DATOS DE IDENTIFICACIÓN DEL C RSO					
DEPARTAMENTO:	CIENCIAS COMPUTACIONALES				
ACADEMIA A LA QUE PERTENECE:	SOFTWARE DE SISTEMAS				
NOMBRE DE LA MATERIA:	COMPILADORES				
CLAVE DE LA MATERIA:	CC317				
CARÁCTER DEL CURSO:	ESPECIALIZANTE				
TIPO DE CURSO:	CURSO				
No. DE CRÉDITOS:	11				
No. DE HORAS TOTALES:	80	Presencial	68	No presencial	12
ANTECEDENTES:	TEORIA DE LA COMPUTACIÓN				
CONSECUENTES:					
CARRERAS EN QUE SE IMPARTE:	ING. COMPUTACIÓN				
FECHA DE ULTIMA REVISIÓN:	20 DE ENERO 2009				

PROPÓSITO GENERAL

Conocer el problema al que se enfrenta un compilador. Aprender que dada la complejidad del problema es necesario separarlo en varios módulos, donde cada módulo está encargado de una tarea específica, para después entregar un resultado al siguiente módulo.
Comprender los algoritmos que se utilizan para resolver cada una de las fases del compilador.
Dominar completamente el proceso para convertir un lenguaje de alto nivel a un lenguaje de bajo nivel.

OBJETIVO TERMINAL

Comprender las técnicas utilizadas para el diseño de un compilador. Aprender cómo funcionan y como implementar cada una de las etapas del compilador: el análisis léxico, análisis sintáctico, análisis semántico y la generación de código. Además de saber cómo interactúan cada una de las etapas mencionadas.

CONOCIMIENTOS PREVIOS

Teoría de la computación, programación estructurada, estructura de datos, estructura de archivos, programación orientada a objetos, programación de sistemas.

HABILIDADES Y DESTREZAS A DESARROLLAR

El alumno será capaz de diseñar nuevos lenguajes, así como programas que los reconozcan. El alumno tendrá la adquirirá la habilidad suficiente para escribir programas en ensamblador, para después poder realizar la generación de manera automática. El alumno será capaz de trabajar con

las diferentes estructuras de datos del compilador: símbolos, árbol sintáctico, tabla de símbolos.

Las técnicas aprendidas en el curso podrán ser aplicadas para resolver problemas de concordancia de cadenas y reconocimiento de patrones, traducción de lenguajes. Tendrá la habilidad de diseñar e implementar gramáticas independientes de contexto y definiciones dirigidas por la sintaxis que le permitan resolver problemas de procesamiento de lenguajes.

ACTITUDES Y VALORES A FOMENTAR
Autoformación didáctica, respeto, puntualidad, disciplina, trabajo en equipo, ética profesional

METODOLOGÍA DE ENSEÑANZA APRENDIZAJE								
Método	Método tradicional de exposición	Método Audiovisual	Aula Interactiva	Multimedia	Desarrollo de proyecto	Dinámicas	Estudio de casos	Otros (Especificar)
%	60				10	30		

CONTENIDO TEMÁTICO

MODULO 1. Introducción				2 HRS
Conocer el problema que debe resolver un compilador, su complejidad así como las fases que lo conforman.				
1.1	Introducción al compilador			0.5 HRS
	Conocer los conceptos básicos las tareas de un compilador así como sus aplicaciones.			
	1.1.1	Descripción del compilador		
	1.1.2	Procesadores de lenguajes		
		Comprender que es el procesamiento de lenguajes.		
	1.1.3	Construcción de un compilador		
		Conocer el problema al que se enfrenta un compilador,		
	1.1.4	Aplicaciones		
		Identificar las posibles aplicaciones de la teoría de compiladores.		
1.2	Estructura del compilador			1.5 HRS
	Aprender cuales son cada una de las fases del compilador, así como las tareas que deben realizar.			
	1.2.1	Análisis léxico		
		Comprender como funciona el análisis léxico y como será utilizado en la fase del análisis sintáctico.		
	1.2.2	Análisis sintáctico		
		Comprender como funciona el análisis sintáctico, las tareas que debe realizar.		
	1.2.3	Análisis semántico		
		Comprender como funciona el análisis sem ntico, las tareas que debe realizar.		
	1.2.4	Generación de código		
		Comprender como se puede generar código		

MODULO 2. Análisis léxico	4 HRS
Conocer el funcionamiento del análisis léxico, el tipo de gramáticas que debe reconocer y la forma en la que se puede implementar.	

2.1	Descripción del análisis léxico			1 HRS
	OBJETIVO DEL TEMA			
	2.1.1	Función del analizador léxico		
		Determinar el objetivo del analizador léxico.		
	2.1.2	Especificación de los componentes léxicos mediante gramáticas y expresiones regulares		
		Aprender cómo pueden ser descritos los elementos léxicos formalmente		
	2.1.3	Determinación de los componentes léxicos mediante autómatas finitos		
		Aprender como reconocer componentes léxicos utilizando autómatas finitos.		
2.2	Lenguajes formales			1 HRS
	Descripción de lenguajes utilizando los conceptos de teoría de la computación.			
	2.2.1	Definición		
		Conocer cual es la definición de un lenguaje formal		
	2.2.2	Conceptos básicos		
		Aprender los conceptos principales utilizados para describir lenguajes formales.		
	2.2.3	Lenguajes regulares		
		Aprender a identificar los lenguajes regulares		
	2.2.4	Gramáticas regulares		
		Aprender a definir gramáticas regulares para reconocer lenguajes regulares		
	2.2.5	Expresiones regulares		
		Aprender a describir los patrones de los elementos léxicos utilizando expresiones.		
2.3	Autómatas finitos			2 HRS
	Aprender a definir maquinas capaces de reconocer lenguajes regulares			
	2.3.1	Definición formal		
		OBJETIVO DEL SUBTEMA		
	2.3.2	Representación gráfica (diagrama de transición)		
		Aprender a construir el digrama de transición de un autómata finito.		
	2.3.3	Tipos de autómatas		
		Conocer las diferencias entre autómata finito determinista y no determinista.		

	2.3.4	Tabla de transición		
		Aprender a construir la tabla de transición de un autómata finito.		
	2.3.5	Conversión de una expresión regular a un autómata finito determinista		
		Comprender como puede construirse un autómata finito determinista que acepte el mismo lenguaje que una expresión regular.		
	2.3.6	Implementación		
		Aprender a realizar la implementación del analizador léxico.		
MODULO 3. Análisis sintáctico				18 HRS
Comprender como funciona el analizador sintáctico, las gramáticas que debe aceptar y como implementarlo.				
3.1	Introducción			0.5 HRS
	OBJETIVO DEL TEMA			
	3.1.1	Función del analizador sintáctico		
		Conocer el objetivo principal del analizador sintáctico.		
	3.1.2	Representación de gramáticas		
		Conocer como pueden ser representadas las gramáticas que utilizará el analizador sintáctico.		
3.2	Gramáticas libres de contexto			1.5 HRS
	Conocer cuáles son las características de una gramatica libre de contexto para así poder identificarlas.			
	3.2.1	Definición formal		
		Conocer como se describe formalmente una gramática libre de contexto.		
	3.2.2	Derivaciones		
		Comprender como se puede generar una palabra del lenguaje mediante derivaciones.		
	3.2.3	Árbol de análisis sintáctico		
		Aprender a construir arboles de análisis sintáctico		
	3.2.4	Ambigüedad		
		Aprender a detectar si una gramática es ambigua.		
	3.2.5	Eliminación de la recursividad por la izquierda		
		Aprender a quitar la recursividad por la izquierda de una gramática.		
	3.2.6	Factorización por la izquierda		

		Aprender a factorizar gramáticas.		
3.3	Análisis sintáctico descendente mediante método recursivo			2 HRS
	Aprender a utilizar y construir analizadores descendentes			
	3.3.1	Descripción		
		Conocer las características más importantes del método.		
	3.3.2	Gramáticas LL		
		Ser capaz de identificar las características que tienen las gramáticas que son LL.		
	3.3.3	Implementación del método		
		Realizar la implementación del método		
	3.3.4	Uso de producciones		
		Aprender a utilizar reglas durante la implementación del algoritmo.		
3.4	Análisis sintáctico descendente mediante método no recursivo			6 HRS
	Aprender a utilizar y construir analizadores descendentes			
	3.4.1	Descripción		
		Comprender de forma general el método		
	3.4.2	Gramáticas LL(1)		
		Comprender cuales son las gramáticas LL(1)		
	3.4.3	Algoritmo de análisis sintáctico LL(1)		
		Aprender a utilizar el algoritmo de análisis sintáctico LL(1)		
	3.4.4	Calculo del conjunto primero		
		Aprender a calcular el conjunto primero los no terminales de una gramática.		
	3.4.5	Calculo del conjunto siguiente		
		Aprender a calcular el conjunto siguiente los no terminales de una gramática.		
	3.4.6	Construcción de tablas de análisis sintáctico predictivo no recursivo		
		Comprender como se realiza la construcción de tablas para este método.		
3.5	Análisis sintáctico ascendente			6 HRS
	Aprender a utilizar y construir analizadores ascendentes			
	3.5.1	Introducción		
		Comprender de forma general el método		
	3.5.2	Gramáticas LR(1)		

		Comprender cuales son las gramáticas LR(1)		
	3.5.3	Algoritmo de análisis sintáctico LR(1)		
		Aprender a utilizar y construir analizadores ascendentes		
	3.3.4	Conjunto cerradura		
		Conocer como se construye el conjunto cerradu a		
	3.3.5	Función de transición		
		Aprender a utilizar función de transición		
	3.3.6	Construcción del conjunto de elementos		
		Ser capaz de construir el conjunto de elementos		
	3.3.6	Construcción de tablas de análisis sintáctico ascendente		
		Tener la capacidad de diseñar tablas de análisis sintáctico descendente.		
3.6	Gramáticas ambiguas			2 HRS
	Aprender a analizar gramáticas ambiguas.			
	3.6.1	Precedencia y asociatividad		
		Comprender las ideas de precedencia y asociatividad		
	3.6.2	La ambigüedad del else		
		Conocer el caso del else ambiguo		
	3.6.3	Construcción de analizadores LR para gramáticas ambiguas		
		Comprender como se realiza la construcción de tablas para este método		
3.7	Definiciones dirigidas por sintaxis			4 HRS
	Conocer cómo se puede agregar atributos a las gramáticas.			
	3.7.1	Gramáticas con atributos		
		Aprender a utilizar atributos de una gramática.		
	3.7.2	Creación de árboles sintácticos utilizando programación orientada a objetos		
		Aprender a construir árboles n-arios utilizando programación orientada a objetos.		
	3.7.3	Creación de árboles sintácticos durante el análisis sintáctico		
		Comprender como se construyen árboles sintácticos desde la fase de análisis sintáctico.		

MODULO 4. Traducción mediante análisis sintáctico — 8 HRS

Aprender a utilizar los métodos de análisis sintáctico para realizar la traducción de lenguajes.

4.1	Gramáticas con atributos			1 HRS
	Aprender como se puede aumentar la capacidad expresiva una gramática, mediante el uso de atributos.			
	4.1.1	Atributos sintetizados		
		Comprender como funcionan los atributos sintetizados.		
	4.1.2	Atributos heredados		
		Comprender como funcionan los atributos heredados.		
4.2	Construcción de Árboles sintácticos			7 HRS
	Aprender a construir arboles sintácticos utilizando los analizadores sintácticos descendentes y ascendentes.			
	4.1.1	Arboles sintácticos		
		Conocer cómo se puede representar un árbol sintáctico utilizando programación orientada a objetos.		
	4.1.2	Construcción de árboles sintácticos para expresiones		
		Implementar árboles sintácticos para expresiones.		
	4.1.3	Construcción de árboles sintácticos utilizando analizador sintáctico descendente		
		Aprender a construir árboles de manera automática durante el análisis sintáctico descendente.		
	4.1.4	Construcción de árboles sintácticos utilizando analizador sintáctico ascendente		
		Aprender a construir árboles de manera automática durante el análisis sintáctico ascendente.		
MODULO 5. Análisis semántico				4 HRS
Aprender a realizar la verificación de tipos utilizando los árboles sintácticos y la tabla de símbolos.				
5.1	Comprobación de tipos			2 HRS
	Aprender a definir formalmente las expresiones validas del lenguaje. Comprender como realizar la comprobación de expresiones del lenguaje			
	5.1.1	Expresiones de tipos		
		Aprender a definir formalmente las expresiones válidas del lenguaje.		
	5.1.2	Sistema de tipos		
		Implementar a definir las expresiones de tipos aceptadas por el lenguaje para construir el sistema de tipos.		
	5.1.3	Conversión de tipos		
		Aprender a definir los tipos que pueden ser convertidos de		

		manera automática.		
	5.1.4	Comprobación estática y dinámica		
		Aprender las diferencias entre los tipos de comprobación.		
	5.1.5	Comprobación de tipos en expresiones		
		Aprender a definir expresiones de tipos para expresiones del lenguaje.		
	5.1.6	Comprobación de tipos en proposiciones		
		Aprender a definir expresiones de ti os para proposiciones del lenguaje.		
5.2	Tabla de símbolos			2 HRS
	Conocer como se implementa el almacenamiento de los tipos de datos de las variables, durante la compilación.			
	5.2.1	Estructura de la tabla de símbolos		
		Conocer la estructura que se utiliza para implementar una tabla de símbolos.		
	5.2.2	Declaraciones de variables, procedimientos y funciones		
		Conocer como se almacenan las declaraciones de variables, procedimientos y funciones en la tabla de símbolos.		
	5.2.3	Reglas de ámbito y estructuras de bloques		
		Comprender los conceptos de ámbito y estructuras de bloques.		
MODULO 6. Generación de código intermedio				6 HRS
Aprender a traducir lenguajes a código intermedio.				
6.1	Código de tres direcciones			6 HRS
	Conocer la sintaxis del código de tres direcciones.			
	6.1.1	Traducción de expresiones		
		Aprender a traducir expresiones de un lenguaje de alto nivel a código intermedio.		
	6.1.2	Traducción de referencias a arreglos		
		Aprender a traducir referencias a arreglos de un lenguaje de alto nivel a código intermedio.		
	6.1.3	Traducción de expresiones lógicas		
		Aprender a traducir expresiones lógicas de un lenguaje de alto nivel a código intermedio.		
	6.1.4	Código de corto circuito		
		Aprender implementar código de corto circuito en el código intermedio.		

	6.1.5	Instrucciones para controlar el flujo del programa		
		Aprender a controlar el flujo de programas en código intermedio		
	6.1.6	Procedimientos		
		Aprender a definir procedimientos en código intermedio.		
MODULO 7. Generación de código objeto				10 HRS
Aprender a implementar programas en código objeto, para después aprender a traducir código de un lenguaje de alto nivel a código objeto.				
7.1	Introducción			0.5 HRS
	Conocer los conceptos básicos			
	7.1.1	Maquina objeto		
		Conocer los conceptos básicos de la maquina objeto.		
	7.1.2	Instrucciones de la maquina objeto		
		Conocer el conjunto de instrucciones disponibles de la maquina objeto.		
	7.1.3	Modos de direccionamiento		
		Comprender los métodos de direccionamiento existentes.		
7.2	Código ensamblador			7.5 HRS
	Aprender a escribir programas en el lenguaje ensamblador			
	7.2.1	Asignación de valores		
		Aprender a asignar valores a variables		
	7.2.2	Expresiones aritméticas		
		Aprender a realizar expresiones aritméticas		
	7.2.3	Control de flujo		
		Aprender a utilizar las instrucciones para controlar el flujo del programa		
	7.2.4	Procedimientos y funciones		
		Aprender a definir y utilizar procedimientos y funciones.		
7.3	Generación de código objeto a partir de árboles sintácticos			2 HRS
	Aprender a generar código objeto de manera automática utilizando árboles sintácticos.			
MODULO 8. Optimización				8 HRS
Aprender a realizar optimizaciones al código generado por el compila or.				
8.1	Optimización de mirilla			4 HRS
	Conocer las optimizaciones de mirilla.			

	8.1.1	Eliminación de instrucciones redundantes		
		Aprender a eliminar instrucciones que son redundantes en el código generado.		
	8.1.2	Eliminación de código inalcanzable		
		Aprender a eliminar código inalcanzable en el código generado.		
	8.1.3	Optimizaciones de flujo de control		
		Aprender a optimizar las instrucciones de control de flujo		
	8.1.4	Simplificación algebraica y reducción por fuerza		
		Aprender a reducir expresiones algebraicas		
8.2	Generación de código optimo para expresiones			4 HRS
	Aprender a generar código para expresiones que utilice de manera optima los registros disponibles de la maquina objeto.			
	8.1.1	Números de Ershov		
		Conocer cuáles son los números Ershov y para que se pueden utilizar.		
	8.1.2	Generación de código a partir de árboles de expresión etiquetados		
		Aprender a generar código para expresiones a partir de árboles etiquetados		
	8.1.3	Generación de código a partir de árboles de expresión etiquetados con un número insuficiente de registros		
		Aprender a generar código para expresiones optimizado con un número insuficiente de registros.		

CRITERIOS DE EVALUACIÓN

40% Exámenes departamentales
40% Exámenes parciales
10% Tareas
10% Proyecto

BIBLIOGRAFÍA

BÁSICA

TITULO	AUTOR	EDITORIAL	AÑO DE EDICIÓN	% DE COBERTURA DEL CURSO

Compiladores Principios técnicas y herramientas Segunda Edición	Alfred V. Aho Monica S. Lam Ravi Sethi Jeffrey D. Ullman	Pearson	2008	90%
Compiladores Principios técnicas y herramientas Primera Edición	Alfred V. Aho Ravi Sethi Jeffrey D. Ullman	Pearson	1990	90%

COMPLEMENTARIA

TITULO	AUTOR	EDITORIAL	AÑO DE EDICIÓN	% DE COBERTURA DEL CURSO
Construcción de compiladores	Kenneth C. Louden	Thomson	2004	70%
Teoría de la computación	J. Glenn Brookshear	Addison Wesley	1995	30%
Teoría de autómatas y lenguajes formales	Dean Kelly	Prentice Hall	1995	20%
Estructura de datos y Algoritmos	Aho, A. J. Hopcroft y J.	Addison Wesley		20%

REVISIÓN REALIZADA POR:	
NOMBRE DEL PROFESOR	FIRMA
Dr. Carlos Alberto López Franco	

Vo.Bo. Presidente de Academia	Vo.Bo. Jefe del Departamento
	Dr. Carlos Alberto López Franco

miércoles, 28 de enero de 2009