

Campus: São José dos Campos		
Curso (s): Engenharia de Computação e Ciência da Computação		
Unidade Curricular (UC): Compiladores		
Unidade Curricular (UC): <i>Compilers</i>		
Código da UC: 2615		
Docente Responsável: Luiz Eduardo Galvão Martins		Contato (e-mail): legmartins@unifesp.br
Docente (s) Colaborador/a (es/as):		Contato (e-mail): [opcional]
Ano letivo: 2022	Termo: 6º	Turma (s): Noturno
Nome do Grupo/Módulo/Eixo da UC (se houver): Unidade Curricular Fixa		Idioma predominante em que a UC será oferecida: ( X ) Português ( ) English ( ) Español ( ) Français ( ) Libras ( ) Outro:
UC: ( X ) Fixa ( ) Eletiva ( ) Optativa	Oferecida como: ( X ) Disciplina ( ) Módulo ( ) Estágio ( ) Outro:	Oferta da UC: ( X ) Semestral ( ) Anual
Ambiente Virtual de Aprendizagem: ( X ) Moodle ( ) Classroom ( ) Outro: ( ) Não se aplica		
Pré-Requisito (s) - Indicar Código e Nome (s) da (s) UC: 2616 - Linguagens Formais e Autômatos		
Carga horária total (em horas): 72h		
Carga horária teórica (em horas): 36h	Carga horária prática (em horas): 36h	Carga horária de extensão (em horas, se houver):
Se houver atividades de extensão, indicar código e nome do projeto ou programa vinculado na Pró-Reitoria de Extensão e Cultura (ProEC):		
Ementa: Sistema de Varredura - Análise Léxica; Gerador de Analisador Léxico; Análise Sintática Descendente; Análise Sintática Ascendente; Gerador de Analisador Sintático; Análise Semântica; Geração de Código; Otimização de Código.		
Conteúdo programático: Introdução: Importância dos compiladores; histórico e evolução; visão geral do processo de compilação. Partida rápida e transposição. Análise Léxica: Uso de expressões regulares para descrição de padrões de <i>tokens</i> ; algoritmo de Thompson; construção de subconjuntos; otimização dos autômatos finitos determinísticos; transformação de autômatos finitos em programas de reconhecimento de cadeias; gerador de analisadores léxicos (Flex). Análise Sintática: Árvores sintáticas; análise sintática descendente; análise sintática ascendente; gerador de analisadores sintáticos (YACC-Bison). Análise Semântica: Algoritmos para computação de atributos; tabela de símbolos; tipos de dados e verificação de tipos. Geração de Código: Código intermediário (código de três endereços); geração de código para referências e estruturas de dados; geração de código para declarações de controle e expressões lógicas; geração de código para chamadas de procedimentos e funções. Otimização de Código: Escolha de modos de endereçamento. Substituição de instruções. Eliminação de operações redundantes.		
Objetivos: <u> Gerais:</u>		

A disciplina tem o objetivo de apresentar os conceitos fundamentais sobre compiladores, por meio de abordagem teórica e prática.

Específicos:

Apresentar aos alunos técnicas consolidadas de projeto e construção de compiladores; Capacitar os alunos para a especificação e utilização de gramáticas usadas na construção de compiladores; Habilitar os alunos a compreender as fases de análise léxica, sintática e semântica; Capacitar os alunos para o uso de geradores automáticos de analisadores léxicos e sintáticos; Apresentar aos alunos uma visão geral do processo de síntese realizado por um compilador; Proporcionar aos alunos a experiência de projetar e construir um compilador.

Metodologia de ensino:

Aulas expositivas com auxílio de quadro branco e projetor multimídia, intercaladas com aulas de exercícios e laboratório, participação dos alunos de forma oral e escrita. Vídeo aulas. Desenvolvimento de um compilador.

Avaliação:

Serão adotados os seguintes instrumentos de avaliação:

- 1 prova escrita (peso 4).
- 1 projeto da disciplina – trabalho final (peso 6): o projeto será desenvolvido individualmente ou em duplas, dependendo do tamanho da turma. A avaliação do projeto será feita mediante a apresentação do projeto para o professor. O projeto consistirá no desenvolvimento das fases de análise de um compilador.

A nota final será calculada da seguinte forma:

$$NF = (4Prova + 6Projeto) / 10$$

A promoção do aluno na UC obedecerá aos critérios estabelecidos pela Pró-Reitoria de Graduação, tal como discutido no projeto pedagógico do curso.

Bibliografia:

Básica:

1. Louden, Kenneth C; Silva, Flávio S.C. Compiladores: princípios e práticas. São Paulo: Thomson, 2004. 569 p. ISBN 978-85-221-0422-2.
2. Aho, Alfred V et al. Compiladores: princípios, técnicas e ferramentas. 2a ed. São Paulo: Person Addison Wesley, 2007. 634 p. ISBN 978-85-88639-24-9. tradução de "Compilers: principles, techniques, and tools.
3. Appel, Andrew W; Palsberg, Jens. Modern compiler implementation in Java. 2 ed. New York: Cambridge at the University Press, 2002. 501 p. ISBN 978-0-521-82060-8.
4. Ricarte, I. Introdução à Compilação. Editora Elsevier/Campus, 2008.

Complementar:

1. Scott, Michael L. Programming language pragmatics. New York: Morgan Kaufmann, c2009. 910 p. ISBN 978-0-12-374514-9.
2. Hopcroft, John E; Motwani, Rajeev; Ullman, Jeffrey D. Introdução à teoria de autômatos, linguagens e computação. [Introduction to automata theory, languages, and computation.]. Rio de Janeiro: Campus, 2002. 560 p. ISBN 978-85-352-1072-9.
3. Ullman, Jeffrey D; Motwani, Rajeev; Hopcroft, John E. Introduction to automata theory, languages, and computation. 3.ed. Boston (USA): Pearson, 2006. 535 p. ISBN 978-0-321-45536-9.
4. Price, Ana Maria de Alencar; Toscani, Simão Sirineo. Implementação de linguagens de programação: compiladores. 3ª ed. Porto Alegre: Bookman, 2008. 9. 195 p. ISBN 978-85-7780-348-4.