微型计算机硬件软件及其应用

（修订版）

周明德 编著

清华大学出版社

目　录

第一章 概 述

第一节 引 言

随着计算机应用的推广和普及，随着大规模集成电路技术的飞速发展，70年代初诞生了一代新型的电子计算机——微型计算机（Microcomputer）。它利用大规模集成电路技术把计算机的中央处理单元(CPU——Central Processing Unit)即计算机的运算器和控制器集成在一个芯片上称为微处理器（Microprocessor)。在1975、1976年先后生产了三大8位微处理器系列，即Intel公司的8080、8085，Motorola公司的M6800和Zilog公司的Z80；在1978至1980年先后生产8086、Z8000和M68000等16位微处理器；近年又推出了80386和68020等32位微处理器，片上的集成度已超过20万个晶体管。同样利用大规模集成电路技术制造了容量相当大的内存储器（Memory)芯片，如16K×4位的静态存储器和64K×1位、256K×1位的动态读写存储器（或称为随机存取存储器）RAM（Random Access Memory）和32K×8位的只读存储器ROM（Read Only Memory)；同时又把各种通用的或专用的，可编程序的接口电路（与外部设备相连接的电路）集成在一个片子上。这样，把CPU配上一定容量的RAM，ROM以及接口电路（例如并行接口电路PIO，串行接口电路SIO等）和必要的外设（通常包括CRT终端，打印机，软盘或硬盘（温盘）驱动器等等）就形成了一个微型计算机。

在有些专用的场合，还把CPU，一定容量的RAM和ROM以及输入输出接口电路集成在一个芯片上，形成单片计算机（Single Ghip Computer)。或把CPU，RAM和ROM，输入输出接口装在一块印刷电路板上，成为单板计算机（Single Board Computer）。

总之，微型计算机以利用大规模和超大规模集成电路技术为特征，大大缩小了计算机的体积，同时也大大降低了成本。例如，国内推广的微型机优选系列的低档机——紫金Ⅱ（与AppleⅡ兼容），具有64K字节内存，两个$5^1/_4$″软盘驱动器，彩色显示器以及80列打印机，售价约为6000元。高档机——0520CH（与IBM-PC/XT兼容）具有512K字节RAM，固化的二级汉字字库，高分辨显示器（可显示28行40列汉字），20兆温盘，两个$5^1/_4$″软盘驱动器，24针打印机，售价约为32000元。而以Z80为CPU，具有4—8K字节RAM的单板计算机约为500元。因而发展极为迅速，应用极为广泛，已经深入到工农业生产、国防、文教、科研以及日常生活等各个领域。

在我国，微型计算机的科研和生产已有相当发展，应用也已开始深入到各个领域。

本书从应用的角度，较全面系统地分析和论述了微型计算机系统的硬件和软件。硬件方面着重于分析CPU的结构和时序；分析半导体存储器的选用及与CPU的接口；各种外设与CPU的接口技术以及各种并行的、串行的接口片子，定时器与计数器电路；中断功能和D/A、A/D转换片子及与CPU的接口等。在软件方面，重点介绍了汇编语言的程序设计；介绍了汇编程序，监控和调试程序；8位机的典型操作系统CP/M。在书中介绍了大量的各种应用程序的例子。所有这些都是从工程技术人员如何利用微型计算机来解决各种工程问题，如何编应用程序的角度来分析、论述问题的。

第二节　计算机中的数和编码系统

一、计算机中的数制

计算机最早是做为一种计算工具出现的，所以它的最基本的功能是对数进行加工和处理。数在机器中是以器件的物理状态来表示的，一个具有两种不同的稳定状态且能相互转换的器件，就可以用来表示一位二进制数。所以，二进制数的表示是最简单而且可靠的；另外，二进制的运算规则也是最简单的。因此，目前在计算机中，数几乎全是用二进制表示的。

（一）二进制数

一个二进制数，具有以下两个基本特点：

1．具有两个不同的数字符号，即0和1；

2．逢二进位。

由于是逢二进位的，所以同一个数字符号在不同的数位所表示的值是不同的。例如

111.11

小数点左边第一位的“1”代表的值就是它本身；小数点左边第二位的“1”，是由第一位逢二进上来的，所以它的值为1×2^1；则左边第三位的“1”的值为1×2^2；小数点右面第一位的“1”代表1×2^{-1}；右面第二位的“1”代表1×2^{-2}……。

可见，每一个数位有一个基值与之相对应，这个基值称为权。小数点左面各数位的权是2的正次幂，小数点右面各数位的权是2的负次幂。

一个二进制数的值可以用它的按权展开式来表示，即

$$(111.11)_2=1\times2^2+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}=(7.75)_{10}$$

$$(1011.101)_2=1\times2^3+1\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-3}=(11.625)_{10}$$

于是，一个任意的二进制数可以表示为

$$\begin{aligned}(B)_2&=B_{n-1}\times2^{n-1}+B_{n-2}\times2^{n-2}+\cdots+B_1\times2^1+B_0\times2^0+B_{-1}\times2^{-1}\\&\quad+B_{-2}\times2^{-2}+\cdots+B_{-m}\times2^{-m}\\&=\sum_{i=-m}^{n-1}B_i\times2^i\end{aligned}$$

其中n为整数部分的位数，m为小数部分的位数；B_i的值为0或1取决于一个具体的数。

（二）十六进制数

目前，大部分微型机的字长是4的整数倍，所以广泛地采用十六进制数来表示。一个十六进制数的特点为：

1．具有十六个数字符号，采用0—9和A—F。这16个数字符号与十进制数和二进制数之间的关系如表1-1所示。

2．逢16进位

由于是逢16进位，所以同一个数字符号，在不同的数位所代表的值是不同的，即每一个数位有一个权与之相对应。小数点左边各数位的权是16的正次幂，小数点右边各数位的权是16的负次幂。一个16进制数的值可以用它的按权展开式来表示，即

表 1-1　二进制、十进制、十六进制数码对照表

十　进　制	十 六 进 制	二　进　制
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000

$$(32)_{16}=3\times16^{1}+2\times16^{0}=(50)_{10}$$

$$(FF)_{16}=15\times16^{1}+15\times16^{0}=(255)_{10}$$

$$(3AB.11)_{16}=3\times16^{2}+10\times16^{1}+11\times16^{0}+1\times16^{-1}+1\times16^{-2}=(939.0664)_{10}$$

于是，一个任意的16进制数D可以表示为

$$(D)_{16}=D_{n-1}\times16^{n-1}+D_{n-2}\times16^{n-2}+\cdots+D_{1}\times16^{-1}+D_{0}\times16^{0}$$
$$+D_{-1}\times16^{-1}+D_{-2}\times16^{-2}+\cdots+D_{-m}\times16^{-m}$$
$$=\sum_{i=-m}^{n-1}D_{i}\times16^{i}$$

其中，n是整数部分的位数，m是小数部分的位数；D_i的值在范围0—9和A—F中。

但是，在机器中，数并不是用16进制表示的，由于一开始提到的理由，在机器中数仍是用二进制表示的。由于二进制和十六进制之间存在着一种特殊关系，即$2^4=16$。

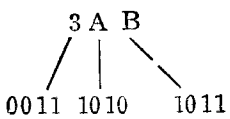
于是，一位十六进制数可以用四位二进制数表示，它们之间存在着直接的而又是唯一的对应关系，如表1-1所示。

因此，二进制数和十六进制数之间的转换是十分简捷而又方便的。

1．16进制转换为二进制

不论是16进制的整数或小数，只要把每一位16进制的数用相应的四位二进制数代替，就可以转换为二进制数。

例：$(3AB)_{16}$ 可转换为



∴ $(3AB)_{16}=(0011\ 1010\ 1011)_2=(11\ 1010\ 1011)_2$

$(0.7A53)_{16}$可转换为

$$\begin{array}{cccc} 0.7 & A & 5 & 3 \\ \diagup & | & | & \diagdown \\ 0111 & 1010 & 0101 & 0011 \end{array}$$

∴ $(0.7A53)_{16}=(0.0111\ 101001010011)_2$

2．二进制转换为16进制

二进制的整数部分由小数点向左，每四位一分，最后不足四位的前面补0；小数部分由小数点向右，每四位一分，最后不足四位的后面补0。然后把每四位二进制数用相应的16进制数代替，即可转换为16进制数。

例：$(11011\ 1110\ 0011.1001\ 01111)_2$可转换为

$$\begin{array}{cccccccc} 0001, & 1011, & 1110, & 0011. & 1001 & 0111 & 1000 \\ | & | & | & | & | & | & | \\ 1 & B & E & 3 & 9 & 7 & 8 \end{array}$$

∴ $(11011\ 1110\ 0011.1001\ 01111)_2=(1BE3.978)_{16}$

总之，数在机器中是用二进制表示的。但是，一个二进制数书写起来太长，容易出错。而目前大部分微型机的字长是四位、八位、十六位或三十二位的，都是四的整数倍，故在书写时可用十六进制表示。一个字节（八位）可用两位十六进制数表示，两个字节（十六位）可用四位十六进制数表示等，书写方便且不容易出错。

二、二进制编码

如上所述，在计算机中，数是用二进制表示的。而计算机又应能识别和处理各种字符，如大小写的英文字母，标点符号，运算号等等。这些字符应如何表示呢？由于计算机中的基本物理器件是具有两个状态的器件，所以各种字符只能用若干位的二进制码的组合来表示，这就是二进制编码。

（一）二进制编码的十进制数

因为二进制数实现容易、可靠，二进制的运算规律十分简单，所以，在计算机中采用二进制。但是，二进制数不直观，于是在计算机的输入和输出时通常还是采用十进制数表示。不过这样的十进制数要用二进制编码来表示。

一位十进制数用四位二进制编码来表示。表示的方法极多，较常用的是8421 BCD码，表1-2列出了一部分编码关系。

8421 BCD码有十个不同的数字符号，且它是逢“十”进位的，所以，它是十进制数；但它的每一位是用四位二进制编码来表示的，因此，称为二进制编码的十进制数（BCD-Binary Coded Decimal）。

BCD码是比较直观的。

例：$(0100\ 1001\ 0111\ 1000.0001\ 0100\ 1001)_{BCD}$

可以很方便地认出为：

4978.149

所以，只要熟悉了BCD的十位编码，立即可以很容易地实现十进制与BCD码之间的转换。

表 1-2 BCD编码表

十进制数	8421 BCD码	十进制数	8421 BCD码
0	0000	8	1000
1	0001	9	1001
2	0010	10	0001 0000
3	0011	11	0001 0001
4	0100	12	0001 0010
5	0101	13	0001 0011
6	0110	14	0001 0100
7	0111	15	0001 0101

但是，BCD码与二进制之间的转换是不直接的，要先经过十进制。即：BCD码先转换为十进制码，然后再转换为二进制；反之亦然。

（二）字母与字符的编码

如上所述，字母和各种字符也必须按特定的规则用二进制编码才能在机中表示。编码也可以有各种方式——即规定。目前在微型机中最普遍的是采用ASCII（American Standard Code for Information Interchange美国标准信息交换码）码，编码表见附录1。

它是用七位二进制编码，故可表示128个字符，其中包括数码(0—9)，以及英文字母等可打印的字符。从表中可看到，数码0—9相应用0110000—0111001来表示。因微型机通常字长为8位，所以bit7通常用作奇偶校验位，但在机中表示时常认其为零，故用一个字长（即一个字节）来表示一个ASCII字符。于是，0—9的ASCII码为30H—39H；大写字母A—Z的ASCII码为41H—5AH。

三、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。

（一）二进制加法

二进制加法的规则为：

① 0 + 0 = 0

② 0 + 1 = 1 + 0 = 1

③ 1 + 1 = 0　进位1

④ 1 + 1 + 1 = 1　进位1

若有两数1101和1011相加，则加法过程如下：

```
进位      1 1 1
被加数    1 1 0 1
加数   +  1 0 1 1
          1 1 0 0 0
```

可见，两个二进制数相加时，每一位有三个数——即相加的两个数和低位的进位参加运算，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  数  + 0 0 0 0 1 1 1 1
和      1 1 0 0 0 1 0 0
```

（二）二进制减法

二进制减法的运算规则为：

① 0 − 0 = 0

② 1 − 1 = 0

③ 1 − 0 = 1

④ 0 − 1 = 1　有借位

例：11000100 − 001100101，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——本位的被减数、减数和低位的借位参加运算。为便于计算，式中列出了低位向高位的借位，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
```

（三）二进制乘法

二进制乘法的运算规则为：

① 0 × 0 = 0

② 0 × 1 = 0

③ 1 × 0 = 0

④ 1 × 1 = 1

规则是十分简单的：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
              1 1 1 1
            0 0 0 0
          1 1 1 1
        1 1 1 1
1 1 0 0 0 0 1 1
```

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的乘数位对齐。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，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
1 1 0 1	1 1 1 1	0 0 0 0
① 乘数最低位为1，把被乘数加至部分积上，		+ 1 1 1 1
		1 1 1 1
然后把被乘数左移	1 1 1 1 0	
② 乘数为0，不加被乘数，		
被乘数左移	1 1 1 1 0 0	
③ 乘数为1，加被乘数（已左移后的）		1 1 1 1
		1 1 1 1 0 0
被乘数左移	1 1 1 1 0 0 0	1 0 0 1 0 1 1
④ 乘数为1，加被乘数		1 1 1 1 0 0 0
得乘积		1 1 0 0 0 0 1 1

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
1 1 0 1	1 1 1 1	0 0 0 0
① 乘数最低位为1，加被乘数		+ 1 1 1 1
		1 1 1 1
部分积右移		0 1 1 1 | 1
② 乘数为0，不加被乘数		
部分积右移		0 0 1 1 | 1 1
③ 乘数为1，加被乘数		+ 1 1 1 1 |
		1 0 0 1 0 | 1 1
部分积右移		1 0 0 1 | 0 1 1
④ 乘数为1，加被乘数		+ 1 1 1 1 |
		1 1 0 0 0 | 0 1 1
部分积右移		1 1 0 0 | 0 0 1 1

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

（四）二进制除法

除法是乘法的逆运算。与十进制的类似，从被除数的最高有效位 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……

```
          0 0 0 1 1 1
      ┌──────────────
1 0 1 )1 0 0 0 1 1
        1 0 1 ↓ 1
      ──────────
          0 1 1 1 |
            1 0 1 ↓
      ──────────────
              1 0 1
              1 0 1
      ──────────────
                  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：

```
                0 0 0 1 1 0 1
          ┌──────────────────
除数  1 1 0 )1 0 0 1 1 1 0      被除数
              1 1 0 ↓ | |
            ──────────
                1 1 1 | |
                1 1 0 ↓ ↓
            ──────────────
                    1 1 0
                    1 1 0
            ──────────────
                        0
```

四、带符号数的表示法

（一）机器数与真值

上面提到的二进制数，没有提到符号问题，故是一种无符号数的表示。但是在机器中，数显然会有正、有负，那么符号是怎么表示的呢？通常一个数的最高位为符号位，即若是字长为 8 位，则D_7为符号位，D_6—D_0为数字位。符号位用 0 表示正，用 1 表示负。如

$$X = (01011011)_2 = +91$$

而

$$X = (11011011)_2 = -91$$

这样连同一个符号位在一起作为一个数，就称为机器数；而它的数值称为机器数的真值。

为了运算方便(把减法变为加法)，在机器中负数有三种表示法——原码、反码和补码。

（二）原码

按上所述，正数的符号位用 0 表示，负数的符号位用 1 表示。这种表示法就称为原码。如：

$X = +105$，　$[X]_{原} = 0\ 1101001$

$X = -105$，　$[X]_{原} = 1\ 1101001$

符号位　　数值

其中，最高位为符号位，后面 7 位（在字长为 8 位时，若字长为16位，则后面的15位）是数值。在原码表示时，+105和－105它们的数值位相同，而符号位不同。

原码表示简单易懂，而且与真值的转换方便。但若是两个异号数相加（或两个同号数相减），就要做减法。为了把上述运算转换为加法运算就引进了反码和补码。

（三）反码

正数的反码表示与原码相同，最高位为符号位，用“0”表示正，其余位为数值位。如：

$[(+4)]_{反} = 0\ 0000100$

符号位　二进制数值

$[(+31)]_{反} = 0\ 0011111$

符号位　二进制数值

$[(+127)]_{反} = 0\ 1111111$

符号位　二进制数值

而负数的反码表示为它的正数的按位取反（连符号位）。如：

$[+4]_{反} = 0\ 0000100$

$[-4]_{反} = 1\ 1111011$

$[+31]_{反} = 0\ 0011111$

$[-31]_{反} = 1\ 1100000$

$[+127]_{反} = 0\ 1111111$

$[-127]_{反} = 1\ 0000000$

$[+0]_{反} = 0\ 0000000$

$[-0]_{反} = 1\ 1111111$

负数的反码表示与原码表示有很大的区别，最高位仍为符号位，负仍用“1”表示；但数值位就不同了。例：

$[-127]_{反} = 1\ 0000000$

符号位　的反=数值

表 1-3 数 的 表 示 法

二进制数码表示	无符号二进制数	原　码	补　码	反　码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
01111100	124	+124	+124	+124
01111101	125	+125	+125	+125
01111110	126	+126	+126	+126
01111111	127	+127	+127	+127
10000000	128	−0	−128	−127
10000001	129	−1	−127	−126
10000010	130	−2	−126	−125
⋮	⋮	⋮	⋮	⋮
11111100	252	−124	−4	−3
11111101	253	−125	−3	−2
11111110	254	−126	−2	−1
11111111	255	−127	−1	−0

这是要十分注意的。

8位二进制数的反码表示如表1-3所示。

它有以下特点：

1. “0”有两种表示法。

2. 8位二进制反码所能表示的数值范围为+127——-127。

3. 当一个带符号数由反码表示时，最高位为符号位。当符号位为0（即正数）时，后面的七位为数值部分；但当符号位为1（即负数）时，一定要注意后面几位表示的不是此负数的数值，一定要把它们按位取反，才表示它的二进制值。例如一个反码表示的数

$$10010100$$

它是一个负数，它不等于$(-20)_{10}$，而等于

$$-1101011=-(1\times2^6+1\times2^5+1\times2^3+1\times2^1+1)$$
$$=-(64+32+8+3)=(-107)_{10}$$

（四）补码

正数的补码表示与原码相同，即最高位为符号位，用“0”表示正，其余位为数值位。如：

$$[+4]_{补}=\underbrace{0}_{\text{符号位}}\ \underbrace{0000100}_{\text{数值位}}$$

$$[+31]_{补}=\underbrace{0}_{\text{符号位}}\ \underbrace{0011111}_{\text{数值位}}$$

$$[+127]_{补}=\underbrace{0}_{\text{符号位}}\ \underbrace{1111111}_{\text{数值位}}$$

而负数的补码表示即为它的反码，且在最后位（即最低位）加1所形成。如：

$$[+4]_{原}=00000100$$
$$[-4]_{反}=11111011$$
$$[-4]_{补}=11111100$$
$$[+31]_{原}=00011111$$
$$[-31]_{反}=11100000$$
$$[-31]_{补}=11100001$$
$$[+127]_{原}=01111111$$
$$[-127]_{反}=10000000$$
$$[-127]_{补}=10000001$$
$$[+0]_{原}=00000000$$
$$[-0]_{反}=11111111$$
$$[-0]_{补}=00000000$$

8位带符号位的补码表示也列在表1-3中。它有以下特点：

1. $[+0]_{补}=[-0]_{补}=00000000$

2. 8位二进制补码所能表示的数值为

$$+127——-128$$