UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD POLITÉCNICA
INGENIERIA EN INFORMÁTICA
PLAN 2008
PROGRAMA DE ESTUDIO
Resolución Nº 17/04/50-00 Acta Nº 992/20/02/2017

I. IDENTIFICACIÓN

1.	Asignatura	: Diseño de Compiladores
2.	Semestre	: Décimo
3.	Horas semanales	: 7 horas
	3.1. Clases teóricas	: 4 horas
	3.2. Clases prácticas	: 3 horas
4.	Total real de horas disponibles	: 112 horas
	4.1. Clases teóricas	: 64 horas
	4.2. Clases prácticas	: 48 horas

II. JUSTIFICACIÓN

Los principios y técnicas aplicadas a las problemáticas de los compiladores computacionales tienen una importancia fundamental en la formación adecuada del estudiante de Ingeniería Informática.

Si bien es probable que durante el desempeño profesional del ahora estudiante de Ingeniería, pocas veces se enfrente ante el desafío del diseño y/o mantenimiento de un compilador, los conocimientos aprendidos se convierten en herramientas que metodológicamente aplicadas permite enfrentar y resolver una serie de situaciones prácticas relacionados con el diseño del software, análisis y programación estructura, así como teoría de las estructuras de datos aplicadas al tratamiento de problemas reales.

III. OBJETIVOS GENERALES

Al término de la asignatura el estudiante podrá:

1. Aplicar los principios y técnicas del curso a una amplia área de sus responsabilidades profesionales, basados en metodologías y experiencias particulares que vaya adquiriendo durante el desarrollo programático.
2. Enfatizar en la problemática del desarrollo de los analizadores léxicos y compiladores, buscando obtener una mejor comprensión y aumentar la capacidad de análisis y escritura de códigos fuentes de lenguajes de alto nivel en forma más eficiente y eficaz.
3. Identificar y manejar una serie de problemas generales y específicos de compiladores y analizadores de programas fuentes.

IV. OBJETIVOSESPECÍFICOS

1. Describir los principios de la teoría de Compiladores.
2. Aplicar los conocimientos teóricos mediante la programación de las fases de un compilador.
3. Aplicar técnicas de la teoría de Compiladores a problemas específicos.
4. Definir los fundamentos teóricos de la Ciencia Computacional.

CONOCIMIENTOS

1. Describir las fases de un Compilador, diferenciando traductores, intérpretes, máquinas abstractas/virtuales (*virtual machines*).
2. Explicar diferencias entre un código compilado e interpretado, y trabajar con gramáticas de lenguajes de programación de diferentes niveles de abstracción y paradigmas.
3. Comprender gramáticas de libre contexto modeladas mediante la notación BNF
4. Comprender un área de la Ciencia Computacional referente a la teoría de lenguajes, el reconocimiento de programas fuentes y la generación de código objeto.

HABILIDADES

1. Implementar reconocedores de lenguajes de programación (*parsers*) a partir de gramáticas de libre contexto y expresiones regulares.
2. Implementar generadores de código basados en reglas de traducción dirigida por sintaxis.
3. Modelar un lenguaje de programación utilizando la notación BNF.

COMPETENCIAS

1. Capacidad de aplicar los conocimientos teóricos en la práctica.
2. Capacidad de formular modelos mentales para la resolución de problemas mediante la aplicación de algoritmos de la teoría de lenguajes.
3. Capacidad para desarrollar soluciones de complejidad avanzada.

V. PRE - REQUISITO

1. Algoritmos y Estructuras de Datos III.
2. Estructura de los Lenguajes.

VI. CONTENIDO

6.1. Unidades programáticas

1. Introducción a la teoría de Compiladores.
2. Análisis Léxico y Tabla de Símbolos.
3. Análisis Sintáctico.
4. Traducción Dirigida por Sintaxis.
5. Verificación de Tipos.
6. Organización en Tiempo de Ejecución.
7. Generación de Código Intermedio.
8. Generación y Optimización de Código Máquina.

6.2. Desarrollo de las unidades programáticas

1. Introducción a la teoría de Compiladores.
   - 1.1. Compiladores.
   - 1.2. Análisis del programa fuente.
   - 1.3. Las fases de un compilador.
   - 1.4. Agrupamiento de fases.
   - 1.5. Herramientas para la construcción de compiladores.
   - 1.6. Compilador de una pasada.
     - 1.6.1. Perspectiva.
     - 1.6.2. Definición de la sintaxis.
     - 1.6.3. Traducción dirigida por la sintaxis.
     - 1.6.4. Análisis sintáctico.
     - 1.6.5. Traductor de expresiones simples.
2. Análisis Léxico y Tabla de Símbolos.
   - 2.1. Análisis léxico.
   - 2.2. Función del analizador léxico.
   - 2.3. Buffers de entrada.
   - 2.4. Especificación y reconocimiento de los componentes léxicos.
   - 2.5. Incorporación de una tabla de símbolos.
   - 2.6. Teoría de las tablas de símbolos.
3. Análisis Sintáctico.
   - 3.1. Rol del analizador sintáctico.
   - 3.2. Gramáticas independientes del contexto.
   - 3.3. Análisis sintáctico descendente y ascendente.
   - 3.4. Analizadores sintácticos y gramáticas LL, LR.
4. Traducción Dirigida por Sintaxis.
   - 4.1. Definiciones y construcción de árboles sintácticos.
   - 4.2. Evaluación descendente.
   - 4.3. Definiciones con atributos por la izquierda.
   - 4.4. Traducción descendente.
5. Verificación de Tipos.
   - 5.1. Sistema de tipos.
   - 5.2. Especificaciones de un comprobador de tipos sencillo.
6. Organización en Tiempo de Ejecución.
   - 6.1. Aspectos del lenguaje fuente.
   - 6.2. Organización de la memoria.
   - 6.3. Estrategias para la asignación de la memoria.
7. Generación de Código Intermedio.
   - 7.1. Lenguajes intermedios.
   - 7.2. Declaraciones.
   - 7.3. Proposición de asignación.
8. Generación y Optimización de Código Máquina.
   - 8.1. Diseño de un generador de código.
   - 8.2. La máquina objeto.
   - 8.3. Administración de la memoria durante la ejecución.
   - 8.4. Bloques básicos y grafos de flujo.
   - 8.5. Principales fuentes para la optimización del código

VII. ESTRATEGIAS METODOLÓGICAS

La metodología a utilizar se basa en:

1. Clases orientadas por el profesor para la exposición de conceptos y guías de estudio.
2. Ejercicios prácticos en clase o sala de máquinas, en forma individual o grupal.
3. Propuesta de temas de investigación y debate en clase.
4. Trabajo práctico
5. Clases con invitados profesionales especialistas en temas del contenido curricular del curso.
6. Como complementación, está previsto un trabajo práctico en el que se apliquen los conceptos desarrollados en el curso en la construcción de un compilador básico.

A. Medios Auxiliares

1. Pizarras acrílicas.
2. Proyectores multimedia.
3. Plataforma virtual “EDUCA”.
4. Ejercitarios.
5. Sala de máquinas.

VIII. EVALUACIÓN

La evaluación se llevará a cabo mediante un examen parcial escrito y un trabajo práctico grupal. El puntaje acumulado en ambas evaluaciones, de acuerdo a la escala vigente, habilitará al estudiante a acceder al examen final. Puntuaciones extra se darán en función a los trabajos de investigación y participación en clase.

IX. BIBLIOGRAFÍA

MATERIALES BIBLIOGRÁFICOS DISPONIBLES EN LA BIBLIOTECA DE LA FACULTAD POLITÉCNICA

- Aho, A. V., Lam, M. S., Sethi, R. & Ullman, J. D. (2008). *Compiladores: principios, técnicas y herramientas*. (2° Ed.). México: Pearson Educación.
- Grune, D., Bal, H. E., Jacobs C. J. H.& Langendoen (2007). *Diseño de compiladores modernos.* Madrid: McGraw-Hill.
- Lauden, K. C. (2004).*Construcción de compiladores: principios y prácticas*. México: Thomson
- Ruiz Catalán, J. (2010). *Compiladores: teoría e implantación*. México: Alfaomega

RECURSOS DISPONIBLES A TRAVÉS DE CICCO

- Cooper, K. D., & Torczon, L. (2004). *Engineering a Compiler*. San Francisco, Calif: Morgan Kaufmann. Recuperado de: http://eds.b.ebscohost.com
- Garrido Alenda, A. (2002). *Diseño de compiladores*. [Alicante]: Digitalia. Recuperado de: http://eds.b.ebscohost.com
- Singh, R., Sharma, V., & Varshney, M. (2009). *Design and Implementation of Compiler*. New Delhi: New Age International. Recuperado de: http://eds.b.ebscohost.com