

Academic Year: (2024 / 2025)

Review date: 12-05-2024

Department assigned to the subject: Computer Science and Engineering Department

Coordinating teacher: GARCIA HERRERO, JESUS

Type: Compulsory ECTS Credits : 6.0

Year : 3 Semester : 2

REQUIREMENTS (SUBJECTS THAT ARE ASSUMED TO BE KNOWN)

Programming (Course: first - Semester: first)
Automata and Formal Language Theory (Course: second - Semester: second)

SKILLS AND LEARNING OUTCOMES

- ¿ Know the techniques of lexical, syntactic and semantic analysis, and know how to develop a parser for a programming language or a domain-specific language.
- ¿ Know the most common code generation and optimisation techniques, and know how to determine their impact
- ¿ Know and know how to apply the most common techniques for error detection and recovery.

DESCRIPTION OF CONTENTS: PROGRAMME

Descriptors: Representation of formal languages, lexical analysis, syntactic analysis, semantic analysis, code generation, error recovery, code optimization

UNIT I: Introduction

History of compilers and languages

Basics

Languages and grammars

Formal definitions of Grammar, Regular Expressions and Automata

Phases and structure of a compiler

Tombstone diagrams

TOPIC II: Lexical Analysis

Design of a Lexical Analyzer

Finite Automata Regular Languages ¿¿recognizers

Construction of a Finite Automaton. Examples

Automatic Lexical Analyzer Generator: LEX

Handling Lexical Errors

THEME III: Parsing

Introduction to Syntactic Analysis

Classification of methods of syntactic analysis

Descending Scan, Syntactic LL

LL obtaining table (1). Examples

Ascending Scan, Syntactic LR

Treatment of Ambiguous Grammars. Examples

Automatic Parser Generator: YACC

UNIT IV: Treatment of Syntactic Errors

Errors. Detection and Recovery Strategies. Examples

Recovery with different analyzers

Descent parser LL

Up operator precedence parser

Ascending LR Parser

UNIT V: Semantic Analysis

Attribute Grammars, Examples, Registration
Specifying a translator: Translation Directed by Syntax and Translation Schemes
Evaluation of grammars
Construction of Abstract Syntax Trees

UNIT VI: Verification of Types

Introduction
Type expressions
Type systems. Checking static and dynamic
Sample construction and verification of simple types
Equivalence of type expressions
Overloading and Object Orientation

UNIT VII: Intermediate Code Generation

Types of Intermediate Languages
Codes three directions. Alternatives
Intermediate code generation: statements, arithmetic expressions, arrays
Control Flow Statements

UNIT VIII: Machine Code Generation

Machine and target machine code
Options machine code
Instructions and addressing and cost
Simple code generation from intermediate language
Basic blocks and flow graphs
Register allocation
Translation of other instructions

UNIT IX: Table of Symbols and Execution Environment

Memory allocation
Static and dynamic allocation
Stack and heap management. Examples
Function calls
Activation records
Passing parameters
Operations and organization of the symbol table

UNIT X: Code Optimization

Code optimization concept
Local optimization of basic blocks
Function preserving transformations
Elimination of dead code
Loops optimizatio
Global analysis of the data stream

UNIT XI: Specific Aspects

Other language processor
interpreter
Preprocesadotes and macroprocesadores
Language desig
Data structures and contro
Aspects of compilation for specific types of language
Examples of compilers

LEARNING ACTIVITIES AND METHODOLOGY

Theoretical lectures: 1.5 ECTS. To achieve the specific cognitive competences of the course.

Practical lectures: 1,5 ECTS. To develop the specific instrumental competences and most of the general competences, such as analysis, abstraction, problem solving and capacity to apply theoretical concepts.

Besides, to develop the specific attitudinal competences. They consist in proposing during the practical lectures a compiler/interpreter project to be developed in teamwork.

-Guided academic activities (present teacher): 1 ECTS. The student proposes a project according to the teachers guidance to go deeply into some aspect of the course, followed by public presentation.

-Guided academic activities (absent teacher): 1.5 ECTS. Exercises and complementary readings proposed by teacher. Exercises and examination: 0,5 ECTS. To complete the development of specific cognitive and procedural capacities.

ASSESSMENT SYSTEM

% end-of-term-examination:	40
% of continuous assessment (assignments, laboratory, practicals...):	60

Exercises and examinations are both learning and evaluation activities. The evaluation system includes the assessment of guided academic activities and practical cases, with the following weights:

Exercises and examination: 40%

Practical case: 40%

Guided academic activities

- Present teacher: 15%

- Absent teacher: 5%

BASIC BIBLIOGRAPHY

- A. V. Aho and Ravi Sethi and J. D. Ullman Compiladores: Principios, Técnicas y Herramientas, Addison-Wesley Iberoamericana, 1990.

- Kenneth C. Loudon Construcción de Compiladores. Principios y práctica, Thomson, 2004.

ADDITIONAL BIBLIOGRAPHY

- Dick Grune, Henri E. Bal, Cerial J.H. Jacobs, Koen G. Langendoen Modern Compiler Design, John Wiley & Sons, 2000.

- Doug Brown, John Levine, Tony Mason Lex & Yacc, O'Reilly Media, Inc., 1995.

- F. J. Sanchis and C. Galán Compiladores: Teoría y Construcción, Paraninfo, 1986.

- Garrido, Iñesta, Moreno, Pérez Diseño de Compiladores, Publicaciones Universidad de Alicante, 2002.