

FICHA TÉCNICA DEL CURSO: Organización de Lenguajes y Compiladores 2

No.	Descripción		
	Código 781	Créditos 5	
1	Escuela Ciencias y Sistemas	Área a la que pertenece: Computación	Vigencia: Segundo Semestre 2022
2	Horas por semana 4	Horarios A : Lunes de 7:10 a 8:50 hrs. y Sábado de 12:10 a 13:50 hrs. N: Martes y Jueves de 19:00 a 20:40 hrs D: Martes de 7:10 a 10:30 hrs	
3	Prerrequisitos: 772 (Estructuras de Datos) 777 (Organización de Lenguajes y Compiladores 1)		
4	Posrequisito: 281 (Sistemas operativos 1)		
5	Secciones: A, N, D		
6	<p>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</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>II. Competencia General Que el estudiante diseñe e implemente un compilador o un intérprete de un lenguaje de alto nivel utilizando herramientas para construir los analizadores léxico, sintáctico, y semántico, además genere código intermedio y lo optimice.</p> <p>III. Objetivos Objetivo General Desarrollar los conceptos básicos de las fases de un compilador, específicamente en la fase de análisis semántico, generación de código intermedio y optimización de código.</p> <p>Objetivos Específicos</p> <ol style="list-style-type: none"> 1. Proveer una base teórica que permita diseñar un compilador para un lenguaje de alto nivel. 2. Aplicar los conceptos de compiladores en el desarrollo de proyectos. 3. Utilizar las herramientas de análisis de léxico, sintáctico y semántico, para la construcción de compiladores o intérpretes, de un lenguaje de alto nivel. <p>III. Contenido</p> <ol style="list-style-type: none"> 1. Traducción dirigida por la sintaxis <ol style="list-style-type: none"> 1.1. Definiciones dirigidas por la sintaxis <ol style="list-style-type: none"> 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 <ol style="list-style-type: none"> 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 <ol style="list-style-type: none"> 1.3.1. Construcción de árboles de análisis sintáctico 1.3.2. La estructura de tipos 1.4. Esquemas de traducción orientados por la sintaxis <ol style="list-style-type: none"> 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 		

	<ul style="list-style-type: none">1.4.4. Eliminación de la recursividad por la izquierda de los esquemas de traducción1.4.5. Esquemas de traducción orientados a la sintaxis para definiciones con atributos heredados por la izquierda1.5. Implementación de definiciones dirigidas por la sintaxis con atributos heredados por la izquierda<ul style="list-style-type: none">1.5.1. Traducción durante el análisis sintáctico de descenso recursivo1.5.2. Generación de código al instante1.5.3. Las definiciones dirigidas por la sintaxis con atributos heredados por la izquierda y análisis sintáctico LL1.5.4. Análisis sintáctico ascendente de las definiciones dirigidas por la sintaxis con atributos heredados por la izquierda
2.	<ul style="list-style-type: none">2. Generación de código intermedio<ul style="list-style-type: none">2.1. Variantes de los árboles sintácticos<ul style="list-style-type: none">2.1.1. Grafo dirigido acíclico para expresiones2.1.2. Método número de valor para GDA2.2. Código de tres direcciones<ul style="list-style-type: none">2.2.1. Direcciones e instrucciones2.2.2. Cuádruplos2.2.3. Tripletas2.2.4. Forma de asignación individual estática2.3. Tipos y declaraciones<ul style="list-style-type: none">2.3.1. Expresiones de tipos y equivalencias2.3.2. Declaraciones y distribución de almacenamiento2.3.3. Secuencias de las declaraciones2.3.4. Campos en registros2.4. Traducción de expresiones<ul style="list-style-type: none">2.4.1. Operaciones dentro de expresiones2.4.2. Traducción incremental2.4.3. Direccionamiento de los elementos de un arreglo2.4.4. Traducción de referencias a arreglos2.5. Comprobación de tipos<ul style="list-style-type: none">2.5.1. Reglas para la comprobación de tipos2.5.2. Conversiones de tipos2.5.3. Sobrecarga de funciones y operadores2.5.4. Inferencia de tipos y funciones polimórficas2.5.5. Un algoritmo para la unificación2.6. Flujo de control<ul style="list-style-type: none">2.6.1. Expresiones booleanas2.6.2. Código de corto circuito2.6.3. Instrucciones de flujo de control2.6.4. Traducción del flujo de control de las expresiones booleanas2.6.5. Evitar goto redundantes2.6.6. Valores booleanos y código de salto2.7. Parcheo de retroceso<ul style="list-style-type: none">2.7.1. Generación de código de una pasada2.7.2. Técnica de retroceso2.7.3. Instrucciones de flujo de control2.8. Instrucciones switch<ul style="list-style-type: none">2.8.1. Traducciones de switch2.8.2. Traducción orientada por la sintaxis de switch2.9. Código intermedio para procedimientos
3.	<ul style="list-style-type: none">3. Optimización de código<ul style="list-style-type: none">3.1. Optimización de bloques básicos<ul style="list-style-type: none">3.1.1. Representación GDA3.1.2. Búsqueda de subexpresiones locales comunes3.1.3. Eliminación de código muerto3.1.4. Uso de identidades algebraicas3.1.5. Representación de referencias a arreglos3.1.6. Asignación de apuntadores y llamadas a procedimientos3.1.7. Reensamblado de bloques básicos3.2. Optimización de mirilla<ul style="list-style-type: none">3.2.1. Eliminación de instrucciones redundantes3.2.2. Eliminación de código inalcanzable3.2.3. Optimizaciones de flujo de control

3.2.4. Simplificación algebraica y reducción por fuerza

IV. Metodología:

Clase Magistral para explicación de teoría.
Resolución de problemas y auto estudio
realización de proyectos y actividades de laboratorio

V. Evaluación:

32 puntos para laboratorio, correspondientes a dos proyectos.
68 puntos de la parte teórica, que incluye 35 puntos de tres parciales, 8 puntos de exámenes cortos y 25 puntos del examen final.

Para aprobar el curso es necesario obtener como mínimo 19.52 puntos de laboratorio y como mínimo 36 puntos de zona.

CALENDARIO DE EXÁMENES

Primer Examen Parcial	20 de agosto UNIDAD 1. Traducción dirigida por la sintaxis
Segundo Examen Parcial	24 de septiembre UNIDAD 2. Generación de código intermedio
Tercer Examen Parcial	24 de octubre UNIDAD 2. Código intermedio para procedimientos UNIDAD 3. Optimización de código
Examen Final	De acuerdo al calendario oficial TODAS LAS UNIDADES

Observaciones:

Direcciones de correo electrónico para consultas:
Ing. Bayron López: blopezw@yahoo.com
Ing. Edgar Sabán : edgarsaban@gmail.com
Ing. Luis Espino: usac.sistemas@gmail.com

7	Bibliografía	Libro de Texto: Compiladores. Principios, Técnicas y Herramientas Aho, Sethi y Ullmam. PEARSON ADDISON-WESLEY, 2008, segunda edición.
8	Catedráticos y tutores académicos	Catedráticos:: Ing. Bayron López, Ing. Edgar Sabán, Ing. Luis Espino Tutores: Alex Yovani Jerónimo Tomás, Aldo Rigoberto Henández Ávila