第一章 概 述

第一节 引 言

微处理器(Microprocessor)和微型计算机(Microcomputer)自 70 年代初崛起以来,发展极为迅猛。在短短的十年里已经经历了四代。应用的发展也极为迅速,已经渗透到各个技术领域,渗透到文化、教育领域,渗透到家庭、日常生活的各个领域。

十多年来微型计算机的主要发展趋势是两大方面。

1. 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。拿 Intel 公司的产品来说,1971 年的 4004,集成度为 2500 个/片;1976 年的 8085,集成度为 9000 个/片;1978 年的 8086,集成度为 29000 个/片;1985 年的 80386,集成度为 260000 个/片。

现在 16 位、32 位的微处理器已经大量出现。

半导体存储器的集成度也越来越高,64K 位的早已有商品出售,256K 位的也已经出现。

软磁盘的存储密度日益提高,Winchester 硬磁盘(温盘)的研制成功,为微型机系统提供了一种价格低廉而存储容量很大的外存储设备,大大扩大了微型机系统的功能。

各种微型机的操作系统,如 CP/M、MP/M、CP/M86、MS-DOS、Unix(以及各种变型)、P 系统等等,以及在各种操作系统支持下的大量高级语言,象雨后春笋般涌现,从而大大丰富了微型计算机的系统软件。

为各种微型计算机配制硬件配件、选件,特别是编制应用软件,出售软件包的公司也大量涌现,成为一种新的行业。现在,没有应用软件支持的微型机已成为裸机,大大影响销售量。

总之,微型计算机系统的性能已经赶上甚至超过了 70 年代小型机的水平。

2. 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。例如,利用家用电视机及录音机,能运行 BASIC 语言的机器,价格在 $100 以下。

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、甚至深入家庭、形成个人计算机(Personal Computer)的重要条件。

目前,各种微处理器片子的年产量为数亿片;各种微型计算机的年产量和销售量为数千万台。微型计算机的普及应用方兴未艾,正以越来越高的势头飞速发展。

微型计算机的普及应用,已经引起了各种科学技术领域的深刻变革,甚至引起了生活领域的变革。越来越多的人在关心在议论第四次工业革命的到来。

IBM 公司生产的个人计算机简称 IBM PC,是 1981 年下半年推出来的,但是,由于它的性能价格比较好;也由于 IBM 公司在计算机行业中的地位,赢得了用户的信任;也由于有上千家公司围绕 IBM PC 做硬件的配件、选件、扩充件,配制各种系统软件和语言,出售各种软件包;也由于 IBM 公司计划在微型机方面形成系列,考虑到软件的兼容、标准化与系列化,以及与大型机在软件上的兼容性等等。因而 IBM PC 发展十分迅速,在 83 年大约销售了 40 万台,84 年生产了约 400 万台。现在美国 IBM PC(包括 PC/XT)装机台数已超过了 1000

万台。

我国电子工业部选定与 IBM PC 兼容的长城 0520 机系列，作为我国准 16 位微机的重点机型，有较强的汉字处理功能，能连成网络。在研制、生产、应用服务等方面形成一个完整的体系，近几年来在国内得到了大量的推广及普及，到 1987 年底已装机 25 万台。

16 位微机已形成国内的主流机型，在今后几年仍会得到广泛的应用。而且，今后几年内会大量普及推广的 32 位高档微机，主流是以 Intel 的 80386 为 CPU 的，它与 IBM PC／XT 等 16 位微机是向上兼容的。为了适应我国微机应用的需要，我们向大家奉献这本教材。

第二节　计算机中的数和编码系统

一、计算机中的数制

计算机最早是作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，这个基值就称为权。

一个二进制数的权，小数点左面的是 2 的正次幂；小数点右面的是 2 的负次幂。

一个二进制数的值，就可以用它的按权展开式来表示。即

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

$$(1011.101)_2 = 1\times 2^3 + 1\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-3} = (11.625)_{10}$$

于是，一个任意的二进制数可以表示为

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

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 16 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。

表 1-1

十进制	十六进制	二进制	十进制	十六进制	二进制
0	0	0000	9	9	1001
1	1	0001	10	A	1010
2	2	0010	11	B	1011
3	3	0011	12	C	1100
4	4	0100	13	D	1101
5	5	0101	14	E	1110
6	6	0110	15	F	1111
7	7	0111	16	10	10000
8	8	1000			

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。小数点左边的权是 16 的正次幂，小数点右边的权是 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。

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

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

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

$$= (939.0664)_{10}$$

于是一个任意的十六进制数 D 可以表示为：

$$(D)_{16} = D_{n-1} \times 16^{n-1} + D_{n-2} \times 16^{n-2} + \cdots + D_1 \times 16 + D_0 \times 16^0 + D_{-1} \times 16^{-1} + D_{-2} \times 16^{-2} + \cdots + D_{-m} \times 16^{-m}$$

$$= \sum_{i=-m}^{n-1} D_i \times 16^i$$

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，数并不是用十六进制表示的，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. 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

$$\begin{array}{ccc} 3 & A & B \\ | & | & | \\ 0011 & 1010 & 1011 \end{array}$$

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

$(0.7A53)_{16}$ 可转换为

```
  7     A     5     3
  |     |     |     |
0111  1010  0101  0011
```

∴ $(0.7A53)_{16}=(0.0111\ 1010\ 0101\ 0011)_2$

2. 二进制转换为十六进制

二进制的整数部分由小数点向左，每四位一分，最前面不足四位的在前面补 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十六进制数代替，即可转换为十六进制数。

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

```
0001,  1011,  1110,  0011.  1001  0111  1000
  |      |      |      |      |     |     |
  1      B      E      3      9     7     8
```

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

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，且容易出错。而且目前大部分微型机的字长是四位，八位的，十六位或三十二位的，都是四的整数倍，我们在书写时用十六进制来表示。一个字节(八位)就可以用两位十六进制数表示，两个字节(十六位)可以用四位十六进制表示等，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，运算符号等等，这些又如何表示呢？由于计算机中的基本物理器件是具有两个状态的器件，所以各种字符又只能用若干位的二进制码的组合来表示，这就称为二进制编码。

(一) 二进制编码的十进制数

因二进制数实现容易、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，表示的方法可以极多，较常用的是 8421 BCD 码，表 1-2 列出了一部分编码关系。

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

8421BCD 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

即，只要熟悉了BCD的十个编码，立即可以很容易地实现十进制与BCD码之间的转换。

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，反之也然。

(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，它是相应用0110000～0111001来表示的。因微型机通常字长为8位，所以通常bit7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) 汉字的编码

当计算机在我国应用时，特别是把计算机用于管理等事务处理领域时，就要求计算机能够输入、处理和输出汉字。显然汉字在计算机中，也只能用若干位的二进制编码来表示。但是用8位编码来表示汉字是远远不够的，那么要用多少位来表示汉字呢？这首先取决于一个计算机中要能输入、存储、处理的汉字的个数。国家根据汉字的常用程序定出了一级和二级汉字字符集，并规定了编码，这就是中华人民共和国国家标准信息交换用汉字编码，GB2312-80中汉字的编码即国标码。该标准编码字符集共收录汉字和图形符号7445个，其中包括:

1. 一般符号202个，包括间隔符、标点、运算符、单位符号和制表符等。

2. 序号60个。它们是1.～20.(20个)，(1)～(20)(20个)，①～⑩(10个)和(一)～(十)(10个)。

3. 数字22个。它们是0～9和Ⅰ～Ⅻ。

4. 英文字母52个，大、小写各26个。

5. 日文假名169个，其中平假名83个，片假名86个。

6. 希腊字母48个，其中大、小写各24个。

7. 俄文字母66个，其中大、小写各33个。

8. 汉语拼音符号26个。

9. 汉语注音字母37个。

10. 汉字6763个。这些汉字分为两级，第一级汉字3755个，第二级汉字3008个。

这个字符集中的任何一个图形、符号及汉字都是用两个7位的字节表示(在计算机中当然用两个8位字节，每个字节的最高位为0来表示)。国标码的每一个字节的定义域在21H到7EH。如啊字国标编码为30H，21H。即为00110000，00100001这两个字节。

国标码中，汉字的排列顺序为: 一级汉字按汉语拼音字母顺序排列，同音字母以笔划顺序为序; 二级汉字按部首顺序排列。

为了使汉字的编码与常用的 ASCII 码相区别，在机器中，汉字是以内码形式存储和传输的。一种机器常用若干种汉字输入方式(输入码)，但其内码是统一的。通常就是用汉字的国标码的两字节的最高位都置"1"形成的。如汉字"啊"的国标码为 00110000,00100001，则它的内码为 10110000,10100001 即 B0H,A1H。

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，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 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

(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，每一位有三个数参加运算：本位的被减数和减数，以及低位来的借位。为了便于计算，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 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 $0\times0=0$

② $0\times1=0$

③ $1\times0=0$

④ $1\times1=1$

这是十分简单的，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，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
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
然后把被乘数左移	1 1 1 1 0	1 1 1 1
② 乘数为 0，不加被乘数，		
被乘数左移	1 1 1 1 0 0	
		1 1 1 1
③ 乘数为 1，加被乘数(已左移后的)		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
部分积右移		1 1 1 1
② 乘数为 0，不加被乘数		0 1 1 1 | 1
部分积右移		
③ 乘数为 1，加被乘数		0 0 1 1 | 1 1
部分积右移		+ 1 1 1 1
④ 乘数为 1，加被乘数		1 0 0 1 0 | 1 1
部分积右移		1 0 0 1 | 0 1 1
		+ 1 1 1 1
		1 1 0 0 0 | 0 1 1
		1 1 0 0 | 0 0 1 1

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。

(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
    ---------------
101 ) 1 0 0 0 1 1
        1 0 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  110 ) 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，$D_6 \sim D_0$ 为数字位。符号位用 0 表示正，用 1 表示负。如

$$X = (0101\ 1011)_2 = +91$$

而

$$X = (1101\ 1011)_2 = -91$$

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。

$$X = +105,\quad [X]_{原} = 0\ \ 1101001$$

$$X = -105,\quad [X]_{原} = 1\ \ 1101001$$

符号位　数值

其中，最高位为符号位，后面 7 位是数值（在字长为 8 位时；若字长为 16 位，则后面的 15 位）。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。为了把减法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，即为它的正数的按位取反（连符号位）而形成的。

$[+4]_{反} = 0$　　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就有很大的区别：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

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。

表 1–3

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

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

$$10010100$$

这是一个负数,它不等于$(-20)_{10}$,而等于

$$\begin{aligned} -1101011 &= -(1\times 2^6 + 1\times 2^5 + 1\times 2^3 + 1\times 2^1 + 1) \\ &= -(64+32+8+3) = (-107)_{10} \end{aligned}$$

(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:

$[+4]_{补} = 0\,0000100$　　0 为符号位,0000100 为数值位

$[+31]_{补} = 0\,0011111$　　0 为符号位,0011111 为数值位

$[+127]_{补} = 0\,1111111$　　0 为符号位,1111111 为数值位

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

$$+127 \sim -128$$

3. 一个用补码表示的二进制数,最高位为符号位,当符号位为"0"(即正数)时,其余七位即为此数的二进制值;但若符号位为"1"(即负数)时,其余几位不是此数的二进制值,把它们按位取反,且在最低位加 1,才是它的二进制值。例:$[X]_{补} = 10010100$

它不等于 $(-20)_{10}$,它的数值为由 0010100 按位取反得 1101011,然后再加 1 为 1101100。

即
$$X = -1101100 = -(1 \times 2^6 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2)$$
$$= -(64 + 32 + 8 + 4) = (-108)_{10}$$

当负数采用补码表示时,就可以把减法转换为加法。例如

$$X = 64-10 = 64+(-10)$$
$$[X]_{补} = [64]_{补} + [-10]_{补}$$
$$+64 = 01000000$$
$$+10 = 00001010$$
$$[-10]_{补} = 11110110$$

于是

$$\begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ -\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \\ \hline 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \end{array} \qquad \begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ +\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0 \\ \hline \boxed{1}0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \end{array}$$

自然丢失

由于在字长为 8 位的机器中,从第 7 位的进位是自然丢失的,故做减法与补码相加的结果是相同的。又例如

$$34-68 = 34+(-68)$$
$$[+34]_{补} = 00100010$$
$$[+68]_{补} = 01000100$$
$$[-68]_{补} = 10111100$$

则

$$\begin{array}{r} 0\,0\,1\,0\,0\,0\,1\,0 \\ +\,1\,0\,1\,1\,1\,1\,0\,0 \\ \hline 1\,1\,0\,1\,1\,1\,1\,0 \end{array}$$

即为和的补码，符号位为 1，肯定和为负数，数值部分应由后七位按位取反再加 1，即为 0100010

$$\therefore \quad 和 = (-34)_{10}$$

在日常生活中，有不少“补”数的例子。就以对表来说：若标准时间为 6 点整，而现在有一只表为 10 点，要拨到 6 点，可以有两种拨法：

① $10-4=6$　倒拨

② $10+8=6$　顺拨

所以，在这里 10 加 8 与减 4[或加(−4)]是相同的。这当然是有条件的。这是因为在钟表中

$$10+8=12+6$$

而 12 称为模，即一个系统的量程或此系统所能表示的最大的数，它是自然丢掉的，即

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

意思是，等号两边同除以 12，它们的余数相同，12 称为模。或称(10−4)与(10+8)对模 12 是同余的，也即(+8)与(−4)对模 12 互为补数。

一般说来，若数 Z、Y、K 满足下列关系

$$Z=nK+Y \qquad (n\text{ 为整数})$$

则称 Z 与 Y 对模 K 是同余的，记为

$$Z \equiv Y \qquad (\mathrm{mod}K)$$

在上面的例子中，K＝12。因此(−4)与+8，(−5)与+7，(−6)与+6 等对于模 12 都是同余的，也即它们对模 12 来说互为补数。

由上述的例子和同余的概念，不难得出结论：对于某一确定的模，某数减去小于模的一数，总可以用加上该数的负数与其模数之和(即补数)来代替。所以，引进了补码以后，减法就可以转换为加法了。

根据同样的原理可以得出 10 的补码：

例如有一个五位十进制里程表。起始位置为 00000，往前走一里为 00001，当往前走三里后，即为 00003。但若由起始点往回倒走一里则为 99999，再退回一里则为 99998……。可见在这个里程表中

00001　　表示+1

00002　　表示+2

而

99999　　表示−1

99998　　表示−2

若在前进了三里，里程表指示为 00003 以后，又倒回二里，则我们知道里程表应指示为 00001。也即

$$3-2=1$$

但是也可以表示为

$$3+(-2)=00003+99998=00001$$

这是由于这个里程表的模(即最大量程)100000是自然丢掉的。

其中99998就是在这个数字系统中(−2)的10的补码。可见,当负数用补码来表示时,就可以把减法转换为加法。

由此可引伸出一般情况:某一十进制数X的10的补码,即为10^n-X。n即为此数字系统的最大位数。在上例中,n=5,所以数1的10的补码为100000−1=99999,数2的10的补码为100000−2=99998。536的10的补码为

$$100000-536=99464$$

若有876−536,我们通常做减法

$$\begin{array}{r} 8\ 7\ 6 \\ -\ 5\ 3\ 6 \\ \hline 3\ 4\ 0 \end{array}$$

把−536用它的10的补码来代替,则就可以把减法转换为加法,即

$$\begin{array}{r} 8\ 7\ 6 \\ +\ 9\ 9\ 4\ 6\ 4 \\ \hline \boxed{1}0\ 0\ 3\ 4\ 0 \end{array}$$

自然丢失

结果是相同的,故99464即为(−536)的10的补码。

当负数用10的补码来表示时,就可以把减法转换为加法。但是,在得到补码的过程中,还是不可避免地要做减法,所以,问题并没有解决,因而在实际上10的补码是不用的。

但是,这样的概念可以推广至2的补码,当一个二进制的负数,用2的补码(以后我们把2的补码简称为补码)来表示时,就可以把减法转换为加法,而2的补码我们可以不用减法而由另一途径很方便地找到。

例如,在字长为8位的二进制数字系统中,其模为$2^8=(256)_{10}$。若有

$$64-10=64+(-10)=64+[256-10]=64+246=256+54=54$$

$$\begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ -\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \\ \hline 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \end{array} \qquad \begin{array}{r} 6\ 4 \\ -\ 1\ 0 \\ \hline 5\ 4 \end{array} \qquad \begin{array}{r} 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ +\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0 \\ \hline \boxed{1}0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \end{array} \qquad \begin{array}{r} 6\ 4 \\ +\ 2\ 4\ 6 \\ \hline 5\ 4 \end{array}$$

自然丢失

可见在字长为8位(模2^8)的情况下(64−10)与(64+246)的结果是相同的。所以,(−10)与246互为补数。而这里的$(246)_{10}=11110110$实际上就是我们上述的(−10)的补码表示。所以在上述的负数的补码表示中,实际上我们是利用了补码的概念,把减法转换为加法,而所有的负数(−X)的补码都可由模2^8-X来得到。但是我们是利用了把正数连符号位按位取反再加1这样的简便方法来得到的,避免了求补码过程中的减法,使2的补码的运算具有实用价值。因此,在微型机中,凡是带符号数一律是用补码表示的,所以一定要记住运算的结果也是用补码表示的。

由于计算机的字长是有一定限制的,所以一个带符号数是有一定的范围的,在字长为8位用补码表示时其范围为

$$+127\sim-128$$

当运算的结果超出这个表达范围时,就不正确了,这就称为溢出。这时,数就要用多字节例如16、24位等来表示。

第三节　计算机基础

一、计算机的基本结构

电子数字计算机一开始是作为一个计算工具出现的。我们来看一下若要计算机能脱离人的直接干预，自动地完成计算，它应该具有哪些主要部分呢？

我们以算盘为例来着手分析。若要求用算盘计算下述问题。

$$163 \times 156 + 166 \div 34 - 120 \times 36$$

首先我们需要有一个算盘作为运算的工具；其次要有纸和笔，用来记录原始的数据，记录中间结果，以及最后的运算结果；而整个运算工作是在人的控制下进行的；人首先把要计算的问题和数据记录下来，然后第一步先算 163×156，把计算的中间结果记录在纸上，再计算 $166 \div 34$，把它和上一次的结果相加，再记在纸上，然后计算 120×36，再把它从上一次的结果中减去，就得到了最后的结果。

现在我们要求用一个计算机来完成上述计算过程，显然，它首先要有能代替算盘进行运算的部件，这就称之为运算器；其次要有能起到纸和笔的作用的器件，它能记忆原始题目，原始数据，和中间结果以及为了使机器能自动进行运算而编制的各种命令。这种器件就称为存储器；再次要有能代替人的控制作用的控制器，它能根据事先给定的命令发出各种控制信息，使整个计算过程能一步步地进行。但是光有这三部分还不够，原始的数据与命令要输入，所以需要有输入设备；而计算的结果（或中间过程）需要输出，就要有输出设备，这样就构成了一个基本的计算机系统，如图 1-1 所示。

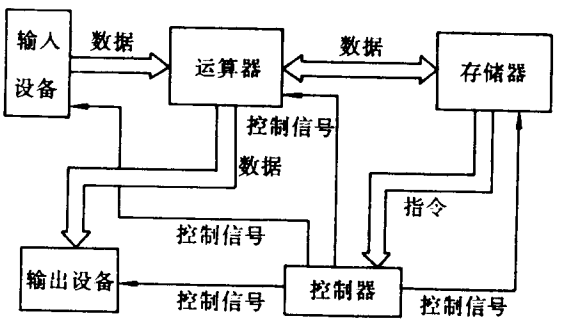

图 1-1　计算机的结构图

在计算机中，基本上有两股信息在流动，一种为数据，即各种原始数据、中间结果、程序等，这些要由输入设备输入至运算器，再存于存储器中，在运算处理过程中，数据从存储器读入运算器进行运算，运算的中间结果要存入存储器中，或最后由运算器经输出设备输出。人给计算机的各种命令（即程序），也以数据的形式由存储器送入控制器，由控制器经过译码后变为各种控制信号。所以，另一股即为控制命令，由控制器控制输入装置的启动或停止，控制运算器按规定一步步地进行各种运算和处理，控制存储器的读和写，控制输出设备输出结果等等。

存储器通常又可以分为内存和外存两部分。内存容量较小（目前通用型的微型机常用的为 640K 至几兆字节），但存取速度快。常用的有磁心或半导体存储器等，目前，在微型机中大部分采用半导体存储器。外存容量大，但存取速度慢。常用的软磁盘、硬磁盘以及磁带等。

输入设备常用的有键盘，纸带读入机，卡片读入机等。输出设备常用的有 CRT 显示，电传打字机，纸带穿孔机，行打印机等。

当把计算机用于控制时，当然输入输出还有各种现场信息和控制命令。

上述的图 1-1 中的各部分构成了计算机的硬件（Hardware）。

在上述的硬件中，人们往往把运算器、存储器和控制器合在一起称为计算机的主机；而把各种输入输出设备统称为计算机的外围设备（或外部设备——Peripheral）。

在主机部分中，又往往把运算器和控制器合在一起称之为中央处理单元——CPU (Central Processing Unit)。

二、常用的名词术语

1. 位(Bit)

位是计算机所能表示的最基本最小的数据单元。因为在计算机中广泛采用二进制数，所以位就是一个二进制位，它只能有两种状态："0"和"1"。由若干个二进制位的组合就可以表示各种数据、字符等。

2. 字(Word)和字长

字是计算机内部进行数据处理的基本单位，通常它与计算机内部的寄存器、运算装置、总线宽度相一致。计算机的每一个字所包含的二进制位数称为字长。

不同类型的微型计算机有不同的**字长**，一般有：1 位、4 位、8 位、16 位、32 位等。1 位和 4 位微型计算机通常用在专用设备中；70 **年代通**用微型机的主流机型是 8 位(字长为 8 位)的，例如用 Intel 的 8080、8085，Motorola 的 6800 **和** Zilog 的 Z-80 为 CPU 的各种微机；80 年代初、中期的主流机型是准 16 位和 16 位的，例如以 8088 为 CPU 的 IBM PC、PC／XT，以 80286 为 CPU 的 IBM PC／AT 机和以 Motorola 的 MC68000 为 CPU 的各种微机等；从 1987 年起的 3～5 年内，主流机型将变为以 80386 和 68020 为 CPU 的 32 位微机。

一个计算机的字长与它能表达数的范围不是一回事，因为可以用单字长、也可以用多字长来表示数。

3. 字节(Byte)

为了表示方便，把相邻的 8 位二进制数位称为一个字节(1Byte＝8Bit)。字节的长度是固定的，但不同计算机的字长是不同的。8 位微机的字长就等于 1 个字节，而 16 位微机的字长等于 2 个字节，32 位微机的字长等于 4 个字节。

目前，在 16 位、32 位微机蓬勃发展的时候，为了表示方便，常把 1 个字节定为 8 位，把一个字(Word)定为 16 位，一个双字(Double Word)定为 32 位。

三、指令程序和指令系统

上一节我们提到了计算机的几个主要部分，这些构成了计算机的硬件(Hardware)的基础。但是，光有这样的硬件，还只是具有了计算的可能。计算机要真正能够进行计算还必须要有软件的配合。首先是各种程序(Program)。

我们知道，计算机所以能脱离人的直接干预，自动地进行计算，这是由于人把实现这个计算的一步步操作用命令的形式——即一条条指令(Instruction)预先输入到存储器中，在执行时，机器把这些指令一条条地取出来，加以翻译和执行。

就拿两个数相加这一最简单的运算来说，就需要以下几步(假定要运算的数已在存储器中)：

第一步：把第一个数从它所在的存储单元(Location)中取出来，送至运算器；

第二步：把第二个数从它所在的存储单元中取出来，送至运算器；

第三步：相加；

第四步：把加完的结果，送至存储器中指定的单元。

所有这些取数、送数、相加、存数等等都是一种操作，我们把要求计算机执行的各种操作用命令的形式写下来，这就是指令。通常一条指令对应着一种基本操作，但是计算机怎么能辨别和执行这些操作呢？这是由设计时设计人员赋予它的指令系统决定的。一个计算机能执行什么样的

操作，能做多少种操作，是由设计计算机时所规定的指令系统决定的。一条指令，对应着一种基本操作；计算机所能执行的全部指令，就是计算机的指令系统(Instruction Set)，这是计算机所固有的。

在我们使用计算机时，必须把我们要解决的问题编成一条条指令，但是这些指令必须是我们所用的计算机能识别和执行的指令，也即每一条指令必须是一台特定的计算机的指令系统中具有的指令，而不能随心所欲。这些指令的集合就称为程序。用户为解决自己的问题所编的程序，称为源程序(Source Program)。

指令通常分成操作码(Opcode—即 operationcode)和操作数(Operand)两大部分。操作码表示计算机执行什么操作；操作数指明参加操作的数的本身或操作数所在的地址。

因为计算机只认得二进制数码，所以计算机的指令系统中的所有指令，都必须以二进制编码的形式来表示。例如在 Intel 8088 中，从存储区取数(以 SI 变址寻址)至累加器 AL 中的指令的编码为 8A04(两字节指令)，一种加法指令的编码为 02 C3，向存储器存数(一种串操作指令)的编码为 AA(一字节指令)等等。这就是指令的机器码(Machine Code)。在 8 位的微型机中字长较短，用一个字节不能充分表示各种操作码和操作数。所以，有一字节指令，有两字节指令，也有多字节指令如四字节指令。

计算机发展的初期，就是用指令的机器码直接来编制用户的源程序，这就是机器语言阶段。但是机器码是由一连串的 0 和 1 组成的，没有明显的特征不好记忆，不易理解，易出错。所以，编程序成为一种十分困难十分繁琐的工作。因而，人们就用一些助记符(Mnemonic)——通常是指令功能的英文词的缩写来代替操作码。如在 Intel 8088 中，数的传送指令用助记符 MOV (MOVE 的缩略)，加法用 ADD，转移用 JMP 等等。这样，每条指令有明显的特征，易于理解和记忆，也不易出错，这就前进一大步，此谓汇编语言阶段。用户用汇编语言(操作码用助记符代替，操作数也用一些符号——Symbol 来表示)来编写源程序。

要求机器能自动执行这些程序，就必须把这些程序预先存放到存储器的某个区域。程序通常是顺序执行的，所以程序中的指令也是一条条顺序存放的。计算机在执行时要能把这些指令一条条取出来加以执行。必须要有一个电路能追踪指令所在的地址，这就是程序计数器 PC (Program Counter)。在开始执行时，给 PC 赋以程序中第一条指令所在的地址，然后每取出一条指令(确切地说是每取出一个指令字节)PC 中的内容自动加 1，指向下一条指令的地址(ADDRESS)，以保证指令的顺序执行。只有当程序中遇到转移指令，调用子程序指令，或遇到中断时，PC 才转到所需要的地方去。

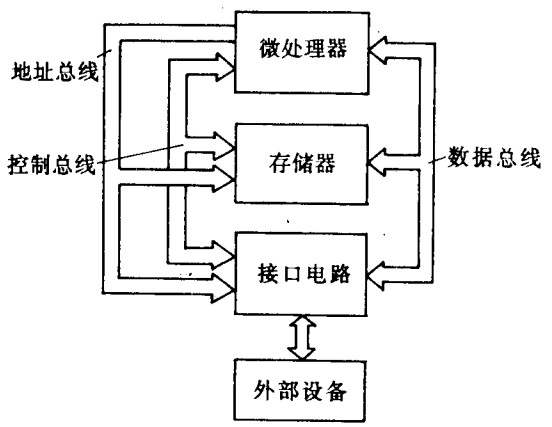

图 1-2　微型计算机结构图

四、初级计算机

在我们开始接触计算机内部结构时，一个实际的微型机结构就显得太复杂了，会使人不知所措，抓不住基本部件、基本概念和基本工作原理。所以，我们先从一个以实际结构为基础，经过简化的模型机，着手来分析基本原理，然后加以扩展，回到实际结构。

图 1-2 所示的是微型机的结构图。它是由微处理器(CPU)、存储器、接口电路组成，通过三条总线(BUS)——地址总

线(Address Bus),控制总线(Control Bus)和双向数据总线(Data Bus)来连接。为了简化问题,我们先不考虑外部设备以及接口电路,认为要执行的程序以及数据,已存入存储器内。

(一) CPU 的结构

模型机的 CPU 结构,如图 1-3 所示。

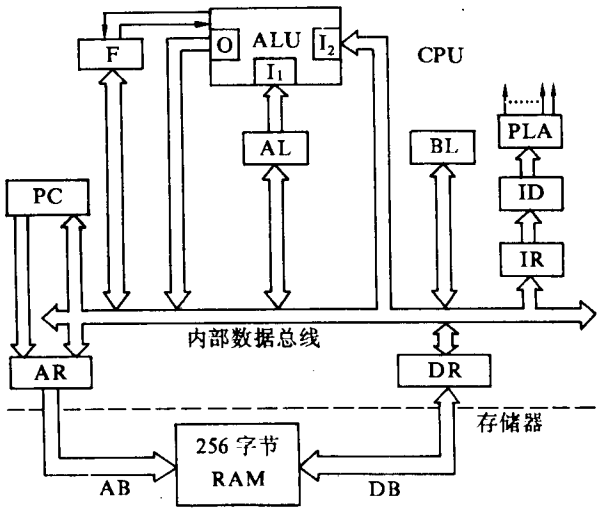

图 1-3 一个模型机的 CPU 结构

算术逻辑单元 ALU(Arithmetic Logic Unit)是执行算术和逻辑运算的装置,它以累加器 AL(Accumulator)的内容作为一个操作数;另一个操作数由内部数据总线供给,可以是寄存器(Register)BL 中的内容,也可以是由数据寄存器 DR(Data Register)供给的由内存读出的内容等;操作的结果通常放在累加器 AL 中。

F(Flag)是标志寄存器,由一些标志位组成;它的功用我们在后面分析。

要执行的指令的地址由程序计数器 PC 提供,AR(Address Register)是地址寄存器,由它把要寻址的单元的地址(可以是指令——则地址由 PC 提供;也可以是数据——则地址要由指令中的操作数部分给定)通过地址总线,送至存储器。

从存储器中取出的指令,由数据寄存器送至指令寄存器 IR(Instruction Register),经过指令译码器 ID(Instruction Decoder)译码,通过控制电路,发出执行一条指令所需要的各种控制信息。

在我们的模型机中,字(Word)长(通常是以存储器一个单元所包含的二进制信息的数表示字长的)为 8 位即为一个字节 Byte(在字长较长的机器中为了表示方便,把 8 位二进制位定义为一个字节),故累加器 AL、寄存器 BL、数据寄存器 DR 都是 8 位的,因而双向数据总线也是 8 位的。在我们的模型机中又假定内存为 256 个单元,为了能寻址这些单元,则地址也需 8 位($2^8=256$),可寻址 256 个单元。因此,这儿的 PC 及地址寄存器 AR 也都是 8 位的。

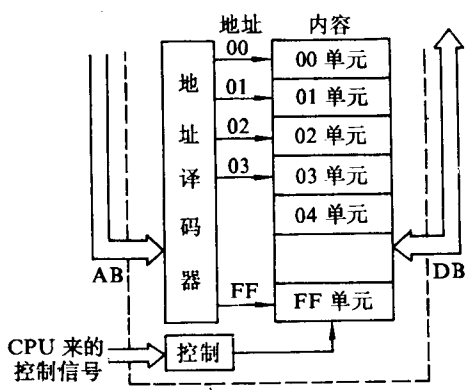

图 1-4 模型机的存储器结构图

在 CPU 内部各个寄存器之间及 ALU 之间数据的传送也采用内部总线结构,这样扩大了数据传送的灵活性,减少了内部连线,因而减少了这些连线所占的芯片面积,但是采用总线结构,则在任一瞬时,总线上只能有一个信息在流动,因而使速度降低。

(二) 存储器

存储器结构如图 1-4。

它由 256 个单元组成,为了能区分不同的单元,对这些单元分别编了号用两位十六进制数表示,这就是它们的地址如 00、01、02、……FF 等;而每一个单元可存放 8 位二进制信息(通常也用