8086/8088
微型计算机系统
体系结构和软硬件设计

刘玉成　G. A. 吉布森　著

科学出版社

8086/8088 微型计算机系统

体系结构和软硬件设计

刘玉成 G. A. 吉布森 著
刘寿和 梁 适 等 译
阎 玉 铭 校

科 学 出 版 社
1 9 8 7

内 容 简 介

随着IBM-PC及其升级机型的推广应用，Intel 8086/8088微处理机迅速占据了16位微型计算机系统的主要市场．本书即是关于8086/8088微型计算机系统的专著，主要介绍8086/8088 16位微处理机及其支援器件．书中结合大量典型实例由浅入深地论述了8086/8088的体系结构、指令系统和以8086为基础的微型计算机系统的软、硬件设计，重点介绍了模块化程序设计、I/O接口、多处理机结构以及实用的设计技术．最后介绍了微型计算机领域的最新发展．各章末尾附有大量习题，以帮助读者复习重点概念．

本书可作为大专院校有关专业及培训班的教科书或教学参考书，亦可供从事微型计算机设计、生产、应用和维护的技术人员阅读．

Yu-Cheng Liu Glenn A.Gibson

MICROCOMPUTER SYSTEMS: THE 8086/8088 FAMILY

Architecture,Programming,and Design

Prentice-Hall, 1984

8086/8088 微型计算机系统

体系结构和软硬件设计

刘玉成 G. A. 吉布森 著

刘寿和 梁 适 等 译

阎玉铭 校

责任编辑 孙月湘

科学出版社出版

北京朝阳门内大街137号

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

1987年4月 第 一 版 开本：787×1092

1987年4月第一次印刷 印张：29 1/4

印数：0001—6,000 字数：673,000

统一书号：15031·793

本社书号：4947·15—8

定价：6.85元

译 者 序

七十年代末至八十年代初，第三代十六位微处理机8086，Z 8000 和 MC68000 先后问世，称它们为新一代微处理机的三巨头．三者中8086的性能最差，但由于其最早投放市场，造成“先入为主”的局面，加上它有大量外围支援器件和丰富的软件，因而在十六位微型计算机市场上一直占据优势．

八十年代初期，IBM 公司推出以与8086相兼容的 8088 为 CPU 的 IBM-PC 个人计算机．接着，各种IBM-PC及其兼容机、升级机种（如 PC/XT 等）纷纷涌现，倾销于市．使8086/8088 微处理机“抬高身价”，市场上更加畅销．在国内，近二、三年间以8086/8088为 CPU 的各种型号的微型计算机系统猛增至数万台．为了满足这种形势的要求，我们翻译了本书，裨供广大用户和读者参考．

本书是一部关于8086/8088微处理机的专著，主要介绍8086/8088微处理机的结构、原理及其支援器件．书中结合大量典型示例，深入浅出地介绍了8086/8088的体系结构、指令系统以及软件、硬件设计技术，重点介绍了模块化程序设计、I/O 接口和多处理机结构．

本书由刘寿和翻译第一至第四章；梁适翻译第五至第七章；冯大本翻译第八至第十章；夏川江翻译第十一、十二章．全书由阎玉铭同志校订、整理．由于译校者水平有限，加之时间仓促，书中难免有欠妥之处，希读者不吝指正．

译 者

原　序

本书宜作为大学高年级计算机课程的教科书，供一学期教学用；也可供从事实际工作的工程师和技术人员作为参考书．读者至少应熟悉一种高级语言（如FORTRAN，BASIC或PASCAL），并具有逻辑电路的基础知识．第一章简单介绍了一般计算机的概念；比较熟悉计算机的读者可以有选择地浏览．每章末的大量习题突出了各章的重点．这些习题均按这一章介绍的内容排列，以帮助读者检验对这一章所介绍内容的理解程度．有些习题可以增长读者在某些方面的知识．

本书旨在深入探讨微型计算机系统中的硬件和软件．尽管所介绍的概念是很基本的，但都是结合具体的微处理机——Intel 8086/8088及其配套支持器件和软件展开讨论的．软件方面的内容重点放在Intel公司的ASM-86汇编程序上；但并不逐一讨论该汇编程序的所有细节，因为我们认为包罗万象会冲淡讨论的重点．

由于本书涉及的内容已超出了把8位微处理机作为简单控制器的应用范围，所以选择16位微处理机作为讨论的对象．选择较流行的16位单片处理机8086/8088的原因在于，它提供了功能很强的指令系统（其中包括有扩充了的算术运算和字符串处理指令），体系结构具有若干先进的性能，能支持多道程序设计和多道处理．此外，Intel公司还研制了各种配套器件，例如8087数字数据处理机和8089 I/O处理机，它们可使系统协调地具有某些重要的多道处理功能．通过剖析8086微处理机，可以系统地讨论多道程序设计和多道处理技术．在有关8位微处理机的书中未作过专门研究，而本书则进行了深入讨论的其他重要课题还包括与系统软件、字符串处理功能和模块化程序设计有关的问题．随着应用复杂性的增加，研究这些课题显得更为重要．

第一章概述了计算机系统和微处理机的基础知识．第二章介绍了8086微处理机的内部体系结构和机器语言指令．虽然8086和8088执行同样的指令系统，但其内部结构却稍有差异，第二章的最后一节介绍了这些差异．第三章讨论汇编语言程序设计基础．这一章首先介绍了汇编语言的概念，然后分析了8086的基本指令，其中包括算术、逻辑和控制传递指令．以丰富的实例帮助读者加深对各类指令用途的理解，同时还讨论了单模块程序所需的伪指令，说明了汇编的过程．第四章把有关汇编语言程序设计的内容引入多模块程序结构，并介绍了先进的程序设计技术．第四章还专辟程序设计一节，讨论了正确地设计多模块程序的开发过程．

第五章讨论8086的字符串处理功能，还介绍了编写有效I/O例程所需的各种代码转换方法．第六章介绍I/O程序设计技术，包括可编程I/O、中断驱动的I/O和直接存贮器存取．举例阐明了中断过程和中断向量的用法．初步介绍了缓冲技术，以使读者能理解实时操作系统如何处理I/O操作．第七章介绍多道程序设计的基本知识．虽然在本书中详细讨论多道程序设计是不切实际的，但本章还是扼要介绍了多道程序设计中比较重要的问题，阐明了多道程序设计与微型计算机系统设计的关系．讨论的内容包括数据共享、进程

同步、重入码以及存贮器管理等.

本书其余各章着重探讨微型计算机系统中的硬件问题. 第八章介绍系统总线和微处理机之间的接口、中断管理以及总线的活动和时序. 第九章讨论 I/O 和 海 量 存贮设备与系统总线之间的接口，介绍了串行和并行接口、直接存贮器存取控制器、定时器和海量存贮器控制器. 第十章讨论存贮器子系统. 第十一章分析与 Intel 器件兼容的多处理机配置. 这一章介绍了8087数字数据处理机和8089 I/O 处理机. 第十二章扼要介绍了 Intel 公司更先进的超大规模集成器件，简述了80130，80186和80286. 附录中列出了8086/8088的指令系统.

刘玉成

G.A.吉布森

目　录

第一章 引 论

微电子设备的用途分为两大类：控制与数据处理。微电子设备在简单控制器中的应用称为低档应用，而在很复杂的控制器（例如机器人、航空电子设备和先进的军事设备）以及通用数据处理中的应用称为高档应用。用复杂的电路来实现低档应用是不经济的，所以并不是越复杂越好。尽管低档微电子产品系列仍在进行种种改进，但新技术发展的推动力是在高档应用中。高档应用中有着意想不到的新用途等待我们去发现去开辟。在高档应用中，对缩小电路尺寸和提高复杂性的要求是永无止境的。

微电子器件根据其复杂程度分为如下几类：

小规模集成电路（SSI）——少于十个门电路。

中规模集成电路（MSI）——十到一百个门电路。

大规模集成电路（LSI）——一百到一、二千个门电路。

超大规模集成电路（VLSI）——几千个门电路。

当前微电子技术的尖端领域是在 VLSI 方面，工艺的核心集中在单片16位处理机及其配套器件上。目前，在16位微处理机领域中的三个领先产品是 Zilog 公司的 Z 8000、Motorola 公司的 M68000 和 Intel 公司的 8086。本书主要介绍Intel8086 及其配套器件，但大部分内容也适用于其他 16 位微处理机。

本章旨在介绍几个基本定义和其他一些基础知识，其中许多内容是复习性的。第 1-1 节概要介绍了计算机系统。第 1-2—1-4 节讨论计算机内数据的存贮方法、计算机内信息的存取方法和计算机执行程序时的一般操作。第 1-5 节分析基本数字系统的设计标准，并通过 Intel 公司微处理机系列的发展历程说明微处理机的演变过程。

1-1 微型计算机系统概述

微型计算机系统与任何其他计算机系统一样，包括硬件和软件这两个主要组成部分。当然，硬件是电路、机壳等等，而软件是指挥计算机执行任务的一组程序。

1-1-1 硬件

计算机的体系结构是其主要部件的总体布局、部件的主要性能以及这些部件相互间的连接方式。图 1-1 是典型微型计算机系统的总体结构。图中所示的部件有：中央处理部件（CPU）、定时电路、存贮器、输入/输出子系统、总线控制逻辑和系统总线。在微型计算机中，CPU 是一种微处理机，常称为微处理部件(MPU)。CPU 的作用是对指令进行译码并根据指令要求来控制系统内的活动。CPU 还完成全部算术和逻辑运算。定时电路（即时钟）产生一串或几串间距相等的脉冲，以便使微处理机内的活动与总线控制逻辑同步。定时电路输出的脉冲串通常有相同的频率，但时间上存在偏差，即相位不同。微处理机需要用

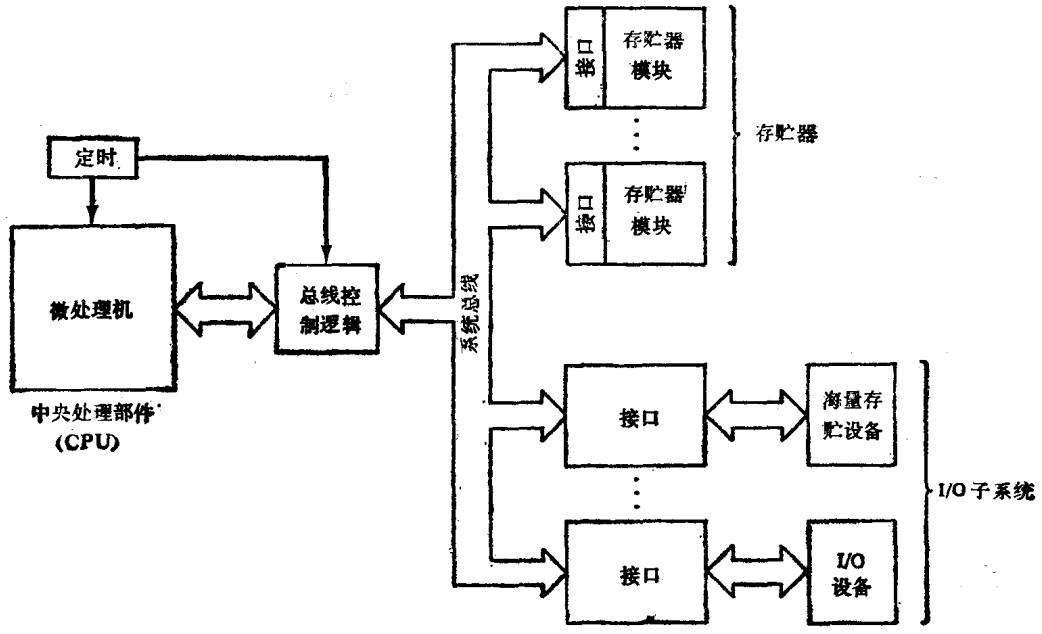


图 1-1 典型微处理机的体系结构

一相到四相脉冲，早期的处理机要求用一相以上的脉冲．在许多最新的微处理机中，除了振荡器以外，定时电路都与处理电路做在同一块集成电路（IC）上．

存贮器用来存贮当前正在使用的数据和指令．存贮器通常分为几个模块，每个模块有几千个单元．每个存贮单元可存入数据或指令的一部分或全部，而且都有一个称为存贮器地址（或简称为地址）的标识符与之相对应．CPU 工作时连续地从存贮器中取出指令，并执行指令所规定的任务．

输入/输出子系统包括与外界通信和存贮大量信息用的各种设备．卡片输入机、读带机和数/模转换器则属于输出设备．有些设备，例如终端，既提供输入能力又提供输出能力．永久性地存贮程序和数据的计算机部件称为海量存贮器．磁带和磁盘是比较常见的海量存贮设备，但最新的技术已采用磁泡存贮器（MBM）和电荷耦合器件(CCD)．虽然海量存贮器可用来存贮程序和数据，但是程序在执行之前必须传送给内存贮器．

系统总线是用来连接 CPU 及存贮器和 I/O 设备的一组导线，这些导线可以是电缆，也可以是印刷板上的连线．所有的信息都通过总线传送．信息在总线上的具体传送方法取决于总线的技术规范．在一般情况下，构成总线的导线分为三类：

1．传送信息的数据线；

2．指示欲传送信息的来源或归宿的地址线；

3．管理总线上的活动的控制线．

总线上的信号必须与连在总线上的各个部件所产生的信号协调．将总线连至某个设备的电路称为接口，而总线控制逻辑则是总线与 CPU 的接口．为了便于接口和总线控制逻辑的设计，大多数厂商都提供有各种各样的集成电路器件．根据系统复杂程度的不同，总

线控制逻辑可以部分或全部地做在 CPU 芯片上.

存贮器接口主要由地址译码逻辑、数据缓冲逻辑和执行存贮器读写操作的电路组成.地址译码逻辑对正在访问的存贮单元的地址进行译码，数据缓冲逻辑对来往于总线上的数据进行缓冲. I/O 接口既可能相当简单，也可能十分复杂. 所有的 I/O 接口必须能对送入总线和来自总线的数据进行缓冲，能接收来自 CPU 的命令，将有关设备的状态信息传送给 CPU. 此外，连接海量存贮器的接口必须能与存贮器直接通信，因此要求这些接口具有控制系统总线的能力. I/O 接口与数据总线之间的通信利用称为 I/O 口的寄存器来完成.

1-1-2 软件

计算机软件通常分为两大类：系统软件和用户软件. 系统软件是生成、准备和执行其他程序所需的一组程序. 用户软件是系统的各种用户用计算机解题时所编制的那些程序.

究竟要有多少系统软件才能满足特定计算机系统的需要，这取决于具体的用途. 如果一台计算机始终用于完成某个任务（例如用于执行重复性任务的控制器），那么只需要执行一个程序. 相反，用于通用数据处理的计算机可能要求各种各样的系统程序. 本书在大部分情况下假设讨论对象是通用计算机. 在这种通用的计算机中，程序通常（但并非总是如此）构成有明确定义的分级结构，如图 1-2 所示.

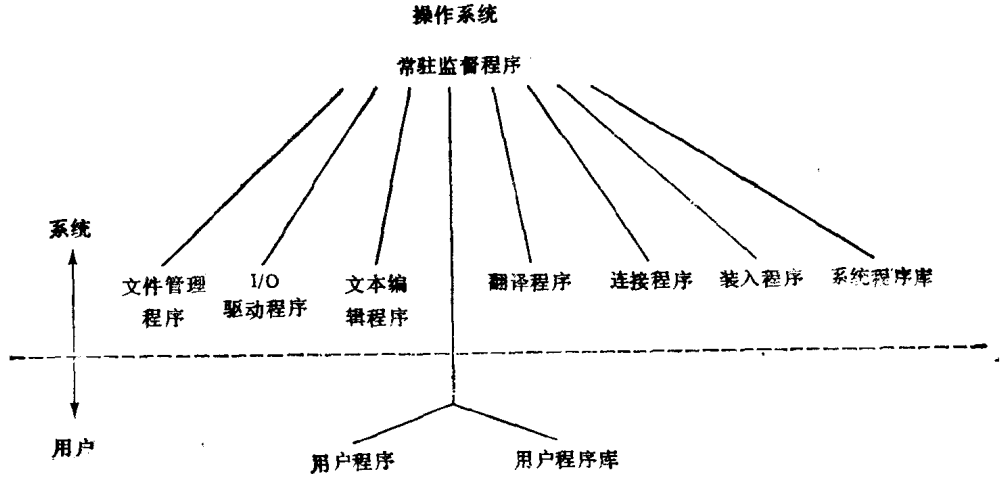


图 1-2 软件的分级结构

操作系统是一套系统程序，用于提供人机接口和充分发挥计算机的效率. “操作系统”这一术语尚无精确的定义，特别是操作系统中究竟包括哪些系统程序，在不同的参考手册和书籍中是不同的. 操作系统中最为重要的部分是常驻监督程序.计算机开机后,常驻监督程序始终保存在计算机的存贮器中. 常驻监督程序必须能接收用户命令，并启动操作系统执行相应动作.

操作系统还包括 I/O 驱动程序和文件管理例程. 前者用于执行 I/O 操作；后者用于管理存在海量存贮器中的大量数据集合. 每当用户程序或其他系统程序需要使用 I/O设备时，通常并不是由该程序执行操作，而是要求操作系统利用 I/O 驱动程序来执行任务. 这样能使操作系统更好地控制计算机，而且可使用户程序不再包括 I/O 子程序.文件管理

例程与海量存贮器 I/O 驱动程序配合使用，用于文件的存取、复制和其他处理.

此外，系统软件还可包括各种高级语言翻译程序、汇编程序、文本编辑程序以及辅助编写其他程序的程序．程序设计分为三级：

1．机器语言程序设计；

2．汇编语言程序设计；

3．高级语言程序设计.

机器语言程序是计算机能理解和直接执行的程序．汇编语言指令与机器语言指令基本上一一对应，不过前者是用字符串书写的，因此更易于理解．高级语言指令十分接近英语，其结构也就很自然地安排得适应于程序员的思路．归根结底，汇编语言程序或高级语言程序都必须用翻译程序转换为机器语言程序．若所翻译的程序是汇编语言程序，则相应的翻译程序称为汇编程序；若所翻译的程序是高级语言程序，则所用的翻译程序称为编译程序或解释程序.

文本编辑程序是输入或修改要存入或已存入海量存贮器的文本（字母、数字和标点符号等）用的程序．文本可构成汇编语言程序或高级语言程序、一组数据或一组报表（本书的原著就是用文本编辑程序写成的）.虽然文本编辑程序有好几种用途，但我们对它感兴趣的是文本编辑程序可用来生成程序.

在编写供执行的程序时还需要另外两种系统程序，通常用于不同程序要执行相同任务的情况．大多数操作系统都可以创造一组子程序，称为程序库．程序库中的子程序可附在任何系统程序或用户程序上．一般都有一个通用的系统程序库，用户还可以建立自己的程序库．把待执行的程序与程序库及其他已翻译好的程序连接起来所用的准备程序称为连接程序或连接编辑程序．另一种准备程序称为装入程序，用来把待执行的程序送入存贮器．有时，连接与装入功能合成为一个程序．第三章将详细讨论生成和执行程序的整个过程.

1-2 数据的表示法

为使计算机更加可靠，易于制造，计算机均用只能处于两个状态的器件构成，一个状态用“0”表示，另一个状态用“1”表示．将多个 0 和 1 组合在一起，便可表示任意多个不同的数．只有一个“0”或一个“1”的组合称为二进制的一位．一般来说，用 n 位可区分 2^n 个不同的项，且每增加一位，就使可能的组合数加倍.

计算机利用位串来表示数、字母、标点符号和其他任何有用的信息单元．数将根据一定的格式与位组合状态对应起来．三种基本的数的格式是：

1．二进制数（或整数）；

2．浮点数（或实数）；

3．二-十进制编码数（BCD 数或十进制数）.

（整数和浮点数格式对应于 FORTRAN 和其他高级语言中用的整数类型和实数类型.）

字母数字代码是使位组合格式与字母和其他字符对应起来．由于分配给十进制数字的位组合格式也包括在字母数字代码内，所以这种代码还可用来存贮和处理数（在高级语言

中，字符串用字母数字代码表示).除了第十一章讨论的浮点格式外，下面要详细讨论数的格式和字母数字代码。

1-2-1 二进制格式

“数”是可以用各种规则和形式表示的抽象实体．表示非负整数时通常要选择一个基数 x 和 x 个不同的符号（称为数字），然后用一串数字来表示一个数．其规则是

$$a_n a_{n-1} \cdots a_1 a_0$$

表示的数是

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

例如，若基数为10，则 65,308 表示

$$6 \times 10^4 + 5 \times 10^3 + 3 \times 10^2 + 0 \times 10 + 8$$

虽然日常生活中常用 10 作为基数，但是任何大于 1 的整数都可以作为基数．由于计算机是由二态器件构成的，故用基数 2．在这种数制下，10110 表示

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2 + 0$$

基数 8 和 16 在使用计算机时也经常会遇到．与基数 2,8,10 和 16 对应的数制系统分别称为二进制、八进制、十进制和十六进制．这些数制中表示数字的符号示于图 1-3 中．如果使用哪种数制从上下文不易分辨时，应该在数后加上表示其基数的下标.例如，1110_2 是十四，1110_{10} 是一千一百一十．

二进制	八进制	十进制	十六进制
0	0 (000)	0 (0000)	0 (0000)
1	1 (001)	1 (0001)	1 (0001)
	2 (010)	2 (0010)	2 (0010)
	3 (011)	3 (0011)	3 (0011)
	4 (100)	4 (0100)	4 (0100)
	5 (101)	5 (0101)	5 (0101)
	6 (110)	6 (011)	6 (0110)
	7 (111)	7 (0111)	7 (0111)
		8 (1000)	8 (1000)
		9 (1001)	9 (1001)
			A (1010)
			B (1011)
			C (1100)
			D (1101)
			E (1110)
			F (1111)

注：括号内为等效的二进制数.

图 1-3 重要的数制系统

将基数为 x 的数转换为基数为 10 的数的过程就是根据已知的 a_i 求下式中的 d_i

$$a_n x^n + \cdots + a_1 x + a_0 = d_m \times 10^m + \cdots + d_1 \times 10 + d_0$$

这是很容易的，只要将 x 和 a_i 用十进制表示，然后执行必要的十进制运算即可．例如

$$\begin{aligned} 10111011_2 &= 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2 + 1 \\ &= 128 + 0 + 32 + 16 + 8 + 0 + 2 + 1 \\ &= 187_{10} \end{aligned}$$

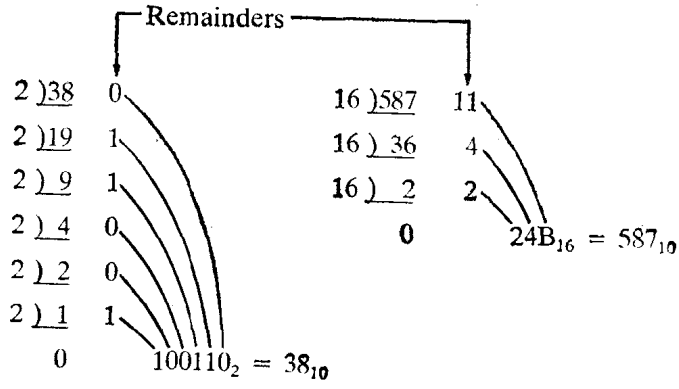
又如

$$51A_{16}=5\times16^2+1\times16+10=1306_{10}$$

这些转换也可用霍纳规则完成．该规则是

$$a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0=((\cdots(a_nx+a_{n-1})x\cdots)x+a_1)x+a_0$$

运用霍纳规则并连续除以 x，就能把十进制数转换成 x 进制数，如下例所示：



将二进制数转换为十六进制数，只要将二进制数字按 4 位一组分组，并将每组转换成相应的十六进制数即可，例如

$$\underbrace{0110}_{6}\quad\underbrace{1011}_{B}\quad\underbrace{0111}_{7}$$

反过来，要把十六进制数转换为二进制数，只要把每个十六进制数字转换成相应的二进制数即可，例如

$$\underbrace{A}_{1010}\quad\underbrace{1}_{0001}\quad\underbrace{9}_{1001}$$

二进制和八进制之间的转换也与此类似，只是按三位分组而不是按四位分组．

虽然计算机只能用二进制数工作，但在计算机的手册和书籍中，常利用二进制/八进制或二进制/十六进制的分组关系以八进制或十六进制来表示数．其原因是，这样可使二进制数的书写紧凑得多．例如，16位二进制数 1011010100111010 可写成十六进制数 B53A．本书和 8086 手册就是利用十六进制数来表示二进制数的．

任何数制系统都可进行算术运算，其算法与十进制中所用的算法相同．但是由于人们不熟悉八进制数和十六进制数的加法和乘法表，故难以对这些数直接进行算术运算．可以把操作数转换为十进制数，然后进行运算，最后再把运算结果转换成原来所用的数制的数．图 1-4 给出了二进制算术运算的实例．

负整数可通过增加一位符号位来表示．在这种情况下，符号位为0一般表示正数，为1表示负数．这样的格式称为符号-数值格式．但用符号-数值格式表示负整数的场合极为罕见．计算机几乎总是使用二进制补码格式来表示负数．它把负数表示为

$$-b=2^n-b$$

式中 b 是整数的量值，n 是表示此量值的位数．若 $n=16$，且