|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UNIDADE -** CULDADE DE COMPUTAÇÃO E INFORMÁTICA | | | | |
| **CURSO –** CIÊNCIA DA COMPUTAÇÃO | | | | |
| **DISCIPLINA – COMPILADORES** | | | | **CÓDIGO DA DISCIPLINA** ENEX01047 |
| **PROFESSOR(ES)**  FABIO APARECIDO GAMARRA LUBACHESKI  ROBERTO CASSIO DE ARAUJO | | **DRT**  1146330  1121945 | | **ETAPA**  6º |
| **CARGA HORÁRIA**  4h/a (2 teoria | 0 laboratório | 2 EAD) | | | | **SEMESTRE LETIVO**  2020/2 |
| **EMENTA**  Estudo das fases dos processos de compilação e linguagens. Estudo dos esquemas de análise léxica,  análise sintática, análise semântica, geração de código intermediário, alocação de registradores e  geração de código Assembly. Estudo comparativo de ferramentas de geração de compiladores. | | | | |
| **OBJETIVOS** | | | | |
| **FATOS E CONCEITOS** | **PROCEDIMENTOS E HABILIDADES** | | **ATITUDES, NORMAS E VALORES** | |
| - Apresentar os conceitos e  algoritmos fundamentais para  construção de compiladores de  linguagens.  - Utilizar ferramentas  automáticas para geração de  analisadores.  - Construir um compilador  completo para uma pequena  linguagem livre de contexto. | - Descrever a organização  arquitetural dos compiladores e  seu funcionamento.  - Descrever as técnicas de  correção de erros utilizadas nos  compiladores.  - Identificar as formas de geração  e de representação de código  intermediário.  - Identificar, avaliar e utilizar  ferramentas de apoio na  construção de compiladores. | | - Compreender os aspectos  ligados ao projeto de linguagens  de programação.  - Compreender e implementar  os principais algoritmos de  análise léxica.  - Compreender e implementar  os principais algoritmos de  análise sintática.  - Compreender e implementar  os processos de análise  semântica.  - Compreender as técnicas de  otimização de código. | |
| **CONTEÚDO PROGRAMÁTICO**  1. Fundamentos do processo de compilação  1.1 Noções elementares de linguagens formais  1.2. Compilação x interpretação  1.3. Módulos e interfaces de um compilador  1.4. Ferramentas de geração automática de analisadores  2. Análise léxica  2.1. Tokens léxicos  2.2. Expressões regulares  2.3. Autômatos finitos  2.4. Geradores de analisadores léxicos  3. Análise sintática  3.1. Gramáticas livres de contexto  3.2. Análise preditiva  3.3. Análises LL e LR  3.4. Geradores de analisadores sintáticos  4. Sintaxe abstrata  4.1. Ações semânticas  4.2. Árvores de sintaxe abstrata (AST)  4.3. Percursos em AST e o padrão Visitor  5. Análise semântica  5.1. Tabelas de símbolos  5.2. Verificação de tipos  6. Registros de Ativação  6.1. Organização de programas em tempo de execução  6.2. Stack frames  7. Geração de código intermediário  7.1. Máquinas virtuais  7.2. Árvores de representação intermediárias  7.3. Geração de árvores de representação intermediárias  7.4. Árvores canônicas  8. Geração de código Assembly  8.1. Seleção de instruções  8.2. Análise de sobrevivência  8.3. Alocação de registradores  8.4. Padrões de geração de código | | | | |
| **METODOLOGIA**  - Aulas expositivas  - Utilização do ambiente Mackenzie Virtual  - Desenvolvimento de um compilador a partir de uma linguagem reduzida e fictícia (aulas práticas) | | | | |
| **CRITÉRIO DE AVALIAÇÃO**  **--- N1 ---**  Avaliação 1: 40% – NOTA A (TIA)  Projeto Etapa 1: 30% – NOTA B (TIA)  Projeto Etapa 2: 30% – NOTA C (TIA)  **--- N2 ---**  Avaliação 2: 40% – NOTA F (TIA)  Projeto Etapa 3: 30% – NOTA G (TIA)  Projeto Etapa 4: 30% – NOTA H (TIA)  **--- Média intermediária (MI) ---**  **MI = (N1 + N2)/2**  **CRITÉRIOS DE APROVAÇÃO**  se FREQUENCIA >= 65% e MI >= 6.0, **APROVADO**.  se FREQUENCIA >= 65% e (MI+PROVA FINAL)/2 >= 6.0, **APROVADO**.  OBS: o aluno tem o direito de fazer uma PROVA SUBSTITUTIVA para substituir uma nota de uma Avaliação ou Etapa de Projeto que não tenha feito. A PROVA SUBSTITUTIVA contém todo o conteúdo do semestre. Caso o aluno não tenha feito mais de uma Avaliação ou Etapa de Projeto, será substituída a nota de MAIOR PESO. | | | | |
| **BIBLIOGRAFIA BÁSICA**  CAMPBELL, B., IYER, S., DELIBAS, B.A. **Introduction to Compiler Construction in a Java World**. New York: Chapman and Hall/CRC, 2012.  GRUNE, D., van REEUWIJJK, K., BAL, H.E., JACOBS, C.J.H., LANGENDOEN. **Modern Compiler Design**. New York: Springer, 2012.  REIS, A.J. **Compiler Construction Using Java, JavaCC and YaCC**. New York: Wiley-IEEE, 2011. | | | | |
| **BIBLIOGRAFIA COMPLEMENTAR**  AHO, A.V., LAM, M.S., SETHI, R., ULLMAN, J.D. **Compilers: Principles, Techniques and Tools**. 2.ed. New York: Addison-Wesley, 2006.  APPEL, A.W. **Modern Compiler Implementation in Java**.2.ed. Boston: Cambridge University Press, 2002.  COOPER, K., TORCZON, L. **Engineering a Compiler**.2.ed. New York: Morgan Kaufmann, 2011.  MAK, R. **Writing Compilers and Interpreters: An Engineering Approach**. New York: Wiley, 2009.  WILHELM, R. **Compiler Design: Virtual Machines**. New York: Springer, 2011. | | | | |