Program	Level	Second cycle
	Name of the program	Theoretical Computer Science

COURSE

Course title	**System Programming and System Software**			
Course code	Semester	Course status	ECTS	Contact hours (L+AE+LE)
CS420	I	Mandatory course	7	3+0+2
Lecturer				
Course Goals	The aim of the course is to introduce students with many common procedural languages, as well as representational functional, logic-oriented and object-oriented languages.			
Learning Outcomes	Upon successful completion of this course, students are expected to be able to: - identify and explain the functions of primary CPU components such as registers, ALU, control unit, memory, input-output devices and typical microprocessor instructions. - demonstrate the ability to write simple programs in assembly language - explain the process of translating programs from high-level languages to low-level languages - understand the code generation and optimization process in the production of low-level programming code			

COURSE CONTENT

A programmer's view of processor organization. Concept of memory and memory addresses. Registries. Program counter. Intel IA-32 architecture processor instructions. Addressing data at the system level: Approach data in the registers. Constants. Direct and indirect addressing. Index addressing. Access data across the stack. Linear memory and its alternatives (segments, pages). Machine code and its generation: Assembly and binary representation of instructions. Data transfer instructions. Instructions for arithmetic and logical operations. Unconditional jump instructions. Conditional jumps. Stack. Subroutines. Shifting and rotating. Floating point. Input and output: Memory and I/O mapped input and output. Principle of operation of keyboard, disk, screen, communication devices at low level and API level of operating systems. Interrupts/events and their service routines: Interrupt table. Hardware interfaces. Software traps. Processor exceptions. Data storage during service routine processing. The most important routines. Compilers. A simple compiler. Representation of syntax diagrams by syntax procedures. Code generation: memory, stack, global variables, dynamic and static data, code generation from the compiler. Realization of expressions, operators, procedures, local and global variables, program structures. Builders, linkers: Principle of linker operation. Make bilder. Assembly principle, one-pass and two-pass. Execution environment: Loaders, executable file format, role of registers, system functions, static and dynamic libraries. Virtual machines. Concurrency control techniques: Parallel execution, threads, semaphores, mutual exclusion, Performance evaluation and optimization: Profilers. Benchmark programs. Evaluation of algorithms

LITERATURE

[1] S. Ribić, Skripta sa tekstom predavanja dostupna na web stranici i u štampanom obliku
[2] IA-32 Software developers manual, Intel corporation
[3] Paul A. Carter: PC Assembly Language (www.drpaulcarter.com/pcasm/)
[4] R.E. Bryant and D. R. O'Hallaron: Computer Systems: A Programmer's Perspective, Prentice Hall, 2003,.
[5] Andrew S. Tanenbaum: Structured Computer Organization, 4th ed., Prentice Hall, 1999

STUDENT WORKLOAD (hours in a semester)

Lectures	45	Tutorial	30	Individual work	100	T o t a l	175

GRADING			REMARKS
Criterion	Maximum points	Minimum points	2x20 points written tests, remaining 10 points are earned for work during the semester. 5 homeworks worth 2 points each.
Midterm exams	40	20	
Homework assigments	10		
Final exam	40	10	
T o t a l	100	55	