

# **IBM® PC ASSEMBLY LANGUAGE AND PROGRAMMING**

Fourth Edition

## **IBM® PC 汇编语言与程序设计**

第 4 版

**Peter Abel**

Professor Emeritus

British Columbia

Institute of Technology

清华大学出版社

Prentice-Hall International, Inc.

# (京)新登字 158 号

IBM PC assembly language and programming 4th ed./Peter Abel

© 1998 by Prentice Hall, Inc.

Original edition published by Prentice Hall, Inc.

All Rights Reserved.

For sale in Mainland China only.

Prentice Hall 公司授权清华大学出版社在中国境内(不包括中国香港、澳门特别行政区和台湾地区)独家出版发行本书影印本。

本书任何部分未经出版者书面同意,不得用任何方式抄袭、节录或翻印。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

北京市版权局著作权合同登记号: 01-98-0391

## 图书在版编目(CIP)数据

IBM PC 汇编语言与程序设计:第4版:英文/埃布尔(Abel, P.)著. - 影印版.

- 北京:清华大学出版社,1998.2

(大学计算机教育丛书)

ISBN 7-302-02830-3

I . I… II . 埃… III . 程序语言-程序设计-英文 IV . TP311.11

中国版本图书馆 CIP 数据核字(98)第 07060 号

出版者:清华大学出版社(北京清华大学学研大厦,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印刷者:清华大学印刷厂

发行者:新华书店总店北京发行所

开本:850×1168 1/32 印张:19.625

版次:1998年5月第1版 2001年9月第4次印刷

书号:ISBN 7-302-02830-3/TP·1487

印数:13001~15000

定价:30.00 元

## 出版者的话

今天,我们的大学生、研究生和教学、科研工作者,面临的是一个国际化的信息时代。他们将需要随时查阅大量的外文资料;会有更多的机会参加国际性学术交流活动;接待外国学者;走上国际会议的讲坛。作为科技工作者,他们不仅应有与国外同行进行口头和书面交流的能力,更为重要的是,他们必须具备极强的查阅外文资料获取信息的能力。有鉴于此,在国家教委所颁布的“大学英语教学大纲”中有一条规定:专业阅读应作为必修课程开设。同时,在大纲中还规定了这门课程的学时和教学要求。有些高校除开设“专业阅读”课之外,还在某些专业课拟进行英语授课。但教学双方都苦于没有一定数量的合适的英文原版教材作为教学参考书。为满足这方面的需要,我们陆续精选了一批国外计算机科学方面最新版本的著名教材,进行影印出版。我社获得国外著名出版公司和原著作者的授权将国际先进水平的教材引入我国高等学校,为师生们提供了教学用书,相信会对高校教材改革产生积极的影响。

我们欢迎高校师生将使用影印版教材的效果、意见反馈给我们,更欢迎国内专家、教授积极向我社推荐国外优秀计算机教育教材,以利我们将《大学计算机教育丛书(影印版)》做得更好,更适合高校师生的需要。

清华大学出版社

《大学计算机教育丛书(影印版)》项目组

1999.6

# Contents

<b>PREFACE</b>	<b>xiii</b>
<b>Part A Fundamentals of PC Hardware and Software</b>	<b>1</b>
<b>1 BASIC FEATURES OF PC HARDWARE</b>	<b>1</b>
Introduction	1
Bits and Bytes	1
Binary Numbers	3
Hexadecimal Representation	6
ASCII Code	7
The Processor	7
Internal Memory	9
Segments and Addressing	11
Registers	13
Key Points	17
Questions	18
<b>2 REQUIREMENTS FOR USING PC SOFTWARE</b>	<b>20</b>
Introduction	20
Features of the Operating System	20
The Boot Process	21
Input-Output Interface	22
The System Program Loader	22

The Stack	23
Addressing of Instructions and Data	25
Instruction Operands	27
Key Points	27
Questions	28

### **3 EXECUTING COMPUTER INSTRUCTIONS** **29**

Introduction	29
Using the DEBUG Program	30
Viewing Memory Locations	31
Machine Language Example I: Immediate Data	34
Machine Language Example II: Defined Data	38
An Assembly Language Example	41
Using the INT Instruction	42
Saving a Program from Within DEBUG	45
Using the PTR Operator	45
Key Points	46
Questions	47

## **Part B Fundamentals of Assembly Language** **49**

### **4 REQUIREMENTS FOR CODING IN ASSEMBLY LANGUAGE** **49**

Introduction	49
Assemblers and Compilers	50
Program Comments	50
Reserved Words	51
Identifiers	51
Statements	52
Directives	53
Instructions for Initializing a Program	57
Instructions for Ending Program Execution	59
Example of a Source Program	59
Initializing for Protected Mode	60
Simplified Segment Directives	61
Data Definition	62
Directives for Defining Data	65
The EQU Directive	69
Key Points	70
Questions	71

### **5 ASSEMBLING, LINKING, AND EXECUTING A PROGRAM** **73**

Introduction	73
Preparing a Program for Execution	73

Assembling a Source Program	74
Using Conventional Segment Definitions	76
Using Simplified Segment Directives	81
Two-Pass Assembler	83
Linking an Object Program	83
Executing a Program	86
Cross-Reference Listing	86
Error Diagnostics	87
The Assembler Location Counter	89
Key Points	89
Questions	89

## **6 SYMBOLIC INSTRUCTIONS AND ADDRESSING**

91

Introduction	91
The Symbolic Instruction Set	91
Instruction Operands	94
The MOV Instruction	97
Move-and-Fill Instructions	98
Immediate Operands	99
The XCHG Instruction	100
The LEA Instruction	101
The INC and DEC Instructions	101
Extended Move Operations	101
The INT Instruction	103
Aligning Data Addresses	104
Near and Far Addresses	104
The Segment Override Prefix	105
Key Points	105
Questions	106

## **7 WRITING .COM PROGRAMS**

108

Introduction	108
Differences Between an .EXE and a .COM Program	108
Converting into .COM Format	109
Example of a .COM Program	110
The .COM Stack	111
Debugging Tips	112
Key Points	112
Questions	113

## **8 PROGRAMMING REQUIREMENTS FOR LOGIC AND CONTROL**

114

Introduction	114
Short, Near, and Far Addresses	115

Instruction Labels	115
The <code>JMP</code> Instruction	116
The <code>LOOP</code> Instruction	118
The Flags Register	119
The <code>CMP</code> Instruction	120
Conditional Jump Instructions	121
Calling Procedures	123
Effect of Program Execution on the Stack	126
Boolean Operations	127
Program: Changing Uppercase to Lowercase	129
Shifting Bits	130
Rotating Bits	132
Jump Tables	134
Organizing a Program	136
Key Points	137
Questions	137
<b>Part C Screen and Keyboard Operations</b>	<b>139</b>
<b>9 INTRODUCTION TO SCREEN AND KEYBOARD PROCESSING</b>	<b>139</b>
Introduction	139
The Screen	140
Setting the Cursor	140
Clearing the Screen	141
<code>INT 21H</code> Function <code>09H</code> for Screen Display	142
<code>INT 21H</code> Function <code>0AH</code> for Keyboard Input	144
Program: Accepting and Displaying Names	145
Using Control Characters in a Screen Display	149
<code>INT 21H</code> Function <code>02H</code> for Screen Display	150
File Handles	150
<code>INT 21H</code> Function <code>40H</code> for Screen Display	151
<code>INT 21H</code> Function <code>3FH</code> for Keyboard Input	152
Key Points	154
Questions	154
<b>10 ADVANCED FEATURES OF SCREEN PROCESSING</b>	<b>156</b>
Introduction	156
Video Adapters	157
Setting the Video Mode	157
Using Text Mode	158
Screen Pages	161
Using <code>INT 10H</code> for Text Mode	161

Program: Displaying the ASCII Character Set	167
ASCII Characters for Boxes and Menus	170
Program: Blinking, Reverse Video, and Scrolling	171
Direct Video Display	173
Using Graphics Mode	175
INT 10H for Graphics	177
Program: Setting and Displaying Graphics Mode	180
Determining the Type of Video Adapter	182
Key Points	182
Questions	183

## **11 ADVANCED FEATURES OF KEYBOARD PROCESSING** **185**

Introduction	185
The Keyboard	186
Keyboard Shift Status	186
The Keyboard Buffer	187
Using INT 21H for Keyboard Input	188
Using INT 16H for Keyboard Input	189
Extended Function Keys and Scan Codes	191
Program: Selecting from a Menu	193
BIOS INT 09H and the Keyboard Buffer	197
Keying in the Full ASCII Character Set	201
Key Points	201
Questions	202

## **Part D Data Manipulation** **203**

### **12 PROCESSING STRING DATA** **203**

Introduction	203
Features of String Operations	204
REP: Repeat String Prefix	204
MOVS: Move String Instruction	205
LODS: Load String Instruction	207
STOS: Store String Instruction	208
Program: Using LODS and STOS to Transfer Data	209
CMPS: Compare String Instruction	211
SCAS: Scan String Instruction	213
Example: Using Scan and Replace	214
Alternative Coding for String Instructions	214
Duplicating a Pattern	215
Program: Right Adjusting a Screen Display	215
Key Points	218
Questions	218

<b>13</b>	<b>ARITHMETIC: I—PROCESSING BINARY DATA</b>	<b>220</b>
	Introduction	220
	Processing Unsigned and Signed Data	221
	Addition and Subtraction	222
	Extending Values in a Register	224
	Performing Arithmetic on Doubleword Values	225
	Multiplication	227
	Performing Doubleword Multiplication	231
	Special Multiplication Instructions	234
	Multiplication by Shifting	235
	Division	235
	Division by Shifting	239
	Reversing the Sign	240
	The Numeric Data Processor	240
	Key Points	242
	Questions	243
<b>14</b>	<b>ARITHMETIC: II—PROCESSING ASCII AND BCD DATA</b>	<b>244</b>
	Introduction	244
	Data in Decimal Format	245
	Processing ASCII Data	246
	Processing Unpacked BCD Data	249
	Processing Packed BCD Data	251
	Converting ASCII Data to Binary Format	253
	Converting Binary Data to ASCII Format	255
	Shifting and Rounding a Product	257
	Program: Converting ASCII Data	257
	Key Points	263
	Questions	264
<b>15</b>	<b>DEFINING AND PROCESSING TABLES</b>	<b>265</b>
	Introduction	265
	Defining Tables	265
	Direct Addressing of Table Entries	267
	Searching a Table	271
	The XLAT (Translate) Instruction	277
	Program: Displaying Hex and ASCII Characters	278
	Sorting Table Entries	280
	Linked Lists	281
	The TYPE, LENGTH, and SIZE Operators	286
	Key Points	286
	Questions	287

**Part E Advanced Input/Output**

289

**16 DISK STORAGE I: ORGANIZATION**

289

- Introduction 289
- Disk Characteristics 289
- The Disk System Area and Data Area 292
- The Boot Record 294
- The Directory 294
- The File Allocation Table 296
- Exercise: Examining the FAT 299
- Processing Files on Disk 302
- Key Points 302
- Questions 302

**17 DISK STORAGE II: WRITING AND READING FILES**

304

- Introduction 304
- ASCII Strings 305
- File Handles 305
- Error Return Codes 305
- File Pointers 306
- Using File Handles to Create Disk Files 306
- Using File Handles to Read Disk Files 311
- Using File Handles for Random Processing 313
- Program: Processing an ASCII File 320
- Absolute Disk I/O 324
- Disk Services Using File Control Blocks 325
- Key Points 330
- Questions 330

**18 DISK STORAGE III: INT 21H FUNCTIONS FOR SUPPORTING DISKS AND FILES**

332

- Introduction 332
- Operations Handling Disk Drives 333
- Program: Reading Data from Sectors 342
- Operations Handling the Directory and the FAT 345
- Program: Displaying the Directory 346
- Operations Handling Disk Files 348
- Program: Selectively Deleting Files 353
- Key Points 356
- Questions 356

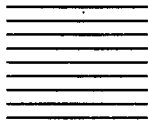
<b>19</b>	<b><i>DISK STORAGE IV: INT 13H DISK FUNCTIONS</i></b>	<b>357</b>
	Introduction	357
	BIOS Status Byte	358
	Basic INT 13H Disk Operations	358
	Program: Using INT 13H to Read Sectors	360
	Other INT 13H Disk Operations	363
	Key Points	368
	Questions	368
<b>20</b>	<b><i>FACILITIES FOR PRINTING</i></b>	<b>369</b>
	Introduction	369
	Common Printer Control Characters	370
	INT 21H Function 40H: Print Characters	370
	Program: Printing With Page Overflow and Headings	371
	Program: Printing ASCII Files and Handling Tabs	374
	INT 21H Function 05H: Print Character	378
	Special Printer Control Characters	379
	INT 17H Functions for Printing	380
	Key Points	381
	Questions	381
<b>21</b>	<b><i>OTHER INPUT/OUTPUT FACILITIES</i></b>	<b>383</b>
	Introduction	383
	Mouse Features	383
	Mouse Functions	384
	Common INT 33H Operations	385
	Program: Using the Mouse	391
	Ports	394
	String Input/Output	395
	Generating Sound	397
	Key Points	397
	Questions	398
<b>Part F</b>	<b>Advanced Programming</b>	<b>400</b>
<b>22</b>	<b><i>DEFINING AND USING MACROS</i></b>	<b>400</b>
	Introduction	400
	Two Simple Macro Definitions	401
	Using Parameters in Macros	402
	Macro Comments	403

Using a Macro Within a Macro Definition	405
The LOCAL Directive	407
Including Macros from a Library	407
Concatenation	410
Repetition Directives	410
Conditional Directives	412
Key Points	416
Questions	417
<b>23 LINKING TO SUBPROGRAMS</b>	<b>419</b>
Introduction	419
The SEGMENT Directive	420
Intrasegment Calls	421
Intersegment Calls	422
The EXTRN and PUBLIC Attributes	423
Using EXTRN and PUBLIC for an Entry Point	425
Defining the Code Segment as PUBLIC	427
Using Simplified Segment Directives	429
Defining Common Data as PUBLIC	430
Defining Data in Both Programs	431
Passing Parameters to a Subprogram	433
Linking Pascal with an Assembly Language Program	436
Linking C with an Assembly Language Program	440
Key Points	442
Questions	444
<b>24 MEMORY MANAGEMENT</b>	<b>445</b>
Introduction	445
The Main DOS Programs	445
The High-Memory Area	446
The Program Segment Prefix	447
Memory Blocks	452
Memory Allocation Strategy	455
The Program Loader	456
Allocating and Freeing Memory	462
Loading or Executing a Program Function	463
Program Overlays	464
Resident Programs	470
Key Points	475
Questions	476

<b>Part G</b>	<b>Reference Chapters</b>	<b>478</b>
<b>25</b>	<b>BIOS DATA AREAS AND PROGRAM INTERRUPTS</b>	<b>478</b>
	Introduction	478
	The Boot Process	478
	The BIOS Data Area	479
	Interrupt Services	482
	BIOS Interrupts	483
	BIOS:DOS Interface	487
	DOS Interrupts	487
	INT 21H Services	488
	Key Points	494
	Questions	495
<b>26</b>	<b>OPERATORS AND DIRECTIVES</b>	<b>497</b>
	Introduction	497
	Type Specifiers	497
	Operators	498
	Directives	504
<b>27</b>	<b>THE PC INSTRUCTION SET</b>	<b>525</b>
	Introduction	525
	Register Notation	526
	The Addressing Mode Byte	526
	Two-Byte Instructions	527
	Three-Byte Instructions	528
	Four-Byte Instructions	528
	Instruction Set	529
	<b>APPENDIXES</b>	
	A Conversion Between Hexadecimal and Decimal Numbers	557
	B ASCII Character Codes	560
	C Reserved Words	562
	D Assembler and Link Options	564
	E The DEBUG Program	572
	F Keyboard Scan Codes and ASCII Codes	579
	<b>ANSWERS TO SELECTED QUESTIONS</b>	<b>583</b>
	<b>INDEX</b>	<b>595</b>



# PREFACE



The heart of a personal computer is a microprocessor, which handles the computer's requirements for arithmetic, logic, and control. The microprocessor had its origin in the 1960s, when research designers devised the integrated circuit (IC) by combining various electronic components into a single component on a silicon "chip." The manufacturers set this tiny chip into a device resembling a centipede and connected it into a functioning system. In the early 1970s, Intel introduced the 8008 chip, which ushered in the first generation of microprocessors.

By 1974, the 8008 had evolved into the 8080, a popular second-generation microprocessor with general-purpose use. In 1978, Intel produced the third-generation 8086 processor, which provided some compatibility with the 8080 and represented a significant advance on its design. Next, Intel developed the 8088, a variation of the 8086 that provided a slightly simpler design and compatibility with then-current input/output devices. The 8088 was selected by IBM in 1981 for its forthcoming personal computer. An enhanced version of the 8088 is the 80188, and enhanced versions of the 8086 are the 80186, 80286, 80386, 80486, Pentium (or 586), PentiumPro (or 6x86) each of which provides additional operations and processing power.

Each family of processors has its own unique set of instructions that are used to direct its operations, such as accept input from a keyboard, display data on a screen, and perform arithmetic. This set of instructions is known as the system's machine language, which (as you will soon discover) is too complex and obscure for developing programs. Software

suppliers provide an assembly language for the processor family that represents the various instructions in a more understandable symbolic code.

High-level languages such as C and BASIC were designed to eliminate the technicalities of a particular computer, whereas a low-level assembly language is designed for a specific family of processors.

The use of assembly language provides a number of advantages:

- A program written in assembly language requires considerably less memory and execution time than a program written in a high-level language.
- Assembly language gives a programmer the ability to perform highly technical tasks that would be difficult, if not impossible, in a high-level language.
- A knowledge of assembly language provides an understanding of machine architecture that no high-level language can ever provide.
- Although most software specialists develop new applications in high-level languages, which are easier to write and maintain, a common practice is to recode in assembly language those routines that have caused processing bottlenecks.
- Resident programs (that reside in memory while other programs execute) and interrupt service routines (that handle input and output) are almost always developed in assembly language.

The following material is required for learning PC assembly language:

- Access to an IBM personal computer (any model) or equivalent compatible.
- A copy of the DOS operating system (preferably a recent version) and familiarity with its use.
- A copy of an assembler translator program (preferably a recent version). Common suppliers include Microsoft, Borland, and SLR systems.

The following are *not* required for learning assembly language:

- Prior knowledge of a programming language, although such knowledge may help you grasp some programming concepts more readily.
- Prior knowledge of electronics or circuitry. This book provides all the information about the PC's architecture that you require for programming in assembly language.

## OPERATING SYSTEMS

The major purposes of an operating system are (1) to allow users to instruct a computer regarding actions it is to take (such as executing a particular program) and (2) to provide means of storing (cataloging) information on disk and of accessing it.

The basic operating system for the PC and its compatibles is MS-DOS from Microsoft, known as PC-DOS on the IBM PC. Each version of DOS has provided additional features that have extended the capability of the PC. It is much easier to learn the intricacies of assembly language while within a relatively simple operating system like DOS

rather than attempt it from within the OS/2 or Windows environment. Within DOS, you can freely experiment and can later step up to more advanced systems.

## FOCUS OF THIS BOOK

The primary aim of this book is to assist readers in learning assembly programming. To this end, the book first covers the simpler aspects of the hardware and the language and then introduces instructions as they are needed. As well, the text emphasizes clarity in program examples. Thus the examples use those instructions and approaches that are the easiest to understand, even though a professional programmer would often solve similar problems with more sophisticated—but less clear—code.

The programs also omit macro instructions (explained in Chapter 22); although professional programmers use macros extensively, their appearance in a book of this nature would interfere with learning the principles of the language. Once you have learned these principles, you can then adopt the clever techniques of the professional.

## THE APPROACH TO TAKE

This book can act as both a tutorial and a permanent reference. To make the most effective use of your investment in a PC and software, work through each chapter carefully, and reread any material that is not immediately clear. Key the program examples into your computer, convert them into executable “modules,” and get them to execute (or “run”). Also, be sure to work through the exercises at the end of each chapter.

The first nine chapters furnish the foundation material for the book and for assembly language. After studying these chapters, you can proceed with Chapters 12, 13, 15, 16, 20, 21, or 22. Chapters 25, 26, and 27 are intended as references. Chapters related to each other are:

- 9 through 11 (on screen and keyboard operations)
- 13 and 14 (on arithmetic operations)
- 16 through 19 (on disk processing)
- 23 and 24 (on subprograms and memory management)

On completing this book, you will be able to:

- Understand the hardware of the personal computer.
- Understand machine-language code and hexadecimal format.
- Understand the steps involved in assembling, linking, and executing programs.
- Write programs in assembly language to handle the keyboard and screen, perform arithmetic, convert between ASCII and binary formats, perform table searches and sorts, and handle disk input and output.
- Trace machine execution as an aid in program debugging.
- Write your own macro instructions to facilitate faster coding.
- Link separately assembled programs into one executable program.

Learning assembly language and getting your programs to work is an exciting and challenging experience. For the time and effort invested, the rewards are sure to be great.

## NOTES ON THE FOURTH EDITION

This fourth edition reflects a considerable number of enhancements to the previous edition, including the following:

- More features of the Intel 80486 and later processors
- More program examples and exercises
- Earlier introduction to interrupt operations
- Inclusion of material on more recent assembler versions
- Considerable reorganization and revision of explanations throughout the text
- Revised and additional questions at the end of each chapter.

Users of the third edition should note that the contents of Chapters 25 and 26 have been combined in this edition. Also, the conditional jump instructions described in Chapter 27 ("The PC Instruction Set") are now organized and summarized in a table.

## ACKNOWLEDGMENTS

The author is grateful for the assistance and cooperation of all those who contributed suggestions for, reviews of, and corrections to earlier editions.