Siberian Federal University

05.13.11 - Mathematical and software support for computers, complexes and computer networks

Course Guide

This course contributes to the requirements for the Degree of Candidate of Science in Computer Science.

Krasnoyarsk, 2020.

1. Course Description

This course contributes to the requirements for the Degree of Candidate of Science in Computer Science.

Title of the Academic Program	Postgraduate Programs in English "Mathematical and software support for computers, complexes and computer networks"
Type of the course	core /mandatory
Course period	2 semesters First semester: from October, the 1st to February, the 1st (18 weeks) Second semester: from February, the 1st to June, the 1st (18 weeks)
Study credits	6 ECTS credits
Duration	216 hours
Language of instruction	English
Academic requirements	− Master's Degree in Computer Science or equivalent (transcript of records), − good command of English (certificate or other official document) **Prerequisites:** − **Advanced knowledge** of math**,** digital electronics, programming skills.

1.1 Course overview

"Mathematical and software support of computers, complexes and computer networks» is a core course.

The course is designed to develop the skills of graduate students in scientific research in the field of modern trends and prospects for the development of informatics and computing technology. The discipline reflects the main achievements in the field of system and applied software for computer architectures and computer networks at various levels of the hierarchy. When considering the diversity of computer architectures, various options for constructing mathematical and software tools are considered. Special attention is paid to the development of new technologies and tools that ensure the creation of operating systems and large software systems.

1.2 Special features

The course enables graduate students to work personally and as part of groups for the development of mathematical and software for promising software systems and computing systems. A graduate student will be able to go all the way from the birth of a scientific idea to its implementation.

1.3 Course aims and objectives

Course Aim

The aim of the course is to training of highly qualified specialists who have mastered the achievements of science and advanced practice and are able to successfully carry out practical activities in the field of increasing the efficiency and reliability of processing and transmission of data and knowledge in computers, complexes and computer networks.

Course Objectives

- study of the most general patterns and development trends in the field of mathematical and software support for computers, complexes and computer networks.

1.4 Learning outcomes

By the end of the course, graduate students will know:

- methods of development and application of programming theory;
- methods of creating and maintaining software for various purposes;

- Methods and ways to improve the efficiency and reliability of processing and transmission of data and knowledge in computers, complexes and computer networks;
- research methods and experimental work in the field of software development.

By the end of the course, students will be able to:

- create models, methods and algorithms for design, analysis, verification and testing of programs and software systems;
- research programming languages and programming systems, program semantics, database and knowledge management systems, symbolic computing software systems, operating systems, human-machine interfaces; models, methods, algorithms and software for computer graphics, visualization, image processing, virtual reality systems, multimedia communication;
- explore models and methods for creating programs and software systems for parallel and distributed data processing;
- create languages and tools for parallel programming;
- create models, methods, algorithms and software infrastructure for organizing globally distributed data processing.

By the end of the course, students will possess:

- technologies for assessing the quality and standardization of software systems;
- the ability to apply: knowledge in the field of mathematical and software support for computers, complexes and computer networks;
- methods of expert assessment of various types of project assignments and development of recommendations for the creation of new and improvement of existing structures, mechanisms and models of mathematical and software for computers, complexes and computer networks, in order to increase the efficiency and reliability of their functioning;
- technologies for organizing and managing research and development and expert and analytical work using advanced knowledge in the field of mathematical and software support for computers, complexes and computer networks, including the fields of education, law, defense, health care and environmental protection.

2. Course Lecturer, Contact Information

Oleg V. Nepomnuashchiy,

Ph.D. in Engineering, Professor, Head of Computer Science Dept, School of Space and Information Technologies, Siberian Federal University.

(room ULK 3-12B) 26, Kirenskogo st, Krasnoyarsk, Russia

e-mail: ONepomnuashy@sfu-kras.ru

Google Scholar page: https://scholar.google.ru/citations?user=JxdeoasAAAAJ&hl=ru

Additional information is available at:

https://structure.sfu-kras.ru/node/2153

Tel: +7 391 291 2931

3. Prerequisites

A background in basic programming will help in faster and better understanding of every topic. Nevertheless, each part of the course includes a short introduction of methods that are required for its study. Therefore, a graduate student without the denoted experience must be encouraged to make some additional efforts in education.

4. Course Outline

Week	Lectures	Seminars/ Assignments	Hours Lec/Lab/HA
Semester 3			
1-6	Computer systems and networks. (Part 1).	Computing systems. Classification of computing systems by the way of organizing parallel processing.	5/8/36
6-12	Computer systems and networks. (Part 2).	Computing networks. Purpose, architecture and principles of building information - computer networks (ICS).	5/5/36
13-18	Programming languages and systems. Software development technology (Part 1).	Programming languages. Procedural programming languages.	8/5/36
Semester 4			
1-4	Programming languages and systems. Software development technology (Part 2).	Distributed programming Processes and their synchronization.	4/2/2
5-9	Programming languages and systems. Software development technology (Part 3).	Basics of constructing translators. The structure of the optimizing translator.	4/2/2
10-13	Operating Systems (Part 1)	The functioning of computing systems. Modes of functioning of computing systems, structure and functions of operating systems.	5/2/2
14-18	Operating Systems (Part 2)	Interaction of processes. Parallel processes, generation and control schemes. Organization of interaction between parallel and	5/3/3

		asynchronous processes: messaging, organization of mailboxes.	
	36	**27**	**180**
36	**Final Exam**		**36**

4.1 Course requirements

4.1.1 Web-page of the course

Course materials and required reading materials are available on the webpage of the **Elements and devices of computer technology and control systems**, SibFU E-learning portal, www.e.sfu-kras.ru. You must be logged in to access this course. https://e.sfu-kras.ru/course/view.php?id=27719

4.1.2 Required reading

The main book for this course is The ***Course Book***. It provides students with all the information they need to master methods and tools for research in science field.

1. Jon Stokes. Inside the Machine: An Illustrated Introduction to Microprocessors and Computer Architecture. 1st Edition. ARS Technica, 2010. p.320. SBN-13: 978-1593276683, ISBN-10:1593276680

2. Steve McConnell. Code Complete: A Practical Handbook of Software Construction. Microsoft press 2015 p. 855. ISBN-13: 978-0735619678

ISBN-10: 0735619670

3. Max Kanat-Alexander. Code Simplicity: The Fundamentals of Software. O'Reilly Media, Inc. 2012. p. 65 ISBN-13: 978-1449313890, ISBN-10: 1449313892

4. Donald Knuth. Art of Computer Programming, Volumes 1-4A. Addison-Wisley, 2016.

4.1.3 Course materials

The main book that will guide a student through the course is ***Mathematical and software support for computers, complexes and computer networks*** book. It contains all of topics of this course according to the schedule. It will provide you

with useful links at the end of each chapter that will help graduate students to improve their understanding of the topics.

4.1.4 Required feedbacks

Graduate students are free to contact the lecturer by email. The name of department and a number of a group should be written in the subject or in the beginning of the letter for convenience. More information on how to contact the lecturer can be found in «Lecturer information» section of this Guide.

Student's Home or Lab Assignment reports must be attached as a separate pdf file. Student's name and group number should be written on the first page of the file. It is recommended to insert Assembler or C code with short comments for key elements of the code. Students send this report in electronic form only before the deadline.

If necessary, the lecturer will schedule a video-conference, upon request.

4.2 Course Structure

Learning Activities	Hours
Lectures	36
Practice sessions / Seminars,	27
Self-study Assignments	117
Final Exam (including preparation)	36
Total study hours	**216**

4.3 Time schedule of the course and course outline

№	Theme	Week	Learning Activities	Hours	Home Assignment and Reading
Semester 3					
1	Computer systems and networks. (Part 1).	1-6	**Lecture 1** «Classification of computing systems by the way of organizing parallel processing. Multiprocessor and multicomputer systems. Computing clusters. Problem-oriented parallel structures: matrix computer systems, systolic structures, neural networks»	5	Course Book Chapter 1. Computer systems and networks. (Part 1). Answer the test questions on the topic in the e-course
			Lab 1 «matrix computer systems»	8	Design and debug application according to tasks. Choose a number of tasks in Course Book annex 1-1.
			Home assignment 1	36	Read the book: Max Kanat-Alexander. Code Simplicity: The Fundamentals of Software. Finish the Lab 1 and issue a report about.
2	Computer systems and networks. (Part 2).	7-12	**Lecture** 2 «Computing networks. Purpose, architecture and principles of building information - computer networks (ICS). Local and global IVS, hardware and software for combining various networks. Methods and means of data transmission in IVS, data transmission protocols Features of the architecture of local networks (Ethernet, Token Ring, FDDI). Internet network, domain organization, TCP / IP protocol family. Computer networks and distributed information processing»	5	Course Book: Chapter 2 Computer systems and networks. (Part 2). Answer the test questions on the topic in the e-course

№	Theme	Week	Learning Activities	Hours	Home Assignment and Reading
			Lab 2 «Ethernet»	5	Design and debug application according to tasks. Choose a number of tasks in Course Book annex 1-2.
			Home assignment 2	36	Read the book: Jon Stokes. Inside the Machine: An Illustrated Introduction to Microprocessors and Computer Architecture.. Finish the Lab 2 and issue a report about.
3	Programming languages and systems. Software development technology (Part 1).	13-18	**Lecture** 3 «Programming languages. Procedural programming languages. (Fortran, C), Functional Programming Languages (Lisp), logical programming (Prolog), object-oriented programming languages (Java). Procedural programming languages Basic control structures, program structure. Working with data: variables and constants, data types (boolean, integer, floating, character, range and enumeration types, pointers), data structures (arrays and records). Procedures (functions) procedure call, parameter passing (by reference, by value, by result), localization of variables, side effects. Exception handling. Libraries of procedures and their use»	8	Course Book: Chapter 3 Programming languages and systems. Software development technology (Part 1).. Answer the test questions on the topic in the e-course.
			Lab 3 «Procedural programming languages»	5	Design and debug application according to tasks. Choose a number of tasks in Course Book annex 1-2.
			Home assignment 3	36	Read the book: Steven F. Barrett, Daniel J. Pack. Microchip AVR® Microcontroller

№	Theme	Week	Learning Activities	Hours	Home Assignment and Reading
					Primer: Programming and Interfacing. Finish the Lab 2 and issue a report about.
Semester 4					
4	Programming languages and systems. Software development technology (Part 2).	1-4	**Lecture 4** «Distributed programming Processes and their synchronization. Semaphores, Hoare monitors. Object Oriented Distribution Lane programming CORBA. Parallel programming over shared memory. Threads. Oren MP standard interface. Parallelization of sequential programs. Parallel programming over distributed memory. Paradigms of RMV and MIMD. Standard MPI interface»	4	Course Book: Chapter 4 Programming languages and systems. Software development technology (Part 2). Answer the test questions on the topic in the e-course
			Lab 4 «Parallel programming»	2	Design and debug application according to tasks. Choose a number of tasks in Course Book annex 1-4.
			Home assignment 4	2	Read the books: Donald Knuth. Art of Computer Programming, Volumes 1-4A. Steve McConnell. Code Complete: A Practical Handbook of Software Construction Finish the Lab 4 and issue a report about.
5	Programming languages and systems. Software	5-9	**Lecture** 5 « Basics of constructing translators. The structure of the optimizing translator. Intermediate program representations: sequence of characters,	4	Course Book: Chapter 5 Programming languages and systems. Software development technology (Part 3). Answer

№	Theme	Week	Learning Activities	Hours	Home Assignment and Reading
	development technology (Part 3).		sequence of tokens, syntax tree, abstract syntax tree. Intermediate presentation levels: high, medium, low. Intermediate submission forms. Analysis of the source program in the compiler. Automaton (regular) grammars and scanning, context free grammars and parsing, organization of a program symbol table with a block structure, hash functions. Descending (LL (1) grammars) and ascending (LR (1) grammars) parsing methods. Attribute grammars and semantic programs, construction of an abstract syntax tree. Automatic construction of lexical and parsers from formal descriptions of grammars. Lex and uass systems. Gentle system. Optimization of programs during compilation. Optimization of basic blocks, cleaning of cycles. Analysis of control flow graphs and data flow. Dominance relation and its properties, building the border of the vertex dominance area, highlighting strongly connected components of the graph. Building a dependency graph. Translation of the program into SSA-representation and back. Global and interprocedural optimization. Generating object code in compilers. Retargetable Compilers, cc (Gnu Compiler Collection). Term rewriting Application of optimization heuristics (integer programming, dynamic programming) for automatic generation of object code generators (BEG systems, Iburg, etc.)»		the test questions on the topic in the e-course
			Lab 5 «Intermediate program»	2	Design and debug application according to tasks. Choose a number of tasks in Course

№	Theme	Week	Learning Activities	Hours	Home Assignment and Reading
					Book annex 1-5.
			Home assignment 5	2	Read the books: Donald Knuth. Art of Computer Programming, Volumes 1-4A. Steve McConnell. Code Complete: A Practical Handbook of Software Construction Finish the Lab 5 and issue a report about.
6	Operating Systems (Part 1)	10-13	**Lecture** 6 « The functioning of computing systems. Modes of functioning of computing systems, structure and functions of operating systems. Basic blocks and modules. Basic hardware support for operating systems (OS) functions: interrupt system, memory protection, address translation mechanisms in virtual memory systems, channel and peripheral device management. Types of processes and their control in modern operating systems. Representation of processes, their contexts, generation hierarchies, states and interactions. Multitasking (multi-program) operation mode. Process control commands. Means of interaction of processes. The client-server model and its implementation in modern operating systems. The working set of pages (segments) of the program, algorithms for its determination. Control of external devices»	5	Course Book: Chapter 5 Operating Systems (Part 1). Answer the test questions on the topic in the e-course
			Lab 6 «structure and functions of operating systems »	2	Design and debug application according to tasks. Choose a number of tasks in Course Book annex 1-6.

№	Theme	Week	Learning Activities	Hours	Home Assignment and Reading
			Home assignment 6	2	Read the books: Steve McConnell. Code Complete: A Practical Handbook of Software Construction. Max Kanat-Alexander. Code Simplicity: The Fundamentals of Software. Finish the Lab 6 and issue a report about.
7	Operating Systems (Part 2)	14-18	**Lecture** 7 «Interaction of processes. Parallel processes, generation and control schemes. Organization of interaction between parallel and asynchronous processes: messaging, organization of mailboxes. Critical sections, primitives of mutual exclusion of processes, Dijkstra semaphores and their extensions. Deadlock problem in asynchronous process execution, deadlock detection and prevention algorithms. Operational means of process control during their implementation on parallel and distributed computing systems and networks: standards and software PVM, MPI, OrenMP, POSIX. Single-level and multi-level disciplines of cyclic servicing of processes on the central processor, the choice of a quantum. Data access control. File system, organization, distribution of disk memory. Management of data exchange between disk and RAM. Working set of pages (segments) of the program, algorithms for its determination. Control of external devices»	5	Course Book: Chapter 7 Operating Systems (Part 2). Answer the test questions on the topic in the e-course
			Lab 7 « OpenMP»	3	Design and debug application according to tasks. Choose a number of tasks in Course Book annex 1-7.

№	Theme	Week	Learning Activities	Hours	Home Assignment and Reading
			Home assignment 7	3	Read the books: Steve McConnell. Code Complete: A Practical Handbook of Software Construction. Max Kanat-Alexander. Code Simplicity: The Fundamentals of Software. Finish the Lab 7 and issue a report about.
8	Final exam			36	Prepare to final exam. Preparation for answering exam questions (available at e-courses and course book). Preparation for solving control problems using the course book, main books and the e-course.

5.Assessment

Assessment strategy	Points, max	Evaluation criteria
Tests	10	Test questions for lectures in the e-course
Lab works	40	Lab report
Individual Project	40	Electrical schematics, code, report on the project, presenting the project
Final exam	10	2 questions and a practical task that require preparatory reading and knowledge of the concepts explained

Grade policy for final assessment is:

A (excellent work) 91–100 points

B (above average work) 81–90 points

C (average work) 71–80 points

D (below average work) 50–70 points

F (failed work) < 50 points

The final exam is oral and written test. Students should be able to:

• Answer two short theoretical questions;

• Develop a general algorithm for embedded software according to the assignment;

• Write a fragment of the program to initialize the built-in nodes of the microcontroller.

6. Attendance Policy

Graduate students are expected to attend classes regularly. In case of missing an in-lab activity a student should perform additional work submitted to the instructor within a week after a class was missed.

Every topic involves an assignment. A written report on the assignment should be submitted within two weeks from the moment students received a list of problems. The final mark will rely on the same grading policy as for the final exam.

7. Required Course Participation

There are no special requirements for the course participation. The preferred type of report submission is the electronic one. Students can use the web-version of the course (link) for a better progress. All problems for solution could be found there together with text from the course book.

8. Facilities, Equipment and Software

Facilities:

The auditorium is equipped with personal computers with Internet access, as well as a multimedia projector and an electronic board. The material and technical support of the discipline includes:

- library fund of GOU VPO "SFU"

- computer workplaces for laboratory studies and testers;

- multimedia equipment for giving lectures, showing presentations.

Software:

MS Office (MS Word, MS PowerPoint, MS Excel), Adobe Acrobat, Adobe Flash Player или KMPlayer, аудиопроигрыватель AdobeFlash до Winamp, Maxima, SciLab.

Annex 1 Example of Self-Study Assignment

The task: Describe the OpenMP execution model

Solution:

The OpenMP API uses the fork-join model of parallel execution. Multiple threads of execution perform tasks defined implicitly or explicitly by OpenMP directives. OpenMP is intended to support programs that will execute correctly both as parallel programs (multiple threads of execution and a full OpenMP support library) and as sequential programs (directives ignored and a simple OpenMP stubs library). However, it is possible and permitted to develop a program that executes correctly as a parallel program but not as a sequential program, or that produces different results when executed as a parallel program compared to when it is executed as a sequential program.

Furthermore, using different numbers of threads may result in different numeric results because of changes in the association of numeric operations. For example, a serial addition reduction may have a different pattern of addition associations than a parallel reduction. These different associations may change the results of floating-point addition.

An OpenMP program begins as a single thread of execution, called the initial thread. The initial thread executes sequentially, as if enclosed in an implicit task region, called the initial task region, that is defined by an implicit inactive parallel region surrounding the whole program.

When any thread encounters a parallel construct, the thread creates a team of itself and zero or more additional threads and becomes the master of the new team. A set of implicit tasks, one per thread, is generated. The code for each task is defined by the code inside the parallel construct. Each task is assigned to a different thread in the team and becomes tied; that is, it is always executed by the thread to which it is initially assigned. The task region of the task being executed by the encountering thread is suspended, and each member of the new team executes its implicit task. There is an implicit barrier at the end of the parallel construct. Beyond the end of the parallel construct, only the master thread resumes execution, by resuming the task region that was suspended upon encountering the parallel construct. Any number of parallel constructs can be specified in a single program parallel region may be arbitrarily nested inside each other. If nested parallelism is disabled, or is not supported by the OpenMP implementation, then the new team that is created by a thread encountering a parallel construct inside a parallel region will consist only of the encountering thread. However, if nested parallelism is supported and enabled, then the new team can consist of more than one thread.

When any team encounters a worksharing construct, the work inside the construct is divided among the members of the team, and executed cooperatively instead of being

executed by every thread. There is an optional barrier at the end of each worksharing construct. Redundant execution of code by every thread in the team resumes after the end of the worksharing construct.

When any thread encounters a task construct, a new explicit task is generated. Execution of explicitly generated tasks is assigned to one of the threads in the current team, subject to the thread's availability to execute work. Thus, execution of the new task could be immediate, or deferred until later. Threads are allowed to suspend the current task region at a task scheduling point in order to execute a different task. If the suspended task region is for a tied task, the initially assigned thread later resumes execution of the suspended task region. If the suspended task region is for an untied task, then any thread may resume its execution. In untied task regions, task scheduling points may occur at implementation defined points anywhere in the region. In tied task regions, task scheduling points may occur only in task, taskwait, explicit or implicit barrier constructs, and at the completion point of the task. Completion of all explicit tasks bound to a given parallel region is guaranteed before the master thread leaves the implicit barrier at the end of the region. Completion of a subset of all explicit tasks bound to a given parallel region may be specified through the use of task synchronization constructs. Completion of all explicit tasks bound to the implicit parallel region is guaranteed by the time the program exits.

Synchronization constructs and library routines are available in OpenMP to coordinate tasks and data access in parallel regions. In addition, library routines and environment variables are available to control or to query the runtime environment of OpenMP programs.

OpenMP makes no guarantee that input or output to the same file is synchronous when executed in parallel. In this case, the programmer is responsible for synchronizing input and output statements (or routines) using the provided synchronization constructs or library routines. For the case where each thread accesses a different file, no synchronization by the programmer is necessary.

Annex 2 Example of Pre-Course Test Questions

1. Algorithm concept. Equivalence of these formal models of algorithms. The concept of algorithmic undecidability.
2. Formal languages and ways of describing them. Classification of formal grammars. How to use in lexical and parsing?
3. Multiprocessor and multicomputer systems. Computing clusters. Problem-oriented parallel structures: matrix systems, systolic structures, neural networks.
4. Methods and means of transferring data to computer systems, data transfer protocols.
5. Features of the architecture of local networks (Ethernet, Token Ring, FDDI).
6. Internet network, domain organization, TCP / IP protocol family.
7. Distributed programming. Processes and synchronization. Object-oriented distributed programming Parallel programming by shared memory. Parallel programming by distributed memory.
8. Basics of constructing translators. Optimizing translator structure. Intermediate program representations. Intermediate presentation levels.
9. Analysis of the source program in the compiler. Automatic (regular) grammars and scanning, context free grammars and parsing, organization of a program symbol table with a block structure, hash functions. Automatic construction of lexical and parsers from formal descriptions of grammars.
10. Optimization of programs during compilation Optimization of basic blocks, cleaning of cycles. Analysis of control flow and data flow graphs Building a dependency graph. Global and interprocedural optimization.
11. Generation of object code (retargetable) compilers, Recycling terms optimization heuristics dynamic programming) for automatic generation of object code generators (systems BEG, Iburg, etc.).
12. Software development and maintenance technology. The life cycle of the program. Development stages, degree and ways of their automation. Modules, interaction between modules, hierarchical program structures.
13. Debugging, testing, verification and evaluation of the complexity of programs. Generation of tests. Test generation systems. Slices of programs (slice, chop) and their use when debugging programs and for generating tests.
14. Methods for the specification of programs. Schematic, structural, visual programming User interface development, multimedia interface interaction environments.

Annex 3 Outlines of Lab works

(List one. The title)

"SIBERIAN FEDERAL UNIVERSITY"
Institute of Space and Information Technologies
Department of Computer Science
Master's Degree Programs “Digital intelligent control systems”
Group No ***(Group identifier)***

REPORT ON LABORATORY WORK No. ***(Number of lab)***
Theme: ***(Theme of task).***

Tutor: ***(Tutor’s / Lecture’s Name and Surname).***

Student: ***(Student`s Name and Surname).***

Krasnoyarsk, 2020

(List two, etc. The progress)

Main aim: ***(Describe the aim of lab).***
The task: ***(Describe the task of lab).***
Solution: ***(short description (no more than 2-3 pages) of the problem solving process).***

Annex A Diagram(s)

(diagrams and graphs).

Annex B Code(s)

(source code Included comment).

Annex 4 Example of Final Oral Exam Questions

1. Types of processes and their management in modern operating systems. Representation of processes, their contexts, generation hierarchies, states and interactions. Multitasking (multi-program) operation mode. Process control commands. Means of process interaction.
2. Parallel processes, generation and control schemes. Organization of interaction between parallel and asynchronous processes: messaging, organization of mailboxes.
3. Operational means of process control during their implementation on parallel and distributed computing systems and networks: standards and software PVM, MPI, OrenMP, POSIX.