COURSE OUTLINE

GENERAL

<table>
<thead>
<tr>
<th>SCHOOL</th>
<th>SCHOOL OF ENGINEERING</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACADEMIC UNIT</td>
<td>COMPUTER ENGINEERING AND INFORMATICS</td>
</tr>
<tr>
<td>LEVEL OF STUDIES</td>
<td>UNDERGRADUATE</td>
</tr>
<tr>
<td>COURSE CODE</td>
<td>CEID_NE5407</td>
</tr>
<tr>
<td>SEMESTER</td>
<td>WINTER</td>
</tr>
</tbody>
</table>

COURSE TITLE
Software & Programming of High Performance Systems

INDEPENDENT TEACHING ACTIVITIES
Lectures, Recitation sections, Laboratory Exercises

<table>
<thead>
<tr>
<th>WEEKLY TEACHING HOURS</th>
<th>CREDITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2(L), 1(RC), 2(LE)</td>
<td>5</td>
</tr>
</tbody>
</table>

Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (d).

COURSE TYPE
Skills development

PREREQUISITE COURSES:
The course requires knowledge acquired from the following courses: “Parallel Processing”, “Introduction to Computers & Programming”, “Introduction to Algorithms”, “Topics in Computer Architecture”, “Modern Topics in Computer Architecture”

LANGUAGE OF INSTRUCTION and EXAMINATIONS:
Greek. It is possible to be delivered in English if there are foreign students.

IS THE COURSE OFFERED TO ERASMUS STUDENTS:
Yes (in English)

COURSE WEBSITE (URL)
https://eclass.upatras.gr/courses/CEID1072/

LEARNING OUTCOMES

Learning outcomes
The course learning outcomes, specific knowledge, skills and competences of an appropriate level, which the students will acquire with the successful completion of the course are described.

Consult Appendix A
- Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area
- Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning and Appendix B
- Guidelines for writing Learning Outcomes

After attending the course the students should be able to:
1. Evaluate software development issues in HPC environments for a specific application
2. Describe the architecture of graphic cards and the Xeon Phi coprocessor
3. Distinguish the elements that differentiate the architecture of classic processors from that of GPUs
4. Describe the CUDA programming model
5. Develop a parallel application using the CUDA programming model for GPUs
6. Develop a parallel application using the OpenMP programming model for the Xeon Phi coprocessor
7. Describe performance optimization techniques for applications running on graphics cards
8. Describe performance optimization techniques for applications running on the Xeon Phi coprocessor

After attending the course the students will have developed the following skills:
1. They will be able to identify the parts of a serial application that have increased chances of
achieving better performance when they are parallelized for execution using CUDA on a GPU or using OpenMP on the Xeon Phi coprocessor

2. They will be able to develop a parallel application using the appropriate tools and programming models for the computing system used

3. They will be able to evaluate the performance of the parallel application they have created

4. They will be able to identify the parts of the parallel application that can be optimized

5. They will be able to apply techniques to optimize performance at these points

General Competences
Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

Search for, analysis and synthesis of data and information, with the use of the necessary technology
Adapting to new situations
Decision-making
Working independently
Team work
Working in an international environment
Working in an interdisciplinary environment
Production of new research ideas
Project planning and management
Respect for difference and multiculturalism
Respect for the natural environment
Showing social, professional and ethical responsibility and sensitivity to gender issues
Criticism and self-criticism
Production of free, creative and inductive thinking

SYLLABUS

1. Software development issues in HPC environments
   a. Cost of Software Development in HPC Environments
   b. Scalability and transferability of code
   c. Data complexity and parallel algorithms

2. Batch Systems
   a. Usage restrictions of resources in HPC environments
   b. SLURM

3. Nvidia GPU architecture
   a. Streaming Processor (SP)
   b. Streaming Multiprocessor (SM)
   c. SM features per GPU generation
   d. Other architectural features
   e. The concept of Compute Capability

4. The CUDA Programming Model
   a. What is CUDA?
   b. The concept of a Host and a Device
   c. Grid and blocks of threads
   d. Limitations on grid and block sizes
   e. Computational kernels
   f. Embedded variables of CUDA
   g. CUDA Runtime Flow
   h. Workload distribution

5. Memory access optimization
   a. Exploitation of the CUDA memory hierarchy
   b. CUDA shared memory programming strategy
      i. Splitting data into smaller tiles
      ii. Data reuse
   c. Performance optimization issues
   d. Barriers
   e. DRAM bursting and exploitation in CUDA applications
6. Program flow control
   a. Warp divergence
   b. Warp divergence avoidance
7. Atomic instructions
   a. Atomic instructions in CUDA
   b. The Compare-And-Swap (CAS) atomic instruction
   c. Implementing other atomic instructions using CAS
8. CUDA Streams
   a. Synchronous & Asynchronous execution
   b. CUDA Streams
   c. Assigning calculations to a stream
   d. Scheduling within a stream
   e. Asynchronous data transfer from/to the GPU
   f. Events
9. Architecture of the Xeon Phi coprocessor
10. Native and offload programming modes
11. Application of the OpenMP programming model on the Xeon Phi coprocessor

TEACHING and LEARNING METHODS - EVALUATION

<table>
<thead>
<tr>
<th>DELIVERY</th>
<th>Face-to-face, Distance learning, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY</td>
<td>Use of ICT in teaching, laboratory education, communication with students</td>
</tr>
<tr>
<td>TEACHING METHODS</td>
<td></td>
</tr>
<tr>
<td>Activity</td>
<td>Semester workload</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Lectures</td>
<td>2*13=26</td>
</tr>
<tr>
<td>Recitation sections</td>
<td>1*13=13</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>2*13=26</td>
</tr>
<tr>
<td>Study and preparation</td>
<td>2*13=26</td>
</tr>
<tr>
<td>Weekend study</td>
<td>2*13=26</td>
</tr>
<tr>
<td>Final exam preparation</td>
<td>4*3=12</td>
</tr>
<tr>
<td>week+2 weeks break</td>
<td></td>
</tr>
<tr>
<td>Course total</td>
<td>129</td>
</tr>
</tbody>
</table>

STUDENT PERFORMANCE EVALUATION

Description of the evaluation procedure

Language of evaluation, methods of evaluation, summative or conclusive, multiple choice questionnaires, short-answer questions, open-ended questions, problem solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other

Specically-defined evaluation criteria are given, and if and where they are accessible to students.

1. Language of evaluation: Greek
   (English if required)
2. Final examination (40% of final mark):
   Written exam that might include short-answer questions, problem solving
3. Laboratory exercise (60% of final mark):
   The exercise requires the application of advanced parallel programming techniques on a modern coprocessor, optimization, performance analysis and application of performance metrics.
4. Grading scale: 1 to 10
   Minimum passing mark: 5
   Grades < 5 correspond to ECTS grade F.
   For the rest of the grades the correspondence is as follows:
   >= 9: ECTS A
   >= 8: ECTS B
   >= 7: ECTS C
   >= 6: ECTS D
### ATTACHED BIBLIOGRAPHY

12. **Suggested bibliography:**
   - Course slides (available through the e-class platform, Greek).

13. **Related academic journals:**
   - IEEE Transactions on Parallel and Distributed Systems
   - ACM Transactions on Parallel Computing
   - International Journal of Parallel Programming
   - Journal of Parallel and Distributed Computing
   - Parallel Computing