Anexo I:

Ficha de la Asignatura ECTS

UNIVERSIDAD DE ZARAGOZA

FICHA ASIGNATURA ECTS

Titulación: Ingeniería Informática
Órgano Responsable: Centro Politécnico Superior
Departamento: Informática e Ingeniería de Sistemas
Área de conocimiento: Lenguajes y Sistemas Informáticos
Nombre Asignatura: Compiladores I
Tipo: Troncal
Curso: Cuarto
Duración: Primer Cuatrimestre
Créditos ECTS: 3'8
Horas totales estimadas de trabajo del estudiante: 100
Horas de docencia teórica: 25
Horas de prácticas: 15 + 5
Horas de trabajo personal y otras actividades: 55
Profesores que imparten la asignatura: Joaquín Ezpeleta (ezpeleta@unizar.es), Raquel Trillo (raqueltl@unizar.es) y Rafael Tolosana (rafaelt@unizar.es)
Objetivos, destrezas y competencias que se van a adquirir: El alumno deberá utilizar correctamente en su trabajo los conceptos básicos de construcción de compiladores y traductores: análisis léxico, análisis sintáctico, análisis semántico. Durante las prácticas el alumnado deberá realizar las tareas de construcción de un analizador léxico, un analizador sintáctico y un analizador semántico integrados en un traductor. Afrontar con éxito la resolución de la construcción de un traductor de forma autónoma.
Prerrequisitos para cursar la asignatura: Tener conocimientos avanzados de programación, arquitectura de computadores y fundamentalmente de lenguajes, autómatas y gramáticas.
Contenido (breve descripción de la asignatura): 1. Introducción a los compiladores y traductores. 2. El análisis léxico. 2.1 Definición de conceptos: tokens, lexemas y patrones léxicos. 2.2 Expresiones regulares. 2.3 Autómatas Finitos Deterministas (AFD) y No Deterministas (AFND). 2.4 Implementación de analizadores léxicos. 2.5 Recuperación de errores léxicos. 2.6 Una herramienta de generación de analizadores léxicos: Flex. 3. Fundamentos de análisis sintáctico. 3.1 Gramáticas: definición, notación y tipos 3.2 Las gramáticas libres (independientes) de contexto: notación, derivaciones por la izquierda, derivaciones por la derecha y árboles de sintaxis. 3.3 La ambigüedad en gramáticas libres de contexto. 3.4 Algoritmos de transformación de gramáticas libres de contexto.

4. Análisis sintáctico descendente.
4.1 Análisis sintáctico descendente recursivo.
4.2 La factorización a izquierda.
4.3 La eliminación de la recursividad a izquierda.
4.4 Análisis sintáctico descendente recursivo predictivo.
4.5 Análisis sintáctico descendente iterativo predictivo: el análisis LL(1).
4.6 Gramáticas de expresiones de lenguajes de programación
5. Análisis sintáctico ascendente.
5.1 Análisis sintáctico ascendente.
5.2 Análisis sintáctico SLR.
5.3 Análisis sintáctico LR canónico.
5.4 Análisis sintáctico LALR.
5.5 Una herramienta de generación de analizadores sintácticos: Bison
6. Análisis semántico.
6.1 Análisis semántico.
6.2 Definición de gramáicas atribuidas.
6.3 Definición de árbol de sintaxis decorado.
6.4 Tipos de atributos: sintetizados y heredados.
6.5 Tipos de evaluación de atributos: por recorrido de árbol y al vuelo.
6.6 Tratamiento de atributos en Bison.
6.7 La evaluación ascendente de atributos sintetizados.
6.7 La evaluación ascendente de atributos heredados de hermanos izquierdos.
6.8 La evaluación ascendente de atributos heredados de hermanos derechos.

PROGRAMA DE PRÁCTICAS DE LABORATORIO:
D1 Definición de un lenguaje de programación imperativo propio.
D2 Construcción de un analizador léxico para el lenguaje de programación diseñado.
D3 Construcción de un analizador sintáctico descendente para el lenguaje de programación diseñado.
D4 Construcción de un analizador sintáctico ascendente para el lenguaje de programación diseñado.
D5 Construcción de un traductor dirigido por la sintaxis del lenguaje de programación diseñado a un lenguaje de programación conocido (Ada, Pascal, C, Java...).

Bibliografía:

1. A. Aho, R. Sethi, J. Ullman.:Compiladores: principios, técnicas y herramientas. Addison-Wesley Iberoamericana, 1990
2. J.Levine, T. Mason, D. Brown: Lex & Yacc. O'Reilly and Associates, 1992.
3. K. C. Louden.: Construcción de compiladores. Principios y práctica. Thomson, 2004.
4. A. Garrido, J.M. Iñesta, F. Moreno, J.A. Pérez.: Diseño de compiladores. Publicaciones Universidad de Alicante, 2002.

Metodología docente:

Metodología expositiva y metodologías activas
Prácticas en grupo
Trabajo individual

Tipo de evaluación:

El 75% de la calificación final (entre 0 y 10) se corresponde a la evaluación de un *examen final*. El 25% de la calificación final (entre 0 y 10) se corresponde a la evaluación de las *prácticas*. En caso de que el alumnado haya hecho las entregas y respectivas defensas en las clases de prácticas, se realizará una *evaluación continua* en la que se irá proporcionando

retroalimentación al alumnado. En otro caso la evaluación de las prácticas será análoga a la que se realiza en el examen final. Además, se podrá optar a un *punto extra* que servirá para inclinar la balanza en los casos dudosos (alumnos/as entre 4 y 5, 6 y 7, etc.). En este punto se considerará la *participación* en clase, la realización de mapas conceptuales de cada tema, la realización de fichas de ejercicios, y el *interés* mostrado
Lugar de impartición: Edificio Ada Byron, CPS, A0.02 y L0.03
Fechas de impartición: Septiembre a Enero
Idioma en que se imparte: Español
Observaciones:

UNIVERSIDAD DE ZARAGOZA

ECTS SUBJECT FORM

Degree: Computer Ingeneering
Faculty: Centro Politécnico Superior
Department: Computer science and system engineering
Knoledge area: Computer Systems and Languages
Subject name: Compilers I
Type: Troncal
Year: Fourth
Length: First four-month period
ECTS credits: 3’8
Estimated total hours of student work: 100
Teaching classes hours: 25
Laboratory hours: 15 +5
Hours of personal work and other activities: 55
Teachers: Joaquín Ezpeleta (ezpeleta@unizar.es), Raquel Trillo (raqueltl@unizar.es) y Rafael Tolosana (rafaelt@unizar.es)
Goals, skills and competences: The student will use correctly in his work the basic concepts of compiler construction and interpreter: scanner, parsers, semantic parsers. During laboratory hours the student will build a scanner, a parser and an interpreter in an autonomous way.
Previous requirements: Have knowledge about programming languages, computer architecture and grammars and automaton.
Contents (brief description of the subject): 1. Introduction. 2. Scanners. 2.1 Definitio of concepts: tokens, lexems y pattern. 2.2 Regular expressions. 2.3 Determinist finite automaton (DFA) and non determinist finite automaton (NDFA). 2.4 Implementation of scanners. 2.5 Lexical errors. 2.6 A tool to generate scanners: Flex. 3. Introduction to parsers. 3.1 Grammar: definition, notation y types 3.2 Context independent grammars: notation, left-derivation, right-derivation and syntax tree. 3.3 Ambiguous context independent grammars. 3.4 Algorithms to transform context independent grammars. 4. Top-down parsers. 4.1 Recursive Top-down parsers.

4.2 Left factoritation.
4.3 Remove left recursion in grammars.
4.4 Predictive recursive top-down parsers.
4.5 Predictive iterative top-down parsers: LL(1).
4.6 Grammars to expressions in programming languages
5. Botton-up parsers.
5.1 Introduction.
5.2 SLR parser.
5.3 LR(1) parser.
5.4 LALR parser.
5.5 A tool to build parsers: Bison
6. Semantic Parsing.
6.1 Introduction.
6.2 Definition of concepts.
6.4 Types of attributes.
6.5 How to calculate the value of the attributes.
6.6 Bison and the calculation of attributes.

SCHEDULING OF LABORATORY CLASSES:
D1 Definition of a custom programming language.
D2 Build a scanner of the custom programming language.
D3 Build a top-down parser of the custom programming language.
D4 Build a botton-up parser of the custom programming language.
D5 Build an interpreter from the custom programming language to a known programming language such as Ada, Pascal ...

Bibliography:

1. A. Aho, R. Sethi, J. Ullman.: Compilers: principles, techniques and tools. Addison-Wesley Iberoamericana, 1990
2. J.Levine, T. Mason, D. Brown: Lex & Yacc. O'Reilly and Associates, 1992.
3. K. C. Louden.: Construction of compilers. Principles and practices. Thomson, 2004.
4. A. Garrido, J.M. Iñesta, F. Moreno, J.A. Pérez.: Diseño de compiladores. Publicaciones Universidad de Alicante, 2002.

Teaching methodology:
Expository and active methodologies
group Labotary practices and final project

Assesment:
75% an exam.
25% practices in the lab.
Besides it is possible get one extra point if there is hard work during the course.

Teaching room: Ada Byron building, CPS, A1.05 y L0.05

Teaching dates: September ot January

Teaching language: Spanish

Observations: