# Chapter 1. Computer Numerical System

## 1.1 Concept of coding

Encoding is the process of converting information from its original human-readable format to a format that is more suitable for storage, transmission and processing by a computer. The reverse process is called decoding.

The main purposes of coding include:

1. Ease of technical implementation;

2. Convenience of information perception;

3. High speed of transmission and processing of information messages;

4. Cost-effectiveness achieved by reducing redundancy in messages;

5. Reliability, ensuring information protection from accidental distortion;

6. Security of information and protection from unauthorized access.

At different stages of working with information, different goals may arise, which leads to the need for repeated recoding. It should be taken into account that these goals may be mutually contradictory.

## 1.2 Storing numbers in computer memory

Numbers are stored in electronic computers (computers) using digital binary code. This is due to the fact that the electronic elements of a computer can only be in two stable states - 0 or 1. The basic unit of information is a bit, and 8 bits form 1 byte of memory. Each byte has its own sequence number, which is called an address.

Memory elements are combined into cells called machine words [34]. The address of a machine word is determined by the address of its low byte. The amount of information that can be represented in a computer is limited by its memory capacity, which affects the accuracy of the representation of numerical information, depending on the architecture of the system.

For example, if a computer has 2 kilobytes of RAM, this corresponds to 2048 bytes. It is important to note that in computer terminology, 1 kilobyte (KB) is equal to 1024 bytes because the binary number system is used. So, 2 KB = 2 × × 1024 = 2048 bytes. Since the numbering starts from zero, the address of the last byte will be 2047.

To consolidate the material, solutions to the problems are given.

Task:

The computer's RAM contains 163,840 machine words, which is equivalent to 0.625 megabytes. It is necessary to determine the length of the machine word in bytes.

Solution:

1. You need to convert megabytes to bytes:

0.625 MB = 0.625 × 1024 KB = 640 KB

640 KB = 640 × 1024 B = 655360 B

2. To calculate the length of a machine word, do the following:

Machine word length = Total number of bytes/Number of machine words

Machine word length = 655360 B / 163840 words = 4 B

Answer:

The length of a machine word is 4 bytes.

Task:

The amount of RAM is 1 megabyte, and the address of the last machine word is 1048574. It is necessary to determine the size of the machine word.

Solution:

1. Converting memory size to bytes:

The amount of RAM is 1 megabyte. Let's convert this value into bytes:

1 MB = 1024 KB = 1024 × 1024 B = 1048576 B

2. Determining the address of the last byte:

Given that byte numbering starts at 0, the address of the last byte will be one less than the total number of bytes:

Last byte address = 1048576 - 1 = 1048575

3. Analysis of the address of the last machine word:

The address of the last machine word is 1048574, which implies that the last machine word occupies addresses 1048574 and 1048575.

4. Definition of machine word size:

To find out how many bytes the last machine word occupies, consider two addresses:

The last machine word covers addresses 1048574 and 1048575. Therefore, the machine word size is 2 bytes.

Answer:

The size of a machine word is 2 bytes.

Task:

Determine the address of the last byte of 4 kilobytes (KB) of RAM.

Solution:

1. Converting the amount of RAM into bytes:

The amount of RAM is 4 kilobytes:

4 KB = 4 × 1024 B = 4096 B

2. Determining the address of the last byte:

Since addressing starts from zero, the address of the last byte can be determined as follows:

Last byte address = Total memory – 1 = 4096 – 1 = 4095

Answer:

The address of the last byte of 4 kilobyte RAM is 4095.

## 1.3 Computer data formats

Data formats in computers play a key role in processing and storing information. The main data formats are logical codes, fixed point numbers and floating point numbers. Logical codes represent symbolic quantities, unsigned numbers, and bit quantities.

The American standard ASCII code is widely used to exchange character information. This code is generated by external devices such as printers and is used to transfer data between external devices and the computer's RAM [24]. Each character in ASCII code takes up 1 byte of memory, and the seventh bit is always 0. Since 1 byte consists of 8 bits, the ASCII table contains 28 characters, 256 in total.

In an ASCII table, the standard are the first 128 characters, which are numbered from 0 to 127. This range stores Latin letters, numbers, punctuation marks, and service characters. Characters 128 to 255 are used to represent characters from national alphabets, scientific symbols, and pseudographics.

Example: A word of 14 characters occupies 14 bytes of memory, which corresponds to 112 bits (14 bytes × 8 bits).

To consolidate the material, a solution to the problems is given:

Task:

The ASCII character 'i' has the code 105₁₀. Identify the word encrypted in the sequence 108 105 110 107₁₀.

Solution:

The numbering of characters in the ASCII table is alphabetical. It is necessary to correlate the characters with their codes and get the word "link".

Answer: link.

Task:

The word "stop" in binary must be represented in hexadecimal.

Solution:

To represent the word "stop" in hexadecimal, you first need to convert each letter to binary and then to hexadecimal.

1. Translation of letters to ASCII:

's' = 115

't' = 116

'o' = 111

'p' = 112

2. Converting ASCII to binary:

's' = 01110011

't' = 01110100

'o' = 01101111

'p' = 01110000

3. Combining binary code:

01110011 01110100 01101111 01110000

4. Converting binary code to hexadecimal:

To convert to a given number system, the binary code is divided into groups of 4 bits, starting from the right edge.

0010 0011 0111 0100 0110 1111 0111 0000

5. Converting each group to hexadecimal value:

0010 = 2

0011 = 3

0111 = 7

0100 = 4

0110 = 6

1111 = F

0111 = 7

0000 = 0

6. Combining hexadecimal values: 23746F70

Answer: 23746F70.

Thus, understanding data formats and their encoding is an important aspect of working with computer information.

## 1.4 Fixed-point number format in computers

The fixed-point number format is used to represent integers in computers. The range of integers that can be represented depends on the size of memory cells known as machine words. If a k-bit machine word is used to store an integer, then it contains k bits of memory. Thus, in a k-bit machine word it is possible to store 2k different integers.

Consider a 16-bit machine word. In this case, you can store 216 different integers, which is 65536 values.

Positive Numbers: If only positive numbers are stored, the range of possible values ​​will be 0 to 65535 (0 to 2k - 1).

Positive and Negative Numbers: If both positive and negative numbers are stored in equal numbers in memory, the range of values ​​will be -32768 to 32767 (-2k-1 to 2k-1 - 1).

Thus, the fixed-point number format allows integers to be represented efficiently, providing a variety of ranges depending on the number of bits used.

## 1.5 Algorithms for representing integers in computers

In modern computing systems, integers can be represented in a variety of formats, including binary and hexadecimal number systems. Let's consider algorithms for representing both positive and negative integers in a two-byte machine word.

Algorithm for representing a positive integer:

1. Definition of bit depth: Let a number be stored in a k-bit machine word.
2. Conversion to binary: The given positive integer N is converted to binary.
3. Zero padding: The resulting binary result is left padded with leading zeros to the k-digits, which represents the number in the computer.
4. Conversion to hexadecimal: The binary number is converted to hexadecimal because data is often stored packaged in this format.

For example, for the number 160710, which is stored in a two-byte machine word, the algorithm will be as follows: the number is converted to the binary system, padded with zeros to the required number of digits and then converted to the hexadecimal system (figure 1.1).

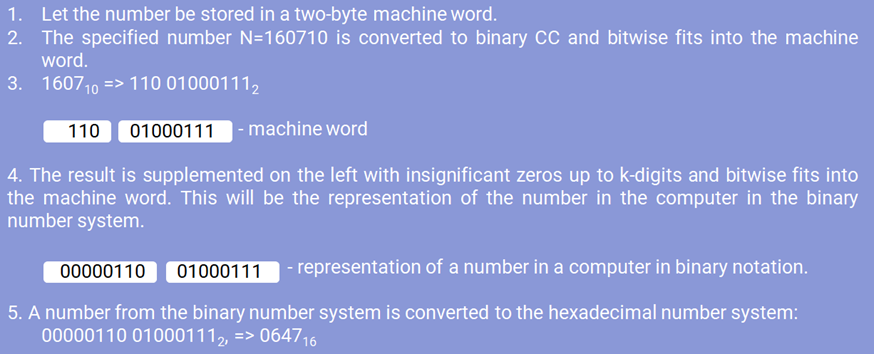


Figure 1.1 – Algorithm for representing the positive integer number 160710 in fixed point format

Algorithm for representing a negative integer:

1. Obtaining an internal representation: The modulus of a given negative number N is converted to the binary number system.

2. Obtaining the reverse code: The reverse code is obtained by replacing all 0s with 1s and vice versa.

3. Obtaining an additional code: One is added to the number obtained in the previous step.

4. Conversion to hexadecimal: A number is converted from binary to hexadecimal to ensure that it is stored in packed form.

When considering the algorithm for representing the negative number -160710 in a two-byte machine word, the computer carries out the following steps (Figure 1.2):

1. The system calculates the magnitude of the negative number -1607, eliminating the negative sign and obtaining the positive number 1607.

2. The resulting positive number is converted from decimal to binary. By dividing by 2 and keeping the remainder, the number 1607 is converted to binary as 11001000111.

3. Given the limitations of a two-byte machine word, the number is padded with zeros and takes the form 0000011001000111.

4. Then the bits of this number are inverted to obtain the reverse code. The reverse conversion turns the number 0000011001000111 into 1111100110111000.

5. The next step is to obtain the complementary code, which is achieved by adding one to the reverse code. Adding 1 to the reverse code 1111100110111000 results in the additional code 1111100110111001.

6. For ease of presentation, the calculated binary result 1111100110111001 is converted to hexadecimal, where it is split into groups of four bits (1111 1001 1011 1001) and replaced with the corresponding hexadecimal characters. As a result, the value of -1607 represented in memory will look like F9B9 in hexadecimal.

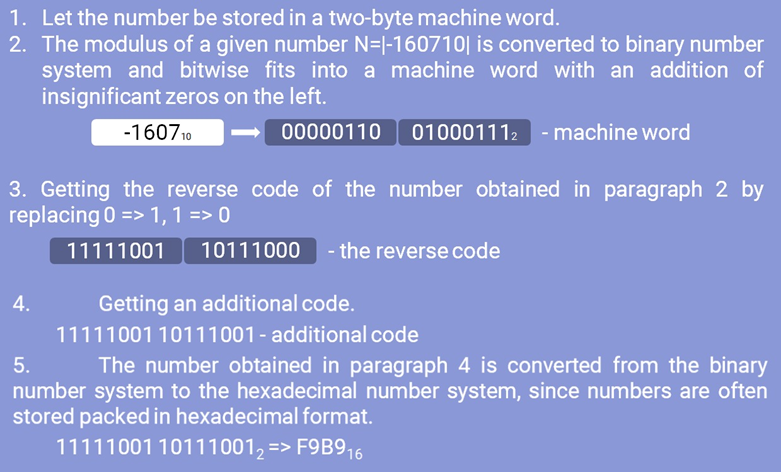


Figure 1.2 – Algorithm for representing the negative integer 160710 in fixed point format

## 1.6 Algorithm for representing a real number in a computer

In mathematics, the representation of real numbers is ambiguous, so an artificial form of representing a real number in a computer in a normalized form is used. It is based on the formula: *k = m + np*, where k is a real number; m – mantissa of the number; n – base of the number system; p – mathematical order of number.

For example, the number 25.3240 can be represented in the form of several expressions:

25.3240 = 2.5324 × 10 = 0.25324 × 102 = 0.0025324 × 104 = 2532.4 × 10-2

1. The normalized representation of real numbers provides compact memory usage by retaining only the most important elements: the mantissa, the exponent, the sign of the number, and the exponent of the real number. This representation does not store the integer part of the number, nor does it need to store the radix. The sign of a number indicates whether it is positive or negative, while the sign of the order determines whether the original number is positive or negative.

Examples of normalized representations of fractional numbers in the binary number system:

101.11 = 0.10111 × 1011

0.01 = 0.1 × 10-10

2. The longer the machine word, the more accurately a real number can be represented in computer memory. When stored in computer memory, the mantissa is represented as an integer that includes only the significant digits before and after the decimal point up to the first significant digit of the mantissa. Consequently, the internal representation of a real number is reduced to storing two integers: the mantissa and the exponent. Different types of computers use different ways of representing numbers in floating point format. Typically, a floating point number takes up either four bytes of memory (for standard precision numbers) or eight bytes (for double precision numbers).

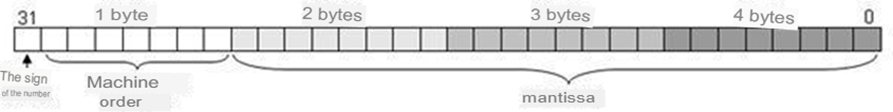


Figure 1.3 – Storing a real number in computer memory

From Figure 1.3 it can be seen that the most significant bit of the first byte stores the sign of the number - 0 (plus) or 1 (minus). The next 7 bits of the first byte contain the machine order. The significant digits of the mantissa (24 bits) are stored in the next three bytes.

The 7 bits of machine order contain binary numbers in the range from 00000002 (010) to 11111112 (12710), making up 128 different values. Given the possibility of both positive and negative order, these 128 values ​​are evenly distributed between positive and negative order values, spanning the range from -64 to 63 - the mathematical order (figure 1.4).

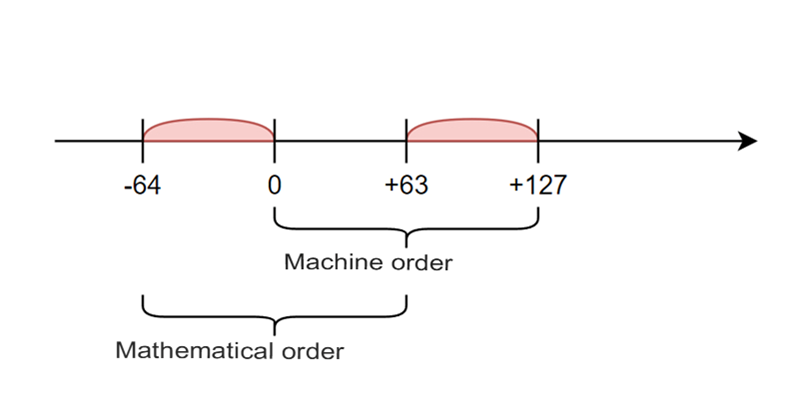


Figure 1.4 – Offset of the machine order relative to the mathematical one

The machine order is shifted relative to the mathematical order and contains only positive values. The offset is chosen so that the minimum mathematical order value corresponds to zero machine order.

Formulas for recalculating orders: *Mp = p + 64; MP2 = p2 + 10000002.*

Algorithm for representing a real number in computer memory:

1. The modulus of the number is converted to binary with 24 significant digits of the mantissa. If significant digits occupy fewer digits, they are padded with zeros to the right until the end of the machine word.

2. The binary number is normalized by the formula *k = m + np*.

The machine order of a number in the binary system is calculated taking into account the formula *Mp2 = p2 + 10000002*.

3. The sign of the number (0 for a positive number and 1 for a negative number), together with the machine order, fits into a 4-byte machine word, forming a single whole with the mantissa (Figure 1.3).

4. The resulting representation of a real number is displayed in the hexadecimal number system.

An example of representing the real number 250.187510 in a four-byte machine word, the computer performs the following steps:

1. Converting a number to binary number system with 24 significant digits: 11111010.00110000000000002.
2. Notated in normalized binary floating point form: 0.11111010001100000000 0000 × 101000.

Calculating the machine order in the binary number system using the formula *Mp2 = p2 + 10000002*: Mr2 = 10002 + 10000002 = 10010002.

1. Writing a number in a 4-byte memory cell, taking into account the sign:

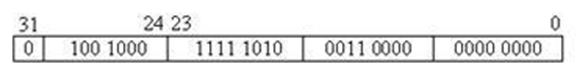


Figure 1.5 – Representation of the real number 250.187510 in a four-byte machine word

1. Hexadecimal form: 48FA3000

Thus, the computer successfully represents the real number 250.187510 in a four-byte machine word using the specified algorithm, and outputs the result in hexadecimal notation format.

As part of the practical task, it is necessary to obtain an internal computer representation of the following numbers in a two-byte machine word:

+19510

-19510

+26310

+345.06510

-164.07810