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:**	Primer semestre 2019
EDIFICIO:	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 12:30 PM – 14:10 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 ACADÉMICOS:**	
Sección A:	Ing. Bayron López	**Sección A:**	Mike Gutiérrez
Sección B+:	Ing. Erick Navarro	**Sección B+:**	Javier Navarro
Sección B-	Ing. Edgar Sabán	**Sección B-:**	Julio Arango

DESCRIPCION 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 más 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 ESPECIFICOS

1. Que el estudiante sea capaz de desarrollar aplicaciones que sean capaces 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 prácticas y proyectos 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 36 puntos distribuidos de la siguiente manera:

Actividad	**Ponderación**	**Porcentaje**
Primer proyecto	16.2	45%
Segundo proyecto	19.8	55%
Total	**36**	**100%**

Observaciones:

- Para aprobar el laboratorio se debe tener una nota final igual o mayor al 61% de los puntos, es decir 21.96 puntos de 36.
- Cada proyecto tiene requerimientos mínimos para que la calificación del mismo se realice.
- 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 para tener derecho a calificación.

CONTENIDO

Unidad 1: Repaso de cursos anteriores

- **Sesión 1 – semana del 28 de enero al 2 de febrero**
  - 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
        - Optimizador de código intermedio
    - Administración de la tabla de símbolos
      - Elementos que deben ser almacenados en una tabla de símbolos
        - Clases e interfaces
        - Nombres de variables
        - Constantes
        - Nombres de procedimientos y funciones
        - Objetos
        - Temporales
        - Etiquetas
      - Elementos esenciales de un símbolo
        - Tipo
        - Identificador
        - Offset de almacenamiento (tamaño)
        - Dirección base
        - 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
  - ♦ Tabla de símbolos 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
- El agrupamiento de fases en pasadas
- Herramientas de construcción de compiladores

➢ Fundamentos de los lenguajes de programación
- La distinción entre estático y dinámico
- Nombres, identificadores y variables
- Alcance estático y estructura de bloques
- Control de acceso explicito
- Alcance dinámico
- Analogía entre el alcance estático y dinámico
- Mecanismos para el paso de parámetros
  - Llamada por valor
  - Llamada por referencia
  - Llamada por nombre

➢ *Evaluación presencial*
- ***20%** Preguntas sobre conceptos dados*
- ***5%** Construcción e implementación de un nodo Símbolo con los atributos necesarios para un lenguaje especifico.*
- ***25%** Reconocimiento de gramática para declaraciones en estructura de bloques*
- ***60%** Implementación de un entorno para el manejo de la tabla de símbolos*
  - ***10%** Método crear entorno*
  - ***10%** Método colocar o agregar*
  - ***40%** Método obtener (nombre de variable más cercano)*

Unidad 2: Traducción dirigida por la sintaxis

❖ **Sesión 2 – semana del 4 de febrero al 9 de febrero**

➢ Definición de sintaxis
- Definición de gramática libre de contexto
- Derivaciones
- Arboles 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
  - Una definición alternativa
- Evaluación de una DDS en los nodos de un árbol de análisis sintáctico.
- Esquema de traducción
- Diferencia entre una DDS y un esquema de traducción.

➢ Orden de evaluación

- Grafos de dependencias
- TDS con atributos por la izquierda

➢ *Evaluación presencial*

- ***30%** Ejercicio de definición dirigida por la sintaxis utilizando exclusivamente atributos sintetizados a criterio de tutor académico.*
- ***30%** Ejercicio de definición dirigida por la sintaxis utilizando atributos heredados y sintetizados a criterio de tutor académico.*
- ***40%** Construcción de Nodo Declaración e impresión de tabla de símbolos para un lenguaje derivado de Java.*

❖ **Sesión 3 – semana del 11 de febrero al 16 de febrero**

➢ Esquemas de traducción orientados por la sintaxis
- Postfijos
  - Implementación con la pila del analizador sintáctico
- 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
- Construcción de árboles de análisis sintáctico
  - Expresiones
    - ♦ Obtener valor implícito
    - ♦ Validación de errores
      - ➢ División por cero
      - ➢ Rango excedido
- La estructura de tipos

➢ *Evaluación presencial*
- ***60%** Construcción de AST para las expresiones y ejecución de las mismas para un lenguaje derivado de Java **cadena de entrada a criterio de tutor.***
  - ***40%** Resultado correcto*
  - ***10%** Validar error de división por cero*
  - ***10%** Validar que variable este definida*
- ***40%** Estructura de tipos para las expresiones de un lenguaje derivado de Java, validación de error de tipos.*

❖ **Sesión 4 – semana del 18 de febrero al 23 de febrero**

➢ Construcción de AST para un lenguaje derivado de Java
- Sentencias de selección
  - If
  - Switch
- Sentencias cíclicas
  - While
  - For

➢ Construcción de AST para un lenguaje derivado de Java
- Sentencias de transferencia
  - Break
  - Continue
- Estructura de datos Display
  - Implementación con un contador
  - Casos en los que se requiere un nodo con más atributos.

➢ *Evaluación presencial*
- ***50%** Construcción de AST para las sentencias de selección y cíclicas, así como ejecución de las mismas para de un lenguaje derivado de Java*
  - ***10%** Sentencia If*

        - ***10%** Sentencia Switch*
        - ***10%** Sentencia While*
        - ***20%** Sentencia For*
    - ***30%** Construcción de AST para las sentencias de transferencia, así como ejecución de las mismas para un lenguaje derivado de Java*
        - ***10%** Break*
        - ***20%** Continue*
    - ***20%** Implementación del Display y reporte de error por sentencia fuera de lugar.*

❖ **Sesión 5 – semana del 25 de febrero al 2 de marzo**

➢ Construcción de AST para un lenguaje derivado de Java
- Métodos y funciones
    - Retorno
    - Tipo
- Validaciones semánticas para métodos y funciones

➢ Construcción de AST para un lenguaje derivado de Java
- Clases y objetos
    - Definición de clases
    - Instanciación
    - Acceso a atributos y métodos

➢ *Evaluación presencial*
- ***50%** Construcción de AST para los métodos y funciones, así como ejecución de las mismas para un lenguaje derivado de Java*
    - ***20%** Llamada a funciones*
    - ***20%** Valor retornado*
    - ***10%** Recursividad simple*
- ***50%** Construcción de AST para las clases y objetos, así como ejecución de las mismas para un lenguaje derivado de Java*
    - ***20%** Acceso a atributos*
    - ***20%** Seteo de atributos*
    - ***10%** Manejo de referencias adecuada*

❖ **Sesión 6 – semana del 4 de marzo al 9 de marzo**

➢ Construcción de AST para un lenguaje derivado de Java
- Arreglos
    - Declaración e inicialización
    - Acceso
- Validación de errores
    - Numero de dimensiones
    - Fuera de limite
    - Tipos incompatibles
    - Índices de tipo entero
    - Definición consistente

➢ *Evaluación presencial*
- ***60%** Construcción de AST para arreglos, así como ejecución de las mismas para un lenguaje derivado de Java*
    - ***20%** Mapeo correcto*
    - ***20%** Acceso a elemento en posición* i
    - ***20%** Seteo de valor a posición* i
- ***40%** Validación de errores con arreglos.*

Unidad 3: Generación de código intermedio

❖ **Sesión 7 – semana del 11 de marzo al 16 de marzo**

➢ Tipos de representaciones intermedias
- Arboles
- Representaciones lineales

- Código de tres direcciones
  - Traducción de expresiones
    - Aritméticas
    - Relacionales
    - Lógicas
      - Convencional
      - Corto circuito
- *Evaluación presencial*
  - ***100%** generación de código 3D para las expresiones de un lenguaje derivado de Java, entrada a criterio del tutor.*

❖ **Sesión 8 – semana del 18 de marzo al 23 de marzo**

- Código de tres direcciones
  - Traducción de sentencias de selección
    - If
    - Switch
  - Traducción de sentencias cíclicas
    - While
    - For
  - Traducción de sentencias de transferencia
    - Break
    - Continue
    - Return
  - Display
- *Evaluación presencial*
  - ***100%** generación de código 3D para las sentencias de selección, cíclicas y de transferencia para un lenguaje derivado de Java, entrada a criterio del tutor.*

Unidad 4: Entornos locales y estructuras en tiempo de ejecución

❖ **Sesión 9 – semana del 25 de marzo al 30 de marzo**

- Organización del almacenamiento
  - Asignación de almacenamiento estática y dinámica
  - Asignación de espacio en pila
    - Arboles de activación
    - Registros de activación
    - Secuencia de llamadas
    - Datos de longitud variable en la pila
    - Variables globales
    - Métodos y funciones
    - Variables locales
    - Parámetros
      - Por valor
      - Por referencia
- **Evaluación presencial**
  - ***100%** generación de código 3D para llenado de pila y manejo de métodos y funciones para un lenguaje derivado de Java*
    - ***30%** Reservar memoria para el entorno local definido por una función*
    - ***30%** Manejo de recursividad guardando temporales*
    - ***20%** Paso de parámetros, retorno y cambio de ámbito.*
    - ***20%** Limpieza de ámbito tras la ejecución de una llamada.*

❖ **Sesión 10 – semana del 1 de abril al 6 de abril**

- 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 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
- Manejo de arreglos de una o más dimensiones
  - Mapeo de un arreglo de una o más dimensiones a uno unidimensional
    - Por fila
    - Por columna
  - 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

- **Evaluación presencial**
  - *100% generación de código 3D para arreglos y cadenas el manejo en el montículo para un lenguaje derivado de Java*
    - *50% Almacenamiento de cadenas y concatenación.*
    - *20% Tamaño y numero de dimensiones*
    - *30% Acceso a posición* i *del arreglo linealizado*

- **Sesión 11 – semana del 22 de abril al 27 de abril**
  - Organización del almacenamiento
    - 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
        - Auto referenciación y acceso a métodos de un objeto
        - Manejo de propiedades OO
          - Herencia
          - Encapsulamiento
          - Sobrecarga de métodos
      - Reducción de la fragmentación
        - Mejor ajuste
        - Siguiente ajuste
      - Administración y coalescencia de espacio libre
        - Coalescencia con etiquetas delimitadoras
        - Coalescencia lista libre incrustada, doblemente enlazada
      - Convenciones y herramientas de programación
      - Introducción a la recolección de basura
        - Seguridad en los tipos
        - Métrica de rendimiento
  - Comprobación de tipos

- Comprobación estática
- Comprobación dinámica
- Equivalencia de expresiones de tipos
- Conversión de tipos
- Sobrecarga de funciones y operadores

➢ *Evaluación presencial*
- *50% implementación de montículo y manejo de atributos de clase y this.*
- *50% generación y ejecución de código 3D para comprobar tipos.*

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

❖ **Sesión 12 – semana del 29 de abril al 4 de mayo**

➢ Errores en tiempo de ejecución
- Limite superado en arreglos
- Tipos incompatibles
- División por cero
- Rango de tipo de dato superado.

➢ 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

➢ *Evaluación presencial*
- *50% generación de código 3D para validar errores en tiempo de ejecución*
- *50% optimizador por mirilla.*
- *50% optimizador por bloques*

CALENDARIZACION DE ACTIVIDADES

Actividad	Publicación de enunciado	Publicación hoja de calificación y archivos de prueba	Fecha de entrega	Días
Primer proyecto	miércoles 13 de febrero	lunes 4 de marzo	miércoles 20 de marzo	35
Segundo proyecto	miércoles 20 de marzo	lunes 15 de abril	miércoles 8 de mayo	49

ENTREGA DE PROYECTOS

- La entrega será virtual y por medio de la plataforma Classroom.
- 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.

CALIFICACION 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.
- Existirá 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 código 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.

BIBLIOGRAFIA

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