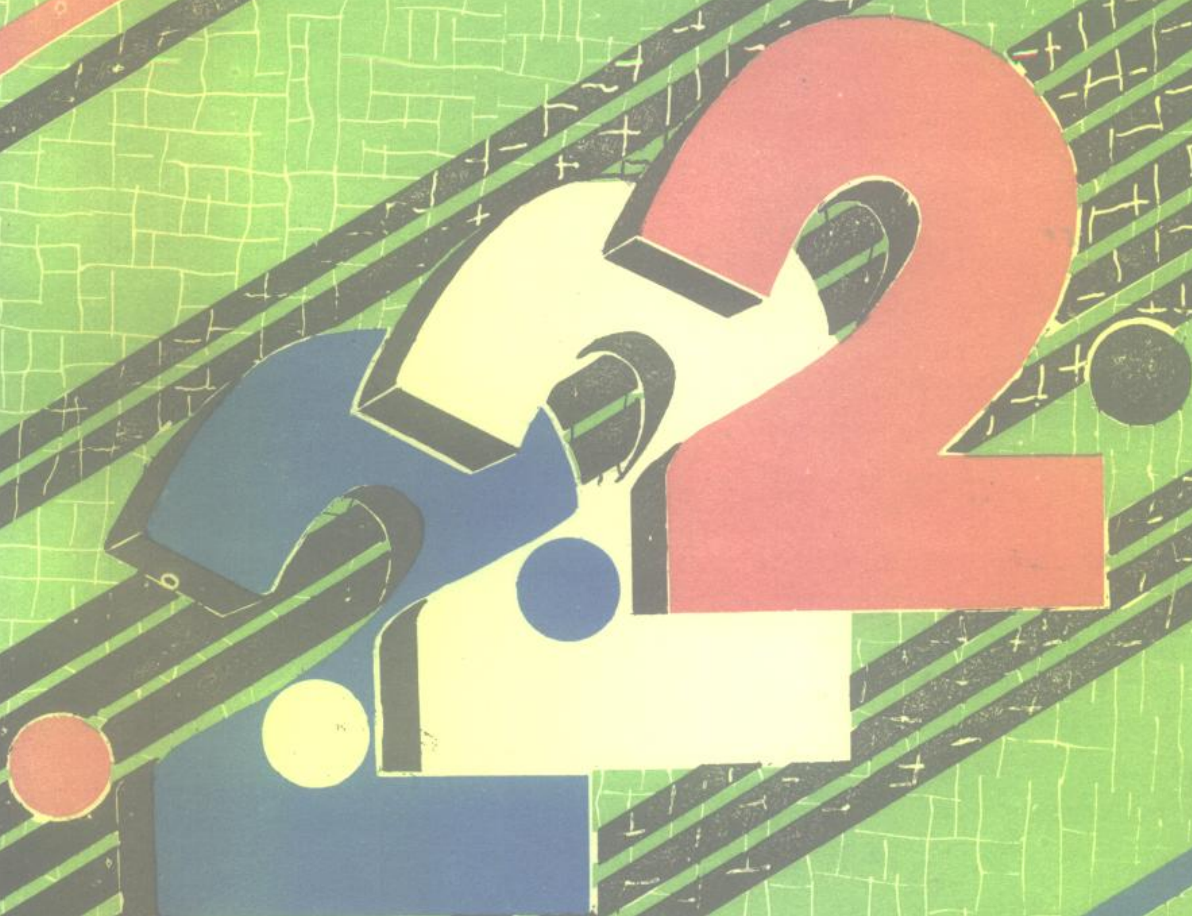
16位微型计算机原理及应用

朱宏兴　编



西北工业大学出版社

16 位微型计算机原理及应用

朱宏兴 编

西 北 工 业 大 学 出 版 社
1995 年 8 月 西安

(陕)新登字 009 号

【内容简要】本书较系统地介绍了16位微型计算机的基本工作原理及接口技术。全书共七章，主要内容有16位微处理器及其指令系统、汇编语言程序设计、半导体存贮器、输入输出接口技术等问题。

本书内容深入浅出，循序渐进，可作为大学本专科院校微型计算机原理及应用课的教材，也可供微型机应用班和科技人员学习参考。

16位微型计算机原理及应用
朱宏兴　编
责任编辑　傅高明
责任校对　钱伟峰
*

(710072　西安市友谊西路127号　电话4253407)
陕西省新华书店经销
西北大学出版社印刷厂排版
陕西西安丰华印刷厂印装
ISBN 7-5612-0790-5/TP·93
*
开本：787×1092毫米 1/16　印张：16.375 字数：399千字
1995年8月第1版　1995年8月第1次印刷
印数：1—5 000册　定价：15.00元

前　言

《微型计算机原理及应用》是高等院校工科本科非计算机专业的必修课程之一，也是工科学生学习和掌握汇编语言程序设计和计算机硬件知识的入门课程。

本书是根据高等学校工科本科非计算机专业《微型计算机原理及应用》课程教学基本要求编写的。参考学时数为 64 学时，通过本课程的学习，应使学生掌握 Intel 8086/8088 微型计算机系统的组成原理，熟练掌握 8086 宏汇编语言程序设计方法，学会使用一系列配套工作的 I/O 接口芯片，以达到使学生具有微机应用系统软、硬件开发的初步能力。

全书共分七章，每章都附有习题：第一章为微型计算机基础；第二章为 8086 微型计算机系统结构；第三章为 8086 指令系统；第四章为汇编语言程序设计；第五章为半导体存贮器；第六章为输入/输出及中断；第七章为输入/输出接口芯片。

由于时间仓促，水平有限，错误和不当之处在所难免，敬请读者批评指正。

编　者

1995 年 3 月

目　录

第一章　微型计算机基础

1.1 概　述

1.1.1 微型计算机的发展概况

1946年在美国诞生了世界上第一台电子计算机ENIAC，该机字长为12位，每秒完成5 000次加法运算，它使用了18 800个电子管、70 000个电阻、1 000个电容、6 000个开关，占地约为170m^2，耗电150kw，重达30t。这个庞然大物被称作第一代电子计算机，为当今的电子计算机奠定了基础。1958年至1964年，用晶体管取代电子管，成为第二代电子计算机，它不仅大大缩小了计算机的体积，而且还降低了成本，同时将运算速度提高了近百倍。1965年集成电路问世，形成了中、小规模集成电路构成的第三代计算机。1970年出现了以大规模集成电路为主体的第四代计算机，这是大规模集成电路迅猛发展的产物。由于在一块芯片上可集成上千万个电子元件，因而使电子计算机的体积大为缩小，这就导致了微型计算机的问世。因为微型计算机具有体积小、功耗低、重量轻、价格低、可靠性高、使用方便等一系列优点，因此获得了广泛的应用和迅速的发展。自微型计算机于1971年问世以来，大约每隔2～4年就更新换代一次，至今已经历了四个阶段的演变。

第一阶段（1971～1973年）为4位和低档8位微处理器及微型计算机。美国Intel公司首先研制成功4位的4004微处理器，以它为核心再配以RAM、ROM和I/O接口芯片就构成了MCS—4微型计算机。随后又研制出8位的8008微处理器及MCS－8微型计算机，其特点是指令系统简单，运算功能较差，速度较慢（平均指令执行时间约为20μs）。

第二阶段（1973～1978年）为中档8位微处理器和微型计算机。其间又分为两个阶段。1973～1975年为典型的第二代，以美国Intel公司的8080和Motorola公司的MC6800为代表，集成度提高1～2倍，运算速度提高一个数量级，1976～1978年为高档的8位微型计算机阶段，被称为第二代半微型计算机，代表产品是美国Zilog公司的Z80和Intel公司的8085微处理器，集成度和运算速度都比典型的第二代提高一倍以上。

第三阶段（1978～1981）是16位微处理器和微型计算机，又称为第一代超大规模集成电路的微处理器。其代表产品是Intel公司的8086/8088，Zilog公司的Z8000和Motorola公司的M68000。这些16位微型计算机都具有丰富的指令系统，并配有强有力的软件系统，时钟频率为4～8MHz，平均指令执行时间为0.5μs。

第四阶段（1981年以后）为32位微处理器和微型计算机。其代表产品是Intel公司的80386和Motorola 68020，时钟频率达16～20MHz，平均指令执行时间约为0.1μs。通常称这类微处理器构成的微型计算机为超级微型机。到目前为止，Intel公司又相继开发出了80486、80586等64位微处理器。

1.1.2 微型计算机的特点和分类

一、微型计算机的特点

由于微型计算机广泛采用了集成度相当高的器件和部件，因此带来以下一系列特点：

1. 体积小、重量轻、耗电省

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小，体积大为缩小。近几年来，微型机还大量地采用大规模集成专用芯片（ASIC）和通用可编程门阵列(GAL)器件，使得微型机的体积又明显缩小。而微型机中的芯片大多采用MOS和CMOS工艺，因此耗电量就很小。这一特点使无法实现的航空、航天等部门中的某些应用领域，现在利用微型机就可很容易实现。

2. 可靠性高

由于微处理器及其配套系列芯片采用大规模集成电路，减少了大量的焊点，简化了外接线和外加逻辑，因而大大提高了可靠性。

3. 系统设计灵活，使用方便

由于微处理器芯片及其可选用的支持逻辑芯片都有标准化、系列化的产品，同时又有许多有关的支持软件可选用，用户可根据不同的要求构成不同规模的系统。

4. 价格低廉

由于微处理器及其配套系统列芯片采用集成电路工艺，集成度高，产品造价十分低廉。一个16位微型机系统IBM XT（包括512K内存，2个5in软盘驱动器，1个硬盘和1个CRT）只需1 000美元左右。

5. 维护方便

由于微处理器及其系列产品已逐渐趋于标准化、模块化和系列化，从硬件结构到软件配置都作了较全面的考虑。一般可用自检诊断及测试发现系统故障。发现故障后，可方便地更换标准化模块或芯片来排除故障。

二、微型计算机的分类

可以从不同角度对微型计算机进行分类。

按微处器的字长，可分为4位、8位、16位、32位、64位微处理器。

按微型计算机的组装形式，可分为单片、单板、多板微型计算机。

按应用领域不同，又可分为控制用、数据处理用微型计算机。

按微处理器的制造工艺，又可分为MOS型器件和双极型器件两大类。

1.2 微型计算机运算基础

1.2.1 不同进位计数制之间的转换

一、十进制数

人们最常用的数制是十进制，它由0～9等十个符号来表示数值。十进制是采用位置记数法的数制，即每个数字的位置决定了它的值或权。同样的数字在不同的位置代表的值或权是不一样的。如

$$555.5=5\times10^2+5\times10^1+5\times10^0+5\times10^{-1}$$

任何一个十进制数N都可表示为

$$(N)_{10}=D_{n-1}\times10^{n-1}+D_{n-2}\times10^{n-2}+\cdots+D_0\times10^0+D_{-1}\times10^{-1}+\cdots+D_{-m}\times10^{-m}$$

$$=\sum_{i=n-1}^{-m}D_i\times10^i$$

其中，i表示数位，D_i表示第i位数码，可以是0～9等十个数码中的任一个，由数N来确定。m、n为正整数，分别表示小数和整数的位数。10为计数制的底数（基数）。

可见，一个十进制数的值，可以用按权展开的多项式来表示，每一个数位都对应一个基值（权）。一个十进制数的权，小数点左面是十的正次幂，小数点右面是十的负次幂。

位置记数法（带权记数法）的数制有以下几个主要特点：

（1）数字的个数等于基数，最大数字比基数小1；

（2）每个数字都要乘以基数的幂次，而该幂次是由每个数所在的位置决定的；

（3）低位向高位的进位是“逢基数进一”。

二、二进制数

在计算机中，广泛使用二进制计数法。它也有三个主要特点：

（1）有两个不同的数字符号，即0和1。

（2）在数的表示中，每个数字都要乘以基数2的幂次，而此幂次是由该数字所在位置决定的，如

$$(101.1)_2=1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}$$
$$=(4+0+1+0.5)_{10}=(5.5)_{10}$$

（3）逢二进一。

二进制数有以下两个主要性质：

（1）二进制数的小数点向右移一位，即整个数向左移一位，数值就增大一倍。反之，小数点向左移一位，即数向右移一位，数值就减小一半，如

$$(101.1)_2=(5.5)_{10}$$
$$(1011)_2=(11)_{10}$$
$$(10.11)_2=(2.75)_{10}$$

（2）对于二进制整数，若最低位是1，则此数为奇数；若最低位是0，则此数为偶数。

由于二进制数只含有两个数字0和1，因此，一个数的每一数位就可以方便地用具有两个不同稳定状态的元件来表示，如开关的闭合与断开，晶体管的截止与导通等。同时，可以节约存贮设备。如要表示0～999这1 000个数，十进制须用三位数，共需3×10=30个稳定状态的设备量。而用二进数表示时，则须用10位（$2^{10}=1\ 024$），只需10×2=20个稳定状态的设备量。

此外，二进制数的运算比较简单，如

加法运算　0+0=0

　　　　　0+1=1+0=1

　　　　　1+1=0，有进位1

减法运算　0−0=0

　　　　　1−1=0

$1-0=1$

$0-1=1$，有借位 1

正因为二进制数只有两个数符，因此可以使用逻辑运算，这为计算机的设计提供了方便。

三、十六进制数

使用二进制数且当数值较大时，书写和阅读都很不方便且易出错。在微型机中，目前通用的字长为 8 位或 16 位，这可用 2 位或 4 位十六进制来表示。因此，十六进制应用十分普遍，它已经成为微处理机产业的标准。

十六进制数也有三个主要特点：

(1) 有 16 个不同的数字符号，即 0～9，A、B、C、D、E、F（其中 A～F 分别代表十进制的 10～15）；

(2) 逢十六进一；

(3) 在数的表示中，每个数字都要乘以基数 16 的幂次，而此幂次由该数字所在位置决定。如

$$(5AF.8)_{16}=5\times16^2+10\times16^1+15\times16^0+8\times16^{-1}$$
$$=(1\,455.5)_{10}$$

四、不同进位数制之间的转换

1. 十进制整数转换为二进制数

采用除 2 取余法，即逐次用 2 去除要转换的十进制数，直至商为 0，每次所得的余数即为二进制数码，最先得到的为整数的最低有效位 K_0，最后得到的为整数的最高有效位 K_{n-1}。如将 25 转换为二进制数

2	25	余数
2	12	$K_0=1$
2	6	$K_1=0$
2	3	$K_2=0$
2	1	$K_3=1$
	0	$K_4=1$

于是 $(25)_{10}=K_4K_3K_2K_1K_0=(11\,001)_2$

2. 十进制小数转换为二进制数

采用乘 2 取整法，即逐次用 2 去乘要转换的十进制小数，将每次所得的整数 0 或 1，依次记作 K_{-1}，K_{-2}…。若乘积的小数部分最后能为 0，则最后一次乘积的整数部分记作 K_{-m}，即为

$$(0.K_{-1}K_{-2}\cdots K_{-m})_2$$

如将 0.625 转换为二进制数

$0.625\times2=1.25$	整数 $1=K_{-1}$
$0.25\times2=0.5$	整数 $0=K_{-2}$
$0.5\times2=1$	整数 $1=K_{-3}$

于是 $(0.625)_{10}=0.K_{-1}K_{-2}K_{-3}=(0.101)_2$

十进制小数并不是都能用有限位的二进制数精确地表示，这时只要根据精度要求，转换到一定的位数即可。

对于一个具有整数和小数部分的十进制数，在转换为二进制数时，只要分别把整数和小数部分转换为二进制数，最后用小数点把这两部分连接起来即可。

3. 十六进制与二进制数之间的转换

因为4位二进制数表示1位十六进制数，所以，十六进制数与二进制数之间的转换以小数点为界，整数部分自右向左每4位为1组，不够则前加零；小数部分自左向右每4位为一组，不够则后加零。如

$$(1001010.1011100)_2 = (01001010.10111000)_2$$
$$= (4A.B8)_{16}$$

十六进制数到二进制数的转换是上述转换的逆过程。如

$$(6D.5E)_{16} = (01101101.01011110)_2$$

1.2.2 二进制数表示

一、数的定点与浮点表示

一个二进制数1010.01可表示为

$$1010.01 = 2^4 \times 0.101001$$

任意一个二进制数N的通式为

$$N = 2^j \times S$$

其中，j是二进制整数的位数，称其为数N的阶码；S是二进制小数，称其为数N的尾数。尾数S表示数N的全部有效数字，而阶码j指明了小数点的位置。

对任何一个数，若阶码j总是固定不变的，则把这种表示法称为数的定点表示，这样的数称为定点数，采用这种表示法的机器叫做定点计算机。如果阶码j可以取不同的值，则把这种表示称为数的浮点表示，这样的数称为浮点数，采用这种表示法的机器叫做浮点计算机。

若定点计算机的阶码j=0，则该定点数只能是小数，其表示的格式为

符号	.	数　值

小数点的位置在符号位与尾数部分最高位之间。

按上述约定，在定点计算机中参与运算的数的绝对值必须小于1。通常，在实际需要计算的数中，不一定都是小数。为了在定点计算机中用这些数进行运算。可以乘上一个比例因子，使其变成小数。例如258.98乘上一个10^{-3}的比例因子，即成为0.258 98，当计算机输出后，再乘上比例因子10^3就恢复原来的值。

比例因子要选择恰当，不能选得太小，否则计算机运算结果仍可能超过1，即计算机出现了溢出。若比例因子选得过大，小数点后好几位都是0，虽然不会溢出，但计算结果的精度将受到很大影响。

为了扩展数的表达范围，提高有效数字的位数，就要采用浮点表示法。在浮点计算机中，浮点数的表示由四部分组成：阶符、阶码、数符（尾数的符号）、数码（尾数）。其格式为

阶符阶码	数符	.	数　码

若定点计算机的字长为9位，其中8个二进制数码位，一个符号位，则该计算机所能表示的数值范围（绝对值）是

$$0.00000001—0.11111111$$

或
$$2^{-8} \sim 1-2^{-8}$$
若计算机的字长是n位，则其所能表示的数值范围是
$$2^{-(n-1)} \sim 1-2^{-(n-1)}$$
对于浮点计算机，总是规定尾数部分的最高位是1，即
$$\frac{1}{2} \leqslant S < 1$$
若浮点计算机的字长为13位，阶符为1位，阶码为3位，数符为1位，数码为8位，则所能表示数的范围是

最小值：11110.10000000

最大值：01110.11111111

或
$$2^{-7} \times 2^{-1} \sim 2^{7} \times (1-2^{-8})$$
若阶码是m位，数码是n位，阶码和数码各有一位符号位，则所能表示数的范围为
$$2^{-(2^m-1)} \times 2^{-1} \sim 2^{(2^m-1)} \times (1-2^{-n})$$
应当注意，浮点数的正负号是由尾数的正负号决定的，而阶码的正负号只决定小数点的位置，即决定浮点数的绝对值大小。

因为浮点计算机中数的小数点是浮动的，故一般的数就可以不用比例因子了。

浮点计算机通用性强，但结构复杂；定点计算机结构简单，但计算结果的精度受影响，且使用不太方便。

通常，微型计算机多为定点计算机（有的附加硬件电路也可进行浮点运算），而且又是整数，即小数点固定在数的末尾。因此，以下的讨论仅限于定点数。

二、符号数的表示法

1. 机器数与真值

通常，约定一个数的最高位为符号位，若字长为8位，则最高位D_7为符号位，$D_6 \sim D_0$为数字位，符号位用0表示正，1表示负。如
$$x=(01010101)_2=+85$$
$$x=(11010101)_2=-85$$
这样，数的符号在机器中也就数码化了。把连同一个符号位在一起的一个数称为机器数，而把它的数值为机器数的真值。

为了运算方便，机器数有3种表示法：原码、反补和补码。

2. 原码

设　$x=x_1x_2\cdots x_{n-1}$
$$[x]_{原}=\begin{cases}0 \quad x_1x_2\cdots x_{n-1} & 当\ x \geqslant 0 \\ 1 \quad x_1x_2\cdots x_{n-1} & 当\ x \leqslant 0\end{cases}$$
例如

若 $x=+85$　则 $[x]_{原}=01010101$

若 $x=-85$　则 $[x]_{原}=11010101$

8位二进制的原码有以下几个特点：

（1）“0”有两种表示法

$[+0]_{原}=00000000$　　$[-0]_{原}=10000000$

(2) 数的表示范围为　$+127 \sim -127$

(3) 数的数码部分不变，而仅仅由 0 和 1 分别来表示数的正负。

原码表示简单易懂，而且与真值转换方便，但是当两个异号数相加或两个同号数相减，就要做减法，致使机器结构复杂化或增加机器的运算时间。为了把减法运算转换为加法运算，就引进了数的反码和补码。

3. 反码

$$[x]_{反}=\begin{cases}0 \quad x_1x_2\cdots x_{n-1} & 当\ x \geqslant 0 \\ 1 \quad \bar{x}_1\bar{x}_2\cdots\bar{x}_{n-1} & 当\ x \leqslant 0\end{cases}$$

其中

$$\bar{x}_i=\begin{cases}0 & 当\ x_i=1 \\ 1 & 当\ x_i=0\end{cases}.$$

正数的反码表示与原码相同，符号位为 0，其余位为数码位；而负数的反码表示，符号位为 1，其余位为数码位按位取反。例如

若 $x=+85$　则 $[x]_{反}=01010101$

若 $x=-85$　则 $[x]_{反}=10101010$

8 位二进制的反码有以下几个特点：

(1) “0” 有两种表示法

$[+0]_{反}=00000000$　　$[-0]_{反}=11111111$

(2) 数的表示范围为　$+127 \sim -127$

(3) 当正数时，后面的 7 位为数码部分；当负数时，必须把后面 7 位按位取反，才表示它的二进制值。

4. 补码

$$[x]_{补}=\begin{cases}0 \quad x_1x_2\cdots x_{n-1} & 当\ x \geqslant 0 \\ 1 \quad \bar{x}_1\bar{x}_2\cdots\bar{x}_{n-1}+1 & 当\ x \leqslant 0\end{cases}$$

正数的补码表示与原码相同；而负数的补码表示，符号位为 1，其余位为数码位按位取反后再加 1。例如

若 $x=+85$　则 $[x]_{补}=01010101$

若 $x=-85$　则 $[x]_{补}=10101010+1=10101011$

8 位二进制的补码有以下几个特点：

(1) “0” 只有一种表示法

$[+0]_{补}=[-0]_{补}=00000000$

(2) 数的表示范围为　$+127 \sim -128$

(3) 当正数时，后面的 7 位为数值部分；当负数时，其余 7 位必须取反后再加 1 才是它的二进制值。

三、补码运算

对一个正数的补码连同符号位在内求反加 1（称为求补运算），即为该正数相应的负数的补码。同样，对一个负数的补码进行求补运算，即得该数正数的补码。如

$x=85$　　$[x]_{补}=01010101$，进行求补运算后得

$[-x]_{\text{补}}=[-85]_{\text{补}}=10101011$

同样，$[-x]_{\text{补}}=10101011$，进行求补运算后得

$[x]_{\text{补}}=[85]_{\text{补}}=01010101$

这一特性在补码的加、减法运算中很有用。

1. 加法运算

根据补码加法规则 $[x+y]_{\text{补}}=[x]_{\text{补}}+[y]_{\text{补}}$，只要将 x 的补码和 y 的补码相加，而不必去考虑它们的正、负号，就能得到两数和的补码。

例

$$\begin{array}{r} 32 \\ +16 \\ \hline 48 \end{array} \qquad \begin{array}{r} 00100000 \\ +00010000 \\ \hline 00110000 \end{array}$$

$$\begin{array}{r} +32 \\ +(-16) \\ \hline 16 \end{array} \qquad \begin{array}{r} 00100000 \\ +11110000 \\ \hline 00010000 \end{array}$$

$$\begin{array}{r} +16 \\ +(-32) \\ \hline -16 \end{array} \qquad \begin{array}{r} 00010000 \\ +11100000 \\ \hline 11110000 \end{array}$$

$$\begin{array}{r} -16 \\ +(-32) \\ \hline -48 \end{array} \qquad \begin{array}{r} 11110000 \\ +11100000 \\ \hline 11010000 \end{array}$$

可见，采用补码以后，可使正、负数的加法运算简化为简单的加法运算。

2. 减法运算

根据补码减法规则 $[x-y]_{\text{补}}=[x]_{\text{补}}+[-y]_{\text{补}}$，减法运算可以通过将减数求补相加来实现。

例

$$\begin{array}{r} 16 \\ -32 \\ \hline -16 \end{array} \qquad \begin{array}{r} 00010000 \\ +11100000 \\ \hline 11110000 \end{array}$$

$$\begin{array}{r} 32 \\ -(-16) \\ \hline 48 \end{array} \qquad \begin{array}{r} 00100000 \\ +00010000 \\ \hline 00110000 \end{array}$$

$$\begin{array}{r} -16 \\ -(+32) \\ \hline -48 \end{array} \qquad \begin{array}{r} 11110000 \\ +11100000 \\ \hline 11010000 \end{array}$$

$$\begin{array}{r} -16 \\ -(-32) \\ \hline 16 \end{array} \qquad \begin{array}{r} 11110000 \\ +00100000 \\ \hline 00010000 \end{array}$$

可见，减法运算可以通过采用进行简单的求补相加运算来进行。

四、二进制编码

在计算机中，所有信息都是采用二进制代码，因而在计算机中表示的数、字母、符号等

都是要以特定的二进制代码来表示，即进行二进制编码。

1. 二进制编码的十进制数

一位十进制数用4位二进制编码来表示，方法很多，而较常用的是8421BCD码。它有十个不同的数字符号，但它是逢十进位的，所以实质上仍是十进制数，但由于它的每一位都是用4位二进制编码来表示的，因此称为二进制编码的十进制数，简称为BCD码(Binary Coded Decimal)，BCD编码如表1-1所示。

表1-1　BCD编码

十进制数	8421 BCD 码	十进制数	8421 BCD 码
0	0000	10	0001　0000
1	0001	11	0001　0001
2	0010	12	0001　0010
3	0011	13	0001　0011
4	0100	14	0001　0100
5	0101	15	0001　0101
6	0110	16	0001　0110
7	0111	17	0001　0111
8	1000	18	0001　1000
9	1001	19	0001　1001

BCD码是比较直观的，如十进制数85.6可方便地写成

$$(10000101.0110)_{BCD}$$

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，即先将BCD码转换为十进制数后，再转换为二进制数。反之亦然。

2. 字母与其它字符的编码

在计算机应用中，各种字符也必须用二进制代码来编码，如26个英文字母、10个阿拉伯数字、运算符号、标点符号以及一些特殊的控制字符，如换行、换页、回车等。

目前在微型机中最常用的一种编码是ASCII码 (American Standard Code for Information Interchange，即美国信息交换标准代码)。它采用7位二进制代码来对一个字符进行编码，故可表示128个字符。因为微型机通常字长为8位，所以最高位可用作奇偶校验位。不过在机器内部通常将此位视为零。故用一个字节 (8个二进制位称为一个字节，即一个Byte) 即可存放一个ASCII码。数字0～9的ASCII码为30H～39H；大写英文字母A～Z的ASCII码为41H～5AH；小写英文字母a～z的ASCII码为61H～6AH (H意指其前的两个数码为十六进制数码)。

1.3　基本逻辑电路

一种用字母和符号代替文字来进行运算推理的方法称为逻辑代数或布尔代数或开关代数。它和普通代数一样，可以写成如下表达式

$$Y=F\ (A,\ B,\ C,\ D)$$

逻辑代数有两个特点：

(1) 表达式中的变量 A、B、C、D 等的取值只有两种可能，即 0 和 1。它并无大小之意，只代表事物的两个不同的性质，代表命题的“真”或“假”。

(2) 各种逻辑关系，用逻辑代数概括起来，只有三种基本关系：逻辑加法（“或”运算)、逻辑乘法（“与”运算)、逻辑否定（“反”运算或称“非”运算)。

能实现“与”、“或”、“非”等逻辑运算的电路称为逻辑门电路。通常，以高电平表示逻辑“1”，低电平表示逻辑“0”

1.3.1 “与”门

能实现“与”运算的电路称为“与”门。它有两个或更多个信号输入端和一个输出端。两个输入端的“与”门电路的逻辑符号如图 1-1 (a) 所示。它完成逻辑“与”运算

$$Y=A\cdot B=A\wedge B$$

若输入 A、B 中，只要有一个为低电平，则输出 Y 为低电平，故有

$$Y=A\cdot B=1\cdot 0=0\cdot 1=0\cdot 0=0$$

若输入 A、B 均为高电平时，输出才为高电平，即有

$$Y=A\cdot B=1\cdot 1=1$$

1.3.2 “或”门

能实现“或”运算的电路称为“或”门。它有两个或更多的输入端和一个输出端。两个输入端的“或”门逻辑符号如图 1-1 (b) 所示。它完成逻辑“或”运算

$$Y=A+B=A\vee B$$

若输入 A、B 均为低电平时，输出才为低电平，故有

$$Y=A+B=0+0=0$$

若输入 A、B 中，只要有一个为高电平，则输出为高电平，即有

$$Y=A+B=1+0=0+1=1+1=1$$

1.3.3 “非”门

实现“非”运算的电路称为“非”门，又称反相器。它只有一个输入端和一个输出端。“非”门的逻辑符号如图 1-1 (c) 所示。它完成“非”运算

$$Y=\overline{A}$$

若输入 A 为高电平，则输出为低电平

$$Y=\overline{A}=\overline{1}=0$$

若输入 A 为低电平，则输出为高电平

$$Y=\overline{A}=\overline{0}=1$$

1.3.4 “异或”门

实现“异或”运算的电路称为“异或”门。“异或”逻辑表达式为

$$Y=A\oplus B=A\overline{B}+\overline{A}B$$

“异或”门只有两个输入端和一个输出端。其逻辑符号如图 1-1 (d) 所示。它完成的“异或”运算为

$$Y=A\oplus B=1\oplus 0=0\oplus 1=1$$
$$Y=A\oplus B=1\oplus 1=0\oplus 0=0$$

1.3.5 “与非”门

任何复杂的逻辑电路都可以利用“与”、“或”、“非”等门电路组合而成。而“与非”门是由“与”门和“非”门组合而成。它有两个或更多的输入端和一个输出端。两个输入端的“与非”门逻辑符号如图1-1（e）所示。它完成的逻辑运算为

$$Y=\overline{A\cdot B}$$

若输入A、B均为高电平时，则输出为低电平

$$Y=\overline{A\cdot B}=\overline{1\cdot 1}=0$$

若输入A、B中，只要其中有一个为低电平，则输出为高电平

$$Y=\overline{A\cdot B}=\overline{1\cdot 0}=\overline{0\cdot 1}=\overline{0\cdot 0}=1$$

1.3.6 “或非”门

“或非”门是由“或”门和“非”门组合而成。它有两个或更多的输入端和一个输出端。两个输入端“或非”门的逻辑符号如图1-1（f）所示。它完成的逻辑运算为

$$Y=\overline{A+B}$$

若输入A、B均为低电平时，则输出为高电平

$$Y=\overline{A+B}=\overline{0+0}=1$$

若输入A、B中，只要其中一个为高电平，则输出为低电平

$$Y=\overline{A+B}=\overline{1+0}=\overline{0+1}=\overline{1+1}=0$$

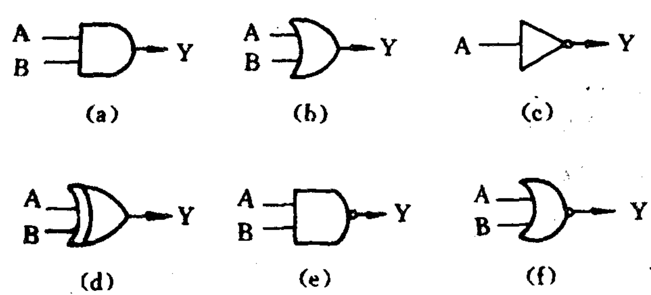


图1-1 基本逻辑电路的符号

1.4 微型计算机的基本组成

电子计算机由运算器、控制器、存贮器、输入设备和输出设备等五大部分组成。通常把运算器和控制器合称为中央处理器CPU (Central Processing Unit)，而把CPU和存贮器合称为计算机的主机。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）。

随着大规模集成电路的发展，作为微型计算机的标志，把运算器和控制器（即CPU）集成在一个或几个芯片上，称之为微处理器。以下对微处理器等基本概念作简单的说明。