COMPILACIÓN I
http://ji.ehu.es/konpi1

Profesores: Eneko Agirre, Nerea Ezeiza, Julian Gutierrez
(Información sobre tutorías, teléfonos, etc. en página web)

Objetivos:
El propósito de esta asignatura es doble. Por un lado, se pretende que el alumno se familiarice con las técnicas que se utilizan para traducir los constructores de los lenguajes de programación de alto nivel en código objeto, por otro, que sea capaz de aplicar los conocimientos anteriores en la resolución de problemas prácticos en distintos campos.
Dentro de esta filosofía general se hará especial hincapié en conseguir que el alumno:

- Conozca los principales conceptos en el área: Compilación, interpretación, traducción cruzada, *bootstrapping*, *linkedition*,...
- Conozca las distintas fases del compilador y las principales técnicas aplicadas en cada una de ellas.
- Conozca y sepa utilizar algunas herramientas de ayuda a la construcción de compiladores, especialmente en el entorno UNIX (LEX e YACC).
- Pueda aplicar los conocimientos teóricos adquiridos para resolver, por medio de alguno de los mecanismos de construcción de traductores estudiados, problemas que se encuadren dentro de la extensa área de la traducción.

Práctica
Durante el curso se realizará la siguiente práctica:

Diseño e implementación de un traductor sencillo a partir de las ideas presentadas en los tres primeros temas.

Esta práctica (10 semanas) se desarrollará durante el periodo Octubre-(Diciembre/Enero), siendo su objetivo que el alumno conozca y utilice algunas de las técnicas básicas de construcción de traductores. La evaluación de la práctica se realizará utilizando como criterio los objetivos abarcados.

Clases prácticas
La organización de las clases prácticas se concibe como complemento de las exposiciones realizadas en las clases de teoría, con una planificación paralela y una organización y metodología específica. El objetivo perseguido para las clases prácticas es fomentar el trabajo continuo y en equipo, acumular elementos para una evaluación más completa y dar claves concretas a los estudiantes sobre la forma de evaluar la asignatura.
La asistencia a las clases prácticas podrá ser *libre* (no influye en la calificación) o *reglada* (constituye un elemento complementario de evaluación).
La opción de asistencia reglada está limitada exclusivamente a estudiantes que tengan que desarrollar la práctica durante el curso y tiene las siguientes características: control de absentismo, trabajo en grupo y un régimen de entregas y presentaciones obligatorias.

Evaluación

La evaluación de la asignatura se basará en la calificación de la práctica y del examen final*. La nota final de la asignatura será la media de la obtenida en la práctica con la obtenida en el examen. Para aprobar la asignatura se deben superar el **examen** y la **práctica**.

Los alumnos que aprobaron la práctica el curso anterior están exentos de volver a realizar la práctica básica (5), salvo que manifiesten explícitamente su voluntad en sentido contrario. Presentarse al examen provoca el incremento en el número de convocatoria. Presentar únicamente la práctica no.

*La nota del examen final de los estudiantes acogidos a las clases prácticas regladas **con evaluación positiva** será aumentada en un punto, a partir de la calificación 4.

Temario de la asignatura

Tema 1. Introducción.
Tema 2. La fase de análisis léxico.
Tema 3. Aproximación a la definición y desarrollo de un traductor sencillo.
Tema 4. La fase de análisis sintáctico.
Tema 5. Gestión de memoria en tiempo de ejecución.

Bibliografía

- **Compilers. Principles, Techniques and Tools. Second Edition. Aho, A.V.; Lam, M.S.; Sethi, R.; Ullman, J.D. Addison-Wesley Computing. 2006.**
- **Compiladores. Principios, técnicas y herramientas. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman. ADDISON-WESLEY IBEROAMRICANA - 1990**
- Construcción de compiladores: Principios y práctica. Louden, K. Ed. Thomson. 2004.
- LEX: Un generador de reconocedores léxicos. (Master en Ciencia y Tecnología de los Computadores - 1993) José Miguel Alonso y José Miguel Blanco Arbe
- YACC: Una herramienta para el desarrollo de traductores (Master en Ciencia y Tecnología de los Computadores - 1993) José Miguel Alonso y José Miguel Blanco Arbe
- Compiladores. Conceptos fundamentales. Teufel, Schmidt & Teufel Addison-Wesley Iberoamericana – 1995
- Object-Oriented Compiler Construction. Jim Holmes. Prentice-Hall -1995
- Compilers. Their Design and Construction Using Pascal, R. Hunter. John Wiley & Sons Ltd - 1985
- Introduction to Compiler Construction with UNIX. Schereider, A.T., Friedman, H.G. Jr. Prentice-Hall: Englewood Cliffs, NJ. - 1985
- Compiler Construction, T. Waite, G. Goos. Springer-Verlag, 1984
- High-Level Languages and their Compilers, D. Watson. Addison-Wesley, 1989

Tema 1. Introducción

1.1 Compiladores y traductores
- Intérpretes

1.2 Características de los lenguajes de programación
- Evolución histórica
- Principales características
- Definición de un lenguaje de programación
- Estructura del lenguaje, tipos de datos y estructuras de control
- Clasificación de los lenguajes de programación

1.3 Estructura de un compilador
- Análisis léxico
- Análisis sintáctico
- Análisis semántico
- Tratamiento de errores
- Manejo de la Tabla de Símbolos
- Generación de código intermedio
- Optimización de código
- Generación de código
- Front-end / Back-end

1.4 Compiladores de varias pasadas

1.5 Entorno software del compilador
- Preprocesadores
- Ensambladores
- Linkeditores y cargadores
- Decompiladores

1.6 Herramientas para a construcción de compiadores
- Generación de analizadores y traductores
- Generación automática de código

1.7 Especificación y diseño de compiladores
- Diagramas-T
- Bootstrapping (arranque)
- Traducción incremental/interactiva
- De proyecto UNCOL a la JVM
- Compilación cruzada

Referencia básica: Capítulos 1 y 11 de [Aho,90], capítulo 1 de [Aho, 06; Louden, 04]

Referencias complementarias: [Weiser,92], capítulo 1,2 y 3 de [Watson,89], capítulos 1 y 2 de [Waite, 84], capítulos 2 y 17 de [Sánchez, 84] y capítulos 1, 11 y 12 de [Sanchís,86].

Objetivos: Este tema tiene un objetivo fundamental: que el alumno conozca con detalle la estructura general de un compilador, estructura en la que se irán incluyendo los contenidos presentados a lo largo del curso. Además se pretende que el alumno conozca una serie de términos, conceptos, técnicas y herramientas propias del área de la traducción de los lenguajes de programación y, más concretamente, que:

- Diferencie claramente compiladores de intérpretes, y conozca la estructura de ambos.
- Conozca la evolución de las técnicas de compilación y su relación con la evolución de los lenguajes de programación de alto nivel.
- Conozca el significado de términos propios del área de la traducción de lenguajes, como la compilación en varias pasadas, preprocesamiento o decompilación.
- Conozca la existencia de herramientas de ayuda a la generación automática de traductores.
- Maneje los diagramas-T como herramientas de especificación de compiladores.
- Conozca y comprenda el significado de términos como bootstrapping, traducción interactiva o compilación inversa.

Tema 2: La fase de análisis léxico

2.1 Introducción
- Su relación con el resto del compilador
- Los tokens y sus atributos
- Problemas que conllevan algunas características del lenguaje fuente
- Errores léxicos

2.2 Conceptos básicos
- Lenguajes regulares
- Expresiones regulares y definiciones regulares
- Autómatas finitos

2.3 Diseño e implementación de un analizador léxico
- Construcción de un analizador léxico a partir de un autómata finito determinista.
- Algunas consideraciones de implementación:
    - Utilización de buffers
    - El tratamiento de las palabras clave (palabras reservadas)
    - Acciones asociadas a los estados finales, su implementación
    - Implementación de la función de transición.

2.4 Un lenguaje de especificación de analizadores léxicos: LEX

Refencia básica: Capítulo 3 de [Aho,90; Aho, 06], capítulo 2 de [Louden, 04] y [Miguel, 93a]

Referencias complementarias: [Levine,92]

Objetivos: El objetivo general de este tema es que el alumno sea capaz de construir de una manera sistemática, analizadores léxicos eficientes. Al finalizar el tema el alumno debe:

- Ser capaz de especificar un analizador léxico para un lenguaje de programación utilizando las expresiones regulares.

- Poder diseñar e implementar un analizador léxico basado en la interpretación de un autómata finito determinista sobre la cadena de entrada.

- Manejar con soltura los problemas prácticos asociados a la construcción de analizadores léxicos, como pueden ser el manejo de buffers, el reconocimiento de palabras reservadas o el tratamiento de errores léxicos.

- Ser capaz de construir un reconocedor léxico sencillo utilizando LEX.

Tema 3: Aproximación a la definición y desarrollo de un traductor sencillo

3.1 Introducción

3.2 Definición sintáctica de un lenguaje

3.3 Traducción dirigida por la sintaxis
- Esquemas de traducción dirigidos por la sintaxis
- Recorrido en profundidad del árbol de análisis
- Atributos sintetizados

3.4 Análisis sintáctico
- Análisis descendente
- Análisis predictivo
- Construcción de un analizador predictivo
- Transformación de una gramática en predictiva

3.5 Traducción de los principales constructores de los L.P.
- Expresiones aritméticas
- Expresiones booleanas
- Instrucciones condicionales
- Instrucciones repetitivas

3.6 Análisis semántico
- Validación estática y dinámica
- Especificación de un verificador de tipos
- Equivalencia y conversión de tipos
- Funciones y operadores sobrecargados

3.7 La Tabla de Símbolos
- Contenido y manejo de la Tabla de Símbolos
- Representación de la Tabla de Símbolos
- El alcance de la información

3.8 Tipos de código intermedio
- Códigos de tres direcciones
- Máquinas abstractas con pila
    - »JVM
    - »p-code
- Notación postfija, árboles sintácticos y AST's

3.9 Extensión de la traducción de los principales constructores de los L.P.
- Declaraciones
- Instrucciones de asignación
- Expresiones booleanas
- Referencias a Tablas

Referencia básica: Capítulos 2, 4, 5, 6 y 8 de [Aho,90], capítulos 2, 4, 5 y 6 de [Aho,06] y capítulos 3, 4 y 6 de [Louden, 04].

Referencias complementarias: Capítulo 2 de [Fischer, 88]

Relación con otros temas: Con el tema de introducción, concretamente con el apartado en que se presentan las fases del compilador. Enlaza con el tema 2 que habla del análisis léxico. A su vez supone una introducción a los contenidos que se presentarán en los temas 4, 5 y 6. Especialmente en lo que hace relación al análisis descendente predictivo y a la traducción dirigida por la sintaxis.

Objetivos: El objetivo general de este tema consiste en hacer posible que el alumno pueda enfocar el problema de construir un traductor de un lenguaje de programación de una manera sistemática y en una fase muy temprana del desarrollo de la asignatura. Cuando termina la exposición del tema el alumno debe poder:

- Obtener a partir de una gramática que tenga recursividad a izquierdas o prefijos comunes (características que impiden la traducción descendente predictiva) una gramática equivalente sin esos problemas.

- Construir un analizador descendente predictivo recursivo a partir de una gramática y comprender su funcionamiento.

- Especificar una traducción sencilla utilizando esquemas de traducción dirigida por la sintaxis y construir un programa que implemente este tipo de especificaciones.

- Relacionar de una manera operativa un analizador sintáctico que le solicita sucesivamente tokens.

Contenidos: Este tema es una introducción a unas técnicas concretas asociadas a la construcción de front-ends: las de construcción de analizadores descendentes recursivos predictivos y las especificaciones basadas en la sintaxis del lenguaje de entrada. El tema se desarrolla en paralelo a la solución de un problema "sencillo": el diseño de un traductor que, tomando como entrada una expresión aritmética, genera como resultado su traducción a código de tres direcciones. De esta manera sucesivamente se motiva el concepto de especificación basada en la sintaxis, a continuación se introduce el método de construcción de analizadores, se relacionan especificación léxica y sintáctica del lenguaje y se acaba con el diseño completo del traductor. A partir de la asimilación de estos contenidos los alumnos pueden comenzar el diseño de un traductor para un lenguaje suficientemente significativo.

Tema 4. Análisis sintáctico

4.1 El analizador sintáctico
- Su relación con el resto del compilador
- Análisis descendente y ascendente

4.2 Definición sintáctica del lenguaje fuente
- Gramáticas independientes del contexto
- Ambigüedad
- Recursividad a izquierdas
- Factorización a izquierdas

4.3 Análisis descendente
- Analizadores predictivos
- Condiciones LL(1)
- Análisis descendente recursivo
- Análisis descendente predictivo no-recursivo

4.4 Análisis ascendente
- Análisis por reducción-desplazamiento
- Analizadores basados en la precedencia de operadores
- Analizadores LR

4.5 Tratamiento de errores sintácticos

4.6 YACC: generador de analizadores sintácticos

Referencia básica: Capítulos 3 y 4 de [Aho, 90; Aho 06], capítulos 3, 4 y 5 de [Louden, 04] y [Miguel, 93b]

Relación con otros temas: Este tema queda suficientemente enfocado tras el tema 1. En esta organización del curso retoma de una manera más formal y metódica las técnicas relativas al análisis sintáctico presentadas en el tema 2.

Objetivos. El objetivo general de este tema es que el alumno pueda diseñar e implementar, utilizando métodos adecuados, analizadores sintácticos eficientes. Más concretamente se pretende que el alumno:

- Domine los principios básicos del análisis predictivo, tanto ascendente como descendente.
- Pueda utilizar con fluidez las gramáticas independientes del contexto no sólo como herramienta de especificación de lenguajes sino también para la construcción de analizadores sintácticos. En consecuencia debe, por ejemplo, conocer los problemas que conlleva trabajar con gramáticas ambiguas y las posibles soluciones.
- Sea capaz de construir analizadores recursivos predictivos y de obtener gramáticas que cumplan las condiciones LL(1) y pueda incluir un mecanismo de recuperación de errores en sus analizadores.
- Comprenda el proceso de análisis LR, los principios que lo sustentan, los problemas que se plantean y las soluciones a dichos problemas.
- Ser capaz de construir un analizador sintáctico sencillo utilizando Yacc.

Tema 5: Gestión de memoria en tiempo de ejecución.

5.1 Introducción y asignación estática.

5.2 Asignación dinámica de memoria utilizando una pila
- Acceso a variables no locales.
- Paso de parámetros.

5.3 Asignación dinámica de memoria: "Heap allocation"
- Asignación y desasignación explícita e implícita de memoria.
- Técnicas para la asignación dinámica de la memoria.

Referencia básica: Capítulo 7 de [Aho, 90; Aho, 06]

Referencias complementarias: Capítulo 10 de [Sanchis, 86].

Objetivos. El objetivo general de este tema es que el alumno pueda construir un traductor que genere código para una máquina abstracta con direccionamiento a posiciones de memoria. Este objetivo debe ser posible considerando lenguajes que permitan la asignación estática de memoria o aquellos que soporten la recursividad, la imbricación de bloques o la asignación dinámica de memoria en tiempo de ejecución.

Objetivos detallados

- Identificar claramente las implicaciones que desde el punto de vista de la gestión de memoria en tiempo de ejecución tienen determinadas características de los lenguajes de programación.

- Poder especificar por medio de un E.T.D.S. el proceso de generación de código para traductores de lenguajes en que es posible la asignación estática o por medio de una pila de la memoria.

- Asociar, para lenguajes que permiten el anidamiento de bloques, la gestión de la tabla de símbolos durante la validación semántica con el proceso de generación de código.

- Conocer las técnicas básicas de implementación de la *recolección de basuras* y pueda plantearse el diseño de un recolector de memoria asociado a un intérprete.