

Universidad Católica San Pablo
Facultad de Ingeniería y Computación
Escuela Profesional de
Ciencia de la Computación
SILABO



CS240S. Compiladores (Obligatorio)

2010-1

1. DATOS GENERALES

1.1 CARRERA PROFESIONAL	:	Ciencia de la Computación
1.2 ASIGNATURA	:	CS240S. Compiladores
1.3 SEMESTRE ACADÉMICO	:	8 ^{vo} Semestre.
1.4 PREREQUISITO(S)	:	CS343. Lenguajes de Programación. (7 ^{mo} Sem)
1.5 CARÁCTER	:	Obligatorio
1.6 HORAS	:	2 HT; 2 HP; 2 HL;
1.7 CRÉDITOS	:	4

2. DOCENTE

3. FUNDAMENTACIÓN DEL CURSO

Que el alumno conozca y comprenda los conceptos y principios fundamentales de la teoría de compilación para realizar la construcción de un compilador

4. SUMILLA

1. PL/Visión General de los Lenguajes de Programación. 2. PL/Introducción a la Traducción de Lenguajes. 3. PL/Sistemas de Traducción del Lenguaje. 4. Paralelismo a nivel de instrucción 5. Optimización para el paralelismo y la localidad

5. OBJETIVO GENERAL

- Conocer las técnicas básicas empleadas durante el proceso de generación intermedio, optimización y generación de código.
- Aprender a implementar pequeños compiladores.

6. CONTRIBUCIÓN A LA FORMACIÓN PROFESIONAL Y FORMACIÓN GENERAL

Esta disciplina contribuye al logro de los siguientes resultados de la carrera:

- a) Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. [Nivel Bloom: 3]
- b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. [Nivel Bloom: 4]
- j) Aplicar la base matemática, principios de algoritmos y la teoría de la Ciencia de la Computación en el modelamiento y diseño de sistemas computacionales de tal manera que demuestre comprensión de los puntos de equilibrio involucrados en la opción escogida. [Nivel Bloom: 4]

7. CONTENIDOS

UNIDAD 1: PL/Visión General de los Lenguajes de Programación.(8 horas)	
Nivel Bloom: 4	
OBJETIVO GENERAL	CONTENIDO
<ul style="list-style-type: none"> ▪ Listar la evolución de los lenguajes de programación identificando como es que su historia nos ha conducido a los paradigmas actuales. ▪ Identificar al menos una característica distintiva para cada uno de los paradigmas de programación cubiertos en esta unidad. ▪ Evaluar las ventajas y desventajas entre los diferentes paradigmas, considerando temas tales como: eficiencia de espacio, eficiencia en el tiempo (para ambas partes computadora y programador), seguridad y el poder de las expresiones. ▪ Distinguir entre la programación a menor y mayor escala. 	<ul style="list-style-type: none"> ▪ Historia de los lenguajes de programación. ▪ Breve revisión de los paradigmas de programación. ▪ Lenguajes procedurales. ▪ Lenguajes orientados a objetos. ▪ Lenguajes funcionales. ▪ Lenguajes declarativos y no algorítmicos. ▪ Lenguajes de <i>scripts</i>. ▪ Los efectos de la escalabilidad en las metodologías de programación.
Lecturas: [Louden, 2004b], [Pratt and V.Zelkowitz, 1998]	

UNIDAD 2: PL/Introducción a la Traducción de Lenguajes.(12 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	CONTENIDO
<ul style="list-style-type: none"> ▪ Comparar y contrastar modelos de ejecución interpretados y compilados, resaltando los méritos de cada uno. ▪ Describir las fases de la traducción de programas desde el código fuente hasta llegar al código ejecutable y los archivos producidos por estas fases. ▪ Explicar las diferencias entre la traducción dependiente e independiente de máquina y donde estas diferencias son evidentes en el proceso de traducción. 	<ul style="list-style-type: none"> ▪ Comparación entre intérpretes y compiladores. ▪ Fases de traducción del lenguaje (análisis léxico, análisis sintáctico, generación de código, optimización). ▪ Aspectos de traducción dependientes e independientes de la máquina.
Lecturas: [Aho et al., 2008], [Aho, 1990], [Teufel and Schmidt, 1998], [Louden, 2004a], [Appel, 2002]	

UNIDAD 3: PL/Sistemas de Traducción del Lenguaje.(24 horas)	
Nivel Bloom: 2	
OBJETIVO GENERAL	CONTENIDO
<ul style="list-style-type: none"> ▪ Describir los pasos y algoritmos usados por traductores lenguajes. ▪ Reconocer los modelos formales subyacentes tales como los autómatas finitos, autómatas de pila y su conexión con la definición del lenguaje a través de expresiones regulares y gramáticas. ▪ Discutir la efectividad de la optimización. ▪ Explicar el impacto de la facilidad de la compilación separada y la existencia de librerías de programas en el proceso de compilación. 	<ul style="list-style-type: none"> ▪ Aplicación de las expresiones regulares en analizadores léxicos. ▪ Análisis sintáctico (sintaxis concreta y abstracta, árboles de sintaxis abstracta). ▪ Aplicación de las gramáticas libres de contexto en un parseo dirigido por tablas o recursivo descendente. ▪ Administración de tablas de símbolos. ▪ Generación de código por seguimiento de un árbol. ▪ Operaciones específicas de la arquitectura: selección de instrucciones y asignación de registros. ▪ Técnicas de optimización. ▪ El uso de herramientas como soporte en el proceso de traducción y las ventajas de éste. ▪ Librerías de programas y compilación separada. ▪ Construcción de herramientas dirigidas por la sintaxis.
Lecturas: [Aho et al., 2008], [Aho, 1990], [Louden, 2004a], [Teufel and Schmidt, 1998], [A.Lemone, 1996], [Appel, 2002]	

UNIDAD 4: Paralelismo a nivel de instrucción (4 horas)	
Nivel Bloom: 2	
OBJETIVO GENERAL	CONTENIDO
<ul style="list-style-type: none"> ▪ Describir la importancia y poder de la extracción de paralelismo de las secuencias de instrucciones. ▪ Explicar los conceptos de bloques básicos y código global. ▪ Distinguir los conceptos entre canalización de instrucciones por software. 	<ul style="list-style-type: none"> ▪ Arquitectura de procesadores. ▪ Restricciones de programación de código. ▪ Programación de bloques básicos. ▪ Programación de código global. ▪ Canalización por software.
Lecturas: [Aho et al., 2008]	

UNIDAD 5: Optimización para el paralelismo y la localidad (4 horas)	
Nivel Bloom: 2	
OBJETIVO GENERAL	CONTENIDO
<ul style="list-style-type: none"> ▪ Diseñar, codificar programas para cálculos paralelos. ▪ Identificar las propiedades básicas del paralelismo. ▪ Aplicar los fundamentos del paralelismo en la programación. 	<ul style="list-style-type: none"> ▪ Conceptos básicos. ▪ Multiplicación de matrices. ▪ Espacios de iteraciones. ▪ Índices de arreglos afines. ▪ Análisis de dependencias de datos de arreglos. ▪ Búsqueda del paralelismo sin sincronización. ▪ Sincronización entre ciclos paralelos.
Lecturas: [Aho et al., 2008]	

8. METODOLOGÍA

El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.

El profesor del curso presentará demostraciones para fundamentar clases teóricas.

El profesor y los alumnos realizarán prácticas

Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. EVALUACIONES

Evaluación Permanente 1 : 20 %

Examen Parcial : 30 %

Evaluación Permanente 2 : 20 %

Examen Final : 30 %

Referencias

[Aho, 1990] Aho, A. (1990). *Compiladores Principios, técnicas y herramientas*. Addison Wesley.

[Aho et al., 2008] Aho, A., Lam, M., Sethi, R., and Ullman, J. D. (2008). *Compiladores. Principios, técnicas y herramientas*. Addison Wesley, 2nd edition. ISBN:10-970-26-1133-4.

[A.Lemone, 1996] A.Lemone, K. (1996). *Fundamentos de Compiladores*. CECSA-Mexico.

[Appel, 2002] Appel, A. W. (2002). *Modern compiler implementation in Java*. Cambridge University Press, 2.a edición edition.

[Louden, 2004a] Loudon, K. C. (2004a). *Construcción de Compiladores Principios y Practica*. Thomson.

[Louden, 2004b] Loudon, K. C. (2004b). *Lenguajes de Programacion*. Thomson.

[Pratt and V.Zelkowitz, 1998] Pratt, T. W. and V.Zelkowitz, M. (1998). *Lenguajes de Programacion Diseño e Implementacion*. Prentice-Hall Hispanoamericana S.A.

[Teufel and Schmidt, 1998] Teufel, B. and Schmidt, S. (1998). *Fundamentos de Compiladores*. Addison Wesley Iberoamericana.