PROGRAMA DE CURSO
DISEÑO E IMPLEMENTACIÓN DE COMPILADORES

A. Antecedentes generales del curso:

Departamento	Ciencias de la Computación					
Nombre del curso	Diseño e Implementación de Compiladores		Código	CC5116	Créditos	6
Nombre del curso en inglés	*Compiler Design and Implementation*					
Horas semanales	Docencia	3	Auxiliares	1,5	Trabajo personal	5,5
Carácter del curso	Obligatorio			Electivo	X	
Requisitos	CC4101: Lenguajes de Programación CC4301: Arquitectura de Computadores					

B. Propósito del curso:

El propósito de este curso es introducir el estudiante a la implementación de compiladores modernos y eficientes para lenguajes de programación de alto nivel. El curso se centra en las conexiones entre los distintos mecanismos de los lenguajes de programación y su impacto en el diseño de un compilador, incluyendo problemas algorítmicos y pragmáticos.

A lo largo del curso, el estudiante trabaja de manera recursiva, en el desarrollo de compiladores para lenguajes cada vez más complejos, desde el *parseo* hasta la generación de código *assembler* x86-64. Los mecanismos estudiados incluyen múltiples tipos de datos, interacción con C, procedimientos, gestión de la recursión, verificación e inferencia de tipos, estructuras de datos, funciones de primera clase, objetos, y gestión de la memoria. Además, se abordan temas de eficiencia de ejecución y optimizaciones.

A través del continuo trabajo de implementación, el cual se sugiere se haga en equipo, el estudiante también aprenderá lecciones importantes sobre Ingeniería de Software y *Testing*.

El curso tributa a las siguientes competencias específicas (CE) y genéricas (CG):

CE1: Analizar problemas computacionales, construir modelos, expresándolos en representaciones y lenguajes formales adecuados.

CE2: Analizar, diseñar y/o adaptar algoritmos y estructuras de datos que cumplan con las garantías requeridas de correctitud y eficiencia.

CE5: Concebir, diseñar y construir soluciones de software, siguiendo un proceso sistemático y cuantificable, acorde a los fundamentos, eligiendo el paradigma y las técnicas más adecuadas.

CE6: Desarrollar software en una amplia variedad de plataformas y lenguajes de programación.

CE7: Gestionar proyectos de diseño, desarrollo, implementación y evolución de soluciones de software, contemplando tanto los procesos involucrados como el producto esperado, su calidad y respuesta efectiva al problema que aborda.

CE8: Diagnosticar y resolver problemas en el funcionamiento de software cercano a la plataforma para mejorar su desempeño.

CG1: Comunicación académica y profesional

Comunicar en español de forma estratégica, clara y eficaz, tanto en modalidad oral como escrita, puntos de vista, propuestas de proyectos y resultados de investigación fundamentados, en situaciones de comunicación compleja, en ambientes sociales, académicos y profesionales.

CG2: Comunicación en inglés

Leer y escuchar de manera comprensiva en ingles variados tipos de textos e informaciones sobre temas concretos o abstractos, comunicando experiencias y opiniones, adecuándose a diferentes contextos de acuerdo a las características de la audiencia.

CG3: Compromiso ético

Actuar de manera responsable y honesta, dando cuenta en forma crítica de sus propias acciones y sus consecuencias, en el marco del respeto hacia la dignidad de las personas y el cuidado del medio social, cultural y natural.

CG4: Trabajo en equipo

Trabajar en equipo, de forma estratégica y colaborativa, en diversas actividades formativas, a partir de la autogestión de sí mismo y de la relación con el otro, interactuando con los demás en diversos roles: de líder, colaborador u otros, según requerimientos u objetivos del trabajo, sin discriminar por género u otra razón.

C. Resultados de aprendizaje:

Competencias específicas	Resultados de aprendizaje
CE1	RA1: Identifica y analiza las distintas fases de la compilación tales como *parsing*, verificación de tipos, gestión de la pila y del montón, optimizaciones, entre otras, y el impacto de los mecanismos del lenguaje fuente sobre estas, considerando su importancia para el diseño e implementación.
CE2	RA2: Analiza e implementa soluciones algorítmicas para distintos problemas involucrados en la compilación de lenguajes de alto nivel, considerando correctitud y eficiencia como criterios de implementación.
CE5	RA3: Diseña e implementa compiladores, en un nivel creciente de complejidad, utilizando de manera efectiva el sistema de tipo expresivo del lenguaje de implementación.
CE5, CE6	RA4: Valida implementaciones de compiladores mediante el uso de herramientas de depuración y testing sistemático, considerando la naturaleza multilenguaje de un compilador.

CE6	RA5: Maneja, con propiedad y pertinencia, un lenguaje funcional *tipado* para diseñar, implementar, y evolucionar un software complejo, considerando buenas prácticas y patrones de diseño.
CE6, CE8	RA6: Implementa componentes de soporte para la ejecución de programas compilados, tales como gestión de errores y recolector de basura, usando lenguajes de bajo nivel como C y *Assembler*, respetando convenciones y consideración de desempeño.
Competencias genéricas	Resultados de aprendizaje
CG1	RA7: Reporta, en forma escrita, los resultados asociados a diversas entregas del diseño e implementación de compiladores, en un nivel creciente de complejidad, respecto de los requerimientos, especificaciones, el código usado y la documentación respectiva de respaldo.
CG2	RA8: Lee en inglés, de manera analítica y comprensiva, textos y apuntes sobre lenguajes de programación a fin de generar nuevos conocimientos aplicables a la solución computacional de problemas de diseño e implementación de compiladores.
CG3, CG4	RA9: Trabaja con su equipo en las actividades de diseño e implementación de compiladores, considerando una buena comunicación y colaboración entre los pares para cumplir de manera responsable los compromisos y acuerdos adquiridos.

D. Unidades temáticas:

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
1	RA1, RA2, RA5, RA8	Introducción a los compiladores	2 semanas
Contenidos		**Indicador de logro**	
1.1. ¿Qué es un compilador? 1.2. OCaml. 1.3. Assembler x86.		El estudiante: 1. Identifica y analiza las distintas fases de la compilación tales como *parsing*, verificación de tipos, generación de código, optimizaciones, y ambiente de ejecución, considerando el impacto de los mecanismos del lenguaje fuente sobre estas. 2. Analiza la función de un compilador, considerando los desafíos involucrados en cada una de las etapas de su desarrollo. 3. Usa el lenguaje de programación *OCaml*, y el ecosistema de herramientas asociadas, en problemas simples de programación.	

	4. Maneja conceptos básicos del *Assembler x86* y uso de *NASM* para ejemplos básicos de programación. 5. Lee de manera comprensiva diversas fuentes en inglés sobre compilación, para aplicar estos conocimientos en el diseño de compiladores.
Bibliografía de la unidad	(1) (2) (3) capítulos de introducción (4) Lecture 1 Referencias: (5-7) (8-14)

Número	RA al que tributa	Nombre de la unidad	Duración en Semanas
2	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8	Compilador básico	2 semanas
Contenidos		Indicador de logro	
2.1. Gramática, sintaxis abstracta. 2.2. Constantes y operadores básicos. 2.3. Identificadores y su alcance. 2.4. Gestión básica de la pila. 2.5. Condicionales. 2.6. Operadores infijos.		El estudiante: 1. Describe la gramática y sintaxis abstracta de un lenguaje de programación dado. 2. Construye su primer compilador para un lenguaje con solo constantes, considerando desde el archivo fuente hasta la generación de código e impresión del resultado final. 3. Diseña e implementa, en un estado inicial, una extensión del compilador donde el lenguaje fuente soporte identificadores, utilizando la pila. 4. Diseña e implementa, en una primera aproximación, una extensión del compilador para que el lenguaje fuente soporte operadores y condicionales, considerando conceptos de orden de evaluación. 5. Evalúa la correctitud del *pipeline* del compilador para un lenguaje con solo constantes, usando herramientas de testing sistemático. 6. Planifica y entrega sus tareas, las que documenta debidamente, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 7. Lee de manera comprensiva diversos textos sobre programación y compilación, aplicando los conceptos adquiridos al diseño e implementación de compiladores básicos.	
Bibliografía de la unidad		(4) Lectures 2, 3, 4 (1) Capítulos 4, 6 (2) Capítulo 7.3	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
3	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Tipos de datos y representación	1 semana
Contenidos		**Indicador de logro**	
3.1. Booleanos. 3.2. Representaciones de tipos de datos. 3.3. Actualizaciones de operadores. 3.4. Desbordamiento de enteros.		El estudiante: 1. Diseña una estrategia de codificación binaria para soportar dos tipos de datos: enteros y booleanos. 2. Diseña e implementa una extensión del compilador para tomar en cuenta la representación de los datos. 3. Detecta y reporta casos de desbordamiento de enteros a partir de ejemplos. 4. Evalúa la correctitud del *pipeline* del compilador extendido, usando herramientas de testing sistemático. 5. Planifica y entrega sus tareas, las que documenta debidamente, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 6. Trabaja con su equipo en las actividades de diseño e implementación de compiladores, considerando una buena comunicación y colaboración entre los pares. 7. Lee de manera comprensiva diversas fuentes en inglés sobre tipos de datos y representación, aplicando los conceptos al diseño e implementación de compiladores.	
Bibliografía de la unidad		(4) Lecture 5 (2) capítulos 7.4-7.6	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
4	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Llamar funciones definidas en C	1 semana
Contenidos		**Indicador de logro**	
4.1. Convención de llamado básica. 4.2. Convención de llamado 64 bits. 4.3. Llamar cualquier función C.		El estudiante: 1. Identifica y analiza el concepto de convención de llamado requerida para llamar funciones a nivel de assembler. 2. Implementa llamados de funciones definidas en C, respetando dichas convenciones. 3. Evalúa la correctitud del *pipeline* del compilador extendido, usando herramientas de testing sistemático.	

	4. Trabaja con su equipo de manera responsable para cumplir con las actividades propuestas. 5. Planifica y entrega sus tareas, las que documenta debidamente, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 6. Lee de manera comprensiva diversas fuentes en inglés sobre las funciones definidas en C para aplicarlas a la implementación de dichas funciones.
Bibliografía de la unidad	(4) Lecture 6 (2) capítulo 7.9

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
5	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Funciones	2 semanas
Contenidos		Indicador de logro	
5.1. Definir funciones propias. 5.2. Llamados recursivos de cola.		El estudiante: 1. Agrega definiciones de funciones al lenguaje fuente. 2. Compila definiciones de funciones a código *Assembler*. 3. Describe las limitaciones de programación recursiva sin optimización de llamados de cola. 4. Implementa la optimización de llamados de cola, transformando recursión en iteración 5. Evalúa la correctitud y desempeño del *pipeline* del compilador extendido, usando herramientas de *testing* sistemático. 6. Elabora sus tareas las que documenta debidamente, basándose en sus capacidades, sin incurrir en plagio, copia o suplantación de identidad. 7. Lee de manera comprensiva diversas fuentes en inglés sobre funciones para aplicar dichos conceptos en la implementación de los compiladores.	
Bibliografía de la unidad		(4) Lectures 7, 10 (1) capítulos 7, 15.6 (2) capítulos 6, 7.9	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
6	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Tipos	1 semana
Contenidos		**Indicador de logro**	
6.1. Verificación de tipos. 6.2. Inferencia de tipos.		El estudiante: 1. Identifica y analiza el rol y los beneficios de la verificación e inferencia de tipos, considerando ejemplos atingentes. 2. Implementa un verificador y un inferenciador de tipos simples para el lenguaje fuente. 3. Evalúa la correctitud de la verificación e inferencia de tipos, usando herramientas de *testing* sistemático. 4. Planifica y entrega sus tareas debidamente documentadas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 5. Trabaja con su equipo de manera responsable para cumplir con las actividades propuestas. 6. Lee de manera comprensiva sobre tipos, considerando estos conceptos para la implementación de compiladores.	
Bibliografía de la unidad		(4) Lectures 11, 12 (1) capítulos 5, 16 (2) capítulo 4 (3) capítulo 4	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
7	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Estructuras de datos	1 semana
Contenidos		**Indicador de logro**	
6.1. Petición de memoria: el montón. 6.2. Pares. 6.3. Tuplas. 6.4. Estructuras mutables. 6.5. Estructuras recursivas.		El estudiante: 1. Diseña e implementa una extensión del compilador para soportar estructuras de datos en el lenguaje fuente, usando ***el montón*** para administrar la memoria. 2. Evalúa la correctitud del *pipeline* del compilador extendido para soportar estructuras de datos en el lenguaje fuente, usando herramientas de *testing* sistemático.	

	3. Elabora sus tareas, documentando y respaldando debidamente sus entregas. 4. Lee de manera comprensiva diversas fuentes en inglés sobre estructuras de datos y su importancia para el diseño de implementación de compiladores.
Bibliografía de la unidad	(4) Lectures 13, 14 (2) capítulo 7.7

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
8	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Funciones de primera clase	1 semana
Contenidos		Indicador de logro	
7.1. Crear closures. 7.2. Llamar closures. 7.3. Recursión.		El estudiante: 1. Determina algorítmicamente las variables libres de una función. 2. Diseña e implementa una extensión del compilador para soportar funciones de primera clase con alcance léxico, posiblemente recursivas. 3. Evalúa la correctitud del *pipeline* del compilador extendido para soportar funciones de primera clase con alcance léxico, usando herramientas de testing sistemático. 4. Cumple obligaciones y acuerdos, respetando los compromisos adquiridos en sus actividades académicas. 5. Planifica y entrega sus tareas debidamente documentadas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad. 6. Lee de manera comprensiva diversas fuentes en inglés sobre programación y compilación, determinando sus ideas principales.	
Bibliografía de la unidad		(4) Lecture 15 (1) capítulo 15	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
9	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Gestión de memoria	1 semana
Contenidos		**Indicador de logro**	
9.1. Gestión automática de memoria. 9.2. Recolector de basura.		El estudiante: 1. Identifica y analiza distintos algoritmos de recolección de basura. 2. Diseña e implementa un recolector de basura simple, y lo integra a su compilador 3. Evalúa la correctitud y desempeño del *pipeline* del compilador extendido con el recolector de basura, usando herramientas de *testing* sistemático. 4. Lee de manera comprensiva diversas fuentes en inglés sobre gestión de memoria, determinando sus ideas principales. 5. Planifica y entrega sus tareas debidamente documentadas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad.	
Bibliografía de la unidad		(1) capítulo 13, 21 (3) capítulo 7	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
10	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Objetos	1 semana
Contenidos		**Indicador de logro**	
10.1. Representar objetos. 10.2. Crear objetos. 10.3. Invocar métodos		El estudiante: 1. Representa objetos usando los mecanismos existentes de tuplas y funciones de primera clase 2. Compila la creación de objetos e invocación de métodos usando dicha representación 3. Evalúa la correctitud del *pipeline* del compilador extendido con objetos, usando herramientas de *testing* sistemático. 4. Lee de manera comprensiva diversas fuentes en inglés sobre gestión de memoria, determinando sus ideas principales.	

	5. Planifica y entrega sus tareas debidamente documentadas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad.
Bibliografía de la unidad	(4) Lecture 16 (1) capítulo 14

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
11	RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8, RA9	Optimizaciones	2 semanas
Contenidos		Indicador de logro	
11.1.A-Normal Form. 11.2.Asignación de registros. 11.3.Evaluación parcial. 11.4.Otros tipos de optimizaciones.		El estudiante: 1. Identifica y analiza distintas técnicas de optimización, considerando el tipo de problema a resolver. 2. Transforma algorítmicamente un programa a una representación intermedia como ANF para facilitar ciertas optimizaciones. 3. Desarrolla un algoritmo de asignación de registros, considerando las especificidades de la arquitectura x86. 4. Programa la optimización de propagación de constantes evaluando, mediante *benchmarking*, la eficiencia en la ejecución de programas compilados. 5. Lee de manera comprensiva diversas fuentes en inglés sobre gestión de memoria, determinando sus ideas principales. 6. Planifica y entrega sus tareas debidamente documentadas, basándose en sus capacidades, sin incurrir en plagio, copia, suplantación de identidad.	
Bibliografía de la unidad		(4) Lectures 4, 8 (1) capítulos 11, 17-20 (2) capítulos 8-13 (3) capítulos 9-11	

A. Estrategias de enseñanza – aprendizaje:

El curso tiene una componente muy importante de desarrollo de software en grupos de dos estudiantes.

Sobre la materia misma, el curso se distingue de enfoques "por fases" (como en las referencias 1-2-3 de la bibliografía), para adoptar un enfoque "por mecanismos" (como en la referencia 4 de la bibliografía). Esto permite a los estudiantes contar con un compilador completo y operativo, desde *parsing* hasta código ejecutable, desde temprano en el curso, y durante todo el desarrollo de éste.

Por ende, el curso se basa en distintas metodologías que incluyen principalmente:

- **Clases expositivas** en las cuales se hace uso de pizarra y programación en vivo para explicar los conceptos involucrados. Para promover el aprendizaje activo, durante las clases se presentarán mini-problemas por resolver grupalmente, y cuya solución se discutirá de manera colectiva.
- **Trabajo de proyecto**, durante todo el semestre, ya que cada grupo desarrolla su propio compilador, con enfoque en los *test*, la documentación, y la buena distribución de la carga de trabajo.

B. Estrategias de evaluación:

Hay seis entregas de compilador. Estas entregas (NC) se calculan mediante promedio simple. Este trabajo se realiza a partir de actividades grupos de dos o tres estudiantes. Estas tareas evaluarán tanto aspectos prácticos como teóricos, correspondiendo a los resultados de aprendizaje RA1, RA2, RA3, RA4, RA5, RA6, RA7, RA8 y RA9.

Se realizarán además 5 *quizzes* (individuales) sobre conceptos expuestos en clases (NQ), donde cada *quiz* cuenta por igual. Estos quizzes evalúan los RA1, RA2, RA3, RA4, RA5, RA6.

Tanto las entregas como los quizzes cubren todos los resultados de aprendizaje.

La nota final (NF) del curso se calcula de la siguiente manera:

*NF = 0.8 * NC + 0.2 * NQ*

C. Recursos bibliográficos:

Bibliografía obligatoria:

(1) Andrew Appel (1998). *Modern Compiler Implementation in ML*, Cambridge University Press.
(2) Keith D. Cooper and Linda Torczon. (2004). *Engineering a Compiler, 2nd Ed.*, Morgan Kaufmann.
(3) A. Aho, M. Lam, R. Sethi, J. Ullman (2006). *Compilers: Principles, Techniques, and Tools*, Pearson Ed., 2006.
(4) Ben Lerner, *Compiler Design*, Lecture notes, Northeastern University, Boston, USA, 2020.

Recursos OCaml:

(5) OCaml home page and user manual
(6) *Introduction to Objective Caml*, Jason Hickey (online version)
(7) Some OCaml tutorials.

Recursos X86:

(8) x86 interpreter
(9) UVa x86 Assembly Guide
(10) Mac OS Stack Alignment
(11) C Calling Convention
(12) Wikibooks X86 Assembly
(13) X64 Cheat Sheet (Brown University)
(14) A tutorial on NASM

D. Datos generales sobre elaboración y vigencia del programa de curso:

Vigencia desde:	Primavera 2020
Elaborado por:	Éric Tanter
Validado por:	Sergio Ochoa, CTD de Ciencias de la Computación
Revisado por:	Área de Gestión Curricular