**DATA REPRESENTATION**

Computers perform four basic functions: inputting data, processing data, displaying the results using output devices and storing the results for subsequent use. Computer hardware, especially the system unit, is involved in all of these functions.

The term performance refers to how fast a computer can obtain, process, display and store data. To communicate knowledgeably with others about computer hardware capabilities, it is important to know the terminology that is used to describe how computers represent data as well as how much data computers can transfer or store.

Computer performance is often considered to be the same as the speed of a computer’s processor; however, the processor’s capabilities are only part of the picture.

Computers can’t do anything without data to work with. For a computer to work with data, the data must be represented by digits inside the computer. We all use decimal numbers - which consist of 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9- to count. Computers count with binary numbers (also called binary digits or bits for short), which consist of only two digits: 0 and 1. The digit 0 represents the electronic state of off (absence of an electronic charge). The digit 1 represents the electronic state of on (presence of an electronic charge). A bit is the smallest unit of information that a computer can work with. It is like a switch: it has only two possible states and it is always in one or the other. If you have one light switch, then the switch is either on or off. If you have two light switches, then you have four possibilities: both switches are on, both switches are off, the first switch is on and the second is off, or the first switch is off and the second switch is on. Three switches allow for eight possibilities, and so on, up to eight switches, which result in 256 possible combinations.

A byte consists of eight bits and represents one unit of storage. Because it takes eight bits (on/off switches) to make a byte, and eight bits result in 256 possible on/off combinations, you will see the number 256 appearing behind the scenes in many computer functions and applications. A single byte usually represents one character of data, such as the essential numbers (0-9), the basic letters of the alphabet, and the most common punctuation symbols. For this reason, you can use the byte as a baseline for understanding just how much information a computer is storing. For example, a typical college essay contains 250 words per page, and each word contains, on average, 5.5 characters. Therefore, the page contains approximately 1,375 characters. In other words, you need about 1,375 bytes of storage for one page of a college paper.

Bits (1s and 0s) are commonly used for measuring the data transfer rate of computer communications devices such as modems. To describe rapid data transfer rates, the measurements kilobits per second (Kbps), megabits per second (Mbps) and gigabits per second (Gbps) are used. These rates correspond roughly to 1 thousand, 1 million and 1 billion bits per second.

Bytes are commonly used to measure data storage. The measurements kilobyte (K or KB), megabyte (M or MB), gigabyte (G or GB), and terabyte (T or TB) are used to describe the amount of data the computer is managing either in memory or in long term storage on disk.

**Numeral systems commonly used in computer operations**

**Binary arithmetic:**

The binary numeral system, or base-2 number system, represents numeric values using two symbols, 0 and 1. More specifically, the usual base-2 system is a positional notation with a radix of 2. Owing to its straightforward implementation in digital electronic circuitry using logic gates, the binary system is used internally by all modern computers.

In decimal systems we all know that, when we add a single digit (0 to 9) to another digit, we get the rules of 0+0= 0, 0+1=1, right up to 9+9=18.

In binary systems, we perform such additions aligning the digits up and carrying any values into the next column. Thus, 9+9= 8, with one carried into the next column to be added with that column.

Actually, binary addition is simpler as it only involves two values for each digit, but, obviously, we are now more accustomed to decimal addition. There are, thus, four possible combinations when adding two binary digits together (with a carry from the previous column):

**0+0= 0 1+0= 1 1+1=10 1+1+1= 11**

We can perform a binary addition as we do in decimal addition, by lining up the two values with the least significant bit at the right-hand side, and the most significant bit at the left-hand side. Any gaps in bits on the left-hand side are replaced with zero values (as we do with decimal addition, but we typically do not include preceding zeros). For example:

**0010001**

**0001111**

|  |
| --- |
| **0100000** |
| **11111** |

Binary numbers are difficult to work with because many digits are required to represent even a small number. In binary notation, 2 is the base. For example, when you enter the decimal number 14 into your computer, the binary representation is 1110. In addition, it’s time-consuming for computers to translate binary numbers into their decimal equivalents. As a base-2 numbering system is used, to determine the decimal equivalent of a binary number each column is represented by two raised to the power of 0, 1, 2 and so on. For example, the decimal equivalents of 1000 0001 and 0101 0011 are:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | Decimal |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 129 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 83 |

Thus, 01010011 gives: (0x128) + (1x64) + (0x32)+ (1x16)+ (ox8)+ (ox4) + (1x2)+ (1x1)= 83.

**Typical groupings of bits are:**

* nibble: is a group of four bits. A nibble gives 16 (2 la puterea 4) different combinations of ON/OFF, from 0000 to 1111.
* byte: is a group of eight bits. A byte gives 256 (2 la puterea 8) different combinations of On/Off, from 0000 0000 to 1111 1111.
* word: is a group of 16 bits (2 bytes). A word gives 65,536 (2 la puterea 16) different combinations of On/Off, from 0000000000000000 to 1111111111111111.
* long word: is a group of 32 bits (4 bytes). A long word gives 4,294,967,296 (2 la puterea 32) different combinations of ON/OFF.

Because binary numbers are difficult to work with (often, it is difficult to differentiate binary numbers from decimal numbers, as one hundred and one can be seen as 101 in binary, and vice-versa), in programming it is also common to translate binary numbers into hexadecimal (hex, for short) numbers, using the numbers 0 through 9 and the letters A through F, or into octal numbers, using the numbers 0 through 7 and 10-17. These notations are used to write software, as a shorthand way of representing long strings of bits. Thus, the string 01000001 can be represented as octal 101, decimal 65 and hexadecimal 41. (A typical convention is to use a proceeding b for binary numbers, for example 010101111010b.)

Other example may be given: the letter K is represented as the lengthy binary number 01001011 and then quickly translated into 4B in hex, as shown in the table below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decimal, Binary, Octal and Hexadecimal Numbers | | | | | | | | | | | | | | | | |
| Decimal Number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Binary Number | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| Octal  Number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| Hexadecimal Number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |

**Hexadecimal:**

In mathematics and computer science, hexadecimal (also base 16, or hex) is a positional numeral system with a radix, or base, of 16. It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a through f) to represent values ten to fifteen. For example, the hexadecimal number 2AF3 is equal, in decimal, to (2 × 163) + (10 × 162) + (15 × 161) + (3 × 160), or 10,995.

Each hexadecimal digit represents four binary digits (bits) (also called a "nibble"), and the primary use of hexadecimal notation is as a human-friendly representation of binary coded values in computing and digital electronics. For example, byte values can range from 0 to 255 (decimal) but may be more conveniently represented as two hexadecimal digits in the range 00 through FF.

**Hexadecimal is used for the following:**

To specify a memory address. Typically memory addresses in a computer are specified with their binary address, thus there must be a method to display this in a form which the user can easily convert to and from.

To display or set the value of a variable. Sometimes the actual binary contents of a value need to be interrogated or set, thus there must be a form in which the user can easily read and convert it into a binary form.

To specify network addresses. A network address of a computer often needs to be converted into a form which the computer understands, thus there must be a conversion between decimal and binary, or vice versa.

These problems are solved by either converting between binary and decimal, or between binary and hexadecimal (base-16) or octal (base-8). Without the aid of a calculator, the conversion between binary and decimal is relatively difficult for large binary numbers, but hexadecimal and octal conversion makes it easier, as they allow the binary digits to be split into groups of four (for hexadecimal) or three (for octal), and then converted. Hexadecimal is the conversion most often used to specify a memory address or in defining the contents of a memory address.

**Octal:**

The octal numeral system, or oct for short, is the base-8 number system, and uses the digits 0 to 7. Numerals can be made from binary numerals by grouping consecutive binary digits into groups of three (starting from the right). For example, the binary representation for decimal 74 is 1001010, which can be grouped into (00)1 001 010 — so the octal representation is 112.

In decimal systems each decimal place is a base of 10. For example:

\mathbf{74}_{10} = \mathbf{7} \times 10^1 + \mathbf{4} \times  10^0

In octal numerals each place is a power with base 8. For example:

\mathbf{112}_8 = \mathbf{1} \times  8^2 + \mathbf{1} \times  8^1 + \mathbf{2} \times  8^0 

By performing the calculation above in the familiar decimal system we see why 112 in octal is equal to 64+8+2 = 74 in decimal.

Octal is sometimes used in computing instead of hexadecimal.

\*

To represent and process numbers that have fractional parts (such as 1.25) or extremely large numbers, computers use floating-point notation. The term “floating point” suggests how this notation system works: no fixed number of digits is before or after the decimal point, so the computer can work with very large as well as very small numbers. Floating-point notation requires special processing circuitry, which is generally provided by the floating-point unit (FPU). Almost a standard in the circuitry of today’s microprocessors, on older computers the FPU was sometimes a separate chip called the math coprocessor.

It would be difficult to use computers if they just spat out numbers at us. Fortunately, thanks this character code, we can understand computer output.

Character code translates between the computer’s numeric world and the letters, numbers and symbols called characters that we’re accustomed to using. Computers can recognize several different character codes.

The most widely used character code is the American Standard Code for Information Interchange (ASCII), pronounced “ask-ee”, which is used on minicomputers, personal computers and computers that make information available over the Internet. IBM mainframe computers and some other systems use a different code, Extended Binary Coded Decimal Interchaange Code (EBCDIC), pronounced “ebb-see-dic”.

Although ASCII and EBCDIC contain some foreign language symbols, both are clearly insufficient for a global computer market. Unicode can represent many, if not most, of the world”s languages.

**Give the synonyms of the following:** *performace, knowledgeable, capability, storage, to measure, notation, straightforward, to implement, obviously, accustomed.*

**Give the antonyms of the following:** *absence, on, logical, internally, addition, significant, accustomed, right-hand side,continuous.*

**Translate into English:**

Un sistem binar este, în general vorbind, un sistem bazat pe 2 elemente. Sistemul de numeraţie binar foloseşte drept baza numărul 2. În sistemul (de numeraţie) binar există doar două cifre posibile, 0 şi 1. Conform definiţiei lui Claude Shannon, o cifră binară conţine cantitatea de informatie de 1 bit. Sistemul binar este, în acelaşi timp, cel mai natural mod de stocare a informaţiei în calculatoare, deoarece 1 bit găzduieşte unitatea elementară de informaţie: valoarea bitului, 0 sau 1.

În orice sistem informatic, conform definiţiei lui Turing, este nevoie de o memorie fiabilă. Cea mai fiabilă metodă de prelucrare şi stocare a datelor se bazează pe sistemul binar: "celula este magnetizată sau nu este magnetizată", "trece curent sau nu trece curent", "cartela este perforată sau nu este perforată" etc. Alternativă ar fi fost utilizarea sistemului de numeraţie în baza 3, ceea ce ar fi necesitat elemente cu trei stări stabile, ducând la o logică trivalentă de genul "senzorul nu este excitat, senzorul este puţin excitat, senzorul este excitat", ceea ce, deşi induce un nivel mai bun de discernere a informaţiei, reduce, cel puţin în principiu, fiabilitatea potenţială a sistemului.

Datorită uşurinţei implementării sistemului binar în circuitele electronice, el se foloseşte practic la toate calculatoarele moderne. Sistemul de numeraţie binar a început să fie folosit în mod implicit încă din cele mai vechi timpuri, odată cu apariţia logicii bivalente: odată definite noţiunile de "propoziţie adevărată" şi "propoziţie falsă", operaţiile care lucreează cu aceste noţiuni sunt operaţii de tip binar. Folosirea sistemului binar s-a răspândit însă cel mai mult abia recent, odată cu apariţia sistemelor informatice, începând de la cele mai rudimentare şi până la cele curente.