Objetivos

- Noções básicas sobre o processo de compilação
- Implementação de um compilador completo para uma linguagem exemplo
- Utilização de algumas ferramentas
- Integração num único projeto de vários conhecimentos da Computação
- Compreensão mais profunda de alguns mecanismos linguísticos
- Aplicações das técnicas e ferramentas em outros contextos
- Desenvolvimento de um projeto de programação de porte significativo

Pré-requisitos

- Muito bons conhecimentos e experiência de programação em C, ou disposição para aprender
- Conceitos de linguagens de programação
- Conceitos de organização de computadores
- Estruturas de dados (pilhas, árvores, tabelas, ...)
- Técnicas de programação (recursão, modularização, ...)
- Noções de gramáticas e linguagens livres de contexto (desejável)
- Noções de expressões regulares e linguagens regulares (desejável)

Avaliação

- Provas P_1 e P_2: média $P = (4P_1 + 6P_2)/10$
- Projeto: implementação do compilador: C (veja transparência seguinte)
- Aproveitamento A:

$$A = \begin{cases} (P + C)/2 & \text{se } P \geq 5 \text{ e } C \geq 5 \\ \min(P, C) & \text{caso contrário.} \end{cases}$$

- Esta disciplina não tem exame final.

Projeto

Um compilador para a linguagem SL (*Simple Language*) a ser implementedo em quatro passos (tarefas do SuSy). O valor de cada passo para a nota final é:

- analisador léxico: 10%
- analisador sintático: 15%
- construção da árvore de programa: 15%
- compilador final:
  - parte básica de SL: 40%
  - funções como parâmetros: 10%
  - declarações de tipos e matrizes: 10%

Datas

- Prova P_1: 13 de outubro de 2016
- Prova P_2: 1 de dezembro de 2016
- Entrega final do projeto: 9 de dezembro de 2016

Regras

- O projeto é **estritamente individual**.
- Qualquer tentativa de fraude nas provas ou no projeto implicará em aproveitamento zero no semestre para todos os envolvidos, sem prejuízo de outras sanções.
- As transgressões às regras de uso dos sistemas computacionais do Instituto de Computação ou de outras unidades da UNICAMP que possam prejudicar outros usuários ou sistemas, dentro ou fora da Universidade, implicarão em aproveitamento zero no semestre para todos os envolvidos, sem prejuízo de outras sanções.

Programa

1. Introdução: aspectos básicos de compilação
2. Linguagem exemplo
3. Conceitos básicos de gramáticas livres de contexto e notação BNF
4. Análise sintática
5. Análise sintática descendente
6. Análise sintática ascendente
7. Análise léxica: *ad hoc* e expressões regulares
8. Ferramentas
9. Sistema de execução para a linguagem exemplo (máquina virtual)
10. Organização do compilador, análise semântica, tabelas de símbolos e geração de código
11. Tópicos complementares

Bibliografia


D. Grune and C. J. H. Jacobs
Parsing Techniques: a Practical Guide (2nd. ed.)
Springer, 2008


A. V. Aho, M. S. Lam, R. Sethi and J. D. Ullman
Compilers: Principles, Techniques, and Tools (2nd. ed.)
Addison Wesley, 2006


A. W. Appel and M. Ginsburg
Modern Compiler Implementation in C (new ed.)
Cambridge University Press, 2004


A. W. Appel and J. Palsberg
Modern Compiler Implementation in Java (2nd. ed.)
Cambridge University Press, 2002


J. R. Levine, T. Mason and D. Brown
lex & yacc (2nd. ed.)
O'Reilly, 1992

Bibliografia (cont.)


A. V. Aho, R. Sethi and J. D. Ullman
Compilers — Principles, Techniques, and Tools
Addison-Wesley, 1986


A. T. Schreiner and H. G. Friedman, Jr.
Introduction to Compiler Construction with Unix
Prentice Hall, 1985


Tomasz Kowaltowski.
Implementação de Linguagens de Programação
Editora Guanabara Dois, 1983


M. E. Lesk and E. Schmidt
Lex — A Lexical Analyzer
Bell Laboratories, 1978


S. C. Johnson
Yacc: Yet Another Compiler-Compiler
Bell Laboratories, 1978

Bibliografia (cont.)


A. V. Aho and J. D. Ullman
The Theory of Parsing, Translation and Compiling, vol. 1: Parsing
Prentice-Hall, 1972


A. V. Aho and J. D. Ullman
The Theory of Parsing, Translation and Compiling, vol. 2: Compiling
Prentice-Hall, 1973


C. Donnely and R. Stallman
Bison
Free Software Foundation, Inc, 2006
http://www.gnu.org/software/bison/manual


Sourceforge *Flex: a fast lexical analyzer generator*
http://flex.sourceforge.net/manual


Tom Niemann *A Compact Guide to Lex & Yacc*
http://www.epaperpress.com/lexandyacc

Material disponível

- Texto: T. Kowaltowski, *Implementação de Linguagens de Programação* (cópia em PDF autorizada para os alunos da disciplina).
- Cópias das transparências
- Exemplo de utilização das ferramentas na implementação de uma micro-linguagem TL (*Tiny Language*).
- Interpretador da máquina virtual MEPA em Python3.