COURSE DESCRIPTION

1. Program information

1.1 University	**Alexandru Ioan Cuza University of Iaşi**
1.2 Faculty	**Faculty of Computer Science**
1.3 Department	**Department of Computer Science**
1.4 Study domain	**Computer Science**
1.5 Study cycle	**Bachelor**
1.6 Study program / Qualification	**Bachelor in Computer Science**

2. Course Information

2.1 Course name			**Multiprocessor Programming Techniques**				
2.2 Course teacher			**Emanuel Onica**				
2.3 Seminar teacher			**Emanuel Onica, Paul Diac**				
2.4 Year of study	**3**	**2.5** Semester	**1**	**2.6** Evaluation type	**E**	**2.7** Discipline status*	**OP**

** OB – Mandatory / OP – Optional*

3. Total estimated hours (hours per semester and didactic activities)

3.1 Hours per week	**4**	In which: **3.2** course	**2**	**3.3** seminar/laboratory	**2**
3.4 Hours in curriculum	**56**	In which: **3.5** curs	**28**	**3.6** seminar/laboratory	**28**

Time distribution	hours
Manual study, course support, bibliography, and others	**14**
Suplementary documentation in the library, electronic forums, and on the field	**14**
Seminaries/laboratories preparation, homeworks, reports, portfolios, and essays	**28**
Tutoring	**0**
Evaluation	**4**
Other activities	**0**

3.7 Total hours individual study	**56**
3.8 Total hours per semester	**116**
3.9 Credits	**5**

4. Preconditions (if necessary)

4.1 Curriculum	**Advanced Programming, Data Structures, Operating Systems**
4.2 Competencies	**Programming skills using the Java language. Basic knowledge of multithreading. Basic knowledge of data structures.**

5. Conditions (if necessary)

5.1 For course operation	The course will mainly take place in physical format (face-to-face). Online operation will be partially considered depending on opportunity, necessity and specific means availability, and will not account for more than a maximum of 22% of the total course time.
5.2 For seminary/laboratory operation	The laboratory will mainly take place in physical format (face-to-face). Online operation will be partially considered depending on opportunity, necessity and specific means availability, and will not account for more than a maximum of 22% of the total laboratory time.

6. Specific skills acquired

Professional skills	**C1.** Capacity of writing and optimizing computer programs that leverage the use of multiple cores. **C2.** Comprehensive knowledge of specific data structures used in multiprocessor programming and of various advanced synchronization mechanisms. **C3.** Capability of using various packages dedicated for muliprocessor programming available in the Java language.
Transversal skills	**CT1.** The ability to design and implement software using concurrent code for providing vertical scalability. **CT2.** Understanding the necessity of a memory model for a programming language and of specific notions related to the correctness of programs designed for multiple processors.

7. Course objectives

7.1 General objective	The main objective is obtaining the necessary skills on using specific techniques for multiprocessor programming.
7.2 Specific objectives	After taking this course, students will be able to : ▪ Explain notions specific to multiprocessor programming: theoretical conceps like consistency models or guarantees (linearizability, sequential consistency, wait-freedom, lock-freedom), various synchronization mechanisms (types of locks, barriers, etc.), data structures specifically optimized for concurrent access (lists, queues, stacks, etc.), advanced aspects (programming using transactional memory) ▪ Design and implement application modules optimized for using multiple cores ▪ Use advanced techniques for multiprocessor programming in at least one programming language (Java)

8. Contents

8.1	**Lecture**	**Teaching methods**	**Observations** (hours and bibliography)
1.	Introduction. Basic notions recap – from mutual exclusion to producer/consumer and readers-writers. Specific guarantees in multiprocessor programming the importance of parallel execution (Amdahl’s law).	Slides, blackboard	2 hours
2.	Concurrent execution – general concepts. Deadlock-freedom vs. Starvation-freedom in the context of mutual exclusion. Brief introduction of the Java memory model.	Slides, blackboard	2 hours
3.	Concurrent objects – general concepts. Consistency models: sequential consistency, linearizability. Progress conditions: wait-freedom, lock-freedom.	Slides, blackboard	2 hours

4.	Locks – I. General aspects. The contention factor. TAS lock vs. TTAS lock. Queue-based locks: the Anderson lock.	Slides, blackboard	2 hours
5.	Locks - II. Advanced versions: the CLH lock, the MCS lock.	Slides, blackboard	2 hours
6.	Lists with concurrent access - I. General aspects. Variants: coarse-grained, fine-grained, optimistic synchronization.	Slides, blackboard	2 hours
7.	Lists with concurrent access - II. Variants: lazy synchronization, non-blocking synchronization. Comparison of list versions.	Slides, blackboard	2 hours
8.	Recapitulative verification.(*)	Homework presentations.	2 hours
9.	Queues and stacks with concurrent access I. Variants: partial bounded queue, lock-free unbounded queue.	Slides, blackboard	2 hours
10.	Queues and stacks with concurrent access II. The ABA problem. The lock-free unbounded stack.	Slides, blackboard	2 hours
11.	Concurrent hashing. Variants: closed-address (lock-based and lock-free), open-address.	Slides, blackboard	2 hours
12.	Specific techniques for parallel processing I. Concepts: work and critical path. The scheduler impact. Work stealing and work dealing techniques. MapReduce – introductory discussion. Parallel processing of data streams.	Slides, blackboard	2 hours
13.	Specific techniques for parallel processing II. Work stealing and work dealing. Advanced aspects: MapReduce, parallel processing of data streams – introductory discussion.(*)	Slides, blackboard	2 hours
14.	Barierrs. Introductory discussions on programming using transactional memory – general aspects.(*)	Slides, blackboard	2 hours

Bibliography

[1] The Art of Multiprocessor Programming, 2nd edition (Maurice Herlihy, Nir Shavit, Victor Luchangco, Michael Spear) – Morgan Kaufmann, Elsevier, 2021
[2] Concurrent Programming: Algorithms, Principles, and Foundations (Michel Raynal) – Springer Verlag, 2013
[3] Concurrency. The Works of Leslie Lamport (editor – Dahlia Malkhi) – ACM Books, 2019

8.2	Seminar / Laboratory	Teaching methods	Observations (hours and bibliography)
1.	Introduction of the work environment. Recap of basic notions of concurrent programming in Java.	Introductory discussions.	2 hours
2.	Recap of using semaphores in Java. Exercises concerning mutual exclusion.	Exercises proposal and discussions.	2 hours
3.	Atomic types in Java. Exercises concerning fairness in a concurrent context.	Exercises proposal and discussions.	2 hours
4.	Exercises concerning the notions of sequential consistency and linearizability. Exercises for testing the efficiency of various types of locks.	Exercises proposal and discussions.	2 hours
5.	Coordination operations using the Java object monitor and conditions. Example of the lost-wakeup problem in the context of conditioned locks. Homework announcement.	Exercises proposal and discussions.	2 hours
6.	Example of readers-writers locking and exercises.	Exercises proposal and discussions.	2 hours
7.	Exercises concerning concurrent lists.	Exercises proposal and discussions.	2 hours
8.	Presentation of the homework.(*)	Homework verification.	2 hours
9.	Exercises concerning concurrent lists. Optimizing the optimist list synchronization through versioning..	Exercises proposal and discussions.	2 hours
10.	Recapitulative exercises.	Exercises proposal and discussions.	2 hours
11.	Exercises concerning queues. Optimizing the bounded queue with conditioned synchronization.	Exercises proposal and discussions.	2 hours
12.	Exercises concerning concurrent queues and stacks.	Exercises proposal and discussions.	2 hours
13.	Exemplification of the various concurrent data structures available in the Java packages.(*)	Exercises proposal and discussions.	2 hours
14.	Exemplification of a barrier with execution termination detection. Recap exercises.(*)	Exercises proposal and discussions.	2 hours

Bibliography

[1] Java Concurrency in Practice (Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, Doug Lea) - Addison-Wesley, 2006

[2] The Java Memory Model (Jeremy Manson, William Pugh, Sarita V. Adve) – Proceedings of ACM POPL 2005 (also JSR 133)

(*) Activities marked in this manner can take place online, according to conditions mentioned at point 5, using specific methods assisted by technology.

9. Course content synchronization with the expectations of the community representatives, professional associations and employers from the program domain

This discipline aims to develop the capacity of students to understand and use concepts and data structures specific for multiprocessor programming. Such techniques of concurrent programming are necessary in the software industry, especially in the context of late performance capacity growth trends, which mostly favor the addition of multiple cores in processors as alternative to boosts in individual clock frequencies.

10. Evaluation

Activity type	**10.1 Evaluation criteria**	**10.2 Evaluation methods**	**10.3 The weigth of each evaluation form (%)**
10.1 Course	- understanding of basic theoretical concepts required for multiprocessor programming - understanding of specific synchronization aspects of the various presented data structures - capacity to identify and address various contexts fit for possible optimization using parallel processing - quality of stating the answers	Written exam	40%
10.2 Laboratory	- capacity of implementing various data structures that provide concurrent access - capacity to correctly solve various exercises concerning synchronization problems - capacity to apply parallelization techniques for enhancing the performance of an implementation	Solving exercises during the lab (30% - part of ongoing evaluation during semester) Practical homework presentation (30% - part of ongoing evaluation during semester)	60%
10.3 Minimal performance standards			
For the theoretical part, meeting the minimal conditions for graduation imply understanding the theoretical notions concerning multiprocessor programming and the difference between the efficiency of synchronization modes present in the discussed data structures. For the practical part, meeting the minimal conditions for graduations imply the correct implementation of some data structures of the ones presented in the cours and a comparative evaluation of their efficiency, as well as detecting and eliminating some synchronization issues in the homeworks and in the exercises proposed for solvind in the labs during the semester. Meeting the minimal conditions implies accumulating a graduation score out of the maximum possible grade, following the evaluations. This graduation score corresponds to a minimum of 45% of the maximum grade out of which at least a third from the laboratory activity (mandatory for participation in the written exam).			

Completion data,
23.09.2022

Course teacher,
Emanuel Onica

Seminary,
Emanuel Onica, Paul Diac

Date of approval in department,

Director of department,