UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

NOMBRE DEL CURSO: Organización de Lenguajes y Compiladores 2

CODIGO:	781	**CREDITOS:**	5
ESCUELA:	Ciencias y Sistemas	**AREA A LA QUE PERTENECE:**	Ciencias de la Computación
PRE REQUISITO:	777 - Organización de Lenguajes y Compiladores 1 772 - Estructuras de datos	**POST REQUISITO:**	281 - Sistemas Operativos 1
CATEGORIA:	Obligatorio	**SEMESTRE:**	Segundo semestre 2019
EDIFICIO:	T1, T7, T3	**SECCIÓN:**	A, B+,B-
HORAS POR SEMANA DEL CURSO:	4 horas	**HORAS POR SEMANA DEL LABORATORIO:**	2 horas
DÍAS QUE SE IMPARTE EL CURSO SECCION A:	Lunes y sábado	**HORARIO DEL CURSO SECCION A:**	7:10 AM – 8:50 AM 12:10 PM – 13:50 PM
DÍAS QUE SE IMPARTE EL CURSO SECCION B+, B-:	Lunes y viernes	**HORARIO DEL CURSO SECCION B+,B-:**	7:10 AM – 8:50 AM
CATEDRÁTICOS:		**TUTORES ACADEMICOS:**	
Sección A:	Ing. Byron López	**Sección A:**	Luis Lizama
Sección B+:	Ing. Edgar Sabán	**Sección B+:**	Julio Arango
Sección B-	Ing. Erick Navarro	**Sección B-:**	Rainman Sián

DESCRIPCIÓN DEL LABORATORIO

Este laboratorio es la continuación de los laboratorios de Lenguajes Formales y de Programación y de Organización de Lenguajes y Compiladores 1, en conjunto con los conocimientos adquiridos previamente introduce a los estudiantes al diseño e implementación de compiladores de lenguajes de programación.

El estudiante podrá poner en práctica los conceptos que se desarrollen en la clase magistral, como la traducción dirigida por la sintaxis, generación y optimización de código intermedio; que son útiles en muchos otros contextos más allá de compiladores, como ingeniería de software y seguridad.

Quizás el resultado más útil del curso es que los estudiantes comprendan profundamente las capacidades y limitaciones de los compiladores modernos y cómo pueden ser utilizados eficazmente. Este conocimiento no solo es importante para aspirantes a diseñadores de lenguajes, sino también para depurar y optimizar casi cualquier aplicación.

OBJETIVO GENERAL

Poner en práctica los conocimientos teóricos adquiridos en la clase magistral para la implementación de un compilador.

OBJETIVOS ESPECÍFICOS

1. Que el estudiante sea capaz de desarrollar aplicaciones con la habilidad de leer una entrada y producir una acción de salida.
2. Que el estudiante comprenda las aplicaciones de un compilador en el área de ciencias de la computación.
3. Que el estudiante comprenda la relación entre las prácticas de programación mediante el uso de patrones de diseño, estructuras de datos adecuadas y el impacto de la mismas en el desempeño de un programa.
4. Que el estudiante comprenda los conceptos de traducción dirigida por la sintaxis y su implementación para la generación de código intermedio.
5. Que el estudiante aplique la teoría de entornos locales y estructuras en tiempo ejecución, para elaborar un compilador capaz de ejecutar el código intermedio generado.
6. Que el estudiante sea capaz de aplicar la teoría de optimización de código intermedio tanto en el área de compiladores como en el desarrollo de aplicaciones de software en general.

COMPETENCIAS ESPECÍFICAS (CE) DE LAS ACCIONES FORMATIVAS DE LA DISCIPLINA

1. Reconoce, entiende y es capaz de aplicar los conceptos de lenguajes formales y de programación, como la definición de lenguajes formales, gramáticas y su clasificación, análisis léxico y sintáctico descendente (predictivo recursivo).
2. Interpreta, analiza y aplica conceptos y procedimientos de Organización de Lenguajes y Compiladores 1 para la solución de problemas de análisis ascendente, traducción e interpretación de lenguajes definidos por gramáticas libres de contexto.
3. Utiliza software para la generación de analizadores léxicos y sintácticos.
4. Planifica y desarrolla actividades de auto aprendizaje para la solución de problemas por medio de la implementación de trabajos extra-aula realizados de manera individual y/o grupal colaborativo.
5. Razona crítica y lógicamente sobre los procesos y resultados para verificar su validez por medio de la comparación con el conocimiento y la experiencia.

METODOLOGÍA

- Se desarrollarán ejercicios prácticos, que serán la implementación de los ejercicios resueltos en la clase magistral, se mostrará cómo realizar un front-end para el desarrollo de un compilador utilizando como base el apéndice A del libro de texto.
- Se realizarán evaluaciones presenciales prácticas en los periodos de laboratorio, lo cual permitirá evaluar el aprendizaje y la aplicación de los conceptos adquiridos.
- Además de la bibliografía recomendada, se proporcionará al alumno material complementario que le ayude a conocer distintas técnicas en la elaboración de compiladores.
- Se realizarán dos rproyectos para poder evaluar los conceptos adquiridos en clase, tomando en cuenta que pueden incluirse temas de cursos pre-requisito.

EVALUACIÓN DEL RENDIMIENTO ACADÉMICO

El laboratorio tiene una ponderación de 34 puntos distribuidos de la siguiente manera:

Actividad	**Ponderación**	**Porcentaje**
Primer proyecto	15.3	45%
Segundo proyecto	18.7	55%
Total	**34**	**100%**

Observaciones:

- Para aprobar el laboratorio se debe tener una nota final igual o mayor al 61% de los puntos, es decir 20.74 puntos de 34.
- Cada proyecto tiene requerimientos mínimos que se deben cumplir para tener derecho a calificación.
- Copias parciales o totales de los proyectos tendrán una nota de 0 puntos y los responsables serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas.
- Se deben enviar los archivos entregables en las fechas establecidas y por el medio indicado para tener derecho a calificación.
- El tiempo estimado de calificación para cada proyecto será de dos horas por estudiante.

CONTENIDO

Para su mayor comprensión el contenido del laboratorio se dividirá en 3 secciones,

Unidad 1: Repaso de cursos anteriores

- **Sesión 1 – Semana 22 - 27 de Julio**
  - Procesadores de lenguaje
    - Compilador
    - Interprete
    - Traductor
    - Compilador hibrido
    - Comparación entre un intérprete y un compilador
    - Sistema de procesamiento de lenguaje
  - La estructura de un compilador
    - Fases de un compilador
      - Fase de análisis
        - Analizador léxico
        - Analizador sintáctico
        - Analizador semántico
        - Generador de código intermedio
      - Fase de síntesis
        - Generador de código intermedio
        - Optimizador de código
        - Generador de código
    - Administración de la tabla de símbolos
  - Evaluación presencial
    - Parte teórica
    - Parte práctica

- **Sesión 2 – Semana 29 de Julio – 03 de agosto**
  - Mecanismos para el paso de parámetros
  - Llamada por valor

- Llamada por referencia
- Llamada por nombre
- Definición de sintaxis
  - Árboles de análisis sintáctico
  - Ambigüedad
  - Asociatividad de los operadores
  - Precedencia de los operadores
  - Análisis sintáctico por precedencia de operadores
- Conceptos básicos
  - Atributo
  - Definición dirigida por la sintaxis
  - Atributos sintetizados
  - Atributos heredados
- Evaluación presencial
  - Parte teórica
  - Parte práctica (implementación con herramienta definida)

Unidad 2: Traducción dirigida por la sintaxis

- **Sesión 3 – Semana 05 – 10 de agosto**
  - Esquemas de traducción orientados por la sintaxis
    - Con acciones dentro de las producciones
      - Análisis ascendente
      - Análisis descendente
    - Eliminación de la recursividad por la izquierda de esquemas de traducción
    - Con atributos heredados por la izquierda
  - Aplicaciones de la traducción dirigida por la sintaxis
    - Expresiones aritméticas con un esquema de traducción sencillo
    - Construcción de árboles de análisis sintáctico
      - Expresiones
        - Aritméticas
        - Relacionales
        - Lógicas
    - Recorrido de un árbol de análisis sintáctico (interprete)
      - Obtener el valor implícito de una expresión
      - Validación de errores
        - División por cero
        - Rango excedido
      - Comprobación de tipos
      - Conversión de tipos
  - Evaluación presencial
    - Parte teórica
    - Parte práctica (implementación con herramienta definida)

- **Semana 12 – 17 de agosto**
  - ***Resolución Primer Parcial***
  - Evaluación presencial
    - Parte teórica

- Parte práctica (implementación con herramienta definida)

❖ **Sesión 4 – Semana 19 – 24 de agosto**
- ➢ Construcción de árboles de análisis sintáctico
  - Sentencias de selección
    - Bifurcación
    - N-furcación
  - Sentencias cíclicas
  - Sentencias de transferencia
    - Sentencia break
    - Sentencia continue
    - Sentencia return
- ➢ Recorrido de un árbol de análisis sintáctico
  - Validación de errores
    - Condición de tipo booleano
    - Lógica de un ciclo y sus sentencias de transferencia
  - Comprobación de tipos
    - Comprobación estática
    - Comprobación dinámica
  - Estructuras y técnicas auxiliares para manejo de sentencias de transferencia
    - Solución con contadores
    - Solución con pila
- ➢ Evaluación presencial
  - Parte teórica
  - Parte práctica (implementación con herramienta definida)

❖ **Sesión 5 –Semana 26 – 31 de agosto**
- ➢ Construcción de árboles de análisis sintáctico
  - Métodos y funciones
    - Retorno
    - Tipo
    - Parámetros
- ➢ Recorrido de un árbol de análisis sintáctico
  - Validaciones semánticas para métodos y funciones
    - Manejo de llamadas recursivas
    - Entorno de una llamada a un método
    - Tipos de retornos
    - Retorno obligatorio para funciones
  - Instanciación
  - Acceso a atributos y métodos
    - Entorno de un objeto
    - Lista de n accesos
      - ♦ Requisitos
      - ♦ Validaciones de semántica
- ➢ Evaluación presencial
  - Parte teórica
  - Parte práctica (implementación con herramienta definida)

- **Sesión 6 – Semana 02 – 07 de septiembre**
  - Construcción de árboles de análisis sintáctico
    - Arreglos
      - Declaración e inicialización
      - Representación en forma de árbol N-ario
    - Clases y objetos
      - Definición de clases
      - Instanciación
      - Acceso a atributos y métodos
  - Recorrido de un árbol de análisis sintáctico
    - Acceso a arreglos
      - Recorrido recursivo para obtener nodo especifico
    - Validación de errores en arreglos
      - Numero de dimensiones
      - Fuera de limite
      - Tipos incompatibles
      - Índices de tipo entero
    - Definición consistente
  - Evaluación presencial
    - Parte teórica
    - Parte práctica (implementación con herramienta definida)

Unidad 3: Generación de código intermedio

- **Semana 09 – 13 de septiembre**
  - ***Resolución Segundo Parcial***
  - Evaluación presencial
    - Parte teórica
    - Parte práctica (implementación con herramienta definida)

- **Sesión 7 – Semanas 09 – 20 de septiembre**
  - Modelo básico orientado a objetos para la generación de código intermedio
    - ¿Qué es lo mínimo que se necesita para generar código intermedio?
      - Árbol de análisis sintáctico
      - Manejar los punteros principales
        - Puntero del Stack (P)
        - Puntero del Heap (H)
    - Administración de la tabla de símbolos para un generador de código intermedio
      - Elementos que deben ser almacenados en una tabla de símbolos
        - Nombres de procedimientos y funciones
        - Temporales
        - Etiquetas
      - Elementos esenciales de un símbolo
        - Tipo
        - Identificador
        - Puntero a montículo (para arreglos, estructuras y objetos)
        - Tipo de paso de parámetros (referencia o valor)
        - Número y tipo para los parámetros
      - Tablas de símbolos encadenadas por alcance
        - La regla del bloque anidado más cercano

- Definición de entorno
- Implementación de un entorno
  - Crear una nueva tabla de símbolos
  - Agregar una entrada en la tabla actual
  - Obtener una entrada para un identificador

❖ **Sesión 8 – Semana 23 – 28 de septiembre**

- Construcción y recorrido de árbol para generación de código de tres direcciones
  - Traducción de expresiones
    - Aritméticas
    - Relacionales
    - Lógicas
      - Convencional
      - Corto circuito
  - Traducción de sentencias de selección
    - Bifurcación
    - N-furcación
  - Traducción de sentencias cíclicas
    - Iteradoras
    - Para colecciones
  - Traducción de sentencias de transferencia
    - Sentencia break
      - Con etiqueta
      - Sin etiqueta
    - Sentencia continue
      - Con etiqueta
      - Sin etiqueta
    - Salto incondicional goto
  - **Display**
- Evaluación presencial
  - Parte teórica
  - Parte práctica (implementación con herramienta definida)

Unidad 4: Entornos locales y estructuras en tiempo de ejecución

❖ **Sesión 9 – Semana 3 de septiembre – 05 de octubre**

- Organización del almacenamiento
  - Asignación de almacenamiento estática y dinámica
  - Asignación de espacio en pila
    - Secuencia de llamadas
    - Datos de longitud variable en la pila
    - Variables globales
    - Métodos y funciones
      - Resolución de la recursividad
    - Variables locales
    - Parámetros
      - Por valor
      - Por referencia
- Evaluación presencial
  - Parte teórica

- Parte práctica (implementación con herramienta definida)

❖ **Sesión 10 – Semana 07 – 12 de octubre**
- Organización del almacenamiento
  - Administración de la pila de ejecución (Stack)
    - Principios generales útiles
      - Los valores de comunicación van al inicio
      - Los elementos de tamaño desconocido van al final
      - El apuntador P va al tope de la pila
    - Acceso a los datos no locales en la pila
    - Acceso a los datos sin procedimientos anidados
    - Problemas con los procedimientos anidados
    - Enlace de acceso
    - Manipulación de enlaces de acceso
    - Enlaces de acceso para los parámetros de procedimientos
    - Estructura de datos Display
- Evaluación presencial
  - Parte teórica
  - Parte práctica (implementación con herramienta definida)

❖ **Sesión 11 – semana del 14 – 19 de octubre**
- Organización del almacenamiento
  - Administración del montículo
    - Definición, objetivos y funciones del montículo
    - El administrador de memoria
      - Asignación
      - Des asignación
    - La jerarquía de memoria de una computadora
    - Localidad en los programas
    - Manejo de arreglos de una o más dimensiones
      - Información relevante asociada a un arreglo y su almacenamiento en el montículo
        - Referencias de los elementos de un arreglo en el montículo
        - Reservación de memoria
        - Acceso a un arreglo
        - Otras operaciones con arreglos
      - Organización lexicográfica
        - por filas
        - por columnas

- Evaluación presencial
  - Parte teórica
  - Parte práctica (implementación con herramienta definida)

❖ **Sesión 12 – Semana 21 – 26 de octubre**
- Organización del almacenamiento
    - Manejo de cadenas
      - Referencias de cadenas en el montículo
      - Manejo de cadenas a bajo nivel
        - Reservación de memoria

        - Carácter nulo
        - Concatenación
        - Conversión de entero a cadena
        - Conversión de cadena a entero
        - Otras operaciones con cadenas
    - Administración del montículo
        - Manejo de clases y objetos
            - Definición de un objeto
            - Método constructor
            - Atributos de un objeto
            - Referencias al montículo de los atributos de un objeto
    - Introducción a la recolección de basura
        - Seguridad en los tipos
        - Métrica de rendimiento
- Evaluación presencial
    - Parte teórica
    - Parte práctica (implementación con herramienta definida)

Unidad 5: Errores en tiempo de ejecución y optimización de código intermedio

- **Semanas 28 de octubre – 1 de noviembre**
    - ***Resolución Tercer Parcial***
    - Evaluación presencial
        - Parte teórica
        - Parte práctica (implementación con herramienta definida)

- **Sesión 13 – Semana 3 – 8 de noviembre**
    - Manejo de clases y objetos
        - Auto referenciación y acceso a métodos de un objeto
        - Manejo de propiedades OO
            - Herencia
            - Encapsulamiento
            - Sobrecarga de métodos
    - Manejo de errores en tiempo de ejecución y excepciones
        - Limite superado en arreglos
        - Tipos incompatibles
        - División por cero
        - Rango de tipo de dato superado.

- **Sesión 14 – Lectura y material preparado para casa**
    - Fundamentos de optimización de código intermedio
        - Definición, objetivos e importancia de la optimización de código intermedio
    - Optimización por bloques
        - Bloques básicos
        - Grafos de flujo
    - Optimización por mirilla
        - Simplificación algebraica
        - Eliminación de código inalcanzable
        - Optimización de flujo de control

CALENDARIZACION DE ACTIVIDADES

Actividad	Publicación de enunciado	Fecha de entrega	Días
Primer proyecto	Jueves 8 de agosto	Martes 17 de septiembre	40
Segundo proyecto	Miércoles 18 de septiembre	Viernes 8 de noviembre	51

ENTREGA DE PROYECTOS

- La entrega de cada uno de los proyectos es individual.
- Para la entrega del proyecto se deberá cumplir con todos los requerimientos mínimos.
- No se recibirán proyectos después de la fecha de entrega.

CALIFICACIÓN DE PROYECTOS

- La calificación de los proyectos se realizará presencialmente y desde los archivos ejecutables.
- No se puede agregar o quitar algún símbolo en el archivo de entrada. El proyecto deberá funcionar con los archivos que sean proveídos por lo auxiliares para la calificación, sin modificación.
- No será permitido compartir los archivos de entrada durante ni después de la calificación.
- La calificación del proyecto será personal y existirá un tiempo límite. Se debe tomar en cuenta que no pueden estar personas ajenas a la calificación, de lo contrario no se calificará el proyecto.
- Anomalías o copias detectadas de proyectos tendrán de manera automática una nota de 0 puntos y los involucrados serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas, para que se apliquen las sanciones correspondientes.
- Existirán horarios para la calificación de cada proyecto, por el cual el estudiante deberá de elegir el horario que mejor le convenga.
- Anomalías detectadas en los archivos entregables tendrá de manera automática una nota de 0 puntos, por ejemplo: no se envió el código correcto, se envió parte del código y no el código completo, archivos ajenos a los entregables del proyecto, no se hizo uso de las herramientas descritas en el enunciado de cada proyecto, entre otras.

BIBLIOGRAFÍA

Compiladores, Principios, Técnicas y Herramientas Aho, Sethi y Ullmam. PEARSON ADDISON-WESLEY, 2008, segunda edición.