Understanding and Writing Compilers

Macmillan Computer Science Series

Consulting Editor

Professor F. H. Sumner, University of Manchester

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

Richard Bornat, *Understanding and Writing Compilers*

J. K. Buckle, *The ICL 2900 Series*

Derek Coleman, *A Structured Programming Approach to Data**

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

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

David Hopkin and Barbara Moss, *Automata**

H. Kopetz, *Software Reliability*

A. Learner and A. J. Powell, *An Introduction to Algol 68 through Problems**

A. M. Lister, *Fundamentals of Operating Systems, second edition**

Brian Meek, *Fortran, PL/I and the Algols*

Derrick Morris and Roland N. Ibbett, *The MU5 Computer System*

I. R. Wilson and A. M. Addyman, *A Practical Introduction to Pascal*

*The titles marked with an asterisk were prepared during the Consulting Editorship of Professor J. S. Rohl, University of Western Australia.

Understanding and Writing Compilers

A do-it-yourself guide

Richard Bornat

*Department of Computer Science and Statistics,
Queen Mary College,
University of London*

M

© Richard Bornat 1979

Softcover reprint of the hardcover 1st edition 1979

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, without permission.

First published 1979 by

THE MACMILLAN PRESS LTD

London and Basingstoke

Associated companies in Delhi Dublin Hong Kong Johannesburg

Lagos Melbourne New York Singapore and Tokyo

ISBN 978-0-333-21732-0

ISBN 978-1-349-16178-2 (eBook)

DOI 10.1007/978-1-349-16178-2

This book is sold subject to the standard conditions of the Net Book Agreement.

The paperback edition of this book is sold subject to the condition that it shall not, by way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the publisher's prior consent in any form of binding or cover other than that in which it is published and without a similar condition including this condition being imposed on the subsequent purchaser.

Contents

Introduction	1
Host, Object and Source Language in Examples	2
Acknowledgements	3
Section I Modular Organisation of Compilers	5
<u>1 Phases and Passes</u>	<u>8</u>
Tasks and Sub-tasks	9
Translation and Optimisation	12
Object Descriptions in the Symbol Table	13
Run-time Support	13
Source Program Errors	15
Two-pass Compilation	17
An Example of Compilation	18
<u>2 Introduction to Translation</u>	<u>22</u>
Phrases and Trees	23
Tree Walking	25
Linear Tree Representations	27
Improving the Tree Walker	29
Using the Symbol Table Descriptors	33
Translation Error Handling	35
<u>3 Introduction to Syntax Analysis</u>	<u>38</u>
Language Descriptions (Grammars)	40
Bottom-up Analysis of Expressions	42
Top-down Analysis of Statements	48
Building a Node of the Tree	50
Syntax Error Handling	52
<u>4 Lexical Analysis and Loading</u>	<u>56</u>
Reading the Source Program	57
Output for a Loader	66
After the Loader	72

Section II Translation and Crucial Code Fragments	73
Communication between Translation Procedures	75
Node format	76
<u>5 Translating Arithmetic Expressions</u>	77
Reverse Operations	80
Register Dumping	81
Tree Weighting	84
Avoiding Reverse Operations	87
Function Calls and Register Dumping	88
Other Forms of the Tree	90
Combinations of Arithmetic Types	91
<u>6 Translating Boolean Expressions</u>	94
Evaluating a Relational Expression	94
Boolean or Conditional Operators?	95
Jumping Code for Boolean Expressions	97
<u>7 Translating Statements and Declarations</u>	105
Assignment Statement	105
'While' Statement	107
BCPL 'for' Statement	109
FORTRAN 'DO' Statement	111
Compound Statements	114
ALGOL-60-like Blocks	114
Procedure Declarations	116
<u>8 Creating and Using the Symbol Table</u>	118
Table Lookup	119
Hash Addressing	123
Object Descriptions in the Symbol Table	128
Single Names with Multiple Descriptors	131
<u>9 Accessing an Element of a Data Structure</u>	136
Accessing an Element of a Vector	136
Accessing an Element of an Array	141
Record Data Structures	146
<u>10 Code Optimisation</u>	153
Net Effect and Order of Computation	158
Optimisation within a Basic Block	162
Loops and Code Motion	168
Hardware Design and Optimisation	171
Language Design and Optimisation	173

Section III Run-time Support	176
<u>11 Procedure Call and Return</u>	180
The Tasks of Procedure Call and Return Fragments	180
Layout of Procedure Call and Return Fragments	183
Recursion and the 'Efficiency' of FORTRAN	184
Simple Stack Handling	188
<u>12 Arguments and Parameters</u>	195
Different Kinds of Argument Information	196
Passing Argument Information	197
Generating Code for a Single Argument	200
Checking Stack Limits	202
The Evils of Run-time Argument Checking	203
<u>13 Environments and Closures</u>	210
An Explicit Representation of the Environment	213
The Display Vector Mechanism	219
ALGOL 60's 'call by name'	213
The Environment Link Mechanism	216
Block Structure and Data Frames	223
Non-local 'goto' Statements	226
<u>14 Efficiency, Heaps and Lifetimes</u>	228
Procedure Call with PUSH and POP Instructions	229
Addressing the Stack with a Single Register	231
Heap Storage	234
ALGOL 68, Lifetimes and Pointers	239
SIMULA 67 'classes'	244
Section IV Parsing Algorithms	248
The Dangers of Backtracking	251
<u>15 Notation and Formal Language Theory</u>	253
Languages and Sentences	254
Generating a Sentence	255
Grammars and Productions	259
The Chomsky Hierarchy	259
The Parsing Problem	261
Derivations and Sentential Forms	262
Equivalence and Ambiguity	263
Lexical Analysis and type 3 grammars	268
Warshall's Closure Algorithm	272
Context Dependency and Two-level Grammars	274

16	<u>Top-down Syntax Analysis</u>	277
	Factoring to Reduce Backtracking	279
	Removing Left Recursion	281
	One-symbol-look-ahead and One-track Analysers	285
	Context Clashes and Null Symbols	290
	A One-track Grammar is Unambiguous	295
	Error Detection and Processing	295
	Pragmatics and Tricks	300
	Interpretive Top-down Analysis	303
	Automatic Generation of Top-down Analysers	306
17	<u>Operator Precedence Analysis of Expressions</u>	308
	Determining the Precedence of Operators	310
	Numerical Operator Priorities	314
	Reducing a Phrase-pattern to a Single Symbol	316
	An Operator-precedence Grammar is Unambiguous	318
	Input States, Error Detection and Unary Operators	318
18	<u>LR(1) Syntax Analysis</u>	324
	Analyser States and the Stack	327
	Why It Works: Non-deterministic Syntax Analysis	330
	Building an SLR(1) Analyser	335
	Constructing an LALR(1) Analyser	340
	Error Detection and Tree-building Actions	344
	Error Recovery	346
	What are the True Advantages of LR(1) Analysis?	346
	Section V Interpreting and Debugging	350
19	<u>Interpreters and Interpretation</u>	354
	Interpreting the parse tree	355
	Imitative Interpretation	357
	Linearising or Virtual-machine Interpreters	359
	The Design of Languages for Interpretation	365
	Debugging with an Interpreter	366
	Combining Interpretation and Compilation	369
20	<u>Run-time Debugging Aids</u>	373
	Off-line debugging and the panic dump	375
	Interactive debugging	377
	Run-time event tracing	378
	Break points	378
	Producing Debuggable Code	379
	Doing Without the Debugger	380

Appendix A: The BCPL language	381
Appendix B: Assembly code used in examples	385
Bibliography	388
Index	390