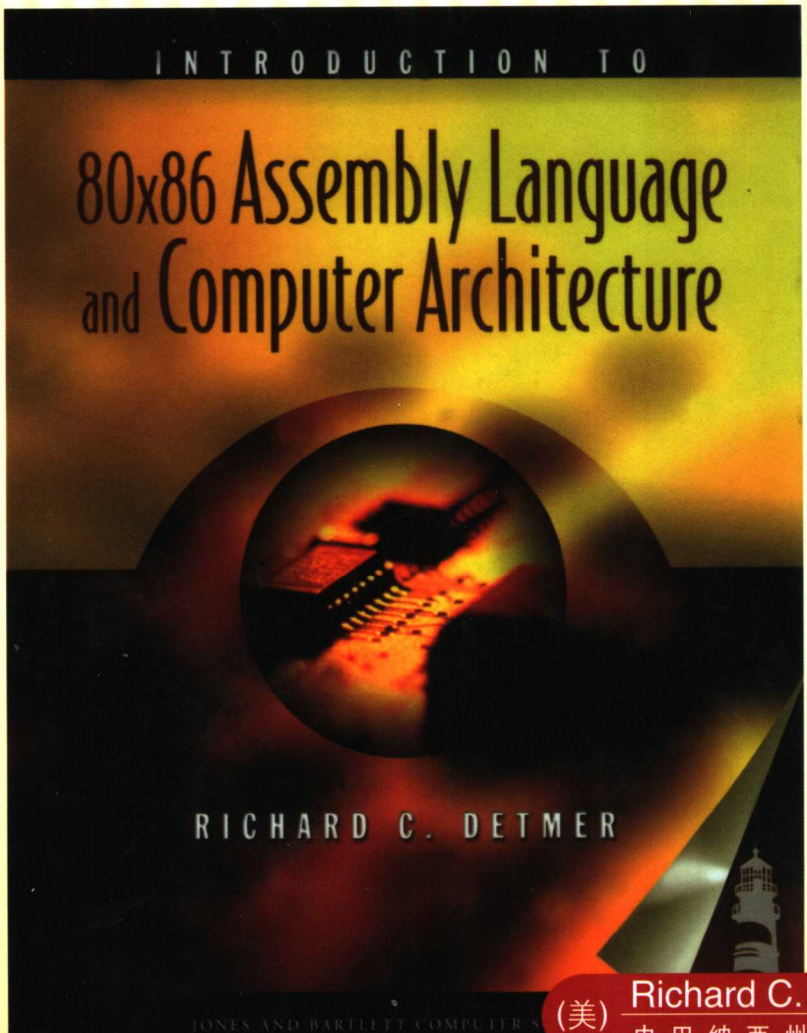
HZ BOOKS

经 典 原 版 书 库

80x86汇编语言与计算机体系结构

（英文版）



附赠 CD-ROM

(美) Richard C. Detmer 著
中田纳西州立大学

机械工业出版社
China Machine Press

经 典 原 版 书 库

80x86汇编语言与计算机体系结构

(英文版)

Introduction to 80x86 Assembly Language and Computer Architecture

(美) Richard C. Detmer
中田纳西州立大学 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2004-5736

图书在版编目（CIP）数据

80x86汇编语言与计算机体系结构（英文版）/（美）德特默（Detmer, R. C.）著. -北京：机械工业出版社，2004.11

（经典原版书库）

书名原文：Introduction to 80x86 Assembly Language and Computer Architecture

ISBN 7-111-15312-X

Ⅰ. 8…　Ⅱ. 德…　Ⅲ. ①汇编语言－英文②计算机体系结构－英文　Ⅳ. ①TP313 ②TP303

中国版本图书馆CIP数据核字（2004）第098877号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

北京中兴印刷有限公司印刷 · 新华书店北京发行所发行

2004年11月第1版第1次印刷

787mm × 1092mm 1/16 · 32.5印张

印数：0 001-3 000册

定价：55.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国

家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzedu@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周立柱	周克定	周傲英	孟小峰	岳丽华
范　明	郑国梁	施伯乐	钟玉琢	唐世渭
袁崇义	高传善	梅　宏	程　旭	程时端
谢希仁	裘宗燕	戴　葵		

秘　书　组

武卫东　　温莉芳　　刘　江　　杨海玲

Dedicated to

my mother, Emma Langenhop Detmer Baldwin Toombs
and my uncle, Carl E. Langenhop
both of whom encouraged me to become a scholar.

PREFACE

A computer can be viewed from many different levels. Many people are interested only in using applications such as word processing or games. A computer programmer, however, often sees the computer as an instrument to create new applications software. A high-level language programmer's image of the computer is provided by the language compiler, which gives the impression that the computer stores object types like `integer`, `real`, and `array of char` in named memory locations, calculates values of expressions, calls procedures, executes `while` loops, and so forth.

However, an actual computer works at even lower levels. This book emphasizes the architectural level, that is, the level defined by the machine instructions that the processor can execute. Assembly-language instructions translate directly into machine-language instructions, so that when you write an assembly-language program, you gain an understanding of how the computer works at the machine-language level.

Although this book emphasizes the assembly-language/machine-language level of computer operations, it also looks at other levels. For instance, it describes how high-level language concepts such as `if` statements are realized at the machine level. It discusses some of the functions of the operating system. It briefly describes the logic gates that are used at the hardware level. It also looks at how assembly language is translated into machine language.

To program effectively at any level, programmers must understand certain fundamental principles at the machine level. These apply to most computer architectures. *Introduction to 80x86 Assembly Language and Computer Architecture* teaches these fundamental concepts:

- **memory addressing, CPU registers and their uses**
- **representation of data in a computer in numeric formats and as character strings**
- **instructions to operate on 2's complement integers**
- **instructions to operate on individual bits**
- **instructions to handle strings of characters**

- instructions for branching and looping
- coding of procedures: transfer of control, parameter passing, local variables, and preserving the environment for the calling program

The primary architecture covered is the Intel 80x86 CPU family used in many personal computers. However, almost every chapter includes information about other architectures, or about different computer levels. Programming in assembly language and studying related concepts in *Introduction to 80x86 Assembly Language and Computer Architecture* prepares the student to program effectively in any programming language, to pursue advanced studies in computer design and architecture, or to learn more about system details for specific computers.

Text Organization and Content

Much of the material in this book is based on my previous book, *Fundamentals of Assembly Language Programming Using the IBM PC and Compatibles*. While teaching this material through the years, I have increasingly come to the conclusion that an assembly language course is the best place to introduce computer architecture to most students. This book reflects a stronger emphasis on architecture than on programming. It also concentrates on general concepts as opposed to the details of a particular computer system.

The minimal prerequisite for my assembly language class is a good understanding of a structured high-level language. Chapters 3 through 6 and Chapter 8 form the core of my one-semester course. I normally cover Chapters 1–8 thoroughly, Chapter 9 quickly, and then choose topics from Chapters 10–12 depending on time and resources available. For instance, I sometimes introduce floating-point operations via in-line assembly statements in a C++ program.

Style and Pedagogy

The text primarily teaches by example. A complete assembly-language program is presented very early, in Chapter 3, and its components are carefully examined at a level that the student is able to understand. Subsequent chapters include many examples of assembly language code along with appropriate explanations of new or difficult concepts.

The text uses numerous figures and examples. Many series of "before" and "after" examples are given for instructions. Examples are included that illustrate the use of a debugger. These examples give the student a stronger sense of what is happening inside the computer.

Exercises appear at the end of each section. Short-answer exercises reinforce understanding of the material just covered, and programming exercises offer an opportunity to apply the material to assembly-language programs.

Software Environment

The "standard" 80x86 assembler is Microsoft's Macro Assembler (MASM), version 6.11. Although this assembler can produce code for 32-bit flat memory model programming appropriate to a Windows 95, Windows NT, or other 32-bit Microsoft operating system environment, the linker and debugger that come with this software package are not suitable for use in such an environment. This book comes with a CD containing the assembler program from MASM (ML), a more recent Microsoft linker, the 32-bit full-screen debugger WinDbg (also from Microsoft), and necessary supporting files. This software package provides a good environment for producing and debugging console applications.

The CD included with the book also contains a package designed to simplify input/output for the student, so that the emphasis remains on architecture rather than operating system details. This I/O package is used extensively through most of the book. Finally, the CD contains source code for each program that appears as a figure in the book.

Instructor's Support

Supplementary materials for this book include an Instructor's Guide that contains some teaching tips and solutions to many exercises. In addition, the author can be contacted at rdetmer@mtsu.edu with questions or comments.

Acknowledgments

I would like to thank my students who suffered through preliminary versions of this text, often getting materials that were duplicated "just in time." These students were very good at catching errors. I also want to thank Hong Shi Yuan, who used a preliminary version of this text in his assembly language class and who offered valuable feedback.

Many thanks to the following people who took the time to review the manuscript: Dennis Bouvier, University of Houston–Clear Lake; Barry Fagin, US Air Force Academy; Glynis Hamel, Worcester Polytechnic Institute; Dennis Fairclough, Utah Valley State College; Thomas Higginbotham, Southeastern Louisiana University; Clifford Nadler, Worcester Polytechnic Institute.

My wife, Carol, deserves credit for her understanding during the many hours that I ignored her and word-processed at my computer.

Richard C. Detmer

CONTENTS

CHAPTER 1

Representing Data in a Computer

When programming in a high-level language like Java or C++, you use variables of different types (such as integer, float, or character). Once you have declared variables, you don't have to worry about how the data are represented in the computer. When you deal with a computer at the machine level, however, you must be more concerned with how data are stored. Often you have the job of converting data from one representation to another. This chapter covers some common ways that data are represented in a microcomputer. Chapter 2 gives an overview of microcomputer hardware and software. Chapter 3 illustrates how to write an assembly language program that directly controls execution of the computer's native instructions.

1.1 Binary and Hexadecimal Numbers

A computer uses **bits** (binary digits, each an electronic state representing zero or one) to denote values. We represent such **binary** numbers using the digits 0 and 1 and a base 2 place-value system. This binary number system is like the decimal system except that the positions (right to left) are 1's, 2's, 4's, 8's, 16's (and higher powers of 2) instead of 1's, 10's, 100's, 1000's, 10000's (powers of 10). For example, the binary number 1101 can be interpreted as the decimal number 13,

1		1		0		1		
one 8	+	one 4	+	no 2	+	one 1	=	13

Binary numbers are so long that they are awkward to read and write. For instance, it takes the eight bits 11111010 to represent the decimal number 250, or the fifteen bits 111010100110000 to represent the decimal number 30000. The **hexadecimal** (base 16) number system represents numbers using about one-fourth as many digits as the binary system. Conversions between hexadecimal and binary are so easy that **hex** can be thought of as shorthand for binary. The hexadecimal system requires sixteen digits. The digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 are used just as in the decimal system; A, B, C, D, E, and F are used for the decimal numbers 10, 11, 12, 13, 14, and 15, respectively. Either uppercase or lowercase letters can be used for the new digits.

The positions in hexadecimal numbers correspond to powers of 16. From right to left, they are 1's, 16's, 256's, etc. The value of the hex number 9D7A is 40314 in decimal since

```
   9   ×   4096      36864    [ 4096 = 16³ ]
+ 13   ×    256       3328    [ D is 13, 256 = 16² ]
+  7   ×     16        112
+ 10   ×      1         10    [ A is 10 ]
                   = 40314
```

Figure 1.1 shows small numbers expressed in decimal, hexadecimal, and binary systems. It is worthwhile to memorize this table or to be able to construct it very quickly.

You have seen above how to convert binary or hexadecimal numbers to decimal. How can you convert numbers from decimal to hex? From decimal to binary? From binary to hex? From hex to binary? We'll show how to do these conversions manually, but often the easiest way is to use a calculator that allows numbers to be entered in deci-

Decimal	Hexadecimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Figure 1.1 Decimal, hexadecimal, and binary numbers

mal, hexadecimal, or binary. Conversion between bases is normally a matter of pressing a key or two. These calculators can do arithmetic directly in binary or hex as well as decimal and often have a full range of other functions available. One warning: Many of these calculators use seven segment displays and display the lowercase letter b so that it looks almost like the numeral 6. Other characters may also be difficult to read.

A calculator isn't needed to convert a hexadecimal number to its equivalent binary form. In fact, many binary numbers are too long to be displayed on a typical calculator. Instead, simply substitute four bits for each hex digit. The bits are those found in the third column of Fig. 1.1, padded with leading zeros as needed. For example,

$$3B8E2_{16} = 11\ 1011\ 1000\ 1110\ 0010_2$$

The subscripts 16 and 2 are used to indicate the base of the system in which a number is written; they are usually omitted when there is little chance of confusion. The extra spaces in the binary number are just to make it more readable. Note that the rightmost