philosophy of Number Systems

— Number system is a basis for counting various items. We are familiar with decimal number system with its 10 digits: 0, 1, 2, 3, 4, . . . , 9.

- But modern computers communicate and operate with binary numbers which use only 0 and 1.

Ex: 18 → 10010.

In decimal it is represented by two digits where as in binary 5 digits.

∴ If decimal quantities are represented in binary form they take more digits.

— for large decimal number people have to deal with very large binary strings and therefore, they do not like working with binary numbers: This fact give rise to there new number systems.

octal, Hexadecimal and Binary coded decimal.

— These number systems represent binary number in a compressed form.

∴ these number systems are now widely used to compress long strings of binary.

Decimal number systems.

- In this we can express any number in units, hundreds, thousand and so on.

Ex: 5678.9

: 5000 + 600 + 70 + 8 + 0.9 = 5678.9

can also be written as $(5678.9)_{10}$ where 10 subscript indicates the radix or base.

- In power of 10 we can write

$5 \times 10^3 + 6 \times 10^2 + 7 \times 10^1 + 8 \times 10^0 + 9 \times 10^{-1}$

- The left most bit which has the greatest weight is called most Significant bit (MSB)
- The Right most bit which has the least weight is called least Significant bit (LSB).

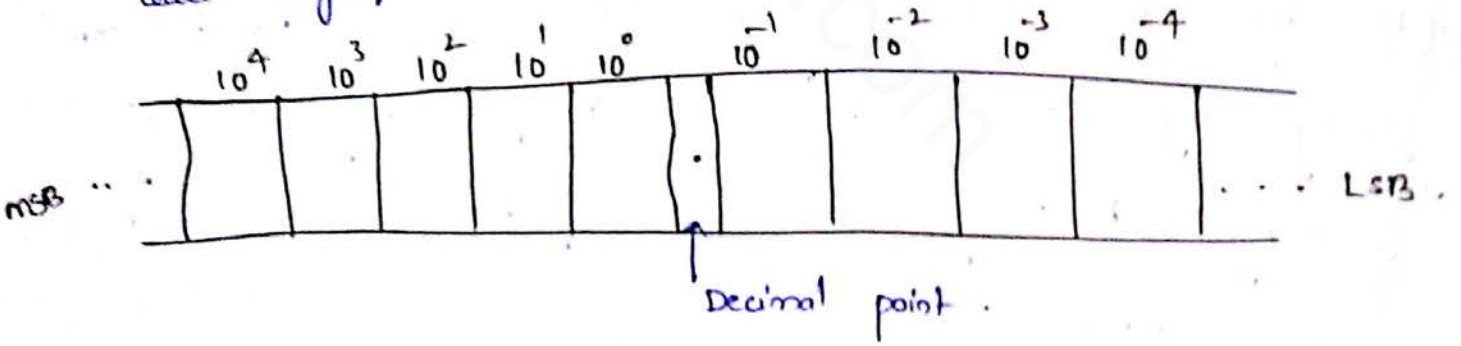


Fig: Decimal position values as powers of 10.

Ex: Represent decimal number 98.72 in power of 10

$9 \times 10^1 + 8 \times 10^0 + 7 \times 10^{-1} + 2 \times 10^{-2}$

for 9 power of 10 is 1 for 2 power of 1/100 is -2

for 8 power of 10 is 0 for 7 power of 1/10 is -1

Binary Number System :—

Binary system with its two digits is a base - two system. The two binary digits are 1 and 0. Each binary digits are known as bit.

- In binary system weight is expressed as power of 2.

Ex: Represent binary number 1101.101 in power of 2 and find its decimal Equivalent

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

$$= 8 + 4 + 0 + 1 + \tfrac{1}{2} + 0 + \tfrac{1}{8}$$

$$= (13.625)_{10}$$

1.3 Octal Number System :—

It uses first eight digits of decimal number System 0, 1, 2, 3, 4, 5, 6, and 7. As it uses 8 digits, its base is 8

Ex: represent octal number 567 in power of 8 and find its decimal Equivalent

$$5\times 8^2 + 6\times 8^1 + 7\times 8^0 = 5\times 64 + 6\times 8 + 7\times 1 = (375)_{10}$$

1.4 Hexadecimal Number System :—

— Its base is 16, it is having 16 digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Since its base is a power of 2, (2^4) it is Easy to convert hexadecimal to binary numbers and vice versa.

It is useful for human communications with a computer

— Each heradecimal digit represents a group of four binary digits called nibbles.

Relation b/n decimal, binary, Hexadecimal, octal

Decimal	Binary	Hexadecimal	Octal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Represent hexadecimal number 3FD in power of 16 and find its decimal Equivalent

$3 \times 16^2 + F \times 16^1 + D \times 16^0$

3 F D

$3 \times 256 + 15 \times 16 + 13 \times 1$

$= 768 + 240 + 13 = (1021)_{10}$.

5. Counting in Radix (Base) :—

— In general, a number represented in radix r, has r characters in its set and x can be any value. They are $0, 1, \ldots\ldots r-1$.

Radix	characters in set
2	0, 1
3	0, 1, 2
4	0, 1, 2, 3
⋮	
7	0, 1, 2, 3, 4, 5, 6,
8 (octal)	0, 1, 2, 3, 4, 5, 6, 7
⋮	⋮
(10) (Decimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
⋮	
(16) (Hexadecimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Ex: find decimal Equivalent of $(231.23)_4$

$= 2\times4^2 + 3\times4^1 + 1\times4^0 + 2\times4^{-1} + 3\times4^{-2}$

$= 32 + 12 + 1 + 0.5 + 0.1875$

$= (45.6875)_{10}$

Ex: Count from 0 to 9 in Radix 5

Sol Radix 5 has 5 characters.

0, 1, 2, 3, 4, 10, 11, 12, 13, 14

5 in radix $(10)_5$ —— $(5)_{10}$

$1\times5^1 + 0\times5^0 = 5+0 = 5_{10}$

$(11)_5$ —— $(6)_{10}$

$(11)_5 = 1\times5^1 + 1\times5^0 = 5+1 = 6_{10}$

$(7)_{10}$ — $(12)_5 \Rightarrow 1\times5^1 + 2\times5^0 = 7_{10}$

$(8)_{10}$ — $(13)5 \Rightarrow 1\times5^1 + 3\times5^0 = 8_{10}$

$(9)_{10}$ — $(14)5 \Rightarrow 1\times5^1 + 4\times5^0 = 9_{10}$

Ex: Represent $(13)_{10}$ in octal

$(13)_{10}$ —— $(?)_8$

$8 \mid 13$

$1 - 5$ ↑ $= (15)_8$

check $(15)_8$ —— $(?)_{10}$

$1\times8^1 + 5\times8^0$

$= 8+5 = (13)_{10}$

Number System Conversion

Binary to octal :—

— for octal Numbers base is 8 and the base for binary is 2.

i.e the base for octal number is the third power of base for binary numbers

∴ By grouping 3 digits of 3 digits of binary numbers and then converting Each group digits to its octal Equivalent

Ex: octal to Binary Convert $(111\ 101\ 100)_2$ to octal

— $(7\ 5\ 4)_8$.

2.2 Octal to Binary

— Each digit of the octal number is individually converted to its binary Equivalent to get octal to binary conversion of the number

Ex: $(634)_8$ — $(?)_2$

$= (110\ 011\ 100)_2$

Ex: $(725.63)_8$ — $(?)_2$

$(111010101.110011)_2$

2.3 Binary to Hexadecimal

— Base for hexadecimal is 16 and the base for binary is 2.

The base for hexadecimal number is the fourth power of the base for binary numbers

∴ By grouping 4 digits of binary number starting from right side and replacing each group digit to its hexadecimal Equivalent.

we can convert binary number to its hexadecimal Equivalent

Ex: Convert $(1101\ 1000\ 1001\ 1011)_2$ to hexadecimal Equivalent

$(D\ 8\ 9\ B)_H$

2.4 Hexadecimal to Binary Conversion:

— Each digit of the hexadecimal number is individually converted to its binary Equivalent to get hexadecimal to binary conversion of the number

Ex: Convert $(5A9.B4)_H$ to binary

$(0101\ 1010\ 1001 . 1011\ 0100)_2$

Ex: Convert $(3FD)_H$ to binary

$(0011\ 1111\ 1101)_2$.

2.5 octal to hexadecimal:

1. Convert octal to binary number
2. Convert binary number to its hexadecimal Equivalent

Ex: $(615)_8$ —— $(\ )_{16}$

(i) $(110\ 001\ 101)_2 \rightarrow (18D) = (18D)_H$

1.6 Hexadecimal to octal Conversion

(i) Convert hexadecimal number to its binary Equivalent

(ii) Convert binary number to its octal Equivalent

Ex: $(25B)_H$ —— $(?)_8$

$(0010\ \ 0101\ \ 1011)_2$

001 001 011 011

1 1 3 3 = $(1133)_8$

Converting any Radix to Decimal

— In general numbers can be represented as

$$N = A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \dots + A_1r^1 + A_0r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \dots + A_{-m}r^{-m}$$

N = Number in Decimal

A = Digit

r = Radix

n = The no. of digits in the integer portion of the number

m = The no. of digits in the fractional portion of number

Ex: $(1101.1)_2$ —— $(?)$

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

$= 8 + 4 + 0 + 1 + 0.5$

$= (13.5)_{10}$

Ex: $(475.85)_8$ ——— $(?)_{10}$

$N = 4\times8^2 + 7\times8^1 + 5\times8^0 + 8\times8^{-1} + 2\times8^{-2}$

$= 256 + 56 + 5 + 0.25 + 0.078125$

$= (317.328125)_{10}$

Ex: $(3102.12)_4$ ——— $(?)_{10}$

$N = 3\times4^3 + 1\times4^2 + 0\times4^1 + 2\times4^0 + 1\times4^{-1} + 2\times4^{-2}$

$= 192 + 16 + 0 + 2 + 0.25 + 0.125$

$= (210.375)_{10}$

Ex: $(614.15)_7$ ——— $(?)_{10}$

$= 6\times7^2 + 1\times7^1 + 4\times7^0 + 1\times7^{-1} + 5\times7^{-2}$

$= 294 + 7 + 4 + 0.142857 + 0.102$

$= (305.244658)_{10}$

Conversion of Decimal Numbers to any Radix Number

Two Steps:

Step 1: Convert integer part. — This is done by Successive division method.

Step 2: Convert fractional part — This is done by Successive multiplication method.

(i) Successive Division for integer part:—

Ex: $(37)_{10}$ ——— $(?)_2$

2 | 37
2 | 18 – 1
2 | 9 – 0
2 | 4 – 1
2 | 2 – 0
1 – 0 ↑ $= (100101)_2$ $32 + 4 + 1 = 37$

Ex: $(3509)_{10}$ ——— $(\ )_{16} - (\ )_H$

16 | 3509
16 | 219 – 5
13 – 11 ↑ $= (DB5)_{16}$

Ex: $(54)_{10}$ — $(?)_4$

4 | 54
4 | 13 – 2
3 – 1 ↑ $= (312)_4$

(ii) Successive multiplication for fractional part:

— The number to be converted is multiplied by radix of new number, producing a product that an integer part and a fractional part.

— The integer part of the product becomes a numberal in the new radix

- The fractional part is again multiplied by the new radix, & the process is repeated until fractional part reaches 0 or u. the new radix number is carried out to sufficient digits. The integer part of each product is read downward to repres the new radix number.

Ex: $(0.8125)_{10}$ ——— $(?)_2$

Fraction	Radix	Result	Recorded carry.	
0.8125	x 2	= 1.625 = 0.625	1	MSB
0.625	x 2	= 1.25 = 0.25	1	
0.25	x 2	= 0.5 = 0.5	0	
0.5	x 2	= 1.0 = 0.0	1	LSB

$(0.1101)_2$

$(0.1101)_2$ — $(0.8125)_{10}$

Ex: $(0.95)_{10}$ —— $(?)_2$.

0.95 x 2 = 1.9 = 0.9	1	— 0.8 is repeated and if we multiply further, we will get repeated sequence. If we stop here, we get 7 binary digits (0.1111001)
0.9 x 2 = 1.8 = 0.8	1	
0.8 x 2 = 1.6 = 0.6	1	
0.6 x 2 = 1.2 = 0.2	1	
0.2 x 2 = 0.4 = 0.4	0	
0.4 x 2 = 0.8 = 0.8	0	
0.8 x 2 = 1.6 = 0.6	1	

Ex: $(0.1289062)_{10}$ —— $(?)_{16}$

0.1289062×16	$= 2.0625$	2
0.0625×16	$= 1.0$	1

$$(0.21)_{16}$$

Ex: Convert $(24.6)_{10}$ — $(?)_2$

Step1: Separate out integer and fraction part.

Integer part : 24, fraction part : 0.6

Step2: find Equivalent binary number for integer part.

2 | 24		
2 | 12	—	0
2 | 6	—	0
2 | 3	—	0
	1	— 1

$(11000)_2$

Step3: find Equivalent binary number for fractional

0.6×2	$= 1.2$	0.2	1
0.2×2	$= 0.4$	0.4	0
0.4×2	$= 0.8$	0.8	0
0.8×2	$= 1.6$	0.6	1
0.6×2	$= 1.2$	0.2	1

$(0.10011)_2$.

$\therefore (11000.10011)_2$

Questions

1) Convert $(725.25)_8$ to dec, bin, hex dec.

Sol (i) $= 7\times8^2 + 2\times8^1 + 5\times8^0 + 2\times8^{-1} + 5\times8^{-2}$

$= 448 + 16\times5 + \frac{1}{4} + \frac{5}{64}$

$= 469 + 0.25 + 0.078125$

$= (469.328125)_{10}$

(ii) $(725.25)_{10}$ — ( )$_2$

$(111010101.010101)_2$

(iii) 000 1 1010 1010 1 . 0101 0100

$(1\ D\ 5\ 54)_{16}$

Weighted Codes:

In this Each digit position of the number represents a specific weight.

Ex: If number is 567 then wt of 5 is 100
wt of 6 is 10
wt of 7 is 1

Ex: Binary, BCD

- BCD is an abbreviation for binary coded Decimal. BCD is a numeric code in which Each digit of a decimal number is represented by a separate group of bits.

Complement Representation of negative numbers:

In digital Computers, to simply the Subtraction operation and for logical manipulation Complements are used.

There are two types of Complements for each radix System:

The radix complement and diminished radix complement.

The first is referred to as the r's complement and Second as the (r-1)'s complement.

for Example, in binary System we Substitute base value 2 in place of r to refer Complements as 2's complement and 1's complement. In decimal number system, we substitute base value 10 in place of r to refer complements as 10's complement and 9's complement.

1's complement Representation:

The In 1's complement Representation, the number changes all 1's to zero's and all zero's to 1's.

Find 1's complement of $(1101)_2$

1101 ← number.
0010 ← 1's complement.

find 1's Complement of 10111001

10111001 ← number
01000110 ← 1's complement.

2's complement Representation.

The 2's complement is the binary number that results when we add 1 to the 1's complement. It is given as

2's complement = 1's complement + 1

The 2's complement form is used to represent negative numbers.

Find 2's complement of $(1001)_2$

```
  1 0 0 1   ← number
  0 1 1 0   ← 1's complement
+       1
-----------
  0 1 1 1   - 2's complement
```

Find 2's complement of $(1010001)_2$

```
  1 0 1 0 0 0 1   ← number
  0 1 0 1 1 1 0   — 1's complement
              + 1
-----------------
  0 1 0 1 1 1 1
-----------------
```

1's complement Subtraction

Subtraction of binary numbers can be accomplished by the direct method by using the 1's complement method, which allows to perform subtraction using only addition.

For subtraction of two numbers we have two cases.

- Subtraction of Smaller number from larger number and
- Subtraction of larger number from smaller number

Subtraction of Smaller number from larger number.

method:

1. Determine the 1's Complement of the Smaller number.
2. Add the 1's Complement to the larger number.
3. Remove the Carry and add 1 to the Result.

1. Subtract $(101011)_2$ from $(111001)_2$ using the 1's complement.

```
  1 1 1 0 0 1
  0 1 0 1 0 0   == 1's Complement
  ---------------
1 0 0 1 1 0 1
  ---------->+1
  ---------------
  0 0 1 1 1 0   => final answer
  ---------------
```

57
43
14

Subtraction of larger number from Smaller number.

method:

1. Determine 1's Complement of larger number.
2. Add the 1's Complement to the Smaller number.
3. Answer is in 1's Complement form. To get the answer in true form take the 1's Complement and assign negative sign to the answer.

Subtract $(111001)_2$ from $(101011)_2$ using the 1's complement method.

```
  1 0 1 0 1 1
- 0 0 0 1 1 0   = 1's Complement of 1 1 1 0 0 1
  ---------------
  1 1 0 0 0 1   = Answer in 1's complement form.
  ---------------
- 0 0 1 1 1 0   = Answer in true form.
```

2's Complement Subtraction

In 2's complement subtraction, the subtraction is accomplished by only addition. Let us see the methods for 2's complement subtraction.

⇒ Subtraction of smaller number from larger number.

method:

1. Determine the 2's complement of a smaller number.
2. Add the 2's complement to the larger number.
3. Discard the carry.

Subtract $(101011)_2$ from $(111001)_2$ using the 2's complement method.

```
  1 1 1 0 0 1                    0 1 0 1 0 0
2's comp→ 0 1 0 1 0 1 .                  +1
  ---------------------          -----------
  (1) 0 0 1 1 1 0                0 1 0 1 0 1
   ↓                             -----------
 Discard
```

0 0 1 1 1 0 — final answer.

Subtraction of larger number from smaller number:

method:

1. Determine the 2's complement of larger number.
2. Add the 2's complement to the smaller number.
3. Answer is in the 2's complement form. To get the answer in the true form take the 2's complement and assign negative sign to the answer.

Subtract $(111001)_2$ from $(101011)_2$ using 2's complement method

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

```
  0 0 0 1 1 0
      + 1
  -----------
  0 0 0 1 1 1   - 2's complement.
```

↳ Answer in 2's complement form

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

Decimal	Signed magnitude	1's complement	2's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001

Subtract $(111001)_2$ from $(101011)_2$ using 2's complement method.

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

```
  0 0 0 1 1 0
      + 1
  -----------
  0 0 0 1 1 1   - 2's complement.
```

↳ Answer in 2's complement form

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

Decimal	Signed magnitude	1's complement	2's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001

230 − 120

2	230
2	115 - 0
2	57 - 1
2	28 - 1
2	14 - 0
2	7 - 0
2	3 - 1
	1 - 1

2	120
2	60 - 0
2	30 - 0
2	15 - 0
2	7 - 1
2	3 - 1
	1 - 1

11100110 −

01111000

10000111

+ 1

10001000

1 1 1 0 0 1 1 0

1 0 0 0 1 0 0 0

(1) 0 1 1 0 1 1 1 0

$2^6+2^5+2^3+2^2+2^1$

$64+32+8+4+2$

110

Signed Binary Numbers

Signed Binary Numbers mainly used for represent Sign of the number.

However, because of hardware limitations, in computers both positive and negative numbers are represented with only binary digits.

The left most bit (sign bit) in the number represents Sign of the number.

The Sign bit is 0 positive numbers

The Sign bit is 1 negative numbers.

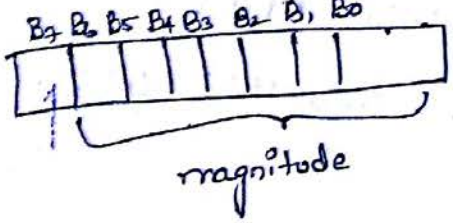
These numbers are represented by the Signed magnitude format.

Below fig shows the Sign magnitude format for 8-bit Signed number.

Here, the most Significant bit (MSB) represents Sign of the number.

If MSB is 1, number is negative and if MSB is 0, number is +ve, The remaining bits represent magnitude of the numbers.

Here Some Examples for Sign-magnitude numbers.



+6

2 | 6
2 | 3 - 0
1 - 1

00000110

-14

2 | 14
2 | 7 - 0
2 | 3 - 1
1 - 1 1

10001110

+24 0001 1000
-64 1100 0000

In case of unsigned 8-bit binary numbers the decimal range is 0 to 255..

For Signed magnitude 8-bit binary numbers the largest magnitude is Reduced from 255 to 127.

because we need to Represent both positive and negative numbers.

maximum +ve number 0111 1111 = +127
maximum -ve number 1111 1111 = -128

We have Seen +ve and -ve number Representation in the Signed magnitude format.

This is the only way to represent +ve numbers!

however there are two more ways represent negative numbers:

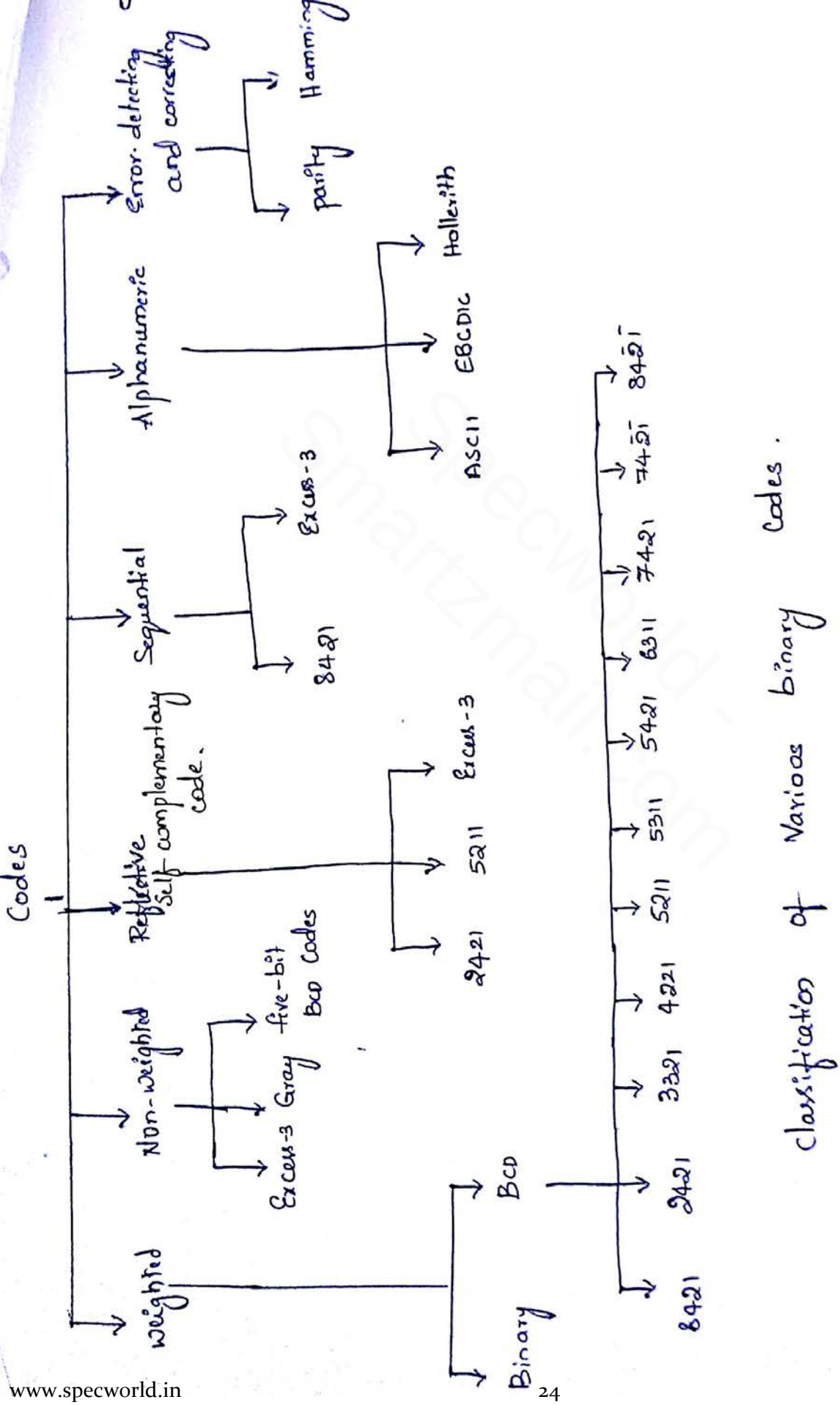
Signed -1's Complement representation.
Signed 2's complement Representation

Let us see how -6 represents in three formats:

Signed - magnitude representation 10000110
Signed 1's complement representation 11111001
Signed 2's complement Representation 11111010

Binary codes :-

The digital data is transmitted, stored and represented as groups of binary digits (bits). The group of bits, also known as binary code,



Alphanumeric codes : The code which consists of both alphabets and numbers.

ASCII Code:- It is a 7-bit code which includes 0-9 - bit code upper case alphabets, lower case alphabets, and other symbols. It is preceded by 011 for decimal no.

0 → 011 0000
9 → 011 1001

A — 65 → 100 0001
Z — 90 → 10 11010
a — 97 → 110 0001
z — 122 — 111 1010.

2	90	
2	45	— 0
2	22	— 1
2	11	— 0
2	5	— 1
2	2	— 1
	1	— 0

EBCDIC:— It is a 8-bit code. It is preceded by 1111 for decimal digits

a — 129
z — 154.

Decimal Digit	BCD Code 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 0 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

— In multidigit coding, each decimal digit is individual coded with 8421 BCD code

Ex: $(58)_{10} \xrightarrow{8421}$ 0101 1000.

— If we want to represent same 58 in binary the equivalent is $(111010)_2$ i.e we require only 6 digits that means 8421 BCD is less efficient than pure binary number.

— Advantage: If is Easy to convert between it and decimal.
— Disadvantage: Low efficiency, the arithmetic operations are more Complex than they are in pure binary.

2421 code:

Decimal Digit	2421
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

2: Non weighted Codes: These are not assigned with any weight to Each digit position. Ex:

Excess -3 Code:

It is a modified form of a BCD number. It can be derived from the natural BCD Code by adding 3 to Each Coded number

Ex: 12 in BCD as 0001 0010

Now adding 3 to Each digit we get Excess-3 code as

0001 0010
+ 11 + 11
0100 0101 — Excess-3 code for $(12)_{10}$

BCD Addition:

The addition of two BCD numbers can be best understood by considering the three cases that occur when two BCD digits are added.

Sum Equal to 9 or less with Carry 0.

let us Consider additions of 3 and 6 in BCD.

```
6      0110  ← BCD for 6.
+3     0011  ← BCD for 3
-----  ----
9      1001  ← BCD for 9.
```

```
  9
+ 3
----
 12
```

Sum greater than 9 with Carry 0.

```
6      0110
8      1000
----   ----
14     1110  ← invalid BCD (1110) > 9.
```

Whenever invalid BCD no. occurs, the Sum has to be Corrected by the addition of Six (0110) in the in valid BCD no. as shown below.

```
6    0110
8    1000
-----------
14   1110  — invalid BCD no.
     0110
-----------
0001 0100  ← BCD for 14.
-----------
 1    4
```

Sum Equals 9 or less with carry 1 .

let us consider addition of 8 and 9 in BCD.

```
 8        1000
 9        1001
---      ------
17      0001 1001   — incorrect BCD result.
        ---------
        0001 0001   — add 6 for correction
           0110
        ---------
        0001 0111
        ---------
         1    7    — BCD for 17.
```

```
24      0010    0100
18      0001    1000
        ----    ----
        0011    1100   invalid BCD no.
                       1100>9.
                0110   add 6 for correction.
        ---------------
        0100    0010   — propagate carry to
        ---------------      next higher digit.
```

```
589     0101    1000    1001
199     0001    1001    1001
---     ---------------------
788      11      11      1
        ---------------------
        0111    0010    0010
                0110    0110
                 11      11
        ---------------------
        0111    1000    1000   — corrected Sum.
        ---------------------
```

79 − 26 = 53

```
0111  1001          99
0111  0011 = 73-9's 26
                    73
1110  1100  for BCD 26.
```

0110 — 1100 > 9, add 6 to result.

```
1111  0010          1111 > 9.
0110
1 1
10101 0010
  └──→ 1
01 01 0011          BCD for 53.
```

89 − 54 = 45

```
99
54
1000  1001
0100  0101
1100  1110          1110 > 9 So add 6.
      0110
1 1 1
1101  0100          1101 > 9 add 6.
0110
1 1
10011 0100
  └──→ 1
0011  0101          BCD for 35.
```

175	0001	0111	0101	
326	0011	0010	0110	
501				
	0100	1001	1011	1011 > 9.
			0110	add 6 to 1011 for correction.
		1 1		
	0100	1010	0001	
		0110		1010 > 9. add 6.
	0101	0000	0001	propagate carry to next higher digit.

BCD Subtraction:

Addition of signed BCD numbers can be performed by using 9's or 10's complement methods.

A negative BCD number can be expressed by taking the 9's or 10's complement.

8	9	8	8045	456
-2.	2	+7	-9161	-436
	7.	15		
		L→1		
		6		

7 9 − 2 6. perform the BCD subtraction using 9's complement method.

BCD Subtraction using 10's complement.

Regular Subtraction

```
  8
- 2
---
  6
```

10's complement Subtraction.

```
  9
  8
---
  7
+ 1
---
  8
```

```
  8
  8
---
 16.
```

```
  8        1000
+ 8        1000
---      ------
 16      10000
           0110
         ------
           0110   -6.
         ------
```

- find the 10's complement of negative numbers.
- Add two numbers using BCD addition
- if carry is not generated find the 10's complement of the result otherwise find the 9's complement of the result.

→

```
  79
- 26
---
  53
```

```
  99
  26
---
  73
+ 1
---
```

```
  79
+ 72
---
```

```
  0111  1001
  0111  0100
------------
1 1 0 1 1 1 0 1     1101 > 9.
        0 1 1 0
------------------
  1 1 1 1  0 0 1 1     1111 > 9
      0 1 1 0
    ------------
discard (1) 0 1 0 1  0 0 1 1
              5      3
```

Excess-3 :- It is a modified form of BCD. & add 3 in this code add 3 to BCD of each digit.

	BCD		Excess-3	
12 —	0001	0010	0100	0101

It is a Self Complementing code.

Excess-3 addition:

Ex:- $(592)_{10}$ —	1000	1100	0101	
9's comp(592)	0111	0011	1010	

Ex: BCD

98 1001 0000

Excess 1100 1011

0011 0100

99

98

01

Excess-3 addition

To perform Excess-3 addition we have to

- Add two Excess-3 numbers.
- if carry = 1 → add 3 to the Sum of two digits
  = 0 → Subtract 3.

```
  8     1011
+ 6     1001
 14
    00010100
     00110011
    ---------
     0100 0111
    ---------
      1.   4
```

```
1   0100 — Excess-3
+2  0101 — Excess-3
 3  1001
Sub 3-0011
    0110 — Excess-3 for 3.
```

Decimal Digit	Excess-3 code
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

— In Excess-3 code, we get 9's complement of a number by just complementing each bit. Due to this Excess-3 code is called self-complementing code.

Ex: 592 in Excess-3 1000 1100 0101 9's complement of 592 is 999

592

407

407 in Excess-3 0111 0011 1010

i.e nothing but just complementing the 592 in Excess-3 you will get 9's complement of 592.

Ex: 403 in Excess-3 0111 0011 0110

9's complement of 403 is 1000 1100 1001

999
- 403

596

596 in Excess-3 is 1000 1100 1001

Gray Code :

It is a special case of unit - distance code. In this bit patterns for two consecutive numbers differ in only one bit position. It is also called as <u>cyclic code</u>.

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

- For gray code any two adjacent code groups differ only in one bit position. It is also called as reflected code, because two least significant bits for 4 through 7 are the mirror images of those for 0 to 3.
- Similarly, the three LSB for 8 to 15 are the mirror images of are from 0 to 7.

Binary to Gray conversion.

Let us represent binary numbers as $B_1, B_2, B_3, B_4, \ldots B_n$ and its Equivalent gray code as $G_1, G_2, G_3 \ldots G_n$

$$G_1 = B_1$$
$$G_2 = B_1 \oplus B_2$$
$$G_3 = B_2 \oplus B_3$$
$$G_4 = B_3 \oplus B_4$$
$$\vdots$$
$$G_n = B_{n-1} \oplus B_n$$

Ex: 1) $(10110110)_2 \xrightarrow{\text{Gray}} (11101101)$

2) $(1010101011 0)_2 \xrightarrow{\text{Gray}} (1111111101)$

3) $(110010)_2 \xrightarrow{\text{Gray}} (101011)$

Gray to Binary Conversion

Steps

1. The MSB of the binary number is the same as the MSB of Gray code number.
2. To obtain the next binary digit, perform an Ex-OR between the bit just written down and the next gray code bit. Write down the result.
3. Repeat steps 2 until all gray code bits have been Ex-ORed with binary digits. The sequence of bits that have been written down is the binary equivalent of gray code.

Ex: 1) Gray code: 1 0 1 0 1 1

1 1 0 0 1 0 (Binary)

2) Gray code: 1 1 1 1 1 1 1 1 0 1

$(1\,0\,1\,0\,1\,0\,1\,0\,1\,1\,0)_2$

3) Gray code 1 1 1 0 1 1 0 1

$(1\,0\,1\,1\,0\,1\,1\,0)_2$

⇒ Alphanumeric codes: It includes numbers, letters, and other symbols.

ASCII (American standard code for information interchange) 011

EBCDIC (Extended binary coded decimal Interchange code) 1111

Error Detecting and Correcting Codes

- When the digital information in the binary form is transmitted from one circuit to another an Error may occur.
- To maintain data integrity between transmitter and receiver Extra bits are added in the data, these allow the detection and Sometimes Correction of Errors.

parity Bits

- It is used for the purpose of detecting Errors during transmission.
- A parity bit is an Extra bit included with a binary message to make the number of 1s either odd or Even.
- The message, including the parity bit is transmitted and then checked at the receiving end for Errors.
- An Error is detected if the checked parity does not correspond with the one transmitted.
- The ckt that generates the parity bit in the Tx is called as parity Generator.
- The ckt that checks the parity in the Rx is called as parity checker.

Decoding the Received Codewords (Detecting the Error):-

At the receiving End the receiver doesnot know the transmitted word. However it knows A matrix used for generation of code words. Its function is to check the message bits using check bits.

Steps:

1. form the matrix H as $H = [A^T : I_r]$

   Where A^T of Sub. matrix A

   I_r = Identity matrix of order of r (no. of check bits)

   matrix H is called parity check matrix

2. Now if $H \cdot R^T = 0$ — Received word is correct

   $H \cdot R^T \neq 0$ Error in received code.

Ex: ~~For~~ a (4,2) block code received code is 10 11

If $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, find received code is correct or wrong.

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad ; \quad A^T = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

n – msg. bits
k – check bits

$$H = [A^T : I_r]$$

$$= \left[\begin{array}{cc:cc} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right]$$

WKT,

$$R = [1\ 0\ 1\ 1]$$

$$R^T = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$H \cdot R^T = \left[\begin{array}{cc:cc} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right] \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.1 \oplus 0.0 \oplus 1.1 \oplus 0.1 \\ 1.1 \oplus 1.0 \oplus 0.1 \oplus 1.1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

for even no. of 1's of Ex-OR = 0

odd no. of 1's of Ex-OR = 1

$\therefore H \cdot R^T = 0$, received code is correct.

Ex: for a (4,2) block code received code is 1010

If $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, find received code is correct or wrong

$$H = [A^T : I_r] = \left[\begin{array}{cc:cc} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right]$$

$$R^T = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$H \cdot R^T = \left[\begin{array}{cc:cc} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right] \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1.1 \oplus 0.1 \oplus 1.1 \oplus 0.0 \\ 1.1 \oplus 1.0 \oplus 0.1 \oplus 1.0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \therefore H \cdot R^T \neq 0$$, received code is wrong

Error Correction:

- In order to correct the codeword we multiply received codes with transpose of parity check matrix to get syndrome.
- Syndrome is compared with the rows of transpose of parity check matrix. (H^T)
- matching row number is the number of bits in Error. Error bit is then inverted to get the correct code.

Steps :-

1. find $S = RH^T$

   $R \rightarrow$ Received code $H^T \rightarrow$ transpose of H.

   $S \rightarrow [S_1, S_2, S_3 \ldots]$ is called Syndrome.

2. Match the result, i.e S, with rows of H^T. The no. of row where the match occur gives the no of bit in Error. This bit is inverted to correct the Error.

Ex:- for a (6,3) block code received code is 1 1 0 1 0 0

If $A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ find the received code is correct / wrong

If wrong Correct it

$$A^T = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$H = [A^T : I_r] = \left[\begin{array}{ccc:ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

(6)

$$R = [1\ 1\ 0\ 1\ 0\ 0]$$

$$R^T = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$H \cdot R^T = \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1\cdot1 \oplus 1\cdot1 \oplus 1\cdot0 \oplus 1\cdot1 \oplus 0\cdot0 \oplus 0\cdot0 \\ 0\cdot1 \oplus 0\cdot1 \oplus 1\cdot0 \oplus 0\cdot1 \oplus 1\cdot0 \oplus 0\cdot0 \\ 0\cdot1 \oplus 1\cdot1 \oplus 1\cdot0 \oplus 0\cdot1 \oplus 0\cdot0 \oplus 1\cdot0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$\because$ $HR^T \neq 0$, received code is wrong.

Let us find $S = R \cdot H^T$

$$S = [1\ 1\ 0\ 1\ 0\ 0] \underset{1\times6}{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{6\times3}}$$

$1\cdot1 \oplus 1\cdot1 \oplus 0\cdot1 \oplus 1\cdot1 \oplus 0\cdot0 \oplus 0\cdot0 \quad 1\cdot0 \oplus 1\cdot0 \oplus 0\cdot1 \oplus 1\cdot0 \oplus 0\cdot1 \oplus 0\cdot0$

$1\cdot1 \oplus 1\cdot1 \oplus 0\cdot1 \oplus 1\cdot0 \oplus 0\cdot0 \oplus 0\cdot1]$

$$= [1\ 0\ 1]$$

$S = [1\ 0\ 1]$ matches with second row of H^T

∴ Second bit of received code is in Error. By inverting second bit we get,

correct code = $R_c = [1\ 0\ 0\ 1\ 0\ 0]$

Hamming Code

- It provides the detection of a bit Error, also identifies which bit is in Error so that it can be corrected. So it is called as Error detecting and correcting code.

Number of parity bits

→ No. of parity bits depend on the no. of information bits

If no. of parity bits is p, then the relation is

$$2^p \geq x + p + 1$$

Ex: If $x = 4$, then p is formed by trial

Let $p = 2$

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

$4 \geq 7$ ⟶ not Satisfied.

So, let $p = 3$

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

$8 \geq 8$

∴ Three parity bits are required to provide single Error.

Correction for four information bits.

Error Detecting & correcting codes-

Error Detecting codes:-

When binary data is transmitted there is a chance of altering the contents (i.e. is to 0's (or) 0's to 1's).

- there are many Error detecting codes.

Eg are parity, checksum, Block parity etc.

parity:-

The Simplest way to detect Error is "by adding Extra bit known as 'parity Bit' to Each word that is being transmitted.

- There Exist two types of parity: odd & Even.

odd parity:- In odd parity, the no. of 1's in msg bits including the parity bit must be in odd number.

Eg:- 1011 → is odd parity.

Even parity:- there no. of the 1's in the msg bits including parity bit must be in Even number.

Eg: 101101 → is Even parity.

Ques:-

use Even parity, find which words contain Error?

ⓐ 10101010 ⓑ 11110110 ⓒ 10111001

Sol

ⓐ contains Even no. of 1's → no Error

ⓑ contains Even no. of 1's → no. Error

ⓒ contains odd

- Now, we have to find the position, where to add the parity bi

problem

Location of parity bits:-

parity bits are always located in positions correspond to powers of two (i.e, 1, 2, 4, 8, ...)

Eg:- If m=4 ; p=3 ; then

D_4	D_3	D_2	P_3	D_1	P_2	P_1
7	6	5	4	3	2	1

where D_1, D_2, D_3, D_4 represents msg (or) data bits.

P_1, P_2, P_3 represent parity bits.

find the values associated with parity bits.

P_1:- P_1 is associated with 1, 3, 5, 7

(as P_1 location is $2^0=1$, which is Equal-1), so take 1, leave 1 start with Parity bit)

P_2:- P_2 is associated with 2, 3, 6, 7

(as P_2 location is $2^1=2$ (which is Equal to 2), So take 2 leave 2 starting with parity bit (P_2)

P_3:-

As P_3 is associated with 4, 5, 6, 7

(as P_3 location is $2^2=4$ (which is Equal to 4), so take 4 leave 4 starting with parity bit (P_3)

problems on Hamming code:-

Determine the Single Error Correcting code for the following code 10111 for odd parity.

Sol:-

$m = 5$

no. of parity Bits are

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

$16 \geq 10$ (satisfied)

∴ 4 parity bits are added in 1, 2, 4, 8 locations

— The total transmitted bits are $m + p = 5 + 4 = 9$.

9	8	7	6	5	4	3	2	1	→ Bit locations
D5	P4	D4	D3	D2	P3	D1	P2	P1	→ Bit designations
1	P4	0	1	1	P3	1	P2	P1	→ Bit information.

— Now find P_1, P_2, P_3 & P_4 value.

$P_1 \rightarrow 1, 3, 5, 7, 9 = P_1, 1, 1, 0, 1$

$P_1 = 0$ (∵ odd parity is given & it have odd no. of 1's)

$P_2 \rightarrow 2, 3, 6, 7 = P_2, 1, 1, 0$

$P_2 = 1$ (∵ it have even 1's ; $P_2 = 1$ to make odd)

$P_3 \rightarrow 4, 5, 6, 7 = P_3, 1, 1, 0$

$P_3 = 1$ (∵ even 1's are present)

$P_4 \rightarrow 8, 9 = P_4, 1$

$P_4 = 0$ (∵ odd 1's are present)

∴ final code is 100111110

The code is 001100l

problems on Hamming code:-

1) Determine the Single Error Correcting code for the following code 10111 for odd parity.

Sol:-

$m = 5$

no. of parity Bits are

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

$$16 \geq 10 \text{ (satisfied)}$$

∴ 4 parity bits are added in 1, 2, 4, 8 locations

— The total transmitted bits are $m + p = 5 + 4 = 9$.

9	8	7	6	5	4	3	2	1	→ Bit locations
D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1	→ Bit designations
1	P_4:	0	1	1	P_3:	1	P_2:	P_1	→ Bit Information.

— Now find P_1, P_2, P_3 & P_4 values.

$P_1 \rightarrow$ 1, 3, 5, 7, 9 = P_1, 1, 1, 0, 1

$P_1 = 0$ (∵ odd parity is given & it have odd no. of 1's)

$P_2 \rightarrow$ 2, 3, 6, 7 = P_2, 1, 1, 0

$P_2 = 1$ (∵ it have Even 1's ; $P_2 = 1$ to make odd)

$P_3 \rightarrow$ 4, 5, 6, 7 = P_3, 1, 1, 0

$P_3 = 1$ (∵ Even 1's are present)

$P_4 \rightarrow$ 8, 9 = P_4, 1

$P_4 = 1$ (∵ odd 1's are present) ∴ final code is

Sol:- Encode the binary word 1011 in to even parity hamming code

Sol:- $m=4$ ∴ no. of parity bits added is

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

$$p=3$$

∴ Total no. of bits $= m+p = 4+3 = 7$

=	6	5	4	3	2	1	⟶ Bit location.
	D_4	D_3	D_2	P_3	D_1	P_2 P_1	⟶ Bit Designation
	1	0	1		1		⟶ Bit Information.

— Now, find the values of P_1, P_2, P_3

P_1 ⟶ 1, 3, 5, 7 = P_1, 1, 1, 1

∴ $P_1=1$ (∵ It is a Even parity hamming code; P_1 should be 1)

P_2 — 2, 3, 6, 7 = P_2, 1, 0, 1

∴ $P_2=0$ (∵ its already have even no. of 1's)

P_3 ⟶ 4, 5, 6, 7 = P_3, 1, 0, 1

∴ $P_3=0$.

— Finally place these P_1, P_2, P_3 values in the Bit designation.

∴ The final code that is transmitted is

$D_4 P_3 D_2 P_2 D_1 P_1$

= 1 0 1 0 1 0 1

The message below coded in the 7-bit hamming code is transmitted through a noisy channel. Decode the msg assuming that atmost a single Error occured in code word 0011011.

Sol:- The given code word is received hamming code.

- there parity type was not specified. ∴ In default consides Even parity.
- The total data bits + parity bits given = 7.

∴ There are 4 msg bits & 3 parity bits.

D_4	D_3	D_2	P_3	D_1	P_2	P_1 ← Bit position Designation
0	0	1	1	0	1	1
7	6	5	4	3	2	1

- check for parity bits.

$P_1 \rightarrow$ 1, 3, 5, 7 = P, , 0, 1, 0 = 1, 0, 1, 0. (LSB)

$P_2 \rightarrow$ 2, 3, 6, 7 = 1, 0, 0, 0 = 1

$P_3 \rightarrow$ 4, 5, 6, 7 = 1, 1, 0, 0 = 0 (MSB)

∴ The resultant word is 010.

- This says the 2nd location in the given hamming code is wrong.

∴ It is already 1 : ∴ change to 0

- The correct code is 0011001

(3) Assume that the Even parity Hamming Code [illegible] transmitted and that 0100011 is received. The receiver does not know what was transmitted. Determine bit location where Error has occurred. find the correct code.

Sol:- Already Hamming code was given. It is 7 bits.

7	6	5	4	3	2	1
0	1	0	0	0	1	1
D_4	D_3	P_2	P_3	D_1	P_2	P_1

- check for parity bits.

$P_1 \rightarrow 1, 3, 5, 7 \longrightarrow 1, 0, 0, 0.$

There is one 1 in the group (odd)

$\therefore$ parity check for Even parity is wrong.

$\therefore$ put 1 (LSB)

$P_2 \rightarrow 2, 3, 6, 7 \longrightarrow 1, 0, 1, 0.$

there are two 1's in the group (Even)

$\therefore 0.$

$P_3 \rightarrow 4, 5, 6, 7 \longrightarrow 0, 0, 1, 0.$

$\therefore$ parity check is wrong $\longrightarrow 1$

- The resultant word is 101

- This says 5th location is 10, change it to 1 Error

$\therefore$ As, the 5th location is 0, change it to 1

$\therefore$ Receiver corrected the code as 0110011.

4 (a) Devise a Single Error Correcting code for a 11-bit group

01101110101

(b) Test the following hamming code sequence for 11-bit msg and correct it if necessary.

(10100101110101 1)

(a) Given is 11-bit msg bits : $2^4 \geq 11+4+1 \Rightarrow 16 \geq 16$

∴ 4 parity bits must be added. Total = 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1
1	0	1	0	1	1	1		0	1	1		0		

- find parity bit values

$P_1 \rightarrow 1, 3, 5, 7, 9, 11, 13, 15 \rightarrow P_1, 0, 1, 0, 1, 1, 1, 1 \Rightarrow P_1 = 1$

$P_2 \rightarrow 2, 3, 6, 7, 10, 11, 14, 15 \rightarrow P_2, 0, 1, 0, 1, 1, 0, 1 \Rightarrow P_2 = 0$

$P_3 \rightarrow 4, 5, 6, 7, 12, 13, 14, 15 \rightarrow P_3, 1, 0, 1, 0, 1, 0, 1 \Rightarrow P_3 = 0$

$P_4 \rightarrow 8, 9, 10, 11, 12, 13, 14, 15 \rightarrow P_4, 1, 1, 1, 0, 1, 0, 1 \Rightarrow P_4 = 1$

∴ putting the above parity bits the final

hamming code : 1010 1111 011 0001

(b) The given hamming code sequence for 11-bit message is

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1
1	0	1	0	0	1	0	1	1	1	0	1	0	1	1

- check for parity bits

$P_1 \rightarrow$ 1, 3, 5, 7, 9, 11, 13, 15 $\rightarrow$ 1 0 0 1 0 0 1 1 $\Rightarrow$ 0 (Even)

$P_2 \rightarrow$ 2, 3, 6, 7, 10, 11, 14, 15 $\rightarrow$ 1 0 1 1 1 0 0 1 $\Rightarrow$ 1 Error.

$P_3 \rightarrow$ 4, 5, 6, 7, 12, 13, 14, 15 $\rightarrow$ 1 0 1 1 0 1 0 1 $\Rightarrow$ 1 Error.

$P_4 \rightarrow$ 8, 9, 10, 11, 12, 13, 14, 15 $\rightarrow$ 1 0 1 0 0 1 0 1 $\Rightarrow$ 0 No Error (even)

The Error word is $P_4 P_3 P_2 P_1 = 0110 = 6$.

$\therefore$ There is an Error in the 6th position.

— previous it is 1; change it to '0' for correction.

$\therefore$ The correct code to be transmitted is

1 0 1 0 0 1 0 1 1 0 0 1 0 1 1.