COMPILADORS II

Enginyeria Informàtica

Troncal: 4,5 crèdits (3+1,5)

OBJETIVOS DE LA ASIGNATURA

Comprender las estrategias avanzadas utilizadas por compiladores e intérpretes. Profundizar en las técnicas de optimización de código intermedio.

Temario

- **TEMA 1. CÓDIGO INTERMEDIO**
  - [Lecciones 1 y 2.] **FORMATOS INTERNOS INTERMEDIOS DE LOS PROGRAMAS FUENTE**. Notación polaca. n-tuplas. Árboles sintácticos abstractos. Código enhebrado. Código para máquinas abstracta.
- **TEMA 2. GENERACIÓN DE CÓDIGO PARA ESTRUCTURAS COMPLEJAS.**
  - [Lección 3.] **GENERACIÓN DE CÓDIGO PARA ESTRUCTURAS DE DATOS COMPLEJAS.** Arrays de tamaño variable. Sparse arrays. Diccionarios.
  - [Lección 4.] **GENERACIÓN DE CÓDIGO PARA ESTRUCTURAS DE CONTROL COMPLEJAS.** Goto normal, asignado, calculado. Case.
- **TEMA 3. OPTIMIZACIÓN DE CÓDIGO.**
  - [Lección 5.] **OPTIMIZACIONES BÁSICAS.** Introducción. Bloques básicos. Ensamblamiento. Eliminación de subexpresiones redundantes.
  - [Lecciones 6 y 7.] **OPTIMIZACIONES DENTRO DE BUCLES**. Expansión de bucles. Reducción de frecuencia. Reducción de potencia.
  - [Lección 8]. **OPTIMIZACIÓN GLOBAL**. Construcción del grafo de flujo. Análisis del grafo de flujo. Aplicaciones de la optimización de programas.
  - [Lección 9]. **OPTIMIZACIÓN DEPENDIENTE DE MÁQUINA**. Introducción. Asignación de Registros. Arquitectura máquina y Generación de Código Real.
- **TEMA 4. GESTIÓN COMPLEJA DE MEMORIA.**
  - [Lecciones 10, 11, 12 y 13.] **ASIGNACIÓN DINÁMICA DE MEMORIA**. Liberación explícita. Liberación implícita: contador de referencias, garbage collection, compactación.
- **TEMA 5. ANÁLISIS SEMÁNTICO AVANZADO.**
  - [Lecciones 14 y 15.] **OBJETOS, SOBRECARGA Y HERENCIA.**
  - [Lección 16.] **TIPOS PARAMÉTRICOS**.
  - [Lección 17.] **INFERENCIA DE TIPOS.** Reglas de inferencia de tipos y Algoritmos.
- **TEMA 6. INTERPRETES.**

- [Lección 18.] **INTRODUCCIÓN.** Ciclo de interpretación. Estrategias de interpretación: Código intermedio y compilador-intérprete. Organización de memoria usuales en los intérpretes..
- [Lección 19.] **INTERPRETACIÓN ITERATIVA.** Interpretación de código máquina. Interpretación de lenguajes de comandos. Interpretación de lenguajes imperativos. Ejemplo: Basic
- [Lección 20]. **FUNCIONALES.** LISP: Ciclo de interpretación. Nodo LISP. Listas. Símbolos. Listas de propiedad. Entornos y ligaduras. Garbage collection.
- [Lección 21]. **LÓGICOS**. PROLOG: Pila de backtracking. Árbol sintáctico abstracto y constructores. Unificación. tratamiento del desconocido. Ciclo de interpretación de PROLOG.

- **TEMA 7. ESPECIFICACIÓN FORMAL DE LENGUAJES.**
  - [Lecciones 22, 23 y 24.] **SEMÁNTICA INTERPRETATIVA**. Máquina virtual. Cálculo. Resultado. Estado. Construcción abstracta. Lenguaje de la máquina virtual. Construcción de la semántica interpretativa para un lenguaje. utilización de una semántica interpretativa.
  - [Lecciones 25, 26, 27 y 28.] **GRAMÁTICAS DE ATRIBUTOS**. Atributo. Atributos calculados. Atributos heredados. Condiciones. Funciones. Gramática de atributos para un lenguaje de programación.
- **TEMA 8. GENERADORES DE COMPILADORES Y HERRAMIENTAS.**
  - [Lección 29.] **GENERADORES DE PARSERS**. Introducción. Generadores de parsers. Generadores de compiladores. YACC.
  - [Lección 30.] **OTRAS HERRAMIENTAS**. Entornos de compilación. Bancos de trabajo. Depuradores. Editores guiados por la sintaxis.

BIBLIOGRAFÍA

- **BIBLIOGRAFÍA BÁSICA**

  - ***Advanced Compiler Design & Implementation*. Steven S. Muchnick. Morgan Kaufmann Publishers, 1997.**
  - ***Building an Optimizing Compiler*. Robert Morgan, Butterworth-Heinemann, 1998.**
  - ***Modern Compiler Design*. Dick Grune et al. Wiley, 2000.**
  - **Warren's Abstract Machine. A TUTORIAL RECONSTRUCTION (410 Kb).**
    HASSAN AÏT-KACI
    Intelligent Software Group. School of Computing Science. Simon Fraser University. Burnaby, British Columbia. V5A 1S6, Canada.
  - ***The Theory and Practice of Compiler Writing*, Jean-Paul Tremblay & Paul G. Sorenson, MCGRAW-HILL, 1985.**
  - ***Programming Language Concepts and Paradigms*, David A. Watt, Prentice Hall, 1990.**
  - ***Thèorie des programmes. Schémes, preuves, sémantique*, C. Livercy, Dunod, 1984.**
  - ***Formal Specification of Programming Languages: A Panoramic Primer*, Frank G. Pagan, Prentice Hall Inc., 1981.**

- **BIBLIOGRAFÍA DE CONSULTA**

  - ***Compiladores: Principios, técnicas y herramientas*, Alfred V. Aho, Ravi Sethi & Jeffrey D. Ullman, Addison-Wesley, 1990.**
  - ***Programming Language Processors*, David A. Watt, Prentice Hall, 1993.**
  - ***Functional Programming*, Anthony J. Field & Peter G. Harrison, Addison-Wesley, 1988.**
  - ***Crafting a Compiler with C, Charles N. Fisher & Richard J. Leblanc jr.*, The Benjamin / Cummings Publishing Company inc., 1991.**
  - ***Programming Languages: Design and Implementation*, Terrence W. Pratt, Prentice Hall International Editions, 1984.**
  - ***Compiler Construction. An Advanced Course*, Edited by G. Goos and J. Hartmanis, Springer-Verlag, 1974.**
  - ***The Design of an Optimizing Compiler*, William Wulf et all., North Holland, 1980.**
  - ***An Implementation Guide to Compiler Writing*, Jean-Paul Tremblay & Paul G. Sorenson, MCGRAW-HILL, 1982.**
  - ***Writing Compilers & Interpreters: An Applied Approach*, Ronald Mak, Wiley, 1991.**
  - ***Writing Interactive Compilers and Interpreters*, J. P. Brown, Wiley, 1979.**
  - ***Compiler Design in C*, Allen Y. Holub, Prentice Hall, 1990.**
  - ***Garbage Collection*, Richard Jones y Rafael Lins, Wiley 1996.**