COURSE DESCRIPTION CARD - SYLLABUS

Course name

PO 2.5.1 Programowanie równoległe i procesy graficzne - EC 2.5.1 Parallel programming and graphic processes

Course

Field of study Teleinformatics	Year/Semester 1/2
Area of study (specialization)	Profile of study general academic
Level of study second-cycle studies	Course offered in Polish
Form of study full-time	Requirements elective

Number of hours

Lecture 30	Laboratory classes 15	Other (e.g. online)
Tutorials 0	Projects/seminars 0/0	

Number of credit points

4

Lecturers

Responsible for the course/lecturer:	Responsible for the course/lecturer:
dr hab. inż. Olgierd Stankiewicz	

Prerequisites

1. Knows basic data structures and algorithms used in programming languages.
2. Has a working knowledge of programming methodologies and techniques in high-level languages.
3. Has basic knowledge of digital signal processing methods in data communications.
4. Is able to acquire information from literature, databases and other sources in Polish or English.

5. Is able to use programming mechanisms and programming environments of object-oriented languages and available library software.
6. Knows the limitations of his/her own knowledge and understands the need to update it.
7. Understands the influence of own work on the team results and the necessity of obeying the rules of teamwork and taking responsibility for the tasks performed together.

Course objective

Learning the basic features of parallel programming. To get acquainted with the existing technical solutions concerning the methods of designing parallel processing algorithms.
Prepare your own implementations of selected algorithms.
Shape and develop the ability to gain knowledge of current developments in parallelized computing.

Course-related learning outcomes

Knowledge

Understands the capabilities and limitations of GPUs, parallel programming techniques and knows basic techniques of computation parallelization. Knows advanced data structures and algorithms used in programming languages and has a working knowledge of programming methodologies and techniques in high-level languages.

Has advanced knowledge of the design of parallelized algorithms. He knows basic libraries and programming languages used to implement parallelized algorithms. Has advanced knowledge of object-oriented design and programming, architecture of object-oriented programming systems and basic object-oriented libraries in various programming languages, including libraries enabling programming of mobile terminals; has extended knowledge of project management.

Knows the correct terminology for parallel processing algorithms and graphics processors. Has an expanded English vocabulary of data communications and technology.

Skills

Can evaluate whether a given algorithm can be implemented using parallel computing techniques. Can critically evaluate the gains and losses of applying techniques of parallel and GPU programming to solve a specific problem. Able to think critically and argue a position.

Able to independently acquire knowledge of parallelized programming techniques and GPU programming. Able to self-educate.

Be able to retrieve data from the literature on parallelized programming techniques. Can use norms and standards of languages used in programming of graphic processors. Can obtain data from literature, databases and other sources in Polish or English, analyze standardization recommendations, integrate obtained information, interpret it, draw conclusions and formulate and justify opinions.

Be able to use advanced techniques for parallelizing computations. Be able to express algorithms in GPU programming languages. Be able to use advanced computational algorithms, data structures and high-level programming languages to solve technical problems related to data communications.

Able to use standard libraries of parallelized programming, including c++ language standard libraries. Is able to use libraries of graphics processor programming languages. Is able to use advanced programming mechanisms, programming environments of object oriented languages, available library software including programming application interfaces.

Be able to design and fully execute software expressed in whole or in part in a GPU programming language. Be able to design and fully execute software using parallelized programming techniques to increase performance. Be able to design and execute entirely software according to the art of software engineering in solving simple technical problems, be able to apply software engineering principles to solve part of a complex computer project.

Social competences

Is aware of a wide range of technical solutions for parallel processing and their continuous development. Knows the limitations of his/her own knowledge and understands the necessity of updating it. Is open to possibilities of continuous education and improvement of professional, personal and social competences.

Understands that parallel programming, including GPU programming, often requires teamwork. Understands the impact of own work on the team results and the necessity of submitting to team work rules, taking responsibility for tasks performed together, sees benefits from the exchange of experiences also in multicultural environment.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Lectures: verification of the assumed learning outcomes is realized by the assessment of knowledge shown in the test. The test consists of answering questions and solving problems.
A minimum score of 50% is required to receive a grade of 3.0; 3.5 - 60%; 4.0 - 70%; 4.5 - 80%; 5.0 - 90%.

Laboratories: on the basis of evaluation of the current progress of the tasks and activity in class.

Programme content

Parallel processing:
- Parallelization of tasks that do not require synchronization,
- Parallelization of tasks that require synchronization,
- Barriers,
- Synchronization points,
- Data access collisions,
- Standard thread library (std::thread).

Multithreaded Programming:
- CUDA,
- Compute Shader Language - OpenGL,
- OpenCL,
- OpenMP.

Laboratory classes will consist of the preparation by students of programs implementing selected algorithms of parallel processing, including on graphic processors, together with experimental verification of their correctness.

Teaching methods

1. Lecture: multimedia presentation, supplemented with current examples and additional explanations on the blackboard.
2. Laboratories: solving tasks, programming.

Bibliography

Basic

Z. Czech, "Introduction to parallel computing," PWN, Warsaw 2013.
Foster I., "Designing and Building Parallel Programs," book available online at http://www-unix.mcs.anl.gov/dbpp
M. Herlihy, N. Shavit "The Art of Multiprocessor Programming" Elsevier, 2008 (Polish edition "Sztuka programowania wielooprocesorowego", PWN 2010)

Additional

Grama A. et al, "Introduction to Parallel Computing" (2nd ed.), Addison-Wesley, 2003
Websites: www.openmp.org, www.mpi-forum.org

Breakdown of average student's workload

	Hours	ECTS
Total workload	86	4.0
Classes requiring direct contact with the teacher	45	2.0
Student’s own work (preparation for tests, preparation for laboratory classes, literature studies)	41	2.0