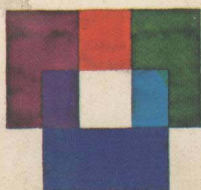


Using the IBM PC: Organization and Assembly Language Programming

Mark Franklin

Assembly
Language



CBS Computer Books

Using the IBM Personal Computer:

Organization and Assembly Language Programming

Mark A. Franklin

Departments of Electrical Engineering and Computer Science
Washington University
St. Louis, Missouri

HOLT, RINEHART AND WINSTON

New York	Chicago	San Francisco	Philadelphia
Montreal	Toronto	London	Sydney
Mexico City	Rio de Janeiro	Madrid	

The cover illustration is the additive color model from Miles Color Art, Tallahassee, Florida prepared using the digital facsimiles process (patent pending), Center for Color Graphics, Florida State University, Tallahassee, Florida.

IBM ® is a registered trademark of International Business Machines Corporation.

Copyright © 1984 CBS College Publishing

All rights reserved.

Address correspondence to:

383 Madison Avenue, New York, NY 10017

First distributed to the trade in 1984 by Holt, Rinehart and Winston general book division.

Library of Congress Cataloging in Publication Data

Franklin, Mark A., 1940—

Using the IBM Personal Computer

Bibliography: p.

Includes index.

1. IBM Personal Computer. 2. IBM Personal Computer—Programming. 3. Assembler language (Computer program language) I. Title. II. Title: The I.B.M. Personal Computer.

QA76.8.I2594F73 1985 001.64 84-10865

ISBN 0-03-062862-8 (pbk.)

ISBN 0-03-062862-8

Printed in the United States of America

Published simultaneously in Canada

4 5 6 039 9 8 7 6 5 4 3 2 1

CBS COLLEGE PUBLISHING

Holt, Rinehart and Winston

The Dryden Press

Saunders College Publishing

**Using the IBM
Personal
Computer:
Organization and
Assembly
Language
Programming**

Preface

This book is intended for those who seek to go beneath the exterior view presented by higher-level computer languages, and understand the operation of the IBM Personal Computer (PC) at a deeper level. Accordingly, the book focuses on the architecture and operation of the IBM PC and its machine and macro assembly languages. In addition, we explore the operating system and BIOS (Basic Input/Output System) facilities available at the assembly language level, and we present techniques for interfacing assembly language programs with those written in higher-level languages. Included is all the essential material necessary to program effectively in assembly language.

The Intel 8088 Microprocessor provides the core computing engine for the IBM PC; consequently, the book contains a great deal of material on the operation of this powerful 16-bit microprocessor. A thorough understanding of this material provides a sound basis for understanding the operation of many contemporary computers.

There are numerous practical situations where knowledge of machine organization and assembly language programming is necessary or even essential. These situations generally require that more control be exerted over the internal operations of the com-

puter than is typically afforded by higher-level languages. Such control, for example, may be needed to achieve maximum speed. A higher-level language may be adequate from a logical point of view, but the programs generated may be too slow for the application in question. Close control over the computer at the assembly language level can often exploit the computer's resources in a more efficient manner. Toward this end, special attention is devoted (Chapter 11) to the problem of interfacing assembly language routines with higher-level-language routines. This allows use of the simpler and more powerful programming constructs available in higher-level languages for most programming activity while using assembly language programs for time-critical sections.

Assembly language programming is also needed when close control over standard input/output devices is required, or when special input/output devices are to be interfaced to the computer. Chapter 10 covers the operation and control of several standard devices using BIOS.

Moreover, special applications sometimes require that modifications or additions be made to the computer's overall control program or operating system. Such modifications typically must be done at the assembly language level. Techniques for implementing such adjustments are discussed in Chapter 9, which contains a review of the disk operating system's function calls and their use.

Although the book centers on the IBM PC, the bulk of the material also applies to the IBM PCjr and IBM PC/XT computers. All of these systems use the Intel 8088 Microprocessor and therefore employ the same machine and assembly languages. By the same token, the book also can be used to learn assembly language programming on the host of compatible computers now available. This includes systems based on the Intel 8086 Microprocessor, which is faster than the Intel 8088 but functionally identical to it.

This text supports a comprehensive introductory course on computer organization and assembly language programming for people with some background in higher-level-language programming and general notions relating to algorithms and programs. There is no hardware or logic design background required, and any material needed in this area is contained in the text. The book can be used in a formal course setting or as a guide for self-study. Numerous examples are provided in the text, and each chapter concludes with a summary of the main points and a set of problems to help reinforce the material. To take full advantage of the material, an IBM Personal Computer (or compatible equivalent) should be available. The best way to learn assembly language, or for that matter any programming language, is to work problems and write, debug, and run programs.

The book is divided into two major parts: Part I, Chapters 1 through 7, contains basic material on computer organization, operation, instruction set, and assembly language which is needed to write assembly language programs. Part II, Chapters 8 through 11, contains advanced material on macros, interrupts, and DOS (Disk Operating System) and BIOS facilities, with the concluding chapter dealing with higher-level-language-assembly language interfacing techniques. Skill in assembly language programming can be achieved by concentrating initially on Part I, after which more specialized material in Part II can be acquired as needed. Once the language and its associated facilities have been mastered, the book will continue to serve as a useful reference.

I am grateful to my colleagues at Washington University and elsewhere for their support in this endeavor. Particular thanks goes to Sy Pollack for his cheerful encouragement and careful reading of the book, to Tom Patterson for his help in developing and debugging example programs, to Howard Bomze for his close reading of Chapters 9 and 10, and to my brother David for his steadfast optimism that it would get done. My thanks also to CBS Educational Publishing and to the CBS book reviewers who made many valuable suggestions. Finally, I want to thank my wife, children, and friends who had to put up with my replies of the form, "I'm sorry, I have to go work on the book," but nevertheless remained faithful.

The book is dedicated to my wife, Barbara, and to my parents, Jack and Celia Franklin, without whose understanding and help this would not have been possible.

Contents

Preface *xiii*

PART I THE BASICS

1 INTRODUCTION AND OVERVIEW 3

- 1.1 Computer Organization 5**
 - 1.1.1 Memory Unit 6
 - 1.1.2 Control Unit 7
 - 1.1.3 Arithmetic/Logic Unit 9
 - 1.1.4 Input and Output Units 11

1.2 The Intel 8088 Microprocessor 11

1.3	Software Organization	13
1.3.1	Software Design and Development	13
1.3.2	Software Environment and Tools	15
1.3.3	Algorithm Development and Documentation	17
1.3.4	Assembler Preliminaries	18
1.3.5	The Overall Implementation Process	23
1.4	Summary	25
	Exercises	25
2	DATA AND NUMBER REPRESENTATION	29
2.1	Bits, Bytes, Words, and Data Pseudo-Operations	30
2.2	Number Representation	32
2.2.1	Number Systems	32
2.2.2	Number System Conversion	34
2.2.3	Signed Numbers	38
2.2.4	Range, Precision, and Floating-point Numbers	44
2.2.5	Binary-Coded Decimal (BCD) Numbers	51
2.3	Alphanumeric Character Representation	53
2.4	Summary	56
	Exercises	57
3	COMPUTER ORGANIZATION	60
3.1	Memory Organization	60
3.1.1	The Memory Space	60
3.1.2	The Segment Registers	62
3.1.3	Instruction Fetching and the Instruction Pointer	64
3.1.4	Segmented Memory	65
3.2	Addressing Modes and Effective Address Calculations	67
3.2.1	The General Registers	67
3.2.2	Addressing Modes	69
3.3	Instruction Formats and Encoding	79

3.4 The Flag Register	84
3.4.1 Status and Control Flags	84
3.5 Summary	87
Exercises	88

4 PROGRAM CONTROL AND DECISION MAKING 91

4.1 Stacks	91
4.2 Procedures	94
4.2.1 Procedure Calls and Returns	95
4.2.2 Procedure Parameter Transfer	100
4.3 Instructions to Transfer Program Control	104
4.3.1 Unconditional Transfers	104
4.3.2 Conditional Transfers	107
4.3.3 Iteration Control	108
4.3.4 Interrupts (An Introduction)	110
4.4 Recursive Procedures	110
4.5 Summary	112
Exercises	113

5 ARITHMETIC INSTRUCTIONS 116

5.1 Integer Addition and Subtraction	116
5.2 Integer Multiplication and Division	120
5.3 Sign Extension	121
5.4 ASCII and Decimal Addition and Subtraction	122
5.5 ASCII Multiplication and Division	124
5.6 Summary	126
Exercises	127

6 OTHER INTERNAL OPERATIONS 129

- 6.1 Instructions for Data Transfer 129**
 - 6.1.1 General-purpose Transfers 131
 - 6.1.2 Address Object Transfers 133
 - 6.1.3 Flag Transfers 134
 - 6.1.4 Input/Output Transfers 134
- 6.2 Bit Manipulation Instructions 134**
 - 6.2.1 Bit-Oriented Logic 134
 - 6.2.2 Shift and Rotate 137
- 6.3 String Instructions 139**
 - 6.3.1 String Moves and Repeats 139
 - 6.3.2 String Compare and Scan 143
 - 6.3.3 Store, Load, and Complex String Operations 145
- 6.4 Process Control Instructions 146**
 - 6.4.1 Flag Instructions 146
 - 6.4.2 Synchronization and **NOP** Instructions 146
- 6.5 Summary 148**
- Exercises 148**

7 ASSEMBLY LANGUAGE 151

- 7.1 An Overview 151**
- 7.2 Constants, Variables, Labels, and Their Attributes 157**
 - 7.2.1 Constants 157
 - 7.2.2 Variables 159
 - 7.2.3 Labels 161
- 7.3 Assembler Expressions and Operators 162**
- 7.4 Pseudo-Operation Statements 168**
 - 7.4.1 Module Definition and Symbol Exchange 168
 - 7.4.2 Segment Definition 170
 - 7.4.3 Procedure Definition 172
 - 7.4.4 Data Definition 172
 - 7.4.5 Symbol Definition 172
 - 7.4.6 Location Counter Specification 174
 - 7.4.7 Other Pseudo-Ops 175

7.5 Summary 176

Exercises 177

PART II ADVANCED TOPICS

8 ADVANCED ASSEMBLY LANGUAGE FEATURES 181

8.1 An Introduction to Assembler Macros 181

8.2 Macro Pseudo-Ops 185

8.2.1 Macro Definition and Expansion 185

8.2.2 Repeated Macros 187

8.2.3 The **LOCAL**, **PURGE**, and **EXITM** Macro
Pseudo-Ops 189

8.3 Macro Libraries 190

8.4 An Introduction to Conditional Assembly 190

8.5 Conditional Pseudo-Ops 193

8.6 Recursive Macro Calls 197

8.7 Summary 199

Exercises 200

9 INTERRUPTS, TRAPS, and DOS 202

9.1 Interrupt Instructions and Types 203

9.1.1 Software Interrupts 203

9.1.2 Hardware Interrupts 209

9.2 8088 and 8259 Hardware Interrupts 211

9.2.1 8088 Interrupts (Types 0 to 7) 211

9.2.2 8259 Interrupts (Types 8 to FH) 213

9.2.3 Miscellaneous Software Interrupts
(Types 18H to 1CH) 216

9.3 DOS Interrupts (Types 20H to 3FH) 219

9.3.1 Displaying Messages with DOS 223

9.3.2 Reading the Keyboard with DOS 228

9.3.3 Printing with DOS 231

9.4 Summary 231

Exercises 233

10 INPUT/OUTPUT PROGRAMMING WITH BIOS 235

10.1 The Keyboard (A Type 16H Interrupt) 236

10.2 The Printer (A Type 17H Interrupt) 240

10.3 The Display (A Type 10H Interrupt) 242

10.3.1 The Monochrome Display 243

10.3.2 The Color/Graphics Display 250

**10.4 The Asynchronous Communication Adapter
(A Type 14H Interrupt) 256**

10.5 The Diskette Adapter (A Type 13H Interrupt) 260

10.6 Generating Sounds 264

**10.7 Determining Equipment Configuration
(Types 11H and 12H Interrupts) 266**

10.8 Summary 269

Exercises 270

11 HIGHER-LEVEL-LANGUAGE INTERFACING 274

11.1 Two Example Applications 275

11.1.1 A Stock Market Simulation Problem 275

11.1.2 Assembly Language Market Simulation Program 277

11.1.3 A Graphics Problem 280

11.1.4 Assembly Language Screen Mode and Line-Drawing
Programs 284

11.2 Interfacing with BASIC 289

11.2.1 BASIC Instructions for Loading and Calling Assembly
Language Procedures 290

11.2.2 Debugging Assembly Language Programs Interfaced to
BASIC 292

11.2.3	Constructing Memory Image Files for Use with BLOAD	294
11.2.4	The Stock Market Simulation Problem with BASIC	294
11.2.5	The Graphics Problem with BASIC	297
11.3	Interfacing with FORTRAN	299
11.3.1	Calling Assembly Language Procedures and Passing Parameters in FORTRAN	299
11.3.2	Debugging Assembly Language Programs Interfaced to FORTRAN	301
11.3.3	The Stock Market Simulation Problem with FORTRAN	302
11.3.4	The Graphics Problem with FORTRAN	303
11.4	Interfacing with PASCAL	304
11.4.1	PASCAL Interfacing Methods	304
11.4.2	The Stock Market Simulation Problem with PASCAL	308
11.4.3	The Graphics Problem with PASCAL	308
11.5	Summary	308
	Exercises	310

APPENDIX A: IBM PC MEMORY MAP 313

APPENDIX B: 8086/88 INSTRUCTION ENCODING 314

APPENDIX C: INSTRUCTION SET SUMMARY 322

REFERENCES 337

INDEX 341

Part
I
The
Basics



1

Introduction and Overview

Development of the Large Scale Integration and Very Large Scale Integration (LSI and VLSI) of digital circuits has dramatically lowered the price and increased the performance of computers. This effect, coupled with a large and favorable market response, has resulted in what is called “the personal computer.” Computers in this class are *personal* in at least two senses. First, they are inexpensive enough so that many “persons” can now afford to purchase one for themselves. Second, the expectation is that they will often be used in ways determined primarily by the individual user. Their low cost is giving users increased freedom to explore new application areas ranging from use as very smart terminals to dedicated use as processors for a host of other applications.

Often the computer is purchased and used as a packaged, or “turnkey,” system where both the hardware and software have been tailored to an application of interest. Dedicated word processing systems fall into this category. In many situations users will program applications for themselves in higher-level languages such as BASIC, FORTRAN, or PASCAL. Such languages significantly ease the task of programming and should be used whenever possible. Use of these languages allows one to concentrate