汇编语言程序设计

张志明　杨明广　主编

重庆大学出版社

内 容 简 介

本书是以 IBM PC 计算机 8086/8088 CPU 为基础，介绍汇编语言的基本原理和程序设计技术。本书重点介绍了 8086/8088 编程结构、指令系统、伪指令、汇编语言程序格式等汇编语言编程基础知识以及顺序、分支、循环、子程序、宏指令等基本程序设计技术和输入输出、中断程序设计技术。最后还介绍了 BIOS 中断调用、DOS 系统调用和模块化程序设计等实用设计方法和系统组织方法。

为了便于学习和实践，书中除了安排大量的例题、习题之外，还专门编写了上机实验一章，使实践环节和书本知识有机地结合在一起。

本书始终贯彻结构化程序设计思想，有利于初学者形成良好的程序设计风格和编程习惯。本书还注意难点分散，将部分指令结构编程技术一起介绍，便于初学者克服入门的困难。

本书可作为高等院校专科教材，以及成人教育、自学考试等同等水平的教材，同时也可作为广大工程技术人员学习汇编语言的参考书。

汇编语言程序设计

张志明　杨明广　主编

责任编辑：梁　涛　　版式设计：梁　涛

责任校对：蓝安梅　　责任印制：秦　梅

*

重庆大学出版社出版发行

出版人：张鸽盛

社址：重庆市沙坪坝正街 174 号重庆大学(A 区)内

邮编：400030

电话：(023) 65102378　　65105781

传真：(023) 65103686　　65105565

网址：http://www.cqup.com.cn

邮箱：fxk@cqup.com.cn (市场营销部)

全国新华书店经销

万州日报印刷厂印刷

*

开本：787× 1092　1/16　印张：20.25　字数：505 千

1997年 7 月第 1 版　　2005年 1 月第 9 次印刷

印数：40 001—43 000

ISBN 7-5624-1335-5/TP・113　　定价：26.00 元

序

面对知识爆炸, 社会学家们几乎都开出了一个相同的药方: 计算机。计算机也深孚众望, 以其强大的功能, 对人类做出了巨大的贡献, 取得了叹观止矣的成就。自它 1946 年 2 月 14 日在美国费城诞生以来, 至今已过“ 知天命 ”的年龄了。现在, 计算机已是一个庞大的家族。如果说, 它的成员占据了世界的每一个角落和每一个部门也并不过分, 甚至找不到这样一个文明人, 他的生活不直接或间接与计算机有关。目前, 全世界计算机的总量已达数亿台, 而且, 现在正以每年几千万台的速度增长。

作为计算机在信息传递方面的应用, 计算机加上网络, 被认为是和能源、交通同等重要的基础设施。这种设施对信息的传递起着异常重要的作用。西方发达国家和我们国家对此都非常重视。例如, 美国的信息高速公路计划, 全球通讯的“ 铱 ”计划, 我国也开始实行一系列“ 金 ”字头的国民经济管理信息化计划。这些计划中唱主角的设备便是计算机。计算机在各个方面的应用不胜枚举, 我们每个人都自觉不自觉地处于计算机包围中。

计算机对社会生产来说是一个产业大户, 对每个现代人来说是一个种工具, 对学生们来说, 它是一个庞大的知识系统。面对计算机知识的膨胀, 面对计算机及其应用产业的膨胀, 计算机各个层次的从业人员的需要也在不断膨胀, 计算机知识的教育也遍及从小学生到研究生的各个层次。

为了适应计算机教学的需要, 重庆大学出版社近几年出版了大量的计算机教学用书, 这一套教材就是一套适应专科层次的系列教材。我们将会看到, 这一套教材以系列、配套、适用对路, 便于教师和学生选用。如果再仔细研究一下, 将会发现它的一系列编写特色:

1. 这些书的作者们是一些长期从事计算机教学和科研的教师, 不少作者在以前都有大量计算机方面的著作出版。例如本系列书中的《Visual Fox Pro 中文版教程》的作者, 十年前回国后最早将狐狸软件介绍到祖国大陆, 这一本书已是他的第八本著作了。坚实的作者基础, 是

这套书成功的最根本的保证。

2. 计算机科学是发展速度惊人的科学, 内容的先进性、新颖性、科学性是衡量计算机图书质量的重要标准, 这一套书的作者们在这方面花了极大的功夫, 力求让读者既掌握计算机的基础知识, 又让读者了解最新的计算机信息。

3. 在内容的深度和知识结构上, 从专科学生的培养目标出发, 在理论上, 从实际出发, 满足本课程及后续课程的需要, 而不刻意追求理论的深度。在知识结构上, 考虑到全书结构的整体优化, 而不过分强调单本书的系统性。这样, 在学过这一套系列教材后, 学生们就可在浩瀚的计算机知识中, 建立起清晰的轮廓, 就会知道这些知识的前因后果, 就会了解这些知识的前接后续。使学生们能在今后的工作实践中得心应手。

4. 计算机是实践性很强的课程, 仅靠坐而论道是学习不了这些知识的。所以从课程整体设置来讲, 包括有最基本的操作技能的教材。对单本书来说, 在技术基础课和专业课中, 都安排有一定的上机实习或实验, 这样可使学生既具备一定的理论知识以利今后发展和深造, 又掌握实际的工作技能胜任今后的实际工作。

编写一套系列教材, 这是一个巨大的工程。这一套书的作者们, 重庆大学出版社的领导和编辑们, 都为此付出了辛勤的劳动。作为计算机工作者, 以此序赞赏他们的耕耘, 弘扬他们的成绩。

1997 年 6 月 15 日

前　言

《汇编语言程序设计》是计算机及其应用专业的核心课程之一。汇编语言是各种计算机语言中与硬件关系最密切、最直接的语言。学习汇编语言有助于了解计算机原理最基础的部分，为进一步学习操作系统、接口技术等课程打下坚实的基础。同时，汇编语言又是各种计算机语言中时间效率和空间效率最高的语言，由于它能够利用计算机所有硬件特性并能直接控制硬件，在计算机应用系统设计和过程控制中也是必不可少的。

汇编语言是面向机器的语言，所谓面向机器，就是说它的语句、语法完全由硬件设计所决定。不同的硬件系统，其根本的区别在于其核心部件 CPU 的设计不同，也就是说 CPU 不同的系统具有不同的汇编语言。尽管如此，各种不同 CPU 的汇编语言之间还是有着基本的、共同的原理与处理方法。只要学通一种汇编语言，就不难面对各种变化的情况。所以，学习汇编语言程序设计必须结合一种具体的 CPU 来进行。当前国内市场上的主流机型是以 Intel 80X86 系列 CPU 为核心的 16 位和 32 微机，我们认为目前在大专层次上采用 8086/8088 系统组织教学仍然是最佳选择。掌握了 DOS 环境下的 8086/8088 汇编语言程序设计，对掌握更高层次的汇编语言也就有了基础。

本书共 12 章。第 1 章为基础知识，第 2 章和第 3 章介绍 8086/8088 编程结构和指令系统，第 4 章是汇编语言程序格式、伪指令和汇编语言上机过程，第 5、6 两章讲解顺序、分支、循环、子程序、宏指令等基本程序设计技术，第 7 章通过数值运算与代码转换程序对程序设计综合举例，第 8、9 两章讲述输入输出和中断程序设计，第 10 章和 11 章介绍 BIOS 中断调用、DOS 系统调用和模块化程序设计等实用的程序设计及系统组织方法。

汇编语言程序设计是一门实践性很强的课程，只有通过大量的练习和上机实践才有可能掌握和提高。为此，本书中除了安排一定数量的例题、习题之外，还专门编写了第 12 章上机实验，有利于使实践环节和课堂教学更有机地结合在一起。

本书编写中，贯彻了结构化程序设计思想，注意对初学者进行良好的程序设计风格和编程习惯的培养。为了便于初学者学习，本教材还注意将难点分散，如将转移地址寻址的寻址方式分散到第 5 章结合分支程序设计介绍，将乘、除法指令、BCD 码调整等内容分散到第 6 章结合数值运算程序设计一起介绍等。

本书第 1、2、4 章由云南工业大学张民坤编写，第 3、5、6、7 章由成都电子高等专科学校杨明广编写，第 9 章由贵阳金筑大学翁培松编写，第 8、10、11、12 章由贵州大学张志明编写。全书由张志明、杨明广主编，由电子科技大学王正智教授主审。

本书是作为高等院校专科教材编写的，同时可供成人教育、自学考试等同等水平的其他教学使用，也可作为广大工程技术人员学习汇编语言的参考书。

由于编者水平有限，书中缺点和错误之处，敬请广大读者批评指正。

作者

1997 年 3 月

第 1 章　基础知识

汇编语言是面向机器的计算机编程语言。比起学习高级语言来,学习汇编语言要求具备更深入的计算机理论知识及必要的数学基础。本章简要介绍了计算机软硬件组成的体系结构和汇编语言的概念,并重点介绍了学习汇编语言程序设计必须掌握的数制转换、机器数的表示方法、字符的 ASCII 编码等基本知识。

1.1　计算机系统

一个完整的计算机系统可以划分为硬件和软件两个子系统。硬件子系统包括主机和各种外部设备,即计算机的电路、机器等物理实体。软件子系统是使用、管理和维护计算机正常运行而编制的各种程序的总和。

1.1.1　硬件子系统

硬件子系统的基本组成如图 1.1 所示。其中主机包括中央处理单元 CPU(Central Processing Unit) 、内存储器(Memory) 和输入/输出接口三个主要组成部分,由系统总线把它们三者连接在一起。

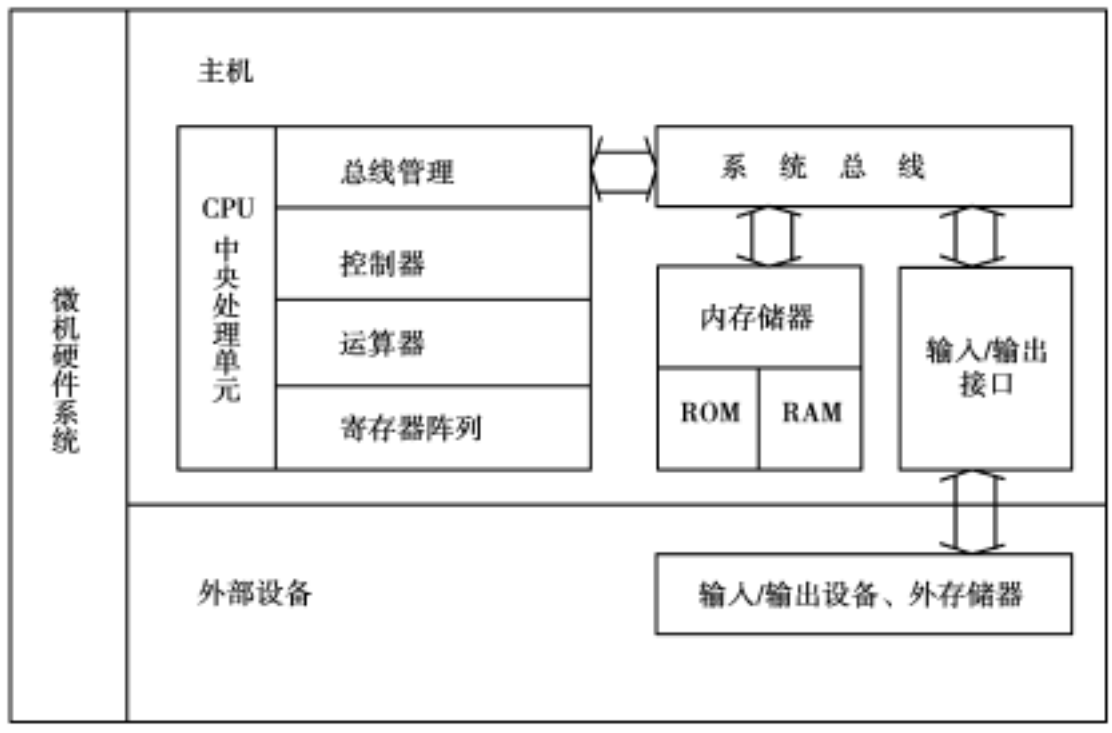


图 1.1　计算机硬件子系统基本组成

中央处理单元是整个计算机系统的心脏,包括运算器和控制器两个主要部分。运算器执行所有的算术和逻辑运算指令,控制器则负责全机的控制操作,将指令逐条从存储器中取出,经译码分析后发出取数、执行、存数等控制命令,以保证正确完成程序所要求的功能。

内存储器是计算机的记忆部件。内存储器由一个个存储单元组成,每个存储单元有一个

地址。人们编写的程序及程序中所用到的数据、编码信息及中间结果在程序运行时都必须放在内存储器中。

内存储器又简称内存,分为只读存储器 ROM(Read Only Memory) 和随机读写存储器 RAM (Random Accese Memory) 两类。ROM 用于存放起动、自检、设备驱动程序等永久性信息; RAM 由利用极间电容表示信息的半导体器件组成,其中存放的信息读出时不变,写入时新的内容取代原来的内容,断电后所有的内容都将丢失。

主机中的接口电路和外部设备一起构成硬件系统中的 I/O 子系统,外部设备包括 I/O 设备及大容量存储器。I/O 设备即输入/输出设备,又简称外设。常见的输入设备有键盘、鼠标器,常见的输出设备有终端显示器、打印机等。大容量存储器则是指存储大量信息的外存储器,如磁盘、磁带、光盘等。由于内存的容量有限,所以计算机用外存储器作为内存的后援设备,它的容量可以比内存大很多,但存取信息的速度比内存慢得多,所以除必要的系统程序(如 DOS 的内部命令) 外, 一般程序(包括数据) 是存放在外存中的。只有当运行时,才把它从外存传送到内存的某个区域,再由中央处理机控制执行。

系统总线把 CPU、存储器和 I/O 设备连接起来,以传送各部件之间的信息。系统总线包括数据总线、地址总线和控制总线。数据总线传送数据(包括指令代码、原始数据、中间数据和结果数据) , 地址总线上的信息(即地址) 指出数据的来源和目的地,控制总线则传送 CPU 对存储器或 I/O 设备的控制命令和 I/O 设备对 CPU 的请求信号。系统总线的工作由总线控制逻辑负责指挥。

1.1.2 软件子系统

计算机软件是计算机系统的重要组成部分,它可以分为系统软件和用户软件两大类。系统软件是由计算机的生产厂家提供给用户的一组程序,这些程序是用户使用机器时为产生、准备和执行用户程序所必需的。用户软件则是用户自行编制的各种程序。图 1.2 表示了计算机软件系统的层次。

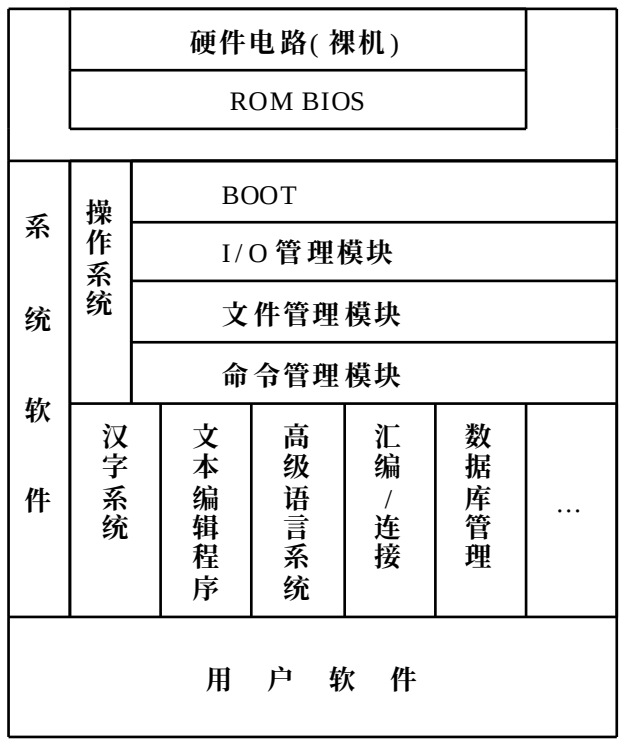


图 1.2 计算机软件层次图

系统软件的核心称为操作系统 (Operating System) 。操作系统是系统程序的集合。它的

主要作用是对系统的硬、软件资源进行合理的管理，为用户创造方便、有效和可靠的计算机工作环境。DOS(Disk Operating System)即磁盘操作系统，是IBM PC系列微机中最常用的操作系统。

操作系统由输入/输出管理、文件管理、命令管理等模块组成，平时存放在外存储器中，开机时由引导程序装入内存，计算机运行时它一直存在于内存中，随时接受用户命令，执行相应的动作。

输入输出管理模块中的I/O驱动程序(I/O Driver)用来对I/O设备进行控制和管理。当系统程序或用户程序需要使用I/O设备时，只要发出命令，执行I/O驱动程序，就可完成CPU和I/O设备之间的信息传送。

文件管理系统(File Management)用来处理存储在外存储器中的大量信息，它可以和外存储器的设备驱动程序相连接，对存储在其中的信息以文件(File)形式进行存取、复制及其他管理操作。

文本是指由字母、数字和符号等组成的信息，它可以是一个用汇编语言或高级语言编写的程序，也可以是一组数据或一篇文章。文本编辑程序(Text Editor)用来建立、输入或修改文本，并把它存入大容量的外存储器中。例如：随DOS早期版本提供的行编辑程序EDLIN可用来建立文件，修改文本，能对文本行进行插入、删除、编辑和显示等操作。DOS 5.0以上的版本中提供的编辑软件EDIT和常见的中西文字处理程序WPS都是全屏幕编辑程序，可提供多种功能及命令菜单，使文本的建立和修改更加方便。

高级语言与数据库管理系统为计算机用户提供了二次开发的编程工具，各种语言的翻译及处理软件将用户源程序翻译为计算机机器能够识别的机器语言目标程序。

随着计算机技术的不断发展，更多的、功能更强大的系统软件还在源源不断地提供给用户。对于用户来说，这些系统软件是十分重要的信息资源。

1.2 计算机语言

1.2.1 机器语言

1. 机器指令

机器指令是由硬件电路设计决定的、能为计算机所接受的一组二进制代码。机器指令规定了要求计算机完成的操作及其操作的对象(数据或存储单元地址)。

不同的机器指令可以有不同的长度和不同的执行时间。

2. 指令系统

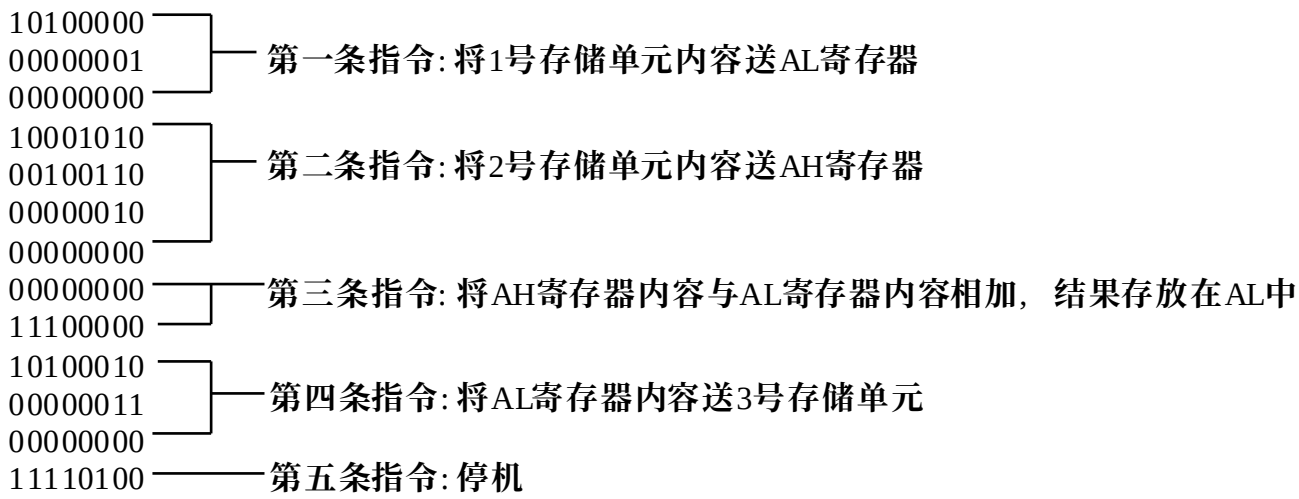
每种特定的CPU所具有的全部指令的集合构成该CPU的指令系统。不同的CPU具有不同的指令系统。

3. 机器语言程序

机器语言程序就是为了让计算机完成特定任务而按一定顺序组织到一起的机器指令序列。计算机的工作过程就是CPU周而复始地从内存储器中取指令、执行指令的过程。

下面是一小段机器语言程序的例子，它完成将1号存储单元和2号存储单元中的两个数

相加，结果送入 3 号存储单元的操作。



从上例中可以看出，机器语言程序完全由二进制代码串组成，它难于记忆和使用，编出的程序也难读懂。在实际工作中很少使用。

1.2.2 汇编语言

由于机器语言在编程的过程中难读、易错，人们想到用一种帮助记忆的符号来代替机器指令。如加法指令的操作码 00000000 用助记符号 ADD 来代替。显然，助记符较二进制代码容易记忆，所以在计算机的发展过程中出现了汇编语言。

1. 汇编指令

汇编指令是用助记符号表示的机器指令，它与机器指令一一对应。

汇编指令的引入使机器语言程序变成人们比较容易识别其含义的汇编语言程序。如上面的例子用汇编指令改写如下：

```
MOV  AL, DATA1
MOV  AH, DATA2
ADD  AL, AH
MOV  RESULT, AL
HLT
```

2. 汇编程序

汇编指令容易被人们所理解，而计算机却不能直接识别。要让机器接受汇编指令还需要有一个将汇编指令翻译为机器指令的过程，这个过程称为汇编。用手工查表完成这一过程称为手工汇编。利用计算机程序自动完成这一过程称为机器汇编。汇编程序就是把用汇编指令编写的程序翻译成机器语言程序的一种系统软件。

IBM PC 机中的汇编程序有 ASM 和 MASM 两种，ASM 称为小汇编程序，它只需较小的存储区。MASM 称为宏汇编程序，它需要的存储区较大，但功能较强，且具有宏汇编能力，ASM 则不具备这种能力。

3. 伪指令

为了完成汇编任务，汇编程序要求用户提供一些必要的信息，如程序的存放地址、变量、常量、标号等符号的定义等。伪指令就是为汇编程序提供指示信息的命令。

伪指令没有对应的机器指令，汇编时不产生机器码。

4. 汇编语言

汇编指令、伪指令和汇编程序一起组成了汇编语言。汇编语言直接面向机器,用汇编语言编制的程序简洁、快速,常用于对运行速度要求较高的实时控制等场合。

用汇编语言编制的用户程序称为汇编语言源程序。

5. 汇编程序的简单工作过程

为了实现对汇编语言源程序的汇编,汇编程序需要借助一些表格。汇编过程主要就是对这些表格进行查询检索。

通常应有以下表格:

1) 机器指令与汇编指令操作码对照表

2) 伪指令操作表

汇编语言为了对源程序进行汇编,必须对源程序逐行扫描,把读到的语句进行分析转换。这种扫描一般应进行两遍。第一遍扫描只完成符号表的建立,将扫描到的所有符号(变量、常量、标号等)及其值集中在一起形成一个符号表。第二遍扫描才将每个程序行转换成机器指令代码或数据。两次扫描之间通过扫描时的地址指针建立符号地址或指令代码地址的联接分配关系。

两次扫描的过程可以用图 1.3 表示。

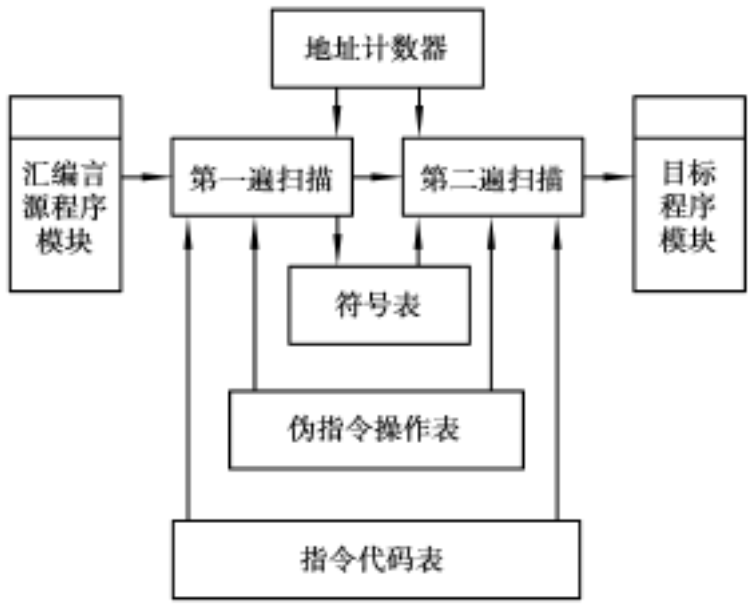


图 1.3　汇编过程简图

1.2.3　高级语言

高级语言是用接近自然语言和数学公式的形式编程的计算机语言。它脱离了机器的硬件系统,用人们更易于理解的方式编写程序,所以说高级语言是面向科学计算和实际问题的语言。

由于计算机本身只能识别机器语言,所以高级语言必须翻译成机器语言才能被计算机所接受和执行。

高级语言翻译成机器语言的方式有两种:一种是先把高级语言源程序一次性翻译成机器语言目标程序,然后再在机器上运行目标程序。这种翻译程序称为编译(Compile),多数高级语言如:PASCAL, FORTRAN 等都采用这种方式。另一种是直接把高级语言程序在机器上运行,一边翻译一边执行。这种逐句翻译、逐句执行的方式称为解释(Interpret),如 BASIC 语言、

dBASE 数据库管理语言即采用这种方式。

由于高级语言需要向翻译程序提供更多的附加信息,将占用更大的存储空间及花费更多的执行时间,其运行效率比直接面向机器的汇编语言要低得多。这是为提高其编程效率而付出的必要的代价。高级语言常用于科学计算和信息管理等方面。

C 语言是目前最为流行的一种高级语言,它的灵活性较强,程序设计人员能够用它直接操纵 CPU 寄存器和内存单元。因此 C 语言又具有低级语言面向机器的一些特征,用 C 语言编写的程序运行速度几乎与汇编语言相当,而且便于在不同的系统之间互相移植。当然,编程的灵活性使得它不如一般高级语言使用方便,因此人们通常又把它称作中级语言。

1.3 计算机中数据的表示方法

1.3.1 数制及其转换

1. 进位计数制

进位计数制是一种计数的方法,人们最常用的是十进制计数法,这是由人类的祖先以双手十指为原始计数器而养成的计数习惯。实际上可以采用任何大于 1 的整数进制。如在生活中,还有以 12 个为 1 打、12 打为 1 盒、12 盒为 1 箱的十二进制以及以 60 s 为 1 min,60 min 为 1 h的六十进制等。

电子计算机的硬件基础是开关量控制的数字逻辑电路。在计算机中为便于信息存储及计算的物理实现而采用二进制。在二进制中只有 0、1 两个数字,并遵循“逢二进一”的规则。

一个任意的十进制数

$$a_n a_{n-1}\cdots a_0 \cdot a_{-1}a_{-2}\cdots a_{-m}$$

可以表示为加权求和的形式:

$$a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \cdots + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + a_{-2} \cdot 10^{-2} + \cdots + a_{-m} \cdot 10^{-m}$$

其中 n 为整数位数减 1, m 为小数位数, a_i(i = 0, 1, …, n) 是 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 十个数码中的一个。

十进制数的基数为 10,即其数码的个数为 10,且遵循逢十进一的规则。上式中相应于每位数字的 10^j 称为该位数字的权,所以每位数字乘以其权所得到的乘积之和即为所表示数的值。例如:

$$5123.78 = 5\times 10^3 + 1\times 10^2 + 2\times 10^1 + 3\times 10^0 + 7\times 10^{-1} + 8\times 10^{-2}$$

由此可以引出进制公式:

若 r 为大于 1 的整数,任一有理数 N 总可以用下式表示:

$$N = \sum_{j=-m}^{n-1} a_j \cdot r^j = a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \cdots + a_0 \cdot r^0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \cdots + a_{-m} \cdot r^{-m}$$

其中: 1) r 为进制基数, r 进制, 逢 r 进 1。

2) a_j 为加权系数, 集合 $A = \{a_j\} = \{0, 1, 2, \cdots, r-1\}$。$a_{max} = r - 1$。

3) r^j 为各位数相应的权。

如: r = 8, 八进制, 逢八进一, $A = \{a_j\} = 0, 1, 2, 3, 4, 5, 6, 7$。 $a_{max} = r - 1 = 7$。

例 1.1

$$(534.76)_8 = 5\times 8^2 + 3\times 8^1 + 4\times 8^0 + 7\times 8^{-1} + 6\times 8^{-2}$$
$$= 5\times 64 + 3\times 8 + 4\times 1 + 7\times 1/8 + 6\times 1/64$$
$$= 348.96875$$

又如: r = 16, 十六进制, 逢十六进一, 需要 16 个数码, 最大数码对应十进制数的 15, 引入字母 A, B, C, D, E, F 表示 10 到 15 六个数码, 则十六进制的数码集合为 $A = \{a_j\} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$。

例 1.2

$$(2AC)_{16} = 2\times 16^2 + 10\times 16^1 + 12\times 16^0$$
$$= 2\times 256 + 10\times 16 + 12\times 1$$
$$= 684$$

再看二进制的情况: r = 2, 二进制, 逢二进一, $A = \{a_j\} = \{0, 1\}$。即二进制只有 0、1 两个数码, 最大数码为 1。

例 1.3

$$(10100101)_2 = 1\times 2^7 + 0\times 2^6 + 1\times 2^5 + 0\times 2^4 + 0\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0$$
$$= 128 + 32 + 4 + 1 = 165$$

在计算机中, 通常用数字后面跟一个英文字母来表示该数的数制。十进制一般用 D(Decimal) 、二进制数用 B(Binary) 、八进制数用 O(Octal) 、十六制数用 H(Hexadecimal) 来表示。例: 117D, 1110101B, 0075H, ……。通常十进制的数制符号 D 可以省略。

2. 不同数制之间的转换

(1) 各种数制转换成十进制

如上节所示, 其他进制数转换成十进制数只需进行加权求和。特别地, 对于二进制到十进制的转换只需对二进制数中为 1 的位进行位权求和即可。

二进制数的位权:

…	10	9	8	7	6	5	4	3	2	1	0	- 1	- 2	- 3	…
…	1024	512	256	128	64	32	16	8	4	2	1	0.5	0.25	.125	…

例如: 对于$(10000011)_2$ 只要将它的第 7, 1, 0 位的位权 128, 2, 1 相加, 即可等到十进制数 131 的结果。

(2) 十进制数转换为二进制数

十进制数转换为二进制数时, 一般对整数部分采用除 2 取余法, 而对小数部分采用乘 2 取整法。

除 2 取余法: 把要转换的十进制数的整数部分连续除以 2, 逐次记下余数直到商为 0 时为止。

例 1.4　把十进制数 119 转换为二进制数。

$$
\begin{array}{r|l}
2 & 119\cdots1 \quad (a_0=1) \\
2 & 59\cdots1 \quad (a_1=1) \\
2 & 29\cdots1 \quad (a_2=1) \\
2 & 14\cdots0 \quad (a_3=0) \\
2 & 7\cdots1 \quad (a_4=1) \\
2 & 3\cdots1 \quad (a_5=1) \\
 & 1 \quad (a_6=1)
\end{array}
$$

所以 119D = 1110111B。

乘 2 取整法: 把待转换的十进制数的小数部分连续乘以 2, 逐次记下乘积整数部分的值, 直到得到满足精度要求的位数为止。

例 1.5　把十进制数 0.725 转换为二进制数(精确到二进制数的 8 位小数) 。

将下图的十字线右下区域中的数乘以 2, 左下区域中记下每次乘积的整数值即相应二进制小数位的值。

在下面的计算中可以看到, 被乘数从小数后第 7 位起发生循环, 无限制地重复下去不再具有实际意义, 根据题目要求取一定位数的近似值即可。

	0	725× 2
(a_{-1} =1)	1	450× 2
(a_{-2} =1)	0	90 × 2
(a_{-3} =1)	1	8 × 2
(a_{-4} =0)	1	6 × 2
(a_{-5} =1)	1	2 × 2
(a_{-6} =1)	0	4 × 2
(a_{-7} =1)	0	8 × 2
(a_{-8} =1)	1	6 × 2

所以 N = 0.725D≈0.10111001B。

以上两例分别为整数和纯小数的转换, 带小数的转换只要对整数部分和小数部分分别进行, 此处不再赘述。

(3) 十六进制数与二进制数之间的互换

在计算机内部, 数的运算和存储都是采用二进制。但是, 二进制数对于人的阅读、书写及记忆都是很不方便的。十进制数虽然是人们最熟悉的一种进位计数制, 但它与二进制数之间并无直接的对应关系。为了便于人们对二进制数的描述, 应该选择一种易于与二进制数相互转换的数制。显然, 使用 2^n 作为基数的数制是能适合这种要求的。

n 位二进制数可以表示 2^n 个不同的数。

例如: 3 位二进制数可以表示 8 个不同数:

二进制数	000	001	010	011	100	101	110	111
相应的十进制数	0	1	2	3	4	5	6	7

4 位二进制数则可表示十进制数的 0 ～15 共 16 个数:

二进制数	0000	0001	0010	0011	0100	0101	0110	0111
相应的十六进制数	0	1	2	3	4	5	6	7
相应的十进制数	0	1	2	3	4	5	6	7
二进制数	1000	1001	1010	1011	1100	1101	1110	1111
相应的十六进制数	8	9	A	B	C	D	E	F
相应的十进制数	8	9	10	11	12	13	14	15

为了便于阅读及书写，通常可以直接按上表的对应关系用八进制数或十六进制数来表示二进制数。计算机中存储信息的基本单位为一个二进制位(bit)，用来表示 0 和 1 两个数码，而计算机中常用 8 位二进制数组成的一个字节(byte)来表示数据或字符。因此，字节也就成为计算机中存储信息的单位。计算机的字长一般都选为字节的整数倍，如 16、32 或 64 位等。一个字节由 8 位组成，它可以用两个四位组(又称半字节)来表示，所以可以比较方便地用十六进制数来表示二进制数。

由于十六进制的基数是 2 的幂，这两种数制之间的转换十分容易。一个二进制数，只要把它从低位到高位每 4 位组成一组，直接用十六进制数码来表示就可以了。若二进制数的位数不是 4 的倍数，则需要在小数点左边的高位或小数点右边的低位用 0 补足 4 位。

例 1.6　将二进制数 1010011.101011B 转换为十六进制数。

将高位补 1 个 0，低位补 2 个 0 后，原数按四位一节表示为：

0 101，0 011 . 1 010，1 100

5　　3　.　A　　C

即　　1010011.101011B = 53.ACH

反之，把十六进制数中的每位用 4 位二进制数展开，就得到相应的二进制数。

例 1.7　将十六进制数 B19AH 展开为二进制数。

B　　1　　9　　A

1011，0001，1001，1010

即　　B19AH = 1011000110011010B

(4) 十六进制数和十进制数之间的转换

十六进制数与十进制数之间的转换，可以用前面讲到的加权求和，除 16 取余、乘 16 取整等基本方法进行。

由于二进制数和十六进制数之间的转换简便易行，在实际进行十六进制与十进制互换时，通常通过二进制进行。为把一个十进制数转换为十六进制数，可以先把该十进制数转换为二进制数，然后再转换为十六进制数。反之，要把一个十六进制数转换为十进制数，也可采用同样的办法。这样做的好处是将所有的计算工作集中到二进制与十进制的转换上，避免记忆更多的公式。

例 1.8　完成下列十六进制数与十进制数的互换。

(1) A031H = _____D

A031H = 1010000000110001

$= 2^{15} + 2^{13} + 2^{5} + 2^{4} + 2^{0}$

= 32768 + 8192 + 32 + 16 + 1

= 41009

(2) C4.8H = _____D

C4.8H = 11000100.1000B

$= 2^{7} + 2^{6} + 2^{2} + 2^{-1}$

= 128 + 64 + 4 + 0.5

= 196.5

(3) 119D = _____H

前面在例 1.4 中已求得 119D = 1110111B, 所以可直接写出十六进制形式:

119D = 1110111B = 77H

(也可用除 16 取余法: 119÷ 16 商 7 余 7)

(4) 1027D = _____H

$$1027D = 1024 + 3 = 2^{10} + 3$$
$$= 10000000000B + 11B$$
$$= 010000000011B$$
$$= 403H$$

本例说明通过二进制实现十进制与十六进制互换, 有可能利用二进制的一些特征数字简化计算, 本例中用到第十位二进制数的位权等于 2 的十次方这一特征数字。

另一类有用的二进制特征数字是全 1 数字, n 位全 1 的二进制数等于 $2^n - 1$。如: 1111B = $2^4 - 1 = 15$, 111111B = $2^6 - 1 = 63$ 等。

(5) 61D = _____H

61D = 63 - 2 = 111111B - 10B = 00111101B = 3DH

1.3.2 二进制数和十六进制数运算

1. 二进制数的算术运算规则

加法:	乘法:
0 + 0 = 0	0× 0 = 0
0 + 1 = 1	0× 1 = 0
1 + 0 = 1	1× 0 = 0
1 + 1 = 10	1× 1 = 1

2. 二进制数的逻辑运算

逻辑与:	逻辑或:	逻辑异或:	逻辑非:
0∧0 = 0	0∨0 = 0	0 0 = 0	$\overline{1} = 0$
0∧1 = 0	0∨1 = 1	0 1 = 1	$\overline{0} = 1$
1∧0 = 0	1∨0 = 1	1 0 = 1	
1∧1 = 1	1∨1 = 1	1 1 = 1	

逻辑运算按位进行, 不同位之间不发生进位或借位关系。

例 1.9 两个变量的值分别为 X = 00FFH 和 Y = 5555H, 求 X∧Y; X∨Y; X Y, $\overline{X}$; $\overline{Y}$

X = 0000 0000 1111 1111

Y = 0101 0101 0101 0101

X∧Y = 0000 0000 0101 0101 = 0055H

X∨Y = 0101 0101 1111 1111 = 55FFH

X Y = 0101 0101 1010 1010 = 55AAH

$\overline{X}$ = 1111 1111 0000 0000 = FF00H

$\overline{Y}$ = 1010 1010 1010 1010 = AAAAH

3. 二进制数移位运算

二进制左移一位等于原数乘以 2, 右移一位等于原数除以 2。这是由于移位时位权倍增或倍减所致。

如: 一个八位二进制数 00010100B = 14H = 20D

若左移一位低位补 0 得: 00101000B = 28H = 40D

而右移一位高位补 0 得: 00001010B = 0AH = 10D

4. 十六进制数的加减运算

十六进制的加法运算可按照逢“ 十六进一 ”的规则直接进行。当两个一位数之和 S 小于 16 时, 与十进制数同样处理, 如两个一位数之和 S≥16 时, 则应该用 S - 16 及进位 1 来取代 S。

十六进制的减法也与十进制类似, 够减时可直接相减, 不够减时服从向高位借 1 为 16 的规则。

例 1.10

(1) 求 9A3C + 6BA5　　(2) 求 65A5 - 0A3C

```
    9A3C              65A5
+   6BA5          -   0A3C
--------          --------
  1 05E1            5B69
```

1.3.3　有符号数的编码

计算机中的数称为机器数, 机器数的数值是用二进制数来表示的, 机器数的符号也是用二进制数表示的。把一个数连同其符号在机器中的表示加以数值化, 这样的数称为有符号机器数, 简称有符号数。一般用最高有效位来表示数的符号, 正数用 0 表示, 负数用 1 表示。有符号数可以用不同的编码来表示, 常用的有原码、补码和反码三种表示方法。由于作加减法运算时补码表示法的符号位可以参加运算, 较好地反映了机器数的运算规律, 所以多数情况下有符号数采用补码表示法。这里只介绍补码表示法。

1. 补码的定义

补码的定义与机器数的位数有关。

n 位二进制数的补码 $〔X〕_{补} = 2^n - X$, 当 n 等于 8 时, $〔X〕_{补} = 2^8 + X$。

如: 求 $[100]_{补}$

因为　$2^8 = 100000000B$, (100D) = 01100100B

所以

```
  100000000B
+ 01100100B
------------
=101100100B
```

由于在机器数中只能存放低 8 位, 它就是 $[100]_{补}$, 即 $[100]_{补} = 01100100B = 64H$。

由此可见, 补码表示法中, 正数采用绝对值表示, 即机器数的最高有效位为 0, 表示符号为正, 机器数的其余部分则表示数的绝对值。

例如: 假设机器字长为 8 位, 则 $[+1]_{补} = 00000001$, $[+127]_{补} = 01111111$, $[0]_{补} = 00000000$。

又如: 求 $[-100]_{补}$

因为　$2^8 = 100000000B$, (-100D) = -(01100100B)

所以　　100000000B

　　+(- 01100100B)

　　=10011100B

即　　[- 100] $_{补}$ =10011100B=9CH

根据同样的算法可以求得[- 1] $_{补}$ = 2^{8} - 1 = 11111111B, [- 127] $_{补}$ = 2^{8} - 127 = 10000001B。显然,最高有效位为1,表示该数的符号为负。

对10000000这个数,在补码表示法中被定义为 - 128的补码。

可以用“求反加1”的简单方法得出一个负数的补码:先写出该负数的相反数(正数)的补码,然后将其按位求反(即0变为1,1变为0),最后再在末位(最低位)加1就得到该负数的补码。

现以机器数长度为8位时的补码为例说明这种方法的根据如下:

将补码定义式变形为

$$〔X〕_{补} = (2^{8} - 1) + X + 1$$

如前所述,(2^{8} - 1)为8位全1的二进制特征数,即

$$(2^{8} - 1) = 11111111B$$

当X为负数时,加上X等于减去X的绝对值,而用全1特征数减去某数等于将该数各位求反,所以对于负数来说,求补码公式变成了对其绝对值的补码“求反加1”。

例1.11　如机器字长为8位,求+46和 - 46的补码。

+46D的补码表示为　　00101110 (2EH)

按位求反　　11010001

末位加1　　11010010 (D2H)

即　　[+46] $_{补}$ =2EH, [- 46] $_{补}$ =D2H

例1.12　机器字长为16位,写出N = - 117D的补码表示。

+117D可表示为　　0000, 0000, 0111, 0101

按位求反后为　　1111, 1111, 1000, 1010

末位加1后为　　1111, 1111, 1000, 1011　(FF8BH)

2.有符号数的表示范围

8位二进制数可以表示 2^{8} =256个数,当它们是用补码表示的有符号数时,数的表示范围是 - 128≤N≤ +127。

一般来说,n位二进制数补码表示的数的表示范围是:

$$-2^{n-1} \leq N \leq 2^{n-1} - 1$$

所以,n=16时的数的表示范围是:

$$-32768 \leq N \leq +32767$$

1.3.4　补码的运算与溢出判断

1.求补运算

如前所述,对一个正数的补码按位求反加1可以得到此数相反数的补码。把这种对一个二进制数按位求反后在末加1的运算称为求补运算。可以证明对一个数的补码作求补运算的