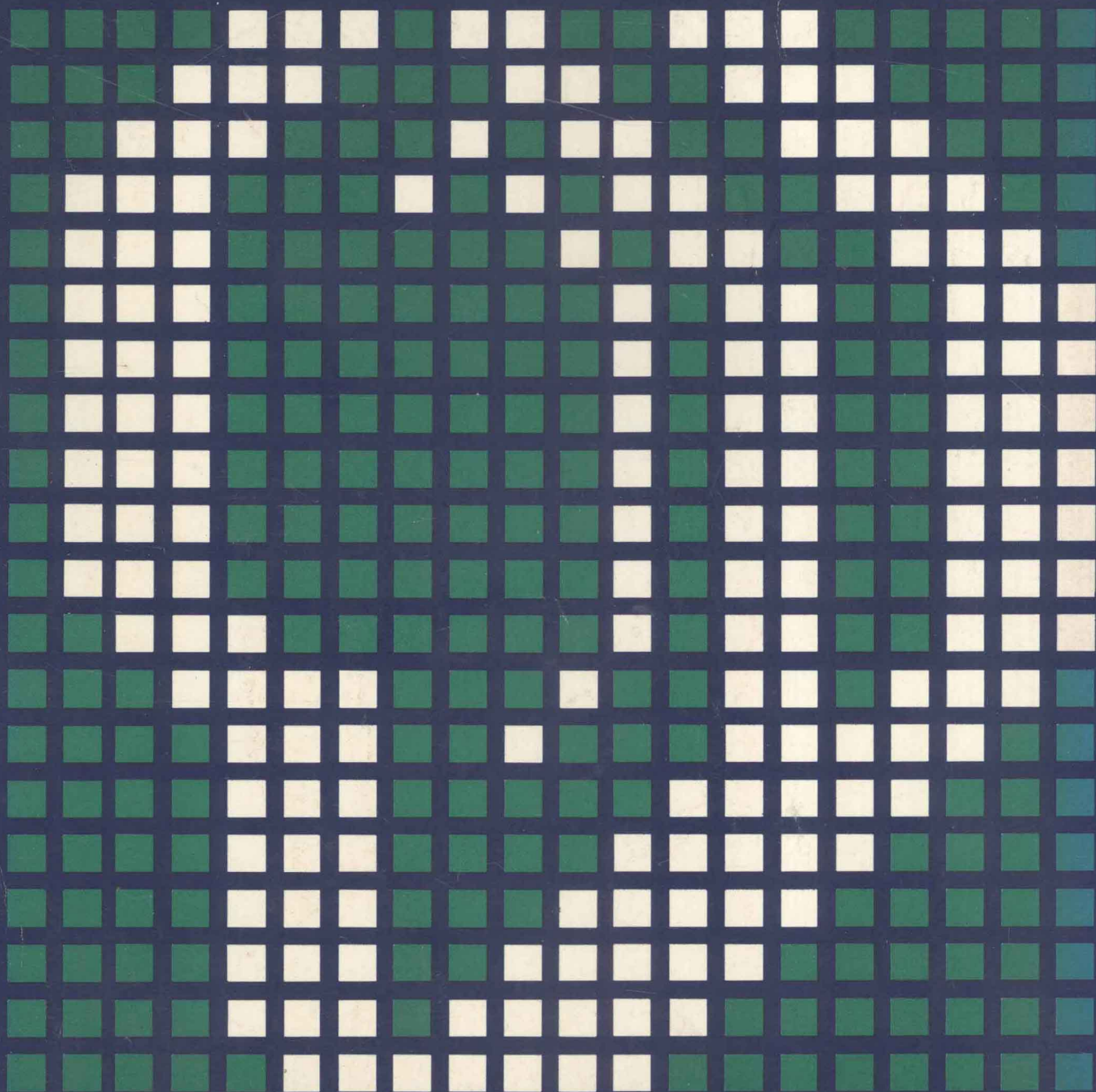


Assembly Language Programming for PDP 11 and LSI 11 Computers

an introduction to computer organization

Edouard J. Desrutels



Assembly Language Programming for PDP 11 and LSI 11 Computers

an introduction to computer organization

Edouard J. Desautels
University of Wisconsin / Madison

wcb

Wm. C. Brown Company Publishers
Dubuque, Iowa

**wcb
group**

Wm. C. Brown Chairman of the Board
Mark C. Falb Executive Vice President

Book Team

Alan R. Apt Editor
Julie A. Kennedy Production Editor
James M. McNeil Designer
Kevin J. Pruessner Design Layout Assistant
Faye Schilling Visual Research Editor
Mavis M. Oeth Permissions Editor

wcb

Wm. C. Brown Company Publishers, College Division

Lawrence E. Cremer President
David Wm. Smith Vice President/Marketing
David A. Corona Assistant Vice President/Production Development and Design
Marcia H. Stout Marketing Manager
Janis M. Machala Director of Marketing Research
William A. Moss Production Editorial Manager
Marilyn A. Phelps Manager of Design
Mary Heller Visual Research Manager

Copyright © 1982 by Wm. C. Brown Company Publishers

Library of Congress Catalog Card Number: 81-70686

ISBN: 0-697-08164-8

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Printed in the United States of America

Assembly Language Programming for PDP 11 and LSI 11 Computers

an introduction to computer organization

**This book is dedicated to my
wife, Jeannine, and my children,
Francine, Nicole, and Philip, for
their help and encouragement.**

preface

The goal of this book is to help readers who have some experience using computers to develop a deeper understanding of how they compute. The very popular PDP-11 computer (and its look-alike, the LSI-11) are used to illustrate the concepts involved.

This is a very substantial book. Don't let it overwhelm you. Among other virtues, it is essentially self-contained. This means you do not have to keep looking through the PDP-11 MACRO-11 reference manual (138 pages plus appendixes), or the PDP-11 Processor Handbook (468 pages plus appendixes), or the PDP-11 Peripherals Handbook (435 pages), or the PDP-11 Terminals and Communications Handbook (344 pages), to understand the topic at hand. One of the major advantages of having all this material integrated in one volume is that it can all be extensively cross-referenced in the index. The index is one of the most important parts of this book. The index is presented in four parts. The first two parts provide a quick reference to the PDP-11 instructions and the MACRO-11 directives. The third part consists of special character references, since these often have a critical role in the writing of computer programs. The fourth part consists of a standard but very comprehensive index. The instructions and directives are repeated in the index for the benefit of the readers may not yet appreciate these distinctions.

The material is sequenced so that the key topics can be studied even in the shortest typical quarter. Each instructor or reader can then select from among the many other topics for semester courses, honors courses, and independent (self-study) courses. Some examples of course organization are as follows.

1. Short course covering only fundamentals:
chapters 1–9; 10 from beginning through Hard-Copy Terminals; 11; 12 (macros only); selected topics from other chapters, as desired.
2. Course for those familiar with some other machine language:
chapters 1 (first two sections); 3–19.
3. Course emphasizing software concepts:
chapters 1–9; 10 and 11 (selected topics); 12; 14 and 15 (selected topics); 18–19.
4. Course emphasizing hardware concepts:
chapters 1–11; 12 and 13 (selected topics); 14; 15 (selected topics); 16–19.

Once the fundamentals have been covered, the instructor or reader can pick and choose among many essentially independent topics. Thus, in chapter 12 assembly-time conditional directives are treated independently of macros, and vice versa. Chapter 14 first discusses high speed input and output; the later sections on spooling, performance, analog data, processor traps, etc., can be included or not, as desired. Chapter 15, "Selected Topics" has a similar structure. Any of the topics may be included in a course or not, as desired.

The programming examples in the text are supplemented by programming examples in the solutions to the over 280 end-of-chapter exercises; the selected solutions, at the end of the book, cover questions that range from simple review exercises to thought-provoking problems.

We hope readers will find the bibliographic essay in the last chapter more interesting and useful than a conventional annotated bibliography.

The goal of this text is not to make readers experts at PDP-11 programming, but to lay the foundation that is essential for any one who wishes to pursue the study of computer organization, computer architecture, the design of compilers for high level languages, and many other facets of computing. This foundation will be very helpful for those who have personal computers of any kind and want to find out what makes them tick.

We assume that readers have programmed some computer in some high level language. This experience should include having used two-dimensional arrays and procedures or subroutines. Readers are therefore assumed to be familiar with the program development cycle: problem definition, data definition, algorithm selection, implementation, documentation, optimization, testing, and enhancement.

Readers must have access to a DEC PDP-11 or an LSI-11 or some system which uses either CPU (any model will do) and which provides some version of the MACRO-11 assembler. Some of the systems which could be used, besides those sold directly by DEC, include the Heath/Zenith H11 and the Terak 8510. We are convinced that merely reading about using machine language and assembly language is not a substitute for writing and running programs at this language level.

The exercises at the end of most chapters are important. Readers should take the time to at least think through an approach to solving the more demanding problems if time does not permit writing out and possibly programming the solutions. Some of the exercises introduce new ideas, new terms, new tools, and useful algorithms.

Material is presented in a carefully thought-out sequence. Readers are presented with relevant factual information and background, but not so much as to overwhelm them. Learning the details of a computer's instruction set can be as deadly as reading a dictionary if the presentation does not take human nature into account. We have partitioned topics into manageable units, interspersed with other topics, and we take advantage of opportunities to use simpler concepts before proceeding to more advanced ones. The presentation is also enlivened by newspaper and magazine articles on computer-based system malfunctions whose cause is very likely to be related directly to the topic at hand. Some of the articles discuss broader issues such as the legal challenges facing the marriage of computing and telecommunications services.

It is important to note that we first examine machine language before we approach assembly language. We recognize that the assembler is a very useful tool, and make extensive use of it in the text. But in the beginning, the assembler and the assembly process obscure what is going on at the machine level just as much as do high level languages. After a brief but illuminating venture into absolute coding in machine language, we can then appreciate and exploit symbolic programming as a means to gaining more insight into computing.

We do not advocate using machine language or assembly language as a general purpose problem-solving tool. These low level languages are like the ancient Latin and Greek that people were supposed to study as a means of gaining insight into the structure of contemporary English. For instance, the text often uses octal memory dumps, and we expect readers to be able to interpret these. These are just a means to an end—to understand what the computer was doing, stripped of all layers of software. Anyone who concludes that octal memory dumps are an efficient tool for debugging programs written in a high level language has missed the point.

Learning about computing can be enjoyable and addictive, particularly if you have sufficiently ready access to a computer so you can try all the “what if” situations that should be popping through your mind as you proceed.

Notes to the Instructor

Our practice has been to assign computer exercises beginning with the first week of class and to have one assignment immediately follow another continuously throughout the semester. Depending on the class size, the computing resources available, the magnitude of the assigned work, and the assistance provided in grading the programs in a timely fashion, from four to eight programming assignments are made in a 13–16 week semester.

Typically, the first assignment involves finding the laboratory, verifying that one can log in, that one can create and modify a small file using the on-line editor. This course has been conducted in batch-processing mode (using a self-service PDP-11); it has also been run with RT-11, RSTS, and UNIX. In those cases it is advisable to also have a bare PDP-11 for one-at-a-time hands-on use, if the multi-user system cannot be taken off the air.

A typical second assignment has students design and implement a small machine-language program, coded in absolute octal. The students use the instruction subset of chapter 2 and installation-specific instructions on loading and running this program. If a hands-on machine is available, then it can be keyed in at the console and run. For many students this may be the only chance they will ever have to completely control a computer, and this experience gives them a great deal of confidence.

The next assignment has students use a few of the PDP-11 addressing modes while writing their first MACRO-11 program. They are generally given a file with instructions on assembling, linking, and executing it. Then they are told to replace the innards with their own code and repeat the process. This is a painless way to get them introduced to seeing and

using tools such as a contingency post-mortem dump, a memory dump, and a register dump, with the setup required for their use (e.g., .MCALLs, .GLOBLs, invocations, etc.).

The next assignments can involve number representation transformations—say, in implementing a very simple two-function decimal calculator. Students may be asked to implement it using ASCII string-to-binary conversion or programmed BCD arithmetic. The more ambitious students might be asked to write the machine-language loading program (in assembly language) and to propose and implement extensions to it. Similarly, all of the other tools used in the class could themselves be the subject of assignments: students take pride in constructing their own dump programs. Many students have a mental block regarding the transformations from bits in a memory to octal digits on paper. Having them write a dump routine, perhaps adding ASCII interpretations to it, is very salutary. It further develops self-confidence and a sense of knowing what is going on, knowing that you could build up a usable system even if you had nothing but a bare machine. If hands-on access is possible, it is very instructive for students to estimate (by hand) what their program execution time should be, then devise an experiment to actually measure this time.

For the assignment just described, students would need to have access to the appropriate PDP-11 or LSI-11 Processor Handbook, or at least to the relevant instruction timing information. This textbook is otherwise sufficiently self-contained that it should not be necessary for students to buy either the Processor Handbook or the MACRO-11 Reference Manual. However, copies should be made available as references. Many of the problems given at the end of each chapter could be used as the basis for programming assignments.

Instructors sometimes regard teaching a course such as this as a way to give students an opportunity to work on very large projects. We think it is entirely inappropriate to assign large projects to be programmed in assembly language. That is the kind of experience students should be getting when using high level languages. The point of writing programs in machine and assembly language in this course should be to develop understanding of some fundamental ideas: indexing, indirection, manipulation of a stack, mastering the overflow bit, fielding an interrupt, etc. These objectives are more likely to be realized by assignment of smaller, more narrowly focussed problems, and more of them.

Careful readers will appreciate that by the time they reach the end of the text they will know how to design and implement each of the basic tools they have been using from the beginning. This should help dispel much of the mystery that shrouds computing.

I would like to thank the following reviewers for their constructive suggestions: William Bregar, Oregon State University; Linda Eshleman, Western Maryland College; Gordon Fish, Bucks County Community College; Bryan Hansche, Arizona State University; Alex Nichols, Cleveland State Community College; Michael Schneider, University of Minnesota; Abraham Silbershatz, University of Texas at Austin; and Larry Symes, University of Regina.

Edouard J. Desautels

contents

Preface xv

- 1** The Computing Context 1
 - Introduction 3
 - Software 3
 - Computers 4
 - Naked Computers 4
 - Multi-User Systems 5
 - Host and Target Computers 5
 - Computing Machines 6
 - Bits 11
 - Binary Codes 12
 - Teletype TTY 13
 - ASCII 14
 - Programmable Calculators 16
 - Analog and Hybrid Computers 16
 - Summary 18
 - Exercises 18
- 2** A Simple Hypothetical Computer 21
 - The Instruction-Fetch-Execute Cycle 24
 - The First Machine-Language Program 27
 - IPL and Boot 29
 - Program Execution 29
 - Program Loops 30
 - Self-Modifying Programs 33
 - Consequences of Stored Programs 35
 - Speed in Perspective 36
 - Summary 37
 - Exercises 38
- 3** Machine-Language Programming for a Real Computer 41
 - A Machine-Language Program 43
 - More About Binary 44
 - Octal Numbers 46
 - A Model for the PDP-11 48
 - Control Unit 48
 - PDP-11 Instruction Subset 51
 - Sample Machine-Language Programs 52
 - A Program-Loading Program 52
 - Common Errors 57
 - Instruction Formats 58
 - PDP-11 Instruction Summary 60
 - Summary 61
 - Exercises 62

| | |
|----------|---|
| 4 | A Better Way: Assembly Language 65 |
| | MACRO-11 Source Statements 68 |
| | Complete Source Programs 70 |
| | Reading an Assembly Listing 72 |
| | After Assembling, Then What Do You Do? 74 |
| | Operating Systems Considerations 74 |
| | Using RT-11 74 |
| | Running on a Multi-User System 76 |
| | Machine Language Revisited 77 |
| | System Services 77 |
| | .MCALL 78 |
| | Summary 79 |
| | Exercises 80 |
| 5 | More Hardware 83 |
| | Registers 85 |
| | An Exception 86 |
| | Renaming Registers and Other Things 86 |
| | Bytes 87 |
| | Character Codes 87 |
| | Byte Manipulations 88 |
| | More Byte Instructions 89 |
| | Single Operand Instructions 90 |
| | CLR, CLRB 90 |
| | INC, INCB 90 |
| | DEC, DECB 91 |
| | NEG, NEGB 91 |
| | Byte-Oriented Assembler Directives 91 |
| | .BYTE 91 |
| | .EVEN 92 |
| | .BLKB, .ASCII, .ASCIZ 93 |
| | Bytes and Registers 94 |
| | Conditional Branch Instructions 95 |
| | Offsets 95 |
| | Conditional Branch Subset 96 |
| | Summary 97 |
| | Exercises 98 |
| 6 | Key Addressing Modes 101 |
| | Indexing 103 |
| | Using Indexing 104 |
| | Mode and Register Fields 104 |
| | Index Mode 106 |
| | Indexing and Offsets 106 |
| | Deferred Addressing 107 |
| | Auto-Increment Mode 108 |
| | Stepping 108 |
| | Why So Many Variations? 109 |
| | .WORD and .BYTE Revisited 111 |
| | Copying 111 |
| | Odds and Ends 112 |
| | Decimal Operands 112 |
| | .RADIX 113 |
| | Immediate Operands 113 |
| | Addressing Mode Summary 114 |
| | Exercises 115 |

| | |
|----------|--|
| 7 | Computer Arithmetic 121 |
| | Negative Numbers 123 |
| | Sign Magnitude Representation 123 |
| | Two's-Complement Representation 123 |
| | Different Orderings Apply 125 |
| | Lack of Symmetry 125 |
| | Pseudo Sign Bit 125 |
| | One's-Complement Representation 126 |
| | Arithmetic Using Two's-Complement Numbers 127 |
| | Addition 127 |
| | Carry and Overflow 128 |
| | Signed Numbers 128 |
| | BVS, BVC 129 |
| | Subtraction and Comparisons 132 |
| | MOVB Revisited 132 |
| | Caution Regarding the C Bit 133 |
| | BCS, BCC, ADC, SBC 133 |
| | Sign Extension SXT 135 |
| | Dealing with Unsigned Numbers 135 |
| | All Branches for Unsigned Numbers 137 |
| | SOB 138 |
| | Comparing Signed Numbers 139 |
| | Signed and Unsigned Conditional Branches 140 |
| | TST and Testing 141 |
| | Which Branch Should I Use? 141 |
| | Summary 142 |
| | Exercises 143 |
| 8 | Subroutines and Stacks 153 |
| | Subroutines, or Necessity Is the Mother of Invention 155 |
| | Deferred-Address Addressing Modes 158 |
| | Program Counter Addressing 160 |
| | Relative Addresses 160 |
| | Immediate Operands 161 |
| | Relative Deferred Addressing 161 |
| | Deferred Indexing 162 |
| | Absolute Addressing 162 |
| | Relocation 163 |
| | Stacks 164 |
| | Stacks in Computers 165 |
| | The System Stack Pointer SP 166 |
| | How Does the System Share the Stack? 171 |
| | Rules for Using the System Stack 172 |
| | Recycling Memory with Stacks 172 |
| | Subroutines and Stacks 174 |
| | JSR PC,sub; RTS PC 174 |
| | Passing Parameters 176 |
| | Passing Values as Parameters 176 |
| | Passing Addresses as Parameters 177 |
| | More General Parameter Passing Techniques 178 |
| | Passing Arguments in the Stack 179 |
| | JSR R,sub; RTS R 180 |
| | Which Linkage Register Should I Use? 182 |
| | Suggestions for Subroutine Design 182 |

| | |
|--|-----|
| Internal versus External Subroutines | 182 |
| .GLOBL | 183 |
| Linking Externally Defined Subroutines | 184 |
| Transparency | 185 |
| Subroutine Nesting | 185 |
| What Can Go Wrong? | 188 |
| Stack Overflow | 188 |
| Stack Underflow | 188 |
| Making the Stack Deeper | 188 |
| Summary | 189 |
| Exercises | 190 |

9 The Remaining General-Purpose Instructions 199

| | |
|------------------------------|-----|
| Format Classification | 201 |
| Classification by Function | 201 |
| Logical Operations | 209 |
| NOT, COM | 209 |
| AND, BIT | 209 |
| OR, BIS | 210 |
| Bit Clear BIC | 210 |
| EXCLUSIVE-OR, XOR | 211 |
| Rotate Instructions | 212 |
| ROR and ROL | 212 |
| Condition Code Instructions | 213 |
| MFPS, MTPS | 214 |
| Arithmetic Shifts | 215 |
| ASR | 215 |
| ASL | 215 |
| Multiplication, MUL | 216 |
| Processing Numeric Data | 217 |
| Decimal-to-Binary Conversion | 218 |
| BCD Arithmetic | 218 |
| Division, DIV | 221 |
| Long Shifts; ASH, ASHC | 222 |
| ASH | 223 |
| ASHC | 223 |
| Summary | 224 |
| Exercises | 224 |
| Case Study | 227 |

10 Keyboards, Codes, and Terminals 235

| | |
|------------------------------|-----|
| Punched Cards and Paper Tape | 237 |
| Glass Teletypes | 237 |
| Keyboards | 238 |
| CRT Displays | 239 |
| Hard-Copy Terminals | 240 |
| Intelligent Terminals | 241 |
| Parity Selection | 242 |
| Half-Duplex Communication | 243 |
| Full-Duplex Communication | 243 |
| Binary Transmission Codes | 245 |
| Baud and Bits/Second | 247 |
| Summary | 248 |
| Exercises | 248 |

| | |
|-----------|--|
| 11 | Character-Oriented Input and Output 251 |
| | Data Registers, Control and Status Registers 254 |
| | Input/Output Instructions 255 |
| | I/O without I/O Instructions 256 |
| | Control and Status Bits 259 |
| | Overlapped I/O and Processing 259 |
| | Interrupt Handling 261 |
| | Processing an Interrupt 261 |
| | Interrupt Priorities 263 |
| | Single-Level Hardware Priority 263 |
| | Interrupt Handlers; RTI 264 |
| | Multi-Level Priorities 264 |
| | Clock Interrupt Handler 267 |
| | CRT Input-Interrupt Handler 267 |
| | Summary 268 |
| | Exercises 269 |
| 12 | The Assembler Revisited 271 |
| | Housekeeping Directives 273 |
| | Documentation Support 273 |
| | .LIST, .NLIST 274 |
| | Conditional Assembly 275 |
| | When Do Things Happen? Assembly Time versus Run Time 275 |
| | Debugging Statements 276 |
| | Include/Exclude at Assembly Time 277 |
| | .IF and .ENDC 277 |
| | .IF Conditions 278 |
| | Reusing Labels 279 |
| | Assembler Location Counter 280 |
| | .ASECT 282 |
| | Immediate ASCII Constants 282 |
| | Simple Macros 283 |
| | Macro Definitions 283 |
| | Processing Macros 284 |
| | .MCALL Revisited 285 |
| | Macros with Arguments 286 |
| | Macro Arguments 286 |
| | Nested Macro Use 288 |
| | What Can Go Wrong? 289 |
| | Summary 290 |
| | Exercises 291 |
| 13 | Solving Real World Problems 293 |
| | Floating-Point Numbers 297 |
| | PDP-11 Floating-Point Representation 299 |
| | Floating-Point Arithmetic 301 |
| | Comparing Floating-Point Numbers 302 |
| | What Can Go Wrong? 302 |
| | Double-Precision Floating-Point Arithmetic 303 |
| | Converting Fractions 303 |
| | Software and Hardware Support for Floating Point 306 |
| | Floating-Point Traps 308 |
| | Feldstein's Computer Error 309 |

| | |
|-------------------------------|-----|
| Arrays | 309 |
| One-Dimensional Arrays | 312 |
| Two-Dimensional Arrays | 313 |
| N-Dimensional Arrays | 315 |
| Computing Array Displacements | 316 |
| In-Line Code | 317 |
| Dope Vectors | 317 |
| A Hybrid Approach | 318 |
| What Can Go Wrong? | 319 |
| Special Arrays | 320 |
| Summary | 320 |
| Exercises | 321 |

| | | |
|-----------|-----------------------------------|-----|
| 14 | High Speed I/O and Other Topics | 323 |
| | Device Types | 325 |
| | Block-Oriented Devices | 326 |
| | Sequential Access Devices | 326 |
| | Information Density | 329 |
| | Tape Data-Transfer Speeds | 330 |
| | Records and Files | 330 |
| | Industry-Compatible Magnetic Tape | 331 |
| | Binary Data | 331 |
| | Tape Labels | 332 |
| | Interrupts and DMA Transfers | 332 |
| | Other Tape Devices | 334 |
| | Cassette Tapes | 334 |
| | Direct Access Devices; Disks | 335 |
| | Disk Drives | 335 |
| | Disk I/O | 339 |
| | Disk Capacity and Performance | 340 |
| | Other Disks | 342 |
| | Recording Media | 342 |
| | Winchester Drives | 343 |
| | Direct Access Storage Devices | 343 |
| | Spooling | 343 |
| | Performance | 345 |
| | Turnaround Time | 345 |
| | Throughput | 346 |
| | Response Time | 346 |
| | Blocking | 347 |
| | RESET | 348 |
| | WAIT | 349 |
| | Bus Organized Computers | 350 |
| | Stealing Memory Cycles | 351 |
| | Analog Data | 353 |
| | Digital I/O | 354 |
| | Traps | 355 |
| | Processor Traps | 357 |
| | Trap Trace | 357 |
| | Summary | 358 |
| | Exercises | 359 |

| | |
|-----------|--|
| 15 | Selected Topics 361 |
| | Variable Length Macros 363 |
| | Other Assembler Features 371 |
| | Generated Labels 371 |
| | Temporary Radix Control 373 |
| | .IRP 374 |
| | .REPT 376 |
| | Nested Macro Definitions 377 |
| | Tables, Lists, Queues, and Trees 380 |
| | Lists 381 |
| | Two-Dimensional Linked Lists 385 |
| | Recursion 386 |
| | Recursively Defined Macros 387 |
| | Using Trees 388 |
| | Another Example of Recursion 390 |
| | Cross Assembly 391 |
| | B and NB 394 |
| | Concatenation 395 |
| | Threaded Code 396 |
| | Speeding Up Programs 399 |
| | Faster Buses and Cache Memory 401 |
| | Cache Memory 402 |
| | On-Line, Real-Time Computing 404 |
| | Reentrant Code 406 |
| | Reentrancy on Large Systems 406 |
| | Reentrancy on a Dedicated Computer 408 |
| | Coroutines 409 |
| | Error Detection and Correction 411 |
| | Summary 414 |
| | Exercises 415 |
| | Case Study 423 |
| 16 | Micros, Minis, Maxis 429 |
| | Common Features 431 |
| | Distinguishing Features 431 |
| | Micros 431 |
| | Minis 432 |
| | Maxis 433 |
| | Virtual Memory Revisited 434 |
| | What Do Computers Cost? 434 |
| | A Large PDP-11/70 System 435 |
| | Pricing Trends 437 |
| | Summary 439 |
| | Exercises 439 |
| 17 | How Does the Hardware Work 441 |
| | Historical Evolution 443 |
| | Functional Elements of a Computer 445 |
| | Storing a Bit 446 |
| | Building an ALU 448 |
| | A Comparator 449 |
| | Performing Binary Arithmetic 449 |
| | Sequencing 450 |
| | From Relays to Tubes 451 |
| | Tubes to Transistors 452 |
| | Microprogramming 453 |
| | Telecommunications, Teleprocessing 454 |

| | | |
|-----------|---|------------|
| | Telephone Dial-up Access | 454 |
| | Automatic Speed Detection | 456 |
| | Leased Lines | 457 |
| | Communication Interfaces | 458 |
| | Modem Control | 459 |
| | Networks | 459 |
| | Packet Switching | 459 |
| | Local Networks, Ethernet | 461 |
| | Trends | 461 |
| | Remote Diagnosis | 463 |
| | Summary | 465 |
| | Exercises | 466 |
| 18 | Beyond Machine and Assembly Language | 469 |
| | Other Computer Languages | 473 |
| | DBMS | 474 |
| | Summary | 477 |
| | Exercises | 477 |
| 19 | Selected Readings and Annotations to Bibliography | 479 |
| | Vendor Publications | 481 |
| | History and Evolution: the PDP-11 | 483 |
| | History and Evolution: Computing | 483 |
| | Keeping Up | 483 |
| | Other Periodicals | 484 |
| | User Groups | 485 |
| | Annotated Bibliography | 485 |
| | Algorithms | 486 |
| | Codes | 486 |
| | Computer Hardware and Computer Organization | 486 |
| | History | 487 |
| | Magazines and Newspapers | 487 |
| | Networks and Telecommunications | 488 |
| | PDP-11 References | 488 |
| | PDP-11 Textbooks | 489 |
| | Programming Languages | 489 |
| | System Software | 490 |
| | Other Computers | 490 |
| | Thought, Food for | 491 |
| | Appendixes | 493 |
| | Appendix 1: PDP-11 Instruction Set, MACRO-11 Syntax | 495 |
| | Appendix 2: MACRO-11 Directives | 505 |
| | Appendix 3: MACRO-11 Assembly-Time Diagnostic Error Codes | 509 |
| | Appendix 4: ASCII Codes | 513 |
| | Appendix 5: Using MACRO-11 with RT-11 | 517 |
| | Appendix 6: Using MACRO-11 with RSX-11 | 519 |
| | Appendix 7: Using MACRO-11 with RSTS | 520 |
| | Appendix 8: Using MACRO-11 with UNIX | 521 |
| | Index | 563 |