Ellis Horwood · Publishers
COMPUTERS AND THEIR APPLICATIONS

18

Pascal Implementation

The P4 Compiler

S. PEMBERTON and M. C. DANIELS

Computers and Their Applications 18
Series Editor:
BRIAN MEEK, Director, Computer Unit,
Queen Elizabeth College, University of London

PASCAL IMPLEMENTATION

STEVEN PEMBERTON and MARTIN DANIELS, Department of Computing and Cybernetics, Brighton Polytechnic

This book offers the first full exposition of the Pascal-P compiler, possibly the most widely-used Pascal compiler, which has been the basis of many Pascal systems, including the well-known UCSD (University of California, San Diego) system. It fills a gap in the literature with its full demonstration of the problems involved in compiler construction, by using an annotated text of a complete compiler.

The book also studies programming methodology through the critical appraisal of other peoples' programs, examining the problems common to programmers: that of maintaining and altering programs written by other people. A one-pass assembler and an interpreter are described, both of which can be used in the study of assemblers, and as an introduction to machine architecture.

Each chapter has been carefully set out to describe a particular aspect of the P-code system, each section discussing a particular procedure, or group of procedures, with explanations of the data-structures used. Some sections are followed by notes on suggestions for improvements, corrections, or alternative methods for comparison. The authors underline throughout the excellent design and style of the Pascal-P compiler, which by its very nature permits readily-found improvements.

The book is being issued in a special format, which comprises the main text in one volume, with the Appendices, and Compiler and Interpreter listings, separately bound, for ease of reference when studying the subject. The two will be packaged together, and cannot be sold separately.

Readership: Computer scientists in all areas; particularly those concerned with programming and software, students and all those wishing to learn more of compiler construction, writing compilers, and structured system programming.

PASCAL IMPLEMENTATION: The P4 Compiler

PASCAL IMPLEMENTATION:

The P4 Compiler

STEVEN PEMBERTON and MARTIN DANIELS

Department of Computing and Cybernetics
Brighton Polytechnic

ELLIS HORWOOD LIMITED
Publishers · Chichester

Halsted Press: a division of
JOHN WILEY & SONS
New York · Brisbane · Chichester · Toronto

First published in 1982 by

ELLIS HORWOOD LIMITED
Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England

The publisher's colophon is reproduced from James Gillison's drawing of the ancient Market Cross, Chichester.

Distributors:

Australia, New Zealand, South-east Asia:
Jacaranda-Wiley Ltd., Jacaranda Press,
JOHN WILEY & SONS INC.,
G.P.O. Box 859, Brisbane, Queensland 40001, Australia

Canada:
JOHN WILEY & SONS CANADA LIMITED
22 Worcester Road, Rexdale, Ontario, Canada.

Europe, Africa:
JOHN WILEY & SONS LIMITED
Baffins Lane, Chichester, West Sussex, England.

North and South America and the rest of the world:
Halsted Press: a division of
JOHN WILEY & SONS
605 Third Avenue, New York, N.Y. 10016, U.S.A.

British Library Cataloguing in Publication Data
Pemberton, Steven
Pascal implementation. – (Ellis Horwood series in computer science)
1. PASCAL (Computer program language)
I. Title II. Daniels, Martin
001.64'24 QA76.73P2

Library of Congress Card No. 81-20184 AACR2

ISBN 0-85312-358-6 Book (Ellis Horwood Ltd., Publishers)
ISBN 0-85312-437-X Compiler (Ellis Horwood Ltd., Publishers)
ISBN 0-470-27325-9 (Halsted Press)

Typeset in Press Roman by Ellis Horwood Ltd.
Printed in Great Britain by R. J. Acford, Chichester

Table of Contents

The sources of the Pascal program are available in machine-readable form on magnetic tape on application to the publishers.

Preface

This book is about compiler construction. But rather than the usual theoretical study, it is a case study of an actual compiling system.

The Pascal-P compiler is possibly the most widely used Pascal compiler – it has been the basis of many Pascal systems, including the well-known UCSD system, and there has even been a computer built specifically to run it. There have been many references to and articles about P-code before, but never a full exposition of it.

Studying the principles of compiler construction can be difficult if the theory is not backed up with some concrete examples of it in use. Realising this, many authors present small sections of a compiler usually for a toy language. Although this can be helpful, it often still does not fully demonstrate the problems involved, such as the problems of type compatibility, parameter passing, and so on. So this book is an attempt to fill this gap by presenting an annotated text of a complete compiler.

Another use of this book is to support the study of programming methodology. It is well known that a good way of learning how to program well is to critically read other people's programs, especially as so much of programmers' time is spent maintaining and altering programs they did not write.

Also presented with this book is a one-pass assembler and interpreter, both of which could be used in the study of assemblers, and as an introduction to machine architecture.

About the Commentary

Each chapter describes a particular aspect of the P-code system, each section discussing a particular procedure or group of procedures, sometimes preceded by an explanation of the data-structures used. References to lines of the programs are enclosed in square brackets [ ].

Usually a section is followed by notes on suggestions for improvements, corrections, or just alternative ways of doing something for comparison. Obviously in a lot of cases whether one method is better than another is a matter of taste, and sometimes the notes may appear critical. However, writing a compiler is a difficult job, and once written it could probably be improved almost indefinitely. The fact that it is possible to understand the whole of this compiler in a relatively short time is witness to its good design and style, and it is this good style that allows improvements to be easily found.

The notes have been arranged where possible to be independent of the main commentary, so that on a first reading, they can be skipped, in order to gain an understanding of the whole compiler, before going back and concentrating on details for a deeper understanding.

Sometimes points are repeated in the notes, for instance when explaining a data-structure and then later when the data-structure is used. This is to facilitate studying sections independently.

The only section of the compiler not discussed is *printtables* [676–845]. This is a procedure for the output of the compiler tables for testing purposes and therefore is in no way essential to the compiler or understanding it; it has been left as an exercise to the reader.

Terminology

Two points on terminology. To avoid repetitious phrases the word *routine* has been used to mean *procedure or function.* To avoid confusion, a type like

type *colour = (red, green, blue);*

is called an *enumeration,* while the phrase *scalar* is used to cover enumerations, subranges, integer, character, boolean, and real.

The Listings

The compiler and assembler/interpreter are as the originals, with the corrections published in *Pascal News* included. The only changes we have made are corrections to the indentation, to some comments, and to the layout of the lines.

Note that the upward arrow '↑' is printed as a carat '^'

We apologise that the listings are separate from the commentary – we would have preferred to have interspersed them, but this was done to keep the price of the book down.

References

Two essential documents that should be referred to in collaboration with this are *The Pascal User Manual and Report,* (Jensen, 1975), the two halves of which are referred to in this book as *The User Manual,* and *The Pascal Report,* and *Pascal-P Implementation Notes* (Nori, 1981), which is the official document distributed with the compiler.

ACKNOWLEDGEMENTS

To Professor Wirth for his co-operative response to our enquiries; to Mike Horwood for his friendly encouragement and enthusiasm; to Mike Smith, who after all provoked the book, and so ungrudgingly put up with us using his computer; to Derek Newson, for helping us with production; to Christine Wieting, who for no other reason than her nice nature, helped with the typing in the face of deadlines; to Mike Dickson, for uncomplainingly taking some of the weight off our shoulders while we wrote; and to the fourth year students 1981 for putting up with being taught it.

Dedication

To David Hitchin
To Carolyn, Rhiannon, William, and Timothy.

Introduction

The compiler presented here is for a close variant of Pascal known as Pascal-P. Rather than producing code for any particular machine, it produces code, that has come to be called 'P-code', for a hypothetical stack-based computer that is in many ways ideal for Pascal compilation.

Also presented here is an assembler and interpreter for P-code defining the actions for this P-machine.

Both these programs are written in Pascal, which at first sight may seem a rather incestuous relationship, but it leaves several options open to the implementor. For instance:

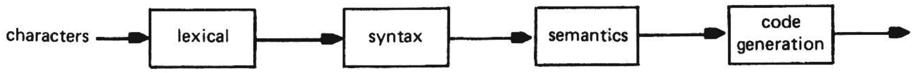
(1) Translate the compiler by hand into some other language that is available.
(2) Find someone who already has a Pascal compiler for another machine, and compile the P-compiler with this to produce a running P-compiler. Then use this new compiler to translate the P-compiler (that is, itself) into P-code.

Armed with this P-code version of the compiler, the interpreter may then be translated by hand into another language and this used to interpret the compiler.

Alternatively, a translator from P-code to an available assembly language could be written. Either way this would be easier than translating the compiler by hand.

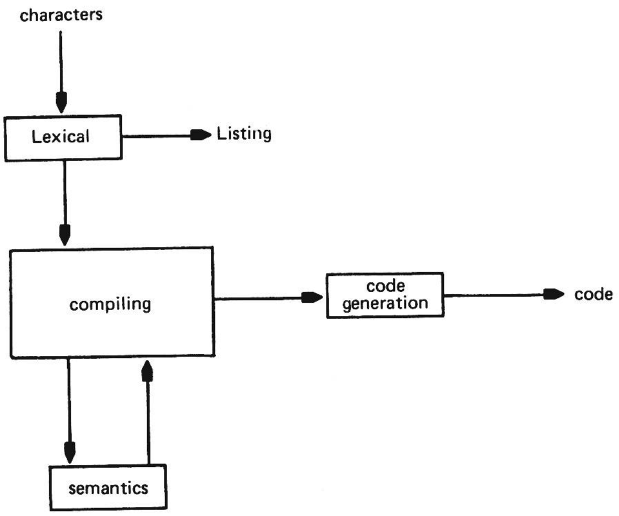
OVERVIEW OF THE COMPILER

Schematically, the information flow in the compiler is like this:



Lexical analysis processes the input characters and recognises the symbols of the language; the *syntax analyser* takes these symbols and recognises the constituent parts of the program; with the knowledge of these constituent parts the *semantic analyser* can gather information about what the program means; with this information the *code generator* can then generate equivalent code.

The actual structure of the compiler is slightly different. Central to it are *compiling procedures* that do the syntax analysis, and call the lexical analyser, semantic analyser, and code generator as sub-modules. Pictorially:



The assembler and interpreter are two separate modules; the assembler produces the code for the interpreter, which then runs the code.

It is worth mentioning here, that while the compiler was designed to be machine independent, the interpreter was written to run on a CDC machine, and so reflects many aspects of the CDC architecture, such as the word-length.

SPECIFIC AND GENERAL READING

History

The P-code compiler developed as an offshoot of an effort to produce a compiler for a CDC 6000 computer. Papers describing this development are:

Amman, U. (1974), The Method of Structured Programming Applied to the Development of a Computer, *International Computing Symposium 1973,* (Ed. Guenter, A. *et al.*) North Holland, 93–99.
Amman, U. (1981a), *The Zurich Implementation,* (see Barron, 1981).
Amman, U. (1981b), *Code Generation of a Pascal Compiler,* (see Barron, 1981).
Wirth, N. (1971), The Design of a Pascal Compiler, *Software – Practice and Experience,* 1, 309–333.

Compiling
General books on the theory of compiling are

Aho, A. V. and Ullman, J. D. (1977) *The Principles of Compiler Design,* Addison Wesley, Reading, Mass.
Bornat, R. (1979), *Understanding and Writing Compilers,* Macmillan.
Gries, D. (1971), *Compiler Construction for Digital Compilers,* Wiley, N.Y.

Syntax Analysis
Books on the specifics of syntax analysis are

Aho, A. V. and Ullman, J. D. (1973), *The Theory of Parsing, Translation, and Compiling,* I and II, Prentice Hall, N.J.
Backhouse, R. C, (1979), *The Syntax of Programming Languages,* Prentice Hall International, London.

Compilers
Books that present the code for a compiler (in all cases except the first, for a mini language) are

Aretz, F. E. J. K. *et al.* (1973), *An Algol 60 Compiler in Algol 60,* Mathematical Centre, Amsterdam.
Welsh, J. and McKeag, M. (1980), *Structured System Programming,* Prentice Hall International, London.
Wirth, N. (1976), *Algorithms + Data Structures = Programs,* Prentice Hall, N. J.
Wirth, N. (1981), *Pascal-S: A Subset and its Implementation,* (see Barron 1981).

The compilers in the last two bear a close similarity to the P-code compiler, though of course are much smaller.

Intermediate Codes
An interesting review of intermediate codes like P-code is

Elsworth, E. F. (1978), Compilation via an Intermediate Language, *Computer Journal,* 22, 3.

P-Code
The following all deal with experience with P-code.

Berry, R. E., (1978), Experience with the Pascal-P Compiler, *Software – Practice and Experience,* 8, 617–627.
Daniels, M. C. and Pemberton, S. (1980), Implementing a Pascal Compiler on an 8085a System, *Journal of Microcomputer Applications,* 4.
Shimashi, M. *et al.* (1980), An Analysis of Pascal Programs in Compiler Writing, *Software-Practice and Experience,* 10, 231–240.

Other Reading

Addyman, A. N., *et al.* (1979), A Draft Description of Pascal, *Software – Practice and Experience,* **9**, 381-424.

Barron, D. W., (Ed) (1981), *Pascal – The Language and Its Implementation,* Wiley, Chichester.

Hartmann, A. C., (1977), *A Concurrent Pascal Compiler for Minicomputers,* Springer-Verlag, Berlin.

Jensen, K., and Wirth, N., (1975), *Pascal User Manual and Report,* second ed., Springer-Verlag, Berlin.

Nori, K. V., *et al.*, (1981), *Pascal – Implementation Notes,* in (Barron, 1981).

Pemberton, S., (1980), Comments on an Error-recovery Scheme by Hartmann, *Software – Practice and Experience,* **10**, 231-240.

Welsh, J., (1978), Economic Range Checks in Pascal, *Software – Practice and Experience,* **8**, 85-97.

Welsh, J., *et al.*, (1981), *Ambiguities and Insecurities in Pascal,* in (Barron, 1981).

Part I
The Compiler