微处理机与微型计算机

〔美〕 布朗柯·索切克 主编

林清波 傅树柱 等译
黄 敞 周卓岑 校
谢 鸿 韩承德

内 容 简 介

全书分三部分，共十六章。第一部分论述了微处理机的程序设计和接口技术的一般原理。第二部分论述了几种典型微处理机，其中包括Intel 4004/4040、Intel8008/8080、M6800、PPS-4、PPS-8、IMP4/8/16和PACE等。第三部分介绍几种新型微处理机及专用微系统，如LSI-11、F8、Intel3000、IM6100等。书中引用了大量的图表、实例和参考资料。

本书可作为大专院校有关专业的教学参考书，也可作为从事实际工作的科技人员和管理人员的入门书。

MICROPROCESSORS AND MICROCOMPUTERS
BRANKO SOUČEK
JOHN WILEY and SONS
1976

*

微处理机与微型计算机

〔美〕 布朗柯·索切克 主编
林清波 傅树桂 等译
黄 敞 周卓岑 校
谢 鸿 韩承德

*

国防工业出版社 出版
新华书店北京发行所发行 各地新华书店经售
国防工业出版社印刷厂印装

*

787×1092 1/16 印张25 3/8 591千字
1981年1月第一版 1981年1月第一次印刷 印数：0,001— 8,400册
统一书号：15034·2076 定价：2.60元

前　言

微处理机和微型计算机的出现，使计算机的发展进入了一个新阶段。具有体积小、重量轻、性能/价格比大等优点的微型机，为计算机的应用开辟了广阔的前景。微型机可以装入各种仪器、机械设备，甚至用作家庭生活用具，它已渗透到社会生活的各个领域。为了促进微型机的普及应用，特将本书翻译出版。

全书共分三部分，十六章。第一部分共有五章，论述了数字代码、逻辑系统、微型机结构、微型机的指令系统、微处理机的程序设计、微处理机的接口技术。第二部分共有六章，详细介绍了几种典型微处理机（Intel 4004/4040、8008/8080、M6800、PPS-4、PPS-8、IMP4/8/16和PACE）的有关内容。对于各种微处理机，详细介绍了它们的商品芯片、I/O总线、指令系统和寻址方式等。第三部分共有五章，介绍了新型微处理机及专用微系统。其中有：其软件与PDP-11小型机兼容的LSI-11微型机、F 8微处理机、SMS微控制器、Intel 3000系列双极微型机、其指令系统与PDP-8小型机相同的IM6100微型机等。

本书内容是以大学低年级文化水平为起点的，然后，再系统地、循序渐进地转到先进的程序设计、接口技术和应用技术的各个领域。在本书中，引用了大量的图表、实例和参考资料。本书所涉及的许多内容，都是从实际问题中引出来的。本书所列举的问题和实例，将有助于加强对学生和初学者应用微处理机技术的实际训练。

本书是由辽宁省电子局情报室组织翻译的。具体担任翻译工作的有林清波、傅树桂、王本琦、李亚琴、陈庆宇、何玉表、胡长忠、马刈非、李启凤等同志。参加校对工作的有黄敞、周卓岑、谢鸿、韩承德等同志。在译校过程中，还有许多同志做了不少工作。由于有关部门的领导同志始终给予大力支持，故使本书的翻译出版工作进展比较顺利。在此，对上述同志一并表示感谢。

在译校中，对原书中的个别错误作了改正。由于水平所限，译文仍会存在一些不足之处，恳请读者批评指正。

目　录

第一部分　微处理机的程序设计和接口技术

第一章　数制和数字代码

本章介绍数制和数字代码。内容的重点是计算机技术中应用最广的二进制代码。介绍二进制数的算术运算；说明并比较用于信息编码的各种代码；介绍在微处理机和微型计算机程序设计过程中广泛采用的八进制和十六进制及特别适用于位置转换器的葛莱码；给出ASCII字符系统的有关规定。

1.1　十进制和二进制

1.1.1　定义

我们对计算机的基本要求是，不仅能够表示和存贮数，而且能对所表示的数进行运算。这些数可用不同的数制来表示。人们很早就在实践中采用十进制表示数量。这种数制之所以是以十为单位计数，大概是由于人们有十个手指得到的启示。这种数制由于采用了十个基本数码（对应于十个手指），所以它的基数为10。这些基本数码为0、1、2、3、4、5、6、7、8和9。若将这些数码放在不同的位置或者加权，那么，我们就可以表示大于10的数。10为十进制的基数。人们还可以采用不同的基数 b 构成任一种数制。

总之，一个以固定的正整数 b 为基数的数 $(N)_b$，可用位置记数法表示为：

$$(N)_b=(P_nP_{n-1}\cdots P_1P_0\cdot P_{-1}P_{-2}\cdots)$$

数 b 为这一数制的基数。它是一个正整数，并在整个数制中固定不变。各位置上的数字 P_i 是整数，例如：

$$0\leqslant P_i\leqslant b-1 \qquad i=\cdots,\ -2,\ -1,\ 0,\ 1,\ 2,\ \cdots,\ n$$

数中每一位置的值一般称为它的位置系数或权。例如：

$$\begin{aligned} 346 &= 3\times 100 = 300 \\ &\quad 4\times 10 = 40 \\ &\quad 6\times 1 = 6 \\ &\quad\quad\quad\ 346 \end{aligned}$$

简单的十进制的权是按下列次序排列的：

$$\cdots 10^3 10^2 10^1 10^0\cdot 10^{-1}10^{-2}\cdots$$

总之，在基数为 b 的数制中，其权可从左向右依次排列为：

$$\cdots b^3b^2b^1b^0\cdot b^{-1}b^{-2}\cdots$$

符号“·”称为小数点。小数点右边的数为小数部分，左边的数为整数部分。在十进制中，小数点称为十进制小数点。

计算机可根据任何一种数制来设计。但是，所有现代电子数字计算机都采用二进制

(基数为 2)。为什么要采用这种新的数制呢？其原因是，区别两种状态要比区别十种状态容易。大多数物理量只有两种状态，例如：电灯的开或关；开关的接通或断开；材料的磁化或退磁；电流的正或负；纸带或卡片上穿孔或不穿孔；等等。设计一种只须区别两种状态（对应于二进制的“0”和“1”）的电路，要比设计一种必须区别十种状态(对应于十进制的 0～9）的电路要容易得多，而且也更为可靠。

二进制的基数为 2，其小数点称为二进制小数点。可用的数码是“0”和“1”。所用的权可以从左至右依次排列为：

$$\cdots 2^3 2^2 2^1 2^0 \cdot 2^{-1} 2^{-2} 2^{-3} \cdots$$

利用上述这种二进制权的排列格式（称为权表），可以将二进制数变换成人们更为熟悉的十进制数。例如，我们可以按照权表求出与二进制数 10101 等值的十进制数(见表1.1)。

表 1.1

2^4	2^3	2^2	2^1	2^0	(权表)	
1	0	1	0	1	(二进制数)	位置系数
				1	X	1 = 1
			0		X	2 = 0
		1			X	4 = 4
	0				X	8 = 0
1					X	16 = 16
						十进制数 = 21

上述这种计数过程可以用来指出二进制与十进制之间的异同点。在十进制中，其计数过程是对某特定位数字按 0，1，2，…，8，9 的顺序增加的。当该特定位的数字达到 0（即 10）时，我们就向左边相邻位进 1。由于二进制仅利用两个数码，所以二进制的每一具体位仅经过两次变化就向左边相邻位进 1。

表 1.2

十进制	二进制
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

表 1.2 列出了用二进制数表示的十进制数 0～10。

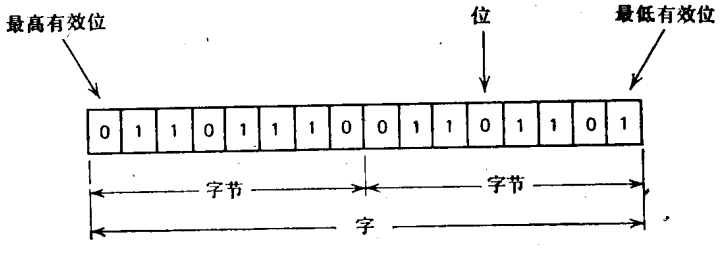


图1.1 位、字节和字

下面介绍一下计算机技术中广泛使用的一些专用名词，如：“位”、“字节”和“字”。

位 一个二进制的数码通常称为 1 位。如数 1010 就称为一个 4 位的二进制数，101就称为一个 3 位的二进制数。该数左端一位称为最高有效位（它具有最大的权)。该数右端一位称为最低有效位（它具有最小的权)。图 1.1 表示了一个 16 位的二进制数。图中指出了

位、字节和字的划分情况。

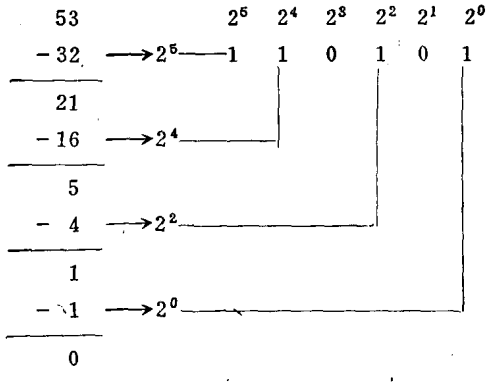
字节 随着计算机和数据设备的发展，出现了部件之间以 8 位为单元的信息交换。这个 8 位单元便被称为一个“字节”。于是许多新型数字计算机和控制机都采用 8 位、16 位、24 位或 32 位（相当于一、二、三、或四个字节）来表示数。图 1.1 所表示的是一个 由两个字节组成的二进制数。

字 计算机是由大量存放二进制信息的存贮单元或寄存器组成的。在一给定计算机中，大多数寄存器都具有相同的字长 n。每个寄存器可用来存放 n 位二进制信息，于是存放于一个寄存器中的信息也就称为一个“字”。图 1.1 表示了 16 位计算机的一个字。

1.1.2 十-二进制转换

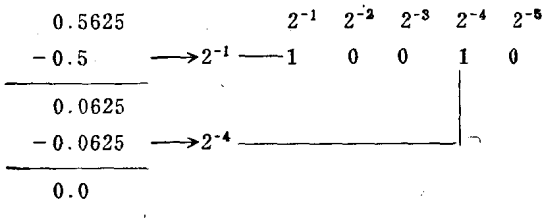
一般来说，一个十进制数可能包括十进制整数部分和十进制小数部分。若分别将每一部分转换成与其等值的二进制数，然后将已转换的这两部分与二进制小数点结合，那么就可以得到一个完整的二进制数。将十进制数转换成与其等值的二进制数的方法通常有两种——相减法和相除法。

相减法 这里，我们参照了参考资料〔4〕的步骤。第一步是，从十进制数中减去 2 的尽可能为最高次的幂，并在部分形成的二进制数的相应权的位置上写“1”。然后，依此类推，将这个过程继续进行下去，一直到十进制数被减为 0 为止。如果在第一次相减之后所剩的差数不够去减 2 的下一次幂，那么就应在部分形成的二进制数的相应权的位置上写“0”。下面所举的是将十进制整数 53 变换成二进制数的例子，其运算过程是：



由上得出：$(53)_{10}=(110101)_2$。

再举一个将十进制小数变换为二进制数的例子：



由上得出：$(0.5625)_{10}=(0.10010)_2$

相除法 如果要将一个十进制整数变换成与其等值的二进制数，则可以用这个数去除若干次 2。该十进制数除以 2 之后，如有余数，就在最低二进制数位上写“1”；如果没有余数，就在最低二进制数位上写“0”。接着将第一次运算结果再除以 2，依此类推，一

直重复下去，直到其结果被减到 0 为止（参见资料 4）。例如：

2	53	余	数	
2	26	⟶	1	
2	13	⟶	0	$(53)_{10}=(110101)_2$
2	6	⟶	1	
2	3	⟶	0	
2	1	⟶	1	
	0	⟶	1	

相乘法 如果要将一个十进制小数变换成与其等值的二进制小数，则可以用这个数去乘若干次 2。如果第一次乘 2 所得的乘积小于 1，则该二进制数的最高有效位就为 0；如果第一次乘 2 所得之乘积大于 1，则该二进制数的最高有效位就为 1。按同样规则，再对第一步得出的乘积的小数部分进行运算，即得到二进制数第二个数位。如此继续进行这一过程，直到达到所需精度为止。例如：

	进位
$0.5625\times 2=1.1250$	⟶ 1
$0.1250\times 2=0.2500$	⟶ 0
$0.250\times 2=0.5$	⟶ 0
$0.5\times 2=1.0$	⟶ 1
$0.0\times 2=0.0$	⟶ 0

由上可得：$(0.5625)_{10}=(.10010)_2$

1.2 二进制数的算术运算

1.2.1 加法

二进制加法与十进制加法一样，只不过不是逢 10 进 1，而是逢 2（即：1 + 1）进 1。例如：

$$\begin{array}{r} 101 = 5_{10} \\ +010 = 2_{10} \\ \hline 111 = 7_{10} \end{array}$$

$$\begin{array}{r} 11 \leftarrow \text{进位} \\ 111 = 7_{10} \\ +101 = 5_{10} \\ \hline 1100 = 12_{10} \end{array}$$

现在看一下第二个例子(111+101)。第一行，1 + 1 = 0，进位写 1。在第二行，1 加上第一行的进位数 1 等于 0，再进上一位。第三行是 1 + 1 = 0 再加上第二行的进位数 1，即 1 + 1 + 1 = 11。其答案是 1100（与其等值的十进制数为 12），它也是相应的 7 + 5 的正确解答。

在数字计算机中，二进制数的加法是由称之为加法器的专用部件实现的。

1.2.2 减法

二进制数可用与十进制减法相同的方法直接相减。设计一种既有加法器又有减法器的机器是完全可能的。但是计算机设计人员并没有这样做。很清楚，如果人们要进行减法运算，只需改变减数的符号然后再进行加法运算即可。问题在于要用适当的方法表示负数。

为了明了计算机中负数是怎样表示的，我们举一个机械寄存器为例，如汽车里程指示器。当寄存器向前转时，它进行加法运算。当寄存器向后转时，就进行减法运算。例如一个5位寄存器向后转时，则依次示出：

0 0 0 0 4
0 0 0 0 3
0 0 0 0 2
0 0 0 0 1
0 0 0 0 0
9 9 9 9 9
9 9 9 9 8
9 9 9 9 7

实际上，数9 9 9 9 7相当于－3。我们可用下列加法验证一下：

$$\begin{array}{r} 0\ 0\ 0\ 0\ 4 \\ +\ 9\ 9\ 9\ 9\ 7 \\ \hline 1\ 0\ 0\ 0\ 0\ 1 \end{array}$$

如果我们略去向左的进位，我们就有效地实现了减法运算。

$$4 - 3 = 1$$

在这个例子中，数9 9 9 9 7称为3的十进制补码。这样，在十进制中，负数可用十进制补码来表示并略去负号。

在数字计算机中，采用同样方法，用二进制数的补码表示负数，从而实现减法运算。

一个数的二进制补码定义为这样一个数，这个数加上原来的数得到的和为1。例如，二进制数010 110 110 110的二进制补码等于101 001 001 010，如下列加法所示：

$$\begin{array}{rcccc} & 010 & 110 & 110 & 110 \\ & 101 & 001 & 001 & 010 \\ \hline 1 & 000 & 000 & 000 & 000 \end{array}$$

要求得一个数的二进制补码，应采取下述两个步骤：

1．先求出反码，方法是将其每一位用其相反的值代替（即0→1，1→0）；

010 110 110 110　　数
101 001 001 001　　数的反码

2．二进制补码等于反码加1。

```
  101  001  001  001   数的反码
+                 1   加1
-------------------
  101  001  001  010   数的补码
```

举一个减法的例子：

```
  7 - 3 = 4                12 - 5 = 7
  0011   3(10)             0101   5(10)
  1100   3 的反码          1010
  1101   3 的补码          1011
+                        +
  0111   7(10)             1100   12(10)
-------                  -------
1 0100   4(10)           1 0111   7(10)
```

1.2.3 乘法

在二进制乘法中，每乘一位，部分乘积就向左移一位。若乘数为 0，则部分乘积也为 0；若乘数为 1，则部分乘积与被乘数相等。举例如下：

```
5 × 3 = 15          5 × 5 = 25          5 × 10 = 50
  101   5(10)         101   5(10)          101   5(10)
   11   3(10)         101   5(10)         1010   10(10)
-----               -----               -----
  101                 101                  000
 101                 000                  101
-----               101                  000
 1111   15(10)      -----                101
                    11001   25(10)       ------
                                         110010   50(10)
```

1.2.4 除法

按照二进制减法和乘法的规则，可以用与十进制除法相同的方法进行二进制除法运算。例如：

```
18 ÷ 2 = 9          10 ÷ 5 = 2          14 ÷ 4 = 3.5
    1001   9(10)          10   2(10)          11.1   3.5(10)
2(10) 10)10010 18(10)  5(10) 101)1010 10(10)  4(10) 100)1110.0 14(10)
   10                     101                   100
  ----                   ----                  ----
   00                     000                   110
   00                     000                   100
  ----                   ----                  ----
   01                      0                    100
   00                                           100
  ----                                          ----
   10                                             0
   10
  ----
    0
```

1.3 八进制、二-十进制(BCD)及葛莱码

1.3.1 八进制

若用二进制表示一个数，则所用数位要比十进制的多得多。例如$(35)_{10}=(100011)_2$。人们在读写多位二进制数时很容易出错。为了简化二进制数的表示法，可以采用八进制。八进制的基数为8，采用的数码为0～7。十进制数0～10在八进制中的表示方法列于表1.3。

表 1.3

十进制	二进制	八进制
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	10
9	1001	11
10	1010	12

因为八进制的基数为$8=2^3$，所以要将二进制数变换成八进制数，必须将二进制数按3位分组。每个3位组可用表1.3所列的等值的八进制数来表示。例如：

110101111001	二进制数
110 101 111 001	按3位分组
6 5 7 1	每组等值的八进制数

因此，$(110\ 101\ 111\ 001)_2=(6571)_8$。

将十进制数变换成等值的八进制数，以及八进制数的算术运算，其原理与二进制相同。

需要说明的是，计算机并不采用八进制进行运算，而是采用二进制进行运算。在某些情况下，人们只是为了简化读、写多位二进制数才使用八进制表示法。

1.3.2 二-十进制(BCD)

显然，十进制是人们最感到方便的数制。因此，许多计算机输入-输出设备的工作方式是：在计算机方面，发送/接收二进制数；在人这方面，接收/发送十进制数。这种代码变换需要增加电子设备或者延长计算时间。采用一种称为二-十进制（BCD）的混合表示法，可将需要的代码变换简化到最简单的程度。

表 1.4

十进制	二进制	二-十进制
0	0	0000
1	01	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001

BCD代码是以10为基数组成的，既和十进制有直接关系，同时又仅采用二进制数码0和1。

BCD表示法是将十进制数中的每位数字，换成表示该数字的4位二进制数。例如，6_{10}将表示为0110。这样，整数0～9的表示法如表1.4所示。

十进制数的BCD表示法是将其十进制数位直接换成BCD中相对应的数。例如：

0110	0011	0100	0111
6	3	4	7

〔例〕 将 BCD 数 0101 0111（十进制数 57）变换成与其等值的二进制数。

权(8421)×10	(8421)×1
0101	0111

我们用 8 + 2 来表示权系数 10，以简化乘法。这样，BCD 数 0101 0111 便等于

$$[(0101)\times(8+2)]+(0111)\times 1$$

上式的二进制加法和乘法为：

$$\begin{array}{rl} 0111 & \times 1 \\ 0101 & \times 2 \\ +0101 & \times 8 \\ \hline 0111001 & \end{array}$$

这就是与十进制数 57 等值的二进制数。因此，可用二进制加法线路将 BCD 代码变换成二进制代码。

常常使用两位 BCD 数字对字母、符号及数进行编码。例如，用两位十进制数 00，01，02，…，09 表示数 0，1，2，…，9。其余的两位数 10～99 可用来表示 A，B，…，Y，Z 以及其它符号（如＋，－，？）。先对字母、符号进行十进制编码，然后对十进制代码进行 BCD 编码，这样就能简单地将某种数制中任何一个数、字母及符号用二进制表示法表示。

表 1.5

十进制	余-3码	余-3码的反码
0	0011	1100
1	0100	1011
2	0101	1010
3	0110	1001
4	0111	1000
5	1000	0111
6	1001	0110
7	1010	0101
8	1011	0100
9	1100	0011

余-3BCD 数 在计算机中有时还采用余-3 BCD 表示法，特别是在电子制表机中，它可用来简化十进制反码的结构。每一位数都是将此位数加 3 并变换成用 4 个二进制位来表示。余-3 码表示的数及其反码列于表 1.5。将所有的 1 换成 0，将所有的 0 换成 1 即可得到反码。

十六进制数 在 BCD 里，由 4 个二进制位组合而成的代码中有 6 个是不能用的，现在用另外 6 个符号来表示它们，这就扩大了自然 BCD 数。我们必须习惯于这种含有十六种符号的计数方法。在这十六种符号中，包含有六种新的数字符号。常见的十六进制计数符号为：

0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F

其中，A～F 表示 1010～1111(十进制数 10～15)的 4 位数组。十六进制数如表 1.6 所示。

1.3.3 葛莱码

二进制代码虽然很适用于计算，但在用于两个相邻二进制编码位置之间转换时可能会出现严重问题。例如，用计算机控制轴的角度位置或连杆直线位置时，就会出现这种情况。二进制编码器的两个相邻位置的代码可能有好几位的“1”和“0”是不相同的，例如，位置 7 和位置 8 的二进制代码为 0111 和 1000。恰好在编码器两段之间的变换点上，编码器

对正确的位置可能产生瞬间混乱。结果，可能产生如 1010，0101，1100 等输出。这些输出与原位置 0111 和新位置 1000 差别很大。数字控制系统将把这种瞬变信息译作位置误差，并产生驱动力以校正这个位置。

采用下述方法可以减少瞬变误差，两个相邻位置的代码仅有一位不同。因为在瞬变时其它位均不改变。所以在变化的瞬间只能读出原位置或某一新位置。这样，错乱点可缩减到原位置或新位置的范围之内，而不象采用二进制代码那样，错乱点出现在编码器的各个位置。

葛莱码是一系列称为反射二进制代码中的一种。它非常适用于位置变换器。葛莱码类似于二进制代码，它们总是具有相同的位长和相同的最高位。葛莱码与二进制代码的主要区别是，葛莱码中的两个相邻的数只有一位是不相同的。现将两种代码数列于表 1.6，做一比较。

表 1.6

十进制	二进制	葛莱码	十六进制
0	0000	0	0
1	0001	01	1
2	0010	11	2
3	0011	10	3
4	0100	110	4
5	0101	111	5
6	0110	101	6
7	0111	100	7
8	1000	1100	8
9	1001	1101	9
10	1010	1111	A
11	1011	1110	B
12	1100	1010	C
13	1101	1011	D
14	1110	1001	E
15	1111	1000	F

1.4 数据格式

计算机是按二进制信息操作进行设计的。而这种二进制信息可以方便地用电子器件表示，并用存贮器存放。二进制信息可以表示定点数、浮点数、二进制编码的字母符号和计算机指令。

1.4.1 定点数

大多数计算机都是用二进制补码进行运算的。这样一来，计算机字可用下述方式用来存贮正数和负数。字的最高有效位为符号位。用 0 表示正数；用 1 表示负数。其余位表示数的数值：若是正数，它就表示数的本身；若是负数，它就表示该数的二进制补码。这样，若计算机的字为 n 位，则用 $n\sim1$ 位表示 0 和 2^{n-1} 之间任一整数的数值。

下面举一个 12 位计算机的例子：

正数：	000	000	000	000	0_{10}
	000	000	000	001	1_{10}
	000	000	000	010	2_{10}
	011	111	111	111	$2^{11}-1=2047_{10}$
负数：	111	111	111	111	-1_{10}
	111	111	111	110	-2_{10}
	100	000	000	001	$-(2^{11}-1)=-2047_{10}$
	100	000	000	000	$-2^{11}=-2048_{10}$

因此，字长为 12 位的计算机可以直接表示从-2048_{10}到$+2047_{10}$之间的数。要表示更大的数，则可以采用两个计算机字，或采用浮点形式来表示。

1.4.2 浮点数

在浮点记数法中，数字分为两个部分，即尾数（数码部分）和阶（对某一基数）。例如，在十进制中，数 15 可以写成下列形式：

尾数	阶
0.15	$\times 10^{2}$
1.5	$\times 10^{1}$
15.0	$\times 10^{0}$
150.0	$\times 10^{-1}$
1500.0	$\times 10^{-2}$

计算机的浮点表示法采用和上例类似的方法。但是，由于计算机使用二进制信息进行运算，所以尾数和阶均用二进制数表示。由于尾数和阶可以为正也可以为负，因此保留两位作为符号位。图 1.2 所示为 36 位字长的数的定点和浮点表示法。对于小型计算机，可用一个以上的字来表示浮点数。

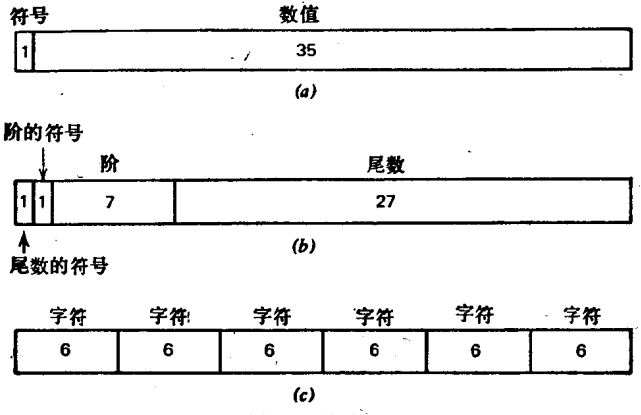


图1.2 格式

（a）定点；（b）浮点；（c）字母数字字符。

1.4.3 字母数字字符

存在这样一种情况，这就是需要用计算机来表示字母和标点符号的字符。这时可用二

进制代码表示这些字符。因为只有26个字母和10个数码，因此，如果每一字符用6位表示，则除了表示64种字符外还可以表示许多专用符号。最常用的编码为ASCII编码(ASCII是美国信息交换标准编码的缩写)。ASCII编码可以使用6位、7位或8位。表1.7列出了6-位及8-位ASCII编码。分为3位一组，每组表示一个八进制数。表1.8为7-位ASCII编码，它也是一种BCD编码。如图1.2(*c*)所示，一个36位计算机字中，含有六组6-位字符。

表1.7 ASCII字符系统

字符	8-位八进制	6-位八进制	字符	8-位八进制	6-位八进制
A	301	01	!	241	41
B	302	02	"	242	42
C	303	03	#	243	43
D	304	04	$	244	44
E	305	05	%	245	45
F	306	06	&	246	46
G	307	07	'	247	47
H	310	10	(	250	50
I	311	11	)	251	51
J	312	12	*	252	52
K	313	13	+	253	53
L	314	14	,	254	54
M	315	15	-	255	55
N	316	16	.	256	56
O	317	17	/	257	57
P	320	20	:	272	72
Q	321	21	;	273	73
R	322	22	<	274	74
S	323	23	=	275	75
T	324	24	>	276	76
U	325	25	?	277	77
V	326	26	@	300	
W	327	27	[	333	33
X	330	30	\	334	34
Y	331	31	]	335	35
Z	332	32	↑	336	36
0	260	60	←	337	37
1	261	61	Leader/Trailer	200	
2	262	62	LINE FEED	212	
3	263	63	Carriage RETURN	215	
4	264	64	SPACE	240	40
5	265	65	RUBOUT	377	
6	266	66	Blank	000	
7	267	67	BELL	207	
8	270	70	TAB	211	
9	271	71	FORM	214	