrow	Difference
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

Subtract the following number $(1110)_2 - (1010)_2 = (0100)_2$

$$\begin{array}{r} 1110 \\ \underline{1010} \\ 0100 \end{array}$$

COMPLEMENT REPRESENTATION OF NEGATIVE NUMBERS

- The 2's (or 1's) complement system for representing signed numbers works like this:
- If the number is positive, the magnitude is represented in its true binary form and a sign bit 0 is placed in front of the MSB.
- If the number is negative, the magnitude is represented in its 2's (or 1's) complement form and a sign bit 1 is placed in front of the MSB.
- The 2's (or 1 's) complement operation on a signed number will change a positive number to a negative number and vice versa.

0	1	1	0	0	1	1	+51 (In sign magnitude form) f)

Sign bit Magnitude

1	1	1	0	0	1	1	-51 (In sign magnitude form)

Sign bit Magnitude

1	0	0	1	1	0	1	-51 (In2's complement form)

Sign bit Magnitude

1	0	0	1	1	0	0	-51 (In1's complement form)

Sign bit Magnitude

Characteristics of the 2's complement numbers:

The 2's complement numbers have the following properties:

1. There is one unique zero.
2. The 2's complement of 0 is 0.
3. The left most bit cannot be used to express a quantity.

   It is a sign bit.

   If it is a 1, the number is negative and if it is a 0, the number is positive.

4. For an n-bit word which includes the sign bit and also there are 2^{n-1} -1 positive integers, -2^{n-1} negative integers and one 0, for a total of 2^n unique states.

Characteristics of the 2’s complement numbers:

5. Significant information is contained in the Is of the positive numbers and 0s of the negative numbers.

6. A negative number may be converted into a positive number by finding its 2’s complement.

1’s complement numbers:

Represent – 99 and – 77.25 in 8-bit 1’s complement form.

We first write the positive representation of the given number in binary form and then complement each of its bits to represent the negative of the number.

(a) + 9 9 = 0 1 1 0 0 0 1 1

– 9 9 = 1 0 0 1 1 1 0 0 (In 1’s complement form)

(b) + 7 7 . 2 5 = 0 1 0 0 1 1 0 1 . 0 1 0 0

– 7 7 . 2 5 = 1 0 1 1 0 0 1 0 . 1 0 1 1 (In 1’s complement form)

Subtraction using 1’s complement :

Step1:Take 1’s complement of subtrahend and add it to the minuend.

Step 2:If there is carry out, bring the carry around and add it to the LSB

Step 3:If the MSB bit is Zero result is positive and is in true binary otherwise result is negative in its 1’s complement form

Subtraction using 1's complement:

Subtract 14 from 25 using the 8-bit 1's complement arithmetic.

$$
\begin{array}{r}
25 \\
-14 \\
\hline
+11
\end{array}
\quad \Rightarrow \quad
\begin{array}{r l}
0\,0\,0\,1\,1\,0\,0\,1 & \\
+\,1\,1\,1\,1\,0\,0\,0\,1 & \text{(In 1's complement form)} \\
\hline
1\;\,0\,0\,0\,0\,1\,0\,1\,0 & \\
+\,1 & \text{(End around carry)} \\
\hline
0\,0\,0\,0\,1\,0\,1\,1 &
\end{array}
$$

The MSB is a 0. So, the result is positive and is in pure binary. Therefore, the result is, $00001011 = +11_{10}$.

Add – 25 to + 14 using the 8-bit 1's complement method.

$$
\begin{array}{r}
+14 \\
-25 \\
\hline
-11
\end{array}
\quad \Rightarrow \quad
\begin{array}{r l}
0\,0\,0\,0\,1\,1\,1\,0 & \\
+\,1\,1\,1\,0\,0\,1\,1\,0 & \text{(In 1's complement form)} \\
\hline
1\,1\,1\,1\,0\,1\,0\,0 & \text{(No carry)}
\end{array}
$$

There is no carry. The MSB is a 1. So, the result is negative and is in its 1's complement form. The 1's complement of 11110100 is 00001011. The result is, therefore, -11_{10}.

Subtraction using 1's complement:

Add + 25 to – 25 using the 8-bit 1's complement method.

$$\begin{array}{r} +25 \\ -25 \\ \hline 00 \end{array} \Rightarrow \begin{array}{r} 0\,0\,0\,1\,1\,0\,0\,1 \\ +\,1\,1\,1\,0\,0\,1\,1\,0 \\ \hline 1\,1\,1\,1\,1\,1\,1\,1 \end{array} \quad \text{(In 1's complement form)}$$

There is no carry. The MSB is a 1. So, the result is negative and is in its 1's complement form. The 1's complement of 11111111 is 00000000. Therefore, the result is – 0.

Subtract 27.50 from 68.75 using the 12 bit 1's complement arithmetic.

$$\begin{array}{r} +68\,.\,75 \\ -27\,.\,50 \\ \hline +41\,.\,25 \end{array} \Rightarrow \begin{array}{r} 0\,1\,0\,0\,0\,1\,0\,0\,.\,1\,1\,0\,0 \\ +\,1\,1\,1\,0\,0\,1\,0\,0\,.\,0\,1\,1\,1 \\ \hline 1\;0\,0\,1\,0\,1\,0\,0\,1\,.\,0\,0\,1\,1 \\ +\,1 \\ \hline 0\,0\,1\,0\,1\,0\,0\,1\,.\,0\,1\,0\,0 \end{array} \quad \begin{array}{l} \\ \text{(In 1's complement form)} \\ \\ \text{(End around carry)} \\ \\ \end{array}$$

The MSB is a 0. So, the result is positive and is in its normal binary form. Therefore, the result is + 41.25.

2's complement numbers:

Express -45 in 8-bit 2's complement form
+45 in 8 bit form is 00101101
Obtain the 1's complement of 45 and add with 1

Positive expression of the given number	0 0 1 0 1 1 0 1
1's complement of it	1 1 0 1 0 0 1 0
Add 1	+ 1
Thus, the 2's complement form of – 45 is	1 1 0 1 0 0 1 1

2’s complement numbers:

Express -73.75 in 12-bit 2’s complement form

+73.75 in 12 bit form is 010001001.1100

Obtain the 1’s complement of 73.75 and add with 1

Positive expression of the given number	0 1 0 0 1 0 0 1 . 1 1 0 0
1’s complement of it	1 0 1 1 0 1 1 0 . 0 0 1 1
Add 1	+ 1
Thus, the 2’s complement of –73.75 is	1 0 1 1 0 1 1 0 . 0 1 0 0

Subtraction using 2’s complement :

Step1:Take 2’s complement of subtrahend and add it to the minuend.
Step 2:If there is carry out, ignore it.
Step 3:If the MSB bit is Zero result is positive and is in true binary otherwise result is negative in its 2’s complement form.Take 2’s complement to find the magnitude in binary.

Subtraction using 2's complement :

Subtract 14 from 46 using the 8-bit 2's complement arithmetic.

+ 1 4	=	0 0 0 0 1 1 1 0	
– 1 4	=	1 1 1 1 0 0 1 0	(In 2's complement form)

+ 4 6		0 0 1 0 1 1 1 0	
– 1 4	⇒	+ 1 1 1 1 0 0 1 0	(2's complement form of – 14)
+ 3 2		1 0 0 1 0 0 0 0 0	(Ignore the carry)

There is a carry, ignore it. The MSB is 0. So, the result is positive and is in normal binary form. Therefore, the result is + 00100000 = + 32.

Add – 75 to + 26 using the 8-bit 2's complement arithmetic.

+ 7 5	=	0 1 0 0 1 0 1 1	
– 7 5	=	1 0 1 1 0 1 0 1	(In 2's complement form)

Subtraction using 2's complement :

Add – 75 to + 26 using the 8-bit 2's complement arithmetic.

+ 2 6		0 0 0 1 1 0 1 0	
– 7 5	⇒	+ 1 0 1 1 0 1 0 1	(2's complement form of – 75)
– 4 9		1 1 0 0 1 1 1 1	(No carry)

There is no carry, the MSB is a 1. So, the result is negative and is in 2's complement form. The magnitude is 2's complement of 11001111, that is, 00110001 = 49. Therefore, the result is – 49.

Add – 45.75 to + 87.5 using the 12-bit 2's complement arithmetic.

+ 8 7 . 5		0 1 0 1 0 1 1 1 . 1 0 0 0	
– 4 5 . 7 5	⇒	+ 1 1 0 1 0 0 1 0 . 0 1 0 0	(–45.75 in 2's complement form)
+ 4 1 . 7 5		1 0 0 1 0 1 0 0 1 . 1 1 0 0	(Ignore the carry)

There is a carry, ignore it. The MSB is 0. So, the result is positive and is in normal binary form. Therefore, the result is + 41.75

Subtraction using 2's complement :

Add + 40.75 to – 40.75 using the 12-bit 2's complement arithmetic.

+ 4 0 . 7 5		0 0 1 0 1 0 0 0 . 1 1 0 0	
– 4 0 . 7 5	⇒	+ 1 1 0 1 0 1 1 1 . 0 1 0 0	(–40.75 in 2's complement form)
0 0 . 0 0		❶ 0 0 0 0 0 0 0 0 . 0 0 0 0	(Ignore the carry)

There is a carry, ignore it. The result is 0.

Perform N1+N2, N1+(-N2) for the following 8-bit numbers expressed in 2's complement representation

N1=00110010 N2=11111101

Here N1=00110010 is positive=$(+50)_{10}$

Here N2=11111101 is negative=$(-3)_{10}$

Subtraction using 2's complement :

$$N1 = 0\,0\,1\,1\,0\,0\,1\,0_2 = +\,0\,1\,1\,0\,0\,1\,0 = +\,5\,0_{10}, \quad \text{and}$$
$$N2 = 1\,1\,1\,1\,1\,1\,0\,1_2 = -\,0\,0\,0\,0\,0\,1\,1 = -\,3_{10}$$

$$\begin{array}{lrr} N1 + N2 = & 0\,0\,1\,1\,0\,0\,1\,0 & +\,5\,0\;+ \\ + & 1\,1\,1\,1\,1\,1\,0\,1 & -\,3 \\ \hline & 1\;0\,0\,1\,0\,1\,1\,1\,1 = +\,4\,7 & +\,4\,7 \end{array}$$

There is a carry. Ignore it. The MSB is a 0. So, answer is positive and the remaining bits indicate the magnitude in normal binary. It is + 47.

$$\begin{array}{lrr} N1 + (-N2) = & 0\,0\,1\,1\,0\,0\,1\,0 & +\,5\,0\;+ \\ + & 0\,0\,0\,0\,0\,0\,1\,1 & +\;3 \\ \hline & 0\,0\,1\,1\,0\,1\,0\,1 = +\,5\,3 & +\,5\,3 \end{array}$$

There is no carry. The MSB is a 0. So, the answer is positive and is in true binary form. It is + 53.

9's complement and 10's complement :

The 9's complement of a decimal number is obtained by subtracting each digit of that decimal number by 9

The 10's complement of a decimal number is obtained by adding 1 to its 9's complement

Find the 9's complement of 3465 and 782.54 and its 10's complement

$$\begin{array}{r} 9999 \\ -\underline{3465} \\ 6534 \end{array}$$

10's complement is 6534+1=6535

$$\begin{array}{r} 999.99 \\ -\underline{782.54} \\ 217.45 \end{array}$$

10's complement is 217.45+.01=217.46

9’s complement Subtraction :

Step 1:Take 9’s complement of subtrahend and add it to the minuend.
Step 2:If there is carry out, indicates result is positive otherwise negative.
Step 3:Add the carry to the LSD of this result to get the answer, if there is no carry result is negative in its 9’s complement form.Take 9’s complement to find the magnitude

9's complement Subtraction :

Subtract the following numbers using the 9's complement method.

(a) 745.81 – 436.62 (b) 436.62 – 745.81

(a)

$$\begin{array}{r} 745.81 \\ -\,436.62 \\ \hline 309.19 \end{array} \Rightarrow \begin{array}{rl} 745.81 & \\ +\,563.37 & \text{(9's complement of 436.62)} \\ \hline 1\;\;309.18 & \text{(Intermediate result)} \\ +1 & \text{(End around carry)} \\ \hline 309.19 & \text{(Answer)} \end{array}$$

The carry indicates that the answer is positive. So answer is +309.19.

(b)

$$\begin{array}{r} 436.62 \\ -\,745.81 \\ \hline -\,309.19 \end{array} \Rightarrow \begin{array}{rl} 436.62 & \\ +\,254.18 & \text{(9's complement of 745.81)} \\ \hline 690.80 & \text{(Intermediate result with no carry)} \end{array}$$

There is no carry indicating that the answer is negative. So, take the 9's complement of the intermediate result and put a minus sign.

The 9's complement of 690.80 is 309.19.

Therefore, the answer is –309.19.

10’s complement Subtraction :

Step 1:Take 10’s complement of subtrahend and add it to the minuend.
Step 2:If there is carry out, indicates result is positive otherwise negative.Here carry is ignored.
Step 3:if there is carry result is negative in its 10’s complement form.Take 10’s complement to find the magnitude

10's complement Subtraction :

Subtract the following numbers using the 10's complement method.

(a) 2928.54 – 416.73 (b) 416.73 – 2928.54

(a)

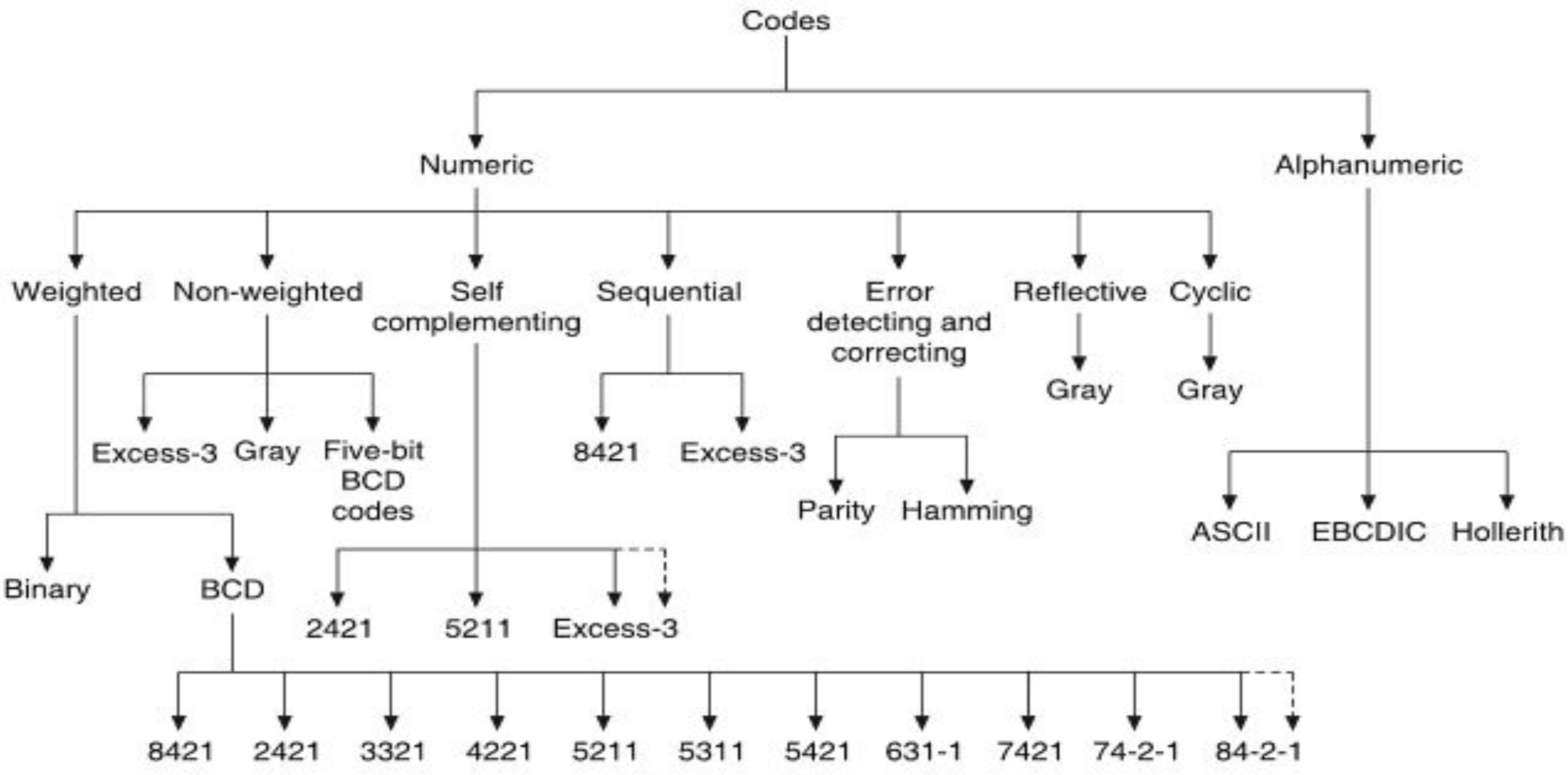
$$\begin{array}{r} 2928.54 \\ -\,0416.73 \\ \hline 2511.81 \end{array} \Rightarrow \begin{array}{r} 2928.54 \\ +\,9583.27 \\ \hline ❶\,2511.81 \end{array} \begin{array}{l} \\ \text{(10's complement of 416.73)} \\ \text{(Ignore the carry)} \end{array}$$

There is a carry indicating that the answer is positive. Ignore the carry.
The answer is 2511.81.

(b)

$$\begin{array}{r} 0416.73 \\ -\,2928.54 \\ \hline -\,2511.81 \end{array} \Rightarrow \begin{array}{r} 0416.73 \\ +\,7071.46 \\ \hline 7488.19 \end{array} \begin{array}{l} \\ \text{(10's complement of 2928.54)} \\ \text{(No carry)} \end{array}$$

There is no carry indicating that the answer is negative. So, take the 10's complement of the intermediate result and put a minus sign.
The 10's complement of 7488.19 is 2511.81.
Therefore, the answer is –2511.81.



Classification of codes.

Binary coded decimal codes

Decimal digit	8 4 2 1	2 4 2 1	5 2 1 1	5 4 2 1	6 4 2 –3	8 4 –2 –1	XS-3
0	0000	0000	0000	0000	0000	0000	0011
1	0001	0001	0001	0001	0101	0111	0100
2	0010	0010	0011	0010	0010	0110	0101
3	0011	0011	0101	0011	1001	0101	0110
4	0100	0100	0111	0100	0100	0100	0111
5	0101	1011	1000	1000	1011	1011	1000
6	0110	1100	1010	1001	0110	1010	1001
7	0111	1101	1100	1010	1101	1001	1010
8	1000	1110	1110	1011	1010	1000	1011
9	1001	1111	1111	1100	1111	1111	1100

BCD

- It is an acronym for the **Binary Coded Decimal exists from 0 to 9**
- Example: 10:0001 0000

  985:1001 1000 0101

BCD Addition:

- To add in BCD, add the BCD numbers by adding the 4-bit groups in each column starting from the LSD.
- If there is no carry out from the addition of any of the 4-bit groups, the sum term is a legal code

BCD

BCD Addition:

- If there is carry out from the addition of any of the 4-bit groups, add 0110 from the sum term of those groups (to skip illegal states).

BCD

Perform the following decimal additions in the 8421 code.

(a) 25 + 13 (b) 679.6 + 536.8

(a)	25	⇒	0010	0101	(25 in BCD)
	+13		+0001	0011	(13 in BCD)
	38		0011	1000	(No carry, no illegal code. So, this is the correct sum.)

(b)	679.6	⇒		0110	0111	1001	.0110	(679.6 in BCD)
	+ 536.8			+0101	0011	0110	.1000	(536.8 in BCD)
	1216.4			1011	1010	1111	.1110	(All are illegal codes)
				+0110	+0110	+0110	+.0110	(Add 0110 to each)
				❶0001	❶0000	❶0101	❶.0100	(Propagate the carry)
			+1	+1	+1	+1		
			0001	0010	0001	0110	.0100	(Corrected sum)
			1	2	1	6	. 4	

BCD

BCD Subtraction:

- To subtract in BCD, add the BCD numbers by adding the 4-bit groups in each column starting from the LSD.
- If there is no borrow out from the subtraction of any of the 4-bit groups, then no correction required.
- If there is borrow out from the subtraction of any of the 4-bit groups, subtract 0110 from the difference term of those groups (to skip illegal states).

BCD

Perform the following decimal subtractions in the 8421 BCD code.

(a) 38 – 15 (b) 206.7 – 147.8

(a)	38	⇒	0011	1000			(38 in BCD)
	–15		–0001	0101			(15 in BCD)
	23		0010	0011			(No borrow. So, this is the correct difference.)
(b)	206.7	⇒	0010	0000	0110	.0111	(206.7 in BCD)
	–147.8		–0001	0100	0111	.1000	(147.8 in BCD)
	58.9		0000	1011	1110	.1111	(Borrows are present, subtract 0110)
				–0110	–0110	–.0110	
				0101	1000	.1001	(Corrected difference = 58.9_{10})

EXCESS-3 CODE

The excess-3 codes are obtained as follows:

$$\text{Decimal Number} \longrightarrow \text{8421 BCD} \xrightarrow[\text{0011}]{\text{Add}} \text{Excess-3}$$

Example

Decimal	BCD				Excess-3			
	8	4	2	1	BCD + 0011			
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

ADDITION IN EXCESS-3

- To add in XS-3, add the XS-3 numbers by adding the 4-bit groups in each column starting from the LSD.
- If there is no carry out from the addition of any of the 4-bit groups, subtract 0011 from the sum term of those groups (because when two decimal digits are added in XS-3 and there is no carry, the result is in XS-6).
- If there is a carry out, add 0011 to the sum term of those groups (because when there is a carry, the invalid states are skipped and the result is in normal binary).

ADDITION IN EXCESS-3

Perform the following additions in XS-3 code.

(a)

37	⇒	0110	1010	(37 in XS-3)
+28		+0101	1011	(28 in XS-3)
65		1011	1 0101	(Carry generated)
		+1		(Propagate the carry)
		1100	0101	(Add 0011 to correct 0101 and
		−0011	+0011	subtract 0011 to correct 1100)
		1001	1000	(Corrected sum in XS-3 = 65_{10})

ADDITION IN EXCESS-3

Perform the following additions in XS-3 code.

(b)	247.6	⇒	0101	0111	1010	.1001	(247.6 in XS-3)
	+359.4		+0110	1000	1100	.0111	(359.4 in XS-3)
	607.0		1011	1111	1 0110	1 .0000	(Carry generated)
				+1	+1		(Propagate the carry)
			1011	1 0000	0111	.0000	
			+1				
			1100	0000	0111	.0000	(Add 0011 to 0000, 0111, 0000
			−0011	+0011	+0011	+.0011	and subtract 0011 from 1100)
			1001	0011	1010	.0011	(Corrected sum in XS-3 = 607.0_{10})

SUBTRACTION IN EXCESS-3

- To subtract in XS-3, subtract the XS-3 numbers by subtracting each 4-bit group of the subtractend from the corresponding 4-bit group of the minuend starting from the LSD.
- If there is no borrow from the next 4-bit group, add 0011 to the difference term of such groups (because when decimal digits are subtracted in XS-3 and there is no borrow, the result is in normal binary).
- If there is a borrow, subtract 0011 from the difference term.

SUBTRACTION IN EXCESS-3

Perform the following subtractions in XS-3 code.

(a) 267 – 175 (b) 57.6 – 27.8

Solution

(a)

267	⇒	0101	1001	1010	(267 in XS-3)
–175		–0100	1010	1000	(175 in XS-3)
092		0000	1111	0010	(Correct 0010 and 0000 by adding 0011and correct 1111 by subtracting 0011)
		+0011	–0011	+0011	
		0011	1100	0101	(Corrected difference in XS-3 = 92_{10})

(b)

57.6	⇒	1000	1010	.1001	(57.6 in XS-3)
–27.8		–0101	1010	.1011	(27.8 in XS-3)
29.8		0010	1111	.1110	(Correct 0010 by adding 0011 and correct 1110 and 1111 by subtracting 0011)
		+0011	–0011	–.0011	
		0101	1110	.1011	(Corrected difference in XS-3 = 29.8_{10})

SUBTRACTION IN EXCESS-3

Perform the following subtractions in XS-3 code using the 10's complement method.

(a) 597 − 239

Solution

(a) 10's complement of 239 = 761

XS-3 code of 239 = 0101 0110 1100

2's complement of 239 in XS-3 code = 1010 1001 0100

XS-3 code of 597 = 1000 1100 1010

597 − 239 = 358	597 + 761	(10's complement of 239)
	❶358	(Ignore the carry)
	358	(Corrected difference in decimal)

	1000	1100	1010	(597 in XS-3)
	+1010	1001	0100	(2's complement of 239 in XS-3)
	❶0010	❶0101	1110	(Propagate the carry)
+1		+1		
1	0011	0101	1110	(Ignore the carry)
	+0011	+0011	−0011	(Correct 1110 by subtracting 0011 and correct 0101 and 0011 by adding 0011)
	0110	1000	1011	(Corrected difference in XS-3 code = 358)

SUBTRACTION IN EXCESS-3

687	687	
−348	+651	(9's complement of 348)
339	1 338	
	+1	(End around carry)
	339	(Corrected difference in decimal)

	1001	1011	1010	(687 in XS-3)
	+1001	1000	0100	(1's complement of 348 in XS-3)
	1 0010	1 0011	1110	(Carry generated)
	-1	+1		(Propagate the carry)
1	0011	0011	1110	
			+1	(End around carry)
	0011	0011	1111	(Correct 1111 by subtracting 0011 and
	+0011	+0011	−0011	correct both groups of 0011 by adding 0011)
	0110	0110	1100	(Corrected difference in XS-3 = 339_{10})

GRAY CODE

- When you count up or down in binary, the number of bit that change with each digit change varies.
  - From 0 to 1 just have a single but
  - From 1 to 2 have 2 bits, a 1 to 0 transition and a 0 to 1 transition
  - From 7 to 8 have 3 bits changing back to 0 and 1 bit changing to a 1
- For some applications multiple bit changes can cause significant problems.

The decimal equivalent of gray code is as follows:

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

Reflection of Gray codes

Gray Code				Decimal	4-bit binary
1-bit	2-bit	3-bit	4-bit		
0	00	000	0000	0	0000
1	01	001	0001	1	0001
	11	011	0011	2	0010
	10	010	0010	3	0011
		110	0110	4	0100
		111	0111	5	0101
		101	0101	6	0110
		100	0100	7	0111
			1100	8	1000
			1101	9	1001
			1111	10	1010
			1110	11	1011
			1010	12	1100
			1011	13	1101
			1001	14	1110
			1000	15	1111

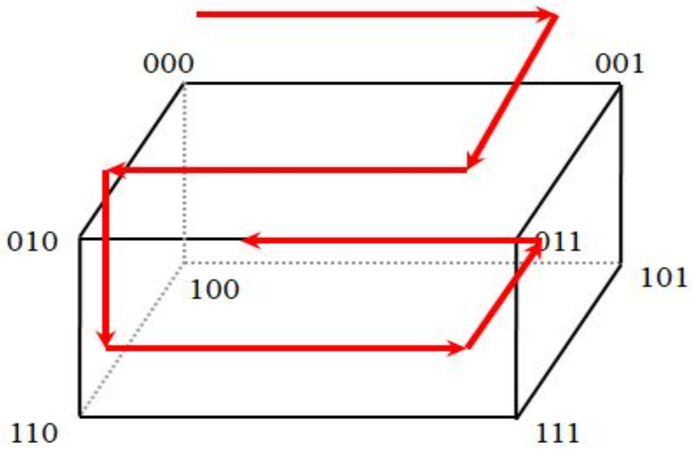
ADVANTAGE OF GRAY CODE

The advantage is that only bit in the code group changes in going from one number to the next.

1)Error detection

2)Representation of analog data

3)Low power design



GRAY CODE

BINARY TO GRAY

$G_n = B_n$

$G_{n-1} = B_n \oplus B_{n-1}$

$G_{n-2} = B_{n-1} \oplus B_{n-2}$

$\vdots$

$G_1 = B_2 \oplus B_1$

GRAY TO BINARY

$B_n = G_n$

$B_{n-1} = B_n \oplus G_{n-1}$

$B_{n-2} = B_{n-1} \oplus G_{n-2}$

$\vdots$

$B_1 = B_2 \oplus G_1$

EXAMPLES OF GRAY CODE

Find the gray code for the Binary number 1010

1 0 1 0

$G_4 = B_4$

$G_3 = B_4 \oplus B_3$

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$

So, $G_4 = 1$

$G_3 = 1 \oplus 0 = 1$

$G_2 = 0 \oplus 1 = 1$

$G_1 = 1 \oplus 0 = 1$

Gray code is 1111

EXAMPLES OF GRAY CODE TO BINARY

Find the Binary code for the Gray code is 1010

1 0 1 0

$B_4 = G_4$

$B_3 = B_4 \oplus G_3$

$B_2 = B_3 \oplus G_2$

$B_1 = B_2 \oplus G_1$

$B_4 = 1$

$B_3 = 1 \oplus 0 = 1$

$B_2 = 1 \oplus 1 = 0$

$B_1 = 0 \oplus 0 = 0$

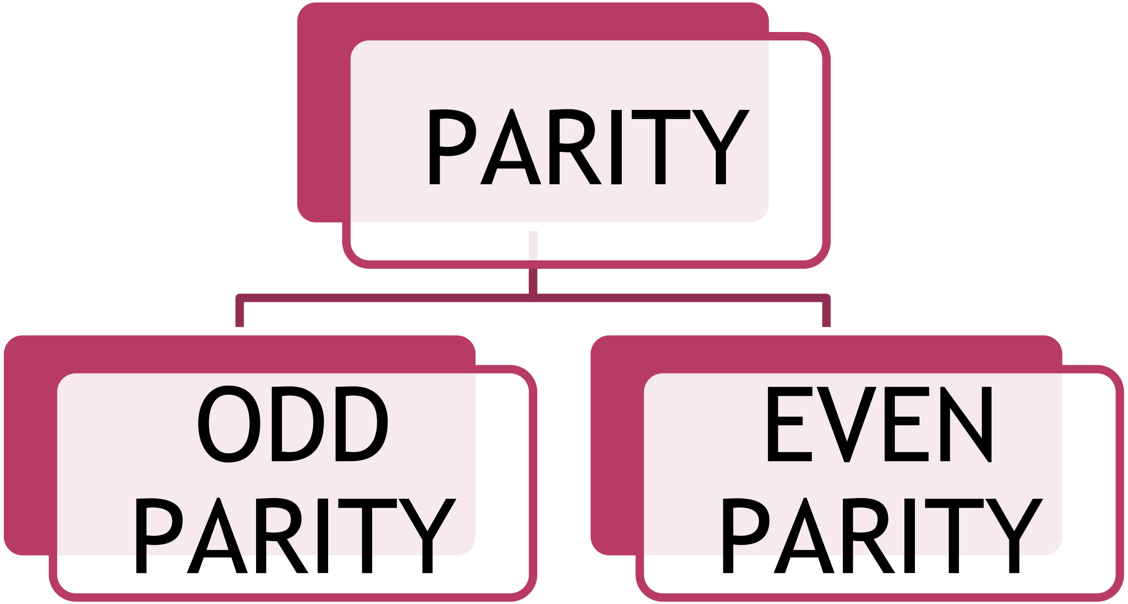
So, 1100

ERROR DETECTION

- When a binary data is transmitted and processed, it is susceptible to noise that can alter or distort its contents.
- The is may get changed to 0s and 0s to 1s. Because digital systems must be accurate to the digit, errors can pose a serious problem.
- Several schemes have been devised to detect the occurrence of a single-bit error in a binary word, so that whenever such an error occurs the concerned binary word can be corrected and retransmitted.

PARITY

The simplest technique for detecting errors is that of adding an extra bit, known as the parity bit, to each word being transmitted.



ODD PARITY

For odd parity, the parity bit 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.

EVEN PARITY

For even parity, the parity bit 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.

PARITY GENERATOR

- A parity generator is a combinational logic circuit that generates the parity bit in the transmitter.
- A combined circuit or devices of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data word.

PARITY GENERATOR

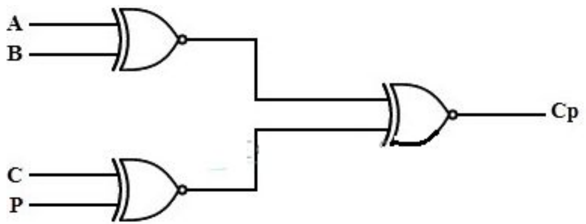
- The basic principle involved in the implementation of parity circuits is that sum of odd number of 1s is always 1 and sum of even number of 1s is always zero.

ODD PARITY

The number of 1-bit must add up to an odd number.

3-bit message			Odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

LOGIC DIAGRAM:

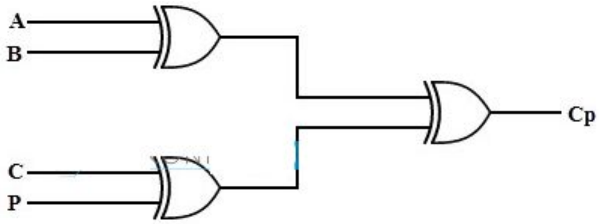


EVEN PARITY

The number of 1-bit must add up to an even number.

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

LOGIC DIAGRAM:



PARITY CHECKER

- Parity checking uses parity bits to check that data has been transmitted accurately.
- The parity bit is added to every data unit (typically seven or eight bits) that are transmitted.
- The parity bit for each unit is set so that all bytes have either an odd number or an even number of set bits.

PARITYGENERATOR/CHECKER APPLICATIONS

- One important application of the use of an Exclusive-OR gate is to generate parity.
- Parity is used to detect errors in transmitted data caused by noise or other disturbances.
- A parity bit is an extra bit that is added to a data word and can be either odd or even parity.
- In an even parity system, the sum of all the bits (including the parity bit) is an even number. In an odd parity system the sum of all the bits must be an odd number.

ERROR CORRECTION

- A code is said to be an error-correcting code, if the correct code word can always be deduced from an erroneous word.
- For a code to be a single-bit error-correcting code, the minimum distance of that code must be three.
- The minimum distance of a code is the smallest number of bits by which any two code words must differ.

7-BIT HAMMING CODE:

- To transmit four data bits, three parity bits located at positions 2^0, 2^1, and 2^2 are added to make a 7-bit code word which is then transmitted. The word format would be as shown below:
- $P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7$ where the D bits are the data bits and the P bits are the parity bits. P1 is set to a 0 or 1 so that it establishes even parity over bits 1, 3, 5, and 7 (P1 D3 D5 D7).
-

7-BIT HAMMING CODE:

•P2 is set to a O or a 1 to establish even parity over bits 2, 3, 6 and 7 (P2 D3 D6 D7). P4 is set to a 0 or a 1 to establish even parity over bits 4, 5, 6, and 7 (P4 D5 D6 D7).

For 7-bit code

C_3	C_2	C_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

7-BIT HAMMING CODE:

- Encode the data bits 1101 into 7 bit even parity Hamming code

$$P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7$$

$$\quad\quad 1 \quad\quad 1 \quad 0 \quad 1$$

For $P_1(1,3,5,7) = (P_1\ 111)$ must have even parity So $P_1 = 1$

For $P_2(2,3,6,7) = (P_2\ 101)$ must have even parity So $P_2 = 0$

For $P_4(4,5,6,7) = (P_4\ 101)$ must have even parity So $P_4 = 0$

Therefore the final code is 1010101

7-BIT HAMMING CODE:

- The message 1001001 coded in the 7-bit hamming code is transmitted through a noisy channel, Decode the message assuming the single error occurred in code word with even parity.

$$P_1 \; P_2 \; D_3 \; P_4 \; D_5 \; D_6 \; D_7$$

$$1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1$$

For (1,3,5,7) =(1001) has even parity So c_1 =0

For (2,3,6,7) =(0001) must have even parity So error c_2 =1

For (4,5,6,7) =(1001) has even parity So c_3 =0

Therefore the error word is $c_3\ c_2\ c_1$=010

7-BIT HAMMING CODE:

Therefore the error word is $c_3\ c_2\ c_1$=010. So complement 2^{nd} bit from the left. So the corrected code is

1101001

12-BIT HAMMING CODE:

- To transmit eight data bits, four parity bits located at positions 2^0, 2^1, 2^2 and 2^3 are added to make a 12-bit code word which is then transmitted. The word format would be as shown below:
- $P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7\ P_8\ D_9\ D_{10}\ D_{11}\ D_{12}$ where the D bits are the data bits and the P bits are the parity bits. P1 is set to a 0 or 1 so that it establishes even parity over bits 1, 3, 5, 7,9,11 (P1 D3 D5 D7 D9 D11).

12-BIT HAMMING CODE:

- P2 is set to a O or a 1 to establish even parity over bits 2, 3, 6,7,10 and 11 (P2 D3 D6 D7 D10 D11).
- P4 is set to a 0 or a 1 to establish even parity over bits 4, 5, 6, 7 and 12 (P4 D5 D6 D7 D12).
- P8 is set to a 0 or a 1 to establish even parity over bits 8,9,10,11 and 12 (P8 D9 D10 D11 D12).

12-BIT HAMMING CODE:

For 12-bit code

C_4	C_3	C_2	C_1
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

12-BIT HAMMING CODE:

- Encode the data bits 01011011 into 12 bit even parity Hamming code

$$P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7\ P_8\ D_9\ D_{10}\ D_{11}\ D_{12}$$

0 1 0 1 1 0 1 1

For P_1(1, 3, 5, 7,9,11) =($P_1$01111) has even parity So P_1 =0

For P_2(2, 3, 6,7,10 11) =(P_2 00101) must have even parity So P_2 =0

12-BIT HAMMING CODE:

- Encode the data bits 01011011 into 12 bit even parity Hamming code

For P_4(4, 5, 6,7,12) =(P_4 1011) must have even parity So P_4 =1

For P_8(8,9,10,11 and 12) =($P_8$1011) must have even parity So P_8 =1

Therefore the final code is 000110111 0111

15-BIT HAMMING CODE:

- To transmit eleven data bits, four parity bits located at positions 2^0, 2^1, 2^2 and 2^3 are added to make a 15-bit code word which is then transmitted. The word format would be as shown below:
- $P_1\ P_2\ D_3\ P_4\ D_5\ D_6\ D_7\ P_8\ D_9\ D_{10}\ D_{11}\ D_{12}\ D_{13}\ D_{14}\ D_{15}$ where the D bits are the data bits and the P bits are the parity bits. P1 is set to a 0 or 1 so that it establishes even parity over bits 1, 3, 5, 7,9,11,13,15 (P1 D3 D5 D7 D9 D11 D13 D15).

15-BIT HAMMING CODE:

- P2 is set to a O or a 1 to establish even parity over bits 2, 3, 6,7,10,11, 14,15 (P2 D3 D6 D7 D10 D11 D14 D15).
- P4 is set to a 0 or a 1 to establish even parity over bits 4, 5, 6, 7,12, 13,14,15 (P4 D5 D6 D7 D12 D13 D14 D15).
- P8 is set to a 0 or a 1 to establish even parity over bits 8,9,10,11,12,13,14,15 (P8 D9 D10 D11 D12 D13 D14 D15).

15-BIT HAMMING CODE:

For 15-bit code

C_4	C_3	C_2	C_1
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

15-BIT HAMMING CODE:

- The message 101001011101011 coded in the 15-bit hamming code is transmitted through a noisy channel, Decode the message assuming the single error occurred in code word with even parity.

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
1	0	1	0	0	1	0	1	1	1	0	1	0	1	1

For (1,3,5,7,9,11,13,15) =(11001001) has even parity So c_1 =0

For (2,3,6,7,10,11,14,15) =(01101011) must have even parity So error c_2 =1

For (4,5,6,7,12,13,14,15) =(00101011) has even parity So c_3 =0

15-BIT HAMMING CODE:

For (8,9,10,11,12,13,14,15) =(11101011) has even parity So c_4 =0

Therefore the error word is $c_4 c_3\ c_2\ c_1$=0010. So complement 2nd bit from the left. So the corrected code is

111001011101011

ASCII

- It is an acronym for the **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
- It is a standard seven-bit code that was first proposed by the American National Standards Institute or ANSI in 1963, and finalized in 1968 as ANSI Standard X3.4.
- The purpose of ASCII was to provide a standard to code various symbols (visible and invisible symbols)
- In the **ASCII character set**, each binary value between 0 and 127 represents a specific character.
- Most computers extend the ASCII character set to use the full range of 256 characters available in a byte. The upper 128 characters handle special things like accented characters from common foreign languages.

- ASCII is a character encoding scheme that encodes 128 different characters into 7 bit integers
- Computers can only read numbers, so ASCII is a numerical representation of special characters
    - Ex: ‘%’ ‘!’ ‘?’

ASCII has some interesting properties:

- Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}
- Upper case A-Z span 41_{16} to $5A_{16}$
- Lower case a-z span 61_{16} to $7A_{16}$

ASCII TABLE

LSBs \ MSBs	000	001	010	011	100	101	110	111
0000	NUL	DEL	Space	0	@	P	p	
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DLE

ASCII

Control characters

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

Refernces

1. http://www.trytoprogram.com/cpp-examples/cplusplus-binary-decimal-program/

2. https://javacodekorner.blogspot.com/2018/12/decimal-to-octal-in-java.html

3. "Switching Theory and Logic Design" by A. Anand Kumar

THANK YOU