UNIVERSIDAD DE GUADALAJARA

1. DATOS GENERALES DE LA UNIDAD DE APRENDIZAJE (UA) O ASIGNATURA			
Nombre de la Unidad de Aprendizaje (UA) o Asignatura			**Clave de la UA**
Traductores de Lenguajes II			I7027
Modalidad de la UA	**Tipo de UA**	**Área de formación**	**Valor en créditos**
Escolarizada	Curso	Básica particular	8

UA de pre-requisito	UA simultaneo	UA posteriores
Teoria de la Computacion(I5802) Traductores de Lenguajes I(17025)	Solucion de Problemas de Traductores de Lenguajes II	Inteligencia Artificial I (I7038)
Horas totales de teoría	**Horas totales de práctica**	**Horas totales del curso**
68	0	68

Licenciatura(s) en que se imparte	Módulo al que pertenece
Ingeniería en Computacion	Módulo arquitectura y programación de sistemas
Departamento	**Academia a la que pertenece**
Ingeniería en Computacion (CUCEI)	Software de Sistemas
Elaboró	**Fecha de elaboración o revisión**
Armando Ramos Barajas	02/07/2018

2. DESCRIPCIÓN DE LA UA O ASIGNATURA

Presentación

Caracterización de la asignatura.

En esta asignatura se debe desarrollar el análisis léxico, análisis, análisis semántico, la optimización y la generación del código objeto para obtener el funcionamiento de un compilador. Esta asignatura busca proveer al estudiante de herramientas, conocimientos y habilidades necesarias para desarrollar un compilador con base en los conocimientos previos de la asignatura de Teoría de la Computacion. La aportación de esta materia es relevante en el ámbito del desarrollo de software de sistemas. Es indispensable distinguir que la carrera de Ingeniería en Computacion se basa no sólo en el desarrollo de software comercial y administrativo, sino también en el desarrollo de software científico y para el desarrollo tecnológico. Esta materia se ubica en la segunda categoría y es indispensable desarrollar software en estos campos para preparar a los egresados y tengan la posibilidad de cursar posgrados de alto nivel. La asignatura trata de concretar un traductor iniciado en la materia previa para que el estudiante comprenda que es capaz, mediante técnicas bien definidas, de crear su propio lenguaje de programación. La aportación de la asignatura al perfil del egresado será específicamente la siguiente:

- Desarrollar, implementar y administrar software de sistemas o de aplicación que cumpla con los estándares de calidad buscando como finalidad apoyar la productividad y competitividad de las organizaciones.
- Integrar soluciones computacionales con diferentes tecnologías, plataformas o dispositivos.
- Diseñar e implementar interfaces hombre – máquina y maquina – máquina para la automatización de sistemas.
- Identificar y comprender las tecnologías de hardware para proponer, desarrollar y mantener aplicaciones eficientes.

Relación con el perfil

Modular	De egreso
Esta materia, junto con las demás que conforman el módulo de "Generaion de Codigo" tiene como finalidad que sus egresados puedan construir modelos de Compiladores identificando las variables de decisión, la función objetivo y las restricciones a partir de una situación o fenómeno real. En particular, en esta materia se pretende que puedan tomar decisiones, a través de la solución a un traductor o compilador.	Esta materia contribuye a desarrollar la habilidad para analizar y diseñar modelos de Traductoes y Compiladores, aplicando técnicas cuantitativas para la optimización de procesos integrando recursos cientificos

Competencias a desarrollar en la UA o Asignatura

Transversales	Genéricas	Profesionales
Utiliza su capacidad de abstracción, análisis y síntesis para identificar y resolver problemas de compiladores Interpreta fenómenos en términos matemáticos para la comprensión y construcción de modelos matemáticos para la construccon de Compiladores y traductores.	**Competencias instrumentales** • Capacidad de análisis y síntesis • Conocimientos básicos de la carrera • Comunicación oral y escrita • Conocimiento de una segunda lengua • Conocimiento generales básicos del lenguaje ensamblador. • Habilidad para buscar y analizar información proveniente de fuentes diversa. • Habilidad lógica para solucionar problemas • Habilidades del manejo de la computadora **Competencias interpersonales**	Elabora modelos matemáticos de Compiladores de un lenguaje de programacion . Emplea lenguaje de bajo nivel ensamblador para el desarrollo de un compilador.

	• Capacidad crítica y autocrítica • Trabajo en equipo interdisciplinario • Habilidades interpersonales **Competencias sistémicas** • Capacidad de aplicar los conocimientos en la práctica • Habilidades de investigación • Estándares de desarrollo para la implementación de soluciones • Capacidad de aprender • Capacidad de generar nuevas ideas (creatividad) • Habilidad para trabajar en forma autónoma • Capacidad para diseñar y gestionar proyectos • Búsqueda del logro	

Saberes involucrados en la UA o Asignatura		
Saber (conocimientos)	**Saber hacer (habilidades)**	**Saber ser (actitudes y valores)**
Características de un problema de Traductores Planteamiento de traductores. Solución de problemas mediante el Automatas y tablas de transicones , método de las dos formas de diseñar un traductor . Elaboración del dual de un problema primal. Solución de problemas mediante el método simplex dual. Variaciones que puede tener un compilador Técnicas de análisis de sensibilidad para la solución de problemas de traductores.	Identificar y organizar la información que se requiere para plantear un problema de traductores. Discriminar y analizar información relevante. Identificar variables de decisión. Identificar la función objetivo. Identificar las restricciones. Analizar las posibles soluciones a partir de la región factible. Identificar y utilizar el método más adecuado para resolver los problemas de acuerdo a sus características. Interpretar la solución encontrada para optimizar los sistemas. Redactar con claridad respetando reglas ortográficas y sintácticas Utilizar software especializado para la solución de problemas. Valorar el empleo de herramientas computacionales en la solución de problemas de traductores.	Muestra seguridad al escribir instrucciones y transmitir mensajes de sintaxis del compilador Cumple con los acuerdos establecidos en equipo Presenta sus fases en tiempo y forma, de tal manera que demuestra la funcionalidad de cada una de sus fases

UNIVERSIDAD DE GUADALAJARA

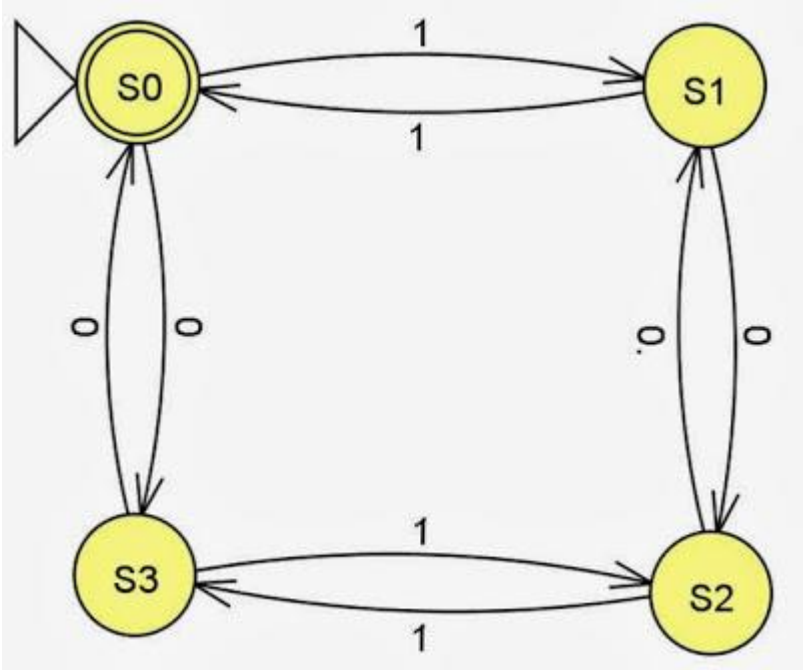
Producto Integrador Final de la UA o Asignatura
Título del Producto: Desarrollo de un compilador utilizando tablas de transiciones y digramas de transiciones, asi como lenguajes de alto y bajo nivel. En esta unidad de aprendizaje se espera que sean capaces de plantear y resolver las 4 etapas de un compilador. Para resolver las etapas se deben de identificar el método a utilizar. Los conocimientos y habilidades involucrados en la asignatura se logran aplicando estos saberes en los problemas que se dejan de tarea. Es mediante exámenes que se puede determinar si lograron desarrollar las fases. Cada uno de los 2 exámenes es un producto integrador. **Objetivo**: Construir un compilador empleando los lenguajes de alto y bajo nivel, usando expresion regulares **Descripción**: Los diagramas de transiciones son grafos dirigidos donde su funcion principal es la de llegar a un estado de aceptacion, asi como tambien las tablas de transiciones son arreglos bidimensionales que nos permiten tener una mejor vision del lenguaje del alfabeto que se ha usar

3. ORGANIZADOR GRÁFICO DE LOS CONTENIDOS DE LA UA O ASIGNATURA

Tabla de Transición de Estados		
Estado	Entrada	
	1	0
S0	S1	S3
S1	S0	S2
S2	S3	S1
S3	S2	S0

Una **tabla de transiciones** es un arreglo (o matriz) bidimensional cuyos elementos proporcionan el resumen de un diagrama de transiciones correspondiente.

Las cadenas que deben analizarse en una aplicación están construidas a partir de un conjunto de símbolos. En cualquier situación encontramos que el conjunto de símbolos es finito, por lo que nuestro primer paso hacia la formalización del proceso de reconocimiento es asumir la hipótesis de la existencia de una conjunto finito, no vacío, de símbolos a partir del cual se construyen las cadenas que se analizaran. A este conjunto de símbolos lo llamamos **alfabeto.**



Decimos que una cadena de símbolos es aceptada por un **diagrama de transiciones** si los símbolos que aparecen en la cadena (de izquierda a derecha) corresponden a una secuencia de arcos rotulados que conducen del círculo designado por el apuntador a un círculo doble.

Los círculos de un **diagrama de transiciones** representan posiciones, o estados, donde no podemos encontrar al evaluar una cadena de símbolos. Es común llamar estados a los círculos de un**diagrama de transiciones**. Él circulo de partida se llama **estado inicial** y los círculos dobles,**estados de aceptación.**

4. SECUENCIA DEL CURSO POR UNIDADES TEMÁTICAS

Unidad temática 1: Analisis Lexico

Objetivo de la unidad temática: Definir, diseñar, construir y programar las fases del analizador léxico y sintáctico de un traductor o compilador.

Introducción: En esta unidad se explicará la funcionalidad de lo que realiza en analisis lexico

Contenido temático	Saberes involucrados	Producto de la unidad temática
1.1 Funciones del analizador léxico 1.2 Componentes léxicos, patrones y lexemas 1.3 Creación de Tabla de tokens 1.4 Errores léxicos 1.5 Generadores de analizadores Léxicos 1.6 Aplicaciones (Caso de estudio)	Concepto de Seminario de solucion de problemas de traductores de Lenguajes II Concepto de optimización. Utilidad del analisis lexico. Aplicaciones . Utiliza en analisis lexico en un lenguajes especifico Presentar su funcionamiento en tiempo y forma, de tal manera que demuestra la gramatica a utilizar Escucha la opinión de sus compañeros y expresa la suya con apertura	Reporte con la descripción del analisis lexico 1. Descripción de la gramatica 2. Tablas de transicones 3. Diagramas de transiciones 4. Jflap

Actividades del docente	Actividades del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado
Explicar el origen de la gramatica utilizada Explicar que son los tokens Explicar la funcionalidad de los tokens Explicar que es un estado Explicar que es una transicion Explicar que esta una tabla de transiciones	Elabora un reporte acerca de la fase inicial de un compiladores, analisis lexico	Reporte elaborado.	Diapositivas opcionales.	3

Unidad temática 2: Analisis Sintactico

Objetivo de la unidad temática: Comprender como funciona el analizador sintáctico, las gramáticas que debe aceptar y como implementarlo

Introducción: La principal tarea del analizador sintáctico (o parser) no es comprobar que la sintaxis del programa fuente sea correcta, sino construir una representación interna de ese programa y, en el caso en que sea un programa incorrecto, dar un mensaje de error.

Contenido temático	Saberes involucrados	Producto de la unidad temática
18 hrs. 2.1 Introducción 0.5 hrs. OBJETIVO DEL TEMA 2.1.1 Función del analizador sintáctico Conocer el objetivo principal del analizador sintáctico 2.1.2 Representación de gramáticas Conocer como se pueden representar las gramáticas que utilizara el análisis sintáctico 2.2 Gramáticas Libres de Contexto 1.5 hrs. Conocer cuáles son las características de una gramática libre de contexto para así poder identificarlas. 2.2.1 Definición Conocer como se describe formalmente una gramática libre de contexto 2.2.2 Derivaciones Comprender como se puede generar una palabra del lenguaje mediante derivaciones 2.2.3 Árbol de análisis sintáctico Aprender a construir arboles de análisis sintáctico 2.2.4 Ambigüedad Aprender a detectar si una gramática es ambigua. 2.2.5 Eliminación de la recursividad por la izquierda Aprender a quitar la recursividad por la izquierda de una gramática. 2.2.6 Factorización por la izquierda Aprender a factorizar gramáticas. 2.3 Análisis sintáctico descendente mediante método recursivo 2 hrs. Aprender a utilizar y construir analizadores descendentes 2.3.1 Descripción Conocer las características más importantes del método 2.3.2 Gramáticas LL Ser capaz de identificar las características que tienen las gramáticas que son LL. 2.3.3 Implementación del método Realizar la implementación del método	Concepto de modelo matemático. Concepto de programación de alto y bajo nivel. Utilidad de la construcción de modelos de programación Orientada a Objetos	Reporte con la descripción del concepto de programación y de los modelos matemáticos de programación. Deberá incluir: 1. Características del analisis sintactico 2. Elementos como tablas de transiciones 3.- Arboles de derivacion .

2.3.4 Uso de producciones Aprender a utilizar reglas durante la implementación del algoritmo. 2.4 Análisis sintáctico descendente mediante método no recursivo 6 hrs. Aprender a utilizar y construir analizadores descendentes 2.4.1Descripción Comprender de forma general el método 2.4.2 Gramáticas LL(1) Comprender cuales son las gramáticas LL(1) 2.4.3 Algoritmo de análisis sintáctico LL(1) Aprender a utilizar el algoritmo de análisis sintáctico LL(1) 2.4.4 Calculo del conjunto primero Aprender a calcular el conjunto primero los no terminales de una gramática. 2.4.5 Calculo del conjunto siguiente Aprender a calcular el conjunto siguiente los no terminales de una gramática. 2.4.6 Construcción de tablas de análisis sintáctico predictivo no recursivo Comprender como se realiza la construcción de tablas para este método. 2.5 Análisis sintáctico ascendente 6 hrs Aprender a utilizar y construir analizadores ascendentes 2.5.1 Introducción Comprender de forma general el método 2.5.2 Gramáticas LR(1) Comprender cuales son las gramáticas LR(1) 2.5.3 Algoritmo de análisis sintáctico LR(1) Aprender a utilizar y construir analizadores ascendentes 2.5.4 Conjunto cerradura Conocer como se construye el conjunto cerradura 2.5.5. Función de transición Aprender a utilizar función de transición 2.5.6 Construcción del conjunto de elementos Ser capaz de construir el conjunto de elementos 2.5.7 Construcción de tablas de análisis sintáctico ascendente Tener la capacidad de diseñar tablas de análisis sintáctico descendente. 2.6 Gramáticas ambiguas 2 hrs Aprender a analizar gramáticas ambiguas. 2.6.1Precedencia y asociatividad Comprender las ideas de precedencia y asociatividad 2.6.2 La ambigüedad del else Conocer el caso del else ambiguo 2.6.3 Construcción de analizadores LR para gramáticas ambiguas Comprender como se realiza la construcción de tablas para este método		

2.7 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.		

Actividades del docente	Actividades del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado
Explica el las características de los problemas de programación lineal (proporcionalidad y aditividad). Explica los componentes o partes	Elabora un reporte sobre el método de Gauss-Jordan.	Reporte elaborado.	Diapositivas opcionales.	1

de un modelo de programación lineal. Explica en qué consiste el método de gauss-jordan.				
Explica cómo se plantea un problema de programación lineal.	Plantea los problemas indicados por el profesor.	Tarea (reporte) donde realice el de planteamiento de problemas.	Diapositivas opcionales.	1

Unidad temática 3: Definicion Dirigida por Sintaxis

Objetivo de la unidad temática: Conocer cómo se puede agregar atributos a las gramáticas**.**

.

Introducción: En esta unidad, se resolverán problemas de programación .

Contenido temático	Saberes involucrados	Producto de la unidad temática
3.1 Gramáticas con atributos 1 hrs Aprender como se puede aumentar la capacidad expresiva una gramática, mediante el uso de atributos. 3.1.1 Atributos sintetizados Comprender como funcionan los atributos sintetizados. 3.1.2 Atributos heredados Comprender como funcionan los atributos heredados. 3.2 Construcción de Árboles sintácticos 7 hrs Aprender a construir arboles sintácticos utilizando los analizadores sintácticos descendentes y ascendentes. 3.1.1 Arboles sintácticos Conocer cómo se puede representar un árbol sintáctico utilizando programación orientada a objetos. 3.1.2 Construcción de árboles sintácticos para expresiones Implementar árboles sintácticos para expresiones. 3.1.3 Construcción de árboles sintácticos utilizando analizador sintáctico descendente Aprender a construir árboles de manera automática durante el análisis sintáctico descendente. 3.1.4 Construcción de árboles sintácticos utilizando analizador sintáctico ascendente Aprender a construir árboles de manera automática durante el análisis sintáctico ascendente.	La tarea esencial de un analizador es determinar si una determinada entrada puede ser derivada desde el símbolo inicial, usando las reglas de una gramática formal, y como hacer esto, existen esencialmente dos formas: • Analizador sintáctico descendente (*Top-Down-Parser*): ..un analizador puede empezar con el símbolo inicial e intentar transformarlo en la entrada, intuitivamente esto sería ir dividiendo la entrada progresivamente en partes cada vez más pequeñas, de esta forma funcionan los analizadores LL, un ejemplo es el javaCC. Una mejora en estos parsers se puede logar usando GLR (Generalized Left-to-right Rightmost derivation). • Analizador sintáctico ascendente (*Bottom-Up-Parser*): un analizador puede empezar con la entrada e intentar llegar hasta el símbolo inicial, intuitivamente el analizador intenta encontrar los símbolos más pequeños y progresivamente construir la jerarquía de símbolos hasta el inicial, los analizadores LR funcionan así y un ejemplo es el Yacc. También existen SLR (Simple LR) o los LALR (Look-ahead LR) como también de los GLL[7] (Generalized Left-to-right Leftmost derivation).	Documento donde resuelva fase del compilador. Documento donde resuelva problemas con el lenguaj de su eleccion.

Actividades del docente	Actividades del estudiante	Evidencia o de la actividad	Recursos y materiales	Tiempo destinado

Explica cómo se resuelve la fase del analisi sintactico usando tablas de transiciones, diagramas de transiciones y arboles de derivaciongráfico.	Resuelve con el método indicado por el profesor.	Tarea (reporte) donde resuelve la fse del compilador.	Diapositivas opcionales.	1

Unidad temática 4: Analisis Semantico

Objetivo de la unidad temática: Aprender a realizar la verificación de tipos utilizando los árboles sintácticos y la tabla de símbolos.
.

Introducción: La semántica descriptiva se dedica al estudio de lo que significan signos concretos dentro de una lengua concreta

Contenido temático	Saberes involucrados	Producto de la unidad temática
4hrs 4.1 Comprobación de tipos 2hrs Aprender a definir formalmente las expresiones validas del lenguaje. Comprender como realizar la comprobación de expresiones del lenguaje 4.1.1 Expresiones de tipos Aprender a definir formalmente las expresiones válidas del lenguaje. 4.1.2 Sistema de tipos Implementar a definir las expresiones de tipos aceptadas por el lenguaje para construir el sistema de tipos. 4.1.3 Conversión de tipos Aprender a definir los tipos que pueden ser convertidos de manera automática. 4.1.4 Comprobación estática y dinámica Aprender las diferencias entre los tipos de comprobación. 4.1.5 Comprobación de tipos en expresiones Aprender a definir expresiones de tipos para expresiones del lenguaje. 4.1.6 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	Convierte un problema de programación lineal a la forma estándar. Modifica la función objetivo para resolver con el método de la con el lenguaje de su eleccion 2 fases. Elabora la tabla para resolver de manera tabular. Realiza la prueba de optimalidad para determinar la variable de entrada. Realiza la prueba de factibilidad Conoce el método de gauss-jordan. Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo Escucha la opinión de sus compañeros y expresa la suya con apertura Muestra seguridad al hablar y transmitir mensajes	Documento donde resuelva problemas con el método de las dos fases .

datos de las variables, durante la compilación. 4.2.1 Estructura de la tabla de símbolos Conocer la estructura que se utiliza para implementar una tabla de símbolos. 4.2.2 Declaraciones de variables, procedimientos y funciones Conocer como se almacenan las declaraciones de variables, procedimientos y funciones en la tabla de símbolos. 4.2.3 Reglas de ámbito y estructuras de bloques Comprender los conceptos de ámbito y estructuras de bloques.		

Actividades del docente	Actividades del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado
Explica cómo se modifica el problema, y resuelve un problema de maximización y un problema de minimización con el método de las 2 fases.	Resuelve con el método de las 2 fases los problemas indicados por el profesor.	Tarea (reporte) donde resuelve problemas con el método de las 2 fases.	Diapositivas opcionales.	4

Unidad temática 5: Generacion de Codigo Intermedio

Objetivo de la unidad temática: Aprender a traducir lenguajes a código intermedio.

Introducción: En esta unidad, se planteará el problema dual a partir de un problema de programación lineal, y se resolverá con el método simplex dual.

Contenido temático	Saberes involucrados	Producto de la unidad temática
6hrs 5.1 Código de tres direcciones 6hrs Conocer la sintaxis del código de tres direcciones. 5.1.1 Traducción de expresiones Aprender a traducir expresiones de un lenguaje de alto nivel a código intermedio. 5.1.2 Traducción de referencias a arreglos Aprender a traducir referencias a arreglos de un lenguaje de alto nivel a código intermedio. 5.1.3 Traducción de expresiones lógicas Aprender a traducir expresiones lógicas de un lenguaje de alto nivel a código intermedio. 5.1.4 Código de corto circuito Aprender implementar código de corto circuito en el código intermedio.	Plantea un problema de programación en bajo nivel como problema dual. Determina el número de variables de decisión del modelo dual. Determina el número de restricciones del problema dual. Resuelve un problema de programación a bajo nivel utilizando lenguaje ensamblador Interpreta los resultados obtenidos	Documento donde plantea el problema dual y resuelve problemas con el método que se elija. .

5.1.5 Instrucciones para controlar el flujo del programa Aprender a controlar el flujo de programas en código intermedio 5.1.6 Procedimientos Aprender a definir procedimientos en código intermedio		

Actividades del docente	Actividad del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado
Explica cómo es la programacion en bajo nivel de la esta etapa el compilador	Plantea el problema dual y resuelve con la version de ensamblador que elija.	Tarea (reporte) donde plantea el problema dual y resuelve la fase.	Diapositivas opcionales.	2 semanas

Unidad temática 6: Generacion de Codigo Objeto

Objetivo de la unidad temática: Aprender a implementar programas en código objeto, para después aprender a traducir código de un lenguaje de alto nivel a código objeto.

Introducción: En esta unidad, podrá identificar las modificaciones que pueden tener los valores de una fase resuelto de programación y realizará las operaciones para obtener el resultado.

Contenido temático	Saberes involucrados	Producto de la unidad temática
10 hrs 6.1 Introducción 0.5 hrs Conocer los conceptos básicos 6.1.1 Maquina objeto Conocer los conceptos básicos de la maquina objeto. 6.1.2 Instrucciones de la maquina objeto Conocer el conjunto de instrucciones disponibles de la maquina objeto. 6.1.3 Modos de direccionamiento Comprender los métodos de direccionamiento existentes. 6.2 Código ensamblador 7.5 hrs Aprender a escribir programas en el lenguaje ensamblador 6.2.1 Asignación de valores Aprender a asignar valores a variables 6.2.2 Expresiones aritméticas Aprender a realizar expresiones aritméticas 6.2.3 Control de flujo Aprender a utilizar las instrucciones para controlar el flujo del programa 6.2.4 Procedimientos y funciones	Identifica la importancia de realizar un análisis de sensibilidad. Identifica los parámetros necesarios para realizar un análisis de sensibilidad. Utiliza la tabla de simbolos para determinar los cálculos a realizar la fase. Identifica los cambios que pueden presentarse y realiza los fases correspondientes.	Documento donde resuelve problemas de análisis de sensibilidad. .

Aprender a definir y utilizar procedimientos y funciones. 6.3 Generación de código objeto a partir de árboles sintácticos 2 hrs Aprender a generar código objeto de manera automática utilizando árboles sintácticos.		

Actividades del docente	Actividad del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado
Explica los procedimientos para hacer un análisis de sensibilidad.	Resuelve diversos problemas de análisis de sensibilidad para los distintos cambios que pueden presentarse.	Tarea (reporte) donde resuelve problemas de análisis de sensibilidad.	Diapositivas opcionales.	3 semanas

Unidad temática 7: Optimizacion

Objetivo de la unidad temática: Aprender a realizar optimizaciones al código generado por el compila or

Introducción: En esta unidad, se resolverán problemas que implican llevar mercancías de n proveedores a m puntos de consumo y también problemas de asignación uno a uno.

Contenido temático	Saberes involucrados	Producto de la unidad temática
8 hrs 7.1 Optimización de mirilla 4 hrs Conocer las optimizaciones de mirilla. 7.1.1 Eliminación de instrucciones redundantes Aprender a eliminar instrucciones que son redundantes en el código generado. 7.1.2 Eliminación de código inalcanzable Aprender a eliminar código inalcanzable en el código generado. 7.1.3 Optimizaciones de flujo de control Aprender a optimizar las instrucciones de control de flujo 7.1.4 Simplificación algebraica y reducción por fuerza Aprender a reducir expresiones algebraicas 7.2 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. 7.1.1 Números de Ershov Conocer cuáles son los números Ershov y para que se pueden utilizar. 7.1.2 Generación de código a partir de árboles de expresión etiquetados Aprender a generar código para expresiones a partir de árboles etiquetados 7.1.3 Generación de código a partir de árboles de expresión	Identifica las características de los problemas de transporte y asignación. Plantea los problemas de transporte y asignación como problemas de programación lineal. Resuelve los problemas de transporte utilizando el método de la Esquina noroeste, método Vogel y método Russel. Aplica la prueba de optimalidad a las soluciones obtenidas para el problema de transporte y si no es óptimo, optimiza. Resuelve problemas de asignación uno a uno con el método húngaro.	Documento donde resuelve problemas de transporte y asignación. .

etiquetados con un número insuficiente de registros Aprender a generar código para expresiones optimizado con un número insuficiente de registros.		

Actividades del docente	Actividad del estudiante	Evidencia de la actividad	Recursos y materiales	Tiempo destinado
Explica los procedimientos para realizar un compilador	Resuelve las etapas del copilador	Tarea (reporte) donde resuelve cada una de las fases del copilador	Diapositivas opcionales.	6

5. EVALUACIÓN Y CALIFICACIÓN

Requerimientos de acreditación:

Para aprobar la Unidad de Aprendizaje el estudiante requiere una calificación mínima de 60.

Se aplicará lo establecido en el REGLAMENTO GENERAL DE EVALUACIÓN Y PROMOCIÓN DE ALUMNOS DE LA UNIVERSIDAD DE GUADALAJARA en especial los artículos siguientes:

Artículo 20. Para que el alumno tenga derecho al registro del resultado final de la evaluación en el periodo ordinario, establecido en el calendario escolar aprobado por el H. Consejo General Universitario, se requiere:

I. Estar inscrito en el plan de estudios y curso correspondiente, y

II. Tener un mínimo de asistencia del 80% a clases y actividades registradas durante el curso.

Criterios generales de evaluación:

A lo largo de la UA se elaborarán diversos reportes e informes por escrito, que deberán seguir los siguientes lineamientos básicos (más los específicos de cada trabajo):

- Entrega en tiempo
- Diseño de portada con datos de la Unidad de Aprendizaje, alumno, profesor y fecha
- El desarrollo del tema se acompañará siempre de una conclusión que rescate los principales aprendizajes. Todas las conclusiones se sustentarán en datos.
- Todas las referencias se citarán adecuadamente conforme al criterio APA
- Queda estrictamente prohibido el plagio

Evidencias o Productos

Evidencia o producto	Competencias y saberes involucrados	Contenidos temáticos	Ponderación
[Rescatar las evidencias o productos de las unidades temáticas]			**%**
Exámenes parciales	Identifica y organiza la información que se requiere para realizar un compilador Discrimina y analiza información relevante	Planteamiento de comoilador, solución de problemas con el lenguaje de programacion abierto	**20 %**
Entrega de tareas con fases resueltas del compilador	Identifica y organiza la información que se requiere para realizar un comilador Presenta sus productos en tiempo y forma, de tal manera que demuestra interés y cuidado en su trabajo	Planteamiento del problema de copiladores , solución de problemas con el lenguaje abierto	**80 %**

Producto final

Descripción	Evaluación	
Título: Entrega de un compilador	**Solo es requisitos**	**Ponderación**
Objetivo: N/A	**Criterios de forma:**	

Caracterización: N/A		

Otros criterios		
Criterio	**Descripción**	**Ponderación**
Examenes Parciales	Elaboracion de 2 examenes parciales	20%
1 Fase	Entrega de Analisis Lexico	20%
2 Fase	Entrega de Analisis Sintactico	20%
3 Fase	Entrega de Analisis Semantico	20%
4 Fase	Entrega de Generacion de Codigo	20%

6. REFERENCIAS Y APOYOS

Referencias bibliográficas

Referencias básicas

Autor (Apellido, Nombre)	Año	Título	Editorial	Enlace o bibliotecar virtual donde esté disponible (en su caso)
Aho, A.V.; Lam, M.S.; Sethi, R.; Ullman	2007.	**Compilers: principles, techniques, and tools**	ddison-Wesley, 2007. ISBN: 9780321491695	http://cataleg.upc.edu/record=b1315390~S1*cat
Parr, T,	2012.	The Pragmatic Programmers, 2012	**The definitive ANTLR 4 reference**	http://cataleg.upc.edu/record=b1440020~S1*cat
Cortadella, J	2015	**Compilers**	**lecture notes** -	http://www.lsi.upc.edu/~jordicf/Teaching/CL/index.html
Referencias complementarias				
Appel, A.W.; Ginsburg, M,	**2004**	**Modern compiler implementation in C**	Cambridge University Press , 2004. ISBN: 0521607655	http://cataleg.upc.edu/record=b1359035~S1*cat
Appel, A.W.; Palsberg, J,	**2002**	**Modern compiler implementation in Java**	Cambridge University Press , 2002. ISBN: 052182060X	http://cataleg.upc.edu/record=b1339772~S1*cat

Apoyos (videos, presentaciones, bibliografía recomendada para el estudiante)

Unidad temática 1: www.youtube.com/watch?v=Zd5k9Wmc_us

.

Unidad temática 2: www.youtube.com/watch?v=3RIeADfiE54

Unidad temática 3: www.youtube.com/watch?v=1K4kP_iACY8

Unidad temática 4: www.youtube.com/watch?v=ges62ZNrJl4

Unidad temática 5www.youtube.com/watch?v=wwffrQM90rA

Unidad temática 6: https://www.youtube.com/watch?v=PDZSTlciv6c

.,

Unidad temática 7: https://www.youtube.com/watch?v=NS2aXgnPKZ0