

## **Control Questions for the Subject "System Programming"**

1. Technical means of information processing. Physical implementation of the computer.
2. Structure of computer software and its components.
3. Basic concepts of system software.
4. Evolution of computing technology. Principles of operation of computer generations.
5. Basic components of computer architecture. Functional blocks of a computer.
6. Classical (Von Neumann) architecture of a computer. Basic model of computer architecture.
7. Basic devices of a computer and their characteristics.
8. Main functions of ports (input-output channels).
9. Main structure of computer software.
10. Software components that manage hardware resources and interact with the user.
11. Main functions of specialized system software.
12. Utility programs for servicing and configuring the operating system.
13. Composition and functions of system processing programs.
14. Low-level programming. Assembly language.
15. Network configuration editor. Software for setting network connection parameters.
16. Operating system loader. Program responsible for loading the operating system into the computer's memory.
17. Describe processors that support macro-commands.
18. Language translators. Programs that convert source code in a high-level language into machine code.
19. Programs for converting text data from one format to another. Language converters.
20. General information about editors. Software tools for creating and editing text files.

21. Debuggers for program code. Tools for finding and fixing errors in program code.
22. Decompiler of machine code. Program that performs reverse conversion of machine code to source code (as far as possible).
23. Cross-system. A system for developing software for another platform.
24. Library managers. Systems for managing software module libraries.
25. Structure and main components of a computing system.
26. Physical components of a computer. Material elements from which the computer is made.
27. Programming systems. A set of tools for software development.
28. Software that controls the computer.
29. Microprocessor control commands in assembly language.
30. Structure and architecture of a computer. Organization and interaction of computer components.
31. Computing system and its structure. The set of hardware and software designed for information processing.
32. Central processor and its registers. The main element of the computer that performs arithmetic and logical operations.
33. Processor instruction set. A set of instructions the processor can execute.
34. Interrupt handling mechanism. A way of responding to external events by the computer.
35. Addressing of registers and memory cells. A method for accessing data in the computer's memory.
36. Methods of organizing and using computer memory.
37. Data types and the amount of memory allocated for storing them.
38. Global and local memory. Memory areas with different accessibility for programs.
39. Static and dynamic memory. Types of memory that differ in allocation and deallocation methods.
40. General-purpose registers. User registers.
41. Registers for organizing segmented memory. Special registers for managing segmented memory.

42. Processor status register. An element of the processor that reflects the current state of calculations.
43. Special function registers. Special-purpose registers.
44. Code segment, data segment, stack segment, extra segment.
45. Hard disk control register. An element that controls the operation of the hard disk.
46. Memory access organization. Methods for reading and writing data to memory.
47. Physical address formation. The process of converting a logical address to a physical address.
48. Running the translator, linker, and debugger. The process of creating an executable file from source code.
49. Interrupt handling mechanism. A way of responding to external events by the computer.
50. Algorithm for reading from RAM. The sequence of operations when reading data from RAM.
51. Random access memory (RAM). Fast, volatile memory for storing current data and programs.
52. Classification of operating systems. Grouping of operating systems based on various features.
53. Features of operating system usage areas.
54. Process management as part of the operating system.
55. Processor life cycle. Phases of instruction execution by the processor from fetching to completion.
56. Process scheduling algorithms. Methods of organizing the queue for process execution in a multitasking system.
57. Functions of the scheduler. Managing the distribution of processor time between processes.
58. Inter-process synchronization and communication tools. Mechanisms for ensuring coordinated operation of multiple processes.
59. Memory allocation and release. Allocating and returning memory blocks to processes as needed.
60. Physical memory expansion mechanism. Ways to increase the effective amount of RAM.

61. Disk file structure. Organization of data on the disk in the form of files and directories.
62. Files and file system. The method of storing and managing information on a storage medium.
63. File allocation table. A data structure that contains information about the locations of files on the disk.
64. History of operating systems. The development of operating systems from the earliest versions to modern ones.
65. Classification of operating systems. Grouping of operating systems based on various features.
66. Classification of programming languages. Grouping of programming languages based on abstraction levels, paradigms, and other features.
67. General translation scheme. The process of converting source code into machine code.
68. Types of modern processors. Processor types used in modern computers.
69. Definition and purpose of a translator. A program that converts source code in one language to an equivalent code in another language.
70. Definition and purpose of a compiler. A program that converts source code into machine code.
71. Definition and purpose of an interpreter. A program that executes source code line-by-line without prior compilation.
72. General working scheme of a compiler. Stages of transforming source code into an executable file.
73. Process of source code transformation. Compilation, assembly, linking.
74. Main functions of a compiler. Lexical analysis, syntax analysis, code generation.
75. Main phases of compilation. Compiler stages: lexical analysis, syntax analysis, semantic analysis, code generation.
76. Multi-pass and single-pass compilers. Compilers that perform multiple or a single pass over the source code.
77. Definition of a language. Syntax and semantics. Rules for constructing programs and their meaningful interpretation.
78. Formal language and formal grammar. Mathematical models for describing languages.

79. Grammar notation in Backus-Naur form. A way to describe language syntax.
80. Chains of symbols and operations on them. Strings and operations with them.
81. Chomsky hierarchy. Classification of formal grammars.
82. Hierarchical classification of grammars. Classification of grammars by abstraction levels.
83. Classification of languages by expressive means. Grouping languages based on their ability to express algorithms.
84. Context-free grammars or grammars for describing programming languages.
85. Tasks of lexical analysis and approaches to their solution.
86. Service tables for lexical analysis. Tables for storing information about lexemes.
87. Organization of the identifier table. Storing information about variables and functions.
88. Principles of hash functions and hash addressing. Methods for fast element searching in a table by key.
89. Parsing problem. Syntax analysis. The process of constructing a parse tree for the source code.
90. Left-to-right and right-to-left derivations. Methods of constructing a parse tree.
91. Parse tree. A graphical representation of the syntactic structure of a program.
92. Main concepts of syntax analysis. Terms used in describing the syntax analysis process.
93. Purpose and functions of syntax analyzers. Checking the correctness of a program's syntax.
94. Syntax graph. A graphical representation of a syntactic structure.
95. Automation of syntax analyzer construction. Creating tools for automatic construction of syntax analyzers.
96. Classes of syntax analyzers. Top-down, bottom-up parsers.
97. Recursive descent method. An algorithm for constructing a syntax analyzer.
98. Shift-reduce method. An algorithm for constructing a parse tree.
99. Main concepts of finite automata. A mathematical model for describing computations.

100. Basic concepts of finite automata. Mathematical model for describing computations.

**Head of the Department of Systematic  
and Applied Programming**

**K.F.Kerimov**