1. **Course Number and Course Title:**
   COE 636 – Advanced Multicore and GPU Computing

2. **Credit Hours:**
   3-0-3

3. **Prerequisites and/or Co-Requisites:**

   Prerequisite: Approval of the CSE Head of Department
   Concurrent: None
   Competencies: Operating systems background and basic Linux/Unix experience.

4. **Name and Contact Information of Instructor:**
   Dr. Gerassimos Barlas

5. **Course Description (Catalog Description):**
   Covers software development on multi-core and many-core systems, including CPUs, GPUs and hybrid systems. Covers performance metrics and performance prediction of parallel algorithms. Examines models of parallel computation and associated software architectures such as master-worker, pipelining, data-flow and streaming. Studies advanced load-balancing mathematical models and algorithms. Uses selected applications as case-studies as well as state-of-the-art software tools such as CUDA and OpenCL.

6. **Textbook and other Supplemental Material:**
   Textbook:
   - Gerassimos Barlas, "Multicore and GPU Programming: An Integrated Approach", 1e, Morgan Kaufmann, ISBN-13 978-0124171374, 2014

   Other Supplemental material:
   - Maurice Herlihy and Nir Shavit, *The Art of Multiprocessor Programming*, 2012, Revised 1e, Morgan Kaufmann, ISBN: 978-0-12-397337-5
   - Shane Cook, *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*, Morgan Kaufmann, 1e, 2012, ISBN: 978-0124159334
   - Ian Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley Pub Co, ISBN: 0201575949, 1995, also available on-line at http://www.mcs.anl.gov/dbpp/
   - Selected material from online sources:
     - NVidia's CUDA Zone

7. **Course Learning Outcomes:**
   Upon completion of the course, students will be able to:
   1. Classify the most important contemporary multi-core and many-core machine architectures.
   2. Apply appropriate metrics for measuring the performance of a parallel algorithm.
   3. Create mathematical models that predict the performance of parallel algorithms.

4. Analyze algorithms for creating and managing concurrent data structures such as lists, stacks and queues.
5. Design and develop process-level parallel programs using MPI.
6. Use OpenMP to program multicore CPUs.
7. Use CUDA threads to speed-up computation on NVidia GPUs.
8. Employ different software architectures (e.g. master-worker, pipelining, etc.) and data organization techniques (distributed data structures) for efficient utilization of multicore platforms.

8. Teaching and Learning Methodologies:

Methods include lectures, problem and project based learning methods (assignments, exams, paper reviews, presentation), and class discussions.

9. Course Topics and Schedule:

Topic	**Weeks**
Parallel computers, Taxonomies, Architectures, Metrics	Week 1
Parallel program design methodologies	Week 2
Concurrency and correctness	Week 3
Concurrent data structures: stacks, lists, priority queues, skip lists, etc.	Week 4
Concurrent data structures without using locks	Week 5
Distributed Computing using MPI	Week 6
MPI collectives, and one-sided communications + Midterm I	Week 7
MPI case studies, profiling and performance monitoring	Week 8
OpenMP	Week 9
OpenMP inter-loop dependencies	Week 10
GPU architecture and CUDA programming	Week 11
Advanced CUDA, optimizations, graphs, events + Midterm II	Week 12
Load balancing	Week 13
Divisible Load Theory and applications	Week 14
Presentations	Week 15
Final Exam	Week 16