

**FICHA TÉCNICA DEL CURSO:** Organización de Lenguajes y Compiladores 2

No.	Descripción		
	<b>Código</b> 781	<b>Créditos</b> 6	
1	<b>Escuela</b> Ciencias y Sistemas	<b>Área a la que pertenece:</b> Computación	<b>Vigencia:</b> Segundo Semestre 2025
2	<b>Horas por semana</b> 4	<b>Horarios</b> A : Lunes de 7:10 a 8:50 hrs. y Sábado de 12:10 a 13:50 hrs. B: Martes de 7:10 a 10:30 N: Martes y Jueves de 19:00-20:40	
3	Prerrequisitos: 772 (Estructuras de Datos) 777 (Organización de Lenguajes y Compiladores 1)		
4	Posrequisito: 281 (Sistemas operativos 1)		
5	Secciones: A, B, N		
6	<p><b>I. Descripción General</b>                      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</p> <p>Se desarrollarán dos proyectos para aplicar los conceptos generales de compiladores, usando herramientas básicas tales como generadores de analizadores de léxico y de sintaxis.</p> <p><b>II. Objetivos</b></p> <ul style="list-style-type: none"> <li>- <b>Objetivo General</b> <ol style="list-style-type: none"> <li>1. Desarrollar los conceptos básicos de las fases de un compilador.</li> </ol> </li> <li>- <b>Objetivos Específicos</b> <ol style="list-style-type: none"> <li>1. Proveer una base teórica que permita diseñar un compilador para un lenguaje de alto nivel.</li> <li>2. Aplicar los conceptos de compiladores en el desarrollo de proyectos.</li> <li>3. Utilizar las herramientas de análisis de léxico, sintáctico y semántico, para la construcción de compiladores o intérpretes, de un lenguaje de alto nivel.</li> </ol> </li> </ul> <p><b>III. Contenido</b></p> <ol style="list-style-type: none"> <li>1. Traducción dirigida por la sintaxis                             <ol style="list-style-type: none"> <li>1.1. Definiciones dirigidas por la sintaxis                                     <ol style="list-style-type: none"> <li>1.1.1. Atributos heredados y sintetizados</li> <li>1.1.2. Evaluación de una definición dirigida por la sintaxis en los nodos de un árbol sintáctico</li> </ol> </li> <li>1.2. Órdenes de evaluación para las definiciones dirigidas por la sintaxis                                     <ol style="list-style-type: none"> <li>1.2.1. Gráficos de dependencias</li> <li>1.2.2. Orden de evaluación</li> <li>1.2.3. Definiciones con atributos sintetizados</li> <li>1.2.4. Definiciones con atributos heredados</li> </ol> </li> <li>1.3. Aplicaciones de la traducción orientada por la sintaxis                                     <ol style="list-style-type: none"> <li>1.3.1. Construcción de árboles de análisis sintáctico</li> <li>1.3.2. La estructura de tipos</li> </ol> </li> <li>1.4. Esquemas de traducción orientados por la sintaxis                                     <ol style="list-style-type: none"> <li>1.4.1. Esquemas de traducción postfijos</li> <li>1.4.2. Implementación de esquemas de traducción orientados a la sintaxis postfijo con la pila</li> <li>1.4.3. Esquema de traducción orientados a la sintaxis con acciones dentro de producciones</li> <li>1.4.4. Eliminación de la recursividad por la izquierda de los esquemas de</li> </ol> </li> </ol> </li> </ol>		

	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.	Parqueo 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:**

Clase Magistral para explicación de teoría.  
 Resolución de tareas, problemas y auto estudio  
 Práctica, realización de proyectos.  
 Actividades de laboratorio

**V. Evaluación:**

32 puntos para laboratorio, correspondiente a dos proyectos.  
 68 puntos de la parte teórica, que incluye tres parciales equivalentes a 36 puntos, 7 puntos de otras actividades y 25 puntos del examen final.

**Para aprobar el curso es necesario obtener como mínimo 21.96 puntos de laboratorio y como mínimo 36 puntos de zona.**

**Calendarización**

Unidad 1	16, 23, 30 de julio 6 de agosto
Primer Examen Parcial	13 de agosto
Unidad 2	20, 27 de agosto 3, 9 de septiembre
Segundo Examen Parcial	17 de septiembre
Congresos	24 de septiembre
Unidad 3	1, 8, 15, 22 de octubre
Tercer Examen Parcial	29 de octubre
Examen Final	De acuerdo con el calendario oficial, del 2 al 15 de noviembre

**Observaciones:**

Direcciones de correo electrónico para consultas:

Ing. Bayron López: [blopezw@yahoo.com](mailto:blopezw@yahoo.com)  
 Ing. Luis Espino: [usac.sistemas@gmail.com](mailto:usac.sistemas@gmail.com)  
 Ing. Edgar Sabán : [edgarsaban@gmail.com](mailto:edgarsaban@gmail.com)

7	Bibliografía	Libro de Texto: Compiladores. Principios, Técnicas y Herramientas Aho, Sethi y Ullmam. PEARSON ADDISON-WESLEY, 2008, segunda edición.
8	No. De Secciones	Tres
19	Catedráticos y tutores académicos	Titulares: Ing. Bayron López, Ing. Luis Espino, Ing. Edgar Sabán Tutores: