

Wydanie II

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

KOMPILATORY

REGUŁY • METODY • NARZĘDZIA



 PWN

KOMPILATORY

Wydanie II

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

KOMPILATORY

REGUŁY • METODY • NARZĘDZIA



Kup książkę

Dane oryginału

Authorized translation from the English language edition, entitled: COMPILERS: PRINCIPLES, TECHNIQUES, AND TOOLS; Second Edition; ISBN 0321486811; by Alfred V. Aho; and by Monica S. Lam; and by Ravi Sethi; and by Jeffrey D. Ullman; published by Pearson Education, Inc, publishing as Addison Wesley. Copyright © 2007 by Bell Telephone Laboratories, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc. Polish language edition published by Scientific Publishers PWN Wydawnictwo Naukowe PWN Spółka Akcyjna. Copyright © 2019.

Przekład **Marek Włodarz** na zlecenie **WITKOM Witold Sikorski**

Konsultant **Janusz Majewski**

Projekt okładki i stron tytułowych **Joanna Andryjowicz**

Wydawca **Edyta Kawala**

Redaktor prowadzący **Jolanta Kowalczuk**

Redaktorzy merytoryczni **Janusz Majewski** (rozdz. 1–8)

Grzegorz Herman (rozdz. 9–12, dod. A i B)

Korekta **Anna Marecka**

Redaktor techniczny **Maria Czekaj**

Koordynator produkcji **Anna Bączkowska**

Skład i łamanie **FixPoint**

Zastrzeżonych nazw firm i produktów użyto w książce wyłącznie w celu identyfikacji.

Copyright © for the Polish edition by Wydawnictwo Naukowe PWN SA
Warszawa 2019

ISBN 978-83-01-20381-8

Wydanie II (I w WN PWN)
Warszawa 2019

Wydawnictwo Naukowe PWN SA
02-460 Warszawa, ul. Gottlieba Daimlera 2
tel. 22 69 54 321, faks 22 69 54 288
infolinia 801 33 33 88
e-mail: pwn@pwn.com.pl, reklama@pwn.pl
www.pwn.pl

Druk i oprawa **Paper&Tinta**

Kup książkę

Spis treści

Przedmowa	XXI
1. Wprowadzenie	1
1.1. Translatory	1
1.1.1. Ćwiczenia do podrozdziału 1.1	4
1.2. Struktura kompilatora	4
1.2.1. Analiza leksykalna	6
1.2.2. Analiza składniowa	7
1.2.3. Analiza semantyczna	9
1.2.4. Generowanie kodu pośredniego	9
1.2.5. Optymalizacja kodu	10
1.2.6. Generowanie kodu	11
1.2.7. Zarządzanie tablicą symboli	11
1.2.8. Grupowanie faz w przebiegi	12
1.2.9. Narzędzia do budowania kompilatorów	12
1.3. Ewolucja języków programowania	13
1.3.1. Przejście na języki wyższego poziomu	13
1.3.2. Wpływ na kompilatory	14
1.3.3. Ćwiczenia do podrozdziału 1.3	15
1.4. Teoria konstruowania kompilatorów	16
1.4.1. Modelowanie w projektowaniu i implementacji kompilatora	16
1.4.2. Nauka o optymalizacji kodu	16
1.5. Zastosowania technologii kompilatorów	18
1.5.1. Implementacja języków programowania wysokiego poziomu	19
1.5.2. Optymalizacje architektur komputerów	21
1.5.3. Projekty nowych architektur komputerów	22
1.5.4. Tłumaczenie programów	24
1.5.5. Narzędzia niezawodności oprogramowania	25
1.6. Podstawy języków programowania	27
1.6.1. Rozróżnienie statyczny/dynamiczny	28
1.6.2. Środowiska i stany	28
1.6.3. Statyczny zasięg i struktura blokowa	30
1.6.4. Jawna kontrola dostępu	34
1.6.5. Zasięg dynamiczny	34
1.6.6. Mechanizmy przekazywania parametrów	36
1.6.7. Aliasowanie	38
1.6.8. Ćwiczenia do podrozdziału 1.6	39
1.7. Podsumowanie	40
1.8. Bibliografia	41

2. Prosty translator sterowany składnią	43
2.1. Wprowadzenie	44
2.2. Definiowanie składni	46
2.2.1. Definicja gramatyki	46
2.2.2. Wyprowadzenia	48
2.2.3. Drzewa rozbioru	49
2.2.4. Niejednoznaczność	51
2.2.5. Łączność operatorów	52
2.2.6. Priorytety operatorów	53
2.2.7. Ćwiczenia do podrozdziału 2.2	56
2.3. Translacja sterowana składnią	57
2.3.1. Notacja postfiksowa	58
2.3.2. Syntetyzowane atrybuty	58
2.3.3. Proste definicje sterowane składnią	60
2.3.4. Przechodzenie drzewa	61
2.3.5. Schematy translacji	63
2.3.6. Ćwiczenia do podrozdziału 2.3	65
2.4. Analiza składniowa	66
2.4.1. Analiza zstępująca	66
2.4.2. Analiza predykcyjna	69
2.4.3. Kiedy używać ϵ -produkcji	71
2.4.4. Projektowanie parsera predykcyjnego	72
2.4.5. Lewostronna rekurencja	73
2.4.6. Ćwiczenia do podrozdziału 2.4	74
2.5. Translator dla prostych wyrażeń	74
2.5.1. Składnia abstrakcyjna i konkretna	75
2.5.2. Dostosowywanie schematu translacji	76
2.5.3. Procedury dla nieterminali	77
2.5.4. Upraszczanie translatora	78
2.5.5. Kompletny program	79
2.6. Analiza leksykalna	82
2.6.1. Usuwanie białych znaków i komentarzy	83
2.6.2. Czytanie z wyprzedzeniem	84
2.6.3. Stałe	84
2.6.4. Rozpoznawanie słów kluczowych i identyfikatorów	85
2.6.5. Analizator leksykalny	87
2.6.6. Ćwiczenia do podrozdziału 2.6	91
2.7. Tablice symboli	91
2.7.1. Tablica symboli dla zasięgu	93
2.7.2. Używanie tablic symboli	96
2.8. Generowanie kodu pośredniego	98
2.8.1. Dwa rodzaje reprezentacji pośrednich	98
2.8.2. Konstruowanie drzew składniowych	99
2.8.3. Kontrole statyczne	104
2.8.4. Kod trójadresowy	106
2.8.5. Ćwiczenia do podrozdziału 2.8	112
2.9. Podsumowanie	112

3. Analiza leksykalna	115
3.1. Rola analizatora leksykalnego	115
3.1.1. Analiza leksykalna kontra analiza składniowa (<i>parsing</i>)	117
3.1.2. Tokeny, wzorce i leksemy	117
3.1.3. Atrybuty tokenów	118
3.1.4. Błędy leksykalne	120
3.1.5. Ćwiczenia do podrozdziału 3.1	121
3.2. Buforowanie wejścia	121
3.2.1. Pary buforów	122
3.2.2. Wartownicy	123
3.3. Specyfikacje tokenów	124
3.3.1. Ciągi i języki	125
3.3.2. Działania na językach	126
3.3.3. Wyrażenia regularne	127
3.3.4. Definicje regularne	129
3.3.5. Rozszerzenia wyrażeń regularnych	130
3.3.6. Ćwiczenia do podrozdziału 3.3	131
3.4. Rozpoznawanie tokenów	135
3.4.1. Diagramy przejść	136
3.4.2. Rozpoznawanie zastrzeżonych słów i identyfikatorów	139
3.4.3. Dokończenie przykładu	140
3.4.4. Architektura analizatora leksykalnego opartego na diagramie przejść	141
3.4.5. Ćwiczenia do podrozdziału 3.4	143
3.5. Lex – generator analizatorów leksykalnych	147
3.5.1. Korzystanie z Lex	147
3.5.2. Struktura programów w języku Lex	148
3.5.3. Rozwiązywanie konfliktów w Lex	152
3.5.4. Operator prawego kontekstu	152
3.5.5. Ćwiczenia do podrozdziału 3.5	153
3.6. Automaty skończone	154
3.6.1. Niedeterministyczne automaty skończone	155
3.6.2. Tablice przejść	156
3.6.3. Akceptowanie ciągów wejściowych przez automat	156
3.6.4. Deterministyczne automaty skończone	157
3.6.5. Ćwiczenia do podrozdziału 3.6	158
3.7. Od wyrażeń regularnych do automatów	159
3.7.1. Konwertowanie NAS na DAS	160
3.7.2. Symulacja NAS	163
3.7.3. Wydajność symulacji NAS	164
3.7.4. Konstruowanie NAS z wyrażenia regularnego	166
3.7.5. Wydajność algorytmów przetwarzania ciągów	170
3.7.6. Ćwiczenia do podrozdziału 3.7	174
3.8. Projektowanie generatora analizatorów leksykalnych	174
3.8.1. Struktura generowanego analizatora	174
3.8.2. Dopasowywanie wzorców na podstawie NAS	177

3.8.3.	DAS dla analizatorów leksykalnych	178
3.8.4.	Implementacja operatora prawego kontekstu	179
3.8.5.	Ćwiczenia do podrozdziału 3.8	180
3.9.	Optymalizacja mechanizmów rozpoznających wzorce oparte na DAS	180
3.9.1.	Istotne stany w NAS	181
3.9.2.	Funkcje wyliczane z drzewa składniowego	183
3.9.3.	Obliczanie <i>nullable</i> , <i>firstpos</i> i <i>lastpos</i>	184
3.9.4.	Obliczanie <i>followpos</i>	185
3.9.5.	Bezpośrednie konwertowanie wyrażenia regularnego na DAS	187
3.9.6.	Minimalizacja liczby stanów w DAS	189
3.9.7.	Minimalizacja liczby stanów w analizatorach leksykalnych	193
3.9.8.	Wybieranie między czasem a pamięcią w symulatorze DAS	193
3.9.9.	Ćwiczenia do podrozdziału 3.9	195
3.10.	Podsumowanie	195
3.11.	Bibliografia	197
4.	Analiza składniowa	201
4.1.	Wprowadzenie	202
4.1.1.	Rola parsera	202
4.1.2.	Reprezentatywne gramatyki	203
4.1.3.	Obsługa błędów składniowych	204
4.1.4.	Strategie przywracania kontroli po błędzie	205
4.2.	Gramatyki bezkontekstowe	207
4.2.1.	Formalna definicja gramatyki bezkontekstowej	207
4.2.2.	Konwencje notacyjne	209
4.2.3.	Wyprowadzenie	210
4.2.4.	Drzewa rozbioru	212
4.2.5.	Niejednoznaczność	214
4.2.6.	Weryfikowanie języka wygenerowanego przez gramatykę	214
4.2.7.	Gramatyki bezkontekstowe a wyrażenia regularne	216
4.2.8.	Ćwiczenia do podrozdziału 4.2	217
4.3.	Tworzenie gramatyki	220
4.3.1.	Analiza leksykalna a analiza składniowa	220
4.3.2.	Eliminowanie niejednoznaczności	221
4.3.3.	Eliminowanie rekurencji lewostronnej	222
4.3.4.	Lewostronna faktoryzacja	225
4.3.5.	Konstrukcje językowe niebędące bezkontekstowymi	226
4.3.6.	Ćwiczenia do podrozdziału 4.3	227
4.4.	Analiza zstępująca	228
4.4.1.	Analiza oparta na zejściach rekurencyjnych	229
4.4.2.	FIRST oraz FOLLOW	231
4.4.3.	Gramatyki LL(1)	233
4.4.4.	Nierekurencyjna analiza predykcyjna	237

4.4.5.	Przywracanie po błędzie w analizie predykcyjnej	239
4.4.6.	Ćwiczenia do podrozdziału 4.4	242
4.5.	Analiza wstępująca	245
4.5.1.	Redukcje	245
4.5.2.	Przycinanie uchwytów	246
4.5.3.	Analiza metodą przesunięcie-redukcja	247
4.5.4.	Konflikty w analizie metodą przesunięcie-redukcja	249
4.5.5.	Ćwiczenia do podrozdziału 4.5	251
4.6.	Wprowadzenie do analizy LR: proste LR (SLR)	252
4.6.1.	Dlaczego parsery LR?	252
4.6.2.	Sytuacje i automat LR(0)	253
4.6.3.	Algorytm parsingu LR	259
4.6.4.	Konstruowanie tablic analizy SLR	264
4.6.5.	Żywotne prefiksy	267
4.6.6.	Ćwiczenia do podrozdziału 4.6	268
4.7.	Bardziej skuteczne parsery LR	270
4.7.1.	Kanoniczne sytuacje LR(1)	270
4.7.2.	Konstruowanie zbiorów sytuacji LR(1)	272
4.7.3.	Tablice analizy kanonicznego LR(1)	275
4.7.4.	Konstruowanie tablic analizy LALR	276
4.7.5.	Wydajne konstruowanie tablic analizy LALR	281
4.7.6.	Kompresowanie tablic analizy LR	286
4.7.7.	Ćwiczenia do podrozdziału 4.7	288
4.8.	Gramatyki niejednoznaczne	289
4.8.1.	Wykorzystanie priorytetów i łączności do rozwiązywania konfliktów	289
4.8.2.	Niejednoznaczność „wiszącego else”	292
4.8.3.	Przywracanie kontroli po błędzie w analizie LR	294
4.8.4.	Ćwiczenia do podrozdziału 4.8	297
4.9.	Generatory parserów	298
4.9.1.	Generator parserów Yacc	298
4.9.2.	Używanie Yacc z gramatykami niejednoznacznymi	302
4.9.3.	Tworzenie analizatorów leksykalnych zgodnych z Yacc przy użyciu Lex	305
4.9.4.	Przywracanie kontroli po błędach w Yacc	306
4.9.5.	Ćwiczenia do podrozdziału 4.9	308
4.10.	Podsumowanie	308
4.11.	Bibliografia	311
5.	Translacja sterowana składnią	315
5.1.	Definicje sterowane składnią	316
5.1.1.	Atrybuty dziedziczone i syntetyzowane	316
5.1.2.	Przetwarzanie SDD w węzłach drzewa rozbioru	318
5.1.3.	Ćwiczenia do podrozdziału 5.1	322

5.2.	Kolejność przetwarzania w SDD	322
5.2.1.	Grafy zależności	322
5.2.2.	Porządkowanie obliczania atrybutów	324
5.2.3.	Definicje S-atrybutowane	325
5.2.4.	Definicje L-atrybutowane	325
5.2.5.	Reguły semantyczne z kontrolowanymi efektami ubocznymi	327
5.2.6.	Ćwiczenia do podrozdziału 5.2	329
5.3.	Zastosowania translacji sterowanej składnią	330
5.3.1.	Konstruowanie drzew składniowych	330
5.3.2.	Struktura opisu typu	334
5.3.3.	Ćwiczenia do podrozdziału 5.3	336
5.4.	Sterowane składnią schematy translacji	336
5.4.1.	Postfiksowe schematy translacji	337
5.4.2.	Implementacja postfiksowego SDT przez stos parsera	338
5.4.3.	SDT z akcjami wewnątrz produkcji	339
5.4.4.	Eliminowanie rekurencji lewostronnej z SDT	341
5.4.5.	SDT dla definicji L-atrybutowanych	344
5.4.6.	Ćwiczenia do podrozdziału 5.4	349
5.5.	Implementacja L-atrybutowanych SDD	350
5.5.1.	Tłumaczenie podczas parsingu na podstawie zejść rekurencyjnych	351
5.5.2.	Generowanie kodu w locie	354
5.5.3.	L-atrybutowane SDD i parsing LL	356
5.5.4.	Analiza wstępująca L-atrybutowanych SDD	361
5.5.5.	Ćwiczenia do podrozdziału 5.5	366
5.6.	Podsumowanie	366
5.7.	Bibliografia	368
6.	Generowanie kodu pośredniego	371
6.1.	Odmiany drzew składniowych	372
6.1.1.	Skierowane grafy acykliczne dla wyrażeń	373
6.1.2.	Metoda numerowania wartości do konstruowania DAG	374
6.1.3.	Ćwiczenia do podrozdziału 6.1	377
6.2.	Kod trójadresowy	377
6.2.1.	Adresy i instrukcje	378
6.2.2.	Czwórki	380
6.2.3.	Trójki	381
6.2.4.	Forma Static Single Assignment	383
6.2.5.	Ćwiczenia do podrozdziału 6.2	384
6.3.	Typy i deklaracje	384
6.3.1.	Wyrażenia określające typy	385
6.3.2.	Równoważność typów	386
6.3.3.	Deklaracje	387
6.3.4.	Rozmieszczenie w pamięci dla nazw lokalnych	388

6.3.5.	Sekwencje deklaracji	390
6.3.6.	Pola rekordów i klas	391
6.3.7.	Ćwiczenia do podrozdziału 6.3	392
6.4.	Translacja wyrażeń	393
6.4.1.	Operacje wewnątrz wyrażeń	393
6.4.2.	Tłumaczenie przyrostowe	395
6.4.3.	Adresowanie elementów tablic	395
6.4.4.	Tłumaczenie odwołań do tablic	397
6.4.5.	Ćwiczenia do podrozdziału 6.4	399
6.5.	Kontrola typów	401
6.5.1.	Reguły kontroli typów	401
6.5.2.	Konwersje typów	402
6.5.3.	Przeciążanie funkcji i operatorów	405
6.5.4.	Inferencja typów i funkcje polimorficzne	405
6.5.5.	Algorytm unifikacji	410
6.5.6.	Ćwiczenia do podrozdziału 6.5	413
6.6.	Przepływ sterowania	413
6.6.1.	Wyrażenia logiczne	414
6.6.2.	Kod krótki	415
6.6.3.	Instrukcje sterujące	415
6.6.4.	Translacja wyrażeń logicznych	418
6.6.5.	Unikanie nadmiarowych skoków goto	420
6.6.6.	Wartości logiczne i skaczący kod	422
6.6.7.	Ćwiczenia do podrozdziału 6.6	423
6.7.	Backpatching	424
6.7.1.	Jednoprzebiegowe generowanie kodu przy użyciu poprawiania wstecznego	425
6.7.2.	Backpatching wyrażeń logicznych	426
6.7.3.	Instrukcje sterujące	429
6.7.4.	Instrukcje break, continue i goto	431
6.7.5.	Ćwiczenia do podrozdziału 6.7	432
6.8.	Instrukcje wyboru	433
6.8.1.	Translacja instrukcji <i>switch</i>	433
6.8.2.	Sterowana składnią translacja instrukcji <i>switch</i>	435
6.8.3.	Ćwiczenia do podrozdziału 6.8	436
6.9.	Kod pośredni dla procedur	437
6.10.	Podsumowanie	439
6.11.	Bibliografia	440
7.	Środowiska wykonania	443
7.1.	Organizacja pamięci	443
7.1.1.	Alokacje statyczne kontra dynamiczne	445

7.2.	Stosowa rezerwacja pamięci	446
7.2.1.	Drzewa aktywacji	446
7.2.2.	Rekordy aktywacji	450
7.2.3.	Sekwencje wywołujące	452
7.2.4.	Dane zmiennej długości na stosie	455
7.2.5.	Ćwiczenia do podrozdziału 7.2	457
7.3.	Dostęp do nielokalnych danych na stosie	458
7.3.1.	Dostęp do danych bez zagnieżdżonych procedur	459
7.3.2.	Problemy dotyczące zagnieżdżonych procedur	459
7.3.3.	Język z zagnieżdżonymi deklaracjami procedur	460
7.3.4.	Głębokość zagnieżdżenia	460
7.3.5.	Wiązania dostępu	462
7.3.6.	Manipulowanie wiązaniami dostępu	464
7.3.7.	Wiązania dostępu a parametry procedur	465
7.3.8.	Tablice display	467
7.3.9.	Ćwiczenia do podrozdziału 7.3	469
7.4.	Zarządzanie stertą	470
7.4.1.	Zarządca pamięci	470
7.4.2.	Hierarchia pamięci komputera	471
7.4.3.	Lokalność w programach	473
7.4.4.	Redukowanie fragmentacji	476
7.4.5.	Ręczne żądania dealokacji	479
7.4.6.	Ćwiczenia do podrozdziału 7.4	482
7.5.	Wprowadzenie do odśmiecania pamięci	482
7.5.1.	Cele projektowe odśmiecaczy pamięci	483
7.5.2.	Osiągalność	486
7.5.3.	Kolektory śmieci ze zliczaniem referencji	488
7.5.4.	Ćwiczenia do podrozdziału 7.5	490
7.6.	Wprowadzenie do odśmiecania bazującego na śledzeniu	491
7.6.1.	Podstawowy odśmiecacz Mark and Sweep	491
7.6.2.	Podstawowa abstrakcja	493
7.6.3.	Optymalizowanie znakowania i zamiatania	495
7.6.4.	Odśmiecacz Mark and Compact	497
7.6.5.	Odśmiecacz kopiujący	500
7.6.6.	Porównanie kosztów	502
7.6.7.	Ćwiczenia do podrozdziału 7.6	503
7.7.	Odśmiecanie z krótkimi pauzami	504
7.7.1.	Przyrostowe zbieranie danych śmieciowych	504
7.7.2.	Przyrostowa analiza osiągalności	506
7.7.3.	Założenia odśmiecaczy częściowych	508
7.7.4.	Generacyjne zbieranie danych śmieciowych	510
7.7.5.	Algorytm pociągowy	511
7.7.6.	Ćwiczenia do podrozdziału 7.7	515
7.8.	Zaawansowane zagadnienia związane ze sprzątaniami pamięci	516
7.8.1.	Równoległe i współbieżne odśmiecanie pamięci	517

7.8.2.	Częściowa relokacja obiektów	519
7.8.3.	Konserwatywne odświeżanie dla języków niebezpiecznych typologicznie	520
7.8.4.	Słabe referencje	521
7.8.5.	Ćwiczenia do podrozdziału 7.8	522
7.9.	Podsumowanie	522
7.10.	Bibliografia	525
8.	Generowanie kodu	527
8.1.	Zagadnienia projektowania generatora kodu	528
8.1.1.	Dane wejściowe generatora kodu	529
8.1.2.	Program docelowy	529
8.1.3.	Wybieranie rozkazów	531
8.1.4.	Przydzielanie rejestrów	532
8.1.5.	Kolejność wykonywania	534
8.2.	Język docelowy	534
8.2.1.	Prosty model maszyny docelowej	534
8.2.2.	Koszty programu i rozkazów	537
8.2.3.	Ćwiczenia do podrozdziału 8.2	538
8.3.	Adresy w kodzie wynikowym	540
8.3.1.	Alokacje statyczne	541
8.3.2.	Alokacja na stosie	543
8.3.3.	Adresy czasu wykonania dla nazw	546
8.3.4.	Ćwiczenia do podrozdziału 8.3	546
8.4.	Bloki podstawowe i grafy przepływu	547
8.4.1.	Bloki podstawowe	548
8.4.2.	Informacja o następnym użyciu	550
8.4.3.	Grafy przepływu	551
8.4.4.	Reprezentacje grafów przepływu	553
8.4.5.	Pętle	553
8.4.6.	Ćwiczenia do podrozdziału 8.4	554
8.5.	Optymalizowanie bloków podstawowych	555
8.5.1.	Reprezentacja bloków podstawowych jako skierowanych grafów acyklicznych (DAG)	555
8.5.2.	Wyszukiwanie lokalnych podwyrażeń wspólnych	556
8.5.3.	Eliminowanie martwego kodu	558
8.5.4.	Korzystanie z tożsamości algebraicznych	558
8.5.5.	Reprezentacja odwołań do tablic	560
8.5.6.	Przypisania przy użyciu wskaźników i wywołania procedur	562
8.5.7.	Odtwarzanie bloków podstawowych z DAG	562
8.5.8.	Ćwiczenia do podrozdziału 8.5	564
8.6.	Prosty generator kodu	565
8.6.1.	Deskryptory rejestrów i adresów	566
8.6.2.	Algorytm generowania kodu	567
8.6.3.	Projekt funkcji <i>getReg</i>	570
8.6.4.	Ćwiczenia do podrozdziału 8.6	572

8.7.	Optymalizacja przez szparkę	572
8.7.1.	Eliminowanie nadmiarowych ładowań i zapisów	573
8.7.2.	Eliminowanie nieosiągalnego kodu	573
8.7.3.	Optymalizacje przepływu sterowania	574
8.7.4.	Uproszczenia algebraiczne i redukcje mocy operatorów	575
8.7.5.	Użycie idiomów języka maszynowego	576
8.7.6.	Ćwiczenia do podrozdziału 8.7	576
8.8.	Przydzielanie i przypisywanie rejestrów	576
8.8.1.	Globalny przydział rejestrów	577
8.8.2.	Liczniki użyć	577
8.8.3.	Przypisywanie rejestrów dla pętli zewnętrznych	580
8.8.4.	Przydział rejestrów przez kolorowanie grafu	580
8.8.5.	Ćwiczenia do podrozdziału 8.8	581
8.9.	Dobór rozkazów przez przekształcanie drzewa	581
8.9.1.	Schematy translacji drzew	582
8.9.2.	Generowanie kodu przez kafelkowanie drzewa wejściowego	584
8.9.3.	Dopasowywanie wzorców przez parsing	587
8.9.4.	Procedury kontroli semantycznej	589
8.9.5.	Ogólne dopasowywanie drzew	589
8.9.6.	Ćwiczenia do podrozdziału 8.9	591
8.10.	Generowanie optymalnego kodu dla wyrażeń	591
8.10.1.	Liczby Ershova	591
8.10.2.	Generowanie kodu na podstawie etykietowanego drzewa wyrażenia	592
8.10.3.	bliczanie wyrażeń przy niedostatecznej liczbie rejestrów	595
8.10.4.	Ćwiczenia do podrozdziału 8.10	597
8.11.	Generowanie kodu przy użyciu programowania dynamicznego	597
8.11.1.	Przetwarzanie po kolei	598
8.11.2.	Algorytm programowania dynamicznego	599
8.11.3.	Ćwiczenia do podrozdziału 8.11	602
8.12.	Podsumowanie	602
8.13.	Bibliografia	603
9.	Optymalizacje niezależne od typu procesora	607
9.1.	Główne źródła optymalizacji	608
9.1.1.	Przyczyny nadmiarowości	608
9.1.2.	Bieżący przykład: Quicksort	609
9.1.3.	Transformacje zachowujące semantykę	611
9.1.4.	Globalne wspólne podwyrażenia	612
9.1.5.	Propagacja kopii	614
9.1.6.	Usuwanie martwego kodu	615
9.1.7.	Przemieszczenie kodu	616
9.1.8.	Zmienne indukcyjne i redukcja mocy	616
9.1.9.	Ćwiczenia do podrozdziału 9.1	619

9.2.	Wprowadzenie do analizy przepływu danych	621
9.2.1.	Abstrakcja przepływu danych	621
9.2.2.	Schemat analizy przepływu danych	623
9.2.3.	Schematy przepływu danych dla bloków podstawowych	625
9.2.4.	Definicje osiąające	626
9.2.5.	Analiza żywotności zmiennych	633
9.2.6.	Wyrażenia dostępne	635
9.2.7.	Podsumowanie podrozdziału 9.2	639
9.2.8.	Ćwiczenia do podrozdziału 9.2	640
9.3.	Podstawy analizy przepływu danych	642
9.3.1.	Półkratki	643
9.3.2.	Funkcje transferu	648
9.3.3.	Algorytm iteracyjny dla ogólnego szkieletu	650
9.3.4.	Sens rozwiązania przepływu danych	653
9.3.5.	Ćwiczenia do podrozdziału 9.3	656
9.4.	Propagacja stałych	657
9.4.1.	Wartości przepływu danych dla szkieletu propagacji stałych	658
9.4.2.	Funkcja spotkania dla szkieletu propagacji stałych	659
9.4.3.	Funkcje transferu dla szkieletu propagacji stałych	659
9.4.4.	Monotoniczność szkieletu propagacji stałych	660
9.4.5.	Niedystrybutywność szkieletu propagacji stałych	660
9.4.6.	Interpretacja wyników	662
9.4.7.	Ćwiczenia do podrozdziału 9.4	663
9.5.	Eliminowanie częściowej nadmiarowości	664
9.5.1.	Źródła nadmiarowości	665
9.5.2.	Czy możliwe jest wyeliminowanie całej nadmiarowości?	667
9.5.3.	Problem opóźniającego przemieszczenia kodu	669
9.5.4.	Częściowa nadmiarowość	670
9.5.5.	Antycypowanie wyrażeń	670
9.5.6.	Algorytm opóźniającego przemieszczenia kodu	671
9.5.7.	Ćwiczenia do podrozdziału 9.5	681
9.6.	Pętle w grafach przepływu	682
9.6.1.	Dominatory	682
9.6.2.	Porządkowanie w głąb	686
9.6.3.	Krawędzie w drzewie rozpinającym w głąb	688
9.6.4.	Krawędzie zwrotne i redukowalność	690
9.6.5.	Głębokość grafu przepływu	691
9.6.6.	Pętle naturalne	691
9.6.7.	Tempo zbieżności iteracyjnych algorytmów przepływu danych	693
9.6.8.	Ćwiczenia do podrozdziału 9.6	696
9.7.	Analiza oparta na regionach	698
9.7.1.	Regiony	699
9.7.2.	Hierarchie regionów dla redukowalnych grafów przepływu	700

9.7.3.	Wprowadzenie do analizy opartej na regionach	703
9.7.4.	Konieczne założenia dotyczące funkcji transferu	704
9.7.5.	Algorytm dla analizy opartej na regionach	706
9.7.6.	Obsługa nieredukowalnych grafów przepływu	710
9.7.7.	Ćwiczenia do podrozdziału 9.7	712
9.8.	Analiza symboliczna	713
9.8.1.	Afiniczne wyrażenia zmiennych referencyjnych	713
9.8.2.	Formułowanie problemu przepływu danych	717
9.8.3.	Analiza symboliczna oparta na regionach	720
9.8.4.	Ćwiczenia do podrozdziału 9.8	725
9.9.	Podsumowanie	726
9.10.	Bibliografia	729
10.	Równoległość na poziomie instrukcji	733
10.1.	Architektury procesorów	734
10.1.1.	Potoki instrukcji i opóźnienia rozgałęzień	734
10.1.2.	Wykonywanie potokowe	735
10.1.3.	Zlecanie wielu instrukcji	736
10.2.	Ograniczenia szeregowania wykonania kodu	737
10.2.1.	Zależność danych	737
10.2.2.	Wyszukiwanie zależności między dostęпами do pamięci	738
10.2.3.	Kompromis między wykorzystaniem rejestrów i równoległością	740
10.2.4.	Kolejność faz alokacji rejestrów i szeregowania kodu	742
10.2.5.	Zależność sterowania	743
10.2.6.	Wsparcie dla wykonania spekulatywnego	744
10.2.7.	Podstawowy model maszyny	746
10.2.8.	Ćwiczenia do podrozdziału 10.2	747
10.3.	Szeregowanie wykonania dla bloków podstawowych	749
10.3.1.	Grafy zależności danych	749
10.3.2.	Szeregowanie listowe bloków podstawowych	751
10.3.3.	Priorytetowy porządek topologiczny	752
10.3.4.	Ćwiczenia do podrozdziału 10.3	753
10.4.	Globalne szeregowanie kodu	754
10.4.1.	Elementarne przemieszczanie kodu	755
10.4.2.	Przemieszczanie kodu w górę	757
10.4.3.	Przemieszczanie kodu w dół	758
10.4.4.	Uaktualnianie zależności danych	760
10.4.5.	Globalne algorytmy szeregowania	760
10.4.6.	Zaawansowane techniki przemieszczania kodu	764
10.4.7.	Interakcja ze schedulerami dynamicznymi	765
10.4.8.	Ćwiczenia do podrozdziału 10.4	765
10.5.	Potokowanie programowe	766
10.5.1.	Wprowadzenie	766
10.5.2.	Potokowanie programowe dla pętli	769
10.5.3.	Alokacja rejestrów i generowanie kodu	771

10.5.4. Pętle Do-Across	772
10.5.5. Cele i ograniczenia potokowania programowego	773
10.5.6. Algorytm potokowania programowego	777
10.5.7. Szeregowanie acyklicznych grafów zależności danych	778
10.5.8. Szeregowanie cyklicznych grafów zależności	779
10.5.9. Usprawnienia algorytmów potokowania	786
10.5.10. Modularne rozszerzanie zmiennych	787
10.5.11. Instrukcje warunkowe	790
10.5.12. Wsparcie sprzętowe dla potokowania programowego	791
10.5.13. Ćwiczenia do podrozdziału 10.5	791
10.6. Podsumowanie	793
10.7. Bibliografia	795
11. Optymalizacja pod kątem równoległości i lokalności	797
11.1. Pojęcia podstawowe	800
11.1.1. Wieloprocessorowość	800
11.1.2. Równoległość w aplikacjach	802
11.1.3. Równoległość na poziomie pętli	804
11.1.4. Lokalność danych	805
11.1.5. Wprowadzenie do teorii transformacji afinicznych	807
11.2. Mnożenie macierzy: pogłębiony przykład	811
11.2.1. Algorytm mnożenia macierzy	812
11.2.2. Optymalizacje	814
11.2.3. Interferencja cache	817
11.2.4. Ćwiczenia do podrozdziału 11.2	818
11.3. Przestrzeń iteracji	818
11.3.1. Konstruowanie przestrzeni iteracji z gniazda pętli	818
11.3.2. Kolejność wykonywania gniazd pętli	821
11.3.3. Postać macierzowa nierówności	821
11.3.4. Uwzględnianie stałych symbolicznych	822
11.3.5. Kontrolowanie kolejności wykonania	823
11.3.6. Zmiana osi	827
11.3.7. Ćwiczenia do podrozdziału 11.3	829
11.4. Afiniczne indeksy tablic	831
11.4.1. Dostępów afiniczne	831
11.4.2. Dostęp afiniczny i nieafiniczny w praktyce	832
11.4.3. Ćwiczenia do podrozdziału 11.4	833
11.5. Ponowne użycie danych	834
11.5.1. Rodzaje ponownego użycia	835
11.5.2. Samodzielne ponowne użycie	836
11.5.3. Samodzielne przestrzenne użycie ponowne	839
11.5.4. Grupowe użycie ponowne	841
11.5.5. Ćwiczenia do podrozdziału 11.5	844
11.6. Analiza zależności danych dla tablicy	845
11.6.1. Definicja zależności danych między dostęпами do tablic	846
11.6.2. Programowanie całkowitoliczbowe (liniowe)	847

11.6.3.	Test NWD	848
11.6.4.	Heurystyki dla całkowitoliczbowego programowania liniowego	850
11.6.5.	Rozwiązywanie ogólnych problemów programowania całkowitoliczbowego	854
11.6.6.	Podsumowanie podrozdziału 11.6	856
11.6.7.	Ćwiczenia do podrozdziału 11.6	856
11.7.	Wyszukiwanie równoległości niewymagającej synchronizacji	858
11.7.1.	Przykład wstępny	858
11.7.2.	Afiniczne podziały w przestrzeni	861
11.7.3.	Ograniczenia podziału w przestrzeni	862
11.7.4.	Rozwiązywanie ograniczeń podziału w przestrzeni	865
11.7.5.	Prosty algorytm generowania kodu	869
11.7.6.	Eliminowanie pustych iteracji	872
11.7.7.	Eliminowanie warunków z najbardziej wewnętrznych pętli	874
11.7.8.	Transformacje kodu źródłowego	877
11.7.9.	Ćwiczenia do podrozdziału 11.7	881
11.8.	Synchronizacja między pętlami równoległymi	883
11.8.1.	Stała liczba operacji synchronizujących	883
11.8.2.	Grafy zależności programu	884
11.8.3.	Czas hierarchiczny	887
11.8.4.	Algorytm zrównoleglania	889
11.8.5.	Ćwiczenia do podrozdziału 11.8	890
11.9.	Potokowanie	891
11.9.1.	Czym jest potokowanie?	891
11.9.2.	Sukcesywna nadrelaksacja (Successive Over-Relaxation – SOR): przykład praktyczny	893
11.9.3.	W pełni przestawialne pętle	894
11.9.4.	Potokowanie w pełni przestawialnych pętli	896
11.9.5.	Teoria ogólna	897
11.9.6.	Ograniczenia podziału w czasie	898
11.9.7.	Rozwiązywanie ograniczeń podziału w czasie przy użyciu lematu Farkasa	902
11.9.8.	Transformacje kodu	905
11.9.9.	Równoległość z minimalną synchronizacją	909
11.9.10.	Ćwiczenia do podrozdziału 11.9	912
11.10.	Optymalizowanie lokalności	914
11.10.1.	Lokalność czasowa danych obliczanych	914
11.10.2.	Kontrakcja tablic	915
11.10.3.	Przeplatanie partycji	918
11.10.4.	Zbieranie wszystkiego razem	922
11.10.5.	Ćwiczenia do podrozdziału 11.10	923
11.11.	Inne zastosowania transformacji afinicznych	924
11.11.1.	Maszyny z pamięcią rozproszoną	924
11.11.2.	Procesory z jednoczesnym zlecaniem rozkazów	925
11.11.3.	Maszyny wektorowe i SIMD	925
11.11.4.	Wczesny odczyt danych	926
11.12.	Podsumowanie	928
11.13.	Bibliografia	930

12. Analiza międzyproceduralna	935
12.1. Podstawowe pojęcia	936
12.1.1. Grafy wywołań	936
12.1.2. Wrażliwość na kontekst	938
12.1.3. Łańcuchy wywołań	940
12.1.4. Analiza kontekstowa oparta na klonowaniu	942
12.1.5. Analiza kontekstowa oparta na podsumowaniu	944
12.1.6. Ćwiczenia do podrozdziału 12.1	946
12.2. Dlaczego potrzebna jest analiza międzyproceduralna?	948
12.2.1. Wywołania metod wirtualnych	948
12.2.2. Analiza aliasowania wskaźników	949
12.2.3. Zrównoleglanie	949
12.2.4. Wykrywanie błędów i podatności na ataki	949
12.2.5. SQL injection	950
12.2.6. Przepełnienie bufora	952
12.3. Logiczna reprezentacja przepływu danych	953
12.3.1. Wprowadzenie do Datalogu	954
12.3.2. Reguły Datalogu	955
12.3.3. Predykaty intensjonalne i ekstensjonalne	956
12.3.4. Wykonywanie programów Datalogu	959
12.3.5. Inkrementalne przetwarzanie programów Datalogu	960
12.3.6. Problematiczne reguły Datalogu	963
12.3.7. Ćwiczenia do podrozdziału 12.3	964
12.4. Prosty algorytm analizy wskaźników	966
12.4.1. Dlaczego analiza wskaźników jest trudna	966
12.4.2. Model dla wskaźników i referencji	967
12.4.3. Niewrażliwość na przepływ sterowania	968
12.4.4. Sformułowanie problemu w Datalogu	969
12.4.5. Wykorzystanie informacji o typach	971
12.4.6. Ćwiczenia do podrozdziału 12.4	973
12.5. Analiza międzyproceduralna niewrażliwa na kontekst	974
12.5.1. Efekty wywołania metody	974
12.5.2. Odkrywanie grafu wywołań w Datalogu	976
12.5.3. Dynamiczne ładowanie i refleksja	977
12.5.4. Ćwiczenie do podrozdziału 12.5	978
12.6. Analiza wskaźników z uwzględnieniem kontekstu	978
12.6.1. Konteksty i łańcuchy wywołań	979
12.6.2. Dodawanie kontekstu do reguł Datalogu	982
12.6.3. Dodatkowe spostrzeżenia dotyczące wrażliwości	983
12.6.4. Ćwiczenia do podrozdziału 12.6	983
12.7. Implementacja Datalogu przez BDD	983
12.7.1. Binarne diagramy decyzyjne	984
12.7.2. Przekształcenia BDD	986
12.7.3. Reprezentowanie relacji przy użyciu BDD	987
12.7.4. Operacje na relacjach jako operacje na BDD	988
12.7.5. Wykorzystanie BDD w analizie miejsc wskazywanych	991
12.7.6. Ćwiczenia do podrozdziału 12.7	991
12.8. Podsumowanie	992
12.9. Bibliografia	995

A. Pełny front-end kompilatora	999
A.1. Język źródłowy	999
A.2. Main	1001
A.3. Analizator leksykalny	1001
A.4. Tabele symboli oraz typy	1004
A.5. Kod pośredni dla wyrażeń	1005
A.6. Kod skaczący dla wyrażeń logicznych	1008
A.7. Kod pośredni dla instrukcji	1012
A.8. Parser	1016
A.9. Budowanie front-endu kompilatora	1021
B. Znajdowanie rozwiązań liniowo niezależnych	1023
Indeks	1027

Przedmowa

W czasie, który upłynął od roku 1986, roku pierwszego wydania tej książki, świat projektowania kompilatorów znacząco się zmienił. Ewolucja języków programowania stworzyła nowe problemy. Architektury komputerów oferują dziś bogactwo zasobów, które projektant kompilatora powinien, a w zasadzie musi wykorzystać. Być może najbardziej interesujące jest to, że szanowane techniki optymalizowania kodu znalazły zastosowania poza kompilatorami. Są dziś używane w narzędziach wyszukujących błędy, a co najważniejsze, luki zabezpieczeń w już istniejącym oprogramowaniu. Zarazem większość technologii „przodowych” – gramatyki, wyrażenia regularne, parsery i translatory sterowane składnią – nadal jest w szerokim użyciu.

Tym samym nasza filozofia prezentowana w poprzednich wersjach tej książki się nie zmieniła. Zdajemy sobie sprawę, że bardzo nieliczni spośród czytelników będą tworzyć, lub choćby utrzymywać, kompilatory dla któregoś z głównych języków programowania. Jednak modele, teoria i algorytmy powiązane z kompilatorami mogą być stosowane w szerokim zakresie problemów projektowania i rozwijania oprogramowania. Dlatego szczególnie wyróżniamy kwestie do rozstrzygnięcia, które są najczęściej spotykane przy projektowaniu procesorów języków, niezależnie od języka źródłowego czy maszyny docelowej.

Korzystanie z książki

Opanowanie całości czy choć większości materiału z tej książki wymaga co najmniej dwóch kwartałów, a może nawet dwóch semestrów. Typowe podejście polega na przedstawieniu pierwszej połowy w ramach wykładu podstawowego, drugą zaś połowę tematyki książki – optymalizowanie kodu – na poziomie dyplomowym lub pośrednim. Oto konspekt poszczególnych rozdziałów:

W **rozdziale 1** zawarto materiały motywujące, a ponadto przedstawiono w nim kilka podstawowych zagadnień architektury komputerów i zasady języków programowania.

W **rozdziale 2** pokazano projektowanie miniaturowego kompilatora i wprowadzono wiele ważnych koncepcji, które zostaną rozwinięte w kolejnych rozdziałach. Sam kompilator został w całości zamieszczony w dodatku na końcu książki.

W **rozdziale 3** zawarto omówienie analizy leksykalnej, wyrażen regularnych, automatów skończonych i narzędzi generujących lekserzy. Materiał ten jest podstawą do przetwarzania tekstów dowolnego rodzaju.

W **rozdziale 4** zaprezentowano główne metody parsingu: zstępujące (metoda zejść rekurencyjnych LL) i wstępujące (LR i jej warianty).

W **rozdziale 5** wprowadzono podstawowe koncepcje definicji kierowanych składni i translacji sterowanej składnią.

W **rozdziale 6** rozwinięto teorię z rozdziału 5 i pokazano, jak można ją wykorzystać do generowania kodu pośredniego dla typowego języka programowania.

W **rozdziale 7** skupiono się na środowiskach wykonawczych, ze szczególnym naciskiem na zarządzanie stosem w czasie wykonania i mechanizmy odśmiecania pamięci.

W **rozdziale 8** omówiono generowanie kodu wynikowego. Obejmuje ono konstruowanie bloków podstawowych, generowanie kodu dla wyrażeń i bloków podstawowych oraz techniki alokowania rejestrów.

W **rozdziale 9** wprowadzono technologie optymalizacji kodu, w tym grafy przepływu, problemy przepływu danych i iteracyjne algorytmy rozwiązywania tych problemów.

W **rozdziale 10** zaprezentowano optymalizacje na poziomie instrukcji. Główny nacisk został tu położony na możliwości wydobywania równoległości z małych sekwencji instrukcji i szeregowania ich na pojedynczych procesorach, które są w stanie wykonywać więcej niż jedną czynność naraz.

W **rozdziale 11** omówiono wykrywanie i wykorzystywanie równoległości w większej skali. W tym miejscu skupiono uwagę na programach numerycznych mogących zawierać wiele ciasnych pętli przebiegających wielowymiarowe tablice.

W **rozdziale 12** zajęto się analizami międzyproceduralnymi. Omówiono tu analizy wskaźników, aliasowania oraz przepływu danych, uwzględniające sekwencje wywołań procedur, które osiągną dany punkt w kodzie.

Wykłady oparte na materiale zawartym w tej książce były prowadzone na uniwersytetach Columbia, Harvard i Stanford. Na Uniwersytecie Columbia regularnie proponowany jest wykład dla pierwszego roku studiów wyższego poziomu na temat języków programowania i translatorów, wykorzystujący materiał z pierwszych ośmiu rozdziałów. Cechę wyróżniającą tego wykładu stanowi trwający semestr projekt, w którym studenci w małych zespołach pracują nad utworzeniem i implementacją prostego języka własnego projektu. Tworzone przez nich języki obejmują wielką różnorodność zastosowań, w tym obliczenia kwantowe, syntezywanie muzyki, grafikę komputerową, gry, operacje na macierzach i wiele innych obszarów. Do zbudowania swoich własnych kompilatorów studenci wykorzystują generatory komponentów kompilatorów, takie jak ANTLR, Lex lub Yacc, oraz techniki translacji sterowanej składnią omówione w rozdziałach 2 i 5. Następujący później zaawansowany wykład skupia się na materiałach rozdziałów 9–12, z wyróżnieniem generowania i optymalizacji kodu dla nowoczesnych maszyn, w tym procesorów sieciowych i architektur wieloprocessorowych.

Na Uniwersytecie Stanforda kwartalny wykład wprowadzający obejmuje w przybliżeniu materiał z rozdziałów 1–8, choć znajduje się w nim wprowadzenie do globalnych technik optymalizacyjnych z rozdziału 9. Drugi wykład obejmuje treść rozdziałów 9–12 oraz bardziej zaawansowany materiał dotyczący odśmie-

ciania pamięci z rozdziału 7. Studenci wykorzystują opracowany na miejscu, oparty na Javie, system o nazwie Joeq do implementowania algorytmów analizy przepływu danych.

Wymagania wstępne

Czytelnik powinien dysponować pewnym „wyrafinowaniem informatycznym”, obejmującym co najmniej zaawansowany wykład na temat programowania oraz wykłady ze struktur danych i matematyki dyskretnej. Przydatna będzie także znajomość kilku różnych języków programowania.

Ćwiczenia

W książce zawarto rozbudowane ćwiczenia, po kilka dla niemal każdego podrozdziału. Trudniejsze ćwiczenia lub ich części zostały wyróżnione wykrzyknikiem. Najtrudniejsze ćwiczenia oznaczono podwójnym wykrzyknikiem.

Podziękowania

Jon Bentley dostarczył wyczerpujących komentarzy o wielu rozdziałach wcześniejszego szkicu tej książki. Pomocne komentarze i poprawki dostarczyli również (w kolejności alfabetycznej): 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 oraz Jia Zeng. Jesteśmy bardzo wdzięczni za pomoc tych wszystkich osób. Wina za wszelkie pozostawione błędy oczywiście spoczywa na nas.

Dodatkowo Monica chciałaby podziękować swoim koleżankom i kolegom z zespołu kompilatora SUIF za 18-letnią naukę kompilowania, są to: Gerald Aigner, Dzintars Avots, Saman Amarasinghe, Jennifer Anderson, Michael Carbin, Gerald Cheong, Amer Diwan, Robert French, Anwar Ghuloum, Mary Hall, John Hennessy, David Heine, Shih-Wei Liao, Amy Lim, Benjamin Livshits, Michael Martin, Dror Maydan, Todd Mowry, Brian Murphy, Jerrey 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 oraz Michael Wolf.

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

Czerwiec 2006 r.