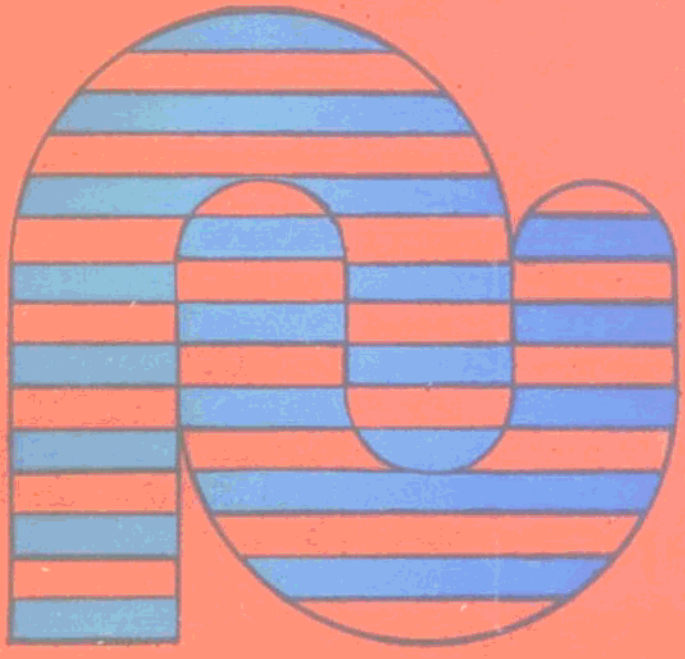


主编 俞云奎
张仁忠

PC机原理与应用

哈尔滨工程大学出版社

前　言

自70年代微处理器诞生以来，以微处理器为中央处理机的微机系统层出不穷。其中，以 Inter 8088 为 CPU 的 16 位微型计算机系统 IBM－PC/XT 是最有代表性的主流机型。由于其结构设计先进、软件配置丰富、硬件资料齐全、性能价格比较好，争取了越来越多的用户。它的许多设计思想、芯片连接、信号关系等都成为更高档微机设计时的参考对象和考虑因素。可见，了解和掌握 IBM－PC/XT 机的原理和应用，对于应用和开发微机系统提供了充分的依据。

本书正是以此为出发点，围绕 IBM－PC/XT 来讲述微型机的结构、工作原理、软件编程、硬件接口等一系列技术。理论紧密结合实际，部分紧密结合系统。

本书的特点是通俗、简明、实用。注重基础内容，如计算机的运算基础、微型机的硬件基础、汇编语言程序设计基础以及中断系统和接口知识基础。掌握这些内容后，将为读者了解和使用微型机打下良好基础。

参加本书编写的有俞云奎(第1、3章)、张仁忠(第5章)、蒋文华(第4、6章)、张玉安(第2章和5.5节)。全书由俞云奎、张仁忠主编，李殿璞教授主审。

由于编者水平有限，书中难免有错误和不妥之处，敬请读者提出宝贵意见。

编　者

1996年1月

目　录

第1章　微型计算机基础

1.1　引　言

1.1.1　微型机的特点与发展

计算机的基本功能是作计算。其特点是：这种计算能自动、高速、相当精确地进行。

从第一台计算机诞生起，至今仅仅40多年历史。然而，它发展之快，普及之广，对科学技术以至整个社会影响之深，是任何其它装置所不及的。40多年来，计算机已发展了四代，现在正向第五代计算机发展。

第一代(1946～1958年)：电子管时代，主要用于科学计算。第二代(1958～1965年)：晶体管时代，除了科学计算之外，数据处理被广泛应用，同时开始用于过程控制。第三代(1965～1970年)：集成电路时代，这一时期，计算机在科学计算、数据处理、过程控制诸方面都得到了广泛应用。第四代(1971年以后)：大规模集成电路时代，这一代发展的一个重要分支是微型计算机。目前，世界各国计算机的主要发展动向是：大型巨型化(主要指性能)、小型微型化(主要指规模和成本)、计算机网络和人工智能。近几年来，美国、日本等国正在加紧研制第五代计算机，这一代计算机主要着眼于机器的智能化，它以知识库为基础，采用智能接口进行逻辑推理，完成判断和决策任务。它是完全新型的一代计算机。

电子计算机通常按体积、性能和价格分为巨型机、大型机、中型机、小型机和微型机五类。从系统结构和基本工作原理上说，微型机和其它几类计算机并没有本质上的区别。所不同的是微型机有以下一系列特点：

·体积小，重量轻，价格低廉。由于采用大规模集成电路和超大规模集成电路，使微型机所含的器件数目大为减少，体积大为缩小。50年代要占地上百平方米，耗电上百千瓦的电子计算机实现的功能，在当今，内部只含几十片集成电路的微型机即已具备，并且，其价格越来越便宜。

·可靠性高，结构灵活。由于内部元器件数目少，所以连线比较少。这样，使微型机的可靠性较高，结构灵活方便。

由于微型机具有上面这些特点，所以它的发展速度很快，自从70年代初第一个微处理器诞生以来，微处理器的性能和集成度几乎每两年提高一倍，而价格却降低一个数量级。

自1971年美国Intel公司的M.E. 霍夫发明了第一台微处理机Intel 4004以来，微型机就如雨后春笋般蓬勃发展，并逐渐占领市场。20多年的时间，发展了四代微型机产品。第一代(1971～1973年)，是低档的四位微处理机，一个芯片上集成了1200个晶体管，基本指令执行时间为10～15μs。第二代(1974～1977年)，是8位微处理机和微型计算机，芯片集成度提高到每片9000个晶体管左右，基本指令执行时间为1～1.3μs。第三代(1978～1980年)，是16位微处理机和微型计算机，其中有代表性的8086在一个芯片上集成了29000个晶体管，指令的最短执行时间为400ns，比8位微处理机快2～5倍。它们的速度赶上和超过了小型计算机。软件方面，可以使用多种高级语言，有常驻的汇编程序、完善的操作系统、大型数据库，可以构

成多微处理机系统。16 位微型机的成功，形成了与小型计算机的竞争局面。它弥补了 8 位微型计算机字长和速度的局限所造成的缺陷，为微型计算机在实时数据处理和实时控制领域中的应用开辟了广阔的前景。第四代(1981 年以后)，是 32 位微处理机和微型计算机，如 Intel 公司的 80386。HP 公司生产的 32 位微处理器芯片集成了 45 万个晶体管，时钟频率为 18MHz。现在，32 位微型计算机大有取代中、小型计算机之趋势。

20 多年来微型计算机的主要发展趋势有两大方面。

·提高性能：微处理器片子的集成度越来越高，几乎每两年翻一番，且性能提高一个数量级。以 Intel 公司的产品来说，1971 年的 4004，集成度为 2500 个/片，1975 年的 8085 集成度为 9000 个/片，1978 年的 8086，集成度为 29000 个/片，1980 年的 IAPX43201，集成度为 100000 个/片。

现在 16 位、32 位微处理机已大量面市。

各种微型机的操作系统，如 CP/M、MP/M、CP/M86、Unix(各种变型)、P 系统等等，以及在种种操作系统支持下的高级语言越来越多，从而大大丰富了微型计算机的系统软件。

·降低价格：微型机发展的另一个重要趋势是降低价格。一方面片子的价格降低，另一方面制造了各种价格低廉的微型机。价格低廉，是微型机真正能够在各行各业应用，能深入到办公室自动化，深入家庭，用作个人计算机(Personal Computer)的重要条件。

1.1.2 微型机的应用

由于微型机具有体积小、价格低、耗电少和可靠性高等优点，所以应用范围十分广阔，被誉为无孔不入。现就应用的如下几个方面加以说明。

1. 科学计算

现在，不少微型机系统具有较强的运算能力，特别是多个微处理器模块构成的系统，其功能往往可与大型机相匹敌，而成本却低到足以使大型机趋于淘汰。

计算机用于科学计算可以缩短计算周期、提高计算效率。

2. 数据处理

在生产组织、企业管理、市场预测、情报检索等方面，存在着大量的数据需要及时进行搜集、归纳、分类、整理、存贮、检索、统计、分析、列表、绘图等等。这类问题的特点是数据量大，而运算比较简单，主要是逻辑运算与判断，其处理结果往往以表格或文件形式存贮或输出。

据统计，目前在计算机应用中，数据处理所占的比重最大。微型机配上数据库管理软件以后，可以很灵活地对各种信息按不同的要求进行处理。它使人们从大量繁杂的数据统计和管理事务中解放出来，大大提高了工作质量、管理水平和效率。

3. 过程控制

使用计算机对连续的工业生产过程进行控制被称为过程控制。现在在制造工业和其它生产厂家中几乎到处都可见到微型机控制的自动化生产线，微型机在这些部门的应用为生产能力和产品质量的迅速提高提供了可能。

4. 计算机辅助设计

在飞机制造、船舶制造、建筑工程设计、大规格集成电路和版图设计、机械制造等行业的复杂设计过程中，为了提高设计质量、缩短设计周期、提高设计的自动化水平都在借助于计算机。这种借助于计算机进行设计的方式称为计算机辅助设计，即 CAD(Computer Aided

Design)。CAD 技术的迅速发展，大大提高了工业生产效率和自动化水平。此外，微型机在仪器仪表控制、智能模拟家用电器等方面都得到了广泛的应用。

1.2 计算机中的数和编码

1.2.1 计算机中的数制

1. 二进制数(Binary)

其特点是：

·只有两个基本数字：0 和 1。

·逢 2 进 1，借 1 当 2。即基数是 2，权值是 2^i。

2. 八进制数(Octal)

其特点是：

·有 8 个基本数字：0～7。

·逢 8 进 1，借 1 当 8。即基数是 8，权值是 8^i。

因此，在八进制数中不会出现 8、9 这两个十进制数字，因为$(8)_{10}=(10)_8$，$(9)_{10}=(11)_8$。

3. 十六进制数(Hexadecimal)

其特点是：

·有 16 个基本数字：0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F。这里的 A、B、C、D、E、F 分别表示十进制数的 10、11、12、13、14、15。

·逢 16 进 1，借 1 当 16。即基数是 16，权值是 16^i。

4. 不同数制之间的转换

(1)转换到十进制

方法是按权值展开相加。例如：

$(10111010.1010)_2=1\times2^7+1\times2^5+1\times2^4+1\times2^3+1\times2^1+1\times2^{-1}+1\times2^{-3}=(186.625)_{10}$

$(FA)_{16}=F\times16^1+A\times16^0=15\times16+10\times1=(250)_{10}$

(2)十进制数到其它进制的转换

①整数部分用重复除基值取余数的方法。

a. 将$(215)_{10}$转换成二进制数。

	余数	低位
2 ⌊215		↑
2 ⌊107	1	|
2 ⌊53	1	|
2 ⌊26	1	|
2 ⌊13	0	|
2 ⌊6	1	|
2 ⌊3	0	|
2 ⌊1	1	|
0	1	|
		高位

即$(215)_{10}=(11010111)_2$

b. 将$(253)_{10}$转换成十六进制数

	余数	低位
16 ⌊ 253		↑
16 ⌊ 15	D	|
0	F	|
		高位

即$(253)_{10}=(FD)_{16}$

②小数部分用基值重复相乘取整数的方法。

将$(0.5625)_{10}$转换成二进制数

取整数	
|	0.5625
|	× 2
|1	.1250
|	× 2
|0	.2500
|	× 2
|0	.5000
|	× 2
|1	.0000
↓	

即$(0.5625)_{10}=(0.1001)_2$

(3)二进制、八进制、十六进制之间的转换。

由于一位八进制数相当于三位二进制数，一位十六进制数相当于四位二进制数，故可按如下方法进行转换。例如：

01110101.10100111B

不足添0凑足三位　　不足添0凑足三位

↓　　↓

001 110 101 . 101 001 110

1　6　5 . 5　1　6Q

01110101 . 10100111

7　5 . A　7H

即 01110101.10100111B = 165.516Q = 75.A7H

反之也成立，即一位八进制数变成三位二进制，一位十六进制变成四位二进制数，排列起来即为对应的二进制数。

5. 小结

(1)在计算机中用电路的不同状态来模拟地表示数。由于双稳态电路最易获得，因此在计

算机中用双稳态电路的两个不同稳态来模拟表示数，即计算机中的数用二进制表示，也就是计算机只能识别二进制数，并只能对二进制数进行运算。但二进制数书写起来很长，容易出错，为了便于书写，引入了八进制和十六进制。由于微型机中常以一个字节(8 位)作为基本单位来处理，一个字节正好可用二位 16 进制数来表示，所以，在微型机中常采用十六进制数来书写。

(2)熟记以下关系，可以使我们较快的识别数。

①$2^8=256$；$2^{10}=1024=1K$；

$2^{16}=65536=64K$；$2^{20}=1048576=1024K=1M$。

②n 个 1 的值是 2^n-1。比如 $11111111=2^8-1=255$

③在第 n 位上 1 的值是 2^{n-1}。比如 $10000000=2^{8-1}=2^7=128$。

1.2.2 二进制编码

计算机采用的是二进制，因此，在计算机中数、字母、符号等都以特定的二进制码来表示，这就是二进制编码。

1.BCD 码——用二进制编码的十进制数

(1)什么是 BCD 码

常用的 BCD 码为 8421BCD 码，即每位 10 进制数码都用四位二进制数来表示，2 进位的权值分别为 8、4、2、1。

(2)为什么要引进 BCD 码

①缓解人机矛盾

因为人习惯于十进制数，而计算机只能识别二进制码。为了让计算机运算，必须先将十进制数转换成二进制数，这种转换从原理上讲很简单，但当运算数据较大时，这种转换又很繁琐。而引进 BCD 码后，可以将它作为一种中间表示法，即在输入时，可以将十进制数化作 BCD 码，当计算机接受 BCD 码后，又可通过预先编制好的程序将其转换成二进制数。比如要输入 65535：

人工转换↓输入 BCD 码

0110010101010100110101

计算机用↓程序转换成二进制数

1111111111111111

②可在计算机内部直接以 BCD 码进行十进制运算，这样运算结果便于显示。

(3)计算机按 BCD 码运算时，必须进行调整。

这是由于在计算机内部只能按二进制规则进行运算，而运算结果四位二进制码很可能超过 9，这在 BCD 码中是非法的。在 BCD 码中要求四位二进制码逢十进一，而二进制中，四位二进制码只能逢十六进一。也就是说，按 BCD 码的要求，二进制码晚进了 6。为了纠正非法码，我们可以命令计算机，对二进制的运算结果进行检查，先检查低 4 位，再检查高 4 位，对超出 1001 的编码或 4 位相加时向上有进位的编码进行加六修整(即后述 DAA 指令的功能)。例如：

```
  10001001        89
+ 00011000        18
----------
  10100001        二进制相加
     ∪
+ 01100110        十进制调整
----------
 100000111        BCD 码结果 107
```

2. 奇偶校验码

在原有代码上，增加 1 位，它的取值使新的代码中包含 1 的个数恒为奇(偶)数。例如，0101 的奇校验码为 10101，其中最高位称奇(偶)校验位，代码传送中，如果任一位发生差错，1 的个数就变成偶(奇)数，从而被检查出来。应注意，奇(偶)校验码只能发现单数性错误。而不能检查出双重错误，如丢失两个 1 等，但发生双重错误的可能是很小的。

3. ASCII 码

计算机在进行输入/输出时，除使用数字外还有各种字母和符号。例如字母 A，计算机是不能直接识别的，只有采用二进制的编码方式后才能识别。

目前国际上普遍采用美国信息交换标准代码，简称 ASCII 码(American Standard Code for Information Interchange)。它由七位二进制码表示一个字符。因 $2^7 = 128$，所以它可以表示 128 个字符。它们分成两大类：一类为可打印的图形字符，它包括：52 个大小写英文字母，0～9 十个数字和 34 个专用符号(如 +、-、*、/、=、%……)；另一类作为命令的控制字符，它包括 32 个命令，例如送毕(EOT)、换行(LF)、换页(FF)、回车(CR)等等。

由于微型机通常字长为 8 位，而 ASCII 码只用 7 位，所以常将最高位 b^7 作为奇偶校验位用。ASCII 编码详见表 1-1。从表可见 0～9 的 ASCII 码为 30H～39H；大写字母 A～Z 的 ASCII 码为 41H～5AH

近年来，在标准 ASCII 码基础上，为表示更多符号，将 7 位 ASCII 码扩充到 8 位，可表示 256 个字符，称扩充的 ASCII 码。扩充的 ASCII 码可以表示一些特定的符号，如"÷"为 F6H，"Ω"为 EAH 等。扩充的 ASCII 码只能在不用最高位做校验位或其它用途时使用。

除了以上三种编码外，还有各种形式的汉字编码，这里不作介绍。

1.2.3 二进制数的运算

一种数字系统可进行两种基本的算术运算：加法和减法。利用加法和减法，就可以进行乘法、除法以及其他数值运算。

1. 二进制加法

二进制加法制规则为：

·0 + 0 = 0

·0 + 1 = 1 + 0 = 1

·1 + 1 = 0　　进位 1

若有两数 1101 和 1011 相加，则加法过程如下：

```
进　位          111
被加数          1101
加　数        +1011
----------
              11000
```

可见，两个二进制数相加，每一位有三个数——即相加的两个数以及低位的进位，用二进制的

加法规则得到本位的和以及向高位的进位。

微型机中，通常字长为 8 位。例：两个八位数相加

进　位　　1 1 1 1 1 1

被加数　1 0 1 1 0 1 0 1

加位数　+0 0 0 0 1 1 1 1

和　　1 1 0 0 0 1 0 0

表 1－1　ASCII(美国信息交换标准码)表

列	0③	1③	2③	3	4	5	6	7③
行	位 654→ ↓3210	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	、	p
1	0001	SOH	DC_1	!	1	A	Q	a	q
2	0010	STX	DC_2	"	2	B	R	b	r
3	0011	ETX	DC_3	#	3	C	S	c	s
4	0100	EOT	DC_4	$	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	Ω①	n	~
F	1111	SI	US	/	?	O	—②	o	DEL

①取决于使用这种代码的机器，它的符号可以是弯曲符号、向上箭头等。

②取决于使用这种代码的机器，它的符号可以是在下面划线、向下箭头、或心形。

③是第 0、1、2 和 7 列特殊控制功能的解释。

NUL	空	DEL	作废
SOH	标题开始	VT	垂直制表
STX	正文结束	FF	走纸控制
EXT	文本结束	CR	回车
EOT	传输结果	SO	移位输出
ENQ	询问	SI	移位输入

ACK	承认	SP	空间(空格)
BEL	报警符	DLE	数据链换码
BS	退一格	DC_1	设备控制 1
HT	横向列表	DC_2	设备控制 2
LF	换行	DC_3	设备控制 3
SYN	空转同步	DC_4	设备控制 4
ETB	信息组传送结束	NAK	否定

2. 二进制减法

二进制减法的运算规则为：

·0－0＝0

·1－1＝0

·1－0＝1

·0－1＝1　有借位

例：11000100－00100101，列出式子为：

借　　位	1 1 1 1 1 1
借位以后的被减数	1 0 1 1 1 0 1
被　减　数	1 1 0 0 0 1 0 0
减　　数	－0 0 1 0 0 1 0 1
差	1 0 0 1 1 1 1 1

与加法类似，每一位有三个数参加运算：本位的被减数和减数，以及低位来的借位，在运算时先用被减数和借位相运算，得到考虑了借位以后的被减数，然后再减去减数，最后可得到每一位的差，以及所产生的借位。下面再举一个例子，说明这样的运算过程。

例：11101110－10111010，式子为：

借　　位	0 1 1 0 0 0 0
借位后的被减数	1 0 0 0 1 1 1
被　减　数	1 1 1 0 1 1 1 0
减　　数	－1 0 1 1 1 0 1 0
	0 0 1 1 0 1 0 0

3. 二进制乘法

二进制乘法的运算规则为：

·0×0＝0

·0×1＝0

·1×0＝0

·1×1＝1

是十分简单的，只有当两个 1 相乘时，积才为 1，否则积为 0。

二进制的乘法也与十进制的类似：

被乘数	1 1 1 1
乘　数	×1 1 0 1

```
    1111
    0000
   1111
  1111
--------
11000011
```

可用乘数的每一位去乘被乘数，乘得的中间结果的最低有效位与相应乘数位对齐，若乘数位为1，所得的中间结果即为被乘数；若乘数位为0，则中间结果为0。最后把这些中间结果一起加起来，就可得到乘积。这种做法由于重复性差，不便于在机器中实现。为了便于在机器中实现，我们把运算方法改变一下。

(1) 被乘数左移的方法：

	乘数	被乘数	部分积
	1101	1111	0000
①	乘数最低位为1，把被乘数加到部分积上，		+1111
	然后把被乘数左移	11110	1111
②	乘数为0，不加被乘数		
	被乘数左移	111100	
③	乘数为1，加被乘数(已左移后的)		1111
	被乘数左移	1111000	111100
④	乘数为1，加被乘数		1001011
	得乘积		1111000
			11000011

从上例中可看到两个 n 位数相乘，乘积为 $2n$ 位。在运算过程中，这 $2n$ 位都有可能要有相加的操作，故需 $2n$ 个加法器。

(2)部分积右移的乘法

上例中是以被乘数左移的方法来实现的，则两个 n 位数相乘，乘积为 $2n$ 位，在运算过程中。这 $2n$ 位都有可能要有相加的操作故需 $2n$ 个加法器。

我们也可以用部分积右移的办法来实现，其过程如下：

	乘数	被乘数	部分积	
	1101	1111	0000	
①	乘数最低位为1，加被乘数		+1111	
	部分积右移		1111	
②	乘数为0，不加被乘数		0111	1
	部分积右移			
③	乘数为1，加被乘数		0011	11
	部分积右移		+1111	
④	乘数为1，加被乘数		10010	11
	部分积右移		1001	011
			+1111	
			11000	011
			1100	0011

最后所得的结果相同。但这种运算方法只有 n 位有相加的操作，故只需 n 个加法器。

4. 二进制除法

除法是乘法的逆运算。与十进制的类似，从被除数的最高位 MSB(Most Significant Bit)开始检查，并定出需要超过除数的位数。找到这个位时商记 1，并把选定的被除数值减除数。然后把被除数的下一位下移到余数上。如果余数不能减去除数(不够减)则商 0，把被除数的再下一位移到余数上；若余数够减则商 1，余数减去除数，把被除数的下一位移到余数上……

```
      000111
   ------------
101) 100011
     101
     ----
      0111
      101
      ----
        101
        101
        ---
          0
```

这样继续下去，直到全部被除数的位都下移完为止。然后把余数/除数作为商的分数，表示在商中。下面再举一个例子说明上述的除法过程：

```
          0001101
       ------------
除 数 110) 1001110    被除数
          110
          ---
           111
           110
           ---
             110
             110
             ---
               0
```

1.2.4 带符号数的表示法

1. 原码表示法

数学上，数有正、负之分，它们分别用“+”、“-”来表示。但对计算机来说，它不认识这两个符号。通常的作法是：将一个二进制码的最高位作为符号位。用“0”表示正，用“1”表示负。这种表示法称为原码表示法。例如：

十进制数 +91 的二进制数为：+1011011　它的原码为：01011011
　　└符号位

十进制数 -91 的二进制数为：-1011011　它的原码为：11011011
　　└符号位

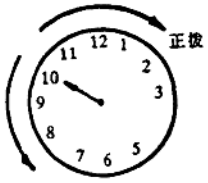
这种符号被数值化了的二进制数称为机器数，而它原来带“+”、“-”号的数就称为机器数的真值。

原码的优点是简单易懂，与真值的转换方便。但是在机器中进行加法运算时就会遇到麻烦。当两数进行相加时，如果是同号，则数值相加，符号不变；如果是异号，则实际上就是做减法，此时需比较两个数哪个绝对值大后，才能确定谁减谁。这种问题对人来说是很容易解决的，但是在计算机中进行，为了判断两数是同号还是异号，比较哪个数的绝对值大，就要增加机器的硬件电路或增加机器的运算时间。为了解决这个矛盾，最好把做减法改为做加法，使计算

机的运算电路只有加法电路，同时又可提高机器的运算速度，从而引出了补码和反码表示法。

2. 补码表示法

我们先来看一个对表的例子，如图 1-1 所示，设现在的正确时间为 6 点，而时钟却指在 10 点的位置，现在我们来校正它，可有两种方法：



正确时间

图 1-1 校表

倒拨：10 - 4 = 6

顺拨：10 + 8 = 6

这里 10 减 4 与 10 加 8 结果是相同的，也就是说减法是可以用加法来代替的。当然，这是有条件的。因为钟表超过 12 以后，它会自动丢失，再从 1 开始计时。这个 12 就是钟表的基数。在数学上把某进位制的基数称为模数，且用符号 mod[ule]表示，即只有在模为 12 的条件下，下列等式才成立。

$$10 - 4 = 10 + 8 \qquad (\mathrm{mod}\ 12)$$

另外，我们可以看出，(-4)和(+8)它们的绝对值之和就是模数 12，即它们对模来说互为补数。所以，求一个负数的补数为：

补数 = 模数 - |原数|

例如，原数为(-4)、(-5)、(-6)的补数对于模 12 来说分别为 8、7、6。

对于计算机中的二进制码来说，如何利用上述的概念来确定模数和补码呢？

对于一个 n 位的二进制数来说，如把它看成是纯整数，则它的模就是 2^n。例如一个四位的计数器，从 0000 计到 1111，再增 1 就回到 0000，向高位的进位将自动丢失，因此它的模就是 $2^4 = 16$。对于 8 位的微型计算机，它的模就是 $2^8 = 256$。

补码定义：

$$[x]_{补} = \begin{cases} x & 2^{n-1} > x \geqslant 0 \\ 2^n - |x| & 0 > x \geqslant -2^{n-1} \end{cases}$$

即正数的补码等于正数本身，负数的补码等于模减去它的绝对值(这里 n 为位数)。

例：当 $n = 4$ 时

$$[x]_{补} = \begin{cases} x & 8 > x \geqslant 0 \\ 2^4 - |x| & 0 > x \geqslant -8 \end{cases}$$

现求$(7)_{10} - (5)_{10} = (0111)_2 - (0101)_2 = ?$

解：先分别求出它们的补码：

$[7]_{补} = 0111$

符号位

$[-5]_{补} = 2^4 - 0101 = 10000 - 0101$

```
  1 0000
- ) 0101
-------
   1011
```

$\therefore [-5]_{补} = 1011$

符号位

变减为加：