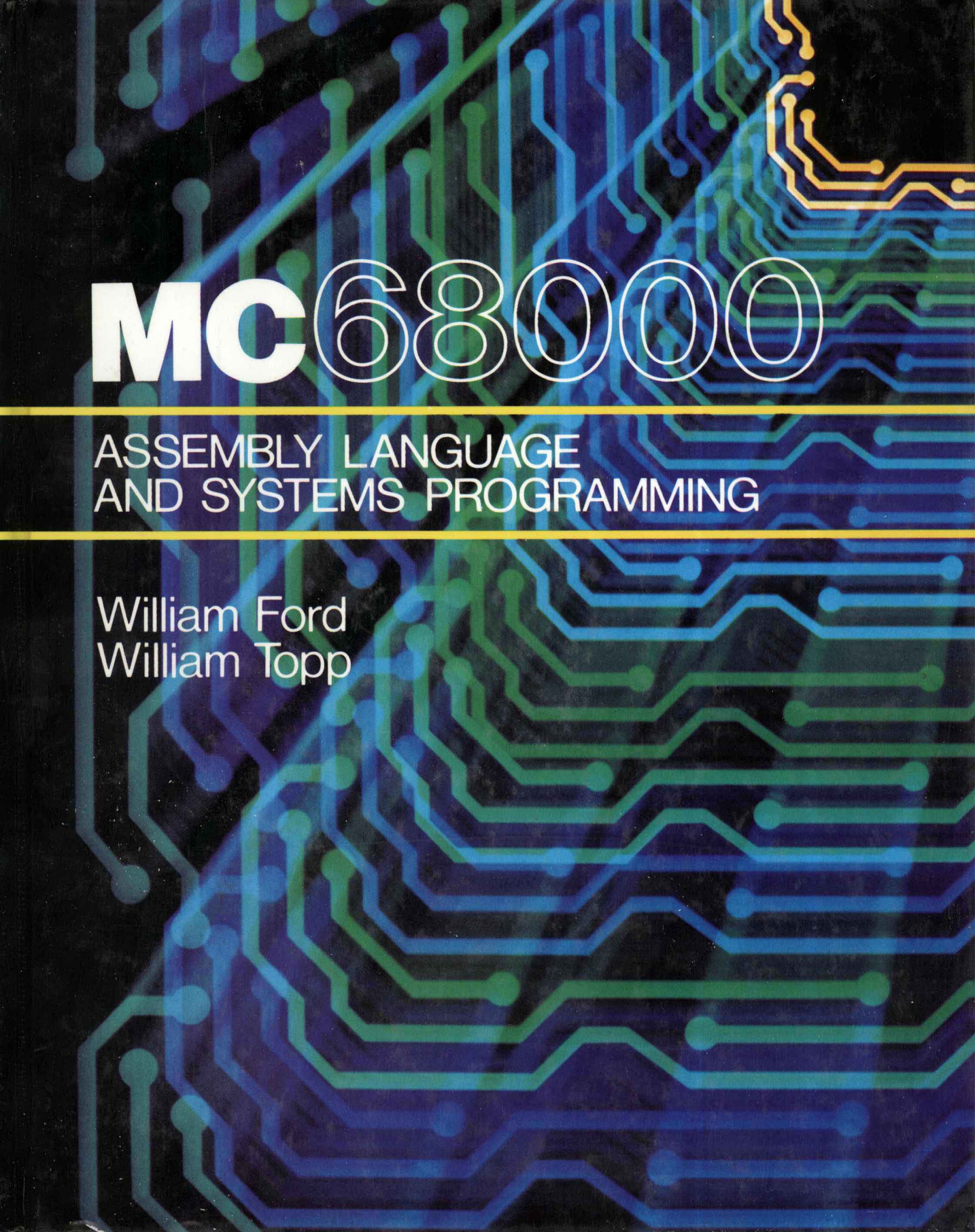
MC68000
ASSEMBLY LANGUAGE
AND SYSTEMS PROGRAMMING
William Ford
William Topp

The MC68000
Assembly Language and Systems Programming

William Ford
University of the Pacific

William Topp
University of the Pacific

D. C. HEATH AND COMPANY
Lexington, Massachusetts Toronto

Cover Photograph: HELIOPTIX by Henry Ries. Design by Miriam Recio.

Published simultaneously in Canada.

Printed in the United States of America.

International Standard Book Number: 0-669-16085-7

Library of Congress Catalog Card Number: 87-81173

10 9 8 7 6 5

Preface

The assembly language and systems programming applications for the Motorola 68000 family of microprocessors constitute the focus of this comprehensive text. The first nine chapters contain material suitable for a one-semester course in assembly language programming. Chapters 10 through 14 cover advanced topics, including high-level language run-time environment, data structures, I/O programming, exception processing, and interrupts, with a strong emphasis placed on major applications of assembly language to systems programming. This advanced material could be used as a major component for courses in programming languages, systems programming, operating systems, microcomputers, and computer organization. In addition, the style and the level of detail in the text permit an individual to use the book for self-study and allow instructors to assign supplemental reading as necessary.

Throughout the text, examples and complete programs are used extensively to illustrate concepts, instructions and addressing modes. Their purpose is to provide the reader with hands-on experience with the MC68000 processor, its assembly language, and applications. The high-level language Pascal is used to clarify the more complex algorithms. Only a limited understanding of Pascal is assumed, and this material can be skipped if desired.

The MC68000 is one of the most widely used 16-bit microprocessors. Its sophisticated instruction set provides a powerful set of addressing modes, and a well-designed assembly language is available for solving even the most complex problems. Since the MC68000 is used in a variety of microprocessor-based applications, developers involved with hardware and software at the systems level are required to have a detailed understanding of its assembly language and architecture. Although the main text describes the MC68000 assembly language, most of the material also applies to the MC68008, the MC68010, and the MC68012 processors. Special instructions for these processors are

detailed in Appendix B. Appendix E includes a comprehensive introduction to the 32-bit MC68020, detailing the architecture, addressing modes, data types, and additional instructions of this newly popular processor.

Since I/O programming is one of the most important applications for assembly language, this text includes three I/O libraries. The main library implements numerical and character stream I/O and can be used with system redirection. A listing of its routines is given in Appendix F. Chapter 12 introduces an I/O library for the stand-alone Motorola Educational Board. This package uses the primitive routines GETCHAR and PUTCHAR that can be modified for other stand-alone systems. Chapter 14 contains a complete interrupt-driven I/O driver with buffering and flow control.

Chapter Descriptions

Chapters 1 through 7 present the basics of MC68000 assembly language and include an introduction to MC68000 computer organization, the operation of an assembler, the main instructions, and all addressing modes.

Chapter 1 introduces the concept of high-level language, assembly language, and machine language code. A brief history of the MC68000 processor is included.

Chapter 2 covers the number systems and arithmetic used in assembly language programming. Readers unfamiliar with these topics will want to read carefully definitions and examples since this material is fundamental to the rest of the book.

Chapter 3 discusses the basic architecture and data organization of the MC68000. The machine code format for instructions, including opcode words and extension words, is covered.

Chapter 4 presents assembly language details along with basic addressing modes and supporting instructions. A discussion of common syntax and runtime errors is included.

Chapter 5 introduces control structures for assembly language programming using the branch instructions. Algorithms to implement high-level language conditionals and loop structures are covered.

Chapter 6 uses the concept of array access to cover the remaining MC68000 addressing modes. PC relative modes are introduced.

Chapter 7 discusses assembly language subroutines. Parameter passing using registers, memory, and the stack is covered.

Beginning with Chapter 8, the book focuses on applications of assembly language. Concepts are implemented by writing external subroutines that are used extensively in applications.

Chapter 8 discusses arithmetic routines. Extended 64-bit operations are included along with BCD numbers. An optional section on floating-point numbers is included.

Chapter 9 includes a variety of topics on string handling. A string handling library is developed. Data conversion for I/O transfer is introduced along with a discussion of data encryption.

Chapter 10 covers selected topics in the high-level language run-time environment. The use of LINK/UNLK for local variables is a major topic. Also included is a discussion of recursion, reentrant code, and position independent code. Sets and matrices are implemented to model high-level language code. This material could be used to supplement a course in programming languages.

Chapter 11 selects classical topics from data structures including stacks, queues, linked lists, and trees. Emphasis is given to the code generated by most compilers to implement data structures.

Chapters 12 through 14 apply to a course in systems programming. Topics covered in these chapters include I/O programming, exception processing, and peripheral device interrupts.

Chapter 12 treats I/O programming on a stand-alone system. The programming details and hardware concepts necessary to deal with an ACIA, timer, and PIA are discussed. A terminal I/O package is fully developed.

Chapter 13 introduces exception processing. The MC68000 exceptions are defined and then applied in a series of test programs that include sample service routines. A complete program to perform single stepping of a program running in user mode is presented.

Chapter 14 completes the discussion of exception processing with a study of interrupt processing. A serial driver with flow control and I/O buffering is implemented. The concept of process switching introduces the topic of concurrent programming and the resulting problems in mutual exclusion.

Chapter Ordering

Chapters 1 through 7 cover the basic topics of assembly language programming. These concepts are integral to an understanding of the book. Except for the introduction of MC68000 instructions, Chapters 8 through 13 may be

covered independently. Chapter 14 requires a complete reading of Chapters 12 and 13. Chapters 3, 8, and 14 also contain optional special topic sections, which have been marked with an asterisk (*). These sections may be skipped if desired.

Support Material

The *M68000 Family Resident Structured Assembler Reference Manual* defines the assembly language syntax used in this book. It may be obtained from Motorola Semiconductor Products Inc.

A complete 68020 assembler written in C and running under 4.2BSD or System V UNIX is available at a nominal cost from the authors. A modification of the codefile generation module is required to run the assembler on another system.

A Macintosh supplement is available which contains page-by-page differences for users of the Apple Macintosh computer and a discussion of the Mac programming environment, including graphics and sound.* It features a library of I/O routines and many example programs (disk available from the supplement's author). In addition, a complete set of the text programs translated for the Mac is available on disk (also from the supplement's author; see order information in the Mac supplement and Instructor's Guide).

An Instructor's Guide containing teaching tips, sample tests, transparency masters, and answers to most exercises is available from the publisher. Also included is order information for a floppy disk or a tape listing of programs and subroutines contained in the text.

Acknowledgments

The authors have been supported by friends, students, and colleagues in the preparation of the book. No one has been more significant than our editor, Karin Ellison, who shared our enthusiasm for the project and worked tirelessly to provide resources and key industry and university contacts.

Greg Winters of Gasboy Development, Kirkland, Washington, has been a valuable technical consultant for the text. He wrote a complete 68020 assembler for our course and made appropriate modifications when requested. As noted previously, this assembler may be purchased from the authors.

Our students lived through multiple versions of the book. Their class testing of material, comments, and blank stares gave us important feedback.

Our reviewers were invaluable in assisting us in the final product. As critics and supporters of our work, they provided detailed comments on both the content and pedagogical approach. Most comments were gladly included in the last revision. We express special thanks to Jack Boudreau of Harvard University. His work motivated us to ask him to write the Macintosh supplement.

---

* Macintosh is a trademark of Apple Computer, Inc.

Others offering very special assistance include Steve Eisenbarth, Baylor University; Jan Harrington, Bentley College; Vincent Manis, University of British Columbia; and Herb Nyser, DeAnza College. The assistance of early reviewers, including James Brand, University of Akron; George Brown, Rochester Institute of Technology; Michael Hennessy, University of Oregon; David Lingle, State University of New York at Stony Brook; John T. O'Donnell, Indiana University; and David Rossiter, Cornell University, helped structure the book.

We appreciate the work of the production staff at D. C. Heath, especially Jill Hobbs.

Motorola Semiconductor has been most cooperative in providing technical documentation. Our special thanks to Quelo, Inc. in Seattle, Washington, for providing a MC68020 cross assembler used to test the code presented in Appendix E.

William Ford
William Topp

Contents

1 Introduction

1.1 Computers and Computer Languages 2
1.1.1 High-Level Languages 2
1.1.2 Machine Language 3
1.1.3 Assembly Language 4
1.2 Why Study Assembly Language? 5
1.3 The Motorola 68000 Microprocessor Family 6
Exercises 7

2 Representation of Data

2.1 Number Systems 9
2.1.1 Binary Numbers 9
2.1.2 Binary-Decimal Conversion 10
2.1.3 Hexadecimal Numbers 11
2.1.4 Binary-Hex Conversion 12
2.2 Binary-Hex Addition and Subtraction 14
2.3 Fixed Length Binary Numbers 15
2.3.1 Odometer Numbers 17
2.3.2 Negative Odometer Numbers 19
2.3.3 Two's Complement Binary Numbers 20
2.3.4 A Summary of Two's Complement Numbers 22
2.3.5 Two's Complement Hex Numbers 23
2.4 Two's Complement Addition and Subtraction 24
2.5 Overflow 25
2.5.1 Signed Overflow 26
2.5.2 Unsigned Overflow 27
2.5.3 Sign Extension of Numbers 28
2.6 Representing Character Data 30
2.6.1 ASCII Codes 30
2.6.2 Control Characters 31
2.7 Logical Operations 32
Exercises 34

3 Machine Organization and Programming

3.1 The Basic Structure of a Computer 38
3.1.1 Main Memory 38
3.1.2 Registers 39
3.1.3 Arithmetic Logic Unit 40
3.1.4 Control Unit 40
3.1.5 Input/Output Units 40
3.1.6 The Bus 43
3.2 Memory Addressing 43
3.3 Data Organization 46
3.3.1 Data Organization in Memory 46
3.4 MC68000 Registers 48
3.4.1 The Data Registers 48
3.4.2 The Address Registers 49
3.5 Basic Machine Instructions 51
3.5.1 Sample Instructions 53
3.5.2 A Machine Code Sequence 56
3.5.3 Decoding Machine Language Instructions 58
3.6 The Instruction Execution Cycle 59
3.6.1 Instruction Prefetch* 62
3.6.2 Instruction Timing 64
Exercises 64

4 Assembly Language Programming

4.1 Program Structure 70
4.1.1 MC68000 Assembler Program: Global View 72
4.1.2 MC68000 Assembler Program: Local View 73
4.1.3 Storage Allocation Directives 75
4.2 Assembling and Running Programs 77
4.2.1 The Run-Time Environment 79
4.3 Introduction to Addressing Modes 82
4.3.1 Absolute and Immediate Addressing Modes 82
4.3.2 Register-Direct Addressing Modes 84
4.3.3 Address Register Instructions 86
4.3.4 Indirect Addressing 88
4.3.5 Hex Input and Output Routines 90
4.3.6 MC68000 Character Input and Output Routines 91
4.3.7 Logical Operations 94
4.4 Errors in the Programming Process 96
4.5 Hand-Translation of Instructions 100
Exercises 102

5 Introduction to Branching

5.1 The Condition Code Register 109
5.1.1 Subtraction and the Carry Bit 111
5.2 Simple Branch Instructions 112
5.2.1 The Test Instruction 114
5.2.2 The Negate Instruction 116
5.2.3 Arithmetic Shift Operations 118
5.3 The Signed Comparison Branches 121
5.3.1 Signed Branches and the CCR 123
5.4 Structured Programming 125
5.4.1 IF..THEN 125
5.4.2 IF..THEN..ELSE 126
5.4.3 WHILE..DO 127
5.4.4 LOOP..EXIT 127
5.4.5 The FOR Loop 128
5.4.6 The Quick Instructions 129
5.5 Unsigned Branches 131
5.5.1 Comparing Addresses 132
5.6 Additional Branch Instructions 134
5.6.1 Long Branches: The Jump Instruction 137
5.7 Branch Instruction Machine Code 138
5.7.1 Sample Code 139
5.7.2 Computing the Offset: An Algorithm 140
Exercises 141

6 Arrays and Stacks

6.1 Sequential Memory Access 146
6.1.1 Sequential Access Addressing Modes 147
6.1.2 Stacks 150
6.2 Indexed Memory Access 154
6.3 The PC Relative Modes 160
6.3.1 Specifying PC Relative Mode 163
6.3.2 The EQU Directive 165
6.3.3 Addressing Categories 167
6.4 Application Programs 168
Exercises 173

7 Subroutines

7.1 Subroutine Call and Return 179
7.1.1 The Stack Pointer 179
7.1.2 Calling a Subroutine 179
7.1.3 Return from Subroutine 182
7.1.4 Structure of a Subroutine 183
7.1.5 The MOVEM Instruction 183
7.1.6 Demonstration Programs 187
7.2 Parameter Passing 190
7.2.1 Using Program Memory 190
7.2.2 Passing Parameters on the Stack 194
7.2.3 PEA—The Call by Reference Instruction 196
7.3 Demonstration Programs 197
7.3.1 Printing Registers—RTR Instruction 199
7.3.2 Parameter Passing with Functions 202
7.3.3 External Subroutines 204
Exercises 206

8 Arithmetic on the 68000

8.1 More MC68000 Arithmetic Instructions 212
8.1.1 The Logical Shift Instructions 212
8.1.2 Hardware Multiplication and Division 214
8.2 Extended Arithmetic: Additive Operations 219
8.2.1 Extended Addition 219
8.2.2 Extended Subtraction 222
8.2.3 Extended Negation 223
8.3 Extended Arithmetic: Multiplicative Operations 227
8.3.1 Extended Shifts 227
8.3.2 Extended Multiplication 232
8.3.3 Extended Division 235
8.4 Decimal Arithmetic 237
8.4.1 BCD Digits 238
8.4.2 Internal BCD Representation 238
8.4.3 MC68000 BCD Instructions 240
8.4.4 BCD Routines 242
8.5 Floating-Point Numbers* 246
8.5.1 Representation of Floating-Point Numbers 247
8.5.2 Floating-Point Internal Format 248
8.5.3 Floating-Point External Format 250
8.5.4 Floating-Point Addition/Subtraction 255
8.5.5 Floating-Point Multiplication 258
8.5.6 Floating-Point Test Program 260
Exercises 263

9 Bit and Byte Operations

9.1 Dealing with ASCII Codes 267
9.1.1 The Terminal Keyboard 268
9.1.2 Bit Manipulation Instructions 269
9.1.3 Parity Checks 270
9.2 String Formats 272
9.3 String Routines 273
9.3.1 I/O Routines 274
9.3.2 String Functions 275
9.3.3 String Operators 278
9.3.4 Demonstration Program: File Names 280
9.4 External Format Conversion 285
9.4.1 Exploring the Issues 285
9.4.2 Format Conversion: Input 286
9.4.3 Format Conversion: Output 290
9.5 Internal Conversion 294
9.5.1 Binary/BCD Conversions 294
9.5.2 Data Encryption 296
Exercises 297

10 High-Level Language Run-Time Environment

10.1 The CASE Statement: Jump Tables 302
10.2 Sets 305
10.2.1 Demonstration Program: Sets 306
10.3 Matrices 307
10.3.1 Matrix Access Functions 308
10.3.2 Matrix Code 310
10.4 Stack Frames 311
10.4.1 The LINK and UNLK Instructions 313
10.4.2 Subroutines with a Variable Number of Arguments 317
10.4.3 Debuggers 319
10.5 Recursion 320
10.5.1 Tower of Hanoi 325
10.6 Reentrant Code 330
10.6.1 Conditions for Reentrant Code 331
10.7 Position-Independent Code 334
10.7.1 Base Relative Addressing 337
Exercises 338

11 Data Structures

11.1 Stacks 344
11.2 Queues 347
11.2.1 Queue Operations 348
11.3 Linked Lists 353
11.3.1 The Node Allocation Function: NEW 354
11.3.2 Inserting Nodes 356
11.3.3 Demonstration Program 358
11.4 Binary Trees 360
11.4.1 Binary Tree Scan Algorithms 361
11.4.2 Demonstration Program 364
11.5 Hashing 366
11.5.1 Hashing Functions 368
11.5.2 Overflow Handling Strategies 369
Exercises 373

12 I/O Programming

12.1 I/O Peripheral Interfaces 376
12.1.1 Memory-Mapped I/O Access 377
12.1.2 Educational Computer Board 378
12.1.3 Running Programs on the ECB System 378
12.2 The MC6850 ACIA 380
12.2.1 Serial Transmission 380
12.2.2 RS232 Control Lines 382
12.2.3 MC6850 Registers 384
12.2.4 Programming the MC6850 ACIA 387
12.2.5 A Terminal I/O Package 387
12.2.6 Echoing Characters 395
12.2.7 ECB Transparent Mode 397
12.3 Timer 399
12.3.1 Timer Registers 400
12.3.2 Initializing the Counter Preload Register 401
12.3.3 Timer Control Register 403
12.3.4 Timer Status Register 404
12.3.5 Programming the Timer 405
12.3.6 Process Switching 413

12.4 Communicating with a Parallel Printer 416
12.4.1 Printer Port Interface 416
12.4.2 PI/T Registers 418
12.4.3 Programming the Printer Port 420
12.4.4 Demonstration Program 422
Exercises 424

13 Exception Processing

13.1 CPU States 428
13.2 The Status Register and System Stack 429
13.2.1 The Status Register 429
13.2.2 Status Register Access 430
13.2.3 The Supervisor and User Stack Pointers 432
13.3 The Exception Processing Cycle 433
13.4 System Initialization 437
13.4.1 System Initialization: Hardware 439
13.4.2 System Initialization: Software 440
13.5 The Trace Exception 442
13.6 Program Code Causing Exceptions 443
13.6.1 The TRAP Exception 443
13.6.2 Unimplemented Instruction 447
13.6.3 Zero Divide and Trap-on-Overflow Exceptions 448
13.6.4 The CHK Exception 450
13.6.5 Demonstration Program 451
13.7 Error Conditions Causing Exceptions 455
13.7.1 Illegal Instruction 456
13.7.2 Privilege Violation Exception 459
13.7.3 Address Error 459
13.7.4 Bus Error 463
Exercises 464

14 Peripheral Device Interrupts

14.1 Priority Interrupts 470
14.1.1 The Interrupt Acknowledge Cycle 472
14.2 MC6850 ACIA Interrupts 474
14.2.1 Initializing the ACIA Registers 474
14.2.2 Demonstration Program: Transparent Mode 475
14.2.3 ACIA Receive/Transmit Interrupts* 477
14.3 MC68230 PI/T Interrupts* 483
14.3.1 PI/T Interrupt Registers 483
14.3.2 Demonstration Program: PRINTSTRING 484

14.4 The Timer Interrupt 487
14.4.1 Sample Timer Program 488
14.5 Multiple Exceptions 490
14.6 An Interrupt-Driven Serial Driver 494
14.7 Concurrent Programming 504
14.7.1 Mutual Exclusion with the MC68000 514
Exercises 517

Appendix A Condition Codes Computation A1
Appendix B Instruction Set Details A5
Appendix C Instruction Format Summary A134
Appendix D MC68000 Instruction Execution Times A151
Appendix E The MC68020 32-Bit Microprocessor A160
Appendix F Utility Programs A232
Appendix G ASCII Code Chart A242
Answers to Selected Exercises A243
Index A261

1 Introduction

To most users a computer is a problem-solving tool that is regarded as a "black box." Packaged software and high-level languages provide simple access to the power of the computer without requiring special understanding of its internal architecture. As with any sophisticated device, some technical understanding of the computer provides control and more effective use of the system. Studying assembly language programming is a good way to gain this understanding.

This text introduces the assembly language of a popular 16-bit microprocessor, the Motorola MC68000. Along with the language, the text examines the basic hardware of the processor, its instruction set, and its access to peripheral devices. In the process, the reader will gain a better understanding of high-level-language constructs and the run-time environment. This study is detailed and requires patience, but the rewards will be a better understanding and appreciation of a computer and its applications.

A computer is a high-speed device that performs arithmetic operations and symbol manipulation through a set of machine-dependent instructions. The heart of a computer is the central processing unit (CPU) that contains the circuitry to define and execute a set of machine instructions (see Figure 1.1).

This text describes a computer system equipped with a Motorola MC68000 processor, a terminal, and printer for input/output (I/O) access (see Figure 1.2). The primary focus of the text is the development of MC68000 assembly language, but the principles of machine architecture, the execution of code written in common programming languages (such as Pascal), and the writing of programs that deal directly with the hardware in a system are also covered.