Syllabus

Course Description

Course Title	Formal Languages and Compilers
Course Code	76214
Course Title Additional	
Scientific-Disciplinary Sector	INF/01
Language	Italian
Degree Course	Bachelor in Computer Science
Other Degree Courses (Loaned)	
Lecturers	Prof. Alessandro Artale, Alessandro.Artale@unibz.it https://www.unibz.it/en/faculties/engineering/academic-staff/person/3026
Teaching Assistant	
Semester	Second semester
Course Year/s	2
CP	6
Teaching Hours	40
Lab Hours	20
Individual Study Hours	90
Planned Office Hours	
Contents Summary	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. During this course the student will learn how to build the different parts of a Compiler with a particular emphasis on Lexical Analyzers (with the use of Lex during the Lab), Parsers (with the

	use of YACC during the Lab) and basics of code generation.
Course Topics	- Formal language theory - Regular languages: automata, regular expressions, regular grammars - Context free languages (stack machines) - Lexical and syntactic analysis: Lexer specification, top-down and bottom-up parsing - Semantic analysis rules for: type checking, symbol table and control flow - Intermediate code generation
Keywords	Language Theory Finite-state Automata Context-Free Grammars and Parsers Semantic Analysis and Code Generation
Recommended Prerequisites	Knowledge of the C programming language and ability to menage Dynamic Data Types. Experience working in Linux environment (preferred)
Propaedeutic Courses	
Teaching Format	The course includes frontal lectures, lab sessions with programming exercises, and team projects.
Mandatory Attendance	Attendance is not compulsory but recommended. Non-attending students must contact the lecturer at the start of the course to agree on the modalities of the independent study. Exam modalities for non-attending students are the same as for attending students.
Specific Educational Objectives and Learning Outcomes	Knowledge and Understanding - D1.7 Possess sound knowledge of the theoretical foundations of computer science - D1.10 Know the concepts of formal languages, and the techniques of compilation of various high level programming languages. Applying knowledge and understanding - D2.8 Be able to develop and construct translators and compilers Ability to make judgments - D3.1 Be able to collect and interpret useful data and to judge information systems and their applicability. - D3.2 Be able to work autonomously according to the own level of

	knowledge and understanding. Communication skills - D4.1 Be able to use one of the three languages English, Italian and German, and be able to use technical terms and communication appropriately. - D4.5 Be able to work in teams for the realization of IT systems. Learning skills - D5.1 Have developed learning capabilities to pursue further studies with a high degree of autonomy.
Specific Educational Objectives and Learning Outcomes (additional info.)	
Assessment	Assessment consists of a team project and a written exam. The project is designed to evaluate the application of acquired knowledge, the ability to make informed judgments, and communication skills, through the collaborative development of a compiler for a small programming language. Successful completion of the project is a prerequisite for admission to the written exam. The written exam is composed of two parts: the first part is based on Formal Language topics and the second on Lexical, Parser and Semantic rules topics. The first part will also be offered as a MidTerm exam. The written exam includes verification questions, knowledge transfer tasks, and practical exercises, and is intended to assess knowledge and understanding, the ability to apply knowledge, and the student's learning skills.
Evaluation Criteria	The final grade is composed of a written exam worth 70% (35% on the Formal Language topics and 35% on Lexical, Parser and Semantic rules topics) and a project on compiler development worth 30%. The written exam will be evaluated based on the correctness and clarity of the answers. The project will be assessed according to the quality of the solution, including the complexity and originality of the designed programming language, the data structures used to implement the symbol table, and the depth of

	the semantic analysis performed. The project evaluation will remain valid for three consecutive regular exam sessions.
Required Readings	• Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition, 2006. ISBN 0321486811. • John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. Introduction to Automata Theory, Languages, and Computation. Pearson/Addison Wesley, Boston, 3rd edition, 2007. ISBN 0321455363 9780321455369 0321462254 9780321462251 0321455371 9780321455376 0321476174 9780321476173. URL: http://infolab.stanford.edu/~ullman/ialc.html.
Supplementary Readings	• Kenneth C. Louden. Compiler Construction: Principles and Practice. PWS Publishing Co., USA, 1997. ISBN 0534939724. • Steven S. Muchnick. Advanced compiler design and implementation. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998. ISBN 1558603204. • David Watt, Deryck Brown, and Robert W. Sebesta. Programming Language Processors in Java: Compilers and Interpreters AND Concepts of Programming Languages. Prentice Hall Press, USA, 2007. ISBN 1408200414.
Further Information	- C (https://gcc.gnu.org) - YACC (https://pubs.opengroup.org/onlinepubs/9799919799/utilities/yacc.html) - LEX (https://pubs.opengroup.org/onlinepubs/9799919799/utilities/lex.html) - Linux (e.g., https://ubuntu.com)
Sustainable Development Goals (SDGs)	Quality education