<table>
<tr><td>Course Title</td><td colspan="5">Parallel Processing</td></tr>
<tr><td>Course Code</td><td colspan="5">ACOE401</td></tr>
<tr><td>Course Type</td><td colspan="5">Elective</td></tr>
<tr><td>Level</td><td colspan="5">Bachelor (1st Cycle)</td></tr>
<tr><td>Year / Semester</td><td colspan="5">4^{th} Year / 8^{th} Semester</td></tr>
<tr><td>Teacher’s Name</td><td colspan="5">Prof Costas Kyriacou</td></tr>
<tr><td>ECTS</td><td>6</td><td>Lectures / week</td><td>3</td><td>Laboratories / week</td><td>1</td></tr>
<tr><td>Course Purpose</td><td colspan="5">This course aims in providing the students with an in-depth understanding of the parallel processing enabling technologies and system architectures related to shared memory and distributed memory systems, as well as the main parallel programming models employed in shared memory systems (OpenMP), distributed memory systems (MPI) and GPU based systems (CUDA).</td></tr>
<tr><td>Learning Outcomes</td><td colspan="5">1. Classify parallel architectures based on the Flynn’s classification and propose other parameters that are essential for the classification of modern parallel processing systems.
2. Justify the need and describe the methodologies employed for synchronization and memory consistency and cache coherence in shared memory systems.
3. Describe and compare the different types of interconnects employed in parallel processing systems.
4. Outline and analyse the features of microarchitecture parallel systems such as superscalar, VLIW, vector, multithreading, CMP multi-core and tile processors.
5. Describe how the performance of a parallel system can be measured, list possible sources for performance losses and propose ways to improve the performance of a system.
6. Write efficient programs for message passing and shared memory parallel processing platforms using MPI and OpenMP, as well as for heterogeneous platforms using CUDA.
7. Outline the present developments in the field of parallel processing and show familiarity with the forefront of the relevant knowledge.</td></tr>
<tr><td>Prerequisites</td><td colspan="2">ACOE301 and ACSC183</td><td colspan="2">Co-requisite</td><td></td></tr>
<tr><td>Course Content</td><td colspan="5">• Introduction to Parallel Processing: Historic evolution of parallel processing. Motivation for parallel processing. Parallel processing applications case studies.
• Parallel Computer Models and Systems: Classification of parallel computer architectures, SIMD and MIMD systems. Shared address space, message passing and data parallel processing. Networks of Workstations, clusters, and MPP.</td></tr>
</table>

	• **Performance Metrics:** Workloads and benchmarks, execution time, throughput, speedup and efficiency. Latency tolerance techniques, load balancing and data locality issues. • **Interconnection Networks:** Communication performance, latency and bandwidth. Interconnection organization, links, switches and interconnection topologies. Switch design, routing and flow control. High speed LANs. Board-to-Board communication and Networks-on-Chip. • **Shared Memory Multiprocessors:** The cache coherence problem, snoop-based and directory based cache coherence. Synchronization mechanisms, mutual exclusion, event and global synchronization. Locks and Atomic operation. Memory consistency models and implementation. • **Microarchitecture Parallelism:** Levels of parallelism, DLP, ILP, TLP. The temperature and power wall problem, clock skewing and soft errors. Single-chip parallel processing architectures such as multithreaded processors, CMP and multi-core processors, and tile architectures. • **Heterogeneous Architectures:** Hardware accelerators, graphics processors (GPUs) and the CUDA programming model, threaded dataflow architectures. • **Parallel Programming:** The parallelization process, decomposition, assignment, orchestration and load balancing. Message passing programming using MPI, basic instructions, point to point communication, collective communication operations and synchronization issues. Performance issues in MPI. Shared memory programming using OpenMP, parallel regions, worksharing, data environment, and synchronization. Performance issues with OpenMP
Teaching Methodology	The taught part of course is delivered to the students by means of lectures, conducted with the help of both computer presentations. Practical examples and exercises are included in the lectures to enhance the material learning process. Lecture notes and presentations are available through the web for students to use in combination with the textbooks. Lectures are supplemented with laboratory work curried out on parallel programming. During laboratory sessions, students develop, test, debug and analyse programs in C using OpenMP and MPI. Students are also expected to submit and present orally a literature review group project on the current developments in the field of parallel processing.
Bibliography	**Textbooks:** • Thomas Sterling, Matthew Anderson and Maciej Brodowicz, ***"High Performance Computing: Modern Systems and Practices"***, Morgan Kaufmann, 2018 https://www.sciencedirect.com/book/9780124201583/high-performance-computing • Bertil Schmidt Jorge Gonzalez-Dominguez Christian Hundt Moritz Schlarb, ***"Parallel Programming, Concepts and Practice"***, Morgan Kaufmann, 2017

	https://www.sciencedirect.com/book/9780128498903/parallel-programming **References:** • Gérard Blanchet, Bertrand Dupouy, ***“Computer Architecture”***, Wiley, 2015 • M. Quinn, ***"Parallel Programming in C with MPI and OpenMP"***, McGraw Hill, 2004
Assessment	• Tests: 30% • Laboratory Work: 20% • Class Project/Literature Review 10% • Final Exam 40%
Language	English