UPPSC-AE 2020

Uttar Pradesh Public Service Commission

Combined State Engineering Services Examination

Assistant Engineer

Electrical Engineering

Elements of Microprocessors & Numerical Methods

Well Illustrated **Theory** *with*
Solved Examples and **Practice Questions**

MADE EASY Publications

Elements of Microprocessors & Numerical Methods

Contents

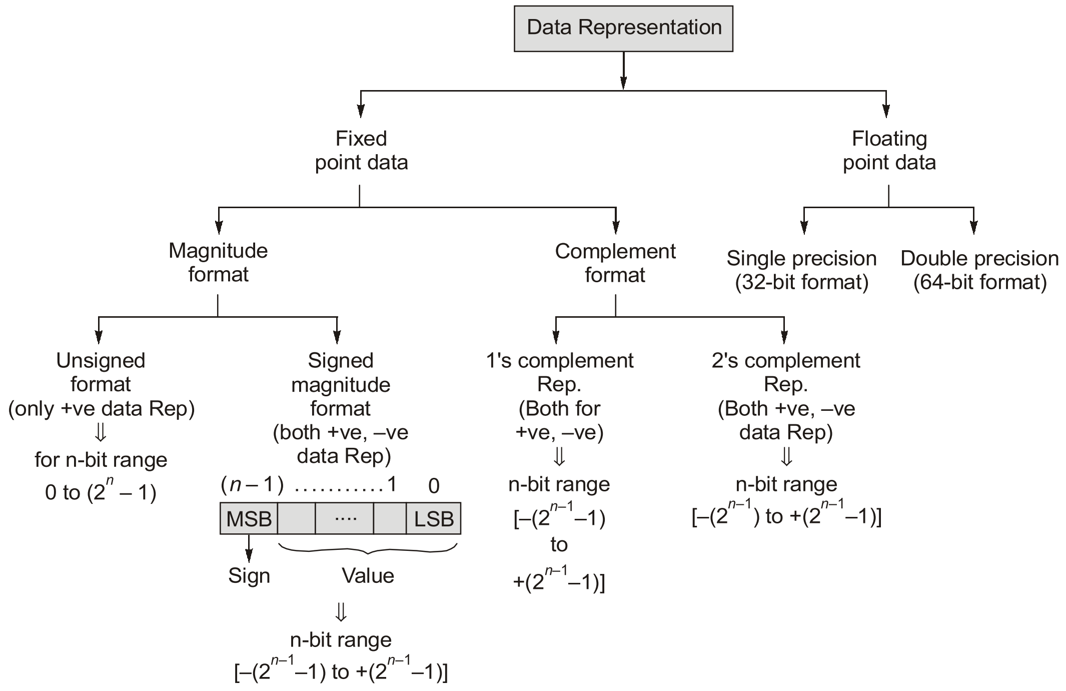
❍❍❍❍

CHAPTER 1

Data Representation and Number Systems

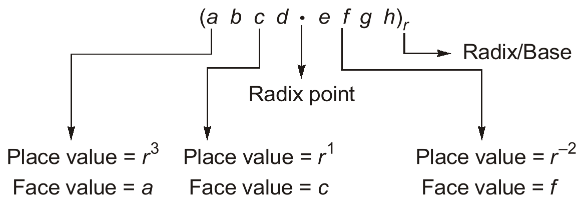
1.1 Digital Number Systems (Positional Weight System)

- In the computer system, data is always represented in a binary format.
- Classification of the data representation is as follows:



Data Representation
Fixed point data
Floating point data
Magnitude format
Complement format
Single precision (32-bit format)
Double precision (64-bit format)
Unsigned format (only +ve data Rep) ⇓ for n-bit range 0 to $(2^n - 1)$
Signed magnitude format (both +ve, –ve data Rep)
$(n-1)$ 1 0
MSB .... LSB
Sign
Value
⇓ n-bit range $[-(2^{n-1}-1)$ to $+(2^{n-1}-1)]$
1's complement Rep. (Both for +ve, –ve) ⇓ n-bit range $[-(2^{n-1}-1)$ to $+(2^{n-1}-1)]$
2's complement Rep. (Both +ve, –ve data Rep) ⇓ n-bit range $[-(2^{n-1})$ to $+(2^{n-1}-1)]$

- Many number systems are used in digital technology.
- A number system is simply a way to count, the most commonly used number systems are:
  - (i) Decimal number system
  - (ii) Binary number system
  - (iii) Octal number system
  - (iv) Hexadecimal number system



- Place value = positional weight
- The digit present in greatest positional weight = Most Significant Digit (MSD)

- The digit present in lowest positional weight = Least Significant Digit (LSD)
- Radix(r) = Different symbols used to represent a number in a number system
  - (i) Decimal ⇒ 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
  - (ii) Binary ⇒ 2 (0, 1)
  - (iii) Octal ⇒ 8 (0, 1, 2, 3, 4, 5, 6, 7, 8, 7)
  - (iv) Hexadecimal ⇒ 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

Example - 1.1 In a particular number system, 24 + 17 = 40. Find the base of the system.

(a) 8 (b) 9
(c) 10 (d) 11

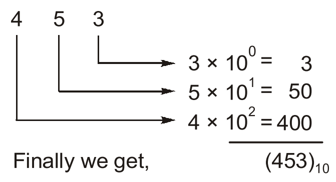
Solution: (d)

$$[(2\times r)+(4\times 1)]+[(1\times r)+(7\times 1)] = (4\times r)+(0\times 1)$$
$$3r+11 = 4r$$
$$r = 11$$

1.1.1 Decimal Number System

- This system has **'base 10'**.
- It has 10 distinct symbols (0, 1, 2, 3, 4, 5, 6, 7, 8 and 9).
- This is a positional value system in which the value of a digit depends on its position.

⇒ Let we have $(453)_{10}$ is a decimal number
then,



∴ We can say "3" is the least significant digit(LSD) and "4" is the most significant digit(MSD).

1.1.2 Binary Number System

- It has base '2' i.e. it has two base numbers 0 and 1 and these base numbers are called "Bits".
- In this number system, group of "Four bits" is known as "Nibble" and group of "Eight bits" is known as "Byte".

i.e. $\boxed{\text{4 bits = 1 Nibble;} \quad \text{8 bits = 1 Byte}}$

Binary to Decimal Conversion : A binary number is converted to decimal equivalent simply by summing together the weights of various positions in the binary number which contains **'1'**.

Decimal to Binary Conversion : The integral decimal number is repeatedly divided by '2' and writing the remainders after each division until a quotient '0' is obtained.

Example - 1.2 Convert $(13)_{10}$ into its equivalent binary number.

(a) $(1010)_2$ (b) $(1011)_2$
(c) $(1100)_2$ (d) $(1101)_2$

Solution: (d)

	Quotient	Remainder	
$13 \div 2$	6	1	LSB
$6 \div 2$	3	0	
$3 \div 2$	1	1	
$1 \div 2$	0	1	MSB

$\therefore$ $(13)_{10} = (1101)_2$

NOTE

To convert Fractional decimal into binary, Multiply the number by '2'. After first multiplication integer digit of the product is the first digit after binary point. Later only fraction part of the first product is multiplied by 2. The integer digit of second multiplication is second digit after binary point, and so on. The multiplication by 2 only on the fraction will continue like this based on conversion accuracy or until fractional part becomes zero.

Example - 1.3 Convert $(0.65625)_{10}$ into its equivalent binary number.

(a) $(1.10101)_2$ (b) $(1.01010)_2$
(c) $(0.01010)_2$ (d) $(0.10101)_2$

Solution: (d)

0.65625	0.31250	0.62500	0.25000	0.50000
× 2	× 2	× 2	× 2	× 2
1.31250	0.62500	1.25000	0.50000	1.00000
1	0	1	0	1

Thus, $(0.65625)_{10} = (0.10101)_2$

1.1.3 Octal Number System

- It is very important in digital computer because by using the octal number system, the user can simplify the task of entering or reading computer instructions and thus save time.
- It has a base of '8' and it posses 8 distinct symbols (0,1...7).
- It is a method of grouping binary numbers in group of three bits.

Octal to Decimal Conversion : An octal number can be converted to decimal equivalent by multiplying each octal digit by its positional weightage.

Decimal to Octal Conversion :

- It is similar to decimal to binary conversion.
- For integral decimal, number is repeatedly divided by '8' and for fraction, number is multiplied by '8'.

Example - 1.4 Convert $(3287.5100098)_{10}$ into its equivalent octal number.

(a) $(5323.4051)_8$ (b) $(5323.5103)_8$
(c) $(6327.4051)_8$ (d) $(6327.5103)_8$

Solution: (c)

For integral part:

	Quotient	Remainder
$3287 \div 8$	410	7
$410 \div 8$	51	2
$51 \div 8$	6	3
$6 \div 8$	0	6

$\therefore$ $(3287)_{10} = (6327)_8$

Now for fractional part:

$$0.5100098 \times 8 = 4.0800784 \downarrow 4 \qquad 0.0800784 \times 8 = 0.6406272 \downarrow 0 \qquad 0.6406272 \times 8 = 5.1250176 \downarrow 5 \qquad 0.1250176 \times 8 = 1.0001408 \downarrow 1$$

$\therefore$ $(0.5100098)_{10} = (0.4051)_8$

Finally, $(3287.5100098)_{10} = (6327.4051)_8$

Octal-to-Binary Conversion : This conversion can be done by converting each octal digit into binary individually.

Binary-to-Octal Conversion : In this conversion the binary bit stream is grouped into groups of three bits starting at the LSB and then each group is converted into its octal equivalent. After decimal point grouping has to start from left.

Example - 1.5 Convert $(1011011110.11001010011)_2$ into its equivalent octal number.

(a) $(1336.3223)_8$
(b) $(1336.3246)_8$
(c) $(1336.6246)_8$
(d) $(1336.6223)_8$

Solution: (c)

For left-side of the radix point, we grouped the bits from LSB:
Here two 0's at MSB are added to make a complete group of 3 bits.
For right-side of the radix point, we grouped the bits from MSB:
Here a '0' at LSB is added to make a complete group of 3 bits.
Finally, $(1011011110.11001010011)_2 = (1336.6246)_8$

$$\underbrace{001}_{1}\ \underbrace{011}_{3}\ \underbrace{011}_{3}\ \underbrace{110}_{6}$$

$$\underset{\substack{\uparrow\\ \text{radix point}}}{\bullet}\ \underbrace{110}_{6}\ \underbrace{010}_{2}\ \underbrace{100}_{4}\ \underbrace{110}_{6}$$

1.1.4 Hexadecimal Number System

- The base for this system is "16", which requires 16 distinct symbols to represent the numbers.
- It is a method of grouping 4 bits.
- This number system contains numeric digits (0, 1, 2,....9) and alphabets (*A*, *B*, *C*, *D*, *E* and *F*) both, so this is an "ALPHANUMERIC NUMBER SYSTEM".
- Microprocessor deals with instructions and data that use hexadecimal number system for programming purposes.
- To signify a hexadecimal number, a subscript 16 or letter '*H*' is used i.e. $(A7)_{16}$ or $(A7)_H$.

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

Hexadecimal-to-Decimal Conversion : An hexadecimal number can be converted to decimal equivalent by multiplying each hexadecimal digit by its positional weightage.

Example - 1.6 Convert $(3A.2F)_{16}$ into its equivalent decimal number.

(a) $(58.1836)_{10}$ (b) $(58.1936)_{10}$
(c) $(58.1867)_{10}$ (d) $(58.1863)_{10}$

Solution: (a)

$$(3A.2F)_{16} = 3\times 16^1 + 10\times 16^{\circ} + 2\times 16^{-1} + 15\times 16^{-2}$$

$$= 48 + 10 + \frac{2}{16} + \frac{15}{16^2} = (58.1836)_{10}$$

Decimal-to-Hexadecimal Conversion :

- It is similar to decimal to binary conversion.
- For integral decimal, number is repeatedly divided by '16' and for fraction, number is multiplied by '16'.

Example - 1.7 Convert $(675.625)_{10}$ into its equivalent Hexadecimal number.

(a) $(2A2.A)_{16}$ (b) $(2A3.A)_{16}$
(c) $(2A3.B)_{16}$ (d) $(2A2.B)_{16}$

Solution: (b)

For Integral Part:

	Quotient	Remainder
$675 \div 16$	42	3
$42 \div 16$	2	$10 = A$
$2 \div 16$	0	2

$\therefore$ $(675)_{10} = (2A3)_{16}$

For Fractional Part:

$625 \times 16 = 10 = A$

$\therefore$ $(0.625)_{10} = (0.A)_{16}$

Finally, $(675.625)_{10} = (2A3.A)_{16}$

Hexadecimal-to-Binary Conversion: For this conversion replace each hexadecimal digit by its 4 bit binary equivalent.

Binary-to-Hexadecimal Conversion : For this conversion the binary bit stream is grouped into pairs of four (starting from LSB) and hex number is written for its equivalent binary group.

Example - 1.8 Convert $(10100110101111)_2$ into its equivalent hexadecimal number.

(a) $(29AB)_{16}$ (b) $(29AE)_{16}$
(c) $(29AD)_{16}$ (d) $(29AF)_{16}$

Solution: (d)

$$\underbrace{00\,10}_{\substack{\downarrow\\2}}\;\underbrace{10\,01}_{\substack{\downarrow\\9}}\;\underbrace{10\,10}_{\substack{\downarrow\\A}}\;\underbrace{1111}_{\substack{\downarrow\\F}}$$

Here two 0's at MSB are added to make a complete group of 4 bits.

$\therefore \quad (10100110101111)_2 = (29AF)_{16}$

The number systems can also be classified as weighted binary number and unweighted binary number. Where weighted number system is a positional weighted system for example, Binary, Octal, Hexadecimal *BCD*, 2421 etc. The unweighted number systems are non-positional weightage system for example Gray code, Excess-3 code etc.

1.2 Codes

- When numbers, letters or words are represented by a special group of symbols, we say that they are being encoded, and the group of symbols is called "CODE".

$$\begin{tikzcd}
 & \textbf{CODES} \arrow[dl] \arrow[d] \arrow[dr] & \\
\textbf{Non-Binary} & \textbf{Binary} & \textbf{Alphanumeric}
\end{tikzcd}$$

Non-Binary
Ex: Morse code
⇒ In Morse code series of dots and dashes represents the letters of the alphabet.

Binary
Ex: Gray code
BCD, Excess-3 Code,
8-4-2-1 code Error detecting and correcting codes

Alphanumeric
Ex: Boudot's code (5 bit code)
Internal code (6 bit)
ASCII code (7 bit + 1bit parity)
EBCDIC (8 bit)

1.2.1 Binary Coded Decimal Code (BCD)

- In this code, each digit of a decimal number is represented by binary equivalent.
- It is a 4-bit binary code.
- It is also known as "8-4-2-1 code" or simply "BCD Code".
- It is very useful and convenient code for input and output operations in digital circuits.
- Also, it is a "weighted code system".

For example:

$(943)_{\text{decimal}} \longrightarrow (\ldots\ldots\ldots)_{\text{BCD}}$

$$\Rightarrow \quad \begin{matrix} 9 & 4 & 3 \\ \downarrow & \downarrow & \downarrow \\ 1001 & 0100 & 0011 \end{matrix}$$

$\therefore \quad (943)_{10} = (100101000011)_2$

Advantages of BCD Code:

- The main advantage of the BCD code is relative ease of converting to and from decimal.
- Only 4-bit code groups for the decimal digits “0 through 9” need to be remembered.
- This case of conversion is especially important from the hardware standpoint.

$\Rightarrow$ In 4-bit binary formats, total number of possible representation = $2^4 = 16$

Then, Valid BCD codes = 10

Invalid BCD codes = 6

$\Rightarrow$ In 8-bit binary formats,

Valid BCD codes = 100

Invalid BCD codes = 256 – 100 = 156

1.2.2 Excess-3 Code

- It is a 4-bit code.
- It can be derived from BCD code by adding “3” to each coded number.
- It is an “unweighted code”.
- It is a “self-complementing code” i.e. the 1’s complement of an excess-3 number is the excess-3 code for the 9’s complement of corresponding decimal number.
- This code is used in arithmetic circuits because of its property of self complementing.

Example - 1.9 Convert $(48)_{10}$ into Excess-3 code.

(a) (0111011) (b) (1000100)
(c) (1000011) (d) (0111110)

Solution: (a)

$$\begin{array}{cc} 4 & 8 \\ +3 & +3 \\ \hline 7 & 11 \\ \downarrow & \downarrow \\ 0111 & 1011 \end{array}$$

$$\therefore \quad (48)_{10} = \underset{\substack{\Downarrow \\ \text{equivalent} \\ \text{4-bit binary}}}{(01111011)}$$

1.2.3 Gray Code

- It is a very useful code also called “minimum change codes” in which only one bit in the code group changes when going from one step to the next.
- It is also known as “Reflected code”.
- It is an unweighted code, meaning that the bit positions in the code groups do not have any specific weight assigned to them.
- This code is not well suited for arithmetic operations but it finds application in input/output devices.
- These are used in instrumentation such as shaft encoders to measure angular displacement or in linear encoders for measurement of linear displacement.

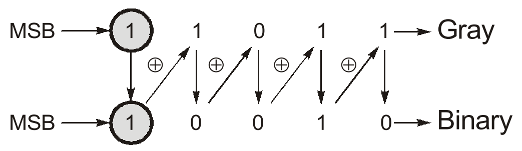
1. **Binary-to-Gray Conversion:**
   - 'MSB' in the gray code is same as corresponding digit in binary number.
   - Starting from "Left to Right", add each adjacent pair of binary bits to get next gray code bit. (Discard the carry if generated).

2. **Gray-to-Binary Conversion:**
   - "MSB" of Binary is same as that of gray code .
   - Add each binary bit to the gray code bit of the next adjacent position (discard the carry if generated), to get next bit of the binary number.

Example - 1.10 Convert the given Gray code 11011 to Binary code.

(a) $(10110)_2$ (b) $(10010)_2$
(c) $(11010)_2$ (d) $(10011)_2$

***Solution:* (b)**



∴ $(11011)_{Gray} = (10010)_2$

3. **Various Binary Codes:**

Decimal Number	Binary				BCD				Excess-3				Gray			
	B_3	B_2	B_1	B_0	D	C	B	A	E_3	E_2	E_1	E_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1
2	0	0	1	0	0	0	1	0	0	1	0	1	0	0	1	1
3	0	0	1	1	0	0	1	1	0	1	1	0	0	0	1	0
4	0	1	0	0	0	1	0	0	0	1	1	1	0	1	1	0
5	0	1	0	1	0	1	0	1	1	0	0	0	0	1	1	1
6	0	1	1	0	0	1	1	0	1	0	0	1	0	1	0	1
7	0	1	1	1	0	1	1	1	1	0	1	0	0	1	0	0
8	1	0	0	0	1	0	0	0	1	0	1	1	1	1	0	0
9	1	0	0	1	1	0	0	1	1	1	0	0	1	1	0	1
10	1	0	1	0									1	1	1	1
11	1	0	1	1									1	1	1	0
12	1	1	0	0									1	0	1	0
13	1	1	0	1									1	0	1	1
14	1	1	1	0									1	0	0	1
15	1	1	1	1									1	0	0	0

1.3 Arithmetic Operations

We are all familiar with the arithmetic operations like addition, subtraction, multiplication and division using decimal numbers. Such operations can also be performed on digital numbers.

1.3.1 Binary Addition

$$\begin{aligned} 0 + 0 &= 0; \quad 0 + 1 = 1 \\ 1 + 0 &= 1; \quad 1 + 1 = 10 \end{aligned}$$

For example:

Add the binary numbers 110110 and 101101

$$\Rightarrow \quad \begin{array}{r} 1\;1\;0\;1\;1\;0 \\ +\,1\;0\;1\;1\;0\;1 \\ \hline \textcircled{1}\,1\;0\;0\;0\;1\;1 \end{array} \quad \longrightarrow \text{Arrow indicates the carry operation}$$

Carry

1.3.2 Binary Subtraction

$$\begin{aligned} 0 - 0 &= 0; \quad 10 - 1 = 1 \text{ (Borrow)} \\ 1 - 0 &= 1; \quad 1 - 1 = 0 \end{aligned}$$

While subtracting a large number from a smaller number, we can subtract the smaller from the larger and change the sign.

For example:

Subtract two binary numbers 11011 and 10110.

$$\Rightarrow \quad \begin{array}{r} 1\;1\;0\;1\;1 \\ -\,1\;0\;1\;1\;0 \\ \hline 0\;0\;1\;0\;1 \end{array} \quad \longrightarrow \text{Represents borrow}$$

1.3.3 Binary Multiplication

Multiply two binary numbers 1010 and 101

$$\Rightarrow \quad \begin{array}{r} \underline{1010 \times 101} \\ 1010 \\ 000\times \\ 1010\times\;\; \\ \hline 110010\;\; \end{array}$$

1.3.4 Octal Addition

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 2 &= 2 \\ 1 + 6 &= 7 \\ 1 + 7 &= 0 \quad \text{with carry} = 1 \end{aligned}$$

Whenever the generated number is greater than 7 then, after decimal addition it should be converted into octal.

For example:

$$7 + 2 = 9 \qquad \begin{array}{r|l} 8 & 9 \\ \hline & 1 \rightarrow 1 = 11 \end{array}$$

$$7 + 17 = 26 \qquad \begin{array}{r|l} 8 & 14 \\ \hline & 1 \rightarrow 6 = 16 \end{array}$$

Example - 1.11 Add two octal numbers 567 and 243.

(a) 1032 (b) 2064

(c) 3094 (d) 4096

Solution: (a)

$$\Rightarrow \qquad \begin{array}{r} {}^{1\,1} \\ 567 \\ +\,243 \\ \hline 1032 \end{array} \qquad \text{Here,} \qquad \begin{array}{l} 7+3=10 \\ 8 \,|\, 10 \\ \quad 1 \rightarrow 2 \\[1em] 10+1=11 \\ 8 \,|\, 11 \\ \quad 1 \rightarrow 3 \end{array}$$

1.3.5 Octal Subtraction

Let, the two octal numbers to be subtracted are 723 and 564.

$$\Rightarrow \qquad \begin{array}{r} {}^{8}\ \ {}^{8} \\ 7\ 2\ 3 \\ -\,5\ 6\ 4 \\ \hline 1\ 3\ 7 \end{array} \qquad \longrightarrow \text{Represents the borrow}$$

1.3.6 Hexadecimal Addition

$$\begin{aligned} 1+1 &= 2 \\ 1+9 &= A \\ 1+15 &= 0 \quad \text{with carry '1'} \\ A+A &= 14 \end{aligned} \qquad \begin{array}{l} 16 \,|\, 20 \\ \quad 1 \rightarrow 4 \end{array}$$

1.3.7 Hexadecimal Subtraction

Subtract 974B to 587C

$$\Rightarrow \qquad \begin{array}{r} {}^{16}\ {}^{16}\ {}^{16} \\ 9\ 7\ 4\ B \\ -\,5\ 8\ 7\ C \\ \hline 3\ E\ C\ F \end{array} \qquad \longrightarrow \text{Represents borrow}$$

1.3.8 BCD Addition

- Addition is the most important operation because subtraction, multiplication, and division can all be done by a series of additions or two's-complement additions.
- The procedure for BCD addition is as follows:
  (i) Add the BCD numbers as regular true binary numbers.
  (ii) If the sum is 9(1001) or less, it is a valid BCD answer; leave it as it is.
  (iii) If the sum is greater than 9 or if there is a carry-out of the MSB, it is an invalid BCD number.
  (iv) If it is invalid, add 6 (0110) to the result to make it valid. Any carry-out of the MSB is added to the next-more-significant BCD number.
  (v) Repeat steps 1 to 4 for each group of BCD bits.

Example - 1.12 Convert the decimal number $(76)_{10}$ and $(94)_{10}$ in BCD and add them. Convert the result back to decimal to check the answer.

(a) $(170)_{10}$ (b) $(175)_{10}$
(c) $(180)_{10}$ (d) $(185)_{10}$

Solution: (a)

$$\begin{array}{rcll} 76 &=& 0111 & 0110 \\ +\,94 &=& 1001 & 0100 \\ \hline && \boxed{10000} & \boxed{1010} \end{array}$$

From the rule we add (0110)

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

In BCD $\Rightarrow (1\ 7\ 0)_{10}$

Also, we have,

$$\begin{array}{r} (76)_{10} \\ +(94)_{10} \\ \hline (170)_{10} \end{array}$$

1.4 Unsigned Integers

- Unsigned integers can represent zero and positive integers, but not negative integers.
- The value of an unsigned integer is interpreted as "the magnitude of its underlying binary pattern".
- An *n*-bit unsigned integer can represent integers from 0 to $(2^n - 1)$.

n	Minimum	Maximum
8	0	$2^8 - 1$
16	0	$2^{16} - 1$
32	0	$2^{32} - 1$
64	0	$2^{64} - 1$

Example - 1.13 Suppose that n = 8 and the binary pattern is $(0100\ 0001)_2$, the value of this unsigned integer is

(a) $(32)_{10}$ (b) $(60)_{10}$
(c) $(62)_{10}$ (d) $(65)_{10}$

Solution: (d)

Given, binary pattern $(0100\ 0001)_2 = 1 \times 2^6 + 1 \times 2^0 = (65)_{10}$

1.5 Signed Number Representation

	Binary pattern with n-bits		Equivalent decimal w.r.t. sign magnitude binary	Equivalent decimal w.r.t. 1's complement binary	Equivalent decimal w.r.t. 2's complement binary
	0000	$\longrightarrow$	+ 0	$\boxed{+0}$	+ 0
	0001	$\longrightarrow$	+ 1	+ 1	+ 1
	0010	$\longrightarrow$	+ 2	+ 2	+ 2
	0011	$\longrightarrow$	+ 3	+ 3	+ 3
	0100	$\longrightarrow$	+ 4	+ 4	+ 4
	0101	$\longrightarrow$	+ 5	+ 5	+ 5
	0110	$\longrightarrow$	+ 6	+ 6	+ 6
	0111	$\longrightarrow$	+ 7	+ 7	+ 7
MSB $\longleftarrow$	1000	$\longrightarrow$	$\boxed{-0}$	– 7	– 8
$\Downarrow$ (–ve)	1001	$\longrightarrow$	– 1	– 6	– 7
	1010	$\longrightarrow$	– 2	– 5	– 6
	1011	$\longrightarrow$	– 3	– 4	– 5
	1100	$\longrightarrow$	– 4	– 3	– 4
	1101	$\longrightarrow$	– 5	– 2	– 3
	1110	$\longrightarrow$	– 6	– 1	– 2
	1111	$\longrightarrow$	– 7	$\boxed{-0}$	– 1

'0' is the redundant
Drawback: Two possible representations +0, –0 i.e. $+0 \rightarrow 0000$, $-0 \rightarrow 1000$

No need of redundency of '0' because unique represent of '0'.

- So far, we have considered only positive numbers but the representation of negative number is also equally important.
- There are basically two ways of representing signed number as
  (i) Sign magnitude representation.
  (ii) *r*'s complement representation.

1.5.1 Sign Magnitude Representation

- In sign magnitude form, the most significant bit (MSB) is used to represent sign (0 for positive and 1 for negative number) and the remaining bits are used to represent the magnitude of the number. For example, a 4-bit binary number 0011 represents a positive number (3) and 1011 represents a negative number (–3).
- In general maximum positive number that can be represented using sign magnitude form is $(2^{n-1}-1)$ and the maximum negative number that can be represented is $-(2^{n-1}-1)$; where *n* is the number of bits.

1.5.2 r's Complement Representation

- In *r*'s complement representation '*r*' represents the radix. It can be divided as
  (i) $(r-1)$'s complement. (ii) *r*'s complement.
- Depending upon the radix or base of different number systems the complement representations are
- Binary number system:
  - → 1's complement
  - → 2's complement
- Decimal number system:
  - → 9's complement
  - → 10's complement
- Octal number system:
  - → 7's complement
  - → 8's complement
- Hexadecimal number system:
  - → F's complement
  - → 16's complement
- To determine the $(r-1)$'s complement subtract the given number from the maximum possible number in the given base (i.e. (2^n-1)) number in binary.
  for example, for 4 bit maximum possible number = $2^4-1=(15)_{10}=(1111)_2$

Example - 1.14 Determine 9's complement of a decimal number 2689?

(a) 7299 (b) 7310
(c) 7399 (d) 7400

Solution: (b)

For a decimal number maximum possible number of 4 digit is 9999

$$\therefore \text{ 9's complement of 2689 is } \begin{array}{r} 9999 \\ -\,2689 \\ \hline 7310 \end{array}$$

Note: To determine the r's complement, first write (r – 1)'s complement of given number then add 1 in the least significant position.