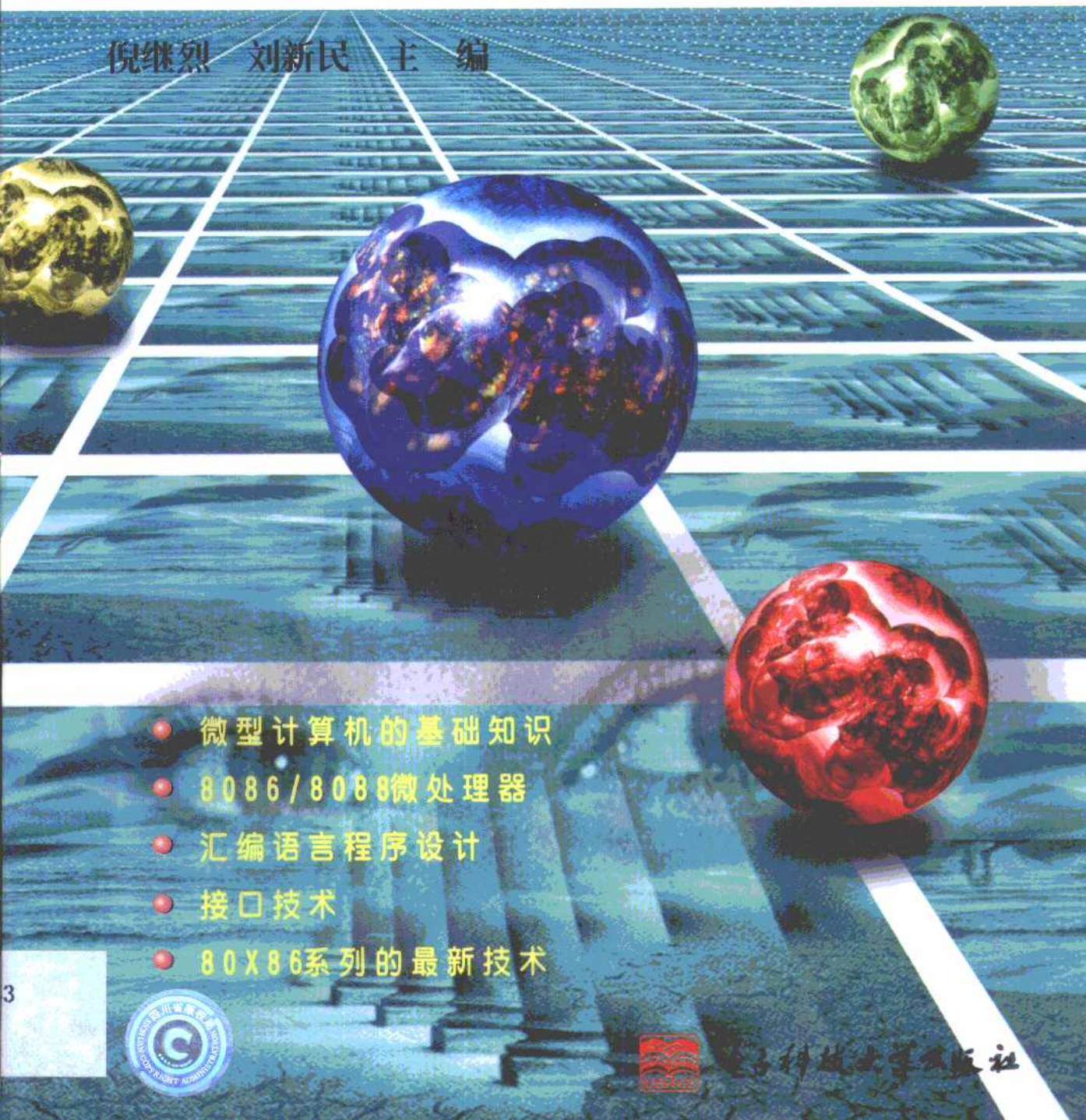
高等学校计算机系列教材

微机原理与接口技术

倪继烈　刘新民　主　编



- 微型计算机的基础知识
- 8086/8088微处理器
- 汇编语言程序设计
- 接口技术
- 80X86系列的最新技术

科学出版社

高等学校计算机系列教材

微机原理与接口技术

丛书主编：刘甫迎
丛书副主编：朱晋蜀　党晋蓉　杨明广　廖亚平
丛书编委：邓礼清　王道学　姜文国　倪继烈　刘枝盛　许鸿川
蒋正萍　宋国明　刘新民　刘　虹　张　京　陈　琳
岳德坤　李　琦　刘光会　饶　斌　蔡方凯
主　　编：倪继烈　刘新民

电子科技大学出版社

内 容 提 要

本书以当前国内外广泛使用的16/32位微处理器为背景，全面、系统地讨论了微型计算机的基本概念、工作原理和实用接口技术。

全书共分8章，主要介绍微机系统的基础知识、8086/8088的内部结构；外部引脚、基本配置和总线时序；指令系统与汇编语言程序设计方法；半导体存储器的工作原理与接口设计；输入/输出与中断技术；常用接口设计技术；从80286到Pentium微处理器的最新技术发展。

本书内容先进，结构新颖，深入浅出，便于教学与自学。它既可以作为大专院校各专业"微机原理与接口技术"课程的通用教材、成人高等教育的培训教材和自学读本，也可供广大科技工作者参考。

高等学校计算机系列教材

微机原理与接口技术

倪继烈　刘新民　主编

出　　版：电子科技大学出版社　（成都建设北路二段四号，邮编：610054）
责任编辑：朱　丹
发　　行：新华书店经销
印　　刷：四川导向印务有限公司
开　　本：787×1092　1/16　**印张**　24.5　**字数**　635千字
版　　次：2000年2月第一版
印　　次：2000年4月第二次印刷
书　　号：ISBN 7-81065-326-1/TP·205
印　　数：4001—7000册
定　　价：28.00元

前　言

本教材是以 Intel 8086/8088 系列微型计算机为背景，由浅入深、全面系统地阐述微型计算机的基本概念和原理。同时，介绍了从 Intel 80286 到 Pentium 微处理器的最新技术发展。

本教材是作者在多年讲授“微机原理与接口技术”课程原稿的基础上几经修改而成的。编写时力求循序渐进、由浅入深地讲述 Intel 8086 CPU 的内部结构、工作原理、外部引脚、汇编语言程序设计、中断系统以及 I/O 接口技术等内容。

本教材第一章介绍计算机的基础知识，计算机中的数制与编码，计算机的工作过程；第二章介绍 8086 微处理器的内部结构，8086 寄存器结构，8086 系统结构和时序；第三章介绍 8086 指令系统、指令格式、寻址方式、指令功能；第四章介绍汇编语言程序设计、语句结构、伪指令、汇编语言程序设计实例；第五章介绍存储器的分类，RAM 与 ROM 存储器的内部结构、工作原理、常用 RAM 与 ROM 芯片以及存储器与微处理器的连接；第六章介绍输入、输出和中断，输入/输出传送方式，中断的基本概念，8086 中断系统，8259 中断控制器；第七章介绍接口技术，8255A 可编程并行接口芯片，8253A 定时/计数器芯片，8251A 串行接口芯片，8237A 可编程 DMA 控制器，键盘与显示器接口，A/D 与 D/A 转换器芯片；第八章介绍从 80286 到 Pentium 的最新技术发展；附录介绍“微机原理与接口技术”课程的教学大纲及其实验指导等内容。

本书由倪继烈、刘新民主编，其中，第一、二、三、四、八章、附录由倪继烈编写，第五、六、七章由刘新民编写，全书由倪继烈统稿。

由于编者水平有限，书中难免存在一些缺点和错误，殷切希望广大读者批评指正。

编　者

1999 年 9 月

第一章　微型计算机基础知识

1.1　微型计算机发展过程简介

电子计算机是20世纪最新科技成就之一。自从1946年世界上第一台电子计算机问世以来，随着计算机逻辑元件的不断更新，它已经历了电子管、晶体管、集成电路以及大规模、超大规模集成电路VLSIC计算机4代发展时期。

微型计算机MC是第四代计算机向微型化方向发展的一个非常重要的分支，它的发展是以微处理器MPU的发展为标志的。

自从1971年美国INTEL公司研制成功以INTEL 4004微处理器为核心的4位数计算机以来，微型计算机技术获得了飞速发展。微处理器的集成度差不多每两年翻一番，且性能增长一个数量级。因此，完全可以名副其实地讲，微处理器及微型计算机的发展正日新月异。纵观其发展历史，仅仅20多年来，已经推出了4代微处理器产品。

第一代：(1971年至1972年)

第一代微处理器是以INTEL公司1971～1972年推出的4004和8008作为典型代表，其集成度为2千只晶体管/片，时钟频率为2MHz。

第二代：(1973年至1977年)

第二代微处理器的代表产品是美国INTEL公司的8080/8085、MOTOROLA公司的6800和ZILOG公司研制的Z80，其集成度为9千只晶体管/片，时钟频率为5MHz，它们是高性能的8位微处理器。

第三代：(1978年至1981年)

代表产品是美国INTEL公司的8086/8088，ZILOG公司的Z8000和MOTOROLA公司的68000，它们是16位微处理器，又称为第一代超大规模集成电路的微处理器。其集成度为2.9万只晶体管/片，时钟频率为8MHz，它们采用HMOS高密度工艺，运算速度比8位机快2～5倍，赶上或超过了70年代小型机的水平。

第四代：(1981年以后)

80年代以后，微处理器进入第四代产品，向系列化方向发展，INTEL公司相继推出了性能更高、功能更强的80386和80486微处理器，它们与8086向上兼容，是32位微处理器，又称为超级微处理器。

进入90年代以来，INTEL公司在开发新一代微处理器技术方面继续领先，1993年3月，INTEL发布了微处理器产品Pentium，Pentium的最高工作频率可达66MHz，运行速度达

112MIPS，利用亚微米级的 CMOS 技术，使集成度高达 310 万只晶体管/片。

纵观计算机的发展之所以如此迅速，这主要取决于其独具的特点：体积小、价格廉、可靠性高、通用性强、功耗低以及研制周期短。

当前，微处理器与微型计算机正朝着以下几个方向发展：

(1)发展高性能的 32 位微处理器；

(2)发展专用化的单片微型计算机；

(3)发展带有软件固化的微计算机；

(4)发展多微处理机系统和局域网络；

(5)充实和发展外围接口电路。

1.2 计算机中数的表示方法

1.2.1 数制

数制是人们利用符号来计数的科学方法，数制有很多种，但是在计算机的设计及使用中通常使用的计数方法是二进制、十进制、八进制和十六进制。

1. 数制的基与权

在一个记数制中，表示每个数位上可用字符的个数称为该记数制的基数，例如，十进制记数制中有 0 到 9 十个字符，基数为 10；二进制记数中只有 0 和 1 两个字符，基数为 2。一个数值中的每一位都有一个表示该位在数值中位置的值与之相对应，这个值称为权。

二进制(Binary System)：使用的数字为 0 和 1，二进制数值中各位的权为以 2 为底的幂，如：

1	0	0	1	1	0	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

各位的权为 $2^0,2^1,\cdots,2^7$，即 1,2,4,…,128，有时也顺次称其各位为 0 权位、1 权位、2 权位等。

十进制(Decimal System)：使用的数字为 0,1,2,3,4,5,6,7,8,9，共 10 个数字，十进制值中各位的权为以 10 为底的幂，如：

3	8	7	4	9	0	2	3
10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0

各位的权为 $10^0,10^1,\cdots,10^7$，即 1,10,100,…，有时又称为 0 权位、1 权位、2 权位……

以此类推，八进制(Octave System)的基数为 8，使用的数为 0,1,2,3,4,5,6,7，共 8 个数，各位的权是以 8 为底的幂，即 $8^0,8^1,8^2,8^3$……

十六进制(Hexadecimal System)的基数为16,使用的数为0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,共16个数,各位的权是以16为底的幂,即,16^0,16^1,16^2,16^3,…。在十六进制中,A,B,C,D,E,F代表10进制数中的10,11,12,13,14,15。

2. **二进制数的特点**

尽管人们习惯用十进制数,但计算机中却采用二进制数。将八进制数、十六进制数引入计算机,主要是为了书写方便,这仅仅是一种手段而已。

二进制数与其他数制相比,有以下特点:

(1)数制简单、容易表示。

二进制数只有"0"和"1"两种数码,任何具有两个不同稳定状态的元件,都可以用来表示二进制数的每一位。而制造具有两个稳定状态要比制造多稳定状态(如10个稳定状态)的元件容易得多,如晶体管的导通和截止、电容的充电和放电、磁芯两个不同状态的磁化等。在计算机中通常采用电平的"高"、"低"或脉冲的"有"、"无"来分别表示"1"和"0"。这种简单的工作状态可靠,抗干扰能力强。

(2)运算规则简单。

二进制运算的规则非常简单,所以在计算机中实现二进制运算的线路也大为简化。有关运算规则将在后面介绍。

(3)节省设备。

若采用十进制数,则有0～9十个数码,表示一个数位共需10个完全不同的设备状态。如果用二进制数来表示十进制数,则一位十进数可用4位二进制数来表示。二进制数的每一位只有两个状态,总共8个状态,其表示范围为0000～1111,即0～15。这说明采用二进制数,可以节省设备。

(4)可以使用逻辑代数这一数学工具对计算机逻辑线路进行分析和综合,便于机器结构的简化。

尽管在计算机内部采用二进制操作,但是对于人们来说,使用二进制并不方便,如书写冗长,阅读不便。为此,通常用八进制或十六进制的缩写方式。此外,人们又习惯于用十进制,这就需要解决不同数制之间的相互转换问题。

3. **数制之间的转换**

在任一记数制中的一个数都可用它的按权展开式表示,即:

$$S=\sum_{i=-m}^{n-1} r_i R^i$$

式中,r为该记数制中的任一个数字,R为基数,如:

$$(11\,101.1101)_2=1\times2^4+1\times2^3+1\times2^2+1\times2^0+1\times2^{-1}+1\times2^{-2}+1\times2^{-4}$$

$$(387.46)_{10}=3\times10^2+8\times10^1+7\times10^0+4\times10^{-1}+6\times10^{-2}$$

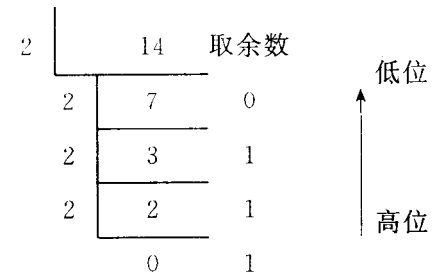
$$(746.1)_8=7\times8^2+4\times8^1+6\times8^0+1\times8^{-1}$$

$$(5F3B.AC)_{16}=5\times16^3+15\times16^2+3\times16^1+11\times16^0+10\times16^{-1}+12\times16^{-2}$$

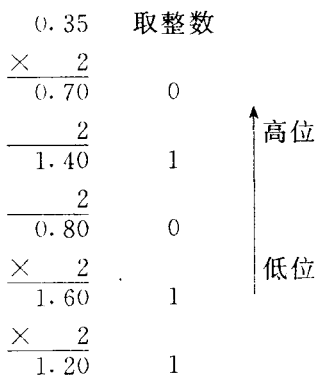
求出按权展开式的值就是该数转换为十进制的等价值。

十进制转换成二进制:整数用"除2取余",小数用"乘2取整"的方法,如$(14.35)_{10}$的转换

方法如下：



$(14.35)_{10}=(1110.01011)_2$



从上面可看出，转换后的小数部分有误差，一般转换到所要求的精度为止。

十六进制转换为二进制：不论是十六进制的整数或小数，只要把每一位十六进制数用相应的四位二进制代替即可，如：

$$(3AB.7E)_{16}=(0011\ 1010\ 1011.0111\ 1110)_2$$

二进制转换成十六进制：整数部分由小数点向左每四位一组，小数部分由小数点向右每四位一组，不足四位的补 0，然后用四位二进制的相应的十六进制代替即可，如：

$$(101,1101,0101,1010.1011,01)_2=(5D5A.B4)_{16}$$

0101 1101 0101 1010 1011 0100

5 D 5 A B 4

表 1-1 列出了各种数制的对照关系。

表 1-1 各种数制的对照表

十进制数	十六进制数	八进制数	二 进 制 数	十进制数	十六进制数	八进制数	二 进 制 数
0	0	0	00000000	9	9	11	00001001
1	1	1	00000001	10	A	12	00001010
2	2	2	00000010	11	B	13	00001011
3	3	3	00000011	12	C	14	00001100
4	4	4	00000100	13	D	15	00001101
5	5	5	00000101	14	E	16	00001110
6	6	6	00000110	15	F	17	00001111
7	7	7	00000111	16	10	20	00010000
8	8	10	00001000	17	11	21	00010001

今后为了便于区别不同数制表示的数，规定在数字后面用一个 H 表示十六进制数，用 Q 表示八进制数，用 B 表示二进制数，用 D(或不加标志)表示十进制数，如 64H、754Q、1101B、369D 分别表示十六进制、八进制、二进制和十进制数。另外，规定当十六进制数以字母开头时，为了避免与其他字符相混，在书写时前面加一个数 0，如十六进制数 B9H，应写成 0B9H。

1.2.2 计算机中的编码

计算机中处理任何问题都是数字化的。由于二进制的数字只有两个数0、1，而具有两个状态的电路较多，易于实现，所以计算机的一切数据均采用二进制的数。但计算机又应识别和处理各种字符和符号，如英文大小字母、标点符号、各种运算符等等，由于计算机中的基本电子器件仅具有两种状态，所以，各种文字和符号只能采用二进制数码的组合来表示，称为二进制编码。

1. 二进制编码的十进制数

机器内部的数是采用二进制数，但输入、输出时通常还是采用十进制数，不过这样的十进制数是用二进制编码表示的，即一个十进制的数用4位二进制编码表示，这就是二进制编码的十进制数，简称BCD码(Binary-Coded Decimal)。

4位二进制数码有16种组合，原则上可任选其中的10种作为代码，分别代表十进制中0～9这10个数字。为便于记忆和比较直观，最常用的方法是8421BCD码，8、4、2、1分别是4位二进制数的位权值。表1-2给出了十进制数和二进制编码的对应关系。

表1-2 8421BCD码

十进制数	8421BCD码	十进制数	8421BCD码
0	0000	5	0101
1	0001	6	0110
2	0010	6	0111
3	0011	8	1000
4	0100	9	1001

这种BCD码与十进制数的关系直观，其相互转换也很简单。

如十进制数的25，用BCD码表示时，2用0010代替，5用0101代替，25的BCD码表示应为00100101(BCD)。可以看出，BCD码只是用二进制代码表示的十进制数，它并不是等价的二进制数，25的等价的二进数应是00011001(2)。BCD码和十六进制数也不同，十六进制与二进制都是进位计数制的一种，而BCD码仅仅是一种代码表示法。又如十进制数125，其值与二、十六进制及BCD码的关系如下：

125　　01111101　　7DH　　0001,0010,0101(125H)

十进制数　二进制数　十六进制数　BCD代码

BCD码的优点是与十进制数转换方便，容易阅读；缺点是用BCD码表示的十进制数的数位要较纯二进制表示的十进制数位更长，使电路复杂性增加，运算速度减慢。

当希望计算机直接用十进制数进行运算时，应将数用BCD码来存储和运算。例如4+3，应是0100+0011=0111。但若是改为4+8，直接运算结果为0100+1000=1100，但从BCD数的运算来说应为0001 0010，亦即十进制数12。因此，在这种情况下，就要对二进制数的运算结果(1100)进行调整，使之符合十进制数的运算和进位规律。这种调整称为十进制调整。其内容

有两条：

(1)若两个 BCD 数相加，结果大于 1001，亦即大于十进制数 9，则应作加 0110(即加 6)调整；

(2)若两个 BCD 数相加，结果在本位上并不大于 1001，但却产生了进位，相当于十进制运算大于等于 16，则也要作加 0110(加 6)调整。

如上面提到的 4+8，直接运算结果为 0100+1000=1100，作加 6 调整后所得结果为 1100+0110=0001 0010，亦即十进制数 12，结果正确。因此，BCD 数运算一定要作十进制调整。

【例 1】 用 BCD 数完成 54+48 的运算。

```
    0101 0100
+)  0100 1000
-------------
    1001 1100
+)       0110        加 6 调整
-------------
    1010 0010
+)  0110             高 4 位加 6 调整
-------------
0001 0000 0010
```

此时，低 4 位之和为 1100，应作加 6 调整，调整后高 4 位又为 1010，还应作一次加 6 调整，故最后结果为 0001 0000 0010，即 102。

若是两个 BCD 数相减，则也要进行十进制调整，其规律是：当相减时，若低 4 位向高 4 位有借位，在低 4 位就要作减 0110(减 6)调整。

2. **字符代码**

在计算机中，除数字外，还需处理各种字符，如字母、运算符号、标点符号等等。这些字符都要用代码来表示。最常用的代码是 ASCII 码(美国标准信息交换码的缩写)。它用 7 位 2 进制码对字符进行编码，如表 1-3 所示。

查找字符的 ASCII 码时，首先从表中找到相应的字符，从该字符垂直向上查得十六进制的高位，再从该字符水平向左查得其十六进制的低位，两者合并起来，即得该字符的 ASCII 码。

NUL	空白	DLE	转意
SOH	序始	DC1	设备控制 1
STX	文始	DC2	设备控制 2
ETX	文终	DC3	设备控制 3
EOT	送毕	DC4	设备控制 4
ENQ	询问	NAK	否认
ACK	承认	SYN	同步
BEL	告警	ETB	组终
BS	退一格	CAN	作废
HT	横表	EM	载终
LF	换行	SUB	取代
VT	纵表	ESC	扩展

FF	换页	FS	卷隙
CR	回车	GS	群隙
SO	移出	RS	录隙
SI	移入	US	元隙
SP	空格	DEL	作废

表 1-3 ASCII(美国标准信息交换码)表

16进制高位	0	1	2	3	4	5	6	7
16进制低位	位 654→ ↓ 3210	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	、	P
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	,,	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	,	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	.	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	ST	US	/	?	O	-	o	DEL

例如,数字 0,查得其十六进制高位为 3,十六进制低位为 0,故 0 的 ASCII 码为 30H。同理得到数字 1 的 ASCII 码为 31H,2 的 ASCII 码为 32H……字母 A 的 ASCII 码为 41H,a 的 ASCII 码为 61H,等等。

在 7 位 ASCII 码中,第八位常用作奇偶校验位,以确定数据传输是否正确。该位的数值由所要求的奇偶类型确定。偶数奇偶校验是指每个代码中所有 1 位的和(包括奇偶校验位)总是偶数。例如,传递的字母是 G,则 ASCII 代码是 1000111,因其中有 4 个 1,所以奇偶位是 0,8 位代码将是 01000111。奇数奇偶校验是指每个代码中所有 1 位的和(包括奇偶校验位)是奇数,若用奇数奇偶校验传送 ASCII 代码中的 G,其二进制表示应为 11000111。

1.2.3 计算机中有符号数的表示方法

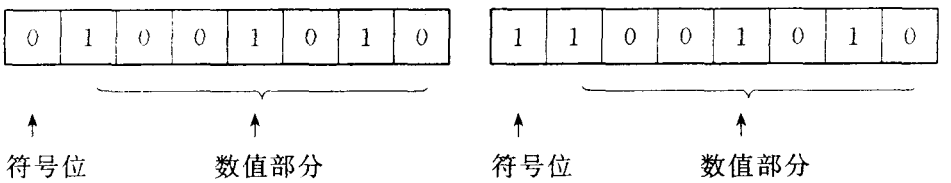
实际的数值是带有符号的,即可能是正数,也可能是负数;而数值本身可以包括整数,也可包括小数。那么,在计算机中是如何表示数的正负号和确定小数点的位置呢?

1. 数的符号表示法

由于在计算机中，二进制数码是用双稳态元件来表示的，因此，对于数的符号“+”或“−”很容易想到也用一位数码来表示，即：

用数码“0”表示正数的符号“+”；用数码“1”表示负数的符号“−”。

有符号数在机器中的表示形式为：



前者表示+74，后者表示−74。

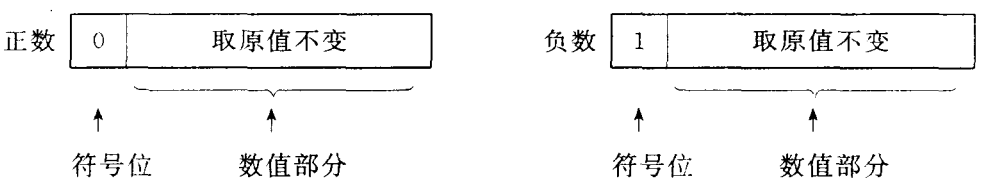
这种连正负号也数字化的数，称为机器数，是计算机所能识别的数；而把这个数本身，即用“+”、“−”号表示的数称为真值。

2. 原码、反码和补码

机器数有3种不同的编码形式，即原码、反码及补码。现分述如下：

(1)原码

在用二进制原码表示的数中，符号位为0表示正数，符号位为1表示负数。其余各数值位取原值不变。有符号数的这种表示法称为原码表示法，在机器中的表示形式为：



①对于正数(字长=8)

$$X=+1101001(+105)$$

$$[X]_{原}=\ 0\quad 1101001$$

符号位 数值位

②对于负数

$$X=-1101001(-105)$$

$$[X]_{原}=\ 1\quad 1101001$$

符号位 数值位

③对于0

可以认为它是(+0)，也可以认为是(−0)，这样，0在原码中有下列两种表示法：

$$[+0]_{原}=00000000$$

$$[-0]_{原}=10000000$$

遇到这两种情况，计算机都当作 0 来处理。

对于 8 位二进制原码，其表示的数值范围为+127～-127，即：

$$[+127]_{原}=01111111$$

$$[-127]_{原}=11111111$$

原码简单易懂，且与真值转换方便。但用原码将两个异号数相加或两个同号数相减，就要作减法，这样，在计算机中还需加一个符号比较电路和一个减法电路，这样就增加了运算电路的复杂性。为了把上述运算转换成加法运算，简化计算机运算电路的结构，在计算机中引入了反码和补码。

(2)反码

反码的定义如下：

①对于正数(字长=8)：

反码的表示法与原码相同：最高位为符号位，用 0 表示，其余各数值位不变，即：

$$[X]_{反}=[X]_{原}$$

例如：

$$X=+1101001(+105)$$

$$[X]_{反}=0\ 1101001$$

符号位　数值位

以+4、+28、+127 为例：

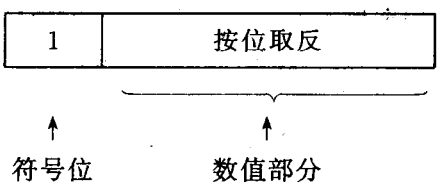
$$[X]_{反}=[+4]_{反}=00000100$$

$$[X]_{反}=[+28]_{反}=00011100$$

$$[X]_{反}=[+127]_{反}=01111111$$

②对于负数(字长=8)：

负数的反码表示，除符号位仍为"1"外，其余各数值位是"按位取反"，即：



例如：

$$X=-1101001(-105)$$

$$[X]_{反}=1\ 0010110$$

符号位 数值位

以-4、-28、-127 为例，并与+4、+28、+127 的反码相对照：

$$[X]_{反}=[+4]_{反}=00000100$$

$$[X]_{反}=[-4]_{反}=11111011$$

$$[X]_{反}=[+28]_{反}=00011100$$

$$[X]_{反}=[-28]_{反}=11100011$$

$$[X]_{反}=[+127]_{反}=01111111$$

$$[X]_{反}=[-127]_{反}=10000000$$

③反码的“0”，有两种表示方法：

$$[+0]_{反}=00000000$$

$$[-0]_{反}=11111111$$

对于 8 位反码表示的数值范围为+127～-127，即：

$$[+127]_{反}=01111111$$

$$[-127]_{反}=10000000$$

(3)补码

①补码的概念

在日常生活中，有不少“补”数的例子。就以钟表为例，假定现在的标准时间是 3 点整，而表的时针指向 6 点整。为了校准时间，可以有两种拨针法：

(a)倒拨 3 小时　　6－3＝3

(b)顺拨 9 小时　　6＋9＝3

在这里 6 加 9 与 6 减 3 是相同的。

这是因为，当时针顺拨时，到达 12 点后，就从 0 重新开始，相当于自动丢失了一个数字 12。

$$6+9=12(\text{自动丢失})+3=0+3=3$$

这个自动丢失的数(12)就称为“模”。由此可以看出，对于一个模数为 12 的循环计数系统来说，6 减 3 与 6 加 9 是等价的，或者说，(－3)与(＋9)对模 12 互为补数。这可以用数学式表示为：

$$6-3\equiv 6+9 \pmod{12}$$

或

$$-3\equiv +9 \pmod{12}$$

mod 12 表示以 12 为模数。当等式两边同除以模 12，它们的余数相同，故上式在数学上称为同余式。和(－3)与(＋9)的同余相仿，(－5)与(＋7)、(－6)与(＋6)、(－7)与(＋5)等等也都同余，或互为补数。不难看出，模和某个数 X(例如－3)相加所得的数[12＋(－3)]＝9 就为该数(－3)对模 12 的补数，用$[X]_{补}$表示，即：

$$[X]_{补}=模+X$$

式中，模＝基数^{n}，n 为计算装置的位数(字长)。其实模也就是计算装置的容量(该装置能够表示的数码总数)，并且在该装置中模的值与零等值。

在计时钟表中，引入补数概念之后，就可以将原来的减法 6－3 转化为加法 6＋9＝12(自动丢失)＋3＝3。

对于任何计数装置来说，都一定存在一个固定的模(容量)。

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，则计算机中数码的总数(包括符号位)为 2^n。因此计算机的补码是以“2^n”为模，即：

$$[X]_{补}=2^n+X \qquad (字长=n)$$

例如：　X＝－1010111　　(字长＝8)

则

$$[X]_{补}=2^8+[-1010111]=100000000-1010111$$

$$=10101001 \pmod{2^8}$$

这种求$[X]_{补}$的方法因为要作一次减法很不方便。其实，要得到二进制数的补码，可以不必用减法，它可由该数的反码加1后得到。

②补码的定义

补码的定义如下：

(a) 对于正数(字长=8)

与反码一样，正数的补码表示与原码相同，最高位为符号位，用0表示，其余各数值位不变，也即正数的补码就是正数本身。用公式表示为：

$$[X]_{补}=2^8+X=0+X=[X]_{原} \quad (X\geqslant 0)$$

例如：$X=+1101001 \quad (+105)$

$[X]_{补}=\ 0 \quad 1101001$

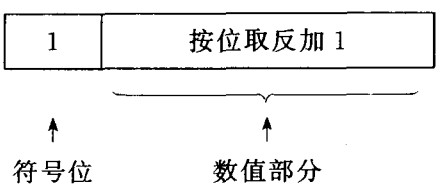
符号位 数值位

(b) 对于负数(字长=8)

负数的补码表示，除符号位仍为"1"外，其余各数值位"按位取反再加1"，即：

$$[X]_{补}=2^8+X=(11111111+X)+1 \quad (X<0)$$
$$=[X]_{反}+1$$

或者



例如：$X=-1101001(-105)$ (字长=8)

则 $[X]_{补}=[X]_{反}+1=10010110+1$

$=\ 1\ 0010111$

符号位 数值位

(c) 补码的"0"只有一种表示方法

$$[+0]_{补}=[-0]_{补}=00000000$$

对于8位二进制补码，其表示的数值范围为+127～-128，即：

$$[+127]_{补}=0111,1111$$
$$[-128]_{补}=1000,0000$$

由于原码、反码中的"0"有两个代码，而补码中的"0"只有一种代码，因此，8位二进制补码可以比原码、反码多表示一个负数，即-128。

3. 原码、反码和补码之间的转换

正数的原码、补码表示方法相同，不存在转换问题。下面讨论负数情况。

(1)已知$[X]_{原}$，求$[X]_{补}$

例如：

$$
\begin{array}{l}
[X]_{原}=10011010 \\
\qquad\quad \downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow \\
\qquad\quad 11100101 \\
+) \qquad\qquad\quad 1 \\
\hline
[X]_{补}=11100110
\end{array}
$$

注意：求反时符号位不变。

(2)已知$[X]_{补}$，求$[X]_{原}$

我们有$[[X]_{补}]_{补}=[X]_{原}$。

例如：

$$
\begin{array}{l}
[X]_{补}=1110110 \\
\qquad\quad \downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow \\
\qquad\quad 1001001 \\
+) \qquad\qquad 1 \\
\hline
[X]_{原}=1001010
\end{array}
$$

(3)求补：已知$[X]_{补}$，求$[-X]_{补}$

所谓求补，就是将$[X]_{补}$连同符号位一起逐位求反，然后在末位加1，即得到$[-X]_{补}$。

应注意的是，不管$[X]_{补}$是正数还是负数，都应按上述方法判别。

例如：

$$
\begin{array}{l}
[X]_{补}=01010110 \\
\qquad\quad \downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow \\
\qquad\quad 10101001 \\
\quad +) \qquad\qquad 1 \\
\hline
[-X]_{补}=10101010
\end{array}
$$

又如：

$$
\begin{array}{l}
[X]_{补}=10101110 \\
\qquad\quad \downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow \\
\qquad\quad 01010001 \\
\quad +) \qquad\qquad 1 \\
\hline
[-X]_{补}=01010010
\end{array}
$$

求补在进行补码的减法运算时会用到。

下面我们将部分十进制及其对应的二进制数、原码、反码和补码的表示归纳如表1-4所示，以此作为一个小结。

4. 补码的加减法运算

在微机中，凡是带符号的数一律用补码表示，而且，运算结果自然也是补码。

补码加减法运算是带符号数加法运算的一种。其运算特点是：符号位与数值部分一起参加运算，并且自动获得结果(包括符号和数值部分)。

(1)补码的加法运算

在进行补码的加法时，不管两个数是正数还是负数，按补码的和等于和的补码进行。

表 1-4　8 位机器数的对照表

二进制数码表示	无符号十进制数	原　码	反　码	补　码
0000 0000	0	+0	+0	+0
0000 0001	1	+1	+1	+1
0000 0010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
0111 1100	124	+124	+124	+124
0111 1101	125	+125	+125	+125
0111 1110	126	+126	+126	+126
0111 1111	127	+127	+127	+127
1000 0000	128	−0	−127	−128
1000 0001	129	−1	−126	−127
1000 0010	130	−2	−125	−126
⋮	⋮	⋮	⋮	⋮
1111 1100	252	−124	−3	−4
1111 1101	253	−125	−2	−3
1111 1110	254	−126	−1	−2
1111 1111	255	−127	−0	−1

因为　$[X]_{补}+[Y]_{补}=2^n+X+2^n+Y=2^n+(X+Y)$

$=[X+Y]_{补}$　　$(\text{mod } 2^n)$

所以　　$[X]_{补}+[Y]_{补}=[X+Y]_{补}$

①两个正数相加

【例 2】　已知 X＝＋1000000，Y＝＋0001000，求两数的补码之和。

由补码表示法有：

$[X]_{补}=01000000$，$[Y]_{补}=00001000$

则

$$\begin{array}{rr} [X]_{补}=01000000 & +64 \\ +)\ [Y]_{补}=00001000 & +)+8 \\ \hline [X]_{补}+[Y]_{补}=01001000 & +72 \end{array}$$

所以有：

$[X+Y]_{补}=01001000$　　$(\text{mod } 2^8)$

此和数为正，其真值为＋72；又因＋64＋(＋8)＝＋72，故结果是正确的。

【例 3】　X＝＋1000000，Y＝＋1000001，进行补码的加法运算。

$$\begin{array}{lll} & [X]_{补}=01000000 & (+64\text{ 的补码}) \\ + & [Y]_{补}=01000001 & (+65\text{ 的补码}) \\ \hline & [X]_{补}+[Y]_{补}=10000001 & (-127\text{ 的补码}) \end{array}$$

↑

符号

两个正数相加，答案却是负的（“和”的最高位为 1），显然这个结果是错误的。其原因是两