INTERNATIONAL MULTIDISCIPLINARY BOOK SERIES
https://ibseries.com/
IMBS, Vol. 01, No. 1, (2023)
Open Access. Online & Print
ISSN:
Foundations of Computer Science and Information Technology, Edited By: Dr. Sunita Chaudhary

CHAPTER 1:
DECODING NUMBERS: FROM HUMAN COUNTING TO MACHINE LOGIC

Dr. Sunita Chaudhary
Professor, Mec, RU, India

Abstract
This chapter provides a comprehensive understanding of number systems used in computer science. It explains the binary, decimal, octal, and hexadecimal number systems, along with detailed methods for conversion between them. The chapter also covers binary arithmetic operations such as addition, subtraction (using One's and Two's complement), and multiplication with step-by-step solved examples. Diagrams, tables, and practice problems are included to enhance conceptual clarity.

Keywords
Binary, Decimal, Octal, Hexadecimal, Number System, Base, Radix, Conversion, One's Complement, Two's Complement, Binary Arithmetic

1. Introduction

A number system is a systematic way of representing numbers using a fixed set of symbols (digits) and rules. Each system is characterized by its base (or radix), which determines how many unique digits are available and how positional values are calculated.

In computer science, number systems are fundamental because all data—numbers, text, images, and instructions—are ultimately represented in binary (base-2) form. Digital hardware such as processors and memory devices operate using two stable electrical states (ON/OFF), naturally corresponding to binary digits 1 and 0.

1.1 Why Number Systems Matter in Computing

- Enable data representation (integers, real numbers, characters)
- Support arithmetic operations inside the CPU
- Provide compact forms like octal and hexadecimal for readability
- Bridge human-readable (decimal) and machine-level (binary) representations

1.2 Key Terminology

Term	Meaning
Base (Radix)	Number of unique digits in a system
Digit	A single symbol used in a number system
Positional Value	Value of a digit based on its position

ISBN (978-81-685212-8-5)

Most Significant Digit (MSD)	Leftmost digit
Least Significant Digit (LSD)	Rightmost digit

1.3 Positional Number System Concept

- A positional number system assigns value to each digit based on its position and the base.
- General Representation Diagram (ASCII)
- Number = $(d_n\ d_{n-1} \ldots d_1\ d_0 . d_{-1}\ d_{-2})_{(b)}$
- Expanded form:

Value = $d_n \cdot b^n + d_{n-1} \cdot b^{n-1} + \ldots + d_1 \cdot b^1 + d_0 \cdot b^0 + d_{-1} \cdot b^{-1} + d_{-2} \cdot b^{-2}$

Where:

d_i = digit at position i

b = base of the number system

1.4 Place Value Table (Example in Base 10)

Position	3	2	1	0	-1	-2
Power	10^3	10^2	10^1	10^0	10^{-1}	10^{-2}
Value	1000	100	10	1	0.1	0.01

Example: $(472.35)_{10} = 4\times10^2 + 7\times10^1 + 2\times10^0 + 3\times10^{-1} + 5\times10^{-2}$

1.5 Comparison of Positional Systems

System	Base	Example	Expansion
Decimal	10	25_{10}	$2\times10^1 + 5\times10^0$
Binary	2	1011_2	$1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$
Octal	8	17_8	$1\times8^1 + 7\times8^0$
Hexadecimal	16	$A3_{16}$	$10\times16^1 + 3\times16^0$

1.6 Visual Interpretation (Binary Example)

Bit Position	3	2	1	0
Weight	8	4	2	1
Digit	1	0	1	1
Contribution	8	0	2	1

Total = $8 + 0 + 2 + 1 = 11_{10}$

1.7 Fractional Representation (Binary)

Position	-1	-2	-3
Weight	1/2	1/4	1/8

Example: $(0.101)_2 = 1\times1/2 + 0\times1/4 + 1\times1/8 = 0.625_{10}$

1.8 Summary of Concepts

- Every number system is defined by a base and digits
- Values are determined using powers of the base
- Binary is the foundation of digital systems
- Understanding positional notation is essential for conversion and arithmetic operations

2. Types of Number Systems

Number systems are categorized based on their base (radix), which determines how many unique symbols (digits) are available and how positional values are computed. In computing, four primary number systems are widely used—decimal, binary, octal, and hexadecimal. Each serves a specific purpose: decimal is human-friendly, binary is

machine-native, while octal and hexadecimal act as compact representations of binary for easier readability and debugging.

Tabular Overview

Number System	Base	Digits Used
Decimal	10	0–9
Binary	2	0, 1
Octal	8	0–7
Hexadecimal	16	0–9, A–F

2.1 Decimal Number System (Base 10)

The decimal system uses ten digits (0–9) and is the standard system for everyday arithmetic. Each position represents a power of 10.

Example: $(538)_{10} = 5\times10^2 + 3\times10^1 + 8\times10^0 = 500 + 30 + 8$

2.2 Binary Number System (Base 2)

The binary system uses only two digits (0 and 1). It is fundamental to digital electronics because hardware circuits operate using two states (ON/OFF).

Example: $(1011)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 8 + 0 + 2 + 1 = 11_{10}$

2.3 Octal Number System (Base 8)

The octal system uses digits from 0 to 7. It provides a more compact representation of binary numbers, where each octal digit corresponds to three binary bits.

Example: $(27)_8 = 2\times8^1 + 7\times8^0 = 16 + 7 = 23_{10}$

2.4 Hexadecimal Number System (Base 16)

The hexadecimal system uses sixteen symbols: 0–9 and A–F (where A=10, B=11, ..., F=15). It is widely used in programming, memory addressing, and color codes.

Example: $(AF)_{16} = 10\times16^1 + 15\times16^0 = 160 + 15 = 175_{10}$

2.5 Comparative Insight

- Decimal is intuitive for humans but inefficient for machines.
- Binary is ideal for hardware but verbose for humans.
- Octal and hexadecimal provide compact and readable representations of binary data.

Number System	Base	Digits Used
Decimal	10	0–9
Binary	2	0, 1
Octal	8	0–7
Hexadecimal	16	0–9, A–F

3. Decimal Number System (Base 10)

The decimal number system is the most widely used number system in everyday life. It is a base-10 positional number system, meaning it uses ten digits ranging from 0 to 9. The value of each digit in a decimal number depends not only on the digit itself but also on its position, which is associated with powers of 10.

In a positional system, each digit is multiplied by a corresponding power of the base (10). The powers increase from right to left, starting from 10^0at the rightmost position. This positional weighting allows numbers to be represented efficiently and unambiguously.

Structure Diagram

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

In this representation, the digit 3 is in the hundreds place, 4 is in the tens place, and 5 is in the units place.

Example

The number $(345)_{10}$ can be expanded using positional values as follows:

$(345)_{10} = 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$

$= 3 \times 100 + 4 \times 10 + 5 \times 1$

$= 300 + 40 + 5$

$= 345$

Thus, the decimal system expresses numbers as a sum of digits multiplied by powers of 10, making it intuitive and suitable for human calculations.

4. Binary Number System (Base 2)

The binary number system is a base-2 positional number system that uses only two digits: 0 and 1. It is the most fundamental number system in computer science because all digital systems, including computers and electronic circuits, operate using two discrete states—commonly represented as OFF (0) and ON (1). Due to this simplicity and reliability, binary is used to represent all forms of data in computing systems.

In the binary system, each position in a number corresponds to a power of 2, increasing from right to left. The rightmost digit represents 2^0, the next represents 2^1, and so on. The value of a binary number is obtained by multiplying each digit by its corresponding power of 2 and summing the results.

Binary Place Value Chart

Position	4	3	2	1	0
Value	16	8	4	2	1

This chart shows how each position contributes to the total value of a binary number.

Example

Consider the binary number $(1011)_2$. Its value can be calculated as follows:

$(1011)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 8 + 0 + 2 + 1$

$= 11_{10}$

Thus, the binary number 1011 is equivalent to 11 in the decimal system.

In summary, the binary number system forms the backbone of modern computing, enabling efficient data processing and storage through simple two-state logic.

5. Octal Number System (Base 8)

Key Idea:

Each octal digit represents 3 binary digits.

6. Hexadecimal Number System (Base 16)

The hexadecimal number system is a base-16 positional number system that uses sixteen distinct symbols to represent values. These include the digits 0 to 9 and the alphabetic characters A to F, where A represents 10, B represents 11, C represents 12, D represents 13, E represents 14, and F represents 15. This system is widely used in computer science because it provides a compact and human-readable representation of binary numbers, with each hexadecimal digit corresponding exactly to four binary bits (a nibble).

In the hexadecimal system, the positional values are based on powers of 16. Similar to other positional number systems, each digit is multiplied by the corresponding power of 16 depending on its position, starting from 16^0 on the rightmost side. This allows large binary numbers to be represented in a much shorter and more manageable form, which is particularly useful in programming, memory addressing, and debugging.

Hexadecimal Digit Table

Decimal	Hex
10	A
11	B
12	C

Decimal	Hex
13	D
14	E
15	F

Example

Consider the hexadecimal number $(A3)_{16}$. Its decimal equivalent can be calculated as follows:

$(A3)_{16} = A\times16^1 + 3\times16^0$

$= 10\times16 + 3\times1$

$= 160 + 3$

$= 163_{10}$

Thus, the hexadecimal number A3 represents the decimal value 163.

In summary, the hexadecimal number system serves as an efficient bridge between binary and decimal systems, making it an essential tool for programmers and computer engineers.

7. Conversion Techniques

Conversion between number systems is an essential skill in computer science, as it enables the transformation of data between human-readable formats (decimal) and machine-level representations (binary, octal, and hexadecimal). Each type of conversion follows a systematic procedure based on the base (radix) of the number system involved.

7.1 Decimal to Binary Conversion

The conversion from decimal (base 10) to binary (base 2) is performed using a repeated division method. In this method, the given decimal number is divided successively by 2, and the remainders are recorded at each step. The process continues until the quotient becomes zero. The binary equivalent is then obtained by reading the remainders from bottom to top.

Flow Process

Step 1: Divide the decimal number by 2
Step 2: Record the remainder (0 or 1)
Step 3: Divide the quotient again by 2
Step 4: Repeat until quotient = 0
Step 5: Write remainders in reverse order

Example

Convert 13_{10} to binary:

$13 \div 2 = 6$ remainder 1

$6 \div 2 = 3$ remainder 0

$3 \div 2 = 1$ remainder 1

$1 \div 2 = 0$ remainder 1

Reading from bottom to top:

$13_{10} = 1101_2$

7.2 Binary to Decimal Conversion

The conversion from binary to decimal is performed by multiplying each binary digit by its corresponding power of 2, based on its position, and then summing all the values.

Example

Convert $(1010)_2$ to decimal:

$(1010)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 8 + 0 + 2 + 0$

$= 10_{10}$

This method highlights the positional nature of the binary number system.

7.3 Octal to Binary Conversion

The conversion from octal (base 8) to binary (base 2) is straightforward because each octal digit corresponds exactly to three binary digits. Therefore, each octal digit is replaced by its equivalent 3-bit binary representation.

Octal to Binary Mapping Table

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

Example
Convert $(57)_8$ to binary:
5 → 101
7 → 111
Therefore,
$(57)_8 = (101111)_2$

7.4 Binary to Hexadecimal Conversion

The conversion from binary to hexadecimal (base 16) involves grouping binary digits into sets of four bits, starting from the right. Each group is then replaced by its corresponding hexadecimal digit.

Conversion Rule

- Group binary digits into 4-bit blocks
- Add leading zeros if necessary
- Convert each group into its hexadecimal equivalent

Example
Convert $(11010110)_2$ to hexadecimal:
Group into 4 bits:
1101 0110
Convert each group:
1101 → D
0110 → 6
Thus,
$(11010110)_2 = (D6)_{16}$

Summary of Conversion Methods
Decimal ↔ Binary: Division and positional expansion
Octal ↔ Binary: 3-bit grouping
Binary ↔ Hexadecimal: 4-bit grouping

These conversion techniques are widely used in programming, digital electronics, and data representation to ensure efficient communication between human users and computer systems.

8. Binary Arithmetic

Binary arithmetic refers to performing mathematical operations such as addition, subtraction, and multiplication using binary numbers (base 2). These operations follow rules similar to decimal arithmetic but are simplified due to the use of only two digits: 0 and 1. Binary arithmetic is fundamental to computer operations, as all calculations inside a computer system are carried out using binary logic.

8.1 Binary Addition

Binary addition is the simplest form of binary arithmetic and follows a set of basic rules based on combinations of binary digits and carry values. When adding binary numbers, a carry is generated whenever the sum exceeds 1.

Truth Table for Binary Addition

A	B	Carry In	Sum	Carry Out
0	0	0	0	0
0	1	0	1	0
1	1	0	0	1

These rules can be extended when a carry-in is present during multi-bit addition.

Example
Consider the addition of two binary numbers:

```
  1011
+ 1101
--------
 11000
```

Step-by-step, the addition proceeds from right to left, carrying over whenever the sum of bits exceeds 1. The final result is $(11000)_2$.

8.2 Binary Subtraction

Binary subtraction can be performed using direct borrowing, but in computer systems, it is more efficiently handled using complement methods, namely One's complement and Two's complement.

One's Complement
The One's complement of a binary number is obtained by inverting all bits (changing 0 to 1 and 1 to 0).

Example:
Original number: 0101
One's complement: 1010

Two's Complement Method
The Two's complement is widely used in computers for subtraction. It is obtained by:
Step 1: Taking the One's complement
Step 2: Adding 1 to the result

Example
Subtract 5 from 9 using Two's complement:
9 = 1001
5 = 0101

Step 1: One's complement of 5 → 1010
Step 2: Add 1 → 1011 (Two's complement)

Now add:

```
  1001
+ 1011
--------
 10100
```

Discard the overflow carry:
Result = 0100 (which is 4 in decimal)
Thus, 9 − 5 = 4.

8.3 Binary Multiplication

Binary multiplication is similar to decimal multiplication but uses only binary digits (0 and 1). Each digit of the multiplier is multiplied with the multiplicand, and intermediate results are shifted and added accordingly.

Example

Multiply $(101)_2$ by $(11)_2$:

```
  101
×  11
--------
  101
+1010
--------
 1111
```

Step-by-step:
Multiply 101 by 1 → 101
Multiply 101 by the next 1 (shift left) → 1010
Add the partial results
Final result = $(1111)_2$

Summary of Binary Arithmetic

- Binary addition follows simple carry rules
- Subtraction is efficiently performed using Two's complement
- Multiplication involves shifting and adding partial products
- Binary arithmetic forms the core of all computations performed by digital systems and processors.

9. Applications of Number Systems

Number systems play a crucial role in various domains of computer science and digital technology. In digital circuits, binary numbers are used to represent electrical signals in the form of high (1) and low (0) states, forming the basis of logic gates and processors. In programming, number systems such as binary, octal, and hexadecimal are used for low-level coding, memory addressing, and debugging purposes.

Furthermore, number systems are essential in data storage, where all types of data—including text, images, audio, and video—are encoded in binary form for efficient storage and retrieval. In the field of networking, number systems are used in IP addressing, subnetting, and data transmission, enabling communication between devices over networks. Thus, number systems form the foundation of modern computing and communication systems.

10. Advantages of Binary System

The binary number system offers several advantages that make it ideal for use in digital systems. One of its primary benefits is its simple implementation, as it requires only two states (0 and 1), which can be easily represented using electronic components. Additionally, binary systems are highly reliable in electronic environments, as they are less susceptible to noise and signal distortion compared to multi-level systems.

Another key advantage is that binary arithmetic is efficient for computation, allowing computers to perform calculations quickly and accurately. These characteristics make binary the most suitable and universally adopted number system in computing.

11. Summary

In conclusion, number systems form the backbone of computer science and digital technology. This chapter has explored various number systems, including decimal, binary, octal, and hexadecimal, along with their structures, conversions, and arithmetic operations. A strong understanding of these concepts—especially binary arithmetic and conversion techniques—is essential for comprehending how computers represent, process, and manipulate data at the fundamental level.

References

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT Press.
2. Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). Concrete mathematics: A foundation for computer science (2nd ed.). Addison-Wesley.
3. Knuth, D. E. (1997). The art of computer programming (Vol. 1: Fundamental algorithms, 3rd ed.). Addison-Wesley.
4. Mano, M. M., & Ciletti, M. D. (2017). Digital design (6th ed.). Pearson.
5. Patterson, D. A., & Hennessy, J. L. (2017). Computer organization and design: The hardware/software

interface (5th ed.). Morgan Kaufmann.
6. Latif, S., Ullah, R., & Jan, H. (2011). A step towards an easy interconversion of various number systems. arXiv preprint arXiv:1107.1663.
7. Serbenyuk, S. (2019). Modeling numeral systems. arXiv preprint arXiv:1907.10534.
8. Rompis, L. (2021). Decimal-binary conversion methods. International Journal of Computer Applications, 174(5), 1–6.
9. Lande, D. R. (2014). Development of the binary number system. The Mathematics Enthusiast, 11(3), 1–12.
10. ScienceDirect. (n.d.). Binary number system. Retrieved from https://www.sciencedirect.com
11. ACM Digital Library. (n.d.). Number systems in computing. Retrieved from https://dl.acm.org
12. Binary systems form the basis of digital computation (Patterson & Hennessy, 2017).
13. Conversion between number systems is essential in computing (Latif et al., 2011).
14. Positional number systems depend on base powers (Knuth, 1997).