PROGRAMA SINÓPTICO POR COMPETENCIAS

I. DATOS DE IDENTIFICACIÓN

PROGRAMA ACADÉMICO:	Ingeniería en Sistemas Computacionales		
NOMBRE:	Lenguajes y Autómatas II.		**CLAVE:** SCC-1016
TIPO DE CURSO:	Obligatorio/Opcional		
HORAS: (T.P.C.)	**TEÓRICAS:** 2	**PRÁCTICAS:** 3	**CRÉDITOS ACADÉMICOS:** 5
SEMESTRE:	Séptimo (7$^{ro.}$)		
FECHA DE ELABORACIÓN:	13 de febrero de 2014		
ELABORADO POR:	SNIT		

II. COMPETENCIAS A DESARROLLAR:

Implementa un compilador para un lenguaje específico considerando las etapas del mismo.

III.CONTENIDOS:

UNIDAD I: Análisis semántico.	
COMPETENCIA ESPECÍFICA DE LA UNIDAD: Diseña mediante el uso de reglas semánticas dirigidas por sintaxis, un analizador semántico para un compilador.	**CONTENIDO:** Árboles de expresiones, Acciones semánticas de un analizador Sintáctico, Comprobaciones de tipos en expresiones, Pila semántica en un analizador sintáctico, Esquema de traducción, Generación de la tabla de símbolo y tabla de direcciones, Manejo de errores semánticos.

UNIDAD II: Generación de código intermedio.	
COMPETENCIA ESPECÍFICA DE LA UNIDAD: Realiza una aplicación dando solución a un problema del entorno usando el paradigma de la programación funcional.	**CONTENIDO:** Notaciones: Prefija, Infija, Postfija, Representaciones de código Intermedio: Notación Polaca, Código P, Triplos, Cuádruplos, Esquema de generación: Variables y constantes, Expresiones, Instrucción de asignación, Instrucciones de control, Funciones, Estructuras.

UNIDAD III: Optimización	
COMPETENCIA ESPECÍFICA DE LA UNIDAD: Conoce e identifica los diferentes tipos de optimización que permita eficientar el código intermedio.	**CONTENIDO:** Tipos de optimización: Locales, Ciclos, Globales, De mirilla, Costo: Costo de ejecución. (memoria, registros, pilas), Criterios para mejorar el código, Herramientas para el análisis del flujo de datos.

UNIDAD IV: Generación de código objeto	
COMPETENCIA ESPECÍFICA DE LA UNIDAD: Utiliza un lenguaje de bajo nivel para traducir el código construido a lenguaje máquina para su ejecución.	**CONTENIDO:** Registros, Lenguaje ensamblador, Lenguaje máquina, Administración de memoria.

IV. FORMA DE EVALUACIÓN:

Para evaluar las actividades de aprendizaje se recomienda solicitar: mapas conceptuales, reportes de prácticas, estudios de casos, exposiciones en clase, ensayos, problemarios, reportes de visitas, portafolio de evidencias y cuestionarios, cuadro sinóptico.

Para verificar el nivel del logro de las competencias del estudiante se recomienda utilizar: listas de cotejo, listas de verificación, matrices de valoración, guías de observación, coevaluación y autoevaluación

V. REFERENCIA BIBLIOGRÁFICA:

1. Aho Alfred V., U. J. (2007). Compiladores. Principios, técnicas y herramientas (2da. ed.). México: Pearson Educación.
2. Alfonseca Moreno, M. (2006). Compiladores e intérpretes: teoría y práctica (1ra ed.). España: Pearson/Prentice Hall.
3. Carrión Viramontes, J. E. (2008). Teoría de la computación. México: Limusa.
4. Hopcroft John E., M. R. (2002). Introducción a la Teoría de Autómatas, Lenguajes y Computación (2da. ed.). Madrid: Addison-Wesley.
5. Isasi Pedro, M. P. (1997). Lenguajes, gramáticas y autómatas. Un enfoque Práctico. AddisonWesley.
6. Kelley, D. (1995). Teoría de Autómatas y Lenguajes Formales, (1ra. ed.). Madrid: Prentice Hall.
7. Lemone, K. A. (1996). Fundamentos de compiladores: cómo traducir al lenguaje de computadora. México D.F.: Compañía Editorial Continental.
8. Martin, J. (2004). Lenguajes formales y teoría de la computación. México: McGraw-Hill / Interamericana de México.
9. Ruíz, J. (2009). Compiladores-Teoría e implementación. México: Alfaomega.
10. Grune, Dick. (2007). Diseño de compiladores modernos. McGraw-Hill.