Number System

Introduction

Number systems provide the basis for all operations in information processing systems. In a number system the information is divided into a group of symbols; for example, 26 English letters, 10 decimal digits etc. In conventional arithmetic, a number system based upon ten units (0 to 9) is used. However, arithmetic and logic circuits used in computers and other digital systems operate with only 0's and 1's because it is very difficult to design circuits that require ten distinct states. The number system with the basic symbols 0 and 1 is called binary. ie. A binary system uses just two discrete values. The binary digit (either 0 or 1) is called a bit.

A group of bits which is used to represent the discrete elements of information is a symbol. The mapping of symbols to a binary value is known a binary code. This mapping must be unique. For example, the decimal digits 0 through 9 are represented in a digital system with a code of four bits. Thus a digital system is a system that manipulates discrete elements of information that is represented internally in binary form.

Decimal Numbers

The invention of decimal number system has been the most important factor in the development of science and technology. The decimal number system uses positional number representation, which means that the value of each digit is determined by its position in a number.

The base, also called the radix of a number system is the number of symbols that the system contains. The decimal system has ten symbols: 0,1,2,3,4,5,6,7,8,9. In other words, it has a base of 10. Each position in the decimal system is 10 times more significant than the previous position. The numeric value of a decimal number is determined by multiplying each digit of the number by the value of the position in which the digit appears and then adding the products. Thus the number 2734 is interpreted as

$$2 \times 1000 + 7 \times 100 + 3 \times 10 + 4 \times 1 = 2000 + 700 + 30 + 4$$

Here 4 is the least significant digit (LSD) and 2 is the most significant digit (MSD).

In general in a number system with a base or radix r, the digits used are from 0 to r-1 and the number can be represented as

$$N = a_n r^n + a_{n-1} r^{n-1} + \ldots\ldots\ldots\ldots + a_1 r^1 + a_0 r^0 \quad \text{where, for } n = 0,1,2,3,\ldots.(1)$$

r = base or radix of the system.

a = number of digits having values between 0 and r-1

Equation (1) is for all integers and for the fractions (numbers between 0 and 1), the following equation holds.

$$N = a_{-1} r^{-1} + a_{-2} r^{-2} + \ldots\ldots\ldots + a_{-n+1} r^{-n+1} + a_{-n} r^{-n}$$

Thus for decimal fraction 0.7123

$$N = 0.7000 + 0.0100 + 0.0020 + 0.0003$$

where a - 1 = 7

a - 2 = 1

a - 3 = 2

a - 4 = 3

Binary Numbers

The binary number has a radix of 2. As r = 2, only two digits are needed, and these are 0 and 1. Like the decimal system, binary is a positional system, except that each bit position corresponds to a power of 2 instead of a power of 10. In digital systems, the binary number system and other number systems closely related to it are used almost exclusively. Hence, digital systems often provide conversion between decimal and binary numbers. The decimal value of a binary number can be formed by multiplying each power of 2 by either 1 or 0 followed by adding the values together.

Example: The decimal equivalent of the binary number 101010.

$$N = 101010$$

$$= 1 \times 25 + 0 \times 24 + 1 \times 23 + 0 \times 22 + 1 \times 21 + 0 \times 20$$

$$= 43$$

In binary r bits can represent $n = 2^r$ symbols. e.g. 3 bits can represent up to 8 symbols, 4 bits for 16 symbols etc. For N symbols to be represented, the minimum number of bits required is the lowest integer 'r" that satisfies the relationship.

$2^r > N$

E.g. if N = 26, minimum r is 5 since $2^4 = 16 \text{ and } 2^5 = 32$.

Octal Numbers

Digital systems operate only on binary numbers. Since binary numbers are often very long, two shorthand notations, octal and hexadecimal, are used for representing large binary numbers. Octal systems use a base or radix of 8. Thus it has digits from 0 to 7 (r-1). As in the decimal and binary systems, the positional valued of each digit in a sequence of numbers is fixed. Each position in an octal number is a power of 8, and each position is 8 times more significant than the previous position.

Example: The decimal equivalent of the octal number 15.2.

$$N = 15.2_8$$
$$= 1 \text{ x } 8^1 + 5 \text{ x } 8^0 + 2 \text{ x } 8^{-1}$$
$$= 13.25$$

Hexadecimal Numbers

The hexadecimal numbering system has a base of 16. There are 16 symbols. The decimal digits 0 to 9 are used as the first ten digits as in the decimal system, followed by the letters A, B, C, D, E and F, which represent the values 10, 11,12,13,14 and 15 respectively. Table 1 shows the relationship between decimal, binary, octal and hexadecimal number systems.

Decimal	Binary	Octal	Hexadecimal
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	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

Hexadecimal numbers are often used in describing the data in computer memory. A computer memory stores a large number of words, each of which is a standard size collection of bits. An 8-bit word is known as a **Byte.** A hexadecimal digit may be considered as half of a byte. Two hexadecimal digits constitute one byte, the rightmost 4 bits corresponding to half a byte, and the leftmost 4 bits corresponding to the other half of the byte. Often a half-byte is called nibble.

If "word" size is n bits there are 2n possible bit patterns so only 2n possible distinct numbers can be represented. It implies that all possible numbers cannot be represent and some of these bit patterns (half?) to represent negative numbers. The negative numbers are generally represented with sign magnitude i.e. reserve one bit for the sign and the rest of bits are interpreted directly as the number. For example in a 4 bit system, 0000 to 0111 can be used to positive numbers from $+0$ to $+2^{n-1}$ and represent 1000 to 1111 can be used for negative numbers from -0 to -2^{n-1}. The two possible zero's redundant and also it can be seen that such representations are arithmetically costly.

Another way to represent negative numbers are by radix and radix-1 complement (also called r's and (r-1)'s). For example -k is represented as R^n -k. In the case of base 10 and corresponding 10's complement with n=2, 0 to 99 are the possible numbers. In such a system, 0 to 49 is reserved for positive numbers and 50 to 99 are for positive numbers.

Examples:

+3 = +3
-3 = 10^2 -3 = 97

2's complement is a special case of complement representation. The negative number -k is equal to 2 n -k. In 4 bits system, positive numbers 0 to 2^{n-1} is represented by 0000 to 0111 and negative numbers -2^{n-1} to -1 is represented by 1000 to 1111. Such a representation has only one zero and arithmetic is easier. To negate a number complement all bits and add 1

Example:

119 10 = 01110111 2

Complementing bits will result

10001000
+1 *add 1*
10001001
That is 10001001 2 = - 119 $_{10}$

Properties of Two's Complement Numbers

1. X plus the complement of X equals 0.
2. There is one unique 0.
3. Positive numbers have 0 as their leading bit (MSB); while negatives have 1 as their MSB.
4. The range for an n-bit binary number in 2's complement representation is from $-2^{(n-1)}$ to $2^{(n-1)} - 1$
5. The complement of the complement of a number is the original number.
6. Subtraction is done by addition to the 2's complement of the number.

Value of Two's Complement Numbers
For an n-bit 2's complement number the weights of the bits is the same as for unsigned numbers except of the MSB. For the MSB or sign bit, the weight is -2^{n-1}. The value of the n-bit 2's complement number is given by:

$$A_{2's\text{-}complement} = (a^{n-1}) \times (-2^{n-1}) + (a^{n-2}) \times (2^{n-1}) + \ldots (a_1) \times (2^1) + a_0$$

For example, the value of the 4-bit 2's complement number 1011 is given by:

$= 1 \times -2^3 + 0 \times 2^2 + 1 \times 2^1 + 1$
$= -8 + 0 + 2 + 1$
$= -5$

An n-bit 2's complement number can converted to an m-bit number where m>n by appending m-n copies of the sign bit to the left of the number. This process is called sign extension. Example: To convert the 4-bit 2's complement number 1011 to an 8-bit representation, the sign bit (here = 1) must be extended by appending four 1's to left of the number:

$$1011_{\text{4-bit 2's-complement}} = 11111011_{\text{8-bit 2's-complement}}$$

To verify that the value of the 8-bit number is still -5; value of 8-bit number

= -27 + 26 + 25 + 24 + 23 +2 +1

= -128 + 64 + 32 + 16 +8 +2+1

= -128 + 123 = -5

Similar to decimal number addition, two binary numbers are added by adding each pair of bits together with carry propagation. An addition example is illustrated below:

X	190
Y	141
X + Y	331

1 0 1 1 1 1 0 0 0	Carry
1 0 1 1 1 1 1 0	X
+ 1 0 0 0 1 1 0 1	Y
1 0 1 0 0 1 0 1 1	

Similar to addition, two binary numbers are subtracted by subtracting each pair of bits together with borrowing, where needed. For example:

X	229
Y	46
X - Y	183

0 0 1 1 1 1 1 0 0	Borrow
1 1 1 0 0 1 0 1	X
0 0 1 0 1 1 1 0	Y
1 0 1 1 0 1 1 1	X - Y

Two' complement addition/subtraction example

4	0100	-2	1110
- 7	1001	-6	1010
- 3	1101	-8	1 1000

Overflow occurs if signs (MSBs) of both operands are the same and the sign of the result is different. Overflow can also be detected if the carry in the sign position is different from the carry out of the sign position. Ignore carry out from MSB.