**From Hardware to Software:
an introduction to computers**

Macmillan Computer Science Series

Consulting Editor

Professor F. H. Sumner, University of Manchester

S. T. Allworth, *Introduction to Real-time Software Design*
Ian O. Angell, *A Practical Introduction to Computer Graphics*
G. M. Birtwistle, *Discrete Event Modelling on Simula*
T. B. Boffey, *Graph Theory in Operations Research*
Richard Bornat, *Understanding and Writing Compilers*
J. K. Buckle, *The ICL 2900 Series*
J. K. Buckle, *Software Configuration Management*
Robert Cole, *Computer Communications*
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**
J. B. Gosling, *Design of Arithmetic Units for Digital Computers*
David Hopkin and Barbara Moss, *Automata**
Roger Hutton, *Fortran for Students*
Roger Hutton, *Z80 Assembly Language Programming for Students*
H. Kopetz, *Software Reliability*
Graham Lee, *From Hardware to Software: an introduction to computers*
A. M. Lister, *Fundamentals of Operating Systems, second edition**
G. P. McKeown and V. J. Rayward-Smith, *Mathematics for Computing*
Brian Meek, *Fortran, PL/I and the Algols*
Derrick Morris and Roland N. Ibbett, *The MU5 Computer System*
John Race, *Case Studies in Systems Analysis*
B. S. Walker, *Understanding Microprocessors*
Peter J. L. Wallis, *Portable Programming*
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.

From Hardware to Software

an introduction to computers

Graham Lee

*Department of Computer Science,
University of Western Australia, Perth*

© Graham Lee 1982

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

First published 1982

Reprinted (with corrections) 1983

Published by

THE MACMILLAN PRESS LTD

London and Basingstoke

Companies and representatives throughout the world

ISBN 978-0-333-24363-3

ISBN 978-1-349-16786-9 (eBook)

DOI 10.1007/978-1-349-16786-9

Contents

1.	A SIMPLE DIGITAL COMPUTER	1
1.1	Digital and analogue computers	2
1.2	The structure of a simple computer	2
1.3	Storing and adding numbers inside the computer	11
1.4	Character codes	22
1.5	Computer instructions	24
1.6	Physical means of recording information	42
2.	LOGIC FUNCTIONS AND EXPRESSIONS	46
2.1	Logic operators and expressions	46
2.2	Properties of the logic operators	49
2.3	Logic functions and truth tables	53
2.4	Deriving an expression for a function	57
2.5	Simplification of expressions	60
2.6	Functional completeness of sets of operators	66
3.	COMBINATIONAL LOGIC CIRCUITS	69
3.1	Inputs and outputs of logic circuits	69
3.2	Logic gates	70
3.3	Circuits: connecting gates together	70
3.4	Designing circuits	75
3.5	Designing a full adder	76
3.6	Multiplexors	76
3.7	Decoders	80
3.8	Parity-checking circuits	82
3.9	Code converters	86
3.10	Functional completeness of sets of gates	89
4.	PARALLEL, BINARY ADDERS	94
4.1	Ripple-carry adders	95
4.2	Carry-look-ahead adders	98
5.	PARALLEL, BINARY SUBTRACTORS	107
5.1	Subtraction and borrows	107
5.2	Subtraction circuits	108
5.3	Representation of signed integers	108
5.4	A circuit for addition and subtraction	115
6.	REGISTERS AND THE BUS	121
6.1	Flip-flops	121
6.2	Registers	127
6.3	Bus data transfers	128
7.	SHIFT REGISTERS AND COUNTER REGISTERS	132
7.1	Shift operations on bit patterns	132
7.2	Shift registers	135
7.3	Counter registers	140

8.	MULTIPLIERS	148
8.1	Multiplication of positive integers	148
8.2	A multiplication circuit	153
8.3	Efficient multiplication circuits	155
8.4	Multiplication of signed integers	161
9.	DIVIDERS	170
9.1	Integer division	170
9.2	Division of positive integers	170
9.3	A division circuit	180
9.4	Division of signed integers	181
10.	THE FUNCTION UNIT	183
10.1	The registers associated with the Function Unit	183
10.2	Function-Unit operations	185
10.3	The Accumulator	190
11.	THE MEMORY	192
11.1	Data transfers to and from the Memory	192
11.2	The registers associated with the Memory	194
11.3	Internal structure of the Memory	195
11.4	Using the Memory	199
12.	INPUT AND OUTPUT DEVICES	205
12.1	Some example peripherals	205
12.2	Simple input and output instructions	211
12.3	The registers associated with peripherals	213
12.4	Executing input and output instructions	214
13.	THE CONTROL UNIT	217
13.1	The architecture of SDC	217
13.2	A simulator for SDC	220
13.3	Control and function-select signals	222
13.4	Microprograms	225
13.5	Timing	229
14.	LOADERS: RUNNING MACHINE-CODE PROGRAMS	235
14.1	Binary loaders	235
14.2	Manual loading using the Control Panel	239
14.3	Bootstrap loaders	241
14.4	Hardware loaders	245
15.	ASSEMBLY CODE	247
15.1	Labels and symbolic addresses	248
15.2	Load-address directives	250
15.3	Symbols and values	252
15.4	Entry-address directives	252
15.5	An assembly-code programming example	253
15.6	Other assembly-code features	261
15.7	Advantages of assembly code over machine code	264
15.8	Fixed and free-format assembly codes	264

16.	ASSEMBLERS: RUNNING ASSEMBLY-CODE PROGRAMS	268
16.1	The purpose and structure of an assembler	268
16.2	A single-pass assembler for a simple assembly code	272
16.3	Structure of the symbol table	289
17.	SUBROUTINES	294
17.1	Calling, and returning from, a subroutine	294
17.2	Passing parameters and results	298
17.3	Formal assembly-code subroutines	304
17.4	A subroutine-construction example	307
18.	MACROS AND MACRO ASSEMBLERS	313
18.1	Macro definitions	313
18.2	Macro calls	314
18.3	Comparison of macros and subroutines	316
18.4	Conditional assembly	317
19.	ADDRESSING MODES	323
19.1	Indexing	323
19.2	Direct and indirect addressing	331
20.	MULTIPLE-ADDRESS INSTRUCTIONS	337
20.1	One-address instructions	337
20.2	Two-address instructions	338
20.3	Three-address instructions	339
20.4	Zero-address instructions	339
20.5	Comparison of multi-address formats	343
20.6	Multiple accumulators	344
20.7	Reverse-Polish notation	345
21.	FLOATING-POINT ARITHMETIC	354
21.1	Fixed-point representation of real numbers	354
21.2	Floating-point representation of real numbers	358
21.3	Floating-point hardware	364
22.	HIGH-LEVEL PROGRAMMING LANGUAGES	368
22.1	Comparison of high and low-level languages	368
22.2	A brief history of high-level languages	374
22.3	Defining a high-level language	375
22.4	Implementation of high-level languages	381
23.	COMPILERS AND INTERPRETERS	383
23.1	Compilers	383
23.2	The main constituents of a compiler	385
23.3	Using a compiler	389
23.4	A single-pass compiler for a Pascal subset	392
23.5	Interpreters	393
24.	SYNTAX ANALYSIS AND PARSING ALGORITHMS	397
24.1	A lexical-analysis procedure	397
24.2	An error-handling procedure	400
24.3	Dictionary procedures	401
24.4	Syntax diagrams and parsing procedures	403
24.5	A parser for a Pascal subset	407

25.	CODE GENERATION	411
25.1	The run-time store layout of the object program	411
25.2	Code generation for arithmetic expressions	413
25.3	Code generation using tree data structures	417
25.4	Code generation for statements	422
26.	OPERATING SYSTEMS	426
26.1	The open-shop system	426
26.2	The computer operator	428
26.3	The main functions of an operating system	429
26.4	Batch monitors	429
26.5	Auxiliary memory	431
26.6	Batch-processing systems	434
26.7	Interrupts and channels	437
26.8	Multiprogramming	442
26.9	Multiaccess systems	446
	Index	450

Preface

This book was born on April 16, 1976, in a village near Leeds, in the North of England. When it was barely a few chapters old, it emigrated to Perth in Western Australia, where it has grown to 26 chapters. It now returns to the old country so that it might be published.

As its title indicates, the book covers the technical hardware and software aspects of computer systems, at an introductory level. The depth and breadth of treatment render the book suitable for use on introductory undergraduate, U.K. GCE A-level and similar course in Universities, Polytechnics, Colleges and Schools. It has been developed on an introductory undergraduate course in the Department of Computer Science at the University of Western Australia, since 1978. Up to 50 lectures can be taken from the text, and tutorial material can be based on the exercises given at the end of each chapter. No prior knowledge of computers is required; all the relevant concepts and techniques are covered in the book.

The text is divided evenly between hardware and software. Chapter 1 introduces the main topics, including a hypothetical Simple Digital Computer, which is used for illustration. Chapters 2 to 13 deal with the hardware of this computer, and give complete, detailed diagrams for each component. Chapters 14 to 18 discuss simple software, and programming techniques, for this computer. Chapters 19 to 21 discuss the major differences between the hypothetical computer and real, commercially available computers, as well as the variations among the latter. Chapters 22 to 26 discuss more sophisticated software.

The book is written as an integrated whole, so that the reader can learn all computer-system's concepts from one book, rather than from separate books for hardware and software, and possibly separate books for other individual topics. However, various chapters can be read independently: chapter 1 gives an outline of the workings of a computer; chapters 2, 3 and 6 discuss logic-circuit design; chapters 15, 17 and 18 discuss assembly-code programming; chapter 22 discusses programming languages; chapter 23 discusses compilers and interpreters; and chapter 26 presents a historical overview of the development of operating systems.

The essential parts of all the software presented, including a simulator for the simple computer, an assembler, and a compiler, are described formally using the programming language Pascal. Thus the book is ideally suited for use on introductory courses run in parallel with a Pascal-programming course. However, since the software is also explained informally, lack of knowledge of Pascal would not prevent an understanding of the relevant sections. The Pascal fragments are given so that the keener student can construct a simulator for the hypothetical computer, and some software, by following the appropriate exercises.

Acknowledgements

I wish to thank: Professor F. Sumner, Consulting Editor, for many helpful comments on a preliminary draft manuscript; Professor J.S.Rohl of the Department of Computer Science, University of Western Australia, for many invaluable discussions that have influenced the form and content of the book; Tony Addyman of the University of Manchester and R.E.Berry of the University of Lancaster for several useful criticisms of the final manuscript; Lloyd Allison, C.P.Tsang, Chris Hall, Glenn Huxtable, Mike Palm and Paul Nicholson of the Department of Computer Science, and Frank O'Connor of the Western Australian Regional Computing Centre, for critical comments on preliminary and final drafts; Rosemary Lee, Joyce Fisher, Fiona Armitage and Lyn McGrath for typing the several versions that the book has gone through over the past few years; and finally those students who have provided invaluable feedback and comments.

Dedicated to my wife Rosemary, and daughter Rebecca Jane.

Graham Lee,
Perth, Western Australia, October 1981.