UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

COORDINACIÓN DE FORMACIÓN BÁSICA
COORDINACIÓN DE FORMACIÓN PROFESINAL Y VINCULACIÓN UNIVERSITARIA
PROGRAMA DE UNIDADES DE APRENDIZAJE POR COMPETENCIAS

I. DATOS DE IDENTIFICACIÓN

1. Unidad Académica: **Facultad de Ciencias**
2. Programa (s) de estudio: (Técnico, Licenciatura) **Ciencias Computacionales** 3. Vigencia del plan: **2008-1**
4. Nombre de la Asignatura: COMPILADORES 5. Clave: 9840
6. HC: 2 HL: 2 HT: 2 HE: 0 CR: 8
7. Ciclo Escolar: 2009-1 8. Etapa de formación a la que pertenece: TERMINAL
9. Carácter de la Asignatura: Obligatoria X Optativa _____
10. Requisitos para cursar la asignatura: TEORIA DE LA COMPUTACIÓN I, TEORÍA DE LA COMPUTACIÓN II

Formuló: M.C. María Victoria Meza Kubo VoBo. Biól. Marcelo Rodríguez Meraz

Fecha: Junio 2009 Cargo: Subdirector

II. PROPÓSITO GENERAL DE LA UNIDAD DE APRENDIZAJE

El alumno hará un estudio comparativo de los principios, técnicas y herramientas para el desarrollo de cada una de las etapas (de análisis y síntesis) que se encuentran involucradas en el proceso de compilación.

Dar al alumno una visión amplia sobre el uso y aplicación de los compiladores como solución a disitintos problemas reales.

Mostrar al alumno la importancia del desarrollo de compiladores, como una solución a problemas o como una herramienta más en el caso de necesitar un compilador con características especiales para el desarrollo de proyectos futuros.

Esta materia es obligatoria y se encuentra en la etapa terminal.

III. COMPETENCIA (S) DE LA UNIDAD DE APRENDIZAJE

- Conocimiento de los principios, técnicas y herramientas aplicadas en el proceso de compilación.
- Diseño de gramáticas independientes al contexto y lenguajes.
- Construcción de los analizadores y traductores para la implementación de soluciones basados en el proceso de compilación (traducción).

IV. EVIDENCIA (S) DE DESEMPEÑO

Diseño e implementación de un lenguaje y gramática independiente al contexto, así mismo del compilador para la solución de un problema real.

Desarrollo de propuesta formal, presentación del sistema y resultados obtenidos.

V. DESARROLLO POR UNIDADES

UNIDAD I. Introducción

Competencia:

Después de realizar una introducción a los conceptos básicos del proceso de compilación, el alumno podrá:

- Definir de forma general las fases del proceso de compilación
- Identificar un problema real que pueda se resuelto por medio del proceso de compilación

Contenido	**Duración**
Introducción	**6 hrs**

1. Conceptos básicos
2. Idea básica de un compiladores
3. Compilador vs Interpretes
4. Etapas del proceso de compilación
5. Herramientas para la manipulación de programas fuentes
6. Compilación y la presencia de errores
7. Programas relacionados con un compilador
8. Procesamiento de un programa
9. Fases de un compilador
10. Análisis de un compilador sencillo
11. Como desarrollar un compilador

V. DESARROLLO POR UNIDADES

UNIDAD II. Lenguajes y Gramáticas Independientes al Contexto

Competencia:

En base a las técnicas vistas en clase el alumno podrá:

- Diseñar el lenguaje de su sistema.
- Determinar si una gramática es o nó libre de contexto y en caso de no serlo poder convertirla.
- Determinar si una gramática es o nó ambigüa y en caso de tener ambigüedad la gramática eliminarla

Contenido	**Duración**
Lenguajes y Gramáticas Independientes al Contexto	**6 hrs**

1.Repaso de definiciones
2.Operaciones aplicadas a lenguajes
3.Expresiones regulaes
4.Gramáticas independientes al contexto
5.Escritura de una gramática
6.Árbol de análisis sintáctico
7.Ambigüedad
8.Recursión

V. DESARROLLO POR UNIDADES

UNIDAD III. Análisis Léxico

Competencia:

En base a los conceptos y técnicas vistas en clase el alumno podrá:

- Definir las distintas funciones que desempeña el analizador léxico en el proceso de compilación
- Establecer los posibles errores léxicos que requiera detectar en el sistema a ser desarrollado
- Implementar un analizador léxico con recuperación de errores léxicos

Contenido	**Duración**
Análisis Léxico	**12 hrs**

1. Función del analizador léxico
   1.1. Tabla de símbolos
2. Interacción del analizador léxico con el analizador sintáctico
   2.1. Razones para separar el análisis léxico del análisis sintáctico
3. Definiciones
   3.1. Componentes léxicos
   3.2. Patrones
   3.3. Lexema
4. Especificación y reconocimiento de componentes léxicos
5. Errores léxicos
6. Lectura de programas fuentes
   6.1. Manejo de buffers de entrada

V. DESARROLLO POR UNIDADES

UNIDAD IV. Análisis Sintáctico

Competencia:

En base a las técnicas vistas en clase el alumno podrá:

- Identificar los diferentes tipos de analizadores sintácticos, así como sus ventajas y desventajas
- Establecer los errores sintácticos que requiera que sean detectados para su sistema
- Implementar un analizador sintáctico con recuperación de errores sintácticos

Contenido	**Duración**
Análisis sintáctico	**24 hrs**

1. Objetivos del análisis sintáctico
2. Tipos de analizadores sintácticos
3. Manejador de errores en un analizador sintáctico
4. Estrategias para recuperación de errores
5. Análisis sintáctico descendente con retroceso
6. Análisis sintáctico descendente predictivo
7. Análisis sintáctico predictivo no recursivo
8. Análisis sintáctico ascendente por desplazamiento y reducción
9. Análisis sintáctico por precedencia de operadores

V. DESARROLLO POR UNIDADES

UNIDAD V. Traducción dirigida por sintaxis

Competencia:

Por medio de las técnicas vistas en clase el alumno podrá implementar un traductor por sintaxis.

Contenido	Duración
Traducción dirigida por sintaxis	**6 hrs**

1. Definiciones dirigidas por sintaxis
2. Construcción de árboles sintácticos
3. Análisis de definiciones dirigidas por la sintaxis

V. DESARROLLO POR UNIDADES

UNIDAD VI. Comprobación de tipos

Competencia:

En base a los conceptos y técnicas vistas en clase el alumno podrá:

- Identificar la estructura semántica de su gramática para establecer así los errores semánticos requeridos en su sistema
- Implementar un analizador semántico en base a las necesidades definidas anteriormente

Contenido	**Duración**
Comprobación de tipos	**8hrs**

1. Sistemas de tipos
2. Especificación de un comprobador de tipos sencillo
3. Conversiones de tipos
4. Sobrecarga de funciones y operadores
5. Funciones polimórficas

V. DESARROLLO POR UNIDADES

UNIDAD VII. Generación de Código

Competencia:

En base a lo visto en clase el alumno conocerá las diferentes técnicas de generación de código intermedio que se aplican al proceso de compilación. Podrá también determinar cual es la técnica más adecuada para el problema o sistema en desarrollo. Esto se demostrará con la implementación de un generador de código intermedio para el sistema a ser desarrollado.

Contenido	**Duración**
Generación de Código	**12 hrs**

1. Lenguajes intermedios
2. Aspectos del diseño de un generador de código
3. La máquina objeto
4. Un generador de código simple
5. Representación de bloques básicos por medio de GDA
6. Generación de código apartir de los GDA

VI. ESTRUCTURA DE LAS PRÁCTICAS

No. de Práctica	Competencia(s)	Descripción	Material de Apoyo	Duración
1	Analizar una problemática real así como de las alternativas actuales para la solución de la misma y de la aportación que tendrá la aplicación de un sistema de traducción en la solución del problema.	Realizar una propuesta de proyecto en el cual se realice una descripción amplia de una problemática real, antecedentes de solución de la misma, justificación, objetivos generales y específicos, metas, metodología del trabajo, plan de trabajo (calendarización) y limitaciones del trabajo.	• Notas de clase • Papel y lápiz • Internet • Procesador de palabras.	10 hrs
2	Diseño de interfaces de software	Realizar un bosquejo de la interface gráfica del compilador, en el cual se especifiquen las acciones que usuario podrá desarrollar y la forma de interactuar del usuario con el sistema.	• Papel y lápiz • Programa para el diseño de gráficos • Procesador de palabras.	15 hrs
3	Diseño de lenguaje y gramática libre de contexto de forma estándar.	Especificar el lenguaje a ser creado, para la solución propuesta y a ser utilizado en el compilador como lenguaje de entrada. Diseño de gramática libre de contexto en forma estándar que genere el lenguaje diseñado. Determinar los conjuntos de terminales y no terminales que conforman la gramática.	• Papel y lápiz. • Procesador de palabras.	20 hrs
4	Implementación de análisis léxico.	Implementar un analizador léxico que tenga como entrada un programa en el lenguaje creado y una tabla de símbolos; como salida tenga una cadena de tokens y la tabla de símbolos actualizada en base al análisis léxico. El analizador léxico debe ser capáz de reportar errores y el tipo del mismo, y en caso de encontrar errores entonces recuperarse para así continuar con las demás etapas de análisis.	• Papel y lápiz • Procesador de texto • Lenguaje de programación	20 hrs

5	Implementación de análisis sintáctico.	Elaborar estudio de las diferentes técnicas de análisis sintáctico para así aplica el más adecuado a las necesidades de la solución a implementar por el alumno. Implementar la(s) técnica(s) de análisis sintáctico, que tendrá como entrada una cadena de tokens, la cual fue previamente generada por el analizador léxico, para así determinar si se acepta o nó la cadena. El analizador deberá ser capaz de recuperarse ante la presencia de errores y reportar de los mismos y su tipo al usuario.	• Papel y lápiz • Procesador de texto • Lenguaje de programación	30 hrs
6	Implementación de análisis semántico.	Implementar una analizador semántico o comprobador de tipos, el cual recibirá como entrada la cadena de tokens aceptada por el analizador sintáctico y deberá producir como salida la aceptación o negación de la cadena. En caso de encontrar errores éste deberá recuperarse para continuar así el análisis y presentar un listado de errores y el tipo de error encontrado.	• Papel y lápiz • Procesador de texto • Lenguaje de programación	15 hrs
7	Implementación de un traductor dirigido por sintaxis.	Implementar el traductor dirigido por sintaxis para el lenguaje de entrada a lenguaje objeto, éste último puede ser cualquier lenguaje de programación conocido por el alumno. La traducción generada en lenguaje objeto no deberá contener errores para que pueda ser utilizado.	• Papel y lápiz • Procesador de texto • Lenguaje de programación	20 hrs
8	Reportar las resultados.	Elaborar reporte el cuál reflejará el resultado final del proyecto, metas alcanzadas, conclusiones y trabajo a futuro relacionado con el proyecto, Desarrollar una presentación final de 20 min en la cual se presentarán los resultados finales y el compilador desarrollado así como su funcionamiento.	• Papel y lápiz • Procesador de texto • Programa de presentaciones	10hrs

VII. METODOLOGÍA DE TRABAJO

Elaboración de propuesta de proyecto y presentación de la misma

Al término de la unidad 1 de la materia el deberá realizar una propuesta de proyecto aplicando los sistemas traductores (compiladores) en la solución de una problemática real. En la propuesta el equipo debe presentar una descripción de la problemática a ser atacada, la justificación del uso de un compilador como parte de la solución de la problemática.

Implementación de sistema propuesto y presentación final

Los equipos deberán implementar el sistema de traducción (compiladores) propuesto al inicio de semestre, para así mostrar al final los resultados obtenidos en base a lo implementado.

Trabajo en equipo, el proyecto se elaborará en equipo de máximo 3 personas

A lo largo del semestre se irá implementado la propuesta hecha en equipo.

Realización de tareas y ejercicios extra clase.

Durante el semestre se realizarán varios ejercicios extra clase para una mejor comprensión de los temas.

Exámenes escritos y orales.

Por medio de exámenes al final de cada unidad, el alumno podrá verificar su avance o deficiencias y de esta forma poner mayor atención en el tema, de igual forma se evalúa si los temas visto durante la unidad han sido evaluados.

Participación en clase

A lo largo del semestre se desarrollarán ejercicios explicados y dirigidos por el maestro, los cuales servirán de guía para las tareas y ejercicios extra clase a ser realizadas por el alumno.

VIII. CRITERIOS DE EVALUACIÓN

Criterios de calificación:

- Exámenes 20%
- Ejercicios 20%
- Proyecto 50%
- Presentación del proyecto 10%

TOTAL 100%

Criterios de acreditación:

1. Resolver 4 exámenes en tiempo y forma
2. Cumplir con las tareas y ejercicios extra clase en tiempo y forma.
3. Cumplir con la presentación de propuesta de proyecto en tiempo y forma
4. Cumplir con la presentación final del proyecto en tiempo y forma

Criterios de evaluación:

En el caso del proyecto final por equipo, la evaluación se dividirá en dos: documentación e implementación, en el primer caso la calificación será por equipo y los puntos a evaluar serán, contenido, redacción, forma de presentación del trabajo, así como ortografía; para el segundo caso la calificación será en equipo, el proyecto será aprobatorio si fue terminado al 100% de acuerdo con la propuesta hecha en un principio y los puntos a evaluar serán interfaz de usuario, facilidad de utilización.

IX. BIBLIOGRAFÍA

Básica	Complementaria
COMPILADORES: Principios, técnicas y herramientas. Autores: Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman Ed. Addison Wesley Compilers Design in C. Autores: Holub, Allen I. Ed. MT. Books	Machines, Lenguages and Computation Autor: Denning Peter Ed. Prentice Hall