CHAPTER 1

Representing Natural Numbers

Natural numbers are positive whole numbers, such as 1, 2, 3, and so on. -5 is not a natural number. π is not a natural number. $\frac{1}{2}$ is not a natural number.

1.1 Base-10 (Decimal)

You are used to seeing the natural numbers represented in a **base-10** *Hindu-Arabic* or the *Decimal* numeral system. That is, when you see 2531 you think "2 thousands, 5 hundreds, 3 tens, and 1 one." Rewritten, this is:

$$2 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 1 \times 10^0$$

The *radix* is the amount of number of unique values a base can have before increasing the amount of digits. For example, values 0, 1, 2, $\cdots$, 8, 9 are all the values for base-10. Once 1 is added added to 9, we need to increase the amount of digits in a number.

In any Hindu-Arabic system, the location of the digits is meaningful: 101 is different from 110; the position of a number indicates it's magnitude. Here are those numbers in Roman numerals: CI and CX. Roman numerals didn't have a symbol for zero at all.

The Hindu-Arabic system gave us really straightforward algorithms for addition and multiplication. For addition, you memorized the following table:

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15 16	17	18	

When you multiplied two number together, you simply multiplied each pair of digits. 254×26 might look like this:

		2	5	4	
		×	2	6	
			2	4	6×4
		3	0		6×5
	1	2			6×2
			8		2×4
	1	0			2×5
+	4				2×2
6	6	0	4		

For multiplication, you memorized this table:

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
9	0	9	18	27	36	45	54	63	72	81

1.2 Base-2 (Binary)

Binary, on the other hand, has only 2 digits it can have: 0 or 1. This kind of number system is often used for electrical and computer systems because the values can only be **on** or **off**.

Each digit in a binary number represents a power of 2, starting from the rightmost digit (the least significant bit, or LSB). For example:

$$\cdots + 2^3 + 2^2 + 2^1 + 2^0$$

For example,

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

[1]

[1] The subscripts represent the radix or base.

Because the radix is 2, the numbers increase differently:

$$
\begin{aligned}
&0 : 0 \\
&1 : 1 \\
&2 : 10 \\
&3 : 11 \\
&4 : 100 \rightarrow (2^2 \cdot 1) + 0 + 0 = 4 \\
&5 : 101 \\
&6 : 110 \\
&7 : 111 \\
&8 : 1000 \\
&\dots \\
&15 : 1111 \cdots
\end{aligned}
$$

Conversion from Decimal to Binary

To convert 13_{10} to binary, repeatedly divide by 2 and record the remainders until the divisor reaches 0 or 1:

$13 \div 2 = 6$ remainder $16 \div 2 = 3$ remainder $03 \div 2 = 1$ remainder $11 \div 2 = 0$ remainder 1

Reading from bottom to top, $13_{10} = 1101_2$.

1.2.1 Bits and Bytes

A singular binary digit (0 or 1) is called a *bit*. A lightbulb, switch, or any sort of *Boolean* value (true or false) can be represented as a bit. A single binary digit is called a *bit*. Eight bits form a *byte*, which is a common unit of storage in computer systems:

$$1 \text{ byte} = 8 \text{ bits}$$

For example, the binary sequence 01001000 represents one byte of data.

We won't go into the depths of it here, but here are the basics of computer architecture. There are codes of 4-bit sequences that form up the *memory*. This can form the sections of it into different data types:

Common Data Types in Memory

Data Type	Size	Bit Length	Typical Use
Character	1 byte	8 bits	ASCII characters, small integers
Integer	4 bytes	32 bits	Whole numbers
Double	8 bytes	64 bits	Decimal (floating-point) values

Strings are formed by listing characters in consecutive memory addresses such that they are marked by a starting memory address and ending memory address.

8-bit binary numbers can be used to signify positive or negative numbers. The most significant bit (MSB) will alter sign, causing the range to be $[-2^7,\ 2^7 - 1]$. If the byte is negative (in terms of bits), invert each bit and add 1 to the inverted result. The other way is to multiply the MSB by negative one, and add until the value is reached, as seen below.

$$-85_{10} = -1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 10101011$$

Binary can be easy converted to *Hexadecimal*.

1.3 Base-16 (Hexadecimal)

Hexadecimal (from the latin *hexa* and *deca* for 16) is a representation of number with a radix of 16. Digits are represented through numeric numbers 0-9 and *A-F*.

Counting to 16 in hex goes as follows:

1
2
3
...
9
A (decimal value 10)
B (decimal value 11)
C (decimal value 12)
D (decimal value 13)
E (decimal value 14)
F (decimal value 15)

Converting between binary and hexadecimal is easier because the hex radix ($16 = 2^4$) is a power of the binary radix (2^1).

It takes 4 binary digits to represent 1 hex digit: $15 = \text{F}$

For example,

$$0x47 = 0100\ 0111_2$$
$$0xBD = 1011\ 1101_2$$

1.3.1 Hex Color Codes

Hex colors are represented by 6 hexadecimal digits, grouped as two each for red, green, and blue (RGB). Each pair ranges from \$00\$ (0 in decimal) to \$FF\$ (255 in decimal), allowing \$256\$ shades per color channel. For example:

$$\#FF0000 = \text{Red}, \quad \#00FF00 = \text{Green}, \quad \#0000FF = \text{Blue}$$

By mixing values, you can create millions of unique colors, e.g.:

$$\#FFA500 = \text{Orange}, \quad \#800080 = \text{Purple}, \quad \#FFFFFF = \text{White}, \quad \#000000 = \text{Black}$$

There are $256^3 = 16{,}777{,}216$ possible color combinations in the RGB hex system. The higher the red, green, or blue values are (closer to 255), the brighter the component becomes and the overall color shifts closer to white. Conversely, the lower the values are (closer to 0), the darker the component becomes and the overall color shifts closer to black.

1.4 Base-8 (Octal)

Octal is another representation of radix 8. Digits range from 0 to 7, and the number of digits increase the threshold is passed.

Just as in the above systems, octal increases by the radix —— powers of 8. For example,

$$135_8 = (1 \times 8^2) + (3 \times 8^1) + (5 \times 8^0) = 64 + 24 + 5 = 93_{10}$$

1.4.1 Why Octal Matters

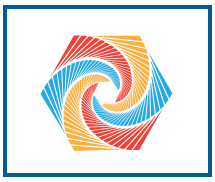
Octal was historically important in computing because early computers often worked with word sizes that were multiples of 3 bits (such as 12, 24, or 36). Since each octal digit corresponds exactly to 3 binary digits, it was a convenient shorthand for binary values before hexadecimal became more widespread. For example:

$$110101_2 = 65_8$$

Even though modern systems mostly use hexadecimal, octal is still used in some contexts, such as Unix file permissions (e.g., `chmod 755`).

Appendix A

Answers to Exercises



Index