Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

SYLLABUS
COURSE DESCRIPTION

COURSE TITLE	**Formal Languages and Compilers**
COURSE CODE	75016
SCIENTIFIC SECTOR	INF/01
DEGREE	Bachelor in Computer Science and Engineering
SEMESTER	1st Semester
YEAR	3rd year
CREDITS	8

TOTAL LECTURING HOURS	48
TOTAL LAB HOURS	24
PREREQUISITES	Knowledge of either Java or C programming languages
COURSE PAGE	https://ole.unibz.it/ and http://www.inf.unibz.it/~artale/Compiler/compiler.htm

SPECIFIC EDUCATIONAL OBJECTIVES	• Type of course: “caratterizzante” for L-31 and “affine o integrative” for L-08 • Scientific area: “discipline informatiche” for L-31 and “formazione interdisciplinare” for L-8 The main objective is to introduce the fundamental notions about formal languages (Chomsky classification of Languages, Regular Languages, Automata, Context Free Grammars) and understand the mechanisms governing the analysis and synthesis of programming languages. Students will learn the most important techniques for the representation and generation of Languages (in particular, regular and context-free languages). Those techniques will be applied to the construction of a compiler for a programming language. In particular, during this course the student will learn how to build the different parts of a Compiler with a particular emphasis on Lexical Analysers, Parsers and a basic form of code generation.

LECTURER	Alessandro Artale, office POS 2.03 Faculty of CS, POS Building, Piazza Domenicani 3, artale@inf.unibz.it, +39 0471 016150 Web Page: http://www.inf.unibz.it/~artale Email: artale@inf.unibz.it

SCIENTIFIC SECTOR OF THE LECTURER	INF/01
TEACHING LANGUAGE	English
OFFICE HOURS	During the lecture time span, Office 2.03. To fix an appointment email at artale@inf.unibz.it
TEACHING ASSISTANT	Same as lecturer
OFFICE HOURS	Same as lecturer
LIST OF TOPICS COVERED	• Formal languages and grammars • Regular languages (automata, regular expressions, regular grammars) • Context free languages (stack machines) • Compiler organization • Lexical analysis • Top-down and bottom-up parsing • Symbol tables, semantic checking • Principles of Code generation
TEACHING FORMAT	Frontal lectures, labs with (programming) exercises, team projects.

LEARNING OUTCOMES	**Knowledge and understanding** • knowing the concepts of formal languages, the techniques of compilation and various programming paradigms. **Applying knowledge and understanding** • being able to develop and construct translators and compilers. **Ability to make judgments** • being able to work autonomously according to the own level of knowledge; • being able to take the responsibility for software development projects. **Communication skills** • being able to work in teams to implement software systems; • being able to use modern communication systems. **Ability to learn** • have developed learning capabilities to pursue further studies with a high degree of autonomy • have acquired learning capabilities that enable them to carry out project activities in companies, public institutions or in distributed development communities

ASSESSMENT	Project conducted in team and a written exam (with an optional mid-term written exam). In the project part of the exam we will assess the learning outcomes related to the application of the acquired knowledge, the ability to make judgments and the communication skills. In fact, the goal of the project is to design a compiler for a small programming language. The project part must be positively evaluated before the written exam.

	In the written exam (including the optional mid-term exam) there will be verification questions, transfer of knowledge questions and exercises. The learning outcome related to knowledge and understanding, applying knowledge and understanding and those related to the student ability to learn and the acquired learning skills will be assessed by the written exam.
ASSESSMENT LANGUAGE	English
EVALUATION CRITERIA AND CRITERIA FOR AWARDING MARKS	• Project: Compiler Development (30%) • Mid-term Written Exam (35%) o Note: the Mid-term is optional and covers the Formal Language part of the final exam. • Final Written Exam o 35% covering a reduced program for students who passed the Mid-term exam, or o 70% covering the full program in case of failure of the Mid-term. Written exam questions will be evaluated in term of correctness and clarity. Project is evaluated in term of quality of the solution: complexity and novelty of the programming language to be compiled, data structures used in implementing the symbol table, depth of the semantic analysis carried on. Note: In case of a positive mark both the mid-term exam and the project will count for 3 regular consecutive exam sessions.

REQUIRED READINGS	Compilers: Principles, Techniques, and Tools (2nd edition). Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeff Ullman. Publisher: Addison Wesley, 2007. Introduction to Automata Theory, Languages, and Computation (3rd edition). J.E. Hopcroft, R. Motwani, J.D. Ullman. Addison Wesley, 2007.
SUPPLEMENTARY READINGS	Compiler Construction: Principles and Practice, Kenneth C. Louden. Publisher: Brooks Cole, 1997. Advanced Compiler Design and Implementation, Steven Muchnick. Publisher: Morgan Kaufmann, 1997. Programming Language Processors in Java: Compilers and Interpreters, David Watt and Deryck Brown. Publisher: Prentice Hall, 2000.
SOFTWARE USED	C or Java, YACC, LEX