Macmillan Computer Science Series

Consulting Editor

Professor F.H. Sumner, University of Manchester

S. T. Allworth and R. N. Zobel, *Introduction to Real-time Software Design, second edition*

Ian O. Angell and Gareth Griffith, *High-resolution Computer Graphics Using FORTRAN 77*

Ian O. Angell and Gareth Griffith, *High-resolution Computer Graphics Using Pascal*

M. Azmoodeh, *Abstract Data Types and Algorithms*

C. Bamford and P. Curran, *Data Structures, Files and Databases*

Philip Barker, *Author Languages for CAL*

A. N. Barrett and A. L. Mackay, *Spatial Structure and the Microcomputer*

R. E. Berry, B. A. E. Meekings and M. D. Soren, *A Book on C, second edition*

G. M. Birtwistle, *Discrete Event Modelling on Simula*

B. G. Blundell, C. N. Daskalakis, N. A. E. Heyes and T. P. Hopkins, *An Introductory Guide to Silvar Lisco and HILO Simulators*

B. G. Blundell and C. N. Daskalakis, *Using and Administering an Apollo Network*

T. B. Boffey, *Graph Theory in Operations Research*

Richard Bornat, *Understanding and Writing Compilers*

Linda E. M. Brackenbury, *Design of VLSI Systems – A Practical Introduction*

J. K. Buckle, *Software Configuration Management*

W. D. Burnham and A. R. Hall, *Prolog Programming and Applications*

P. C. Capon and P. J. Jinks, *Compiler Engineering Using Pascal*

J. C. Cluley, *Interfacing to Microprocessors*

J. C. Cluley, *Introduction to Low Level Programming for Microprocessors*

Robert Cole, *Computer Communications, second edition*

Derek Coleman, *A Structured Programming Approach to Data*

Andrew J. T. Colin, *Fundamentals of Computer Science*

Andrew J. T. Colin, *Programming and Problem-solving in Algol 68*

S. M. Deen, *Fundamentals of Data Base Systems*

S. M. Deen, *Principles and Practice of Database Systems*

Tim Denvir, *Introduction to Discrete Mathematics for Software Engineering*

P. M. Dew and K. R. James, *Introduction to Numerical Computation in Pascal*

M. R. M. Dunsmuir and G. J. Davies, *Programming the UNIX System*

D. England et al., *A Sun User's Guide*

K. C. E. Gee, *Introduction to Local Area Computer Networks*

J. B. Gosling, *Design of Arithmetic Units for Digital Computers*

M. G. Hartley, M. Healey and P. G. Depledge, *Mini and Microcomputer Systems*

Roger Hutty, *Z80 Assembly Language Programming for Students*

Roland N. Ibbett and Nigel P. Topham, *Architecture of High Performance Computers, Volume I*

Roland N. Ibbett and Nigel P. Topham, *Architecture of High Performance Computers, Volume II*

Patrick Jaulent, *The 68000 – Hardware and Software*

P. Jaulent, L. Baticle and P. Pillot, *68020–30 Microprocessors and their Coprocessors*

J. M. King and J. P. Pardoe, *Program Design Using JSP – A Practical Introduction*

continued overleaf

E. V. Krishnamurthy, *Introductory Theory of Computer Science*
 V. P. Lane, *Security of Computer Based Information Systems*
 Graham Lee, *From Hardware to Software – an introduction to computers*
 A. M. Lister and R. D. Eager, *Fundamentals of Operating Systems, fourth edition*
 Tom Manns and Michael Coleman, *Software Quality Assurance*
 G. P. McKeown and V. J. Rayward-Smith, *Mathematics for Computing*
 Brian Meek, *Fortran, PL/I and the Algols*
 A. Mével and T. Guéguen, *Smalltalk-80*
 Barry Morrell and Peter Whittle, *CP/M 80 Programmer's Guide*
 Derrick Morris, *System Programming Based on the PDP11*
 Y. Nishinuma and R. Espesser, *UNIX – First contact*
 Pim Oets, *MS-DOS and PC-DOS – A Practical Guide, second edition*
 Christian Queinnec, *LISP*
 E. J. Redfern, *Introduction to Pascal for Computational Mathematics*
 Gordon Reece, *Microcomputer Modelling by Finite Differences*
 W. P. Salman, O. Tisserand and B. Toulout, *FORTH*
 L. E. Scales, *Introduction to Non-linear Optimization*
 Peter S. Sell, *Expert Systems – A Practical Introduction*
 A. G. Sutcliffe, *Human-Computer Interface Design*
 Colin J. Theaker and Graham R. Brookes, *A Practical Course on Operating Systems*
 M. R. Tolhurst et al., *Open Systems Interconnection*
 J-M. Trio, *8086-8088 Architecture and Programming*
 M. J. Usher, *Information Theory for Information Technologists*
 B. S. Walker, *Understanding Microprocessors*
 Peter J. L. Wallis, *Portable Programming*
 Colin Walls, *Programming Dedicated Microprocessors*
 I. R. Wilson and A. M. Addyman, *A Practical Introduction to Pascal – with BS6192, second edition*

Non-series

Roy Anderson, *Management, Information Systems and Computers*
 I. O. Angell, *Advanced Graphics with the IBM Personal Computer*
 J. E. Bingham and G. W. P. Davies, *A Handbook of Systems Analysis, second edition*
 J. E. Bingham and G. W. P. Davies, *Planning for Data Communications*
 B. V. Cordingley and D. Chamund, *Advanced BASIC Scientific Subroutines*
 N. Frude, *A Guide to SPSS/PC+*
 Barry Thomas, *A PostScript Cookbook*

Compiler Engineering Using Pascal

P. C. Capon and P. J. Jinks

Department of Computer Science
University of Manchester

M
MACMILLAN

© P. C. Capon and P. J. Jinks 1988

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provisions of the Copyright Act 1956 (as amended), or under the terms of any licence permitting limited copying issued by the Copyright Licensing Agency, 33–4 Alfred Place, London WC1E 7DP.

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

First published 1988

Published by
MACMILLAN EDUCATION LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives throughout the world

British Library Cataloguing in Publication Data

Capon, P. C. (Peter C), *1944–*

Computer engineering using Pascal.—
(Macmillan computer science series).

1. Computer systems. Programming languages:
Pascal language. Compilers

I. Title II. Jinks, P. J. (Peter J *1955–*
005.4'53

ISBN 978-0-333-47155-5

ISBN 978-1-349-10401-7 (eBook)

DOI 10.1007/978-1-349-10401-7

Contents

Preface	ix
1 Introduction - What is a compiler?	1
2 A simple language and machine	3
2.1 The programming language - P	3
2.2 The simple instruction set - M	6
2.3 Example M program	8
2.4 Problems	10
3 Compiler organisation	11
3.1 Lexical analysis	12
3.2 Syntax analysis	14
3.3 Semantic analysis	18
3.4 Code generation	19
3.5 Code optimisation	21
3.6 Phases and passes	22
3.7 Problems	25
4 The P compiler	26
4.1 Lexical analysis	26
4.2 Syntax analysis	34
4.3 Semantic analysis	39
4.4 Code generation	41
4.5 Problems	51
5 Loading and running programs	52
5.1 Link-load time	52
5.2 Run time	53

5.3	An M-code interpreter for the P compiler	54
5.4	Testing and validating the compiler system	61
5.5	Problems	61
6	Language grammars	63
6.1	Grammars for describing programming languages	63
6.2	A notation for context free grammars	65
6.3	Regular expression notation	67
6.4	Formal machines to parse grammars	68
6.5	Problems	70
7	Lexical analysis	71
7.1	Lexical analysis for Pascal	71
7.2	Automatic generation of lexical analysers	72
7.3	The symbol table	79
7.4	Problems	84
8	Syntax analysis	85
8.1	Syntax analysis methods	85
8.2	Creation of an LL(1) recursive descent parser	88
	Example conversion of graph to program	91
8.3	Table driven LL(1) parsing	91
8.4	Table driven LR(1) parsing	95
	Construction of SLR parser tables	98
8.5	An automatically generated syntactic analyser for P	102
8.6	Syntax error recovery	108
8.7	Problems	110
9	Semantic analysis	111
9.1	Declarative processing	111
	Label declaration	114
	Constant definition	114
	Type definition	114
	Variable and parameter declarations	116
	Program, procedure and function declaration	116
9.2	The property list	116
9.3	Statement processing	121
9.4	Problems	122

10 Code generation	123
10.1 Instruction sets	123
10.2 A tree based code generation method	125
10.3 Problems	136
11 Compiled code: statements	137
11.1 Simple arithmetic	137
Integer	137
Integer subrange	138
Enumerated types	140
Char type	140
Overflow	140
11.2 Boolean and relational arithmetic	141
Boolean	141
Relational operations	143
11.3 Set arithmetic	144
11.4 Real and mixed arithmetic	144
Real arithmetic	146
Mixed arithmetic	147
11.5 Operation on structures	147
11.6 Control structures	149
11.7 Problems	152
12 Compiled code: run time stack organisation	154
12.1 Variable storage and local access	154
12.2 Non-local access	157
Static chaining	158
The display method	162
12.3 Procedure calling, entry and exit mechanisms	164
12.4 Results and parameters	168
Procedures and functions passed as parameters	170
12.5 Goto non-local labels	170
12.6 Problems	171
13 Compiled code: types and variables	172
13.1 Scalar types	172
13.2 Structured types	174
Records	174
Packed records	175

Arrays	176
Multi-dimensional arrays	178
Addressing vectors	180
Arrays of records	180
Packed arrays of Boolean	180
Set types	182
Files	182
13.3 Use of pointers	183
13.4 Managing the heap	185
13.5 Problems	186
14 Code optimisation	187
14.1 Problems	193
15 Final considerations for Pascal	194
15.1 Run time fault monitoring	194
15.2 Testing and validating the compiler	197
15.3 Implementation language	200
15.4 Problems	201
16 Modula-2	202
16.1 Towards a Modula-2 compiler	202
16.2 Modules	204
16.3 Processes	206
16.4 Problems	208
Bibliography	209
Index	211

Preface

This book is concerned with the implementation of computer languages. It aims to describe the design and construction of language compilers in sufficient detail for the reader to be able to implement a language. The main language considered is Pascal, but consideration is given to other languages where appropriate. It is assumed that the reader is familiar with Pascal. Knowledge of an assembly language, although not essential, will aid understanding of the compiled code sequences in the later chapters.

In chapters 1 to 5 a complete compiler for a simple language is presented. The simple language is a small subset of Pascal. This compiler is written in standard Pascal and generates code for a simple hypothetical machine. The source text of the compiler is available from the authors and may be used as a basis for experimentation and practical coursework. An interpreter for the hypothetical machine code is also provided to allow compiled programs to be run. These chapters form a self contained introduction to compilers and can be read independently from the rest of the book.

In chapters 6 to 15 the organisation of, the algorithms used in, and the form of compiled code generated by a complete Pascal system are considered in some detail. The ideas introduced in the simple compiler are developed, extended and applied to the construction of a Pascal compiler. Each phase of the compiling process is considered in turn. In the lexical phase the requirements of Pascal are discussed and an alternative method of constructing lexical analysers using LEX is also described. The organisation of the symbol table is described in the lexical phase. In the syntactic phase the creation of a recursive descent parser is considered as well as alternative table driven parsing methods. An example of the use of YACC, a specific table driven method, is given. LEX and YACC are illustrated using a revised version of the P compiler. Property lists are described in the semantic phase. The next phase to be considered is code generation. Whereas in the introductory chapters a hypothetical machine was introduced, in this part of the book the compiled code generated by a compiler for real machines is considered in sufficient detail to expose typical constraints imposed by real computers which can be overlooked in hypothetical examples. Examples are given for DEC PDP11 and Motorola MC68000 series processors. The final chapter discusses the

differences between a compiler for Pascal and one for Modula-2.

The book aims to present a systematic, engineered but practical approach to compiler writing. An attempt is made to give a balanced view of the total problem facing the compiler writer, avoiding the pitfall of devoting a disproportionate amount of space to grammatical considerations and syntax analysis, topics which are amenable to more formal treatment. The text is orientated towards practical examples and suggestions for both paper exercises and coursework on the computer are provided. At the end of this text the hope is that the reader will have firstly, a thorough comprehension of the correspondence between high level programming language statements and machine instructions. Secondly, he or she should understand how high level language constructs are transformed or translated into machine instructions. Finally, the reader should be able to tackle a compiler construction project, should the need arise.

The text is based on courses given for a number of years in the Department of Computer Science, University of Manchester. It derives principally from a course given to second year students who are familiar with programming in Pascal and who have some knowledge of computer architecture. This course comprises 20 lectures plus practical laboratory assignments and examples classes where paper exercises are undertaken. Not all the material given here would necessarily be covered in the 20 lectures; some selection of the more advanced topics would be made. If a short course is required the material in the first 5 chapters could be presented alone in about 10 lectures.

Many people have presented the compiling courses given at Manchester at various times. We would especially acknowledge the influence of the contributions of Howard Barringer, Chris Kirkham and Richard Banach on the present book. The referee also made a number of helpful suggestions.

The source of the P compilers may be obtained from the authors over the JANET network at address: *pcapon@uk.ac.man.cs* or *pjinks@uk.ac.man.cs*. Alternatively write to the authors at: Department of Computer Science, University of Manchester, Manchester M13 9PL, U.K. to obtain a copy on disc or tape for a nominal charge.