National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

Department of Computer Engineering

SYSTEM PROGRAMMING

Syllabus

Requisites of the Course	
Cycle of Higher Education	***First cycle of higher education (Bachelor's degree)***
Field of Study	*12 Information Technologies*
Speciality	*121 Software engineering*
Education Program	*Computer Systems Software Engineering*
Type of Course	*Normative*
Mode of Studies	*full-time*
Year of studies, semester	*2 year (4 semester)*
ECTS workload	*4.5credit (ECTS). Time allotment - 135 hours, including 54 hours of classroom work, and 81 hours of self-study.*
Testing and assessment	*4 semester – Exam*
Course Schedule	*3 classes per week by the timetable http://rozklad.kpi.ua/*
Language of Instruction	*English*
Course Instructors	PhD, Associate Professor, Valerii Pavlov, pavlovvg@ukr.net
Access to the course	

Outline of the Course

1. Course description, goals, objectives, and learning outcomes

*The academic discipline "**System programming**"belongs to the mandatory (regulatory) educational components of the education program, namely to the professional training cycle.It has the code **PM 14** in the list of components of the educational program.*

***Reason and motivations for studying**: the need to understand the principles of programmatic control of the Central Processor Unit (CPU) directly. This discipline will give knowledge of how the interaction of Hardware and Software components is carried out in the Computer System.That is, it combines into a single complex knowledge of the architecture of computer systems and their programming.*

***The goal of the "System programming" course is**: train the basic architecture and programming environment of Intel 64 and IA-32 processors, the data-structure formats for hexadecimal and binary numbers of integer and floating-point values, the instruction set of the processor and the opcode structure, fundamental of low-level programming on Assembler language.*

***The purpose of the discipline** is the formation of a number of competencies among students, namely:*

***ABILITY**:*

- *to understand the principles of processor control at the software level;*
- *to understand the sequence of actions during the development of a program in machine language;*
- *to analyze the structure of the program in machine language;*
- *to develop the programs on Assembler language;*
- *to analyze the processes that are carried out during the compilation and linking of programs;*

- *to use macro-assemblers opportunities in programming;*
- *to develop mobile systems, embedded systems and real-time systems (PC16);*
- *to develop and use methods and algorithms of high-performance computing (PC17) by managing memory resources.*

1.2. The main tasks of the discipline.

After mastering the academic discipline, students must demonstrate such learning outcomes:

***KNOWLEDGE**:*

- *machine language command structures and formats;*
- *basic storage formats of data according to IEEE 754 -2008;*
- *Assembler program structures;*
- *know and put into practice the fundamental concepts, paradigms and basic principles of functioning of language, instrumental and computing toolsof software engineering (PLO7);*

***SKILL**:*

- *to determine the sequence of actions for the development of a system program on Assembler language;*
- *to use integrated development environments (IDE) of system software;*
- *to analyze messages to the compiler during debugging programs;*
- *to analyze a program listing in machine language;*
- *to search and correct syntax and logical errors in the system program;*
- *motivated to choose programming languages and development technologies to solve the problems of creating and maintaining software (PLO15);*
- *apply methods of component software development (PLO18).*

2. Prerequisites and post-requisites of the course (the place of the course in the scheme of studies in accordance with curriculum)

Interdisciplinary Connections: To successfully study the «System Programming» course, students must master the material and have certain knowledge, skills and abilities in such disciplines: «History of Science and Technique», «Computer Discrete Mathematics», «Computer Systems and Networks Fundamentals», «Programming Fundamentals», «Object-Oriented Programming», also have a basic level of English proficiency not lower than A2.

Knowledge and skills acquired during the study of the discipline «System Programming» can be used in the future when mastering the following courses:«Software Engineering Components»,«Operating Systems», «Safety of Software»,as well as during course and diploma design.

3. Content of the course

Section 1.Basic concepts of the Assembler language.

Topic 1.1.The processor as a programming object.

- *The place of the Assembler language in the classification of programming languages.*
- *The structure of the processor according to the von Neumann concept.*
- *Chronology processor architecturesof INTEL and AMD.*
- *Principles of organization and modes of use of computer memory.*
- *x86-64 microprocessor register memory organization.*

Topic 1.2. IEEE 754-2008 Standard for number representation.

- *Unsigned integers.*
- *Signed integers.*

- *Floating-point numbers.*
- *Binary-decimal number format BCD.*

Topic 1.3. Processor x86 instruction formats.

- *General command structure in machine language.*
- *The concept of operation code.*
- *The concept of a command prefix.*
- *Varieties of addressing in commands.*
- *Dependence of the command format on the type of addressing.*
- *Mod/RM field.*
- *SIB field.*
- *Assembler command recording formats.*
- *Assembler command code invariance.*

Section 2.Assembler programming.

Topic 2.1. Developing programs in the Assembler language.

- *Stages of program development.*
- *Compiler operation modes.*
- *Linker operation modes.*
- *Using debuggers and disassemblers.*
- *Assembler program structure.*
- *Compiler Directives.*
- *Flags register and main flags.*

Topic 2.2. Using macro-assemblers.

- *Macros and macro declaration.*
- *Macro variables and macro calculations.*

Topic 2.3. Using Procedures and Subroutines.

- *Stack Memory Principles.*
- *Internal and external procedures.*
- *Ways to pass parameters to procedures and return them from a procedure.*
- *Recursive procedure call.*
- *Developing Dynamic Link Libraries.*

4. Bibliography

4.1 Basic:

1. Kip R. Irvine.Assembly Language for x86 Processors. – Florida International University: Pearson, Seventh Edition 2014, - 876p.

2. Randall Hyde. The art of Assembly Language. – San Francisco: No starch press, 2nd Edition 2010, - 764p.

3. Barry Kauler. Windows Assembly Language & Systems Programming. – Lawrence: R&D Books, 2nd Edition1997, - 421 p.

4. Richard C. Detmer. Assembler. Introduction to 80x86 Assembly Language and Computer Architecture – Mississauga, :Jones and Bartlett Publishers, 2001, - 419p.

5. Paul A. Carter. PC Assembly Language – 2004, - 195p.

4.2. Supplementary:

1. Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manual. Volumes 1–3 [Electronic resource]:– 2014. http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

2. Microsoft Portable Executable and Common Object File Format Specification. [Electronic resource]: – http://www.osdever.net/documents/PECOFF.pdf.

3. Jeff Duntemann.Assembly Language Step-by-Step. – Indianapolis: Wiley, 3rd Edition 2009, - 646p.

4. Charles W. Kann. Introduction to MIPS Assembly Language Programming. – Gettysburg College, 2015, - 179p.

5. Dennis Yurichev. Reverse Engineering for Beginners. dennis@yurichev.com:, 2013, - 1083p.

Educational content

5. Methodology

Section and topic titles	Number of hours				
	Total	including			
		Lectures	Practical (Seminars)	Laboratory (Computer Workshop)	Self-study
Section 1. Basic concepts of the Assembler language					
Topic1.1. The processor as a programming object.	8	4	–	–	4
Topic 1.2.IEEE 754-2008 Standard for number representation.	14	4	–	2	8
Topic 1.3.Processor x86 instruction formats.	44	10	–	6	22
Test 1	1	–	–	–	1
Amount by section 1	61	18	–	8	35
Section 2 Assembler programming					
Topic 2.1. Developing programs in the Assembler language.	20	6	–	2	10
Topic 2.2. Using macro-assemblers.	20	6	–	2	10
Topic 2.3. Using Procedures and Subroutines.	36	6	–	6	18
Test2	1	–	–		1
Amount by section 2	67	18	–	10	39
Exam Preparation	7	–	–	–	7
Totalhours	135	36	–	18	81

Labs

The main tasks of the Labs cycle are the receipt by students of the necessary practical skills in the development of separate modules for controlling computing processes, ensuring the performance of special functions on Assembler and low-level programming.

Nin/o	Name of laboratory work (computer workshop)	Number of classroom hours
1	Internal representation of Integers and Floating-point numbers in the ix86 processor.	4
2	Investigation of the structure of COM files.	4
3	investigation of the structure of EXE files	4

4	*Using Macros in a MASM32 Environment*	*4*
5	*Arithmetic and Logical with Integers.*	*4*
6	*Using Coprocessor Commands.*	*4*
7	*Modular programming. Using procedures.*	*4*
8	*Developing and Using Dynamic Libraries.*	*8*
	Total:	*36*

6. Self-study

N in/o	***Names of topics and questions that are submitted for independent study***	***Number of hours of self-study***
1	*Performance of tasks on the topic of each Lecture session (see chapter 5) – 1 hourfor 1 hour of lecture*	*36*
2	*Preparation for Laboratory Classes (see chapter 5) – 1 hourfor 1 hour ofLaboratory Work*	*36*
3	*Preparation for Tests (see chapter 6) – 3 hours for each control*	*2*
4	*Exam Preparation*	*7*
	Total:	*81*

Policy and Assessment

7. Coursepolicy

When counting and evaluating laboratory work, the following factors are taken into account:

- *Completeness of the task on laboratory work on the individual variant;*
- *Timeliness of laboratory work according to the schedule;*
- *Autonomy of laboratory work and no indications of plagiarism;*
- *Answers to questions on the content of laboratory work during its protection.*

When evaluating control works, next consideration is taken into account:

- *Correctness and completeness of tasks;*
- *Number of completed tasks in conditions of limited time;*
- *Autonomy of tasks and no indications of plagiarism;*
- *Number of attempts to run controls that precede the one that is estimated.*

To prepare for the tests students receive a list of theoretical questions and the content of typical exercises that will be in the tasks on the test.

At the first and second attestation, the number of laboratory works and tests that were passed at the time of the attestation is taken into account.

8. Monitoring and grading policy

The system of assessing the grading policy in the discipline "System Programming" is based on the "Regulations on the system of assessment of learning outcomes in the «Igor Sikorsky Kyiv Polytechnic Institute»(**https://document.kpi.ua/files/2020_1-273.pdf**), namely the Rating System of Assessment (**RSA**) of the second type (**RSA-2**).

RSA-2 *consists of two components:*

- *starting(**R_S**);*
- *examination (**R_E**).*

*Starting points are formed as the sum of points that are obtained as a result of current control activities (laboratory works (**R_L**) and tests (**R_T**)), incentive (**R_I**) andpenalty (**R_P**) points:*

$$R_S = R_L + R_T + R_I + R_P,$$

*where R_L for 8 laboratory works is **8 X 5 = 40**points,*

*R_T for 2 tests is **2 X 15 = 30**points.*

*Thus, the maximum amount of base sum of starting points is **40 + 30 = 70** points*

*Behind the main assessment scale there are incentive and penalty points, which are taken into account in the total amount of points, but are not included in the main **RSA** scale.*

Incentive points take into account the answers to questions and the performance of tasks in lecture classes, the quality of the notes.

Penalty points are provided for late performance of laboratory work, that is, with a delay relative to the schedule.

*Thus exam scores are a maximum of **30** points, but for admission to the exam, the student must have at least **60%** of the maximum sum of starting points, which is*

***70 x 0.6 = 42**points.*

*After passing the exam, the starting points**R_S** are summed up with exam points **R_E**. Assessment of learning outcomes is carried out on a 100-point scale with further conversion of grades into a university scale in accordance with the table:*

Score	***Grade***
100-95	***Excellent***
94-85	***Very good***
84-75	***Good***
74-65	***Satisfactory***
64-60	***Sufficient***
Below 60	***Fail***
Course requirements are not met	***Not Graded***

Syllabus of the course

Is designed by teacher PhD, Associate Professor, Valerii Pavlov

Adopted by Department of Computing Technics (protocol #10, May 25, 2022)

Approved by the Faculty Board of Methodology (protocol #10, June 09, 2022)