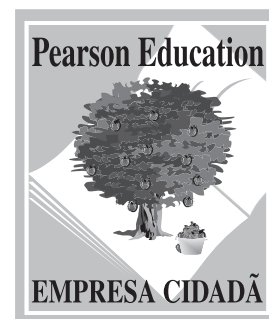
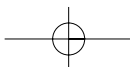
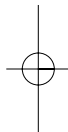
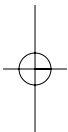
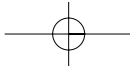


Compiladores

Princípios, técnicas e ferramentas

2^a Edição





Alfred V. Aho Monica S. Lam Ravi Sethi Jeffrey D. Ullman

Compiladores

Princípios, técnicas e ferramentas

2ª Edição

Tradução:
Daniel Vieira

Revisão Técnica:
Mariza Andrade da Silva Bigonha
Professora associado do Departamento de Ciência da Computação
Universidade Federal de Minas Gerais



São Paulo



Brasil Argentina Colômbia Costa Rica Chile Espanha Guatemala México Peru Porto Rico Venezuela

© 2008, Pearson Education do Brasil
© 2007 Pearson Education, Inc.

Tradução autorizada a partir da edição original em inglês, publicada pela Pearson Education, Inc. sob o selo Addison Wesley. Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Pearson Education do Brasil.

Gerente editorial: Roger Trimer
Editora sênior: Sabrina Cairo
Editor de desenvolvimento: Marco Pace
Editora de texto: Eugênia Pessotti
Preparação: Andrea Filatro
Revisão: Lucrécia Freitas e Sandra Scapin
Capa: Celso Blanes (sobre projeto original de S. D. Ullman – Strange Tonic Productions)
Composição Editorial: ERJ Composição Editorial e Artes Gráficas Ltda.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Compiladores : princípios, técnicas e ferramentas / Alfred V. Aho...[et al.];
tradução Daniel Vieira; revisão técnica Mariza Bigonha. – 2. ed. – São Paulo:
Pearson Addison-Wesley, 2008.

Outros autores: Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman
Título original: Compilers: principles, techniques, and tools

Bibliografia
ISBN 978-85-88639-24-9

1. Compiladores (Programas de computador)
I. Aho, Alfred V.. II. Lam, Monica S.. III. Sethi, Ravi. IV. Ullman, Jeffrey D..

07-7541

CDD-005.453

Índices para catálogo sistemático:

1. Compiladores : Linguagens de programação :
Ciências da computação 005.453

2007

Direitos exclusivos para a língua portuguesa cedidos à Pearson Education do Brasil,
uma empresa do grupo Pearson Education
Av. Ermano Marchetti, 1435
CEP: 05038-001 – São Paulo – SP
Fone: (11) 2178-8686 – Fax: (11) 2178-8688
e-mail: vendas@pearsoned.com

SUMÁRIO

Prefácio IX

1 Introdução 1

- 1.1 Processadores de linguagem 1
 - 1.2 A estrutura de um compilador 3
 - 1.3 Evolução das linguagens de programação 8
 - 1.4 A ciência da criação de um compilador 10
 - 1.5 Aplicações da tecnologia de compiladores 11
 - 1.6 Fundamentos da linguagem de programação 16
 - 1.7 Resumo do Capítulo 1 23
 - 1.8 Referências do Capítulo 1 24
-

2 Um tradutor simples dirigido por sintaxe 25

- 2.1 Introdução 25
 - 2.2 Definição da sintaxe 27
 - 2.3 Tradução dirigida por sintaxe 33
 - 2.4 Análise sintática 39
 - 2.5 Um tradutor para expressões simples 44
 - 2.6 Análise léxica 49
 - 2.7 Tabelas de símbolos 55
 - 2.8 Geração de código intermediário 59
 - 2.9 Resumo do Capítulo 2 68
-

3 Análise léxica 70

- 3.1 O papel do analisador léxico 70
- 3.2 Bufferes de entrada 73
- 3.3 Especificação de tokens 75
- 3.4 Reconhecimento de tokens 81
- 3.5 O gerador de analisador léxico Lex 89
- 3.6 Autômatos finitos 93
- 3.7 De expressões regulares a autômatos 97
- 3.8 Projeto de um gerador de analisador léxico 106

3.9	Otimização de casadores de padrão baseado em DFA	110
3.10	Resumo do Capítulo 3	119
3.11	Referências do Capítulo 3	120

4 Análise sintática 122

4.1	Introdução	122
4.2	Gramáticas livres de contexto	125
4.3	Escrevendo uma gramática	133
4.4	Análise sintática descendente	138
4.5	Análise ascendente	149
4.6	Introdução à análise LR simples: SLR	154
4.7	Analisadores sintáticos LR mais poderosos	166
4.8	Usando gramáticas ambíguas	178
4.9	Geradores de analisadores sintáticos	183
4.10	Resumo do Capítulo 4	190
4.11	Referências do Capítulo 4	192

5 Tradução dirigida por sintaxe 194

5.1	Definições dirigidas por sintaxe	194
5.2	Ordens de avaliação para SDDs	198
5.3	Aplicações da tradução dirigida por sintaxe	203
5.4	Esquemas de tradução dirigidos por sintaxe	207
5.5	Implementando SDDs L-atribuídas	216
5.6	Resumo do Capítulo 5	226
5.7	Referências do Capítulo 5	227

6 Geração de código intermediário 228

6.1	Variantes das árvores de sintaxe	229
6.2	Código de três endereços	232
6.3	Tipos e declarações	236
6.4	Tradução de expressões	242
6.5	Verificação de tipo	247
6.6	Fluxo de controle	255
6.7	Remendos	262
6.8	Comandos switch	268
6.9	Código intermediário para procedimentos	270
6.10	Resumo do Capítulo 6	272
6.11	Referências do Capítulo 6	272

7 Ambientes de execução 274

7.1	Organização de memória	274
7.2	Alocação de espaço na pilha	276
7.3	Acesso a dados não locais na pilha	283

7.4	Gerenciamento de heap	289
7.5	Introdução à coleta de lixo	297
7.6	Introdução à coleta baseada em rastreamento	301
7.7	Coleta de lixo com pausa curta	309
7.8	Tópicos avançados sobre coleta de lixo	315
7.9	Resumo do Capítulo 7	318
7.10	Referências do Capítulo 7	319

8 Geração de código 321

8.1	Questões sobre o projeto de um gerador de código	322
8.2	A linguagem objeto	325
8.3	Endereços no código objeto	329
8.4	Blocos básicos e grafos de fluxo	333
8.5	Otimização de blocos básicos	338
8.6	Um gerador de código simples	344
8.7	Otimização peephole	348
8.8	Alocação e atribuição de registradores	350
8.9	Seleção de instrução por reescrita de árvore	353
8.10	Geração de código ótimo para expressões	359
8.11	Geração de código com programação dinâmica	363
8.12	Resumo do Capítulo 8	366
8.13	Referências do Capítulo 8	366

9 Otimizações independentes de máquina 368

9.1	As principais fontes de otimização	368
9.2	Introdução à análise de fluxo de dados	377
9.3	Fundamentos da análise de fluxo de dados	389
9.4	Propagação de constante	398
9.5	Eliminação de redundância parcial	402
9.6	<i>Loops</i> em grafos de fluxo	413
9.7	Análise baseada em região	422
9.8	Análise simbólica	431
9.9	Resumo do Capítulo 9	440
9.10	Referências do Capítulo 9	441

10 Paralelismo de instrução 443

10.1	Arquiteturas de processadores	443
10.2	Restrições no escalonamento de código	445
10.3	Escalonamento de bloco básico	452
10.4	Escalonamento global do código	455
10.5	Software Pipelining	462
10.6	Resumo do Capítulo 10	479
10.7	Referências do Capítulo 10	480

11	Otimização de paralelismo e localidade	482
11.1	Conceitos básicos	483
11.2	Multiplicação de matriz: um exemplo detalhado	490
11.3	Espaços de iteração	494
11.4	Índices de arranjo afins	502
11.5	Reúso de dados	503
11.6	Análise de dependência de dados de arranjo	510
11.7	Localizando o paralelismo sem sincronização	518
11.8	Sincronização entre <i>loops</i> paralelos	534
11.9	Pipelining	539
11.10	Otimizações de localidade	555
11.11	Outros usos das transformações de afins	561
11.12	Resumo do Capítulo 11	564
11.13	Referências do Capítulo 11	565

12	Análise interprocedimental	567
12.1	Conceitos básicos	567
12.2	Por que análise interprocedimental?	575
12.3	Uma representação lógica do fluxo de dados	578
12.4	Um algoritmo simples de análise de apontadores	585
12.5	Análise interprocedimental insensível ao contexto	590
12.6	Análise de apontador sensível ao contexto	593
12.7	Implementação em Datalog pelos BDDs	596
12.8	Resumo do Capítulo 12	601
12.9	Referências do Capítulo 12	602

A	Um <i>front-end</i> completo	604
A.1	A linguagem fonte	604
A.2	Main	605
A.3	Analisador léxico	605
A.4	Tabelas de símbolos e tipos	608
A.5	Código intermediário para expressões	609
A.6	Código de desvio para expressões Booleanas	611
A.7	Código intermediário para comandos	614
A.8	Analisador sintático	617
A.9	Criando o <i>front-end</i>	621

B	Encontrando soluções linearmente independentes	623
----------	---	------------

	Índice remissivo	625
--	-------------------------	------------

	Sobre os autores	635
--	-------------------------	------------

PREFÁCIO

Desde a publicação da primeira edição deste livro, em 1986, o mundo do projeto de compiladores mudou significativamente. As linguagens de programação evoluíram a ponto de apresentarem novos problemas de compilação e as arquiteturas de computador passaram a oferecer diversos recursos dos quais o projetista de compilador simplesmente não pode abrir mão. Mais interessante, talvez, seja que a venerável tecnologia de otimização de código encontrou uso fora dos compiladores. Agora, ela é usada em ferramentas que encontram erros no *software* e, mais importante, encontram brechas de segurança no código existente. E grande parte da tecnologia de *front-end* — gramáticas, expressões regulares, analisadores sintáticos e tradutores dirigidos por sintaxe — ainda é muito usada.

Portanto, nossa filosofia, apresentada na edição anterior deste livro, não mudou. Reconhecemos que poucos leitores construirão, ou mesmo realizarão a manutenção de um compilador para qualquer uma das principais linguagens de programação. Ainda assim, os modelos, a teoria e os algoritmos associados a um compilador podem ser aplicados a uma grande gama de problemas no projeto e desenvolvimento de *software*. Portanto, enfatizamos problemas que são encontrados mais comumente no projeto de um processador de linguagem, independentemente da linguagem fonte ou da máquina alvo.

Uso do livro

São necessários pelo menos dois trimestres, ou até mesmo dois semestres, para estudar todo ou quase todo o conteúdo deste livro. É comum o uso da primeira metade em um curso de graduação e a segunda metade — enfatizando a otimização do código — em um segundo curso, no nível de pós-graduação. Veja a seguir um resumo de cada um dos capítulos.

Capítulo 1: contém material introdutório e apresenta uma visão geral sobre a arquitetura de computador e os princípios de linguagem de programação.

Capítulo 2: desenvolve um compilador em miniatura e introduz muito dos conceitos importantes, que posteriormente são desenvolvidos em outros capítulos. O compilador em si aparece no Apêndice.

Capítulo 3: abrange a análise léxica, expressões regulares, máquinas de estado finito e ferramentas geradoras de analisadores léxicos. Esse material é fundamental para o processamento de textos de todas as espécies.

Capítulo 4: cobre os principais métodos de análise sintática, descendente (descida recursiva, LL) e ascendente (LR e suas variantes).

Capítulo 5: introduz as principais idéias sobre definições dirigidas por sintaxe e traduções dirigidas por sintaxe.

Capítulo 6: baseia-se na teoria do Capítulo 5 e mostra como usá-la para gerar código intermediário para uma linguagem de programação típica.

Capítulo 7: cobre os ambientes em tempo de execução, especialmente o gerenciamento da pilha de execução e a coleta de lixo.

Capítulo 8: trata da geração de código objeto. Ele abrange a construção de blocos básicos, geração de código a partir de expressões e blocos básicos e técnicas de alocação de registradores.

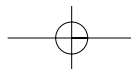
Capítulo 9: introduz a tecnologia de otimização de código, incluindo grafo de fluxo, estruturas de fluxo de dados e algoritmos iterativos para solucionar essas estruturas.

Capítulo 10: abrange a otimização em nível de instrução. A ênfase é dada à extração do paralelismo a partir de pequenas seqüências de instruções e seu escalonamento em processadores individuais que podem efetuar mais de uma tarefa de uma só vez.

Capítulo 11: trata de detecção e exploração de paralelismo em escala maior. Neste capítulo, a ênfase é sobre códigos de aplicação numéricas que possuem muitos laços estreitos que iteram sobre arranjos multidimensionais.

Capítulo 12: apresenta a análise entre procedimentos. Ele aborda a análise de apontador, sinônimos e a análise de fluxo de dados que leva em consideração a seqüência de chamadas de procedimento que alcançam um determinado ponto no código.

Os cursos de compiladores das universidades de Columbia, Harvard e Stanford são baseados neste livro. Em Columbia, uma disciplina de fim de curso de graduação ou de primeiro ano de pós-graduação sobre linguagens de programação e tradutores tem sido oferecido regularmente, usando o material dos oito primeiros capítulos. Um destaque desse curso é um trabalho de



um semestre no qual os alunos se reúnem em pequenas equipes para criar e implementar uma pequena linguagem com seu próprio projeto. As linguagens criadas pelos alunos têm incluído diversos domínios de aplicação, incluindo cálculo quântico, síntese musical, computação gráfica, jogos, operações com matriz e muitas outras áreas. Os alunos utilizam geradores de componente de compilador, como ANTLR, Lex e Yacc, e as técnicas de tradução dirigidas por sintaxe, discutidas nos capítulos 2 e 5, para criar seus compiladores. Um outro curso de graduação utilizou o material dos capítulos de 9 a 12, enfatizando a geração e otimização de código para máquinas contemporâneas, incluindo processadores em rede e arquiteturas de multiprocessadores.

Em Stanford, um curso introdutório de um trimestre abrange parte do material dos Capítulos de 1 a 8, embora exista uma introdução à otimização de código global do Capítulo 9. O segundo curso de compiladores abrange os capítulos de 9 a 12 e o material mais avançado sobre coleta de lixo, do Capítulo 7. Os alunos usam um sistema desenvolvido localmente, baseado em Java, chamado Joeq, para implementar os algoritmos de análise de fluxo de dados.

Pré-requisitos

O leitor deverá possuir alguma 'sofisticação em ciência da computação', incluindo pelo menos um segundo curso sobre programação, e cursos sobre estruturas de dados e matemática discreta. O conhecimento de diversas linguagens de programação diferentes é útil.

Exercícios

O livro contém muitos exercícios, em quase todas as seções. Indicamos os exercícios mais difíceis, ou as partes mais complexas de alguns, com um ponto de exclamação. Os exercícios ainda mais difíceis possuem um ponto de exclamação duplo.



Companion Website

No site de apoio do livro, que pode ser acessado no endereço www.aw.com/aho_br, estão disponíveis o fonte do Apêndice A e uma biblioteca de imagens.

Agradecimentos

A arte da capa é de S. D. Ullman, da Strange Tonic Productions.

John Bentley nos forneceu muitos comentários sobre vários capítulos de um rascunho preliminar deste livro. Recebemos sugestões úteis e erratas da edição anterior das seguintes pessoas: Domenico Bianculli, Peter Bosch, Marcio Buss, Marc Eaddy, Stephen Edwards, Vibhav Garg, Kim Hazelwood, Gaurav Kc, Wei Li, Mike Smith, Art Stamness, Krysta Svore, Olivier Tardieu, e Jia Zeng. Agradecemos a ajuda de todas elas. Os erros que restaram são nossos, naturalmente.

Além disso, Monica gostaria de agradecer aos seus colegas da equipe de compiladores do SUIF por uma lição de 18 anos sobre compilação: Gerald Aigner, Dzintars Avots, Saman Amarasinghe, Jennifer Anderson, Michael Carbin, Gerald Cheong, Amer Diwan, Robert French, Anwar Ghuloum, Mary Hall, John Hennessy, David Heinem, Shih-Wei Liao, Amy Lim, Benjamin Livshits, Michael Martin, Dror Maydan, Todd Mowry, Brian Murphy, Jeffrey Oplinger, Karen Pieper, Martin Rinard, Olatunji Ruwase, Constantine Sapuntzakis, Patrick Sathyanathan, Michael Smith, Steven Tjiang, Chau-Wen Tseng, Christopher Unkel, John Whaley, Robert Wilson, Christopher Wilson e Michael Wolf.

A. V. A., Chatham NJ
M. S. L., Menlo Park CA
R. S., Far Hills NJ
J. D. U., Stanford CA
Junho de 2006.

Agradecimento dos editores brasileiros

Agradecemos aos professores Sandra Maria Aluísio, da Universidade de São Paulo, campus São Carlos, Hélio Aparecido Navarro, da Universidade Estadual de São Paulo e Pedro S. Nicolletti, da Universidade Federal de Campina Grande, por sua colaboração no processo de avaliação desta obra.

