UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
Escuela de Ciencias y Sistemas

FICHA TÉCNICA DEL CURSO: Organización de Lenguajes y Compiladores 2

<table>
<tr><td>No.</td><td colspan="3">Descripción</td></tr>
<tr><td>.</td><td>Código 781</td><td colspan="2">Créditos 6</td></tr>
<tr><td>1</td><td>Escuela
Ciencias y Sistemas</td><td>Área a la que pertenece:
Computación</td><td>Vigencia:

Primer Semestre 2026</td></tr>
<tr><td>2</td><td>periodos por semana
4</td><td colspan="2">Horario Sección A

Lunes de 7:10 a 8:50 y sábado de 12:10 a 13:50</td></tr>
<tr><td>3</td><td colspan="3">Prerrequisitos:
772 (Estructuras de Datos)
777 (Organización de Lenguajes y Compiladores 1)</td></tr>
<tr><td>4</td><td colspan="3">Posrequisito: 281 (Sistemas operativos 1)
972 (Inteligencia Artificial 1)</td></tr>
<tr><td>5</td><td colspan="3">Sección: A</td></tr>
<tr><td>6</td><td colspan="3">
I. Descripción General
Este curso es la continuación del estudio de las fases de un Compilador, específicamente el análisis de semántica y la fase de síntesis. Se tratan con detalle las definiciones dirigidas por la sintaxis, el manejo de la tabla de símbolos, la generación de código intermedio y optimización de código

Se desarrollarán dos proyectos para aplicar los conceptos generales de compiladores, usando herramientas básicas tales como generadores de analizadores de léxico y de sintaxis.

II. Competencia General
Diseña e implementa compiladores o intérpretes de lenguajes de alto nivel mediante el uso de herramientas especializadas para la construcción de analizadores léxicos, sintácticos y semánticos, asegurando la generación de código intermedio funcional y optimizado.

III. Objetivos
Objetivo General
Comprender y aplicar los conceptos fundamentales de las fases que integran un compilador, con énfasis en el análisis semántico, la generación de código intermedio y la optimización del código, para fortalecer el diseño y la eficiencia de procesos de traducción de lenguajes.

Objetivos Específicos
1. Establecer una base teórica sólida que fundamente el diseño de compiladores para lenguajes de alto nivel, a partir del estudio de sus componentes fundamentales.
2. Desarrollar proyectos aplicados que incorporen los conceptos esenciales del proceso de compilación, promoviendo el aprendizaje práctico y contextualizado.
3. Implementar herramientas de análisis léxico, sintáctico y semántico en la construcción de compiladores o intérpretes, que faciliten la transformación eficiente de lenguajes de alto nivel.

III. Contenido

1. Traducción dirigida por la sintaxis
1.1. Definiciones dirigidas por la sintaxis
1.1.1. Atributos heredados y sintetizados
1.1.2. Evaluación de una definición dirigida por la sintaxis en los nodos de un árbol sintáctico
1.2. Órdenes de evaluación para las definiciones dirigidas por la sintaxis
1.2.1. Gráficos de dependencias
1.2.2. Orden de evaluación
1.2.3. Definiciones con atributos sintetizados
1.2.4. Definiciones con atributos heredados
1.3. Aplicaciones de la traducción orientada por la sintaxis
1.3.1. Construcción de árboles de análisis sintáctico
</td></tr>
</table>

1.3.2. La estructura de tipos
1.4. Esquemas de traducción orientados por la sintaxis
1.4.1. Esquemas de traducción postfijos
1.4.2. Implementación de esquemas de traducción orientados a la sintaxis postfijo con la pila
1.4.3. Esquema de traducción orientados a la sintaxis con acciones dentro de producciones
1.4.4. Eliminación de la recursividad por la izquierda de los esquemas de traducción
1.4.5. Esquemas de traducción orientados a la sintaxis para definiciones con atributos heredados por la izquierda
1.5. Implementación de definiciones dirigidas por la sintaxis con atributos heredados por la izquierda
1.5.1. Traducción durante el análisis sintáctico de descenso recursivo
1.5.2. Generación de código al instante
1.5.3. Las definiciones dirigidas por la sintaxis con atributos heredados por la izquierda y análisis sintáctico LL
1.5.4. Análisis sintáctico ascendente de las definiciones dirigidas por la sintaxis con atributos heredados por la izquierda

2. Generación de código intermedio
2.1. Variantes de los árboles sintácticos
2.1.1. Grafo dirigido acíclico para expresiones
2.1.2. Método número de valor para GDA
2.2. Código de tres direcciones
2.2.1. Direcciones e instrucciones
2.2.2. Cuádruplos
2.2.3. Tripletas
2.2.4. Forma de asignación individual estática
2.3. Tipos y declaraciones
2.3.1. Expresiones de tipos y equivalencias
2.3.2. Declaraciones y distribución de almacenamiento
2.3.3. Secuencias de las declaraciones
2.3.4. Campos en registros
2.4. Traducción de expresiones
2.4.1. Operaciones dentro de expresiones
2.4.2. Traducción incremental
2.4.3. Direccionamiento de los elementos de un arreglo
2.4.4. Traducción de referencias a arreglos
2.5. Comprobación de tipos
2.5.1. Reglas para la comprobación de tipos
2.5.2. Conversiones de tipos
2.5.3. Sobrecarga de funciones y operadores
2.5.4. Inferencia de tipos y funciones polimórficas
2.5.5. Un algoritmo para la unificación
2.6. Flujo de control
2.6.1. Expresiones booleanas
2.6.2. Código de corto circuito
2.6.3. Instrucciones de flujo de control
2.6.4. Traducción del flujo de control de las expresiones booleanas
2.6.5. Evitar goto redundantes
2.6.6. Valores booleanos y código de salto
2.7. Parcheo de retroceso
2.7.1. Generación de código de una pasada
2.7.2. Técnica de retroceso
2.7.3. Instrucciones de flujo de control
2.8. Instrucciones switch
2.8.1. Traducciones de switch
2.8.2. Traducción orientada por la sintaxis de switch
2.9. Código intermedio para procedimientos

3. Optimización de código
3.1. Optimización de bloques básicos
3.1.1. Representación GDA
3.1.2. Búsqueda de subexpresiones locales comunes
3.1.3. Eliminación de código muerto
3.1.4. Uso de identidades algebraicas
3.1.5. Representación de referencias a arreglos
3.1.6. Asignación de apuntadores y llamadas a procedimientos

3.1.7. Reensamblado de bloques básicos
3.2. Optimización de mirilla
3.2.1. Eliminación de instrucciones redundantes
3.2.2. Eliminación de código inalcanzable
3.2.3. Optimizaciones de flujo de control
3.2.4. Simplificación algebraica y reducción por fuerza

IV. Metodología:

La asignatura se desarrollará mediante una combinación de estrategias didácticas que fomentan el aprendizaje significativo. Se utilizará la clase magistral para la exposición y análisis de los fundamentos teóricos esenciales. Asimismo, se promoverá la resolución de problemas y el autoestudio como herramientas para afianzar la comprensión de los contenidos y estimular la autonomía intelectual del estudiante.

El proceso formativo se complementará con la realización de proyectos aplicados y actividades de laboratorio, que permitirán poner en práctica los conocimientos adquiridos, desarrollar habilidades técnicas y fortalecer el pensamiento crítico en contextos reales de programación y compilación.

V. Evaluación:

La evaluación se divide en dos componentes principales:

1. Trabajo de Laboratorio (32 puntos):
Se asignarán mediante la ejecución de dos proyectos prácticos, que permitirán aplicar y consolidar los conocimientos técnicos desarrollados en el curso.

2. Evaluación Teórica (68 puntos):
Compuesta por los siguientes instrumentos de evaluación:

Tres evaluaciones parciales: 36 puntos
Exámenes cortos: 7 puntos
Examen final: 25 puntos

Criterios de Aprobación:
Para aprobar el curso, el estudiante deberá cumplir con los siguientes mínimos requeridos:
Laboratorio: Obtener al menos 19.52 puntos sobre los 32 posibles.
Zona: Alcanzar un mínimo de 36 puntos

CALENDARIO DE EXÁMENES

Primer Examen Parcial	21 de febrero UNIDAD 1. Traducción dirigida por la sintaxis
Segundo Examen Parcial	21 de marzo UNIDAD 2. Generación de código intermedio
Tercer Examen Parcial	25 de abril UNIDAD 2. Código intermedio para procedimientos UNIDAD 3. Optimización de código
Examen Final	De acuerdo con el calendario oficial TODAS LAS UNIDADES

Observaciones:
Dirección de correo electrónico para consultas:
Ing. Bayron López: blopezw@yahoo.com

7	Bibliografía	Libro de Texto: Compiladores. Principios, Técnicas y Herramientas Aho, Sethi y Ullmam. PEARSON ADDISON-WESLEY, 2008, segunda edición.
8	Tutor	Diego Cali diegocali123@gmail.com