Chapter 3: Representation of Information and Computer Arithmetic

3.1 Number system revision

A computer's internal storage techniques are different from the way people represent information in lives. Information inside a digital computer is stored as a collection of binary data (0's and 1's).

- Binary information is stored in ***memory*** or ***processor registers.***
- Registers contain either ***data*** or ***control information.***
- Data are numbers and other binary-coded information
- Control information is a bit or a group of bits used to specify the sequence of command signal

Data types found in the registers of digital computers

- *Numbers* used in arithmetic computations
- *Letters* of the alphabet used in data processing
- *Other discrete symbols* used for specific purpose

Data Representation in Computers

Units of Data Representation (Bit, Byte, Word)

The entire circuitry of the computer is designed on the basis of binary system. The Binary number system is a group of zeros and ones. It is obvious that computer's net intelligence is absolutely zero i.e. computer is not capable of understanding anything except zeros and ones. In computer every instruction is interpreted and executed in the form of binary system. From the readability point of view it is very difficult to understand the string of bits. It is also very difficult to write and understand the instruction or program written in binary form. Therefore, the programs are generally written in high-level language or Assembly language. Later on these programs are converted into machine language with the help of appropriate translators such as Compiler, Assembler or Interpreter.

The basic unit of the memory is a Bit. A Bit is an abbreviation for a Binary digit and can be either a 0 or a 1. Group of continuous 4 bits is termed as Nibble and eight adjacent bits designed to store the binary code of a single character (letter, decimal digit or other character) isreferred as Byte. A word is a fixed-sized group of bits that are handled together by the machine. The number of bits in a word (the word size or word length) is an important characteristic of computer architecture.

A Word consists of 32 bits, which is equal to 4 bytes (this depends on the computer i.e., a word may contain 8, 16, or 32 bits). Commonly used notation in modern computers is 32 bits. Double word is 2 times a word. Frequently capacity of memory is represented in terms of K (Kilo) , which is equivalent to 1024 words of 8 bits each (210 bytes).

- 1 KILO = 1024 bytes (approx. 10^3 bytes)
- 1 MEGA = 10^3 K B
- 1 GIGA = 10^3 M B
- 1 TERA = 10^3 G B
- 1 PETA = 10^3 TERA B

These are all the approximate values

B stands for Byte

Why Binary representation is used in computers?
A computer’s internal storage techniques are different from the way people represent information in lives. Information inside a digital computer is stored as a collection of binary data (0’s and 1’s). The logic that computer uses is Boolean logic which is a two valued logic and thus the two states of a binary system can relate directly to two states in a logical system. It was easier to make hardware which can distinguish between two values than multiple values. Other bases need more circuitry as compared to binary and this reduced reliability. This binary system simplifies the design of the circuits, reduces the cost and improves the reliability. Every operation that can be done in decimal system can also be done in binary.

Concept of Number Systems and Binary Arithmetic

Number Systems

Every computer stores numbers, letters, and other special characters in a coded form. Before going into the details of these codes, it is essential to have a basic understanding of the number system. So the goal of this chapter is to familiarize you with the basic fundamentals of number system. A number system defines a set of values used to represent quantity. Today the most common number system in use is Arabic system. But, the number systems can be categorized in to two broad categories: Positional number systems & Non-positional number systems.

Non-Positional number systems: - is a method of counting on fingers such as I for 1, II for 2, III for 3, IIII for 4 etc. It was very difficult to perform arithmetic with such a number system, as it had no symbol for zero.

Positional number systems: - the value of each digit in a number is defined not only by the symbol but also by the symbol’s position. They have a base or radix. In positional number system there are only a few symbols called digits, and these symbols represent different values depending on the position they occupy in the number. The value of each digit in such a number is determined by three considerations.

- The digit itself.
- The position of the digit in the number, and
- The base or radix of the number system(where base is determined as the total number of digits available in the number system)

Base (radix): tells the number of symbols used in the system. Modern computers use the radix 2 because they recognize only two symbols, which are represented in digital circuits as 0s and 1s. The base of a number system is indicated by a subscript (decimal number) and this will be followed by the value of the number.

<u>For example</u> $(7592)_{10:}$ is of base 10 number system.

$(1010)_{2:}$ is of base 2 number system.

Note: For a computer, everything is a number whether it may be numbers, alphabets, punctuation marks etc. Eventually, the number systems that can are generally used by the computers are:

- Decimal system
- Binary system
- Octal system
- Hexadecimal system

Inside the digital computer Data Representation ways are:

- Number systems
    - Binary
    - Octal
    - Hexadecimal
- Coding / Alphabetic Characters systems
    - BCD coding
    - EBCIDIC
    - ASCII-7/8

Number system

Note: For a computer, everything is a number whether it may be numbers, alphabets, punctuation marks etc. Eventually, the number systems that are generally used by the computers are:

- Decimal system
- Binary system
- Octal system
- Hexadecimal system

Table 3.1: Truth table of Number system

Number system	Radix value	Set of Digits	Example
Decimal	R = 10	(0,1,2,3,4,5,6,7,8,9)	$(25)_{10}$
Binary	R = 2	(0,1)	$(11001)_2$
Octal	R = 8	(0,1,2,3,4,5,6,7)	$(31)_8$
Hexadecimal	R = 16	(0,1,2,3,4,5,6,7,8,9,A, B,C,D,E,F)	$(1A)_{16}$

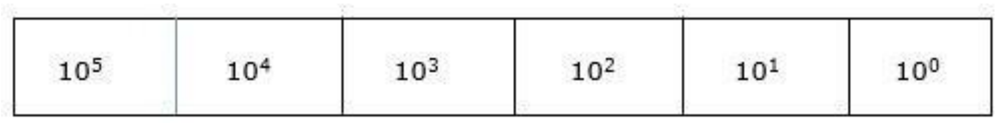
Decimal Number System

Decimal number system is a **base 10** number system having 10 digits from 0 to 9. This means that any numerical quantity can be represented using these 10 digits. Decimal number system is also a **positional value system**. This means that the value of digits will depend on its position. Let us take an example to understand this.

Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−

- In 734, value of 7 is 7 hundreds or 700 or 7 × 100 or 7 × 102
- In 971, value of 7 is 7 tens or 70 or 7 × 10 or 7 × 101
- In 207, value 0f 7 is 7 units or 7 or 7 × 1 or 7 × 100

The weightage of each position can be represented as follows −



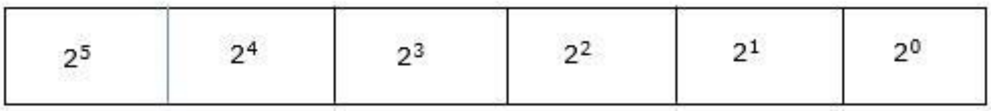
10^5	10^4	10^3	10^2	10^1	10^0

In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed. Let's look at them in detail.

Binary Number System

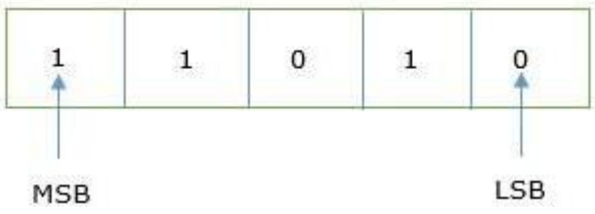
The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two digits – 0 and 1 – is called **binary number system**.

Each binary digit is also called a **bit**. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here.



2^5	2^4	2^3	2^2	2^1	2^0

In any binary number, the rightmost digit is called **least significant bit (LSB)** and leftmost digit is called **most significant bit (MSB)**.



And decimal equivalent of this number is sum of product of each digit with its positional value.

$11010_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 16 + 8 + 0 + 2 + 0$

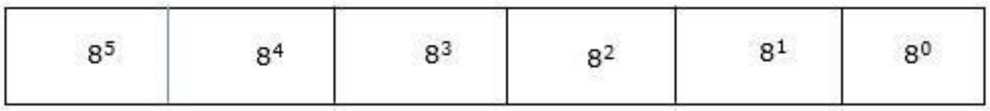
$= 26_{10}$

Computer memory is measured in terms of how many bits it can store. Here is a chart for memory capacity conversion.

- 1 byte (B) = 8 bits
- 1 Kilobytes (KB) = 1024 bytes
- 1 Megabyte (MB) = 1024 KB
- 1 Gigabyte (GB) = 1024 MB
- 1 Terabyte (TB) = 1024 GB
- 1 Exabyte (EB) = 1024 PB
- 1 Zettabyte = 1024 EB
- 1 Yottabyte (YB) = 1024 ZB

Octal Number System

Octal number system has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here −



8^5	8^4	8^3	8^2	8^1	8^0

Decimal equivalent of any octal number is sum of product of each digit with its positional value.

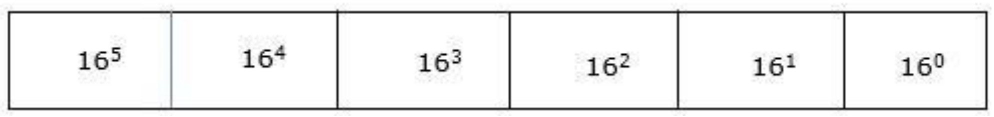
$726_8 = 7\times8^2 + 2\times8^1 + 6\times8^0$

$= 448 + 16 + 6$

$= 470_{10}$

Hexadecimal Number System

Hexadecimal number system has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16, as shown here −



16^5	16^4	16^3	16^2	16^1	16^0

Decimal equivalent of any hexadecimal number is sum of product of each digit with its positional value.

$27FB_{16} = 2\times16^3 + 7\times16^2 + 15\times16^1 + 11\times16^0$

$= 8192 + 1792 + 240 + 11$

$= 10235_{10}$

Number System Relationship

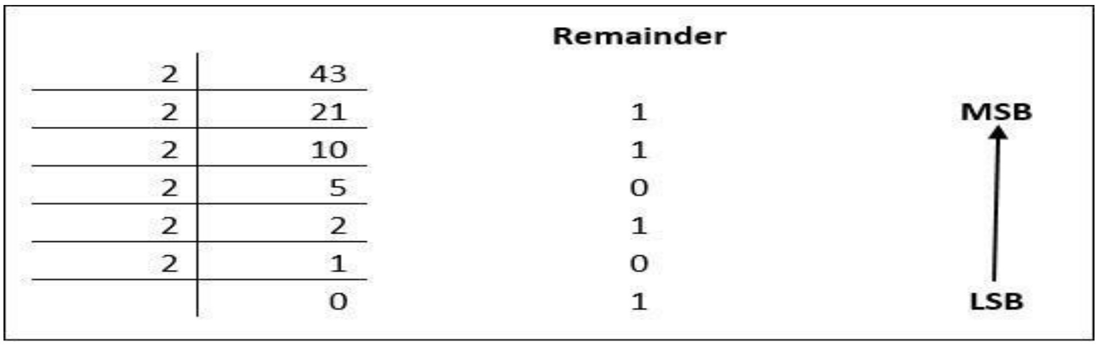
The following table depicts the relationship between decimal, binary, octal and hexadecimal number systems.

Table 3.2: Number System Relationship

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

Decimal to Binary

Decimal numbers can be converted to binary by repeated division of the number by 2 while recording the remainder. Let’s take an example to see how this happens.



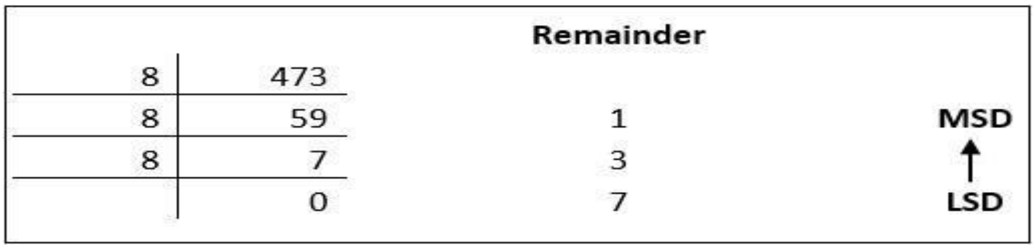
		Remainder	
2	43		
2	21	1	MSB
2	10	1	↑
2	5	0	
2	2	1	
2	1	0	
	0	1	LSB

The remainders are to be read from bottom to top to obtain the binary equivalent.

$43_{10} = 101011_2$

Decimal to Octal

Decimal numbers can be converted to octal by repeated division of the number by 8 while recording the remainder. Let's take an example to see how this happens.



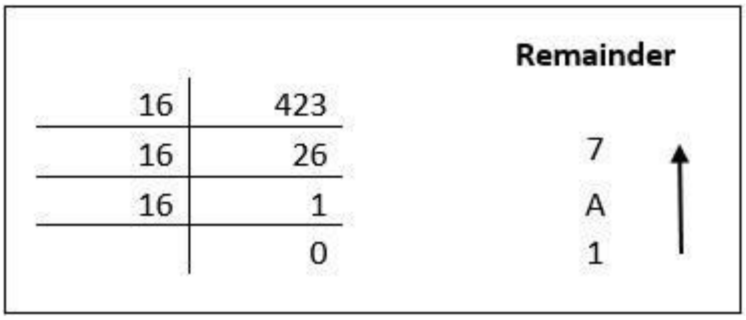
		Remainder	
8	473		
8	59	1	MSD
8	7	3	↑
	0	7	LSD

Reading the remainders from bottom to top,

$473_{10} = 731_8$

Decimal to Hexadecimal

Decimal numbers can be converted to octal by repeated division of the number by 16 while recording the remainder. Let's take an example to see how this happens.



		Remainder	
16	423		
16	26	7	↑
16	1	A	
	0	1	

Reading the remainders from bottom to top we get,

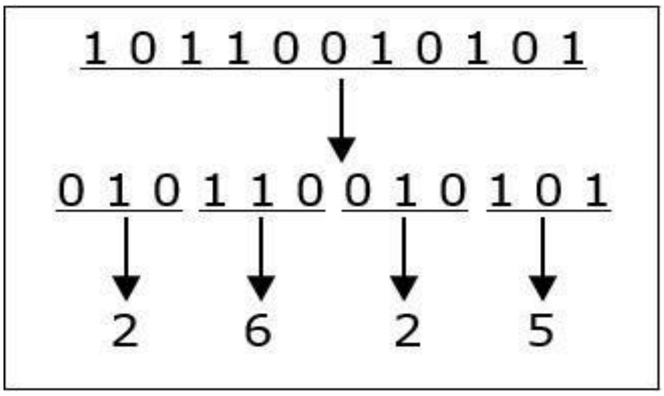
$423_{10} = 1A7_{16}$

Binary to Octal and Vice Versa

To convert a binary number to octal number, these steps are followed −

- Starting from the least significant bit, make groups of three bits.
- If there are one or two bits less in making the groups, 0s can be added after the most significant bit
- Convert each group into its equivalent octal number

Let’s take an example to understand this.



$1011001010_{12} = 2625_8$

To convert an octal number to binary, each octal digit is converted to its 3-bit binary equivalent according to this table.

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

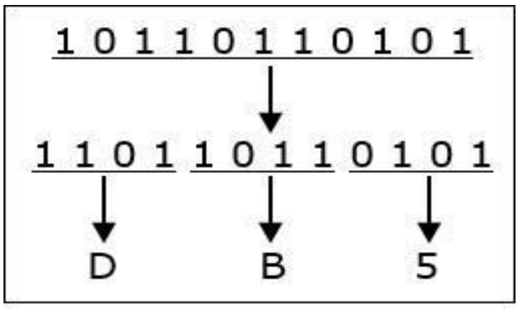
$54673_8 = 101100110111011_2$

Binary to Hexadecimal

To convert a binary number to hexadecimal number, these steps are followed −

- Starting from the least significant bit, make groups of four bits.
- If there is one or two bits less in making the groups, 0s can be added after the most significant bit.
- Convert each group into its equivalent hexadecimal number.

Let’s take an example to understand this.



$10110110101_2 = DB5_{16}$

To convert hexadecimal number to binary, each hexadecimal digit is converted to its 4-bit binary equivalent.

Arithmetic System:-During school days, arithmetic was restricted only to decimal number system. However, in computer, we require arithmetic on other number systems.

- **Binary arithmetic**: Everything that is stored in or manipulated by the computer is a number. The computer understands the numbers 1 and 0. The basic arithmetic operations of the binary number system are: **Addition** and **subtraction**.

- **Binary Addition**: Binary addition is carried out in the same way as the decimal addition is performed. There are four outcomes or rules of the binary addition. These are shown below:

Input		Output	
X	Y	Sum (S)	Carry (R)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Example: Add the binary numbers 1111 and 1010.

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

Thus the binary sum is **$(11001)_2$**.

- **Binary Subtraction**: Subtraction is generally simple in comparison to addition since only two numbers are involved. In binary subtraction the problem ‘borrow’ is similar to that in decimal. If the subtracted bit is equal to or smaller than the minuend bit, then perform subtraction, otherwise borrow one from its left most neighbor. Binary subtraction follows four rules for the operation.

Input		Output	
X	Y	Subtraction(S)	Borrow (R)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- **Complements**

Represent Negative Numbers on Computers

There are different ways of representing negative numbers in a computer.

- One's Complement Method
- Two's Complement Method

1's complement

The 1's complement of binary number is similar to 9's complement in the decimal system. To get 1's complement of a binary number each bit of the binary number is subtracted from 1. The 1's complement of 01 is 10. Thus we can see that the 1's complement of a binary number can be obtained by simply changing each bit 1 to 0 and 0 to 1.

One's Complement Method

- All the 1 bits is changed to 0 and all the 0 bit is changed to 1 to get the complement of a number.
- One's Complement of the number is treated as the negative of that number.

Example: +5 $\boxed{\mathbf{0}}\boxed{101}$ =

-5 $\boxed{\mathbf{1}}\boxed{010}$ =

Ex1: Find 1's complement of 1100101

The 1's complement is 0011010

Ex2: +2 is 00000010

-2 is 11111101

Note that in this representation positive numbers start with a 0 on the left, and negative numbers start with a 1 on the left most bit.

2's Complement:

The 2's complement of the binary system is similar to 10's complement in the decimal system. Thus the 2's complement of a binary number is equal to the 1's complement of the number plus one.

Two's Complement Method

Two's complement of a number is **one's complement of a number + 1.**

Two's complement of a number is treated as the negative representation of the number.

Example 1: $11010_{(2)}$ =? 2's complement

1's complement of 11010 is 00101

$$\begin{array}{r} 00101 \\ +\quad 1 \\ \hline \mathbf{00110} \\ \hline \end{array}$$

⇨ $11010_{(2)}$ = **00110 (2's complement)**

Example 2: $0000_{(2)}$ =? 2's complement

1's complement of 0000 is 1111

$$\begin{array}{r} 1111 \\ 1 \\ \hline \underline{\mathbf{10000}} \end{array}$$

0000 =>**1** **0000** (here 1 is carry)

Note: **Consider the representation of zero in all the above methods.**

Sign and Magnitude Method		***One's Complement Method***		***Two's Complement Method***	
+0	0000	+0	0000	+0	0000
-0	1000	-0	1111	-0	$\boxed{1}$ 0000

- Zero has two different representations in sign and magnitude method and one's complement method which is not correct. But in 2's complement method there is only one way representation for zero. And so 2's complement method is used in modern computing.

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

Table 3.5: Binary Code