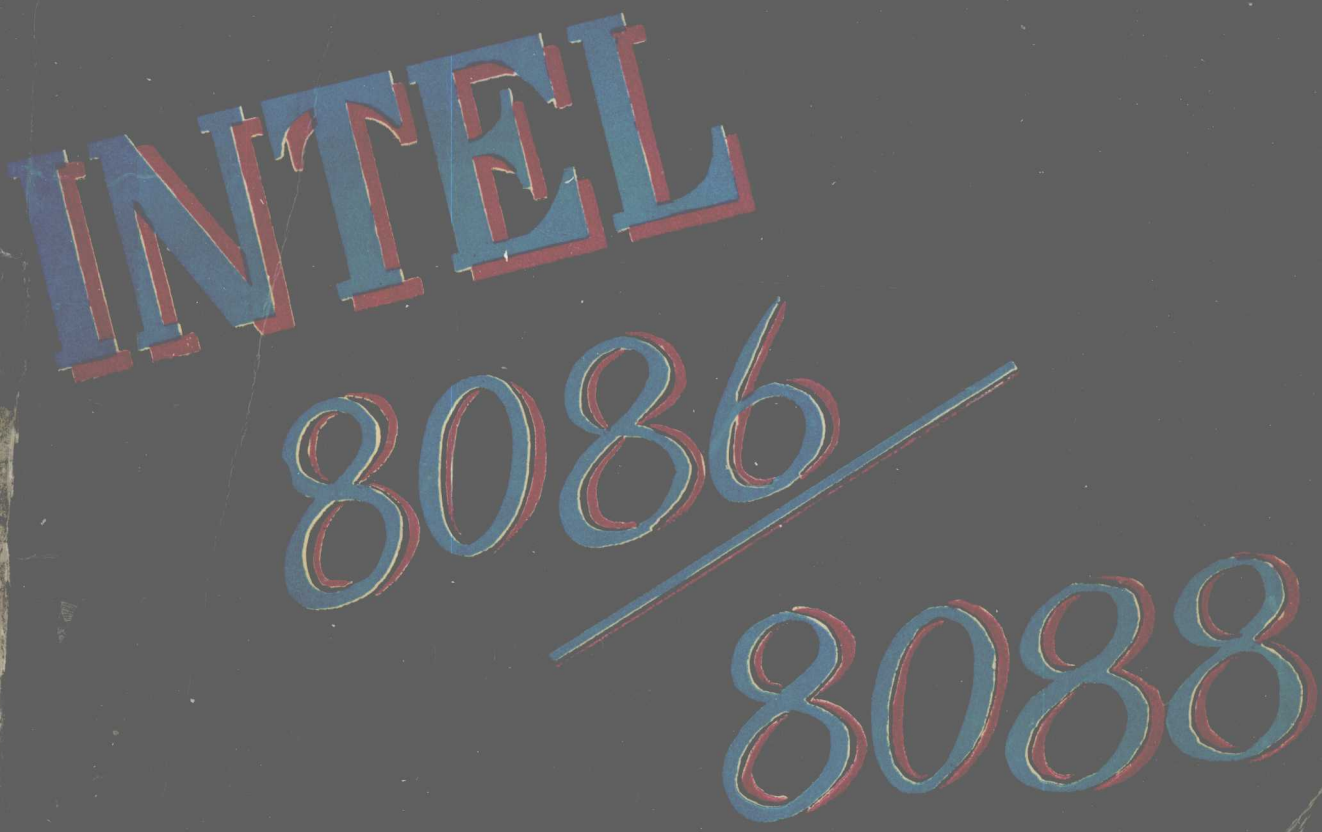
INTEL 8086/8088 系列
微型计算机原理

舒贞权　任伟利　编



西安交通大学出版社

Intel 8086/8088 系列

微型计算机原理

舒贞权　任伟利　编

西安交通大学出版社

内 容 简 介

本书是为工科电子类专业学生编写的教材，内容均结合 Intel 8086/8088 及其配套的支持器和软件展开讨论。全书共分十章，主要介绍微机系统的基础知识；8086/8088 的内部结构；基本配置和总线时序；指令系统的寻址方式；Intel ASM-86 汇编语言程序设计方法；半导体存储器特别是动态存储器的特点和刷新控制方法与接口设计；输入输出程序设计技术；总线标准；Intel 系列各种 I/O 支持器件及其与系统总线的接口和应用；多处理系统以及更先进的 16 位微处理器 80286。

本书除作为本科生教材外，还可作为其它相近专业工程技术人员的参考书。

(陕)新登字 007 号

Intel 8086/8088 系列
微型计算机原理
舒贞权　任伟利　编
责任编辑　赵丽平

西安交通大学出版社出版
(邮政编码　710049)
西安向阳印刷厂印装
陕西新华书店经销
*
开本 787×1092　1/16　印张 27.375　字数：676 千字
1993 年 12 月第 1 版　　1993 年 12 月第 1 次印刷
印数：1—5000
ISBN7-5605-0590-2/TP·69　定价：17.90 元

前　言

本教材是为工科电子类工程专业学生编写的计算机课程教材。本教材旨在较深入地探讨高性能的16位微型计算机系统的硬件和软件，讨论其基本概念和基本原理，全书的内容都是结合具体的微处理器——Intel 8086/8088及其配套的支持器件和软件展开讨论的。

本教材选择Intel 8086/8088系列微型机作为讨论对象的原因是8086/8088 CPU具有功能很强的指令系统和先进的体系结构，能支持多道程序设计和便于组成多处理器系统，是目前世界上最流行的一类16位微型计算机。世界上最大的计算机公司——美国国际商用机器公司(International Business Machine Corpoation，简称IBM公司)于1981年推出的IBM PC微型计算机是以Intel 8088 CPU为核心设计的，随后各种IBM PC兼容机、升级机种(如PC/XT机、PC/AT机)纷纷涌现，倾销于市场，成为销售量最大的一类机型。我国也已自行开发了一批以Intel 8086/8088 CPU为基础的微型计算机，Intel 8086/8088系列微型机已在各个部门得到非常广泛的应用。

本教材在软件方面以介绍8086/8088指令系统为基础，重点讨论Intel公司的ASM-86汇编语言程序设计；硬件着重讨论8086/8088的体系结构、接口技术、Intel公司的的I/O配套支持器件及应用，为学生开发应用8086/8088系列微型计算机包括IBM-PC机打下必要的基础。

本教材共分十章。第一章概述了微型计算机系统的基础知识。第二章介绍8086/8088的内部结构，着重讨论8086/8088系统的基本配置和总线时序。第三章讨论8086/8088指令系统的寻址方式，分析8086/8088的基本指令和使用方法。第四章首先介绍进行汇编语言程序设计所需要的伪指令，简要地介绍汇编过程，然后着重讨论ASM-86汇编语言程序结构和程序设计方法，这一章还介绍了汇编语言程序和PC-DOS的接口以及BIOS中断、系统功能调用的使用方法。第五章讨论半导体存储器，重点讨论动态存储器的特点和刷新控制方法与接口设计。第六章讨论输入/输出程序设计技术，包括程序控制I/O、中断驱动的I/O和DMA，着重介绍8086/8088的中断系统和中断优先级管理。第七章简要地介绍总线技术与标准，包括PC总线、多总线、GPIB及EIA RS-232C标准。第八章较深入地介绍Intel系列的各种I/O支持器件及其与系统总线的接口和应用，包括定时器、并行接口、串行接口、DMAC等，模拟通道(A/D与D/A)的接口也在这一章介绍。第九章简要介绍基于8086/8088的多处理系统。最后一章介绍Intel公司更先进的16位微处理器80286，着重讨论80286的存储器管理与保护及其中断系统。

本教材在选材上尽量注意系统性、完整性和先进性，并尽可能做到深入浅出，循序渐进，便于自学。为适应不同的教学时数，本书前八章中部分非最基本内容，使用小号字体排印。除作为本科生教材外，本书还可作为研究生参考教材或工程技术人员的自学参考书。

西安冶金学院计算机教学研究室刘家　教授对本书进行了全面详细的审核，提出了许多宝贵意见，在此谨表示衷心感谢。

由于编者水平有限，加之时间仓促，教材中必存在不少缺点和错误，敬请读者批评与指正。

编　者

目　录

第六章　输入与输出

第七章　总线标准

第八章　输入与输出接口

第九章　基于 8086/8088 的多处理器系统

第十章　Intel 80286 微处理器

附　录

第一章 绪 论

§1-1 微型计算机发展过程简介

微型计算机是电子计算机技术和大规模(以及超大规模)集成电路工艺技术的结晶,它的出现和发展是和大规模集成电路工艺技术的迅速发展分不开的。

电子计算机是一种自动化的高速计算机器。第一台电子计算机 ENIAC 是 1946 年在美国问世的,它使用了 18000 只电子管,重 30 吨,占用 30 多米长的机房,耗电 100kW,运算速度为每秒 5000 次加法运算。经过几十年的努力,至今已经经历了几代更新,目前发展到以大规模集成电路为主要特征的第四代计算机,运算速度在每秒 10 亿次以上的巨型机已投入运行。

电子计算机在结构上是由运算器、控制器、存储器和输入及输出设备组成,其中的运算器和控制器是核心部分,通常把它们称为中央处理器(Central Processing Unit),其缩写符号为 CPU,随着大规模集成电路工艺技术的发展,到了 70 年代初期,已经能够把原来体积很大的中央处理器集成在一片或几片大规模集成电路芯片上,称为微处理器(简称 μP 或 MPU)。以 MPU 为基础,由 MPU,存储器,输入/输出接口以及其它支持逻辑组成的计算机称为微型计算机。微处理器的出现开创了微型计算机的新时代,为计算机的发展和普及开辟了一条崭新的途径,这是计算机科学划时代的进步。

由于微型计算机具有价格便宜、体积小、重量轻、耗电少、可靠性高、对使用环境要求低,通用性和灵活性好等突出优点,使它获得极为广泛的应用。由于其本身的生命力和集成电路工艺技术的迅速发展,使微型机技术得到极其迅速的发展,从 1971 年以来,每隔 2～4 年就更新换代一次。如今微型计算机的功能已达到传统的中型机的同等水平,微型机与小型机以及中型机在性能上已不再有明显的差别。采用微型机构成的大型系统已在性能价格比和应用方面远远地超过了以往几代的大型机。微型计算机的蓬勃发展改变了计算机市场的产品结构,计算机进入了个人化时代。

在微处理器与微型计算机的发展过程中,位于美国旧金山南部硅谷的 Intel(英特尔)公司具有特别重要的作用。1971 年 Intel 公司研制成功世界上第一种微处理器 4004 及由它组成的 MCS-4 微型计算机。4004 是 4 位微处理器,采用 PMOS 工艺,在一块 0.297×0.404cm² 的硅片上集成了 2250 个晶体管,适用于计算器,尽管它在结构与性能上还很不完善,但它却标志着一类新型的电子计算机——微型计算机的诞生。1972 年 Intel 公司推出了 8 位微处理器 8008 及 MCS-8 微型计算机,8008 是第一只通用的 8 位微处理器,也由 PMOS 工艺制成。这个时期为微型计算机发展史上的第一个阶段。4004 和 8008 是这个时期的代表产品,称为第一代微处理器。第一代微处理器的特点是采用 PMOS 工艺,速度较低,指令系统简单,运算功能差,系统结构还没有超出计算器的范围。

1973 年 Intel 公司研究成功了性能更好的 8 位微处理器 8080。8080 的出现加速了微处理器与微型计算机的发展。生产微处理器的厂家剧增。这一时期,除了 Intel 公司生产的 8080 外,

具有代表性的 8 位微处理器还有 Zilog 公司生产的 Z80，Motorola 公司生产的 M6800，MOS 公司生产的 6502(它是 IBM PC 机问世之前世界上最流行的微型计算机 Apple Ⅱ 的 CPU)。这些高性能的 8 位微处理器是第二代微处理器的代表产品。第二代微处理器采用 NMOS 工艺，除集成度有了提高外，性能也有明显的改进，运算速度约提高了一个数量级，寻址方式增至 10 种以上，基本指令可达一百多条，在结构上已具有典型的计算机体系结构以及具有中断、DMA 等控制功能，考虑了机器间的兼容性，接口的标准化和通用性，使 8 位微处理器达到了成熟的阶段。

1978 年 Intel 公司推出了新一代微处理器 8086，这标志着微处理器的发展进入了第三代。随后，Zilog 公司生产了 16 位微处理器 Z8000，Motorcla 公司生产了 M68000。这些性能更高的 16 位微处理器都采用 HMOS 高密度集成半导体工艺技术。8086 在一片硅片上集成了 29000 个元件，M68000 集成了 68000 个元件。第三代微处理器指令最短执行时间在 300ns 以下，性能指标达到或超过了中档的小型机。这类微处理器都具有丰富的指令系统，采用多级中断系统，具有多种寻址方式，多种数据处理形式，在软件方面可以使用多种高级语言，有完善的操作系统，可以构成多处理器系统。

从 1981 年以后，Intel 公司推出了 32 位微处理器 iAPX432，其它公司相继宣布了不同型号的 32 位处理器，如 Motorola 公司的 M68020，这是第四代微处理器，这些微处理器采用流水线控制，具有面向高级语言的系统结构，有支持高级调度、调试以及开发系统用的专用指令，具有固件的操作系统。以 iAPX432 为例，其寻址能力为 4G 字节，虚存空间可达 64T 字节(2^{46})，最短指令执行时间为 0.125μs。由这些微处理器组成的微型计算机，其性能已达到或超过了高档的小型机。

特别需要指出的是，各个微处理器的生产厂家，都使其生产的产品形成系列，相互兼容，以 Intel 公司为例，该公司在 80 年代初宣布生产了 8 位、16 位和 32 位的 iAPX 系列。就 16 微处理器来说，有 iAPX86/88 系列，iAPX186 系列，iAPX286 系列。在 iAPX86/88 系列中，把单纯的 8086 命名为 iAPX86/10，单纯的 8088(8088 其内部结构，指令系统与 8086 是一样的，只不过其外部只有 8 位数据总线，而 8086 有 16 位数据总线)命名为 iAPX88/10；若 8086 与 8087 配接后称为 iAPX86/20，8087 为数值数据协处理器(NPX)；8086 配接输入/输出处理器 8089(IOP)之后，称为 iAPX86/21；若 8086 配接操作系统固件 80130(OSF)后，称为 iAPX86/30，这种命名法对于以 8088，80168，80286 作为 CPU 的系统也适用，因而有 iAPX286/10，iAPX286/20 等等。iAPX86/20 同 iAPX86/10 相比，其系统的数学运算能力约提高 100 倍，不仅增加了 8086 所没有的双倍字长以上的各种算术运算能力，而且还增加了 32 位、64 位、80 位的浮点运算和 18 位的 BCD 数据运算。8089 是一种专门用于处理输入/输出的处理器，它和 8086/8088 配接后，就能代替 CPU 对各种 I/O 设备的管理。从而大大改善系统的性能。至于 iAPX186 及 iAPX286 系列，则从纵向明显地改善了 8086/8088 的性能。

§1-2 数据的表示法

由于计算机内部硬件是由只能处理两个状态的器件组成，因而计算机内部的任何信息只能用“0”或“1”这两个状态来表示，将多个 0 和 1 组合在一起，便可表示任意多个不同的数，组合在一起的 1 和 0，称为位串，只有一个 0 或 1 的组合称为二进制的一位。

计算机中用位串来表示数、字母、标点符号和其它任何有用的信息。按一定格式构成的位组合状态被用来表示数，数具有三种基本格式：二进制数，浮点数，二-十进制编码数(BCD数)。与字母、数字或其它字符对应的位组合格式称为字母数字代码。下面分别讨论二进制数(整数)，浮点数(实数)、BCD数、与字母数字代码。

一、二进制格式

1. 进位计数制及其相互转换

进位计数制将数划分为不同的数位，按位进行累计，累计到一定数量(此数量称为基数)之后，又从零开始，同时向高位进一。每位数使用相同的一些数字，但由于划分了数位的等级，同一数字在不同的数位其所表征的数值是不同的。若X为基数，则每个数位有X个不同的数字，以a表示X个数字中的某个数。

则任意一个数可表示为：

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

即

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

例如，若基数为10，称为十进制数。对十进制数 75 108_{10}，表示

$$7\times10^4+5\times10^3+1\times10^2+0\times10^1+8$$

上述数字中的下标10表示基数为10，在计算机中，使用2为基数的计数制，即二进制数。这时 101101_2 表示

$$1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1$$

除了十进制数、二进制数之外，还常用八进制数与十六进制数。对十六进制数，十六个数字为0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F。

将基数为X的数(如二进制数，八进制数或十六进制数)转换成基数为10的数的过程是根据已知的 a_i 求下式中的 d_i：

$$a_n X^n + a_{n-1} X^{n-1} + \cdots + a_i X^i + \cdots + a_0 = d_m \cdot 10^m + \cdots + d_i \cdot 10^i + \cdots + d_1 \cdot 10 + d_0$$

只要将X和 a_i 用十进制表示，然后作十进制运算即可得到需要的结果。例如：

$$\begin{aligned} 101110_2 &= 1\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+0 \\ &= 32+0+8+4+2+0 \\ &= 46_{10} \end{aligned}$$

又如

$$\begin{aligned} 2A4_{16} &= 2\times16^2+10\times16^1+4 \\ &= 512+160+4 \\ &= 676_{10} \end{aligned}$$

上述转换也可以用霍纳(Horner)规则完成，该规则是：

$$a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0 = ((\cdots(a_n X + a_{n-1})X\cdots)X + a_1)X + a_0$$

当需要从十进制数转换为X进制数(如二、八、十六进制数)，则只要运用霍纳规则将被转换的十进制，连续除以X，直至商等于零为止。第一次除法的余数是 a_0，而最后一次除法的余数为 a_n，例如，将 150_{10} 转换为二进制数，X=2：

	余数
2 | 150	
2 | 75	0 …… a_0
2 | 37	1 …… a_1
2 | 18	1 …… a_2
2 | 9	0 …… a_3
2 | 4	1 …… a_4
2 | 2	0 …… a_5
2 | 1	0 …… a_6
0	0 …… a_7

∴转换结果为 $a_7a_6\cdots a_0=10010110_2$

又如将 540_{10}转换为十六进制数，X＝16：

	余数
16 | 540	
16 | 33	12(C)…… a_0
16 | 2	1 …… a_1
0	2 …… a_2

即转换 $a_2a_1a_0=21C_{16}$

将二进制数转换为十六进制数，只要从低位向高位按四位一组进行分组，然后每一组以一位十六进制数表示即可，若高位组不够四位，则应以 0 补够四位，例如：

1001	0001	1010
9	1	A

即 $100100011010_2=91A_{16}$，又如：

11	0010	1011
3	2	B

需要在最高位组的 1 1 前面添上 0 0，成为 0011，其对应的十六进制数为 3，因此，$1100101011_2=32B_{16}$。

反过来，若要将十六进制数转换为二进制数，则只要把每位十六进制数以相应的四位二进制数表示即可，例如

F	0	5
1111	0000	0101

即 $F05_{16}=111100000101_2$

尽管计算机只能识别二进制数，但由于二进制数字书写与阅读都不方便，而且，二进制数与八进制数或十六进制数间存在很简单的对应关系，因而常以八进制数或十六进制数表示二进制数。

2. 有符号数的表示法

计算机中，有符号的二进制数可由不同的编码形式表示，它们是原码、补码与反码。在微型

计算机中，进行算术运算操作（包括加、减、乘、除）的有符号数，通常以补码形式表示。下面分别讨论有符号数的几种表示法。

（1）机器数

不带符号的数是数的绝对值，在绝对值前加上表示正负的符号就成了符号数。直接用正号“+”和负号“−”来表示的二进制数叫做符号数的真值。

通常以数的最高位表示符号，若是字长为 8 位，即 D_7 为符号位，$D_6 \sim D_0$ 为数值位。符号位用“0”表示正，用“1”表示负，如：

$$X=(01010110)_2=+86$$

$$X=(11010110)_2=-86$$

把符号数值化以后，就能将它用于机器中，机器数由符号位和数值两部分组成。机器中使用的符号数称做机器数，除了符号数值化而外，机器数的特点还有字长一定，运算精度有限，以及必须规定小数点的位置等。常用的机器数有原码，补码和反码三种形式。

（2）原码

将数的真值形式中的正负号用代码 0 或 1 来表示时叫做数的原码形式，简称原码。例如，若字长为 8 位

$$X=(+85)_{10}=+1010101 \text{ 时，其原码}$$

$$[X]_{原}=01010101$$

$$X=(-85)_{10}=-1010101 \text{ 时，其原码}$$

$$[X]_{原}=11010101$$

若字长为 n 位，原码一般可表示为：

$$[X]_{原}=\begin{cases} X & 0 \leqslant X < 2^{n-1} \\ 2^{n-1}-X & -2^{n-1} < X \leqslant 0 \end{cases}$$

当 X 为正数时 $[X]_{原}$ 和 X 一样。即 $[X]_{原}=X$。

当 X 为负数时 $[X]_{原}=2^{n-1}-X$，由于 X 本身为负数，所以实际上就是将 $|X|$ 的数值部分（绝对值）前面的符号位写成“1”。

原码表示法比较直观，但它的加法运算较复杂，当两数相加时，计算机首先要判断两数的符号是否相同，如果相同则两数相加，若符号不同，则两数相减，在作减法之前，还要判断两数绝对值的大小，然后用大数减去小数。最后再确定差的符号。要实现这些操作，线路就很复杂。为把减法变成加法进行运算，就要借助于数的反码和补码表示法。

（3）补码

为了克服原码运算的缺点，广泛使用另外两种符号数的表示法，即补码和反码。使用补码和反码，可用加法来代替减法，完全消除了加法和减法的界限，这使设备大为简化，另一方面符号位也和数值部分一起参加运算，这就不再需要专门处理符号的附加设备。

在介绍补码之前，先看一个减法的结果可通过加法来实现的例子，假定现在是北京时间 6 点整，而手表指示的是 8 点整，比北京时间快了 2 小时。校准的方法有两种：一种是倒拨 2 小时，一种是正拨 10 小时，若规定倒拨是做减法，正拨是做加法，那么对于手表来讲，减 2 与加 10 是等价的。也就是说减 2 可以用加 10 来实现。这表明减法在一定条件下，是可以用加法代替的。因为对手表来说，最大计时为 12，13 点用 1 点表示，12 自动丢失。这里“12”称为“模”，10 称为“−2”对模（12）的补数。将它们用数学式表达则为：

倒拨： $8-2=6$

正拨： $8+(12-2)=8+10=6$ （模 12）

$\therefore$ $8-2=8+10$ （模 12）

式中“模”12 可以看作是向高位的进位。

从上例中可以得出这样的结论：求两个正整数 8－2 之差。可以用加上被减数(2)的补数(10)，然后舍去进位(模)来实现，补数(10)就是模(12)与减数(2)的负数之和。推广到一般则有：

$$A-B=A+(-B+M) \quad （以 M 为模）$$
$$=A+(-B)_{补}$$

上述表示法，还可用数学中的“同余”概念来描述。对两个整数 A、B，若用某一个正整数 M 去除，所得的余数是相等时，我们就说 A、B 对模 M 同余，记作 $A\equiv B(\mathrm{mod}M)$。如 $A=108$，$B=8$，若此时 $M=100$，则用 100 去除 A 和 B，得到余数都是 8，因此 108 和 8 对模 100 同余，记作 $108\equiv 8(\mathrm{mod}100)$。同样 $-8\equiv 92(\mathrm{mod}100)$。我们说，在模 100 意义下 108 与 8，－8 与 92 都是等价的。

现在我们再回过来考虑一下计算机运算的特点。计算机中的部件都是有固定的位数，假定位数为 n，则计算机中最大的计数值(包括符号位)为 2^n。因此计算机的补码是以“2^n”为模，它可用下述公式求得：

$$[X]_{补}=\begin{cases} X & 0\leqslant X<2^{n-1} \\ 2^n+X & -2^{n-1}\leqslant X\leqslant 0 \end{cases} \quad (\mathrm{mod}2^n)$$

以 2^n 为模的补码，也称为对 2 的补码。

补码有如下几个简单性质：

① 当 X 为正数时，补码和原码相同。

当 X 为负数时，负数的补码等于 $2^n+X=2^n-|X|$，

② $[+0]_{补}$和$[-0]_{补}$是相同的。所以在补码表示中，“0”的表示是唯一的。

③ 字长为 n 位的补码，可表示的数 X 的范围为：

$$-2^{n-1}\leqslant X<2^{n-1}$$

求补码的方法：

由上面的讨论可知，正数的补码就是它本身，并等于原码，只有负数才有求补的问题，这里我们介绍三种求补码的方法。

① 根据定义求

$$X=2^n+X=2^n-|X| \qquad X<0$$

即负数 X 的补码等于模 2^n 加上其真值。(或减去其真值的绝对值)。

如：$X=-1010111$ $(n=8)$

则
$$[X]_{补}=2^8+[-1010111]=2^8-|-1010111|$$
$$=100000000-1010111$$
$$=10101001 \quad (\mathrm{mod}\ \ 2^8)$$

这种方法因为要作一次减法很不方便。

② 利用原码求——符号位除外，求反加 1

计算机通常是将原码数值部分按位求反，即“1”变“0”，“0”变“1”，再在最低位加 1，来求补

码，而符号位不变。

例　$X=-1010101$　　$[X]_{原}=11010101$

则　$[X]_{补}=\boxed{1}\,0101010+1$

$=10101011$

即　$[-1010101]_{原}=\boxed{1}1\,0\,1\,0\,1\,0\,1$

⇓　求反

$\boxed{1}0\,1\,0\,1\,0\,1\,0$

$+\qquad\qquad 1$

$1\,0\,1\,0\,1\,0\,1\,1=[X]_{补}$

如果将$[X]_{补}$再求一次补，即将$[X]_{补}$除符号位以外变反加1就得到$[X]_{原}$。

如果已经知道$[X]_{补}=X_nX_{n-1}X_{n-2}\cdots\cdots X_0$那么对$[X]_{补}$的每一位（包括符号位）都按位求反，然后再加1，结果即为$[-X]_{补}$。

补码运算：

带符号数的运算的基本规定：

① 数的最高位是符号位，0代表正数，1代表负数，把符号也看成数，一同参加运算。

② 参加运算的数都用补码方式表示，正数的补码就是它自已，负数的补码表示为

$$2^n+X=2^{n-1}+(2^{n-1}-|X|)\qquad X<0$$

③ 对于补码的加减法可用下面一般公式表示：

$$[X\pm Y]_{补}=[X]_{补}+[\pm Y]_{补}$$

也即无论是加法还是减法运算，都由加法运算实现，运算结果（和或差）也以补码表示。若运算结果不产生溢出，且最高有效位为0，则表示结果为正数，若为1，则为负数。

补码溢出判别：

我们假定一个装置的位数为n，除去一个符号位外，还有n－1位表示数值，不论哪一种方法表示数，n位数所能表示的数的范围是有限的，若运算结果超出n位数所能表示的数的范围，则运算出错，这种现象称“溢出”。那么，如何判断“溢出”呢？这里主要介绍微型计算机中采用的“双高位”判别法。

所谓“双高位”判别法，就是用字长最高位及次高位的进位状态来判别溢出的方法，两数相加发生溢出，只可能有两种情况，即两数同时为正或同时为负，并且其和的绝对值又大于等于2^{n-1}时。两个正数相加，若数值部分之和大于等于2^{n-1}，则数值部分必有进位，$C_P=1$，而符号位必无进位，$C_S=0$，这种$C_S\,C_P$的状态为“01”时的溢出称为“正溢出”。同样，若两负数相加，如果数值部分无进位，$C_P=0$，而符号位有进位，$C_S=1$，这时$C_S\,C_P$的状态为“10”，这种溢出现象可称“负溢出”。当两数相加或相减时，最高位和次高位若同时有进位或同时无进位，则不产生溢出，得到正确的运算结果。因而补码运算结果是否产生溢出可由下列运算来判断：

$$V=C_P\oplus C_P$$

式中C_S表示最高位的进位状态，若有进位，$C_S=1$，无进位$C_S=0$，而C_P表示次高位的进位状态，同样，有进位$C_P=1$，无进位$C_P=0$。上式运算结果若$V=1$，则表示产生了溢出（运算结果超出了n位补码所能表示的数的范围，即结果$\geqslant 2^{n-1}$，或结果$<-2^{n-1}$），若$V=0$，则表示运算结

果正确，不产生溢出。

(4) 反码

对原码除符号位外，逐位求反，可得负数的另一种表示法，即反码表示，反码定义为：

$$[X]=\begin{cases}X & 0\leqslant X<2^{n-1}\\ (2^n-1)+X & -2^{n-1}<X\leqslant 0\end{cases}$$

反码是以(2^n-1)为模的补码，也称做对 1 的补码。

例如：

$X=(+33)_{10}$　$[X]_{原}=00100001$，　$n=8$

　　$[X]_{反}=00100001$

$X=(-33)_{10}$　$[X]_{原}=10100001$，　$n=8$

　　$[X]_{反}=11011110$

(5) 原码、补码、反码间的转换

当数 X 的真值为正数时，则有

$$[X]_{原}=[X]_{补}=[X]_{反}$$

若数 X 的真值为负，且$-2^{n-1}<X\leqslant 0$时，

$$[X]_{原}\xrightleftharpoons{\text{符号不变，其余位取反}}[X]_{反}$$

即除符号位外，其它位取反，就可从$[X]_{原}$得到$[X]_{反}$或从$[X]_{反}$得到$[X]_{原}$。

若数 X 的真值为负，且$-2^{n-1}<X<0$，则有

$$[X]_{原}\xrightleftharpoons{\text{符号不变，其它位取反，末位加 1}}[X]_{补}$$

即除符号位外，其余各位取反，然后末位加 1，就可实现原码与补码间的转换。

当$-2^{n-1}<X<0$时，$[X]_{反}$的末位加 1，就得$[X]_{补}$；若$[X]_{补}$的末位减 1，则可得$[X]_{反}$：

$$[X]_{反}\xrightleftharpoons[\text{末位减 1}]{\text{末位加 1}}[X]_{补}$$

若包括符号位在内，对所有位取反，末位再加 1，则实现$[X]_{补}$与$[-X]_{补}$之间的转换：

$$[X]_{补}\xrightleftharpoons{\text{所有位取反，末位加 1}}[-X]_{补},\ -2^{n-1}<X\leqslant 0$$

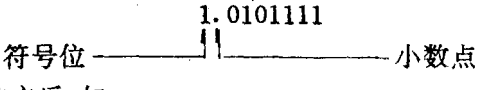
3. 定点数与浮点数

在计算机中，对实数中的小数点有两种处理方法，因而机器数也有定点数与浮点数之分。下面分别给予简要介绍。

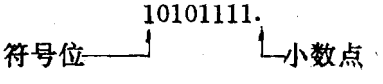
(1) 定点数

所谓定点数，是指小数点的位置固定不变。一般地说，小数点可固定在任何数位之后，但常用下列两种形式：

纯小数：小数点固定在符号位之后，如



纯整数：小数点固定在最低位之后，如



但通常采用纯小数形式。

(2) 浮点数

浮点数指小数点位置不是固定的，它随阶码而浮动。在相同字长的条件下，浮点数所表示的数的范围比定点数大的多，且在计算过程中能保持有效数字的位数。

浮点数由阶码和尾数两部分组成，对于任何一个带符号的数 N 可表示为

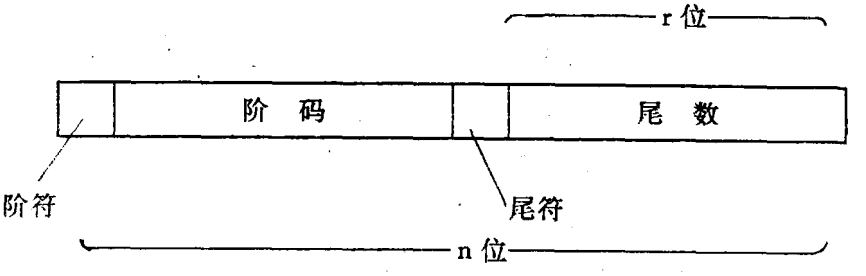
$$N=\pm S\times 2^{\pm P}$$

式中 S 称为尾数，P 称为阶码。S 为纯二进制小数，P 以二进制整数表示。尾数与阶码的符号同样以一位 0 或 1 表示，0 表示正数，1 表示为负。显然，阶的正负规定了小数点的实际位置，当阶码为正 n 阶，则实际小数点应向右移 n 位。例如 S=0.11011，阶码的符号为正，P=101，则数 N 为

$$N=0.11011\cdot 2^{5}=11011$$

即小数点的实际位置应向右移 5 位。

若以 n 位二进制数表示一个浮点数，假定尾数为 r 位(包括一位符号位)，其余 n−r 位为阶码(同样包括一位符号位)，则其格式如下：



显然，阶码位数越多，所能表示的数的范围越大，但在一定的长度(n 位)下，尾数的位数就少，因而数的有效位也就越少，通常在二者间进行折衷，这时 r 与 n 可采用以下关系：

$$r\cdot 2^{r+1}=2^{n}$$

如 n=16，则 r 应取 11，即尾数 10 位(去掉 1 位符号后)，阶码 4 位(去掉一位符号位后)。这时该浮点数能表示的最大数为

$$(0.1111111111)\times 2^{1111}=111111111100000$$
$$=+3.2737\times 10^{4}$$

其所能表示的最小数为

$$(0.0000000001)\times 2^{-1111}=2^{-11001}$$

即最小负数为 -2^{-25}。

二、 二-十进制(BCD)格式

二进制数具有运算简便，运算硬件容易实现等优点，但它表示的数值不直观，不符合人的习惯，于是在计算机的输入/输出及进行某些运算仍然采用十进制数。不过，在计算机中，十进制数必须用二进制编码来表示，最常用的编码形式是 BCD 码(即 8-4-2-1 码)，它用等值的四位二进制数表示一位十进制数字。本书所讨论的 8086/8088 CPU 能够处理两种格式的十进制数：组合的 BCD 格式和分离的 BCD 格式。

组合的 BCD 格式，也即通常所指的 BCD 码，是以四位为一组来表示十进制数字，如 7809

表示为

0111 1000 0000 1001

因此每字节可存放两位十制数。

对于分离的 BCD 格式，每个字节表示一位十进制数，每位十进制数存于 8 位字节的低 4 位，而高 4 位的内容无关紧要，不过通常还是以 0 表示，这样十进制数 7809 应表示为

0000111 00001000 00000000 00001001

对于带符号的 BCD 数也可以用补码表示，同二进制数的补码的定义类似，正数的补码与正数本身相同，但对于负数，则定义为

$$[d]_{补}=10^n+d \qquad (d<0)$$

称为对模 10^n 的补码，n 为 BCD 码位数。对于给定的位数 n，具有唯一定义的整数范围为：

$$-5\times10^{n-1}\sim5\times10^{n-1}-1$$

例如，若 n=8，则可表示的数的范围为

$$-50\ 000\ 000\sim49\ 999\ 999$$

如果采用组合 BCD 格式，则需用 32 位(即四个字节)来存储上述范围的数。

不难理解，凡是 BCD 数的最高位大于等于 5，则该数为负数，否则若小于 5，则为一个正数，例如对于 n=4，且采用组合 BCD 格式，则：

0000 0000 0000 0001 表示的数为 +1
0010 0000 1001 0101 表示的数为 +2095
0101 0000 0000 0000 表示的数为 −5000
0101 0000 0001 1000 表示的数为 −4982
1001 0111 0110 0111 表示的数为 −233

BCD 补码的运算规则同二进制数的补码运算规则相同，例如 262—495＝−233，采用补码运算则为

$$0262-495=0262+9505$$
$$=9767 \qquad (\text{最高位大于 5 为负数})$$

而 $[-233]_{补}=9767$，因而说明上述结果是正确的。

三、字符编码

计算机除了处理数字之外，还需处理字符(包括字母、数字，标点，各种算符等)，因而字符也必须按特定的规则用二进制数进行编码，称为字母数字码，常用的编码有电传码(5 位)，扩充的二-十进制交换码(EBCDIC 码)以及美国信息交换标准码(ASCII 码——American Stadard Code for Information Interchange)。字母数字码主要用于计算机与外界(I/O 设备)交换信息，在微型计算机系统中，最常用的字母数字码是 ASCII 码。ASCII 码的编码见附录 A。

ASCII 码采用 7 位编码，共有 128 个字符，其中除附录 A 表 2 中的 34 个控制字符之外，都是可打印字符。从表 1 可以看出，表示数字 0～9 及字母 A～Z(26 个)的 ASCII 码若看成是十六进制数，则无论是数字还是字母，其对应的十六进制数是按数字或字母顺序递增的，这就给排序运算带来方便。

十六进制数(0～F)的 ASCII 码与十六进制数本身存在简单的对应关系，即对于 0～9，其 ASCII 码为 $30_{16}\sim39_{16}$(即 30H～39H，H 相当于下标 16，表示是十六进制数)，因而减去 30H，