**DISCIPLINA:** Compiladores

<table>
<tr><td colspan="4">Eixo: Fundamentos de Engenharia de Computação</td><td>Período: 8º</td><td>Característica: Não Equalizada/Criada para o curso</td></tr>
<tr><td colspan="4">CARGA HORÁRIA</td><td>NATUREZA</td><td>ÁREA DE FORMAÇÃO DCN</td></tr>
<tr><td colspan="3">HORAS-AULA</td><td rowspan="2">HORAS</td><td rowspan="3">Teórica/Obrigatória</td><td>Profissional</td></tr>
<tr><td>TEORIA</td><td>PRÁTICA</td><td>TOTAL</td><td></td></tr>
<tr><td>60</td><td></td><td>60</td><td>50 h</td><td></td></tr>
<tr><td colspan="3">PRÉ-REQUISITOS</td><td colspan="3">CO-REQUISITOS</td></tr>
<tr><td colspan="3">Teoria da Computação</td><td colspan="3">Não há</td></tr>
<tr><td colspan="6">Ementa:<br>Introdução: A estrutura dos compiladores modernos: front-end, middle-end, back-end. Compiladores de um, dois e três passos.<br>Análise léxica: Operações com expressões regulares. Reconhecimento de linguagens regulares com autômatos finitos. Construção de autômatos finitos deterministas a partir de expressões regulares. Geradores de varredores léxicos.<br>Análise sintática: Sintaxe livre de contexto. Formas de derivação de strings e a árvore de sintaxe concreta. Precedência em expressões aritméticas. Eliminação de ambiguidade e de recursão à esquerda. Gramáticas LL(1) e LR(1). Derivação top-down. Derivação preditiva: fatoração à esquerda. Derivação recursiva: descendente e por tabelas de derivação. Recuperação de erros: o conjunto SYNCH. Gramáticas LL(K). Derivação bottom-up. Formas sentencias à esquerda e definição de manipuladores. Implementação por pilha: derivadores shift-reduce. Gramáticas LR(K). Construção de tabelas LR(0), SLR(1), LR(1), LALR(1).<br>Análise semântica: Problemas sensíveis ao contexto. Ações semânticas em derivadores LL e LR. Gramáticas de atributos. Grafo de dependência de atributos. Estrutura e organização de tabelas de símbolos. Aninhamento léxico e regras de escopo. Descritores de tipos: formas de compatibilidade. Verificação e conversão de tipos em expressões. L-values e R-values. Representação intermediária para análise semântica: árvore de sintaxe abstrata.<br>Ambientes de execução: Classes de armazenamento e acesso a dados não locais. Registros de ativação. Funções de mais alta ordem . Pilha de execução: criação e manipulação de registros de ativação.<br>Geração de representação intermediária: Tipos de representação intermediária: árvores de sintaxe abstrata, grafo acíclico direcionado, grafo de controle do fluxo, código de três endereços. Regras semânticas para geração de código intermediário: atribuição e expressões, desvio de controle, declarações. Tradução em árvores de sintaxe abstrata. Reorganização do código intermediário: árvores canônicas, blocos básicos, aglomerados sequenciais.</td></tr>
</table>

Geração de código de máquina para MIPS ou PENTIUM: Seleção de instruções. Análise de tempo de vida: grafos de fluxo do controle, grafos de interferência. Alocação de registradores: coloração de grafos, coalescência. Exemplo de otimização de laços.

**Bibliografia Básica**

-AHO, Alfred V.; LAM, Monica S.; SETHI, Ravi; ULLMAN, Jeffrey D. Compiladores: princípios, técnicas e ferramentas. 2. ed. São Paulo. 2008.
-APPEL, Andrew W. Modern compiler implementation in Java. 2. ed. Estados Unidos. 2002.
-LOUDEN, Kenneth C. Compiladores: princípios e práticas. São Paulo. 2004.

**Bibliografia Complementar**

-DELAMARO, Márcio Eduardo. Como Construir um Compilador – Utilizando Ferramentas Java. 2004.
-FISCHER, Charles N.; CYTRON, Ron K.; LeBLANC Jr., Richard J. Crafting a Compiler. Estados Unidos. 2009.
-MUCHNICK, Steven S. Advanced Compiler Design and Implementation. San Francisco. 1997.
-PRICE, Ana Maria de Alencar; TOSCANI, Simao Sirineto. Implementação de Linguagens de Programação: Compiladores. Série Livros Didáticos. Número 9. Porto Alegre. 2008.
-SETZER, Valdemar Waingort. A Construção de um Compilador. Rio de Janeiro. 1989.