全国本科计算机应用创新型人才培养规划教材

汇编语言程序设计

主 编 张光长

北京大学出版社
PEKING UNIVERSITY PRESS

全国本科计算机应用创新型人才培养规划教材

汇编语言程序设计

主　编　张光长
副主编　朱振玉　李永新
参　编　马甲军

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书以 80x86 系列微机为特定对象，从程序员角度介绍 80x86 系统结构和相关资源，以及常用的基本指令；在此基础上，主要讲述使用汇编指令构造顺序结构、分支结构和循环结构的一些“标准化”方法，以及模块化程序设计的基本知识、基本原理和相关技术与技巧，如参数传递方法、局部变量分配方法与技巧、多模块程序汇编连接方法等；此外，还介绍浮点运算的程序设计方法及 SIMD 指令集等内容。

本书可以作为计算机专业本科生的专业教材，也可以作为深入学习计算机科学的读者的自学教材，还可以作为非计算机专业的研究生、本科生、专科生和从事汇编语言程序设计的技术人员的参考书。

图书在版编目(CIP)数据

汇编语言程序设计/张光长主编. —北京：北京大学出版社，2009.7

(全国本科计算机应用创新型人才培养规划教材)

ISBN 978-7-301-15250-8

Ⅰ. 汇… Ⅱ. 张… Ⅲ. 汇编语言—程序设计—高等学校—教材 Ⅳ. TP313

中国版本图书馆 CIP 数据核字(2009)第 091142 号

书　　名：汇编语言程序设计
著作责任者：张光长　主编
策 划 编 辑：乐和琴
责 任 编 辑：刘　丽
标 准 书 号：ISBN 978-7-301-15250-8/TP・1012
出　版　者：北京大学出版社
地　　址：北京市海淀区成府路 205 号　100871
网　　址：http://www.pup.cn　http://www.pup6.com
电　　话：邮购部 62752015　发行部 62750672　编辑部 62750667　出版部 62754962
电 子 邮 箱：pup_6@163.com
印　刷　者：北京大学印刷厂
发　行　者：北京大学出版社
经　销　者：新华书店
787mm×1092mm　16 开本　17.25 印张　400 千字
2009 年 7 月第 1 版　2009 年 7 月第 1 次印刷
定　　价：28.00 元

序

本套教材经过全国几十所高等学校老师一年多的努力，终于与广大读者见面了。我相信，它一定会受到全国高等学校计算机界老师和同学们的热烈欢迎。

随着信息技术的飞速发展，单一培养模式已经不能满足社会对计算机专业人才多样化的需求。应对这一变化的最佳办法，就是采用多种模式的培养方式。当前，高等学校的计算机教育正处于从过去的单一培养模式向多种培养模式的转变过程中，多种模式的培养方式将是必然的发展方向。

多种模式的培养方式包括：培养人才的类型不同(研究型，应用型)；专业方向不同(计算机软件，计算机网络，信息安全，信息系统，计算机应用技术等)；课程设置的多样性等。

同时，高等教育对科技人才培养的要求是：不但要培养研究型科技人才，还要为国家培养更多的应用型科技人才(或称工程型科技人才)。也就是说，培养应用型科技人才是百分之九十以上的普通高等学校的主要任务。

本套教材正是为适应多种模式培养方式的要求，并且着重于培养计算机领域高级应用型科技人才的需求，而组织编写的。

本套教材具有如下特点。

1. 基础理论够用

计算机专业所需的基础理论知识以够用为准，不是盲目扩张。如数字系统的基础知识，计算机的基本组成原理和体系结构的基础知识，离散数学的基础知识，数据结构和算法的基础知识，操作系统的基础知识，程序设计的基础知识等，都进行了必要的讲解介绍。

2. 强调理论联系实际，学以致用

每本教材的编写都将“理论联系实际，学以致用”的原则贯彻始终。例如，《计算机组成原理和体系结构》结合现代的计算机讲解，使学生学完之后，确切掌握现代计算机的组成、结构和工作原理；又如，《程序设计》结合实例讲解，使学生学完之后，真正能够动手编写程序。

3. 强调教材的配套性

根据多年组织教材的经验，只有配套性好的教材才最受教师和学生们的欢迎。我们这套教材，尽量做到了课堂教材、实训教材和教学课件完全配套，以方便教学使用。

另外，本套教材提供的是一套应用创新型计算机教育系列教材，可供不同类型学校依照自己的教学计划，根据自身的需要进行选用。

现在把这套教材奉献给全国计算机界的朋友们，真诚希望大家能够喜欢。本套教材难免会有诸多缺点或不到之处，还希望得到大家的批评和指正。

全国高等学校计算机教育研究会课程与教材建设委员会主任

李大友

2009 年 3 月

前　言

汇编语言是一种符号化的机器语言。一方面由于计算机指令与具体的硬件密切相关，其所实现的功能是非常基本、具体的，因而其数目繁多，使初学者在初次接触时难以把指令记准、记全，由此望而却步；另一方面，也正是由于汇编指令的这种功能基本、具体和密切相关的特点，使它在操纵硬件资源方面显得简单而直接。与高级程序设计语言相比，用汇编语言编写的程序具有代码短小、执行高效等优点，深得一些程序员喜爱。

汇编语言和 CPU 密切相关。本书以 80x86 系列微型计算机为特定对象，系统地介绍了汇编语言程序设计的基本知识、基本原理和程序设计技术。从程序员角度简单介绍 80x86 系统结构，然后介绍 80x86 的常用基本指令，在此基础上再讲述汇编语言中的结构化程序设计方法和模块化程序设计方法。全书共分 10 章，内容大致可分成以下几个部分。

第一部分是关于数据表示方面的知识，内容主要集中在第 2 章。在介绍常用数制的基础上，主要讲述数值类型——整数和浮点数的编码方法；非数值类型——西文字符和汉字的编码方法，主要目的是使读者清楚，计算机是一个二值处理系统，任何信息都必须用二进制编码，才能为计算机所识别和处理。此外，本章也介绍了二进制的算术运算和逻辑运算方面的内容。若读者已具有这方面知识，可直接跳过本章。

第二部分是关于计算机硬件及计算机指令方面相关的知识，内容主要集中在第 3 章和第 4 章。第 3 章首先介绍 80x86 系统的基本组成结构，主要讲述 CPU 的功能，以便读者能够更好地理解指令的功能。其次介绍与编程相关的计算机资源：CPU 内部寄存器——用以临时存放指令执行所产生的中间结果存储单元；内存储器——程序中的指令和处理的数据主要存放在此存储空间；I/O 地址空间——计算机与外设通过此接口交换数据。第 4 章通过介绍机器指令的格式方式，引入了指令助记符等概念，然后主要以指令助记符和直接地址编号来表示内存单元的形式，介绍 80x86 的一些常用指令及相关内容，包括：操作数寻址方式是指令在执行期间以什么方式取得所需要的数据，如何确定数据存放的位置；数据传送类指令；算术运算和逻辑运算等运算类指令；串处理类指令；控制转移类指令。如果将编程比做写作，那么该部分内容相当于写作需要用到的字与词，内容虽然繁多，却是编写汇编语言程序所必需的基本要素。

第三部分是关于程序设计方面的知识，内容主要集中在第 6 章、第 7 章和第 8 章。第 6 章主要讲述用汇编语言指令构造顺序结构、分支结构和循环结构的一些“标准化”方法，以便读者能够使用汇编语言编写结构化程序。第 7 章主要讲述模块化程序的编写方法，介绍子程序的定义与调用的基本方法、调用子程序时的参数传递方法、局部变量的概念及在子程序中临时分配局部变量的一般方法、多模块程序的编译(汇编)连接方法等内容，为读者编写大规模程序提供必要的基础。第 8 章则简要讲述如何用汇编指令实现与外部 I/O 设备交换数据，讲述写作中遣词造句和结构安排的方法，这部分内容是汇编语言程序员必须具备的基础知识和基本技能。

需要说明的是，在第 8 章中特别介绍了计算机中断的基本概念和中断服务程序设计的一般方法。

第四部分是第 9 章内容，是关于 80x86 指令扩展与延伸方面的，包括讲述浮点运算的程序设计方法和 SIMD 指令集的简单介绍和程序示例。

余下部分主要是关于 MASM 等方面的，涉及的章节有第 1 章、第 5 章和第 10 章。第 1 章简单介绍汇编、汇编程序和汇编语言的基本概念；符号编程所涉及的指令助记符和变量、标号、子程序等地址符号。第 5 章则侧重于介绍汇编语言源程序的一般结构及其用于编制源程序的各种伪指令。第 10 章则对汇编程序和调试工具的使用方法给予简要说明，本章内容可以根据需要穿插在前面各章中学习，以便在学习过程中能够编写相应的试验程序。

本书内容的组织上首先突出汇编语言的本质特点是机器语言的符号化，在此基础上再介绍其他内容，由浅入深，循序渐进，使读者理解和掌握那些具有普遍意义的指令和关键概念。本书中所有的例子程序都已经使用 MASM 6.15 调试通过，其源程序尽可能地完整，以便读者能够上机调试。

本书的另一个特点是将主要精力放在介绍汇编语言程序设计的基本原理和基本方法上，淡化非程序设计方面的因素。例如，书中尽可能地不过分强调以“段地址:偏移地址”形式表示的逻辑地址与实际物理地址的对应关系，尽可能地少用系统的功能调用。

作为 80x86 汇编语言程序的编程环境既可以是 16 位地址模式的 DOS 环境，也可以是 32 位地址模式的 Windows 环境，仅仅是将它们用做编程环境平台，并不过多涉及其系统功能调用和环境细节。为此本书只在附录中列出极少量 DOS 环境下的系统功能调用和 Win32 环境的 API，并给予简短的说明，以方便读者编程练习时使用。但是由于 Win32 环境下对系统资源有很多保护机制，不利于讲解汇编语言的完整性方面内容，尤其是讲解计算机中断。书中的例子还是以 DOS 环境下的居多。

本书编写分工为：黑龙江科技学院朱振玉编写第 1 章和第 2 章，同济大学张光长编写第 3 章～第 5 章、第 7 章、第 9 章、第 10 章和附录，黑龙江科技学院李永新编写第 6 章，黑龙江科技学院马甲军编写第 8 章。全书由张光长负责统稿和定稿。

由于作者学识和见识有限，书中如有疏漏之处，恳请读者不吝指教。

目　录

第1章 绪 论

1.1 汇编语言概述

从本质上讲，汇编语言符号化的机器语言，与具体的处理器等底层硬件密切相关，这给初学者学习汇编语言带来了相当大的难度。但是使用汇编语言直接利用 CPU 和 I/O 等硬件资源，用它编写的硬件驱动程序简单而直接，和高级语言相比，汇编语言程序的执行代码简短、高效，这一特性对程序员来说又具有极大的吸引力。

在程序设计的早期，程序员普遍使用汇编语言编程，但是随着计算机语言编译技术的完善及硬件性能的提高，目前已经很少有人直接使用汇编语言来开发应用系统了，而代之以高级语言。但是高级语言程序在执行时必须首先转换成一系列计算机指令，然后 CPU 才能执行它。例如，下面的 C 语言语句：

```
if(x!=0x1234) y=x-0x1234;
z=0x102;
```

在 8086 系统上可以转换成 5 条机器指令来执行：减法，等于转移，两条数据传送。假设 x, y, z 是 16 位整型变量，对应的内存单元地址为 0x59A0, 0x59A4, 0x59A8，那么这 5 条指令的机器代码：(以十六进制数表示)

```
B8  A0 59   2D 34 12    74 03   A3 A4 59    C7 06 A8 59 02 01
```

这种以二进制编码形式表示的计算机指令便是**机器指令**，可以直接用机器指令编写计算机程序，这便是**机器语言**程序。显然机器语言程序不能直观地反映程序员的编程思路，而且其代码不易阅读、难以维护。

为了改善机器指令的可读性，引入了符号编程，将难以记忆的二进制编码进行符号化：用一些能反映机器指令功能的单词或单词缩写来代表该机器指令，这便是**指令助记符**；用变量名等表示存放数据的内存单元地址，用标号等表示指令在内存中的位置，这便是**符号地址**；并且将 CPU 内部的各种资源用专门符号来表示，如**寄存器名**。

例如，在 MASM 中用 SUB 表示“减法”指令，用 JNE 表示“等于转移”指令，MOV 表示“数据传送”指令；将 16 位内存单元 59A0, 59A4, 59A8 分别用变量名 V1, V2, V3 表示，将“等于转移”所转向的目标指令在内存中的位置用标号 Loc 表示；用 AX 表示 CPU 中的累加器，那么上述 5 条指令序列可表示如下：

```
      MOV    AX, V1
      SUB    AX, 1234h
      JZ     Loc
      MOV    V2, AX
Loc:  MOV    V3, 0102h
      …
```

这种符号化的指令序列在执行时必须将它们翻译成对应的机器指令序列，才能为 CPU 识

别并执行。将汇编指令序列转换成机器指令序列的过程称为**汇编**。汇编工作可以由程序员手工完成，也可以由软件(程序)处理完成，完成汇编任务的软件叫做**汇编程序**(assembler)，有时也叫做**汇编器**。

为便于翻译，汇编程序有自己的一套规则和约定，如规定代码的开始与结束，变量的定义和存储空间的分配等。这些规则和约定只是汇编程序在翻译过程中使用，故称之为**伪指令**。相比较而言，那些用助记符、符号地址等表示的指令，是在程序运行期间由 CPU 来执行的，所以称之为**汇编指令**。

汇编程序所能处理的所有汇编指令、伪指令及其表示、使用的规则便构成了**汇编语言**(assembly language)。用汇编语言编写的程序称为汇编语言程序，或称为汇编语言源程序，也简称为源程序。

综上所述，汇编语言本质上是机器语言的符号表示，汇编指令和机器指令有着直接对应关系。而高级语言程序在执行时最终都转换成机器指令。一般来讲，一条高级语言的执行语句对应着若干条汇编指令。

1.2 学习汇编语言的目的和方法

目前，高级程序设计语言已经非常成熟，种类也很多，绝大多数计算机应用系统也是使用高级语言来开发的，已经很少有人直接使用汇编语言来开发大的应用系统。因此，有人认为已经没有必要再去学那些烦琐的汇编指令了。汇编语言指令烦琐是事实，但由此说没有必要学习它却不尽然。

一方面，汇编语言是符号化的机器语言，与计算机的 CPU、I/O 等硬件关系密切，所以我们在学习和使用汇编语言的时候，能够感知计算机的运行过程和原理，从而对计算机硬件与应用程序之间的联系和交互形成清晰的认识。这使程序员编程思维更符合机器的硬件逻辑，从而形成一个软、硬兼备的编程知识体系。如果说机器语言是计算机操作的本质，那么汇编语言就是最接近本质的语言。从这一方面看，其他任何高级语言都是难以达到的。

另一方面，高级语言程序最终都是转换成机器指令而运行的。一般来说，一条高级语言的执行语句对应着若干条汇编指令，这个转换是由编译程序来完成的。在学习和使用汇编指令构造高级语言程序结构时，能够加深理解高级语言语句的执行过程，从而使我们能够有意识地编写高效、健壮的程序代码。而且在理解了高级语言语句和汇编指令的对应关系后，对我们在学习和理解编译原理等相关课程大有帮助。

再有一点就是，由于汇编语言具有能够直接有效控制硬件的能力，能够编写出运行速度快、代码量小的高效程序，所以在有些场合是不可替代的。例如，火箭和导弹等发射实时控制系统，使用汇编语言编写最合适；又如，工业控制、仪器仪表、家用电器等这类小系统中的控制系统，由于成本和体积等方面原因，使系统使用的程序空间和内存容量受到限制，因而也常常使用汇编语言编程；再如，现代的操作系统中，某些与硬件有关的功能，如机器的自检，系统的初始化，实际的输入/输出设备操作，及设备驱动程序等，仍由汇编语言编写的程序来完成。

目前，硬件设施普遍使用嵌入式方式开发，而很多嵌入式编程使用的是汇编语言。现在的数码产品很多，而这些数码产品赖以生存的芯片、主板等，都包含了嵌入式程序，这些程序中，汇编语言的使用是相当多的。

汇编语言的另一个典型应用就是现代密码学领域。现代加密算法大都建立在超大数(如1024二进制位)运算基础上，这些运算在实际运行中非常耗时，为提高程序的执行效率，使算法具有实用性，其关键部分就需要用汇编语言来实现。

那么，如何学习汇编语言程序设计呢？一般来说从以下几个方面着手。

首先，学习和掌握编程所必要的汇编指令和CPU资源等。相对高级语言来说，汇编指令是非常基本、具体的，因而数目繁多，初学者在学习中往往觉得眼花缭乱，无所适从。在学习指令的过程中，宜将指令分成几大类，逐一学习。如果其功能一时难以理解，可以先记住指令的主要功能，以后在使用中用到这个指令，再回过头来理解。当然，时间长了，先前掌握的指令可能会遗忘，这是正常的，这时只要查一查手册就可以，也就是说，学习汇编语言必须学会查阅指令手册。

其次，学习和掌握汇编语言的伪指令，先简单，再复杂，循序渐进。MASM伪指令非常繁多，掌握所有的伪指令是不可能也是没有必要的。初学者首先掌握汇编语言程序的基本框架结构，常数、变量和标号定义与分配，以及数据类型等最基本的编程要素，这样就可编写一些简单的汇编语言程序了。在此基础上，再学习其他一些必要的规则和约定及高级的编程技巧。

最后，用汇编指令构造程序的控制结构，实现结构化程序设计和模块化程序设计。汇编语言的指令和伪指令相当于英语中的单词和组词规则，但是仅仅记住英语中的单词，而不会使用它们写作和表达，不能算是掌握了英语。同样，学习汇编语言时，如果只是停留在简单地记住每条指令的功能，也不能说会使用汇编语言。所以，要学会使用和掌握汇编语言，必须能够使用汇编指令构造类似高级语言程序中的控制结构，以及模块化结构。

这本教材以80x86的指令来讲述汇编语言程序设计，但所讲述的编程方法、控制流结构和模块化思想是普遍适用的。例如，学习这本教材以后，在实际工作中可能需要使用C51汇编语言，这时只需简单地用相应的C51指令代替80x86指令即可。

1.3 汇编语言的移植性问题

汇编语言是由汇编指令和伪指令等组成，而汇编指令是直接与机器指令相对应的，所以，不同CPU的电气特性与机器指令集不同，它的汇编语言也不同。通常，大多数人所说的汇编语言指的是80x86指令集的微软宏汇编(MASM)。除此之外，常用的还用51系列指令集的汇编语言，ARM系列使用的专用RISC指令集的汇编语言等。

因为汇编语言只是机器语言的一种符号表示，所以同一类型的机器语言可以有不同的符号表示，只要有相应的翻译程序(即汇编程序)翻译即可。正因为这样，同一类型CPU也可以使用不同的汇编语言，如80x86系列CPU的计算机系统，除了微软的宏汇编MASM，还可以使用NASM或DP11等。

基于上述原因，和高级程序设计语言相比，汇编语言程序在代码可移植性方面较差。

需要指出的是，这里所说的代码可移植性指的是源程序，不是它编译成的可执行代码。实际上，同一个源程序代码在不同类型计算机上运行需要不同的翻译程序。例如，用标准C语言编写的一段程序，如果要让它在Pentium平台上运行，则必须要有相应的翻译程序将它翻译成80x86的机器代码；如果要让它在51单片机上运行，则必须用另外一个翻译程序将它

翻译成 51 CPU 的执行代码。它们的执行代码是不同的。也就是说，C 语言之所以是可移植的，是因为 C 语言有一个统一的标准。这样不同类型的计算机系统根据此标准建立各自的翻译程序，由翻译程序来解决可移植问题。

和高级语言相比，汇编语言没有一个统一标准，因而各类型的计算机系统有各自不同的汇编指令系统，及相应的汇编程序。即使用同一类型的 CPU，虽然指令系统相同，但由于不同的汇编程序可以采用不同的指令助记符，以及各自的规则和约定，所以汇编语言程序的源码一般是不可移植的。

习 题 1

1.1 简述机器语言与汇编语言、高级语言与汇编语言的关系。

1.2 通过例子简要说明什么是指令助记符、变量和标号。

1.3 简要说明汇编程序的用途。

1.4 简要说明汇编指令与伪指令的主要区别。

第2章 基础知识

本章主要介绍和汇编语言程序设计密切相关的一些基础知识，包括：

- 二进制进位记数制及其与二进制数的相互转换。
- 数值型数据的编码：无符号整数、有符号整数、浮点数和BCD码。
- 常用的编码：ASCII码、汉字国标码等。
- 二进制数的算术运算和逻辑运算。

2.1 常用数制及其相互转换

电子计算机的核心是电子电路，其最基本的逻辑电路是电子开关。一个开关只有两种状态：不是断开，就是闭合，若将其中的一种状态记做0(如断开)，另一种状态记做1(如闭合)，那么开关电路作为计数器使用。1路开关可计2个数：0, 1；2路开关可计4个数：00, 01,10, 11；依次类推，n路开关可计2^n个数。电子计算机本质上是由许许多多这样的开关电路组成的高速运转的电子装置，所以经常说计算机只认识两种符号：0和1，即只能处理二进制代码。

由于计算机只能以二进制方式动作，所以对一切数值数据及非数值数据，也只能由0和1这两种符号来表示，即必须转换成二进制编码形式，计算机才能识别并处理。

在实际使用中，由于二进制代码位数较长，书写、阅读和计忆都不方便，所以常采用十六进制数或八进制数形式来表示二进制编码。

但是人类熟悉的是十进制数，并不习惯使用二进制计数，所以在使用中根据需要，计算机中的编码和数经常用十进制、二进制、十进制和八进制形式表示。

为叙述方便，本书中若没有特别标注或说明，数均为十进制计数形式。

2.1.1 十进位计数制

十进位计数制(简称十进制)使用10个数码(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)表示数，低位向高位进位的规则是“逢10进1”，或高位向低位进位的规则是“借1当10”。

十进制数通常在数值后用下标10标识，或加D(Decimal)或d来表示。

例如，十进制数365，可以写成$(365)_{10}$，或写成365D，365d。

2.1.2 二进位、八进位及十六进位计数制

1. 二进制数

二进位计数制(简称二进制)的基数为2，使用两个数码(0和1)表示数。低位向高位进位的规则是“逢2进1”，或高位向低位进位的规则是“借1当2”。

二进制数通常在数值后用下标2，或标以字母B(Binary)或b以示区别。例如，二进制数101101101，可记做$(101101101)_2$，或101101101B, 101101101b。

二进制数所表示的数值可用下列权位展开式计算：

$$d=\sum b_i \times 2^i$$

其中，b_i取 0 或 1，i 在小数点左边(整数部分)自右至左依次取：0, 1, 2, …，小数点右边(小数部分)自左至右依次是：−1, −2, −3, …。

例如，$(1011.01)_2$所表示的数为：$1\times2^3+0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}=(11.25)_{10}$。

二进制数的四则运算除了使用 2 进位规则外，其运算法则类似十进制数运算。

1) 加法运算规则

① 0 + 0=0；② 0 + 1=1；③ 1 + 0=1；④ 1+ 1=0(高位进 1)。

例 2.1 求二进制数 1101 0011 与 1001 0110 的和。

解：

		1	1	0	1	0	0	1	1	(被加数)
+		1	0	0	1	0	1	1	0	(加数)
	1	0	1	1	0	1	0	0	1	(和)

所以，$(1101\ 0011)_2+(1001\ 0110)_2=(1\ 0110\ 1000)_2$。

2) 减法运算规则

① 0 − 0=0；② 0 − 1=1(高位借 1)；③ 1 − 0=1；④ 1 − 1=0。

例 2.2 求二进制数 1101 1010 与 1010 1101 的差。

解：

	1	1	0	1	1	0	1	0	(被减数)
−	1	0	1	0	1	1	0	1	(减数)
	0	0	1	0	1	1	0	1	(差)

所以，$(1101\ 1010)_2+(1010\ 1101)_2=(0010\ 1101)_2$。

3) 乘法运算规则

① 0 × 0=0；② 0 × 1=0；③ 1 × 0=0；④ 1 × 1=1。

例 2.3 求二进制数 1101 与 1011 的积。

解：

				1	1	0	1		(乘数)
			×	1	0	1	1		(乘数)
				1	1	0	1		
			1	1	0	1			
		0	0	0	0				
+	1	1	0	1					
1	0	0	0	1	1	1	1		

所以，$(1101)_2\times(1011)_2=(1000\ 1111)_2$。

4) 除法运算规则

① 0÷1=0；② 1÷1=1。

例 2.4 求二进制数 11 1110 除以 1011 的商。

解：

							1	0	1	(商)
(除数)	1	0	1	1	1	1	1	1	0	(被除数)
				−	1	0	1	1		
					0	1	0	0	1	0
					−	1	0	1	1	
						0	1	1	1	(余数)

所以，$(11\ 1110)_2\div(1011)_2$的商为$(101)_2$，余数为$(0111)_2$。

2. 八进制数

八进位记数制(简称八进制)的基数为 8，使用 0, 1, 2, 3, 4, 5, 6, 7 共 8 个数码。低位向高位进位的规则是“逢 8 进 1”，或高位给低位借位的规则是“借 1 当 8”。

八进制数通常在数值后用下标 8，或标以字母 O(Octal)或 o 以示区别。

例如，八进制数 555 可以记作$(555)_8$，或记作 555O，555o。

字母 O 易与数字 0 混淆，应用中多采用在数值后加 Q 或 q 来表示八进制数。

八进制所表示的数值可用下列权位展开式计算：

$$d=\sum q_i \times 8^i$$

式中，q_i取 0～7，I在小数点左边(整数部分)自右至左依次是：0, 1, 2, …，小数点右边(小数部分)自左至右依次是：−1, −2, −3, …。

例如，$(13.2)_8$所表示的数值为$1\times8^1+3\times8^0+2\times8^{-1}=(11.25)_{10}$。

一个数的二进制表示形式与八进制表示形式有着这样的对应关系：小数点左边(整数部分)自右至左，每 3 个二进制位对应一个八进制位，最左的数位不够 3 位用 0 补；小数点右边(小数部分)自左至右，每 3 个二进制位对应一个八进制位，最右边的数位不够 3 位用 0 补。反之亦然。3 位二进制数码与八进制数码间的对应关系见表 2.1。

表 2.1 3 位二进制数码与八进制数码对应关系表

三位二进制数码	000	001	010	011	100	101	110	111
八进制数码	0	1	2	3	4	5	6	7

例如，在将$(1011.01)_2$转换成八进制数时，应把它理解成$(001\ 011.010)_2$，为$(13.2)_8$。又如，在把$(15.4)_8$转换成二进制数时，直接按顺序将一个八进制位展开成三位二进制位得到$(001\ 101.100)_2$，即$(1101.1)_2$。

3. 十六进制数

十六进位记数制(简称十六进制)的基数为 16，使用 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F 共 16 个数码，其中 A～F 也可用相应的小写字母 a～f，分别与 10, 11, 12, 13, 14, 15 这 6 个数值对应。低位向高位进位的规则是“逢 16 进 1”，或高位给低位借位的规则是“借 1 当 16”。

通常在十六进制数后用下标 16，或标以字母 H(Hexadecimal)或 h 来标识。

例如，十六进制数 16D 可以写成$(16D)_{16}$，或写成 16DH,16Dh。

十六进制数所表示的数值可用下列权位展开式计算：

$$d=\sum h_i \times 16^i$$

其中，h_i取 0～15，i在小数点左边(整数部分)自右至左依次是：0, 1, 2, …，在小数点右边(小数部分)自左至右依次是：−1, −2, −3, …。

例如，$(B.4)_{16}$所表示的数值为$11\times16^0+4\times16^{-1}=(11.25)_{10}$。

一个数的二进制表示形式与十六进制表示形式有着这样的对应关系：小数点左边(整数部分)自右至左，每 4 个二进制位对应一个十六进制位，最左的数位不够 4 位用 0 补；小数点右边(小数部分)自左至右，每 4 个二进制位对应一个十六进制位，最右边的数位不够 4 位用 0 补。反之亦然。4 位二进制数码与十六进制数码之间的对应关系见表 2.2。

表 2.2　四位二进制数码与十六进制数码对应关系表

四位二进制数码	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
十六进制数码	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

例如，在将$(1011.01)_2$转换成十六进制数时，应把它看成$(1011.0100)_2$，为$(B.4)_{16}$。又如，在把$(15.4)_6$转换成二进制数时，直接按顺序将一个十六进制位展开成四位二进制位得到$(0001\ 0101.0100)_2$，即$(10101.01)_2$。

2.1.3　数制间的转换

对于一个给定的数，可以用不同的进位记数制来表示，例如，可以验算，$-(21.25)_{10}$、$-(10101.01)_2$、$-(25.2)_8$、-(15.4)表示同一个数，它们除了外在表现形式不一样，数的本质是一样的。可以根据需要使用不同的数制来记数。

1. 二(八、十六)进制数转换为十进制数

对于二进制、八进制和十六进制等表示形式的数，直接用权位展开式计算即可得出其对应的十进制的值。例如：

$(10101.1011)_2=1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=21.6875$

$(1207.3)_8=1\times8^3+2\times8^2+0\times8^1+7\times8^0+3\times8^{-1}=647.375$

$(1B2E.D)_{16}=1\times16^3+11\times16^2+2\times16^1+14\times16^0+13\times16^{-1}=6958.8125$

2. 十进制数转换成二(八、十六)进制数

十进制转换成二进制、八进制和十六进制，通常要区分数的整数部分和小数部分，并按除基取余数部分和乘基取整数部分两种不同的方法来完成。

1) 十进制整数转换为二进制整数

十进制整数转换为二进制整数采用“除 2 取余，逆序排列”法。具体做法是：用 2 去除十进制整数，可以得到一个商和余数；再用 2 去除商，又得到一个商和余数，如此反复，直到商为零时为止，然后把先得到的余数作为二进制数的低位有效位，后得到的余数作为二进制数的高位有效位，依次排列起来。

例 2.5　将十进制数 105 转换成二进制表示形式。计算过程如下：

```
2 | 105   余数为 1，即 b0=1
  2 | 52   余数为 0，即 b1=0
    2 | 26   余数为 0，即 b2=0
      2 | 13   余数为 1，即 b3=1
        2 | 6   余数为 0，即 b4=0
          2 | 3   余数为 1，即 b5=1
            2 | 1   余数为 1，即 b6=1
              0   商为 0，终止
```

即转换后的二进制整数为：$(b_6b_5b_4b_3b_2b_1b_0)_2=(1101001)_2$。

2) 十进制小数转换为二进制小数

十进制小数转换成二进制小数采用“乘 2 取整，顺序排列”法。具体做法是：用 2 乘十