**SYLLABUS**

**Fall semester 2022-2023 school year**

**on the educational program "6B06102 - Information Systems"**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Discipline’s code** | **Discipline’s title** | **Independent Student Work (IWS)** | **Number of loans** | | | | **Number of credits** | **Independent work of student with teacher (IWST)** |
| **Lectures (L)** | **Practical training (PT)** | | **Laboratory (Lab)** |
| PP 4303 | Parallel programming | 6 | 15 | 0 | | 30 | 3 | 6 |
| **Academic Course Information** | | | | | | | | |
| **Form of education** | **Type of course** | **Types of lectures** | | | **Types of practical training** | | **Form of final control** | |
| Full-time | Theoretical, practical | Problem-oriented | | | Learn how to program in parallel on processors and video cards using Python | | Written exam/Test | |
| **Lecturer - (s)** | Karyukin Vladislav Igorevich | | | | | |  | |
| **e-mail:** | [vladislav.karyukin@gmail.com](mailto:vladislav.karyukin@gmail.com) , [vladislav.karyukin@kaznu.kz](mailto:vladislav.karyukin@kaznu.kz) | | | | | |
| **Phone:** | +77019405992 | | | | | |

|  |  |  |
| --- | --- | --- |
| **Purpose of discipline** | **Expected Training Results (RO) \***  As a result of the study of the discipline, the student will be able to: | **Indicators of RO achievement (ID)**  (for each thrust reverser not less than 2 indicators) |
| This course is aimed at studying parallel programming techniques on processors and video cards using the Python language | 1. (cognitive) Know the theoretical and methodological concepts of parallel programming | 1.1 ability to create basic and advanced programs using parallelisms |
| 1.2 Know the peculiarities of working with cores and memory |
| 1.3 Knowledge of Concurrency Application Development Methods |
| 1. (functional) Apply knowledge of mpi4py, Celery, PyCUDA, and NumbaPRO libraries | 2.1 Develop programs for parallel computing |
| 2.2 Using Libraries for Parallel Computing |
| 2.3developing multifunctional applications that both developers and users understand |
| 1. **(**functional) Developing programs of various complexity levels: from a simple console to a product of academic and industrial importance. | 3.1 Be able to implement program code using parallelism |
| 3.2 Be able to configure applications |
| 3.3 Be able to create applications on Python |
| 1. Building complex, multifunctional applications | 4.1 Creating Parallel Computing Modules |
| 4.2 constructing interaction of various structural elements between each other |
| 4.3 Modifying and Editing Applications |
| 5. Creating Parallel Computing Applications on the GPU | 5.1 Create a New Application |
| 5.2 Implement Parallel Computing on Video Cards |
| 5.3 Test Applications |
| **Prerequisites** | Programming on Python | |
| **Post requisites** | Parallel programming on CUDA | |
| **Information resources \*\*** | **Literature:**  **Basic:** Python parallel programming cookbook by Giancarlo Zaccone. Packt publishing, 2015.Python for Everybody: Exploring Data in Python 3 by Dr. Charles Russell Severance, Sue Blumenberg, Elliott Hauser, Aimee Andrion, 2016Python Cookbook: Recipes for Mastering Python 3 3rd Edition, Kindle Edition by David Beazley, Brian K. Jones, 2013Advanced Python Development: Using Powerful Language Features in Real-World Applications 1st ed. Edition by Matthew Wilkes, 2021 **Additional:**   1. Natural Language Processing with Python and spaCy: A Practical Introduction, Yuli Vasiliev, 2021 2. Learning Scientific Programming with Python, Christian Hill, 2021   **Resources:**  **Software and Internet Resources:**  Python IDE, Anaconda Navigator Python, Microsoft Visual Studio, PyCharm, Microsoft Office Word, WinRAR, WordPad, Power Point, Adobe Reader, Paint.  **Online accessibility**: additional training materials, homework and projects can be found in UMCD on univer.kaznu.kz. | |

|  |  |
| --- | --- |
| **Academic policy of the course in the context of university moral and ethical values** | **Rules of academic behavior:**  1. You should prepare for each class in advance according to the schedule below. The task must be completed before the class in which the topic is discussed.  2. Academic values:  1. Laboratory classes and CCF shall be performed on their own  2. Plagiarism, forgery of documents, the use of cheats, writing off at all stages of knowledge control are unacceptable.  Students with disabilities can receive counseling via e-mail - [vladislav.karyukin@gmail.com](mailto:vladislav.karyukin@gmail.com) |
| **Evaluation and Appraisal Policy** | **Evaluation criteria**: evaluation of training results by descriptors (verification of competencies formation on intermediate control and exams).  **Final evaluation:** evaluation of activity in lessons, evaluation of completed task. |

**Training course content implementation calendar (schedule)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **Topic Title** | **Number of hours** | **Max.**  **score \* \* \*** |
| **1 Foundations of parallel computing** | | | |
| 1 | **L 1.** Getting started with parallel computing | 1 | 0 |
| **Lab 1.** Python in a parallel world | 2 | 10 |
| 2 | **L 2.** Memory organization | 1 | 0 |
| **Lab 2.** Thread-based Parallelism | 2 | 10 |
| **IWST 1.** Consultation on the implementation of IWS 1 | 1 | 0 |
| 3 | **L 3.** Parallel programming models | 1 | 0 |
| **Lab 3.** Thread synchronization with Lock and RLock | 2 | 10 |
| **IWS 1.** Implementation of project with parallel operations in Python | 0 | 15 |
| 4 | **L 4.** Python in parallel programming | 1 | 0 |
| **Lab 4.** Thread synchronization with a condition | 2 | 10 |
| **IWST 2.** Test | 1 | 15 |
| 5 | **L 5.** Thread-based parallelism | 1 | 0 |
| **Lab 5.** Process-based parallelism | 2 | 10 |
| **Module 2 Multiprocessing with Python** | | | |
| 6 | **L 6.** Thread synchronization with Lock and RLock | 1 | 0 |
| **Lab 6.** Using a process in a subclass | 2 | 10 |
| 7 | **L 7.** Multiprocessing | 1 | 0 |
| **Lab 7.** Using the mpi4py Python module | 2 | 10 |
| **IWST 3.** Consultation on the implementation of IWS 2. | 1 | 0 |
| **BC 1** |  |  | 100 |
| 8 | **L 8.** Subclass processes | 1 | 0 |
| **Lab 8.** Distributed python | 2 | 10 |
| **IWS 2.** Creating the application with distributed processes | 0 | 5 |
| 9 | **L 9.** mpi4py Python module | 1 | 0 |
| **Lab 9.** Scientific computing with SCOOP | 2 | 10 |
| 10 | **L 10.** Distributed python | 1 | 0 |
| **Lab 10.** GPU programming with Python | 2 | 10 |
| **IWST 4.** Test | 1 | 5 |
| **Module 3 GPU programming** | | | |
| 11 | **L 11.** GPU programming with python | 1 | 0 |
| **Lab 11.** Matrix multiplication with PyCUDA | 2 | 10 |
| 12 | **L 12.** PyCUDA | 1 | 0 |
| **Lab 12.** GPU programming with Numba | 2 | 10 |
| **IWST 5.** Consultation on the implementation of IWS 3. | 1 | 0 |
| 13 | **L 13.** PyCUDA matrix multiplication | 1 | 0 |
| **Lab 13.** Using the PyOpenCL module | 2 | 10 |
| **IWS 3** Developing the parallel application with CUDA | 0 | 10 |
| 14 | **L 14.** NumbaPRO | 1 | 0 |
| **Lab 14.** Evaluating element-wise operations with PyOpenCL | 2 | 10 |
| **IWST 6.** Test | 1 | 5 |
| **15** | **L 15.** PyOpenCL application | 1 | 0 |
| **Lab 15.** Creating the parallel computing application | 2 | 5 |
| **IWST 7. Consultation on preparation for examination questions.** | 1 | 0 |
| **BC 2** | |  | **100** |

**Dean \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Urmashev B.A.**

**Head of the Department of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Mussiraliyeva Sh. Zh.**

**Lecturer \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Karyukin V.I.**