Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Politechnika Świętokrzyska
Wydział Zarządzania i Modelowania Komputerowego

Kierunek studiów:
Inżynieria danych

Paweł Stąpór

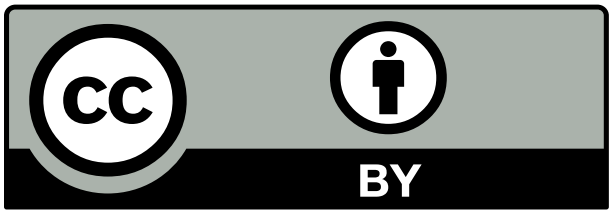
Materiały dydaktyczne do przedmiotu

FUNDAMENTALS OF COMPUTER SCIENCE

opracowane w ramach realizacji Projektu
„Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
FERS.01.05-IP.08-0234/23

Kielce, 2025

Spis treści



1. Introduction

Computer science is the study of computers and computational systems. Unlike electrical and computer engineers, computer scientists deal mostly with software and software systems; this includes their theory, design, development, and application.

Although knowing how to program is essential to the study of computer science, it is only one element of the field. Computer scientists design and analyze algorithms, study the performance of computer hardware and software, design applications and many more. So, what is computer science? Generally speaking, computer science is the study of computer technology, both hardware and software. However, computer science is a diverse field; the required skills are both applicable and in-demand across practically every industry in today's technology-dependent world. As such, the field of computer science is divided among a range of sub-disciplines, most of which are full-fledged specialized disciplines in and of themselves.

The disciplines encompassed by a computer science are incredibly vast, here are a several possible areas of specialization:

- Applied Mathematics,
- Digital Image/ Sound,
- Artificial Intelligence,
- Microprogramming,
- Bioinformatics,
- Networks And Administration,
- Computer Architecture Networks,
- Cryptography,
- Computer Engineering,
- Operating Systems,
- Computer Game Development,
- Robotics,
- Computer Graphics,
- Simulation And Modeling,

- Computer Programming,
- Software Development,
- Software Systems,
- Data Management,
- Web Development,
- Design Databases,
- Parallel Programming,
- iOS Development,
- Mobile Development,
- Memory Systems,
- Computational Physics.

In other words, Computer science is a discipline that spans theory and practice, it requires thinking both in abstract terms and in concrete terms. The practical side of computing can be seen everywhere. Nowadays, practically everyone is a computer user, and many people are even computer programmers. Getting computers to do what you want them to do requires intensive hands-on experience. But computer science can be seen on a higher level, as a science of problem solving. Computer scientists must be adept at modeling and analyzing problems. They must also be able to design solutions and verify that they are correct. Problem solving requires precision, creativity, and careful reasoning. Computer science also has strong connections to other disciplines. Many problems in science, engineering, health care, business, and other areas can be solved effectively with computers, but finding a solution requires both computer science expertise and knowledge of the particular application domain. Thus, computer scientists often become proficient in other subjects. Finally, computer science has a wide range of specialties. These include computer architecture, software systems, graphics, artifical intelligence, computational science, and software engineering. Drawing from a common core of computer science knowledge, each specialty area focuses on particular challenges.

Computer science is practiced by mathematicians, scientists and engineers. Mathematics, the origins of Computer science, provides reason and logic. Science provides the methodology for learning and refinement. Engineering provides the techniques for building hardware and software. Computer science is a discipline that involves the understanding and design of computers and computational processes.

In its most general form it is concerned with the understanding of information transfer and transformation. Particular interest is placed on making processes efficient and endowing them with some form of intelligence. The discipline ranges from theoretical studies of algorithms to practical problems of implementation in terms of computational hardware and software.

A central focus is on processes for handling and manipulating information. Thus, the discipline spans both advancing the fundamental understanding of algorithms and information processes in general as well as the practical design of efficient reliable software and hardware to meet given specifications. Computer science is a young discipline that is evolving rapidly from its beginnings in the 1940's. As such it includes theoretical studies, experimental methods, and engineering design all in one discipline. This differs radically from most physical sciences that separate the understanding and advancement of the science from the applications of the science in fields of engineering design and implementation. In computer science there is an inherent intermingling of the theoretical concepts of computability and algorithmic efficiency with the modern practical advancements in electronics that continue to stimulate advances in the discipline. It is this close interaction of the theoretical and design aspects of the field that binds them together into a single discipline. Because of the rapid evolution it is difficult to provide a complete list of computer science areas.

A professional computer scientist must have a firm foundation in the crucial areas of the field and will most likely have an in-depth knowledge in one or more of the other areas of the discipline, depending upon the person's particular area of practice. Thus, a well educated computer scientist should be able to apply the fundamental concepts and techniques of computation, algorithms, and computer design to a specific design problem. The work includes detailing of specifications, analysis of the problem, and provides a design that functions as desired, has satisfactory performance, is reliable and maintainable, and meets desired cost criteria. Clearly, the computer scientist must not only have sufficient training in the computer science areas to be able to accomplish such tasks, but must also have a firm understanding in areas of mathematics and science, as well as a broad education in liberal studies to provide a basis for understanding the societal implications of the work being performed.

From a different point of view, Computer science is the study of principles, applications, and technologies of computing and computers. It involves the study of data and data structures and the algorithms to process these structures; of principles of computer architecture-both hardware and software; of problem-solving and design methodologies; of computer-related topics such as numerical analysis, operations research, and artificial intelligence; and of language design, structure, and translation technique.

2. Historical context of computer science

From the simple calculator to a modern day powerful data processor, computing devices have evolved in a relatively short span of time:

- 500 BC – invention of ABACUS almost 3000 years ego. It was a mechanical device capable of doing simple arithmetic calculations only.
- 1642 – Blaize Pascal invented a mechanical calculator known as Pascal calculator or Pascaline to do addition and subtraction of two numbers directly and multiplication and division through repeated addition and subtraction.
- 1834 – Charles Babbage invented analytical engine, a mechanical computing device for imputting, processing, storing and displaying the output, which is considered to form the basis of modern computers.
- 1890 – Herman Hollerith designed a tabulating machine for summarising the data stored on the punched card. It is considered to be the first step towards programming.
- 1937 – The Turing machine concept was a general purpose programmable machine that was capable of solving any problem by executing the program stored on the punched cards.
- 1945 – Jhon Von Neumann introduced the concept of stored program computer which was capable of storing data as well as program in the memmory. The EDVAC/ENIAC computers were developed based on this concept.
- 1947 – Vacum tubes were replaced by transistors developed at Bell Labs, using semiconductor materials
- 1970 – An Integreted Circuit (IC) is a silicon chip which contains entire electronic circuit on a very small area. The size of computer drastically reduced because of ICs.
- 1981 – First PC's. IBM introduced its first personal computer (PC) for the home user and Apple introduced Macintosh machines in 1984. The popularity of the PC surged by the introduction of Graphical User Interface (GUI) based operating systems by Microsoft and others in place of computers with only command line interface, like UNIX or DOS.

- Around –1990s the growth of World Wide Web (WWW) further accelerated mass usage of computers and thereafter computers have become an indispensable part of everyday life.

Selected Devices:

[Alt text. Figure shows selected devices: Pascaline – a calculating machine designed by Blaise Pascal.]

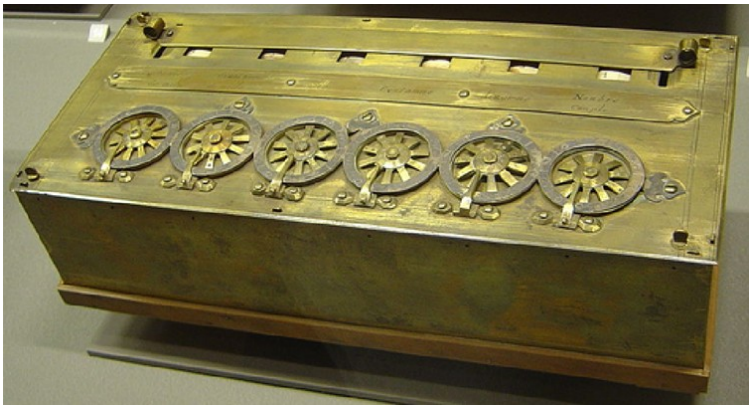


Fig. 1. Pascaline – a calculating machine designed by Blaise Pascal around 1645.

[Alt text. Figure shows selected devices: Z3 was the first operational, fully automatic, variable-program computer built by German engineer Konrad Zuse in 1941.]

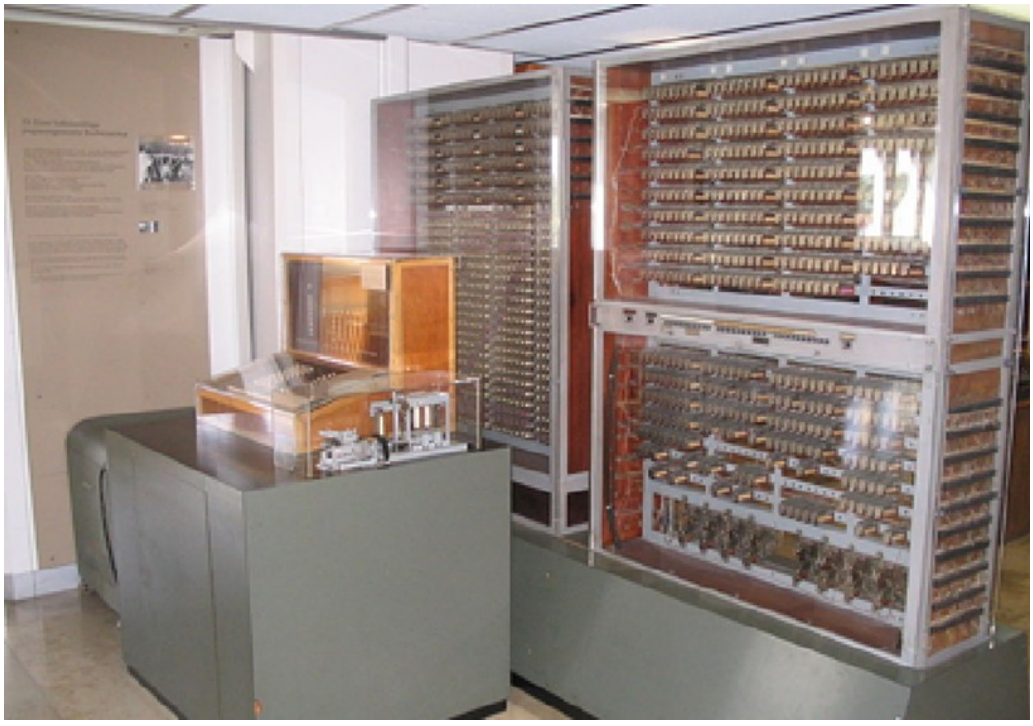


Fig. 2. The Z3 build by German engineer Konrad Zuse in 1941.

The Z3 was the first operational, fully automatic, variable-program computer built by German engineer Konrad Zuse in 1941, based on his earlier mechanical design, the Z1. The machine was used during the war for calculations necessary for wing design.

[Alt text. Figure shows selected devices: Colossus is a series of programmable digital machines based on the theoretical foundations of Alan Turing's work, 1943.]

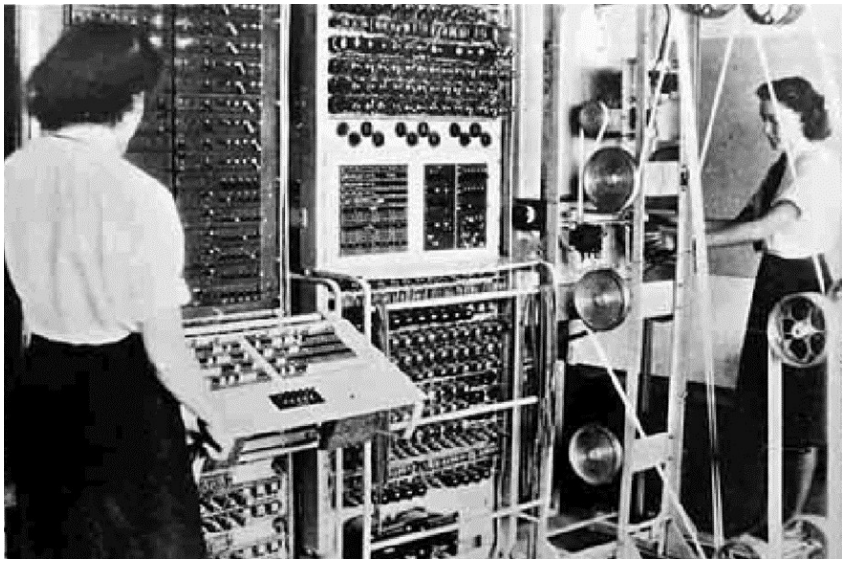


Fig. 3. Colossus built on April 14, 1943, at the British Bletchley Park cryptographic center.

Colossus is a series of programmable digital machines based on the theoretical foundations of Alan Turing's work. The Colossus project was led by Max Newman and Tommy Flowers, with Alan Turing also participating. Built on April 14, 1943, at the British Bletchley Park cryptographic center, Colossus was intended for military use. It was used to decipher the operation of the German Lorenz Machine and break its ciphers.

[Alt text. Figure shows selected devices: ENIAC, or Electronic Numerical Integrator and Computer, was a computer constructed between 1943 and 1945 by J. P. Eckert and J. W. Mauchly.]

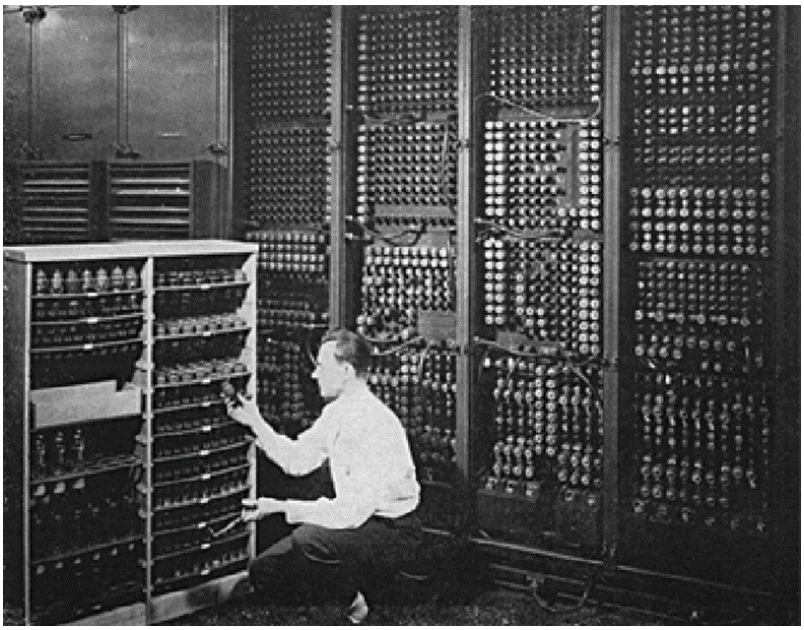


Fig. 4. ENIAC, or Electronic Numerical Integrator and Computer.

ENIAC, or Electronic Numerical Integrator and Computer, was a computer constructed between 1943 and 1945 by J. P. Eckert and J. W. Mauchly at the University of Pennsylvania in the USA. Its use was discontinued in 1955. Until 1975, it was widely considered the world's first computer, but now, following the declassification of British data, the Colossus and the German Konrad Zuse machines are also claiming this title..

[Alt text. Figure shows selected devices: first PC's.]

Fig. 5. First PC's – in 1981 IBM introduced PC and in 1984 Apple introduced Macintosh.

3. Computer system

A computer is an electronic device that can be programmed to accept data (input), process it and generate result (output). A computer along with additional hardware and software together is called a computer system.

Components of a computer system

A computer system primarily comprises a central processing unit (CPU), memory, input/output devices and storage devices. All these components function together as a single unit to deliver the desired output. A computer system comes in various forms and sizes. It can vary from a high-end server to personal desktop, laptop, tablet computer, or a smartphone.

[Alt text. Figure shows a diagram of components of a computer system with CPU.]

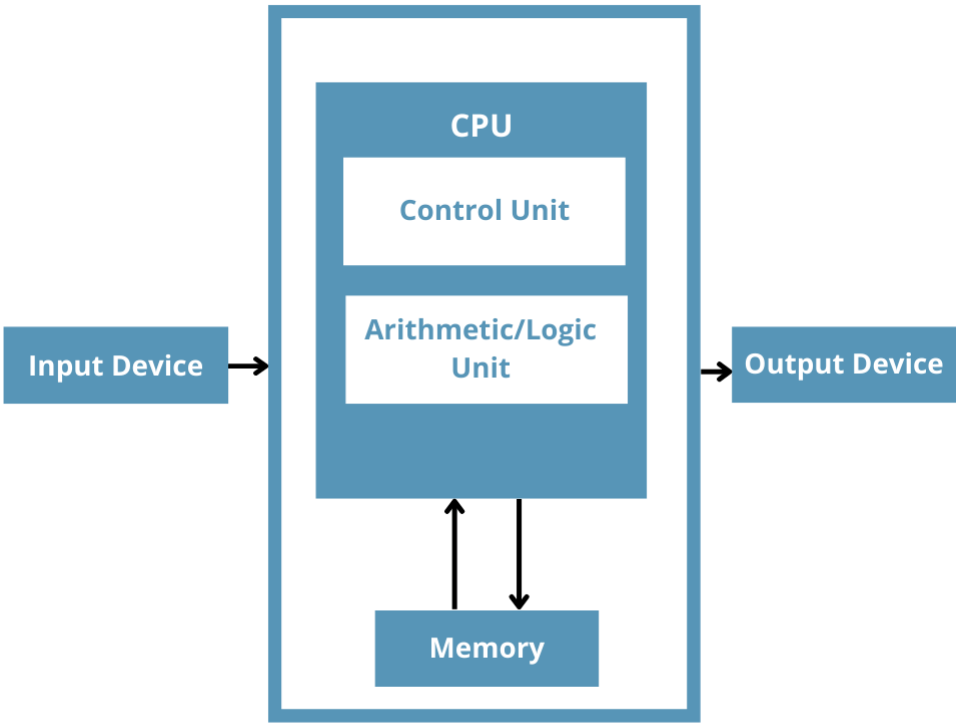


Fig. 6. Components of a computer system.

Central Processing Unit (CPU)

CPU is the electronic circuitry of a computer that carries out the actual processing and usually referred as the brain of the computer. It is commonly called processor also. Physically, a CPU can be placed on one or more microchips called integrated circuits (IC). The ICs comprise semiconductor materials.

The CPU is given instructions and data through programs. The CPU then fetches the program and data from the memory and performs arithmetic and logic operations as per the given instructions and stores the result back to memory. While processing, the CPU stores the data as well as instructions in its local memory called registers.

Registers are part of the CPU chip and they are limited in size and number. Different registers are used for storing data, instructions or intermediate results. Other than the registers, the CPU has two main components — Arithmetic Logic Unit (ALU) and Control Unit (CU). ALU performs all the arithmetic and logic operations that need to be done as per the instruction in a program. CU controls sequential instruction execution, interprets instructions and guides data flow through the computer's memory, ALU and input or output devices. CPU is also popularly known as microprocessor.

[Alt text. Figure shows a real CPU.]

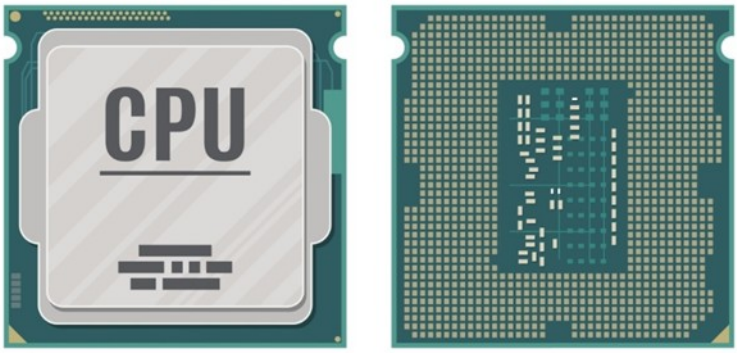


Fig. 7. Real picture of CPU known as microprocessor.

Input output devices

The devices through which control signals are sent to a computer are termed as input devices. These devices convert the input data into a digital form that is acceptable by the computer system. Data entered through input device is temporarily stored in the main memory (also called RAM) of the computer system. For permanent storage and future use, the data as well as instructions are stored permanently in additional storage locations called secondary memory.The device that receives data from a computer system for display, physical production, etc., is called output device. It converts digital information into human- understandable form.

[Alt text. Figure shows examples of input/ouput devices.]

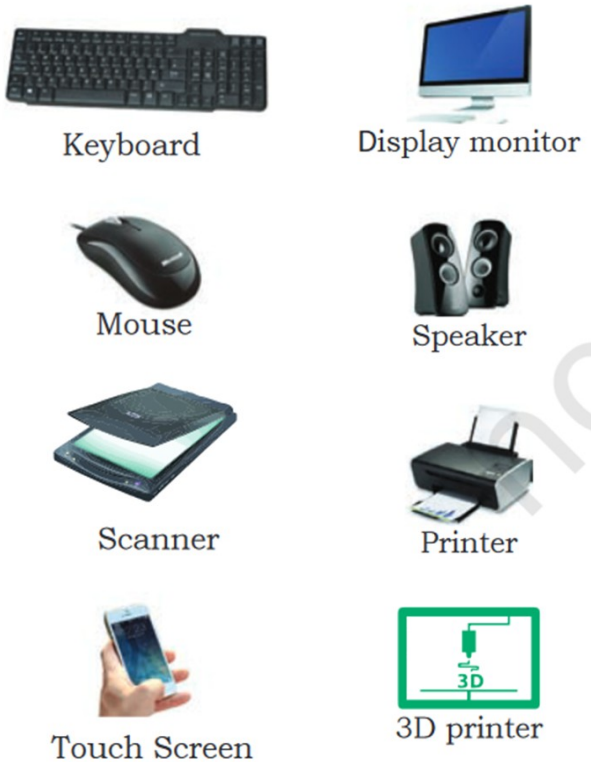


Fig. 8. The examples of input/output devices.

The Von Neumann architecture

The Von Neumann architecture consists of a Central Processing Unit (CPU) for processing arithmetic and logical instructions, a memory to store data and programs, input and output devices and communication channels to send or receive the output data. Electronic Numerical Integrator and Computer (ENIAC) is the first binary programmable computer based on Von Neumann architecture.

[Alt text. Figure shows a diagram of the Von Neumann architecture.]

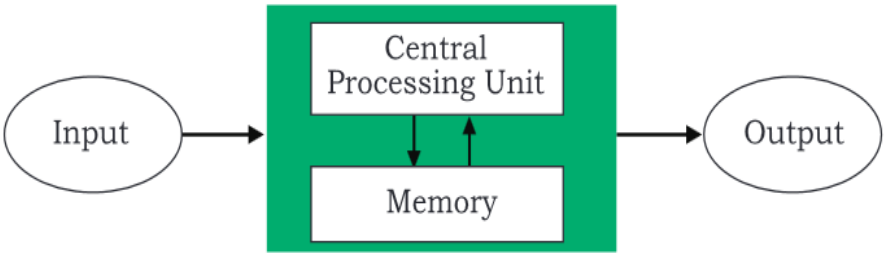


Fig. 9. The Von Neumann architecture.

Computer memory

A computer system needs memory to store the data and instructions for processing. Whenever we talk about the ‘memory’ of a computer system, we usually talk about the main or primary memory. The secondary memory (also called storage device) is used to store data, instructions and results permanently for future use.

[Alt text. Figure shows a real computer memory.]

Fig. 10. A computer memory.

Human beings memorise many things over a lifetime, and recall from memory to make a decision or some action. However, we do not rely on our memory completely, and we make notes and store important data and information using other media, such as notebook, manual, journal, document, etc. Similarly, computers have two types of memory — primary and secondary.

Primary memory is an essential component of a computer system. Program and data are loaded into the primary memory before processing. The CPU interacts directly with the primary memory to perform read or write operation. It is of two types viz. (i) Random Access Memory (RAM) and (ii) Read Only Memory (ROM).

RAM is volatile, i.e., as long as the power is supplied to the computer, it retains the data in it. But as soon as the power supply is turned off, all the contents of RAM are wiped out. It is used to store data temporarily while the computer is working. Whenever the computer is started or a software application is launched, the required program and data are loaded into RAM for processing. RAM is usually referred to as main memory and it is faster than the secondary memory or storage devices.

On the other hand, ROM is non-volatile, which means its contents are not lost even when the power is turned off. It is used as a small but faster permanent storage for the contents which are rarely changed. For example, the startup program (boot loader) that loads the operating system into primary memory, is stored in ROM.

RAM is faster than secondary storage, but not as fast as a computer processor. So, because of RAM, a CPU may have to slow down. To speed up the operations of the CPU, a very high speed memory is placed between the CPU and the primary memory known as cache. It stores the copies of the data from frequently accessed primary memory locations, thus, reducing the average time required to access data from primary memory.When the CPU needs some data, it first examines the cache. In case the requirement is met, it is read fromthe cache, otherwise the primary memory is accessed.

Primary memory has limited storage capacity and is either volatile (RAM) or read-only (ROM). Thus, a computer system needs auxiliary or secondary memory to permanently store the data or instructions for future use. The secondary memory is non-volatile and has larger storage capacity than primary memory. It is slower and cheaper than the main memory. But, it cannot be accessed directly by the CPU. Contents of secondary storage need to be first brought into the main memory for the CPU to access.

[Alt text. Figure shows types of computer memory.]

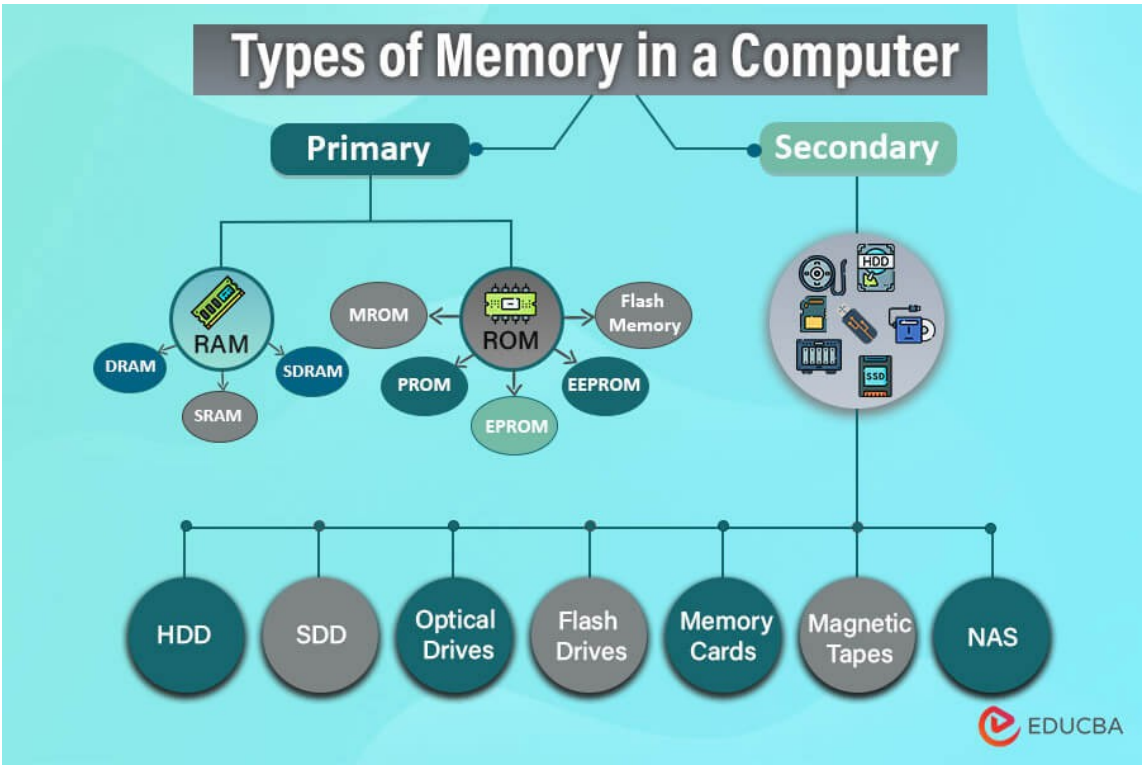


Fig. 11. Types of computer memory.

Data transfer between memory and CPU

Data need to be transferred between the CPU and primary memory as well as between the primary and secondary memory.

Data are transferred between different components of a computer system using physical wires called bus. For example, bus is used for data transfer between a USB port and hard disk or between a hard disk and main memory. Bus is of three types

- Data bus to transfer data between different components,
- Address bus to transfer addresses between CPU and main memory. The address of the memory location that the CPU wants to read or write from is specified in the address bus,
- Control bus to communicate control signals between different components of a computer.

All these three buses collectively make the system bus.

As the CPU interacts directly with main memory, any data entered frominput device or the data to be accessedfrom hard disk needs to be placed in the main memory for further processing.The data is then transferred betweenCPU and main memory using bus. The CPU places on the address bus, the address of the main memory location from which it wants to read data or to write data. While executing the instructions,the CPU specifies the read or write control signal through the control bus.

As the CPU may require to read data from main memory or write data to main memory, a data bus is bidirectional. But the control bus and address bus are unidirectional. To write data into memory, the CPU places the data on the data bus, which is then written to the specific address provided through the address bus. In case of read operation, the CPU specifies the address, and the data is placed on the data bus by a dedicated hardware, called memory controller.

[Alt text. Figure shows diagram of data transfer between memory and CPU.]

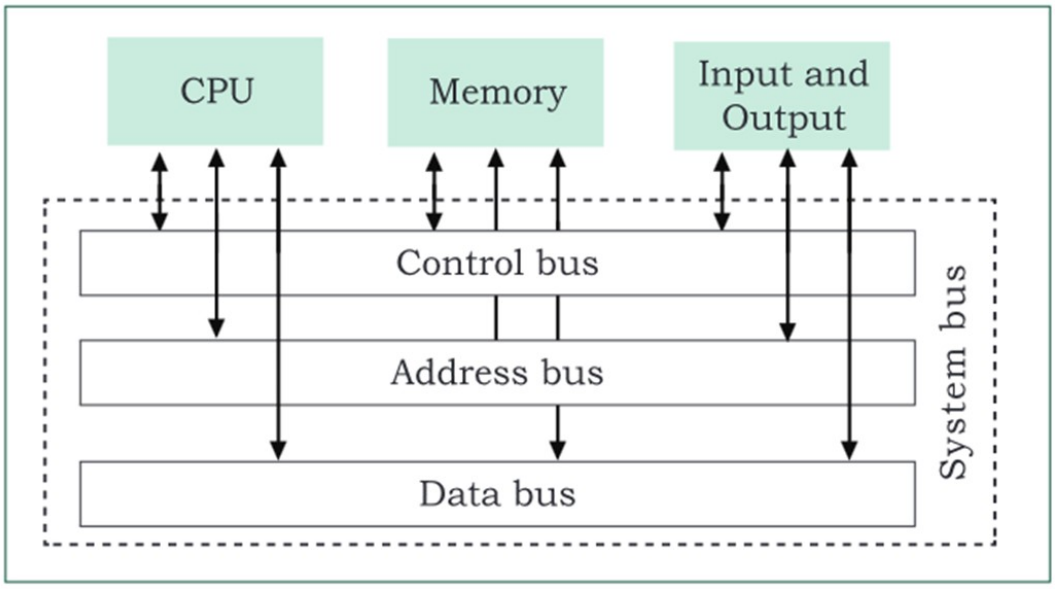


Fig. 12. Data transfer between memory and CPU.

As the CPU may require to read data from main memory or write data to main memory, a data bus is bidirectional. But the control bus and address bus are unidirectional. To write data into memory, the CPU places the data on the data bus, which is then written to the specific address provided through the address bus. In case of read operation, the CPU specifies theaddress, and the data is placed on the data bus by adedicated hardware, called memory controller.

Microprocessors

In earlier days, a computer's CPU used to occupy a large room or multiple cabinets. However, with advancement in technology, the physical size of CPU has reduced and it is now possible to place a CPU on a single microchip only. A processor (CPU) which is implemented on a single microchip is called microprocessor. Nowadays, almost all the CPUs are microprocessors. Hence, the terms are used synonymously for practical purpose.

Microprocessor is a small-sized electronic component inside a computer that carries out various tasks involved in data processing as well as arithmetic and logical operations. These days, a microprocessor is built over an integrated circuit comprising millions of small components like resistors, transistors and diodes.

Microprocessors have evolved over time in terms of their increased processing capability, decreasing physical size and reduced cost. Currently available microprocessors are capable of processing millions of instructions per millisecond.

Microprocessors are classified on the basis of different features which include chip type, word size, memory size, clock speed, etc.

A) Word Size

Word size is the maximum number of bits that a microprocessor can process at a time. Earlier, a word was of 8 bits, as it was the maximum limit at that time. At present, the minimum word size is 16 bits and maximum word size is 64 bits.

B) Memory Size

Depending upon the word size, the size of RAM varies. Initially, RAM was very small (4MB) due to 4/8 bits word size. As word size increased to 64 bits, it has become feasible to use RAM of size upto 16 Exabytes (EB).

C) Clock Speed

Computers have an internal clock that generates pulses (signals) at regular intervals of time. Clock speed simply means the number of pulses generated per second by the clock inside a computer. The clock speed indicates the speed at which the computer can execute instructions. Earlier, it was measured in Hertz (Hz) and Kilohertz (kHz). But with advancement in technology and chip density, it is now measured in Gigahertz (GHz), i.e., billions of pulses per second.

D) Cores

Core is a basic computation unit of the CPU. Earlier processors had only one computation unit, thereby capable of performing only one task at a time. With the advent of multicore processor, it has become possible for the computer to execute multiple tasks, thereby increasing the system's performance. CPU with two, four, and eight cores is called dual-core, quad-core and octa-core processor, respectively.

4. Data and information

A computer is primarily for processing data. A computer system considers everything as data, be it instructions, pictures, songs, videos, documents, etc. Data can also be raw and unorganised facts that are processed to get meaningful information. So understanding the concept of data along with its different types is crucial to understand the overall functioning of a computer. Sometimes people use the terms data, information and knowledge interchangeably, which is incorrect.

A computer system has many input devices, which provide it with raw data in the form of facts, concepts, instructions, etc., Internally everything is stored in binary form (0 and 1), but externally, data can be input to a computer in the text form, numerals 0 – 9, etc. Primarily, there are three types of data.

A) Structured Data

Data which follows a strict record structure and is easy to comprehend is called structured data. Such data with pre-specified tabular format may be stored in a data file to access in the future. Table is an example of structured data. It is clear that such data is organised in row/column format and is easily understandable. Structured data may be sorted in ascending or descending order.

B) Unstructured Data

Data which are not organised in a pre-defined record format is called unstructured data. Examples include audio and video files, graphics, text documents, social media posts, satellite images, etc. Figure shows a report card with monthly attendance record details sent to parents. Such data are unstructured as they consist of textual contents as well as graphics, which do not follow a specific format.

C) Semi-structured Data

Data which have no well-defined structure but maintains internal tags or markings to separate data elements are called semi-structured data. Examples include email document, HTML page, comma separated values (csv file), etc. Figure 13 shows an example of semi-structured data containing student's month-wise attendance details. In this example, there is no specific format for each attendance record. Here, each data value is preceded by a tag (Name, Month, Class, Attendance) for the interpretation of the data value while processing.

[Alt text. Figure shows Semi-structured data.]

Name: *Mohan*	Month: *July*	Class: *XI*	Attendance: *98*
Name: *Sohan*	Month: *July*	Class: *XI*	Attendance: *65*
Name: *Sheen*	Month: *July*	Class: *XI*	Attendance: *85*
Name: *Geet*	Month: *May*	Class: *XI*	Attendance: *82*
Name: *Geet*	Month: *July*	Class: *XI*	Attendance: *94*

Fig. 13. Semi-structured data.

Data structure and alghoritms

Data structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. Data structure is representation of the logical relationship existing between individual elements of data. In other words, a data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.

Data structure affects the design of both structural & functional aspects of a program:

Program=algorithm + sata structure.

You know that a algorithm is a step by step procedure to solve a particular function. That means, algorithm is a set of instruction written to carry out certain tasks & the data structure is the way of organizing the data with their logical relationship retained. To develop a program of an algorithm, we should select an appropriate data structure for that algorithm. Therefore algorithm and its associated data structures form a program.

Data structure are normally divided into two broad categories: *Primitive Data Structure* and *Non-Primitive Data Structure.*

[Alt text. Figure shows classification of data structures.]

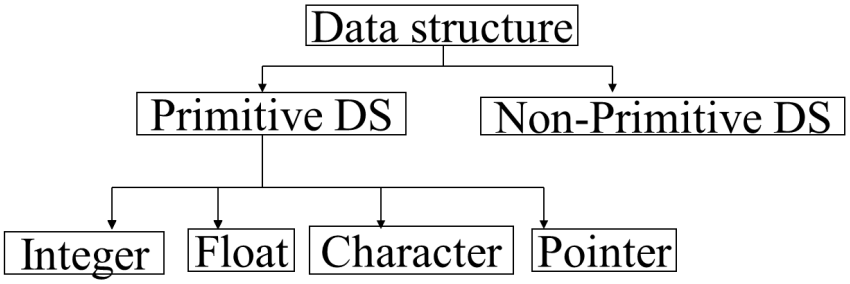


Fig. 14. classification of data structures.

Primitive Data Structure are basic structures which directly operated upon by the machine instructions. In general, there are different representation on different computers. Integer, Floating-point number, character constants, string constants, pointers etc, fall in this category.

A primitive data structure is generally a basic structure that is usually built into the language, such as an integer, a float. A non-primitive data structure is built out of primitive data structures linked together in meaningful ways, such as a or a linked-list, binary search tree, graph etc.

Non-Primitive Data Structure are more sophisticated data structures.These are derived from the primitive data structures. The non-primitive data structures emphasize on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items. Lists, Stack, Queue, Tree, Graph are example of non-primitive data structures. The design of an efficient data structure must take operations to be performed on the data structure.

[Alt text. Figure shows classification of non-primitive data structures.]

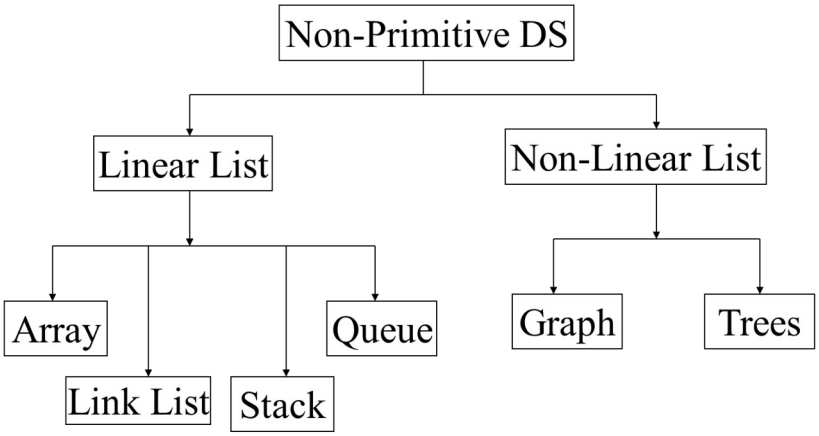


Fig. 15. Classification of non-primitive data structures.

The most commonly used operation on data structure are broadly categorized into following types:

- Create
- Selection
- Updating
- Searching
- Sorting
- Merging
- Destroy or Delete

5. Data representation in computer systems

Key issue in dealing with Computer systems is to understand:

- the fundamentals of numerical data representation and manipulation in digital computers,
- converting between various radix systems,
- how errors can occur in computations because of overflow and truncation,
- the fundamental concepts of floating-point representation,

- the most popular character codes.

Units of information

A bit is the most basic unit of information in a computer. It is a state of “on” or “off” in a digital circuit. Sometimes these states are “high” or “low” voltage instead of “on” or “off..”

A byte is a group of eight bits. A byte is the smallest possible addressable unit of computer storage. The term, “addressable,” means that a particular byte can be retrieved according to its location in memory.

A word is a contiguous group of bytes. Words can be any number of bits or bytes. Word sizes of 16, 32, or 64 bits are most common. In a word-addressable system, a word is the smallest addressable unit of storage.

A nibble is a group of four bits. Bytes, therefore, consist of two nibbles: a “high-order nibble,” and a “low-order” nibble.

Positional numbering systems

Bytes store numbers using the position of each bit to represent a power of 2. The binary system is also called the base-2 system. Our decimal system is the base-10 system. It uses powers of 10 for each position in a number. Any integer quantity can be represented exactly using any base (or radix).

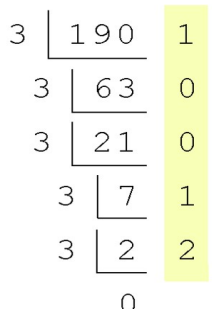
The binary number 11001 in powers of 2 is:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 16 + 8 + 0 + 0 + 1 = 25$$

When the radix of a number is something other than 10, the base is denoted by a subscript. Sometimes, the subscript 10 is added for emphasis: $11001_2 = 25_{10}$.

Converting between bases needs following operations, e.g. converting 190 to base 3:



- Continue in this way until the quotient is zero.
- In the final calculation, we note that 3 divides 2 zero times with a remainder of 2.
- Our result, reading from bottom to top is: $190_{10} = 21001_3$

The binary numbering system is the most important radix system for digital computers. However, it is difficult to read long strings of binary numbers - and even a modestly-sized decimal number becomes a very long binary number.

For example: $11010100011011_2 = 13595_{10}$

For compactness and ease of reading, binary values are usually expressed using the hexadecimal, or base-16, numbering system.

The conversions we have so far presented have involved only unsigned numbers. To represent signed integers, computer systems allocate the high-order bit to indicate the sign of a number. The high-order bit is the leftmost bit. It is also called the most significant bit. 0 is used to indicate a positive number; 1 indicates a negative number. The remaining bits contain the value of the number (but this can be interpreted different ways).

There are three ways in which signed binary integers may be expressed:

- Signed magnitude
- One's complement
- Two's complement

In an 8-bit word, signed magnitude representation places the absolute value of the number in the 7 bits to the right of the sign bit.

For example, in 8-bit signed magnitude representation:

+3 is: 00000011

- 3 is: 10000011

Computers perform arithmetic operations on signed magnitude numbers in much the same way as humans carry out pencil and paper arithmetic. Humans often ignore the signs of the operands while performing a calculation, applying the appropriate sign after the calculation is complete.

Binary addition is as easy as it gets. You need to know only four rules:

0 + 0 = 0 0 + 1 = 1

1 + 0 = 1 1 + 1 = 10

The simplicity of this system makes it possible for digital circuits to carry out arithmetic operations.

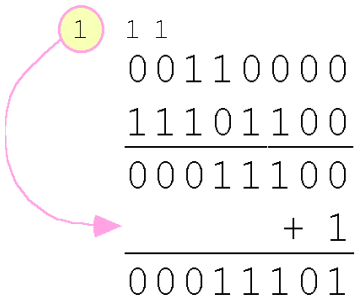
Let's see how the addition rules work with signed magnitude numbers. For example: using signed magnitude binary arithmetic, find the sum of 75 and 46. First, convert 75 and 46 to binary, and arrange as a sum, but separate the (positive) sign bits from the magnitude bits.

```
0   1001011
0 + 0101110
```

Signed magnitude representation is easy for people to understand, but it requires complicated computer hardware. Another disadvantage of signed magnitude is that it allows two different representations for zero: positive zero and negative zero. For these reasons (among others) computers systems employ *complement systems* for numeric value representation.

In complement systems, negative values are represented by some difference between a number and its base. The diminished radix complement of a non-zero number *N* in base *r* with *d* digits is *(rd – 1) – N* . In the binary system, this gives us one's complement. It amounts to little more than flipping the bits of a binary number.

With one's complement addition, the carry bit is "carried around" and added to the sum. For example: using one's complement binary arithmetic, find the sum of 48 and – 19:



We note that 19 in binary is 00010011, so -19 in one's complement is: 11101100.

Although the "end carry around" adds some complexity, one's complement is simpler to implement than signed magnitude. But it still has the disadvantage of having two different representations for zero: positive zero and negative zero. *Two's complement* solves this problem. Two's complement is the radix complement of the binary numbering system; the radix complement of a non-zero number *N* in base *r* with *d* digits is $r^d - N$.

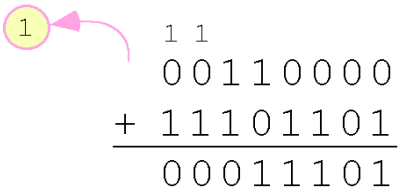
To express a value in two's complement representation:

- If the number is positive, just convert it to binary and you're done.
- If the number is negative, find the one's complement of the number and then add 1.

Example:

In 8-bit binary, 3 is: 00000011, -3 using one's complement representation is: 11111100. Adding 1 gives us -3 in two's complement form: 11111101.

With two's complement arithmetic, all we do is add our two binary numbers. Just discard any carries emitting from the high order bit. For example using two's complement binary arithmetic find the sum of 29 and -19. Note that 19 in binary is: 00010011, so -19 using one's complement is: 11101100, and -19 using two's complement is: 11101101.



When we use any finite number of bits to represent a number, we always run the risk of the result of our calculations becoming too large or too small to be stored in the computer. While we can't always prevent overflow, we can always detect overflow. In complement arithmetic, an overflow condition is easy to detect.

Example:

Using two's complement binary arithmetic, find the sum of 107 and 46.

```
  1 1   1 1 1
  01101011
+ 00101110
----------
  10011001
```

We see that the nonzero carry from the seventh bit overflows into the sign bit, giving us the erroneous result: 107 + 46 = -103.

Using two's complement binary arithmetic, find the sum of 23 and -9.

```
1 ← 1 1 1   1 1 1
    00010111
+   11110111
------------
    00001110
```

We see that there is carry into the sign bit and carry out. The final result is correct: 23 + (-9) = 14.

Rule for detecting signed two's complement overflow: When the "carry in" and the "carry out" of the sign bit differ, overflow has occurred. If the carry into the sign bit equals the carry out of the sign bit, no overflow has occurred.

Floating-Point Representation

The signed magnitude, one's complement, and two's complement representation that we have just presented deal with signed integer values only. Without modification, these formats are not useful in scientific or business applications that deal with real number values. Floating-point representation solves this problem.

If we are clever programmers, we can perform floating-point calculations using any integer format. This is called floating-point emulation, because floating point values aren't stored as such; we just create programs that make it seem as if floating-point values are being used. Most of today's computers are equipped with specialized hardware that performs floating-point arithmetic with no special programming required.

Floating-point numbers allow an arbitrary number of decimal places to the right of the decimal point.

For example: 0.5 x 0.25 = 0.125

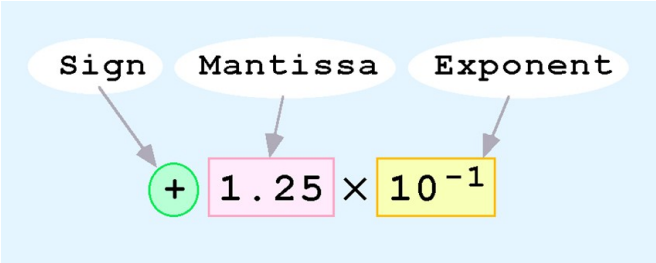
They are often expressed in scientific notation.

For example:

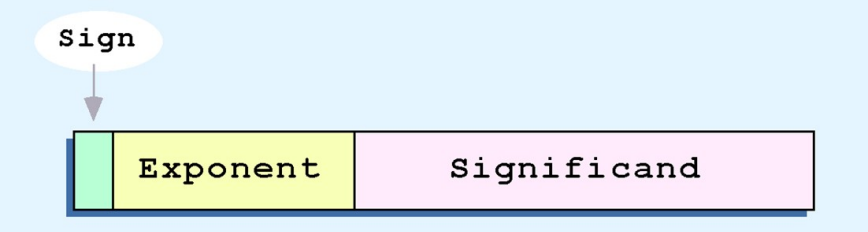
$0.125 = 1.25 \times 10^{-1}$

$5{,}000{,}000 = 5.0 \times 10^{6}$

Computers use a form of scientific notation for floating-point representation. Numbers written in scientific notation have three components:



Computer representation of a floating-point number consists of three fixed-size fields:



This is the standard arrangement of these fields.

We introduce a hypothetical “Simple Model” to explain the concepts. In this model:

- A floating-point number is 14 bits in length
- The exponent field is 5 bits
- The significand field is 8 bits

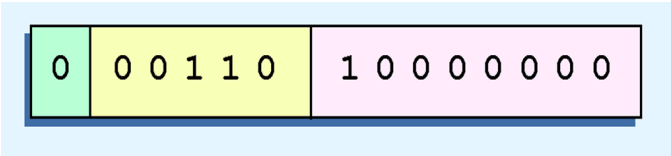
Example:

Express 32_{10} in the simplified 14-bit floating-point model.

We know that 32 is 2^5. So in (binary) scientific notation $32 = 1.0 \times 2^5 = 0.1 \times 2^6$.

In a moment, we’ll explain why we prefer the second notation versus the first.

Using this information, we put 110 (= 6_{10}) in the exponent field and 1 in the significand as shown.



The IEEE has established a standard for floating-point numbers. The IEEE-754 single precision floating point standard uses an 8-bit exponent (with a bias of 127) and a 23-bit significand. The IEEE-754 double precision standard uses an 11-bit exponent (with a bias of 1023) and a 52-bit significand.

In both the IEEE single-precision and double-precision floating-point standard, the significant has an implied 1 to the LEFT of the radix point.

The format for a significand using the IEEE format is: 1.xxx…

For example, 4.5 = .1001 x 2^3 in IEEE format is 4.5 = 1.001 x 22. The 1 is implied, which means is does not need to be listed in the significand (the significand would include only 001).

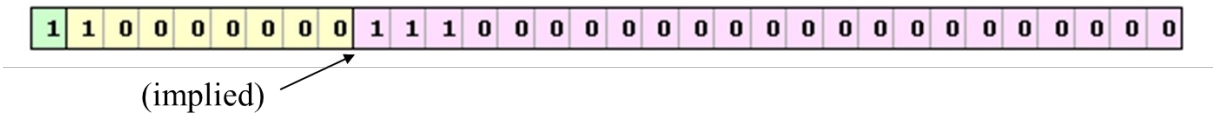
Example: Express -3.75 as a floating point number using IEEE single precision.

First, let's normalize according to IEEE rules:

$3.75 = -11.11_2 = -1.111 \times 2^1$

The bias is 127, so we add 127 + 1 = 128 (this is our exponent)

The first 1 in the significand is implied, so we have:



Since we have an implied 1 in the significand, this equates to

$-(1).111_2 \times 2^{(128-127)} = -1.111_2 \times 2^1 = -11.11_2 = -3.75.$

Character Codes

Calculations aren't useful until their results can be displayed in a manner that is meaningful to people. We also need to store the results of calculations, and provide a means for data input. Thus, human-understandable characters must be converted to computer-understandable bit patterns using some sort of character encoding scheme.

Binary-coded decimal (BCD) was one of these early codes. It was used by IBM mainframes in the 1950s and 1960s.

Decimal Digit	BCD 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

In 1964, BCD was extended to an 8-bit code, Extended Binary-Coded Decimal Interchange Code (EBCDIC). EBCDIC was one of the first widely-used computer codes that supported upper and lowercase alphabetic characters, in addition to special characters, such as punctuation and control characters. EBCDIC and BCD are still in use by IBM mainframes today.

Other computer manufacturers chose the 7-bit ASCII (American Standard Code for Information Interchange) as a replacement for 6-bit codes. While BCD and EBCDIC were based upon punched card codes, ASCII was based upon telecommunications (Telex) codes. Until recently, ASCII was the dominant character code outside the IBM mainframe world.

Many of today's systems embrace Unicode, a 16-bit system that can encode the characters of every language in the world. The Java programming language, and some operating systems now use Unicode as their default character code. The Unicode codespace is divided into six parts. The first part is for Western alphabet codes, including English, Greek, and Russian.

The Unicode codes - pace allocation is shown at the right. The lowest-numbered Unicode characters comprise the ASCII code. The highest provide for user-defined codes.

Character Types	Language	Number of Characters	Hexadecimal Values
Alphabets	Latin, Greek, Cyrillic, etc.	8192	0000 to 1FFF
Symbols	Dingbats, Mathematical, etc.	4096	2000 to 2FFF
CJK	Chinese, Japanese, and Korean phonetic symbols and punctuation.	4096	3000 to 3FFF
Han	Unified Chinese, Japanese, and Korean	40,960	4000 to DFFF
	Han Expansion	4096	E000 to EFFF
User Defined		4095	F000 to FFFE

6. Literatura

1. J. Glenn Brookshear, Dennis Brylow – Computer Science: An Overview, 13th Edition, Pearson, 2023.
2. Computer Science Field Guide – https://www.csfieldguide.org.nz/, (dostępny 30.09.2025).
3. MIT OpenCourseWare – Introduction to Computer Science, https://ocw.mit.edu, (dostępny 30.09.2025).

7. Spis rysunków