UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE CIENCIAS EXACTAS, FÍSICO-QUÍMICAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

CARRERA/S: Licenciatura en Ciencias de la Computación (Cód. 14)

PLAN DE ESTUDIOS: Año 1999 – Versión 1

ASIGNATURA: Taller de Diseño de Software **CÓDIGO: 3306**

DOCENTE RESPONSABLE: Francisco Bavera

EQUIPO DOCENTE: Francisco Bavera, Doctor en Ciencias de la Computación
Cecilia Kilmurray, Licenciada en Ciencias de la Computación

AÑO ACADÉMICO: 2015

REGIMEN DE LA ASIGNATURA: cuatrimestral

RÉGIMEN DE CORRELATIVIDADES:

Aprobada	*Regular*
3303	3304
1959	

CARGA HORARIA TOTAL: 210 hs

TEÓRICAS: 15 hs **PRÁCTICAS:** 15 hs **LABORATORIO:** 180 hs

CARÁCTER DE LA ASIGNATURA: Obligatoria

A. CONTEXTUALIZACIÓN DE LA ASIGNATURA

Segundo cuatrimestre de cuarto año.

B. OBJETIVOS PROPUESTOS

- Capacidad de aplicar los conocimientos adquiridos hasta el momento a la construcción de soluciones computacionales concretas, autodocumentadas, eficaces y eficientes.
- Capacidad autónoma de adquirir destreza en el uso de nuevos herramientas de desarrollo de software.
- Habilidad en el manejo del ambiente de desarrollo Linux.
- Habilidad en el uso del lenguaje Java.
- Capacidad de abordar el diseño de sistemas complejos.
- Capacidad de realizar la descomposición modular de un proyecto y de trabajar en grupo.
- Habilidad en el uso de herramientas para la construcción de compiladores y otros procesadores de lenguajes.
- Habilidad para realizar el *testing* modular y global de un sistema complejo
- Capacidad de redactar documentación técnica.
- Capacidad comprender y utilizar bibliografía escrita en inglés.

C. CONTENIDOS BÁSICOS DEL PROGRAMA A DESARROLLAR

Mediante el desarrollo de un proyecto que dé solución a un problema real, se deberá integrar los conocimientos adquiridos en las asignaturas cursadas. Se deberán desarrollar todas las etapas y producir toda la documentación referente al proyecto, de acuerdo a los estándares en uso.

El proyecto consiste en diseñar y desarrollar un compilador para un subconjunto de un lenguaje de programación. El proyecto se divide en 6 etapas que abarcan todo el proceso de compilación: análisis léxico, análisis sintáctico, análisis semántico, generación de código intermedio, generación de código objeto (assembly) y optimización del código. Cada una de las etapas consta de diseño, implementación y testing de los artefactos desarrollados. Los contenidos básicos a desarrollar son aquellos relacionados con cada etapa: análisis léxico, análisis sintáctico, análisis semántico, código intermedio, código objeto, generación de código intermedio y objeto y optimización del código.

D. FUNDAMENTACIÓN DE LOS CONTENIDOS

Los temas involucrados en el diseño e implementación de Compiladores conforman una parte muy relevante en la currícula de Ciencias de la Computación. El desarrollo de un compilador involucra gran cantidad de técnicas y métodos que permite aplicar una cantidad considerable de conceptos teóricos y de implementar técnicas y algoritmos en casos específicos.

En la materia Taller de Diseño de Software, se aplican un número considerable de técnicas, muchas de estas técnicas son casos particulares de técnicas fundamentales. Pero, se deja librado al alumno: (1) asociar la aplicación de determinada técnica en otros contextos; (2) determinar, o no, la generalización de la técnica; y (3) investigar, o no, otras alternativas de diseño e implementación.

Se guía al alumno para que: (1) pueda vislumbrar que el campo de aplicación de las técnicas dadas es más amplio; y (2) reconozca que existen una gran cantidad de técnicas y algoritmos, no todas vistas en clase, para diseñar e implementar un compilador.

Los contenidos fueron seleccionados con el fin de que el alumno cuente con los conocimientos necesarios para culminar exitosamente el proyecto. Las actividades se seleccionaron con el fin de cumplir con los objetivos propuestos, los cuales, todos contribuyen a fortalecer el perfil del egresado y su práctica profesional.

Se hace especial hincapié en desarrollar la autonomía del alumno para aprender y utilizar las herramientas involucradas en el desarrollo del proyecto (JFlex, CUP, bash, Java).

La forma de evaluación consta de tres partes: (1) seguimiento del desarrollo del proyecto (puntualidad en las entregas, desempeño, participación, calidad, claridad, metodologías usadas y justificación de las actividades realizadas); (2) evaluación, siguiendo los mismos lineamientos, de la entrega final; y (3) evaluación en el desempeño de los alumnos en la resolución de problemas.

Conocimientos esperados de los alumnos de las materias correlativas: métodos y herramientas de diseño y análisis de sistemas. Métodos y prácticas de testing. Documentación.

Otros conocimientos necesarios de materias previas:
Algoritmos I: Estructuras de datos. Implementación de listas (simple y doblemente encadenadas), pilas, diccionarios y tablas de Hashing.
Análisis Comparativo de Lenguajes: conocimientos básicos del Lenguaje Java, manejo de memoria dinámica, pasaje de parámetros y frames de ejecución, conceptos de: compiladores, interpretes, ensambladores y linkers.
Organización del Procesador: Lenguajes ensambladores.

E. ACTIVIDADES A DESARROLLAR

La asignatura está organizada como un taller o laboratorio. En ella mediante la realización de un proyecto (construcción de un compilador para un subconjunto de un lenguaje de programación), se revisan y utilizan los conceptos estudiados en otras asignaturas y más específicamente en Autómatas y Lenguajes, Algoritmos I y Análisis Comparativo de Lenguajes.

CLASES TEÓRICAS: Introducción de los temas, conceptos teóricos y ejemplos de casos prácticos de su uso. Carga horaria: 15 hs. totales, divididas en 6 clases.

CLASES PRÁCTICAS: Resolución de problemas. Carga horaria: 15 hs. totales, divididas en 7 clases.

CLASES DE TRABAJOS PRÁCTICOS DE LABORATORIO: Implementación del proyecto, el mismo esta dividido en 7 etapas, lo que permite desarrollarlo de manera incremental. Cada etapa cuenta con actividades de revisión bibliográfica, diseño, implementación y testing de los artefactos desarrollados.

F. NÓMINA DE TRABAJOS PRÁCTICOS

Trabajo Práctico Nº1: Análisis Léxico y Sintáctico
Trabajo Práctico Nº2: Análisis Semántico
Trabajo Práctico Nº3: Traducción Dirigida por la Sintaxis
Trabajo Práctico Nº4: Generación de Código Intermedio
Trabajo Práctico Nº5: Generación de Código Objeto

G. HORARIOS DE CLASES:

HORARIOS DE CLASES: Lunes y Miercoles de 10 hs a 14 hs.

HORARIO DE CLASES DE CONSULTAS: A coordinar.

H. MODALIDAD DE EVALUACIÓN:

- **Evaluaciones Parciales:** Entrega de cada una de las etapas del proyecto. En cada etapa se fundamentalmente la capacidad de resolución de problemas y de implementación de soluciones utilizando los contenidos introducidos. Como las etapas son incrementales en cada etapa se deben incluir las correcciones de la etapa anterior.

- **Evaluación Final:** Exposición y defensa del proyecto y un examen oral. Se evalúa fundamentalmente la adquisición de los conceptos fundamentales, su vinculación con el resto de la carrera y la capacidad de aplicarlos.

- **CONDICIONES DE REGULARIDAD:** Aprobación de las entregas parciales y culminación de la implementación del proyecto.

- **CONDICIONES DE PROMOCIÓN:** Aprobación con nota mayor a 7 del proyecto, entregas puntuales de cada etapa, culminación de la implementación del proyecto, presentación del informe del proyecto, exposición y defensa del mismo.

PROGRAMA ANALÍTICO

A. CONTENIDOS

Unidad 0. Introducción. Compiladores e interpretes. Fases del compilador. Diseño de un compilador. Ensambladores. Enlazadores.

Unidad I. Gramáticas de atributos. Atributos sintetizados y heredados. Árbol decorado de una sentencia, evaluación, grafo de dependencia, gramáticas evaluables. Definiciones guiadas por sintaxis. Esquemas de traducción. Construcción de árboles sintácticos.

Unidad II. Aplicación de las Definiciones guiadas por sintaxis a la verificación de tipos y la generación de código. Uso de herramientas lex y yacc.

Unidad III. Código Intermedio. Ventajas de introducir una fase de código intermedio. Lenguajes de código intermedio: Usando Grafos y Árboles Sintácticos. Máquinas Abstractas, Máquinas Pila y máquina de tres direcciones. Generación de código para las distintas representaciones.

Unidad IV. Generación de código objeto. Criterios de optimización. Bloques básicos de un programa. Determinación de bloque básicos. Optimizaciones avanzadas. Análisis estático de código. Generación de código objeto en lenguaje ensamblador.

B. CRONOGRAMA DE CLASES Y PARCIALES

Semana	Día/ Fecha	Teóricos	Día/ Fecha	Prácticos	Día/ Fecha	Laboratorios	Parciales / Recup.
1	Mierc 19/08	Introducción - Análisis Léxico			Mierc 19/08	Presentación del Proyecto. Etapa 1: Análisis Léxico	
2	Lunes 24/08	Análisis Sintáctico	Lunes 24/08	Análisis Léxico y Sintáctico			
2			Mierc 26/08	Consulta Prácticas	Mier 26/08	Etapa 2: Definición de la gramática y analizador sintáctico	
3	Lunes 31/08	Traducción dirigida por las sintaxis - Análisis Semántico	Mierc 02/09	Traducción dirigida por la sintáxis			
4	Lunes 07/09	Código Intermedio	Lunes 07/09	Código Intermedio	Mierc 09/09	Etapa 3: Diseño e implementación análisis semántico	Entrega Etapa 1 y 2.
5			Lunes 14/09	Revisión Prácticas	Mierc 16/09	Revisión Proyecto	
6	Mierc 23/09	Código Objeto			Mierc 23/09	Proyecto	
7	Lunes 28/09	Análisis de programas: optimización	Mierc 30/09	Consulta Prácticas	Mierc 30/09	Etapa 4: Generación codigo intermedio	Entrega Etapa 3.
8			Lunes 05/10	Consulta Prácticas	Mirec 07/10	Etapa 5: Generación código objeto enteros	Entrega etapa 4.
9					Mierc 14/10	Proyecto	

10					**Mierc 21/10**	**Proyecto**	
11					**Lunes 26/10**	**Proyecto**	
11					**Mierc 28/10**	**Etapa 6 Generación código objeto reales**	**Entrega Etapa 5.**
12					**Lunes 02/11**	**Proyecto**	
12					**Mierc 04/11**	**Etapa 7: optimización**	**Entrega Etapa 6.**
13					**Lunes 09/11**	**Proyecto**	
13					**Mier 11/11**	**Proyecto**	
14					**Lunes 16/11**	**Testing**	
14					**Mierc 18/11**		**Entrega final**

(Recordar las fechas de parciales deberán ser consensuadas con los responsables de las demás asignaturas del cuatrimestre correspondiente, en acuerdo con la Res. C.S. 356/10)

C. BIBLIOGRFÍA

De lectura obligatoria:

- "Compilers: Principles, Techniques, and Tools" Second Edition, by Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman and Monica S. Lam. 2006.
- Modern Compiler implementation in Java. Andrew Appel, Cambridge University Press, 1998.
- Documentación (on line) de las herramientas del sistema operativo usadas *gnu assembly language, Java, gnu debbuger, JFlex, CUP.*

De consulta:

- Advanced Compiler Design and Implementation. Steven Muchnick. Morgan Kaufmann, 1997.
- Modern Compiler Design. Grune, Bal, Jacobs,Langendoen. John Wiley & Sons. 2001.
- Assembly Language and Programming, Peter Abel. Prentice Hall, 1995
- " A retargetable C Compiler. Fraser & Hamson", Cumming 1995
- "An introduction to Compiler Constructions". Waite, Carter . Harper Collins 1992
- "lex & yacc" J. Levine, T. Mason & D Brown. O'Reilly & Associates 1992
- "The C programming language". Kernighan Ritchie. Prentice Hall 1988.