Number System

Number system is a basic for counting varies items. Modern computers communicate and operate with binary numbers which use only the digits 0 and 1.

We observe that binary number system take more digits to represent the decimal number. For large numbers we have to deal with very large binary strings. So this fact gave rise to three new number systems.

1) Octol [3bit] 2) Hexadecimol [4bit] 3) BCD

□ Binary to Octal :

$\underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{000}_{0}\ \underbrace{100}_{4}\ \underbrace{101}_{5}\ \underbrace{110}_{6}\ \underbrace{111}_{7}$

□ Binary to Hexadecimol :

$\underbrace{0001}_{1}\ \underbrace{0010}_{2}\ \underbrace{0100}_{4}\ \underbrace{1000}_{8}\ \underbrace{1001}_{9}\ \underbrace{1010}_{A}\ \underbrace{1101}_{D}\ \underbrace{1111}_{F}$

Decimol	Binary	Octal	Hexadecimol
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

□ **Octal to Binary :**

$\underset{110}{6}\ \underset{101}{5}\ \underset{011}{3}$

□ **Hexa to Binary :**

$\underset{0100}{4}\ \underset{1111}{F}\ \underset{1101}{D}\ \underset{0111}{7}$

□ **Octal to Decimal :**

$(4057.06)_8$

$4\times8^3+0\times8^2+5\times8^1+7\times8^0+0\times8^{-1}+6\times8^{-2}$

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

$= (2995.0937)_{16}$

□ **Decimal to Octal :**

$(378.93)_{16} = (?)_8$

$10 \,|\, 378$
$16 \,|\, 47 \quad - 2$
4

```
8 | 378          LSB
8 | 44   - 2      |
8 | 5    - 7      |
    0    - 5      v
                 MSB
```

$= 572.$

$8 \,|\, 0.93$

```
MSB  0 | .93
       |  ×8
     -----------
     7 | .44
       |  ×8
     -----------
     3 | .52
       |  ×8
     -----------
     4 | .16
       |  ×8
     -----------
LSB  1 | .28
```

$= 0.73481$

$\therefore (378.93)_{10} = (572.73481)_8$. (Ans)

* Hexa to Decimal :

$(5C7)_{16} = (?)_{10}$

$= 5 \times 16^2 + C \times 16^1 + 7 \times 16^0$

$= 5 \times 16^2 + 12 \times 16^1 + 7 \times 16^0$

$= 1280 + 192 + 7 = (147)_{16}$. (Ans)

* Decimal to Hexa Decimal :

~~2598~~ $(850)_{10} = (?)_{16}$

$$\begin{array}{r|l}
16 & 850 \\
16 & 53 - 2 \\
16 & 3 - 5 \\
 & 0 - 3
\end{array}$$

LSB ⇓ MSB

$= (352)_{16}$. (Ans)

* Octal to Hexa :

$(756.603)_8$

111 101 110 . 110 000 011

0001 1110 1110 . 1100 0001 1000

$= (1EE.C18)_{16}$

* Hexa to Octal :

B9F. AE16

1011 1001 1111 . 1010 1110 0001 0110

101 110 011 111 . 101 011 100 001 110

$= (5637.53426)_8$

□ Binary Addition :

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 [Carry 1]

* 1100101 এর সাথে 1010101 যোগ কর।

$$\begin{array}{r} 1100101 \\ +\ 1010101 \\ \hline 10111010 \end{array}$$

□ Binary subtraction :

0 − 0 = 0

0 − 1 = 1 [Carry 1]

1 − 0 = 1

1 − 1 = 0

* Subtract 14 from 46 using 8bit 2's complement :

$$\begin{array}{rl} +46 = & 00101110 \\ +14 = & 00001110 \\ & 11110001 \downarrow \text{1's complement} \\ & \quad\quad +1 \\ \hline (-14) = & 11110010 \quad \text{2's complement} \end{array}$$

$$\begin{array}{rl} \therefore +46 = & 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 \\ -14 = & 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ \hline -32 = & (1)\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0 \end{array}$$

Carry bit

MSB ⟶ LSB

As MSB is 0. So the result is +ve.

+0010000 = +32 (Ans)

□ Add -2 and -6 (4bit):

$+2 = 0010$

1101

$+1$

$-2 = 1110$

$+6 = 0110$

1001

$+1$

$-6 = 1010$

$\therefore -2 = 1110$

$-6 = 1010$

$-8 = (1)1000$

Addition is said overflow if the result is too big to fit in the available digits.

If the result of addition / subtraction goes beyond this range, overflow occurs

□. ৮ বিট ব্যবহার করে +৮ হতে +৫ বিয়োগ কর।

$+8 = 00001000$

$+5 = 0000101$

1111010

$+1$

$(-5) = 1111011$

$\therefore +8 = 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0$

$-5 = 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1$

$= (1)\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1$

Carry বিবেচনা করা হয় না

চিহ্ন বিট

যেহেতু চিহ্ন বিট = 0 ∴ সুতরাং Answer হল +3. (Ans)

□ বাইনারি বিয়োগ (General) :

```
          1  0  1  0 0 1 0 0
          0  1  1  0 0 1 1 1   , carry 1+1=0
         ------------------
Result    1  0  0  1 1 0 0 1
     ←    0  0  1  1 1 1 0 1
```

□ **8421 Code :** It's a weighted code in which each decimal digit 0 through 9 is represented by a four bit codeword. BCD is a way to express each of the decimal digit with a binary code.

□ **Gray Code :** The reflected binary code (RBC) also Known as ~~refle~~ Gray Code, is an ordering of the binary numeral system such that two successive values differ in only one bit.

□ **Excess-3 Code :** In excess-3 code, numbers are represented as decimal digits, and each digit is represented by four bits as the digit value plus 3.

□ **Error Detection Code :** An error detection code is a binary code that detects digital errors during transmission. To detect error in the received message, we add some extra bits to the actual data.

■ · Represent $(-17)_{10}$ in sign magnitude, 1's complement and 2's complement representation.

Sol^n

$+17 = 00010001$

$-17 = \boxed{\underbrace{1}_{\text{sign}} | \underbrace{0010001}_{\text{Magnitude}}}$

1's complement form :

$+17 = 00010001$

$-17 = 00010001$

$\boxed{1 | \underbrace{1101110}}$

2's complement form :

$+17 = \boxed{0 | 0010001}$

$-17 = 00010001$

$$\begin{array}{r} 11101110 \\ +1 \\ \hline \boxed{1 | 1101111} \end{array}$$

■ Convert $(541.203)_6$ to base 5, base 8 and base 10.

Sol^n

$= 5\times6^2 + 4\times6^1 + 1\times6^0 + 2\times6^{-1} + 0\times6^{-2} + 3\times6^{-3}$

$= 180 + 24 + 1 + 0.33 + 0.01389$

$= (205.3468)_{10}$

$\therefore (205.3468)_{10} = (?)_5 = (?)_8$

$$\begin{array}{r|l}
5 & 205 \\ \hline
5 & 41 - 0 \\ \hline
5 & 8 - 1 \\ \hline
5 & 1 - 3 \\ \hline
 & 0 - 1
\end{array}
\qquad \text{LSB} \downarrow \text{MSB}$$

$= (1310)_5$

$$\begin{array}{r|l}
8 & 205 \\ \hline
8 & 25 - 5 \\ \hline
8 & 3 - 1 \\ \hline
 & 0 - 3
\end{array}
\qquad \text{LSB} \downarrow \text{MSB}$$

$= (315)_8$

~~0.347~~

$$\begin{array}{r|l}
\text{MSB}\ 0 & .347 \\
 & \times 5 \\ \hline
1 & .735 \\
 & \times 5 \\ \hline
3 & .675 \\
 & \times 5 \\ \hline
3 & .375 \\
 & \times 5 \\ \hline
\text{LSB}\ 1 & .875
\end{array}
\quad \uparrow$$

~~= 0.01~~

$= (0.1331)_5$

$$\begin{array}{r|l}
0 & .347 \\
 & \times 8 \\ \hline
2 & .776 \\
 & \times 8 \\ \hline
6 & .208 \\
 & \times 8 \\ \hline
1 & .664 \\
 & \times 8 \\ \hline
5 & .312
\end{array}$$

$= (0.2615)_8$

$\therefore (205.3468)_{10} = (1310.1331)_5 = (315.2615)_8$

\# Binary Coded Decimal : $\boxed{0 \rightarrow 9}$

. Addition : (i) sum ≤ 9, Carry = 0 হলে, General Binary Addition

(ii) sum ≤ 9, Carry = 1

+6 যোগ করতে হবে = + 0110

(iii) sum > 9, add +6 or = + 0110

$(1)_{16} + (5)_{16}$ –

1 → 0001

5 → 0101

6 → 0110

$(7)_{10} + (3)_{10}$

7 = 0111

3 = 0011

1010

~~0101~~

~~= 1111~~

0110

(0 0 0 1 0 0 0 0) BCD code

= 10

. Subtraction :

(i) যাকে বিয়োগ করবো তার 9's complement বের করতে হবে।

(ii) যার থেকে বিয়োগ করবো তার সাথে 9's complement টুক result যোগ করতে হবে।

a + 9's comp. of b

(iii) যোগ করার পর result বা sum যদি 9 এর চেয়ে বড় হয় তাহলে +6 করতে হবে।

sum > 9, $\boxed{+6}$

sum < 9, 9's complement

(iv) Carry থাকলে তা পরে যোগ করতে হবে।

(v) result < 9, result এর 9's complement.

$(8)_{10} - (4)_{10}$

N = Digit number

n = base of the number

(i) $n^N - 1$

$= 10^1 - 1 = 9 - 4 = 5$ [9's complement]

(ii) ~~9 + 8~~. $(5)_{10} + (8)_{10}$

```
5  →     0101
8  →     1000
      ---------
13 ←     1101
+6       0110
      ---------
       (1)0011
            +1
      ---------
(4) =    0100   [BCD]
```

Excess -3 :

1) Addition:

(i) উভয়ের Ex-3 বের করো

(ii) তারপর যোগ করো

(iii) carry আসলে = 1

+3 ;

(iv) carry = 0

−3

* $(2)_{10} + (5)_{10}$

2 ~~0010~~

```
  2        5
 +3       +3
----     ----
  5        8
  ↓        ↓
0101     1000
```

5 → 0101
8 → 1000

$\boxed{0}$; 1101
carry

```
   1101
(-)0011
   1010
```

2) subtraction :

$(8)_{10} - (3)_{10}$

$10^1 - 1$

$= 9$

-3

$= 6$ [9's complement]

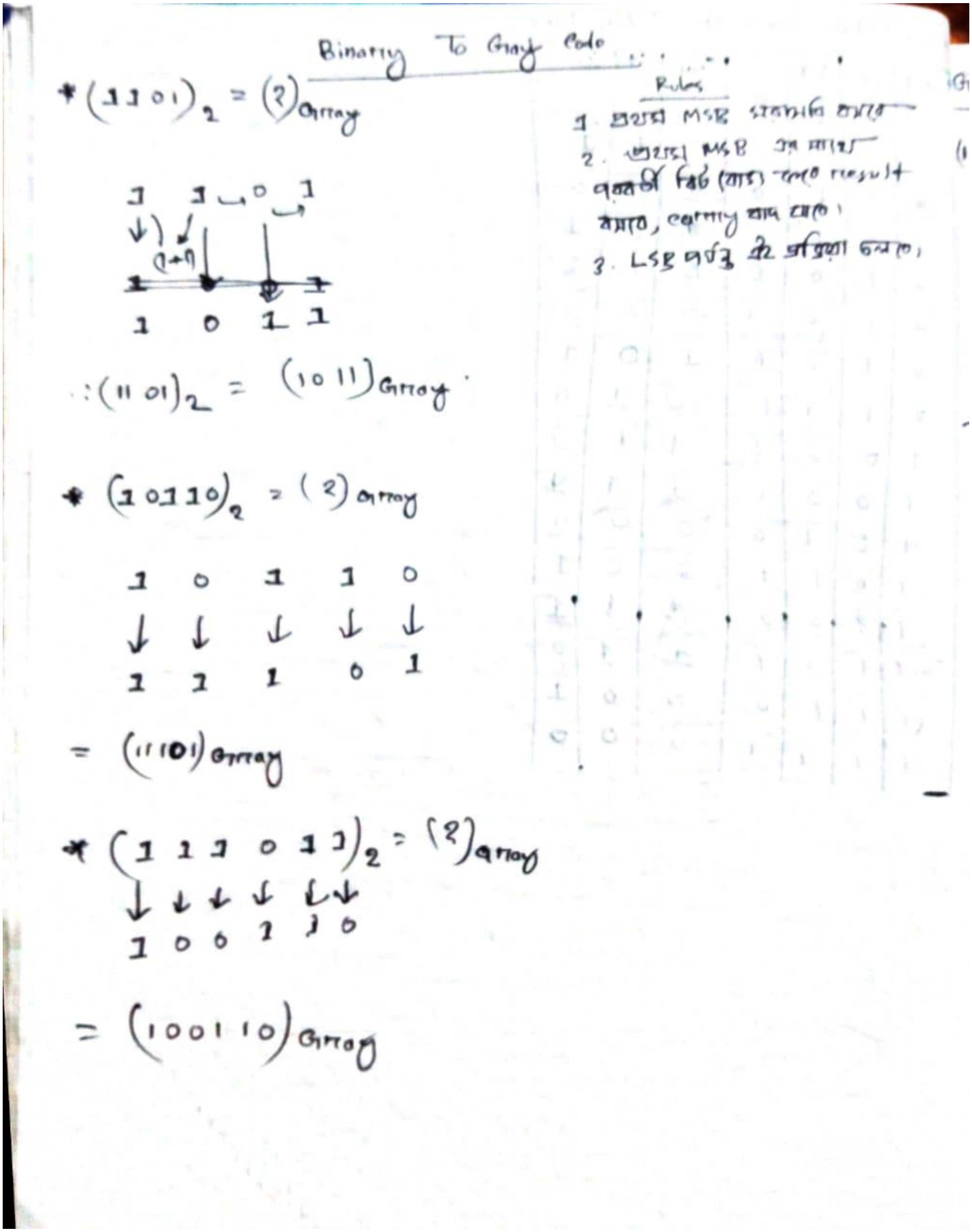
8
+3
11
↓
1011 0110

```
  1011
  0110
[1]0001
   110
[1]0111
```

[illegible]

8 ← 1000

1. যাকে বিয়োগ করবো তার (B)
9's complement বের করবো
2) যার থেকে বিয়োগ করবো/হবে
Ex-3 বের করবো
3) B এর 9's + A এর Ex-3
4) result >9 +6
5) carry = 1 হলে
carry কে বাদ দিয়ে
বাকি result এর সাথে
+1 যোগ করবো।



Binary To Gray Code

Rules

1. প্রথম MSB সরাসরি বসবে
2. প্রথম MSB এর সাথে পরবর্তী বিট (যোগ) করে result বসাবে, carry বাদ যাবে।
3. LSB পর্যন্ত এই প্রক্রিয়া চলবে।

* $(1101)_2 = (?)_{Gray}$

$$\begin{matrix} 1 & 1 & 0 & 1 \\ \downarrow & (1+1) & & \\ 1 & 0 & 1 & 1 \end{matrix}$$

$\therefore (1101)_2 = (1011)_{Gray}$

* $(10110)_2 = (?)_{Gray}$

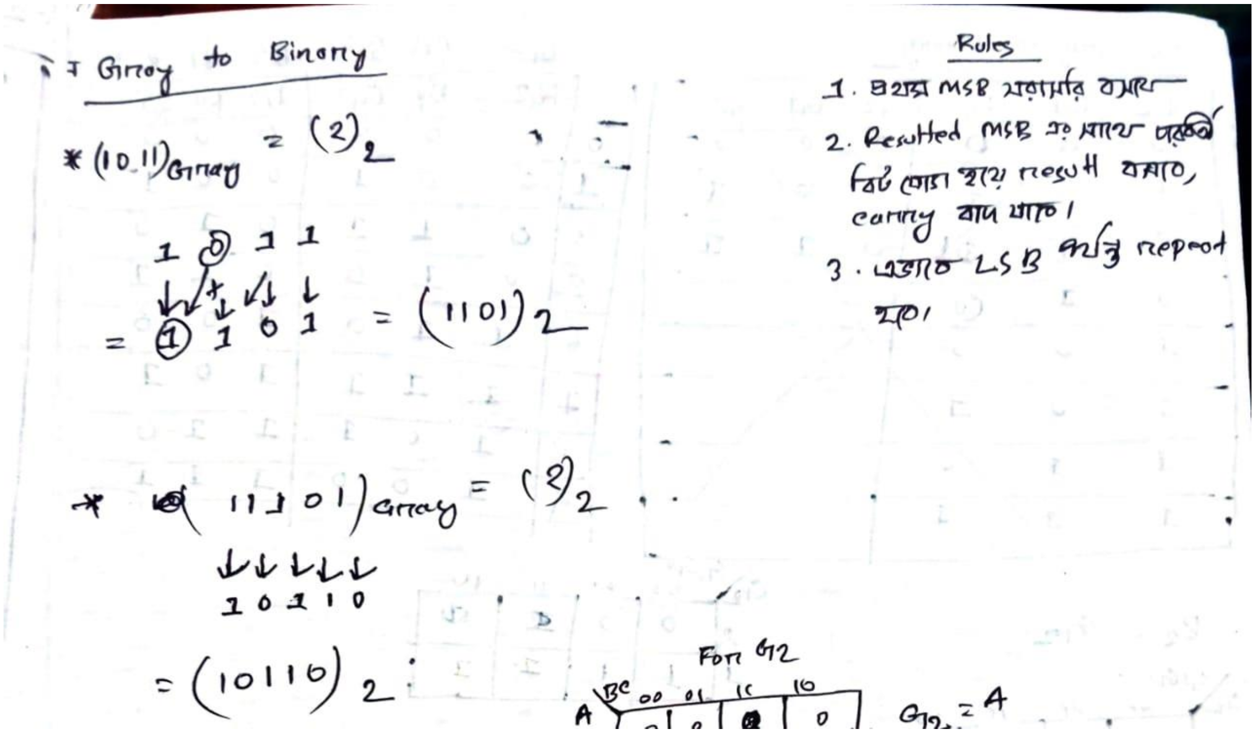
$$\begin{matrix} 1 & 0 & 1 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 1 & 1 & 0 & 1 \end{matrix}$$

$= (11101)_{Gray}$

* $(111011)_2 = (?)_{Gray}$

$$\begin{matrix} 1 & 1 & 1 & 0 & 1 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 0 & 0 & 1 & 1 & 0 \end{matrix}$$

$= (100110)_{Gray}$



Gray to Binary

$*\ (1011)_{Gray} = (?)_2$

$$\begin{array}{cccc} 1 & 0 & 1 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 1 & 0 & 1 \end{array} = (1101)_2$$

$*\ (11101)_{Gray} = (?)_2$

$$\begin{array}{ccccc} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 0 & 1 & 1 & 0 \end{array}$$

$= (10110)_2$

Rules

1. প্রথম MSB সরাসরি বসবে
2. Resulted MSB এর সাথে পরবর্তী বিট যোগ হবে result বসবে, carry বাদ যাবে।
3. এভাবে LSB পর্যন্ত repeat হবে।

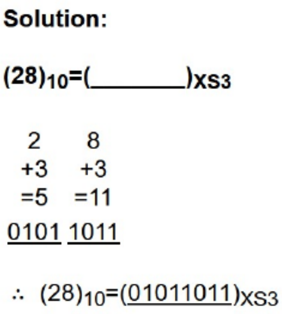
For G2

A
BC
00
01
11
10
0
0
0

$G_2 = A$

16. Represent the decimal number 28 to excess-3 and BCD code.

Decimal 28 to Excess-3:



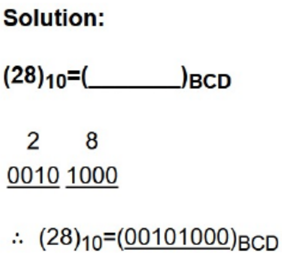
Solution:

$(28)_{10}=(\_\_\_\_\_\_\_)_{XS3}$

2	8
+3	+3
=5	=11
$\underline{0101}$	$\underline{1011}$

∴ $(28)_{10}=(\underline{01011011})_{XS3}$

Decimal 28 to BCD:



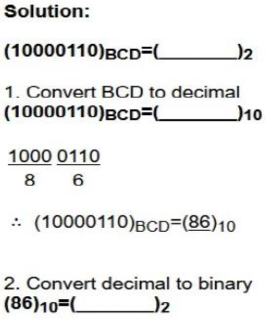
Solution:

$(28)_{10}=(\_\_\_\_\_\_\_)_{BCD}$

2	8
$\underline{0010}$	$\underline{1000}$

∴ $(28)_{10}=(\underline{00101000})_{BCD}$

27. Convert $(1000\,0110)_{BCD}$ to decimal, binary and octal.



Solution:

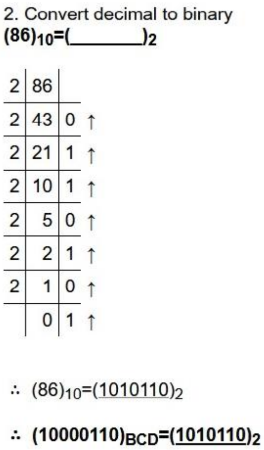
$(10000110)_{BCD}=(\_\_\_\_\_\_\_)_2$

1. Convert BCD to decimal
$(10000110)_{BCD}=(\_\_\_\_\_\_\_)_{10}$

$\underline{1000}$	$\underline{0110}$
8	6

∴ $(10000110)_{BCD}=(\underline{86})_{10}$

2. Convert decimal to binary
$(86)_{10}=(\_\_\_\_\_\_\_)_2$

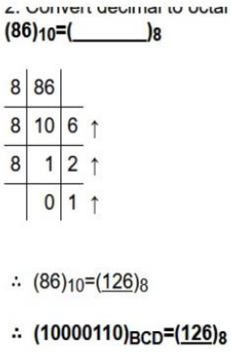


2. Convert decimal to binary
$(86)_{10}=(\_\_\_\_\_\_\_)_2$

2	86	
2	43	0 ↑
2	21	1 ↑
2	10	1 ↑
2	5	0 ↑
2	2	1 ↑
2	1	0 ↑
	0	1 ↑

∴ $(86)_{10}=(\underline{1010110})_2$

∴ $(10000110)_{BCD}=(\underline{1010110})_2$

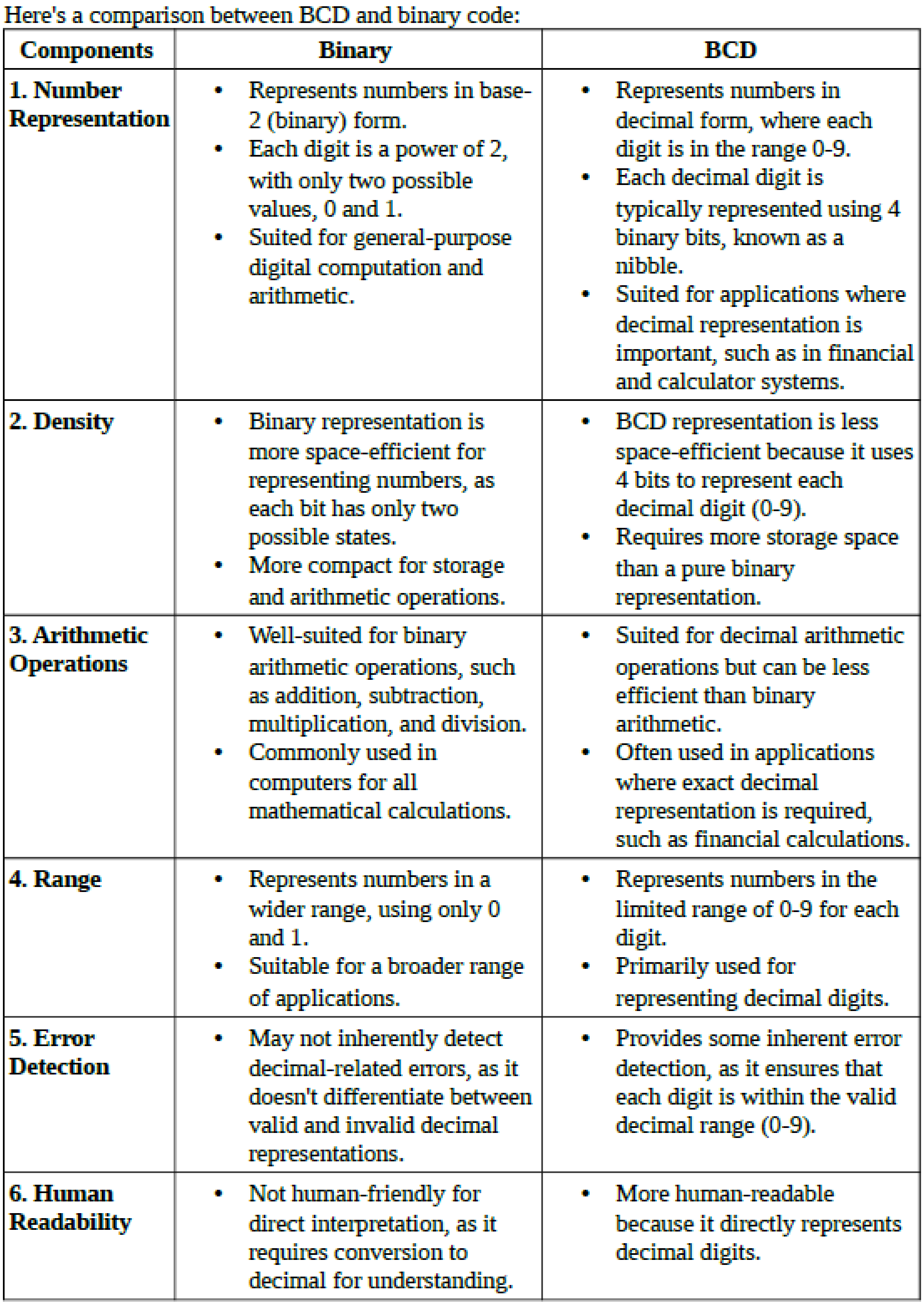


2. Convert decimal to octal
$(86)_{10}=(\_\_\_\_\_\_\_)_8$

8	86	
8	10	6 ↑
8	1	2 ↑
	0	1 ↑

∴ $(86)_{10}=(\underline{126})_8$

∴ $(10000110)_{BCD}=(\underline{126})_8$



Here's a comparison between BCD and binary code:

Components	Binary	BCD
1. Number Representation	• Represents numbers in base-2 (binary) form. • Each digit is a power of 2, with only two possible values, 0 and 1. • Suited for general-purpose digital computation and arithmetic.	• Represents numbers in decimal form, where each digit is in the range 0-9. • Each decimal digit is typically represented using 4 binary bits, known as a nibble. • Suited for applications where decimal representation is important, such as in financial and calculator systems.
2. Density	• Binary representation is more space-efficient for representing numbers, as each bit has only two possible states. • More compact for storage and arithmetic operations.	• BCD representation is less space-efficient because it uses 4 bits to represent each decimal digit (0-9). • Requires more storage space than a pure binary representation.
3. Arithmetic Operations	• Well-suited for binary arithmetic operations, such as addition, subtraction, multiplication, and division. • Commonly used in computers for all mathematical calculations.	• Suited for decimal arithmetic operations but can be less efficient than binary arithmetic. • Often used in applications where exact decimal representation is required, such as financial calculations.
4. Range	• Represents numbers in a wider range, using only 0 and 1. • Suitable for a broader range of applications.	• Represents numbers in the limited range of 0-9 for each digit. • Primarily used for representing decimal digits.
5. Error Detection	• May not inherently detect decimal-related errors, as it doesn't differentiate between valid and invalid decimal representations.	• Provides some inherent error detection, as it ensures that each digit is within the valid decimal range (0-9).
6. Human Readability	• Not human-friendly for direct interpretation, as it requires conversion to decimal for understanding.	• More human-readable because it directly represents decimal digits.

36. What is Gray code? Explain with examples, how do you convert binary to Gray and Gray to binary?**

Gray code

Gray code, also known as reflected binary code or unit distance code, is a binary numeral system where two successive values differ in only one bit position. Gray code is often used in various applications, including rotary encoders, error detection, and analog-to-digital conversion.

Converting Binary to Gray Code

Converting a binary number to Gray code involves a process called "reflecting."
Here's a step-by-step example: Convert the binary number 1101 to Gray code.

Step-01: Start with the most significant bit (leftmost bit), which remains the same in Gray code.
Binary: **1**101
Gray: **1**???

Step-02: For the remaining bits, apply the XOR operation to each bit with the bit to its left.
Binary: 1**1**01
Gray: 1**0**??

Step-03: Continue this process for all bits.
Binary: 11**0**1
Gray: 10**0**0

So, the Gray code representation of the binary number 1101 is 1000.

Converting Gray Code to Binary

Converting Gray code to binary is done by reversing the process. Given a Gray code, you can obtain the corresponding binary representation as follows:
Example: Convert the Gray code 1000 to binary.

Step-01: Start with the most significant bit (leftmost bit), which remains the same in binary.
Gray: **1**???
Binary: **1**???

Step-02: For the remaining bits, apply the XOR operation to each bit with the previously determined bit.
Gray: 10??
Binary: 10**1**?

Step-03: Continue this process for all bits.
Gray: 100**0**
Binary: 101**1**

So, the binary representation of the Gray code 1000 is 1011.

Here are several reasons why the 2's complement method is important and necessary:

- **Representation of Negative Numbers:** The 2's complement method is a widely used technique for representing negative numbers in binary form. It allows both positive and negative numbers to be represented.
- **Addition and Subtraction:** 2's complement simplifies the addition and subtraction of signed binary numbers. In 2's complement representation, subtraction is performed by adding the additive inverse, making it consistent with addition.
- **Efficient Arithmetic Circuits:** Digital arithmetic circuits, such as adders and subtractors, can be designed more efficiently using 2's complement representation. It reduces the need for complex logic to handle sign and magnitude, leading to faster and more compact circuitry.
- **Overflow Handling:** The 2's complement method allows for easy detection and handling of overflow in arithmetic operations. Overflow occurs when the result of an operation is too large to be represented using the available number of bits. 2's complement simplifies the detection of overflow, making it an important aspect of error handling.

48. With the help of example explain excess-3 code.

Excess-3 code, also known as XS-3 or XS-3 binary-coded decimal (BCD), is a binary-coded decimal system where each decimal digit is represented by a 4-bit binary code. The term "excess-3" indicates that the binary representation of each decimal digit is 3 more than the decimal digit itself.

In Excess-3 code, the decimal digits 0 through 9 are represented as follows:

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

Example: Convert the decimal number 648 to Excess-3 code.

1. Break down the decimal number into its individual digits:
   - Hundreds place: 6
   - Tens place: 4
   - Ones place: 8
2. Convert each decimal digit to its Excess-3 code:
   - 6 in decimal is represented as 1001 in Excess-3.
   - 4 in decimal is represented as 0111 in Excess-3.
   - 8 in decimal is represented as 1011 in Excess-3.
3. Combine the Excess-3 codes for each digit to represent the entire number:
   - 648 in decimal is represented as 1001 0111 1011 in Excess-3.

So, the Excess-3 code for the decimal number 648 is 1001 0111 1011.