# COURSE OUTLINE

## (1) GENERAL

| | | | |
|---|---|---|---|
| **SCHOOL** | ENGINEERING | | |
| **ACADEMIC UNIT** | ELECTRICAL AND COMPUTER ENGINEERING DEPT. | | |
| **LEVEL OF STUDIES** | Undergraduate | | |
| **COURSE CODE** | ECE_INF861 | **SEMESTER** | **8** |
| **COURSE TITLE** | Compilers | | |

| **INDEPENDENT TEACHING ACTIVITIES**<br>*if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for the whole of the course, give the weekly teaching hours and the total credits* | **WEEKLY TEACHING HOURS** | **CREDITS** |
|---|---|---|
| Lectures | 3 | |
| Seminars / Practice exercises | 1 | |
| Laboratory | 0 | |
| *Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (4).* | 4 | 5 |

| | |
|---|---|
| **COURSE TYPE**<br>*general background, special background, specialised, general knowledge, skills development* | Special background |
| **PREREQUISITE COURSES:** | Students should have good knowledge of Procedural Programming (ECE_K260) or Object-Oriented Programming (ECE_K430) and Data Structures & Algorithms (ECE_K320) |
| **LANGUAGE OF INSTRUCTION and EXAMINATIONS:** | Greek. Maybe offered in English if there are foreign students. |
| **IS THE COURSE OFFERED TO ERASMUS STUDENTS** | Yes (in English) |
| **COURSE WEBSITE (URL)** | https://eclass.uop.gr/courses/327/ |

## (2) 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*

The course aims to introduce students to translation technology and compilers. The course material aims at understanding and acquiring fundamental knowledge in the translation of programs expressed in programming languages and relevant optimization techniques. An additional goal is to link introductory programming courses and data structures with the most specialized courses in the fields, because it requires global infrastructure knowledge and the offered knowledge can be used in a wide range of applications, such as information retrieval, communication protocols, hardware description languages, microprocessors , natural language processing etc. In particular, it aims

- familiarize with the structure of programming languages

• to understand the use of tools for the implementation of the lectical and syntactic analysis

• to understand the application of algorithms and data structures through the implementation the compilation phases

• approach machine language level code and perform the comparison with high level languages learned in previous courses through the whole translation process

• the practical understanding of computer architecture through the study of the mechanisms of the execution of the final code

• come in contact with complex algorithms and the concept of Instruction Set Architecture through understanding code optimizations and final code generation

Most importantly, the course aims to develop the teamwork ability in the design and implementation of a complex software project, such as a complete compiler, from its first to the last phase. Upon successful completion of the course the student/student will be able to:

- Understand key concepts of translator theory
- Design a new programming language
- Write the grammar of a programming language
- Understand of the basic concepts of compilers, both theoretically and practically.
- Acquire specialized knowledge on key topics found in compilers, so that it can combine them for the design and implementation of a simple compiler.
- Transform a program from one high-level procedural programming language to another
- Extract information from a program about the data structures it uses and to organise this information into appropriate data structures so that it is complete and easily accessible.
- Produce machine code based on previous steps
- Apply code optimization techniques at various levels, like the original code, intermediate or machine language level code.
- Have the skills to apply topics studied in other important core courses, such as algorithms, data structures and programming within a compiler.
- Work with other colleagues to create and present a complex collaborative programming project.

**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* | *Project planning and management* |
| *Adapting to new situations* | *Respect for difference and multiculturalism* |
| *Decision-making* | *Respect for the natural environment* |
| *Working independently* | *Showing social, professional and ethical responsibility and sensitivity to gender issues* |
| *Team work* | *Criticism and self-criticism* |
| *Working in an international environment* | *Production of free, creative and inductive thinking* |
| *Working in an interdisciplinary environment* | *……* |
| *Production of new research ideas* | *Others…* |
| | *…….* |

- Search for, analysis and synthesis of data and information, with the use of the necessary technology
- Working independently
- Team work
- Acquisition of software implementation skills and experiences

- Familiarity with the use of analysis and automatic code production technologies
- Understand the inner workings of a computer system
- Production of free, creative and inductive thinking
- Project planning and management

**(3) SYLLABUS**

1. **Introduction to compilers and translation systems. The Structure of a Compiler:** Languages, compiler development, development tools, composition of software components, terminology, requirements, compilation phases, compiler organization
2. **Lexical Analysis:** automata, regular expressions, internal lexical analyser functionality
3. **Compiler Construction Tools:** Flex, bison, antlr
4. **Syntactic Analysis I:** Syntax analyser functionality, top down parsers, grammars (LL(1), LL(k)), grammatical transformations
5. **Syntactic Analysis II:** Syntax Directed Translation, bottom up parsers, grammars (LR(0), SLR(1), LR(k) και LALR(1))
6. **Symbol Table Management:** Organization of the symbol table, activity chart, retrieval of information from the symbol table, different formats and implementations depending on language requirements, Hash tables, libraries
7. **Semantic Analysis:** Semantic checks, semantic properties, semantic rules.
8. **Intermediate Code Production I:** intermediate languages, arithmetic operations, logical representations, logical operations
9. **Intermediate Code Production II :** Decision structures, repetition structures, functions and processes
10. **Code Optimization I:** Organization of an optimization translator, control flow and data analysis, algebric transformations
11. **Code Optimization II:** Loop transformations, low-level transformations, subprogram transformations
12. **Final Code I Production:** Basic machine language commands, code output for arithmetic operations, jump commands and I/O commands
13. **Final Code II Production:** Memory Organization, Code generation for write and read commands to/from memory, Function and Process Calls, Parameter Passing

The course provides students with all required knowledge in the subject of compilers. The course includes solving exercises as well as mandatory programming work in groups on the of a complete compiler using high-level language (C,Java) and compiler development tools for some simplified programming language.

## (4) TEACHING and LEARNING METHODS - EVALUATION

<table>
<tr><td>DELIVERY<br>Face-to-face, Distance learning, etc.</td><td>Face-to-face</td></tr>
<tr><td>USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY<br>Use of ICT in teaching, laboratory education, communication with students</td><td>• The e-Class platform is used for asynchronous support of the learning process (course operations regulation, distribution of slides, project related material, complementary material, announcements, links and bibliography, etc.).<br>• During lectures a projector and presentations in electronic form are used, which are also posted in the eclass platform.<br>• During lectures and especially at seminars, a computer is used to write and execute example code.</td></tr>
<tr><td>TEACHING METHODS<br><br>The manner and methods of teaching are described in detail.<br><br>Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.<br><br>The student's study hours for each learning activity are given as well as the hours of non-directed study according to the principles of the ECTS</td><td>
<table>
<tr><th>Activity</th><th>Semester workload</th></tr>
<tr><td>Lectures</td><td>13*3 = 39</td></tr>
<tr><td>Seminars</td><td>Presentation of Compiler tools and project phases<br>5*1 = 5</td></tr>
<tr><td>Project</td><td>Team working for compiler development<br>8*5=40</td></tr>
<tr><td>Study and analysis of bibliography</td><td>12*3 = 36</td></tr>
<tr><td>Study for final exam</td><td>5</td></tr>
<tr><td>Course Total</td><td>125 hours<br>(5 ECTS)</td></tr>
</table>
</td></tr>
<tr><td>STUDENT PERFORMANCE EVALUATION<br><br>Description of the evaluation procedure<br><br>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<br><br>Specifically-defined evaluation criteria are given, and if and where they are accessible to students.</td><td>Language of evaluation: Greek (English if required)<br><br>Presentation of project and oral examination for each student (40% of final mark).<br><br>Final examination (60% of final mark): Written exam that might include multiple choice questionnaires, short-answer questions, problem solving<br><br>Grading scale: 1 to 10<br>Minimum passing mark: 5<br>Grades < 5 correspond to ECTS grade F.<br>For the rest of the grades the correspondence is as follows:<br>>= 9: ECTS A<br>>= 8: ECTS B<br>>= 7: ECTS C<br>>= 6: ECTS D<br>>= 5: ECTS E</td></tr>
</table>

| | |
|---|---|
| | Note:<br><br>The evaluation criteria are clearly announced during the first lecture of the semester as well as the individual percentages and the rules governing the grading, both of the above evaluation methods and of the individual percentages of the different phases of the work. The project evaluation criteria and the individual percentages are posted on the asynchronous educational platform with the project announcement. |

## (5) ATTACHED BIBLIOGRAPHY

*- Suggested bibliography:*


- Compilers 2ed, A.V. Aho-M.S. Lam-R.Sethi-J.D.Ullman,ISBN 032-148-681-1, Addison Wesley, 2006
- Engineering: A Compiler, Keith D. Cooper, Linda Torczon, ISBN 978-012-088-478-0, Morgan Kaufmann, 2011
- Introduction to Compiler Design, Torben Aegidius Mogensen, 978-331-966-966-3, HEAL-Link Springer ebooks, 2017
- A Practical Approach to Compiler Construction, Des Watson, ISBN 978-331-952-789-5, HEAL-Link Springer ebooks, 2017


*- Related academic journals:*

- IEEE Transactions on Computers
- IEEE Transactions on Parallel and Distributed Systems
- ACM Transactions on Embedded Computing Systems
- ACM Transactions on Design Automation of Electronic Systems
- Programming Language Design and Implementation (PLDI) Conference Proceedings
- Languages, Compilers, and Tools for Embedded Systems (LCTES) Conference Proceedings