Fundamentals of digital systems

Ali EL AZDI

A description or preface here

February 19, 2024

Contents

Chapter 1

Number Systems

1.1 Digital Representations

In a digital representation, a number is represented by an ordered n-tuple:

> The n-tuple is called a **digit vector**, each element is a **digit**
>
> The number of digits n is called the precision of the representation. (careful! leftward indexing)

$$X = (X_{n-1}, X_{n-1}, \ldots, X_0)$$

Each digit is given **a set of values** D_i (eg. For base 10 representation of numbers, $D_i = \{0, 1, 2, \ldots, 9\}$)

The **set size**, the maximum number of representable digit vectors is: $K = \prod_{i=0}^{n-1} |D_i|$

1.1.1 (Non)Redundant Number Systems

A number system is nonredundant if each digit-vector represents a different integer

1.1.2 Weighted Number Systems

The rule of representation if a Weighted (Positional) Number Systems is as follows :

$$\sum_{i=0}^{n-1} X_i W_i$$

where

$$W = (W_{n-1}, W_{n-2}, \ldots, W_0)$$

1.1.3 Radix Systems

When weights are in this format :

$$\begin{cases} W_0 = 1 \\ W_{i+1} = W_i R_i \ \text{ with } 1 \leq i \leq n-1 \end{cases}$$

Also written : $W_0 = 1, \ \prod_{j=0}^{i-1} R_j$

1.1.4 Fixed and Mixed-Radix Number Systems

In a **fixed-radix system**, all elements of the radix-vector have the same value r (*the radix*)

The weight vector in a fixed-radix system is given by:

$$W = \left(r^{n-1}, r^{n-2}, \ldots, r^2, r, 1\right)$$

and the integer x becomes:

$$x = \sum_{i=0}^{n-1} X_i \times r^i$$

In a **mixed-radix system**, the elements of the radix-vector differ

Example: Decimal Number System

The decimal number system has the following characteristics:

- Radix $r = 10$, it's a fixed-radix system.

The weight vector W is defined as:

$$W = \left(10^{n-1}, 10^{n-2}, \ldots, 10^2, 10, 1\right)$$

An integer x in this system is represented by:

$$x = \sum_{i=0}^{n-1} X_i \times 10^i$$

For example:

$$854703 = 8 \times 10^5 + 5 \times 10^4 + 4 \times 10^3 + 7 \times 10^2 + 0 \times 10^1 + 3 \times 10^0$$

Examples of Fixed and Mixed radix systems

Fixed: The base of number systems.

- Decimal – radix 10
- Binary – radix 2
- Octal – radix 8
- Hexadecimal – radix 16

Mixed: An example of a mixed radix representation, such as time:

- Radix-vector $R = (24, 60, 60)$
- Weight-vector $W = (3600, 60, 1)$

1.1.5 Canonical Number Systems

In a **canonical number system**, the set of values for a digit D_i is with $|D_i| = R_i$, the corresponding element of the radix vector

$$D_i = \{0, 1, \ldots, R_i - 1\}$$

Canonical digit sets with fixed radix:

- Decimal: {0, 1, . . . , 9}
- Binary: {0, 1}
- Hexadecimal: {0, 1, 2, . . . , 15}

Range of values of x represented with n fixed-radix-r digits:

$$0 \leq x \leq r^n - 1$$

A system with fixed positive radix r and a canonical set of digit values is called a radix-r conventional number system.

1.2 Binary/Octal/Hexadecimal to/from Decimal

Conversion Table

The hexadecimal system supplements 0-9 digits with the letters A-F.

Remark. Programming languages often use the prefix 0x to denote a hexadecimal number.

Table 1.1: Conversion table up to 15.

Decimal	**Binary**	**Octal**	**Hexadecimal**
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	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

1.2.1 Convertion examples

Binary to Decimal:

To convert a binary number to decimal, multiply each bit by two raised to the power of its position number, starting from zero on the right.

Binary: $\underline{1011}$

Decimal: $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$

Decimal to Binary:

Let's convert the decimal number 25_{10} to binary.

$$\begin{aligned}
25 \div 2 &= 12 \quad \text{remainder 1 (LSB)}\\
12 \div 2 &= 6 \quad \text{remainder 0}\\
6 \div 2 &= 3 \quad \text{remainder 0}\\
3 \div 2 &= 1 \quad \text{remainder 1}\\
1 \div 2 &= 0 \quad \text{remainder 1 (MSB)}
\end{aligned}$$

Thus, the binary representation of 25_{10} is 11001_2 (reading the remainders in reverse).

Personal Remark The trick is always to try to answer the question, what's the biggest power of 2 I need to form the number?. For 157, the biggest power would be $2^7 = 128$, then 128+64 is greater than 157, 128+32 is still greater than 157, $128+16 = 144$, and so on to obtain : $128+16+8+4+1 = 157$ which can be written as $2^7+2^4+2^3+2^2+2^0 = 157$. Written in binary as 10011101_2

Octal to Decimal:

Each octal digit is converted to decimal by multiplying it by eight raised to the power of its position number, starting from zero on the right.

Octal: $\underline{257}$

Decimal: $2 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 = 128 + 40 + 7 = 175$

Decimal to Octal:

To convert the decimal number 93_{10} to octal.

$$\begin{aligned}
93 \div 8 &= 11 \quad \text{remainder 5}\\
11 \div 8 &= 1 \quad \text{remainder 3}\\
1 \div 8 &= 0 \quad \text{remainder 1}
\end{aligned}$$

Thus, the octal representation of 93_{10} is 135_8 (reading the remainders in reverse).

Hexadecimal to Decimal:

To convert the hexadecimal number $1A3_{16}$ to decimal.

Hexadecimal: $\underline{1A3}$

Decimal:

$$\begin{aligned}
&1 \times 16^2 + A \times 16^1 + 3 \times 16^0\\
&1 \times 256 + 10 \times 16 + 3 \times 1\\
&256 + 160 + 3\\
&419
\end{aligned}$$

Here, A in hexadecimal corresponds to 10 in decimal.

Decimal to Hexadecimal:

To convert the decimal number 291_{10} to hexadecimal.

$$\begin{array}{rcll} 291 \div 16 & = & 18 & \text{remainder } 3 \\ 18 \div 16 & = & 1 & \text{remainder } 2 \\ 1 \div 16 & = & 0 & \text{remainder } 1 \end{array}$$

Thus, the hexadecimal representation of 291_{10} is 123_{16} (reading the remainders in reverse).

1.3 Octal/Hexadecimal to/from Binary

Bit-Vector Representation Summary

- Digit-vectors for binary, octal, and hexadecimal systems are represented using bit-vectors. In binary, 0 and 1 are directly represented as 0 and 1.

- In systems like octal or hexadecimal, a digit is a bit-vector of length k, where k is the number of bits needed to represent the base.

  $$k = \log_2(r)$$

  with r the radix of the system (eg. 8 for octal convertion).

- For example, the hexadecimal digit B is represented as the bit-vector 1101 in binary. *We obtain a length 4 bit-vector because the base is 16 and* $\log_2(16) = 4$

Binary to Octal:

To convert a binary number to octal, group every three binary digits into a single octal digit, because $k = \log_2 8 = 3$.

Binary: $\underline{010}\,\underline{000}\,\underline{100}\,\underline{110}$

Octal: $2_8\,0_8\,4_8\,6_8$

Binary to Hexadecimal:

To convert a binary number to hexadecimal, group every four binary digits into a single hexadecimal digit, because $k = \log_2 16 = 4$.

Binary: $\underline{1011}\,\underline{1110}\,\underline{1010}\,\underline{1101}$

Hexadecimal: $B_{16}\,E_{16}\,A_{16}\,D_{16}$

Octal to Hexadecimal:

Convert the octal number to binary, then group the binary digits in sets of four and convert each group to its hexadecimal equivalent.

Octal:	$\underline{257}$
Binary:	010 101 111 (Octal to binary)
Binary grouped:	$\underline{0101}\,\underline{0111}$
Hexadecimal:	5 7 (Binary to hexadecimal)

1.4 Representation of Signed Integers

1.4.1 Sign-Magnitude Representation (SM)

A signed integer x is represented by a pair (x_s, x_m), where x_s is the *sign* and x_m is the *magnitude* (positive integer).

The sign (positive, negative) is represented by a the most significant bit (MSB) of the digit vector:

$0 \rightarrow$ positive

$1 \rightarrow$ negative

The magnitude can be represented as any positive integer. In a conventional radix-r system, the range of n-digit magnitude is:

$$0 \leq x_m \leq r^n - 1$$

- Examples:

$$\begin{aligned}
01010101_2 &= +85_{10} \\
01111111_2 &= +127_{10} \\
00000000_2 &= +0_{10} \\
11010101_2 &= -85_{10} \\
11111111_2 &= -127_{10} \\
10000000_2 &= -0_{10}
\end{aligned}$$

Note: The Sign-and-Magnitude representation is considered a redundant system because both 00000000_2 and 10000000_2 represent zero.

SM consists of an equal number of positive and negative integers.

An n-bit integer in sign-and-magnitude lies within the range *(because of 0's double representation and that MSB is used for the sign)*:

$$[-\left(2^{n-1}-1\right), +\left(2^{n-1}-1\right)]$$

Main disadvantage of SM: complex digital circuits for arithmetic operations (addition, subtraction, etc.).

1.5 True-and-Complement (TC)

1.5.1 Mapping

A signed integer x is represented by a positive integer x_R, C is a positive integer called the *complementation constant.*

$$x_R \equiv x \mod C$$

For $|x| < C$, by the definition of the modulo function, we have:

$$x_R = \begin{cases} x & \text{if } x \geq 0 \quad \text{(True form)} \\ C - |x| = C + x & \text{if } x < 0 \quad \text{(Complement form)} \end{cases}$$

1.5.2 Unambiguous Representation

To have an unambiguous representation, the two regions should not overlap, translating to the condition:

$$\max |x| < \frac{C}{2}$$

1.5.3 Converse Mapping

Converse mapping:

$$x = \begin{cases} x_R & \text{if } x_R < \frac{C}{2} \quad \text{(Positive values)} \\ x_R - C & \text{if } x_R > \frac{C}{2} \quad \text{(Negative values)} \end{cases}$$

When $x_R = \frac{C}{2}$, it is usually assigned to $x = -\frac{C}{2}$.

Asymmetrical representation simplifies sign detection.

1.6 Two's Complement System

This is the True-and-Compelement system with $C = 2^n$, where n is the number of bits used to represent the integer.

Range is asymmetrical:

$$-2^{n-1} \leq x \leq 2^{n-1} - 1$$

The representation of zero is unique.

1.6.1 Sign Detection in Two's Complement System

Since $|x| < C/2$ and assuming the sign is 0 for positive and 1 for negative numbers:

$$\text{sign}(x) = \begin{cases} 0 & \text{if } x_R < C/2 \\ 1 & \text{if } x_R \geq C/2 \end{cases}$$

Therefore, the sign is determined from the most-significant bit:

$$\text{sign}(x) = \begin{cases} 0 & \text{if } x_{n-1} = 0 \\ 1 & \text{if } x_{n-1} = 1 \end{cases} \quad \text{equivalent to} \quad \text{sign}(x) = x_{n-1}$$

1.6.2 Mapping from Bit-Vectors to Values

The value of an integer represented by a bit-vector $b_{n-1}b_{n-2}\dots b_1b_0$ can be universally expressed as:

$$\text{Value} = (-2^{n-1} \cdot b_{n-1}) + \sum_{i=0}^{n-2} b_i \cdot 2^i$$

where b_{n-1} is the MSB (sign bit) and is 0 for non-negative numbers and 1 for negative numbers.