I. IDENTIFICACIÓN

1.	Asignatura	: **Compiladores y Lenguajes de Bajo Nivel**
2.	Código	: **6.2.A**
3.	Horas semanales	: 5 horas
4.	Total de horas	: 80 horas

II. JUSTIFICACIÓN

La teoría de Compiladores es un área de conocimientos fundamental para la formación del estudiante de Ciencias de la Computación. Posee una relevancia práctica y teórica en la formación del futuro profesional tanto en el aspecto de la programación de computadoras como en la construcción de la base teórica de conocimientos del mismo.

El diseño de compiladores implica conocimientos avanzados en estructuras de datos, diseño de software y arquitecturas de bajo nivel, y la aplicación de estos conocimientos en forma práctica. Esto hace que la materia se convierta en una herramienta efectiva al momento de consolidar la formación del alumno y le brinda capacidades para en el futuro poder aplicar los conocimientos y resolver problemas reales.

III. OBJETIVOS GENERALES

1. Comprender los principios de la teoría de Compiladores.
2. Aplicar los conocimientos teóricos mediante la programación de las fases de un compilador.
3. Aplicar técnicas de la teoría de Compiladores a problemas específicos.
4. Comprender fundamentos teóricos de la Ciencia Computacional.

IV. OBJETIVOS ESPECÍFICOS

Conocimientos

1. Enumerar y explicar las fases de un Compilador, diferenciando traductores, intérpretes, máquinas abstractas/virtuales (*virtual machines*).
2. Explicar diferencias entre un código compilado e interpretado, y trabajar con gramáticas de lenguajes de programación de diferentes niveles de abstracción y paradigmas.
3. Comprender gramáticas de libre contexto modeladas mediante la notación BNF
4. Comprender un área de la Ciencia Computacional referente a la teoría de lenguajes, el reconocimiento de programas fuentes y la generación de código objeto.

Habilidades

1. Implementar reconocedores de lenguajes de programación (*parsers*) a partir de gramáticas de libre contexto y expresiones regulares.
2. Implementar generadores de código basados en reglas de traducción dirigida por sintaxis.
3. Modelar un lenguaje de programación utilizando la notación BNF.

Competencias

1. Capacidad de aplicar los conocimientos teóricos en la práctica.
2. Capacidad de formular modelos mentales para la resolución de problemas mediante la aplicación de algoritmos de la teoría de lenguajes.
3. Capacidad para desarrollar soluciones de complejidad avanzada.

V. PRE – REQUISITO

1. Redes I
2. Estructura de los Lenguajes

VI. CONTENIDO

6.1. Unidades programáticas

1. Introducción a la teoría de Compiladores.
2. Análisis Léxico.
3. Gramáticas de Libre Contexto.
4. Análisis Sintáctico.
5. Análisis Semántico.
6. Traducción Dirigida por Sintaxis.

6.2. Desarrollo de las unidades programáticas

1. Introducción a la teoría de Compiladores
   - 1.1. Historia de los compiladores
   - 1.2. Programas relacionados a los compiladores
   - 1.3. El proceso de traducción
     - 1.3.1. Análisis Léxico
     - 1.3.2. Análisis Sintáctico
     - 1.3.3. Análisis Semántico
     - 1.3.4. Optimizador de Código
     - 1.3.5. Generador de código
     - 1.3.6. Optimizador de código objeto
   - 1.4. Estructuras de Datos de un Compilador
   - 1.5. Otros temas relacionados a compiladores
2. Análisis Léxico
   - 2.1. Proceso de *scanning*
   - 2.2. Expresiones Regulares
   - 2.3. Autómatas Finitos
     - 2.3.1. Autómatas finitos deterministas (DFA)
     - 2.3.2. Autómatas finitos no deterministas (NFA)
   - 2.4. Algoritmo de conversión de Expresiones Regulares a DFA
   - 2.5. Conversión de NFA a DFA
   - 2.6. Generadores de analizadores léxicos
3. Gramáticas de Libre Contexto
   - 3.1. El proceso de *parsing*
   - 3.2. Gramáticas de Libre Contexto (GLC)
   - 3.3. Árboles Sintácticos y árboles de sintaxis abstracta
   - 3.4. Ambigüedad
   - 3.5. Notación BNF y EBNF
   - 3.6. Propiedades formales de una GLC
     - 3.6.1. Jerarquía de Chomsky
4. Análisis Sintáctico
   - 4.1. Análisis Sintáctico Descendente Recursivo
   - 4.2. Análisis Sintáctico LL(1)
   - 4.3. Conjuntos Primero y Siguiente
   - *4.4.* Recuperación de errores
     - 4.4.1. *Tokens* de Sincronización
     - 4.4.2. Corrección de errores
   - 4.5. Análisis Sintáctico Ascendente
   - 4.6. Análisis Ascendente LR(0)
   - 4.7. Análisis Ascendente SLR(1)
5. Análisis Semántico
   - 5.1. Atributos y gramáticas con atributos
   - 5.2. Algoritmos para el cómputo de atributos
   - 5.3. Tabla de símbolos
   - 5.4. Tipos de datos
   - 5.5. Chequeos estáticos
   - 5.6. Chequeos dinámicos
6. Traducción Dirigida por Sintaxis
   - 6.1. Definiciones y construcción de árboles sintácticos
   - 6.2. Evaluación descendente
   - 6.3. Definiciones con atributos por izquierda
   - 6.4. Traducción descendente
   - 6.5.

VII. ESTRATEGIAS METODOLÓGICAS

1. Exposición de contenidos sobre la Teoría de Compiladores y su aplicación en la Ciencia Computacional.
2. Prácticas de programación en Laboratorio, de manera a consolidar conceptos teóricos con la práctica.

3. Tareas de implementación. Aplicación de conceptos para el desarrollo de un compilador.
4. Trabajo Práctico Final, desarrollo de fases de un compilador.

VIII. MEDIOS AUXILIARES

1. Pizarra.
2. Proyector.
3. Textos, transparencias.
4. Ejercitarios.
5. Sala de máquinas.

IX. EVALUACIÓN

1. Sumativa (exámenes parciales y finales).
2. Participación en clase.
3. Presentación de tareas y prácticas.
4. Trabajo Práctico Final.

X. BIBLIOGRAFÍA

- Lounden, K. Construcción de compiladores: principios y práctica / K. Louden: PWS Publishing Company.
- Aho, A. Compiladores: principios, técnicas y herramientas / A. Aho, R. Sethi. J. Ullman : Addison Wesley Longman.
- Sebesta, R. Conceptos de Lenguajes de Programación / R. Sebesta : Pearson Addison Wesley.