汇编语言

李目海◎主编

山东大学出版社

汇编语言

主　编　李目海
副主编　尹晓燕　李　明
　　刘三荣

山东大学出版社

图书在版编目(CIP)数据

汇编语言/李目海主编. —济南:山东大学出版社,2008.1
ISBN 978-7-5607-3550-4

Ⅰ. 汇...
Ⅱ. 李...
Ⅲ. 汇编语言—程序设计
Ⅳ. TP313

中国版本图书馆 CIP 数据核字(2008)第 014944 号

山东大学出版社出版发行
(山东省济南市山大南路 27 号　邮政编码:250100)
山 东 省 新 华 书 店 经 销
山东省恒兴实业总公司印刷厂印刷
787×1092 毫米　1/16　16.25 印张　373 千字
2008 年 2 月第 1 版　2008 年 2 月第 1 次印刷
定价:29.00 元

前　言

汇编语言是一种几乎与机器指令一一对应的计算机程序设计语言，人们大多认为汇编语言编程复杂、应用范围小，而忽视它的重要性。其实汇编语言对每一个希望学习计算机科学与技术的人来说都非常重要，是一门不能不学习的语言。原因很简单，汇编语言是很多专业课程（如数据结构、操作系统、微机原理、单片机和嵌入式程序设计等）的重要基础。不仅如此，对于从事计算机研究的人们来讲，由于其工作平台和研究对象都是计算机，其目的是让计算机代替人类完成某些任务。要想让计算机完成人们设定的任务，就必须将其转化成能被计算机识别的指令（机器指令），而汇编语言正是完成这一功能最直接、最有效的工具。通过学习和使用汇编语言，使读者不仅能感知、理解计算机的逻辑功能，充分获取基于计算机底层的编程经验，而且还能深刻理解计算机程序的运行机制。因此，学习汇编语言，向上可以理解软件，向下能够感知硬件，是读者理解整个计算机系统的最佳起点和最有效的途径。

本书以 Intel 8086/8088 CPU 系列微机为基础机型，较详细介绍了汇编语言程序设计的基础知识和基本方法，所以基于这一机型来讲解汇编语言，是因为目前任何一台与 Intel CPU 兼容的微机均可以使用 Intel 8086/8088 CPU 的工作方式。因此，通过对 Intel 8086/8088 CPU 的寻址方式、指令系统和编程方法的学习，为掌握更高层次的汇编语言（如基于 Intel 80×86 和 Pentium 系列微机的汇编语言）打下坚实的基础。

本书是编者自 1995 年以来从事汇编语言教学与研究工作的结晶。为使读者尽快学会并掌握汇编语言，全书力求通俗易懂、语言简洁、难点分散、循序渐进。为方便读者学习和掌握每章内容的知识点，本书在每章开篇部分列出该章的重点和难点。为避免出现大量单调乏味的汇编指令堆砌，我们将部分汇编指令讲解融入到程序设计的实例中。为方便读者了解掌握知识的程度，本书为每章内容设计了大量习题，并按知识点分解到每节之后，避免了传统教材将习题集中到每章之后带来的种种不便等问题。为方便读者实验，我们将大量实验项目融入到课程实例和习题中，同时在第四章还专门设计一节课，用于详细讲解如何做汇编语言试验，以便为学好本课程创造条件。

为了使读者更好地学习和掌握汇编语言，结合编者学习的经验，下面就如何学习这门语言提出一些建议：

1. 准确掌握重要概念

汇编语言中有许多重要的概念，如汇编、编译、变量、常量、宏、过程和结构化指令等，与其他高级语言相比，这些概念在汇编语言中解释的更清晰、更准确。因此，准确理解这些概念，不仅有助学习汇编语言，而且还有利于高级程序设计语言的学习。

2. 掌握编程思想

学习汇编语言，不仅要掌握它的语法结构及其使用方法，而且更要领会其编程思想，学会通过汇编语言来分析和解决实际问题。

3. 勤练习，多实验

编程思想只有通过编程实践才能熟练掌握。由于汇编指令功能简单，编程相对繁琐，因此，要想学好这门课程应具有足够的耐心和毅力。同时要遵循“模仿”、“变换”和“创新”三步曲。开始学习时，进行大量模仿练习是非常必要的，因此，读者应首先将书中的所有实例和习题上机调试；同时，还应举一反三，变换思维方式，这期间，读者应大量阅读优秀软件的源代码（可通过反汇编获取），学习其中的优秀设计思想，并从不同角度重新设计原来的实验；一旦读者掌握编程技巧，设计并完成创新性实验就会成一件轻而易举的事。

4. 养成良好的编程习惯

编程风格的好坏直接影响到程序的质量。良好的编程风格可以使程序结构更加清晰，程序代码更易于维护、调试和共享。所以，养成良好的编程习惯（如代码缩进和注释等）是非常重要的。

5. 熟练使用 Debug 或 Turbo Debug 软件

这两个软件特别方便汇编程序的开发与调试，熟练掌握它们将会使读者学习汇编编程如虎添翼。

6. 充分利用网上资源

网上有很多关于编程思想、方法、经验和技巧的文章，有大量相关开发工具和优秀的源程序代码、辅导材料等资源。读者若能充分利用这些资源，定会使你的学习事半功倍。

本书的结构：前两章主要讲解汇编语言的基础知识和计算机的基本结构，这是学习第3章指令系统和寻址方式的前提。第3章内容虽然不太多，但对能否学好汇编语言相当关键，只有熟练掌握汇编语言的寻址方式，才能写出正确精练的程序代码。第4章重点讲解做汇编语言实验的方法及汇编语言编程的基础知识。第5～10章按照从易到难，从简单到复杂逐步讲解编写汇编语言程序的方法。读者在学习这几章内容时，应重点学习编程思想，掌握分析问题和编程实现方法。第11～13章是扩展内容，讲解混合编程技术以及在 Win32 下进行汇编程序设计的方法，本部分内容读者可根据需要酌情取舍。

本书由李目海主编，尹晓燕、李明、刘三荣为副主编。其中，第1～6章由宫锋和隆坤编写，第7～10章由李旭宏和刘三荣编写，第11～13章由尹晓燕编写。最后由我统稿。华东师范大学信息科技学院李明教授审阅全书。尽管我们在编写过程中力求完美，但由于编者水平所限，时间仓促，书中难免存在疏漏和不足之处，敬请同行和专家批评指正。本书工作得到国家自然科学基金（项目编号：60573125）和上海重点学科建设项目（项目编号：B411）的部分支持。

李目海

2008 年 1 月于华东师大

目 录

第1章　基础知识

导　读

本章将介绍汇编语言的预备知识，主要包括汇编语言基本常识、进位计数制、数据表示方法等。

主要知识点

◆ 进位计数制及数制转换
◆ 计算机中有符号数的表示
◆ 计算机中字符的表示

1.1　为什么学习汇编语言

与早期编程人员相比，现在的编程人员是相当幸福了，因为现在有多种非常实用方便的编程语言可供选择，如 Delphi，C++，VC++和 Eclipse 等，这些开发环境甚至只需轻击鼠标就可以完成较复杂的应用程序，而汇编语言则不同，所有的操作，包括内存分配与组织都需要编程者自己来完成。既然汇编语言编程如此麻烦，那么为什么我们还要学习它呢？要回答这个问题，需先从计算机程序设计语言的发展史说起。

1.1.1　计算机程序设计语言的发展历史

计算机是硬件和软件的统一体，计算机的工作过程就是一系列指令（或程序）与硬件紧密配合的执行过程。指令是计算机为完成某些操作而发出的指示或命令，一台计算机所有的指令集合称为该计算机的指令系统。程序员为让计算机完成某些特定功能，按照完成工作程序的步骤和要求，在指令系统中选用有关的指令进行编排，这一过程称为编程。而使计算机完成某种任务的一组有序的指令集合称为程序。

计算机程序设计语言经历了一个复杂的发展过程，从最初的机器语言开始至今，已经历了机器语言、汇编语言、高级语言三个阶段。

1. 机器语言

机器语言是计算机诞生和发展初期使用的语言，其指令使用二进制编码形式（即由0，1序列构成的代码），执行指令时，计算机将这些由0，1构成的代码转换成一系列高低电平，并以此驱动计算机的电子器件，从而完成指定的任务。因此，机器语言是被计算机

直接识别并执行的语言。

若计算机微处理器的硬件设计和结构不同，则必须用不同的电平脉冲来驱动和控制，才能使其正常工作，也就是说，不同的微处理器有不同的机器指令集，因此，机器语言是与计算机的硬件密切相关的计算机语言。

下面是一段向计算机屏幕输出一个“$”符号的机器语言程序代码：

```
1011001000100100
1011010000000010
1100110100100001
```

从以上小程序可以看出，书写和阅读机器语言程序绝不是一件容易的事，同时，也暴露出机器语言晦涩难懂、难以维护与查错的弱点。

虽然用机器语言编写程序对程序员有很高的要求，并存在诸多不便，但用它确实可以编写出其他语言难以实现的快速、高效的应用程序。

2. 汇编语言

为克服机器语言可读性差、难以记忆等缺点，人们想出用助记符来表示机器指令的方法，这就形成了用特定符号表示的、与机器指令几乎一一对应并遵循一定语法规则的计算机语言——汇编语言。

例如，前面例子中要求输出“$”的程序用汇编语言书写则为：

```
MOV   DL,'$'
MOV   AH,2
INT   21H
```

从上述例子可以看出，汇编语言书写的代码与我们的平时使用的自然语言非常接近，从而有利于程序代码的理解与记忆(至于每一条汇编语言指令的含义，稍后读者将很快学习到，在此不作解释)。

相对机器语言而言，汇编语言在可读性、方便性等方面前进了一大步，但由于它通过助记符来表示机器指令，从而导致计算机的 CPU 无法直接识别汇编指令，因此，要执行用汇编语言编写的程序，还必须把汇编语言源程序翻译成机器语言，把这种将汇编源程序翻译成机器语言的程序称为汇编程序，翻译过程则称为汇编。如图 1-1 所示。

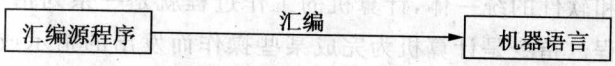

图 1-1　汇编语言源程序转化为机器语言

由于汇编语言的大部分指令都与机器语言指令一一对应，因此，其语句功能相对简单，并对计算机硬件有较强依赖性，可移植性差。为解决这些问题，并提高开发效率，人们研究并开发出通用性强且更易于使用的计算机语言，即高级语言。

3. 高级语言

高级语言起始于 20 世纪 50 年代中期，它与人们日常使用的自然语言和数学语言更接近，可读性更强，也便于用户的理解和运用。由于高级语言对计算机硬件依赖性减小，可移植性好，因此，它们逐渐占领了程序设计的大部分领域。目前，常用的计算机高级语

言有 Delphi,C ++,VC. net,Java 等。

1.1.2 汇编语言的特点

学习汇编语言,必须了解它的特点,一般的,汇编语言主要有以下特点。

1. 对机器依赖程度高

汇编语言是机器指令的符号表示,由于不同类型的 CPU 有不同的机器指令系统,因此汇编语言程序与机器有着非常密切的关系,一般情况下,CPU 不同的计算机间的汇编语言程序是不可移植的。

2. 代码高效

正因为汇编语言对机器的依赖,程序员在编写汇编语言程序时,可充分发挥自己的聪明才智,对机器内部的各种资源进行合理的安排,并充分发挥计算机硬件的功能,让其始终处于最佳状态,从而使程序用最短的代码,达到最高的效率,这是高级语言难以做到的,这也正是汇编语言的魅力所在。

3. 编程复杂

由于汇编语言对机器依赖程度高,其指令与机器指令基本上一一对应,所以,汇编指令也同机器指令一样具有功能单一等问题。要完成某项工作或操作,程序员就必须安排 CPU 的每步工作细节,同时还要对计算机内部的可利用资源进行合理的分配与管理,这就使得编写汇编语言程序比较繁琐、复杂。例如,一个简单数据的输入/输出程序,用汇编语言则需要若干条指令才能完成。

4. 调试复杂

通常情况下,调试汇编语言程序要比调试高级语言程序困难,其主要原因有如下几点:

(1)要求编程者必须了解计算机的内部资源。在调试过程中,要求调试者必须时刻清楚每种资源的变化,以判断 CPU 的执行结果是否符合程序编写的初衷,从而查找出程序编写上的逻辑错误。

(2)汇编中常用的编程技巧极可能破坏程序的可读性。在调试过程中,除了要知道每条指令的执行功能,还要清楚编程技巧在整个解题过程中的作用。

(3)大量使用转移语句。高级语言程序几乎都是禁止或隐式使用“转移语句”,而用汇编语言编写程序时却要大量并显式使用各类转移指令,这些跳转指令大大增加了调试程序的难度。

(4)调试工具相对落后。高级语言程序可以在源程序级进行符号跟踪调试,而汇编语言程序一般只能跟踪机器指令。不过,目前在程序调试方面已有所改善,如 CV(CodeView),TD(Turbo Debug)等软件在源程序级已经能较好完成程序的跟踪和调试。

1.1.3 学习汇编语言的主要目的

通过对汇编语言特点的学习,读者一定会有一种想法,汇编语言编程难、调试难,而高级语言已解决了这些问题,我们学习汇编语言还有什么意义呢! 其实,汇编语言虽然有许多缺点,但它却具有其他语言无法替代的优点,主要优点如下:

1. 有利于加深理解处理器的工作原理

学习和使用汇编语言可以从根本上认识和理解计算机的工作原理。因为计算机执行每个任务，归根到底是执行一段机器语言程序，而汇编语言几乎与机器语言一一对应，因此，通过学习汇编语言，可以更清楚地了解计算机是如何完成各种复杂工作的。

2. 有利于加深理解计算机的系统结构

学习和使用汇编语言需要熟悉和了解计算机的基本系统结构，这对今后深入学习计算机的系统结构有很大帮助，事实上，汇编语言是计算机组成原理和系统结构的先导课。

3. 开发快速高效的程序代码

由于汇编语言指令与机器语言指令几乎一一对应，因此使用汇编语言可以编写短小精悍的程序代码，所以汇编语言特别适合编写要求执行效率高、速度快的代码，如操作系统内核、实时系统和工业控制等领域中的程序代码。

4. 开发与硬件资源相关的软件

现在虽然有许多高级语言，但某些功能仍然需要用汇编语言程序来实现。例如，机器自检、设备驱动程序及受容量限制的软件开发等。

1.1.4 IBM PC 汇编语言程序初步

为便于读者对汇编语言产生感性认识，下面给出一个较完整的汇编语言源程序，其功能是在屏幕上显示 “This is an ASM program!”字符串。

注：读者在此不必深究相关指令的功能及使用方法，所有疑问均会在此后的学习中加以解决。

```
DATA SEGMENT                          ; 定义数据段
   MSG DB 'This is a ASM program!  $' ; 定义变量字符串 MSG
DATA ENDS
STACK SEGMENT STACK                   ; 定义堆栈段
   DB  80H  DUP(0)
STACK ENDS
CODE SEGMENT                          ; 定义代码段
   ASSUME CS:CODE,DS:DATA, SS:STACK
START:                                ; 可执行代码开始标号
   MOV AX,DATA
   MOV DS,AX
   MOV DX,OFFSET MSG                  ; 取变量 MSG 的偏移地址到 DX
   MOV AH,9
   INT 21H                            ; 调用 DOS 的 9 号功能，显示 MSG 存放的内容
   MOV AH,4CH
   INT 21H                            ; 返回 DOS 操作系统
CODE ENDS
   END START
```

上述代码结构主要由三大部分组成，分别为定义数据段、定义堆栈段和定义代码段，它们的定义起始标志都是以 SEGMENT 开始，以 ENDS 结束。在今后编程中我们会经常用到这种定义形式。要使这些汇编源程序代码真正变成计算机可直接运行的程序，还要经过汇编、连接和调试等步骤，具体方法将在第 4 章中学习。

习　题

1. 汇编语言适用于哪些领域，不适合哪些领域？

2. 简述汇编语言的特点。

3. 根据本节实例，编写一个汇编语言程序，要求在屏幕上输出“I like the ASM world!”字符串。

1.2　数据的机内表示及转换

熟悉数据在计算机内的表示方法是掌握汇编语言程序设计的基础之一。在计算机内部，其数据的操作与运算都是通过二进制数(Binary)形式进行的。但编写汇编语言程序时，为了便于书写和查错，常使用十进制(Decimal)或十六进制(Hexadecimal)形式书写数据。因此，学习和熟悉常见的进位计数制及其相互转换是非常必要的。

1.2.1　进位计数制

1. 二进制(Binary)

二进制数由 0，1 两个数字组成，采用“逢二进一”的进位原则。为便于与其他进制数相区别，书写时，在二进制数的后面应加一个“B”字符。例如：1011B，110110110B 都是二进制数。

数据的二进制表示形式简单、明了，但书写较长，通常情况下，编程时一般使用十进制或十六进制。

2. 八进制(Octonal)

八进制数由 0，1，…，7 八个数字构成，运算时采用“逢八进一”的进位原则。书写八进制数时，其数据后面紧跟一个字母“Q”。例如：2637Q，710544Q，125Q 都是八进制数。

3. 十进制(Decimal)

十进制是我们最熟悉的数据表示形式，它由 0，1，…，9 十个数字构成，运算遵循“逢十进一”的进位原则。书写时，在数据后面可紧跟一个字母“D”或不加任何标记。

4. 十六进制(Hexadecimal)

十六进制数是由 0，1，…，9，A，B，…，F(字母不区分大小写)16 个数字组成，其中：字母 A，B，…，F 分别代表 10，11，…，15。

在书写时，为了与其他进制数相区别，在十六进制数后面要加字母“H”。同时要求，当十六进制数的第一个字符是字母时，在第一个字符之前应加“0”。例如：100H，56EFH，0FFH，0ABCDH 都是十六进制数。

1.2.2　数制的转换

1. 非十进制数转换成十进制数

转换方法：将非十进制数根据其进制按位权展开，并以 10 进制的方式计算，得到的结果即为转换后的十进制数。

【例 1.1】 将 101.11B 和 254Q 分别转换成十进制数。

$101.11B = 1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$　　按权展开

$=5.75$　　按十进制方式计算

$254Q = 2\times8^2+5\times8^1+4\times8^0$　　按权展开

$=128+40+4=172$　　按十进制方式计算

2. 十进制数转换成非十进制数

将十进制数转换成非十进制数时，一般分两步，即整数部分和与小数部分转换。其中整数部分采用“除基数倒序取余数法”，小数部分则采用“乘基数顺序取整数法”。

【例 1.2】 将十进制数 110.25 转化为二进制数。

转换过程如图 1-2 所示。

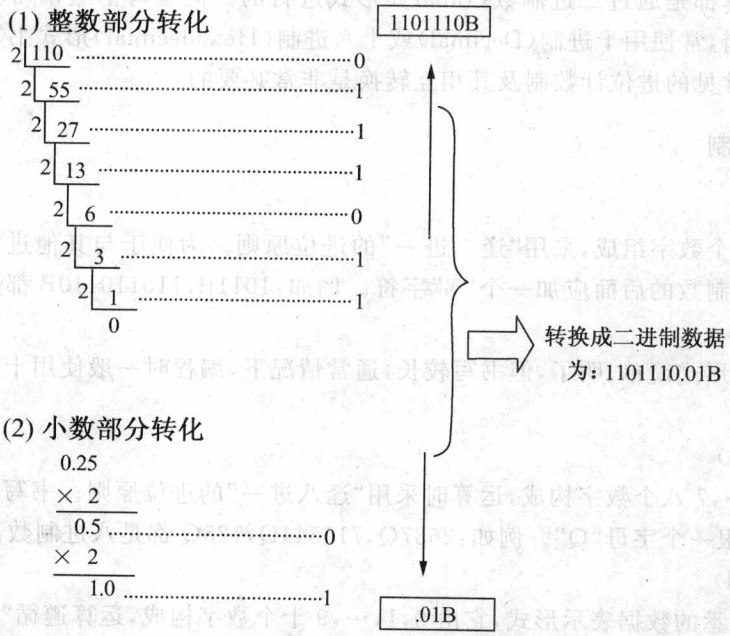

图 1-2　十进制数转换成二进制数

3. 二、八、十六进制数之间的转换

① 二进制转换为八进制的方法：从小数点开始向两边分组，每三个二进制数字为一组，不足的整数部分前补 0，小数部分右补 0，然后将每组转化成八进制即可。

【例 1.3】 011 010 110. 111B→326.7Q

② 二进制数转换成十六进制方法：从小数点开始向两边分组，每四个二进制数字为一组，然后将每组对应的十六进制数字写出来即可。分组时，如位数不够，则整数部分左

边补0，小数部分右补0。

【例1.4】 将11010110.111B转换为八进制和十六进制。

11010110.111B＝011 010 110. 111B＝326.7Q（八进制数）

＝1101 0110. 1110B＝0D6.EH（十六进制数）

③ 将八进制数转换成二进制数的方法：将八进制的每一位转换成三位二进制数即可。同样，将十六进制转换为二进制数，只需要将十六进制的每一位转换成四位二进制数。

【例1.5】 将十六进制数3D2BH转换为二进制数。

3D2BH＝0011 1101 0010 1011B

＝11110100101011B

习　题

1. 将下列十进制数分别转换为二进制、八进制和十六进制数。

①16　　②463　　③1022　　④365

⑤36.5　　⑥78.75　　⑦27　　⑧100

2. 将下列数据转换为十进制数。

①1011010B　　②10110110110.11B　　③3541Q

④256.75　　⑤0A254H　　⑥365.25H

3. 将以下二进制数分别转换为八进制和十六进制数。

①101011010B　　②1100110011B　　③1011110111011B

1.3　计算机中的有符号数的表示

计算机中的数由符号和数字组成。为便于运算，数的符号也用二进制数字表示，一般用数的最高位表示数的符号，当最高位为0时表示正数，为1时表示负数。例如，若把10010110看作8位有符号数，则它是一个负数。这种用数字表示正负的数通称为机器数。在计算机中，机器数可以用不同的表示方式，常用的有n位二进制数的原码、反码、补码三种表示法（其中n可以是8,16,32等）。

1.3.1　原　码

原码是一种比较直观的机器数表示方法，它与真值的区别仅仅是符号数字化。例如：＋1101B的8位原码表示为00001101B，－1101B的8位原码则为10001101B。

采用原码做乘除法运算比较方便，可采用绝对值直接运算，而符号单独处理。但对于加减法运算，原码表示就不太方便了。（为什么？）

1.3.2　反　码

生成反码的规则是：正数反码的符号位是0，其数值部分等于真值数值部分。负数反

码的符号位是1，其数值部分为真值二进制形式按位取反。例如，+1101B的8位反码表示是00001101B，-1101B的8位反码表示是11110010B。

1.3.3 补 码

大多数计算机中的数都是用补码表示的。补码的表示规则是：正数补码的符号位是0，其数值部分等于其真值数值部分。负数补码可按下列公式计算：负数X的补码$=2^n-|X|$（其中n为计算机的字长）。

【例1.6】 分别求1和-1字长为8的补码。

$[1]_{补}=00000001B$

$[-1]_{补}=2^8-1=11111111B$

注：

(1)+0和-0补码相同。

(2)可以用更简单的方法写出负数的补码，方法是：先将要转化数的绝对值扩充到相应的字长并按位取反，然后再加1，最后得到的数即为该负数的补码。

(3)补码运算规则：

① $[X+Y]_{补}=[X]_{补}+[Y]_{补}$

② $-[X]_{补}=[-X]_{补}$

【例1.7】 求-1101B的8位补码。

先将1101B扩展到8位，即00001101B；

对00001101B按位取反再加1，即$\overline{00001101}B+1=11110010B+1=11110011B$；

则11110011B即是-1101B的8位补码表示。

习 题

1. 以下数据均为十进制，求出相应的16位的补码。

① -263　② 47　③ -32760　④ 65323

2. 已知以下数据均为补码，求出其所表示的数据。

① 36H　② 85H　③ 0FF27H　④ 0A362H

1.4 计算机的字符表示

计算机只能识别二进制数，因此，计算机中的数字、字母、符号、控制字符等必须使用二进制数表示，目前最常用的编码的方案有ASCII码、Unicode码等，下面介绍几种最常见的编码方式。

1.4.1 ASCII码

目前计算机中用得最广泛的字符集及其编码是由美国国家标准局(ANSI)制定的

ASCII码(American Standard Code for Information Interchange,美国标准信息交换码),它已被国际标准化组织(ISO)定为国际标准,称为ISO 646标准。它用一个字节来表示字符,其中低7位为字符的ASCII码值,最高位为校验位。它共包含10个阿拉伯数字、52个英文大、小写字母、32个标点符号和运算符、34个控制码,共计128个字符,详细情况可参见附录中的ASCII码表。

注:Unicode码也是一种国际标准编码,采用两个字节编码,它给计算机使用的每个字符提供了一个唯一的编码。目前,它在网络、Windows系统和众多大中型软件中都得到广泛应用。

1.4.2 BCD码

BCD(Binary Code Decimal)码是用二进制形式来表示十进制数,即用四位二进制数表示一位十进制数,而四位二进制数之间的进位却采用十进制的方式,因此BCD码既有二进制数的特点,又有十进制数的特点。

例如:6234=0110 0010 0011 0100 BCD

引入BCD的主要目的是为了解决人们习惯的十进制与二进制之间的矛盾,也便于进行十进制与二进制数之间的转换,但在编程时,BCD码不能加标识符,只要按BCD码指令要求处理即可。

BCD码又分为压缩BCD码和非压缩BCD码两种表示形式。

1. 压缩BCD码

压缩BCD码是用4位二进制数来表示1位十进制数。例如,十进制数67转换成压缩BCD码形式为01100111B。

2. 非压缩BCD码

非压缩BCD码是用8位二进制数来表示1位十进制数,其中高4位为0。例如,十进制数67转换非压缩BCD码为00000110 00000111B。

习 题

1. 把以下数据分别以压缩BCD码和非压缩BCD码表示出来。
   ① 29　② 98　③ 1　④ 35
2. 以十六进制形式写出下列字符的ASCII码。
   ① D　② Y　③ y　④ 6　⑤ −　⑥ +
3. 试写出将以下十六进制数当成无符号整数或字符的ASCII码值时,它们所表示的十进制数或字符分别是什么?
   ① 42H　② 24H　③ 2BH　④ 61H　④ 39H