微型计算机应用

湖南省机械工业学校　谭维瑜　主编

兵器工业出版社

微型计算机应用

湖南省机械工业学校 谭维瑜主编

兵器工业出版社

(京)新登字049号

内 容 简 介

本书根据原国家机械委1987年颁布的《微机原理及应用》课程教学大纲编写，后经机电部中专计算机专业教学指导委员会评审，认为适合作为中专非计算机专业的教材，特予以正式出版。

本书从应用角度出发，主要介绍Z80CPU工作过程及TP801单板机应用。具体内容有微机运算基础，微机基本结构和工作过程，Z80汇编语言程序设计，微机接口技术，微机系统组成和微机应用举例。根据必需和够用原则，简要地介绍典型微机系统的配置和操作系统的功能及使用。每章编有小结，选有大量实用的例题及习题，并编有实验指导书。

本书可作为普通中专、函授、电视和职业中专工科非计算机专业的教材，亦可作为广大工程技术人员自学之用。

微型计算机应用

湖南省机械工业学校 谭维瑜主编

•

责任编辑：贡克勤

封面设计：方 芬

•

兵器工业出版社 出版发行

（北京市海淀区车道沟10号）

各地新华书店总经销

机械工业出版社京丰印刷厂印装

•

开本：787×1092 1/16 · 印张：16 · 字数400千字

1992年11月第1版 · 1992年11月第1次印刷

印数：00,001—25,000 · 定价：8.50元

•

ISBN7-80038-484-5/TP·36

前　言

随着微机应用技术迅猛发展，《微机原理及应用》课程已作为中专工科各专业的一门重要的必修课。为此，我校根据原国家机械委1987年颁布的该课程教学大纲，结合我校近几年对该课程进行教学改革所取得的成果，在对现有教材深入分析和解剖的基础上，博取众家之长编写了本书。

本书根据中专是培养“应用型”人材及课程大纲所规定的任务，从应用角度出发，主要介绍目前在我国使用广泛的Z80CPU工作过程及TP801单板机的应用。

从应用角度学习微机，中心是学习汇编语言程序设计和接口技术。前者是学习和应用微机的基本工具，重要的是掌握指令的功能、执行过程和程序设计技巧；后者是微机与实际应用对象联接时，必须解决的基本方法和技术，重要的是掌握微机与外部设备之间采用什么方式交换信息和怎样实现信息交换。而这些正是贯穿本书的主线。

本书共分六章。第一章微机运算基础。介绍微机所需的运算基础。

第二章微机基本结构和工作过程，第三章Z80汇编语言 程序设计。这两章以程序设计为主线，以指令为重点，介绍程序设计的基本方法和技巧，培养读者阅读和运用指令进行程序设计的能力。在这里，建立了由CPU寄存器和存储器组成的“程序设计模型”，不过多地涉及硬件结构，来讲述程序设计；没有集中地介绍指令系统，而是按不同结构程序编程的需要，分别介绍各种指令的功能、执行过程和应用；没有专列章节介绍十种寻址方式，而是结合指令的执行过程介绍该指令的寻址方式，使读者会查阅指令表。

第四章微机接口技术。以CPU与I/O设备联接为主线，以接口为重点，介绍微机接口技术，培养读者掌握常用芯片的使用和对微机的应用能力。在这里，不分析芯片的内部电路，着重介绍其“外”特性、引脚功能；详尽介绍芯片的工作方式及应用，可编程芯片的使用方法；重点介绍程序传送方式及中断传送方式的编程与实现。站在系统的角度，紧紧抓住CPU的三总线，一层层由里向外，介绍存储器、键盘、显示器、A/D和D/A转换等接口，线条清晰，一气呵成，为建立微机的系统概念奠定基础。

第五章微机系统。在叙述微机最小系统、TP801的硬件结 构的基础上，逐步扩充，介绍了通用微机系统的配置、系统 总线、操作系统的 基本概念和CP/M（MS-DOS）的使用，使读者对微机系统有一个完整的认识。对于非计算机专业，因只学习这一门微机课程，故很有必要。

第六章微机应用举例。列举了控制领域常见的几种实例，使基础知识和应用实例挂上钩，从而进一步帮助读者实现全书内容的融汇贯通。

本书选编大量实用的例题和习题，有易有难。适应不同专业需要，还编有实验指导书。

本书经机电部中专计算机专业教学指导委员会评审，认为：所选内容能满足该课程要求，在结构体系和内容叙述方法上作了改革，保证了基本内容，突出了应用，有利于教学，适合作为中专工科非计算机专业的教材，并推荐正式出版。

本书由我校高级讲师谭维瑜任主编，编写第一、二、三章，计算机专业教研室副主任廖哲智编写第四、五、六章，杨晖老师编写实验指导书。山东省机械工业学校高级讲师、计算机教研室主任徐仁贵任主审。由于编者水平有限，错漏难免，敬请读者指正。

湖南省机械工业学校

1992年2月

目 录

前言 …… I
第一章 微型计算机运算基础 …… 1
第一节 微型计算机的数制 …… 1
一、进位计数制 …… 1
二、各数制间的转换 …… 2
第二节 二进制数的运算 …… 3
一、算术运算 …… 4
二、逻辑运算 …… 5
第三节 计算机中数的表示方法 …… 7
一、数的符号表示法 …… 7
二、原码、反码、补码 …… 7
三、数的小数点位置表示法 …… 9
第四节 计算机中的编码 …… 10
一、BCD码 …… 10
二、ASCII码 …… 11
小 结 …… 11
习 题 …… 12
第二章 微型计算机基本结构和工作过程 …… 13
第一节 微型计算机概述 …… 13
一、微型计算机发展概况 …… 13
二、微型计算机的特点 …… 13
三、微型计算机的应用 …… 14
第二节 微型计算机系统组成 …… 14
第三节 微型计算机基本结构 …… 15
一、微型计算机基本结构 …… 15
二、输入/输出接口电路 …… 16
三、总线 …… 16
第四节 存储器工作过程 …… 17
一、存储器主要性能指标 …… 17
二、存储器内部结构 …… 18
三、存储器读/写工作过程 …… 18
第五节 Z80微处理器内部结构 …… 19
一、寄存器 …… 19
二、运算器 …… 21
三、控制器 …… 22
第六节 微型计算机工作过程 …… 22
一、一个程序实例 …… 22

二、程序执行过程 ……22
三、指令周期和机器周期 ……23
小 结 ……24
习 题 ……25
第三章 Z80汇编语言程序设计 ……26
第一节 Z80汇编语言程序设计基础 ……26
一、程序设计语言 ……26
二、Z80汇编语言简介 ……28
三、Z80程序设计模型 ……30
四、Z80指令格式和寻址方式 ……31
五、程序设计步骤 ……33
小 结 ……35
习 题 ……35
第二节 顺序程序设计 ……35
一、程序程序结构 ……36
二、数据传送指令 ……36
三、算术和逻辑运算指令 ……44
四、通用算术指令 ……51
五、顺序程序设计举例 ……52
小 结 ……54
习 题 ……55
第三节 分支程序设计 ……57
一、分支程序结构 ……57
二、转移指令 ……57
三、位操作指令 ……62
四、交换指令 ……63
五、分支程序设计举例 ……63
小 结 ……70
习 题 ……70
第四节 循环程序设计 ……73
一、循环程序结构 ……73
二、数据块传送指令 ……73
三、数据块搜索指令 ……76
四、移位与循环移位指令 ……79
五、循环程序设计举例 ……87
小 结 ……92
习 题 ……93
第五节 子程序和堆栈 ……96
一、子程序概念 ……96
二、堆栈技术 ……97
三、子程序调用和返回指令 ……99
四、子程序设计举例 ……100
小 结 ……103

习　题 …… 103
第六节　综合举例 …… 105
一、软件延时程序 …… 105
二、代码转换 …… 105
三、数据查找 …… 106
四、数据排列 …… 108
第四章　微型计算机接口技术 …… 109
第一节　Z80-CPU的引脚功能 …… 109
第二节　存储器及其与80-CPU的接口 …… 111
一、半导体存储器的分类 …… 111
二、Z80-CPU与存储器的接口 …… 113
第三节　输入／输出接口技术基础 …… 119
一、接口的一般结构 …… 119
二、I/O寻址方式及指令 …… 120
三、Z80-CPU与I/O设备的数据传送方式 …… 123
四、中断系统的基本概念 …… 127
五、Z80的中断系统 …… 132
第四节　通用接口芯片 …… 135
一、Z80-PIO …… 135
二、Z80-CTC …… 142
第五节　显示器接口 …… 149
一、信号显示器的接口 …… 149
二、数字显示器的接口 …… 149
第六节　键盘接口 …… 155
一、TP801键盘结构 …… 155
二、键盘(扫描)程序 …… 156
第七节　D/A和A/D转换及接口 …… 159
一、D/A转换接口及其应用 …… 159
二、A/D转换接口及其应用 …… 163
小　结 …… 169
习　题 …… 171
第五章　微型计算机系统 …… 173
第一节　微型计算硬件系统 …… 173
一、微机的最小系统 …… 173
二、单板机 …… 173
三、微机的基本系统 …… 175
第二节　微型计算机系统总线 …… 179
一、总线的规定 …… 179
二、内部总线 …… 180
三、外部总线 …… 187
第三节　微型计算机软件系统 …… 190
一、微型计算机软件概述 …… 190
二、TP801监控调试程序 …… 192

三、CP/M操作系统 …… 193
小 结 …… 199
第六章 微型计算机应用举例 …… 201
第一节 顺序控制系统 …… 201
第二节 数据采集 …… 204
第三节 步进电机的开环控制 …… 206
实验指导书 …… 211
附 录 …… 235
附录A ASCII码表 …… 235
附录B Z80系统指令表 …… 236
附录C 标志操作摘要表 …… 243
附录D Z80指令机器周期表 …… 244
参考文献 …… 248

第一章　微型计算机运算基础

微型计算机是一种能够高速、自动地进行算术、逻辑运算和信息处理的工具。

本章介绍计算机中有关数值运算的基础知识，如数制、码制、数的表示方法等。

第一节　微型计算机的数制

一个数可用不同的数制来表示，例如1斤重，可用500g、0.5kg、10两来表示。人们习惯使用十进制，而在计算机中采用二进制。这是因为二进制只有0和1两个数码，可以用两种相反的状态来表示这两个数码，例如电位的高和低，脉冲的有和无，晶体管的导通和截止。

采用二进制还具有逻辑电路简单、运算法则简便，可以用逻辑代数分析和综合等优点。

为了适应人们的习惯，为了简化书写和阅读，在编写程序时，还使用十、八、十六进制。

一、进位计数制

数制又称进位计数制，它是按进位的原则进行计数的方法。例如，十进位计数制，是根据"逢十进一"的原则进行计数的。

（一）数制的基数和位权

数制所使用的数码的个数称为该数制的"基数(R)"。

数制中每一位置上所具有的值称为"位权"或"权"。各位权是以基数为底的幂$(R)^i$。

每一位数值的大小用该位上的数码与该位的权的乘积来表示。

"基数"和"位权"是进位计数制中的两要素。

1．十进制（Decimal）　它有0～9十个数码，基数为10。各位的权是10^0、10^1、10^2…。按"逢十进一"原则计数。

2．二进制（Binary）　它只有0和1两个数码，基数为2。各位的权是2^0、2^1、2^2…。按"逢二进一"原则计数。

3．八进制（Octal）　它有0～7八个数码，基数为8。各位的权是8^0、8^1、8^2…。按"逢八进一"原则计数。

4．十六进制（Hexadecimal）　它有0～9，A,B,C,D,E,F十六个数码，基数为16。各位的权是16^0、16^1、16^2…。按"逢十六进一"原则计数。

（二）按权展开式

以R表基数；R^i表各位的权；K_i表数码中的任一数码；m、n是幂指数均匀为正整数，m为整数部分的位数，n为小数部分的位数。任意进制的数可用按权展开相加式来表示：

$$N=\pm[K_{m-1}R^{m-1}+K_{m-2}R^{m-2}+\cdots+K_1R^1+K_0R^0+K_{-1}R^{-1}+K_{-2}R^{-2}+\cdots+K_{-n}R^{-n}]$$

$$=\pm\sum_{i=m-1}^{-n}K_iR^i$$

例如，十进制数563.25可表示为：

$$563.25=5\times(10)^2+6\times(10)^1+3\times(10)^0+2\times(10)^{-1}+5\times(10)^{-2}$$

二、各数制间的转换

计算机只"看得懂"二进制数，而人们又不习惯于使用，因此需要进行数的等值转换。

(一) 二、八、十六进制数转换成十进制数

将原进制数按权展开相加求和即成。

例1-1

$(101.101)_2=1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=4+1+0.5+0.125=5.625$

$(24.6)_8=2\times8^1+4\times8^0+6\times8^{-1}=16+4+0.75=20.75$

$(2AF.8)_{16}=2\times16^2+10\times16^1+15\times16^0+8\times16^{-1}=512+160+15+0.5=687.5$

表1-1 常用数制对照表

十进制数	二进制数	八进制数	十六进制数	十进制数	二进制数	八进制数	十六进制数
0	0000	0	0	9	1001	11	9
1	0001	1	1	10	1010	12	A
2	0010	2	2	11	1011	13	B
3	0011	3	3	12	1100	14	C
4	0100	4	4	13	1101	15	D
5	0101	5	5	14	1110	16	E
6	0110	6	6	15	1111	17	F
7	0111	7	7	16	10000	20	10
8	1000	10	8				

(二) 十进制数转换成二进制数

1. 十进制整数转换成二进制整数　采用"除2取余法"——将十进制数除以2并记下余数，再用2除商数，又记下余数，如此继续下去，直到商数得0为止。然后以最后所得余数为二进制数的最高位，最先所得余数为最低位的顺序，即得二进造整数。

例1-2

2|53		
2|26	余数	$1=K_0$
2|13		$0=K_1$
2|6		$1=K_2$
2|3		$0=K_3$
2|1		$1=K_4$
0		$1=K_5$

$\therefore\ 53=(K_5K_4K_3K_2K_1K_0)_2=(110101)_2$

2. 十进制小数转换成二进制小数　采用"乘2取整法"——将十进制小数乘以2取其整数，再以2乘所得积的小数部分，又取其整，如此继续下去，直到积为0或达到所要求的精度为止。然后以最先所取的整数为二进制小数的最高位，最后所取整数为最低位顺序，即得二进制小数。

例1-3

0.375		
× 2		
0.750	整数	$0=K_{-1}$
× 2		
1.500		$1=K_{-2}$
× 2		
1.000		$1=K_{-3}$

$\therefore 0.375=(.K_{-1}K_{-2}K_{-3})_2=(0.011)_2$

如果是整数和小数混合的十进制数，则分别转换，然后用小数点将两部分连接起来。

如将十进制数转换为R进制数，则采用"除R取余法"或"乘R取整法。"。

（三）八进制数与二进制数之间的转换

因为$2^3=8^1$，即3位二进制数与1位八进制数是完全对应的。

1．八进制数转换成二进制数　采用"一换三法"——将每位八进制数用3位等值的二进制数替换。

例1-4

八进制数	5	4	·	3	1
	↓	↓		↓	↓
二进制数	101	100	·	011	001

$\therefore (54.31)_8=(101100.011001)_2$

2．二进制数转换成八进制数　采用"三换一法"——以小数点为界，向左右，每3位二进制数分为一组，最后一组如不足3位以零补足。每组用1位等值的八进制数替换。

例1-5

$(11010.1011)_2$

	001	010	·	101	100
	↓	↓		↓	↓
	3	2	·	5	4

$\therefore (11010.1011)_2=(32.54)_8$

（四）十六进制数与二进制数之间的转换

因为$2^4=16^1$，所以1位十六进制数与4位二进制数是完全对应的。

1．十六进制数转换成二进制数　采用"一换四法"——将1位十六进制数用等值的4位二进制数替换。

例1-6

十六进制数	3	E	·	A	8
	↓	↓		↓	↓
二进制数	0011	1110	·	1010	1000

$\therefore 3E\cdot A8H=111110\cdot 10101B$

2．二进制数转换成十六进制数

采用"四换一法"——以小数点为界，向左右，每4位二进制数分为一组，最后一组若不足4位以零补足，每组用1位等值的十六进制数替换。

例1-7

111010.011B

	0011	1010	·	0110
	↓	↓		↓
	3	A	·	6

$\therefore 111010.011B=3A.6H$

第二节　二进制数的运算

在计算机中经常的运算分为四类：算术运算和逻辑运算。算术运算包括加、减、乘、除，逻辑运算包括与、或、非、异或。

一、算术运算

加减法是基本的运算，利用加减法可进行乘除和其他运算。

（一）加法

二进制数的加法法则为：

$$0+0=0 \quad 0+1=1+0=1$$

$$1+1=10\ (\text{进位}1) \quad 1+1+1=11(\text{进位}1)$$

例1-8

```
进位      0 0 0 1 1 1 1
被加数    1 0 1 0 0 1 0 1
加数    + 0 0 0 0 1 1 1 1
        ---------------
和        1 0 1 1 0 1 0 0
```

可见，两个二进制数相加，每一位有三个数--即相加的两个数和从低位来的进位，用加法法则得到本位的和及向高位的进位。

（二）减法

二进制数的减法法则为：

$$0-0=0 \quad 1-1=0$$

$$1-0=1 \quad 0-1=1\ (\text{借位}1)$$

例1-9

```
被减数    1 1 0 0 0 1 0 0
减数      0 0 1 0 0 1 0 1
借位    -   1 1 1 1 1 1
        ---------------
差        1 0 0 1 1 1 1 1
```

（三）乘法

二进制数乘法法则为：

$$0\times0=0 \quad 0\times1=0$$

$$1\times0=0 \quad 1\times1=1$$

例1-10

```
      1 0 0 1    被乘数
    × 1 0 1 1    乘  数
    -----------
      1 0 0 1  \
    1 0 0 1    |
  0 0 0 0      | 部分积
+ 1 0 0 1      /
  -----------
  1 1 0 0 0 1 1   最后结果
```

二进制数乘法过程与十进制数乘法类似。但有其特点：①若乘数某一位为0，则部分积为0；若为1，则部分积为被乘数。②从第二个部分积开始，下一个部分积都相对于上一个部分积向左移一位。③计算机每次只能加两个数，不能一次把所有的部分积相加，而是先把第一、二两个部分积相加，再将其和与等三个部分积相加，如此类推，例1-10可表示如下：

例1-11

```
    1 0 0 1      被乘数
  × 1 0 1 1      乘  数
  -------
    1 0 0 1      第一个部分积
+ 1 0 0 1        第二个部分积左移一位
  -------
    1 1 0 1 1    第一、二部分积之和
+ 0 0 0 0        第三个部分积左移一位
  -------
    0 1 1 0 1 1  前三个部分积之积
+ 1 0 0 1        第四个部分积左移一位
  -------
  1 1 0 0 0 1 1  前四个部分积之和即最后结果
```

可见，两个二进制数的乘法是采用边移位边相加的方法。所以计算机在进行二进制数乘法时是用移位和相加两种操作来实现的。

（四）除法

二进制数的除法法则为：

$$0 \div 1 = 0 \qquad 1 \div 1 = 1$$

$1 \div 0$ 和 $0 \div 0$ 无意义

例1-12

```
            1 1 0
      ---------------
1 1 0 ) 1 0 0 1 1 0
        1 1 0
        -------
          1 1 1
          1 1 0
          -----
            1 0
```

可见，两个二进制数相除是采用相减和移位的方法。所以计算机在进行二进制数除法时是用移位和相减两种操作来实现的。

（五）乘2和除2

若将一个二进制数左移一位，则可将原数扩大2倍即乘2；若右移一位，则可将原数缩小2倍即除2。

例1-13

如将$(1101)_2=13$左移一位为$(11010)_2=26$，右移一位为$(110.1)_2=6.5$

二、逻辑运算

逻辑运算是两个二进制数逐位按指定的逻辑运算符进行运算，无进位或借位关系。

（一）"与"运算(AND)

"与"运算通常用符号"×"、"·"或∧"表示，它的运算法则为：

$0 \wedge 1 = 0$　读作0与1等于0

$1 \wedge 0 = 0$　读作1与0等于0

$0 \wedge 0 = 0$　读作0与0等于0

$1 \wedge 1 = 1$　读作1与1等于1

可见，"与"运算法则和算术乘法法则相似，故又称为"逻辑乘"。

例1-14

$$\begin{array}{r} 11000101 \\ \wedge\ 00001111 \\ \hline 00000101 \end{array}$$

凡同0相"与"的位结果为0，凡同1相"与"的位结果维持不变。计算机在处理数据时，常用"与"运算来使某些位清零(屏蔽)，使某些位保留。

(二) "或"运算(OR)

"或"运算通常用符号"+"、"∨"表示，它的运算法则为：

$0 \vee 0 = 0$　读作0或0等于0

$0 \vee 1 = 1$　读作0或1等于1

$1 \vee 0 = 1$　读作1或0等于1

$1 \vee 1 = 1$　读作1或1等于1

可见，"或"运算法则与算术加法法则相似，故又称为"逻辑加"。

例1-15

$$\begin{array}{r} 10100101 \\ \vee\ 00001111 \\ \hline 10101111 \end{array}$$

凡同0相"或"的位结果维持不变，凡同1相"或"的位结果为1。常用"或"运算使某些位置1。

(三) "非"运算(NOT)

"非"运算又称"逻辑否定"，在逻辑变量上方加一横线表示。它的运算法则为：

$\overline{0} = 1$　读作0非等于1

$\overline{1} = 0$　读作1非等于0

例1-16

对10100011进行"非"运算，得到01011100。实质上是各位取其反变量。

(四) "异或"运算(XOR)

"异或"运算通常用符号"⊕"表示，它的运算法则为：

$0 \oplus 0 = 0$　读作0异或0等于0

$1 \oplus 1 = 0$　读作1异或1等于0

$0 \oplus 1 = 1$　读作0异或1等于1

$1 \oplus 0 = 1$　读作1异或0等于1

例1-17

$$\begin{array}{r} 11001010 \\ \oplus\ 00001111 \\ \hline 11000101 \end{array}$$

两个相同的逻辑变量相"异或"结果为0，两个不同的逻辑变量相"异或"结果为1。故常用A⊕A对A清零，或对某些位进行处理（例如保留或求反）。

第三节　计算机中数的表示方法

一个二进制数，如果不考虑是正数还是负数，称为"无符号数"。实际的数值会有正有负的，而且还会有整数和小数。因此，在计算机中对数的正和负以及小数点的位置应有一定的表示方法。本节介绍正负数的表示方法，对负数的减法以及小数点位置的表示方法。

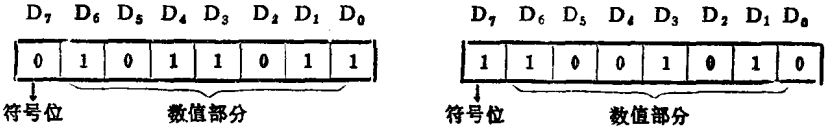
一、数的符号表示法

在计算机中采用了二进制数码，对于数的正负符号也可用二进制数码来表示。我们约定"＋"号用0表示，"－"号用1表示，这样数的正负符号也就数码化了。

（一）带符号数的表示形式

带有正负符号的数称为带符号数或符号数，以最高位作为符号位，其余位为数值部分。

如$N_1=+(1011011)_2=+91$，$N_2=-(1001010)_2=-74$。则带符号数是$N_1=(01011011)_2$，$N_2=(11001010)_2$。

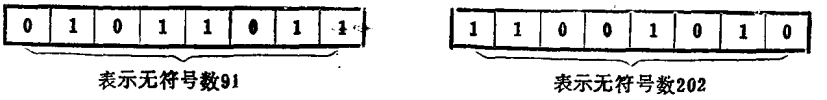


我们把带符号数称为"机器数"，而它的数值称为机器数的"真值"。

（二）无符号数的表示形式

不带符号的数称为"无符号数"，机器的全部有效位都用来表示数值，它相当于数的绝对值。

如上例机器数N_1和N_2，若看作无符号数，则八位全用来表示数值。



可见，对于正数，带符号数与无符号数的表示形式是一样的，数值亦相同。但对最高位是1的数，则需确定是带符号数还是无符号数。机器是不能进行辨别的，需要用户加以认定。

机器数表示形式简单直观，且与真值转换方便。但使机器进行加减法运算变得复杂了。例如，进行两异号数相加或两同号数相减时，需做减法运算，就要判断哪个数的绝对值大，确定谁减去谁；而运算结果的符号需取绝对值大的。这样会使运算速度降低，机器结构复杂。

二、原码、反码、补码

为了解决减法运算问题，引入了数的补码，它可使正、负数的减法运算简化为加法运算。而为了求补码的方便又引入了原码和反码。

因此，带符号数在计算机中有三种表示形式—原码、反码和补码。由于正数，带符号数与无符号数的形式是一样，我们约定，正数的原码、反码和补码的形式是相同的。负数的这三种形式是不同的。我们先介绍这三种形式的求法，然后介绍补码的运算法则。。

（一）原码

带符号数的表示形式，就是原码的表示形式。

$$X=+105,\quad [X]_{原}=01101001$$

$$X=-105,\qquad [X]_{原}=11101001$$

（二）反码

正数的反码表示形式与原码相同。

$$X=+105,\ [X]_{原}=[X]_{反}=01101001$$

负数的反码表示，是将它的原码除符号位外，其余各位按位取反（即1变0，0变1）。

$$X=-105,\ [X]_{原}=11101001,\ [X]_{反}=10010110$$

一个负数的反码，符号位以后各位表示的不是此负数的值。例如，$[X]_{反}=10010110$，它不等于−22。只有将负数的反码再求反，才是此负数的值。

（三）补码

正数的补码表示形式与原码相同。

$$X=+105,\ [X]_{原}=[X]_{补}=01101001$$

负数的补码表示是将它的反码的最低位加1；或者，将它的原码，除符号位外，各位取反，最低位加1。

$$X=-105,\ [X]_{反}=10010110,\ [X]_{补}=10010111$$

与负数的反码表示类似，负数的补码，符号位以后的各位表示的不是此负数的值。例如，$[X]_{补}=10010111$，它不等于−23。只有将负数的补码再求补，才是此负数的值。

表1-2 数的表示法

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

由上表可见：

① 8位二进制数的原码，所能表示的数值范围为+127～−127。

② 8位二进制数的反码"0"有"+0"和"−0"两种表示方法，反码所能表示的数值范围为+127～−127。

③ 8位二进数的补码，$[+0]_{补}=[-0]_{补}=00000000$，所能表示的数值范围为+127～−128。当运算的结果超出这个范围时，结果就不正确了，称为"溢出"。这时要增加位数，例如用16位表示。