汇编语言程序设计

（修订版）

高　等　学　校

工科电子类规划教材精选系列

罗万钧　田立炎
冯子纲　冯世蔚　编

- 8088/8086硬件结构
- 8088/8086指令系统
- 8088/8086汇编语言
- 应用程序设计方法及实例
- 80386/80486程序设计基础

……

西安电子科技大学出版社

高 等 学 校
工科电子类规划教材精选系列

汇编语言程序设计

（修订版）

罗万钧　田立炎
冯子纲　冯世蔚　编

西安电子科技大学出版社
1998

内 容 简 介

本书以 8086/8088 指令系统为主，讲解汇编语言程序设计的基本理论和方法。全书共分 13 章，第 1 章为计算机基础。第 2 至 12 章全面介绍 8086/8088 的系统结构、指令系统、汇编语言约定及程序设计方法。第 13 章介绍 80386/80486 程序设计方法基础。该书内容新颖、系统性好、实用性强，有较多的应用程序实例，并配有相应的上机实习和习题解答辅导教材。

本书是为大学专科计算机应用专业编写的教材，也可作为大学本科(专科)电子类(智能化仪器仪表专业、控制专业)及中高级程序设计培训班的教材或参考书。

高 等 学 校
工科电子类规划教材精选系列
汇编语言程序设计
(修订版)
罗万钧　田立炎
冯子纲　冯世蔚　编
责任编辑　马武装

西安电子科技大学出版社出版
西安电子科技大学印刷厂印刷
陕西省新华书店发行　新华书店经售
开本 787×1092 1/16　印张：23.75　字数：564 千字
1988 年 6 月第 1 版　1998 年 6 月修订版　1998 年 6 月第 7 次印刷　印数：30 001 - 34 000

ISBN 7 - 5606 - 0598 - 2/TP · 0301(课)　　定价：21.50 元

出 版 说 明

为做好全国电子信息类专业“九五”教材的规划和出版工作，根据国家教委《关于“九五”期间普通高等教育教材建设与改革的意见》和《普通高等教育“九五”国家级重点教材立项、管理办法》，我们组织各有关高等学校、中等专业学校、出版社，各专业教学指导委员会，在总结前四轮规划教材编审、出版工作的基础上，根据当代电子信息科学技术的发展和面向21世纪教学内容和课程体系改革的要求，编制了《1996—2000年全国电子信息类专业教材编审出版规划》。

本轮规划教材是由个人申报，经各学校、出版社推荐，由各专业教学指导委员会评选，并由我们与各专指委、出版社协商后审核确定的。本轮规划教材的编制，注意了将教学改革力度较大、有创新精神、有特色风格的教材和质量较高、教学适用性较好、需要修订的教材以及教学急需、尚无正式教材的选题优先列于规划。在重点规划本科、专科和中专教材的同时，选择了一批对学科发展具有重要意义，反映学科前沿的选修课、研究生课教材列入规划，以适应高层次专门人才培养的需要。

限于我们的水平和经验，这批教材的编审、出版工作还可能存在不少缺点和不足，希望使用教材的学校、教师、学生和其他广大读者积极提出批评和建议，以不断提高教材的编写、出版质量，共同为电子信息类专业教材建设服务。

电子工业部教材办公室

前　言

本书是以1988年由西安电子科技大学出版社出版的《汇编语言程序设计》教材为基础，删去了某些陈旧的内容，增加了80386/80486程序设计新内容后，重新修订出版。

本教材由苏州市广播电视大学罗万钧副教授担任主编，西安电子科技大学陈其昌教授担任主审。

本课程的参考学时数为80学时，主要内容为三个部分。第一部分为第1章至第6章，主要内容是8088的指令系统、汇编语言约定及基本程序设计方法，它是本课程的理论基础，也是基本程序设计方法和上机实习的基础。第二部分为第7章至第12章，针对程序的特殊应用如数值运算、数据处理、输入输出、中断及DOS功能调用、BIOS中断调用、与高级语言的接口等进行了专门的介绍，这些程序设计有自身的特点，实用性较强，故列出了某些典型的应用实例。这对于进一步提高汇编程序设计的能力有重要意义。第三部分为第13章80386/80486程序设计基础，主要介绍80386/80486的结构特点、指令系统和工作模式。

由于8086/8088指令系统比较复杂，而汇编语言程序的编写又要求对计算机的硬件有相应的了解，加上汇编语言本身比较抽象，故增加了学习和掌握汇编语言程序设计的困难。因此本教材对重点和难点用一定篇幅加以说明，并编写了相应的实验辅导与习题解答，强调了实践性环节的训练。教材的深度适中，比较适合我国大专计算机应用的实际情况，相信能够收到较好的效果。

本教材由苏州市广播电视大学罗万钧副教授编写第5章、第6章、第9章至第13章。由该校的冯子纲副教授编写第1、第2章。由该校的田立炎讲师编写第7、第8章。由冯世蔚老师编写第3、第4章。冯子纲统编全稿。由于编者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

本书的出版，得到了西安电子科技大学出版社的大力支持和协助，谨此表示诚挚的谢意。

编者

1996年4月

目　录

第1章 计算机基础

1.1 计算机的基本结构与组成

汇编语言是面向机器的语言，它与机器的指令系统有关。由于各类计算机使用的微处理器(或CPU)的指令系统是不同的，因而相应的汇编语言也就不同。本教材以 Intel 8086/8088 指令系统为主，介绍汇编语言程序设计的方法。本书介绍的 Microsoft 宏汇编软件包 5.0 版不仅支持 8086/8088 指令系统和寻址模式，而且支持 80386 的指令系统及寻址模式。如果我们熟悉了 16 位的 8086/8088 微处理器的特性，那么进一步掌握 32 位的 80386 微处理的使用也就不会感到困难。在学习本书时，应当对计算机的基本结构与组成有个大致的了解，知道计算机是如何工作的，了解计算机执行指令的过程，然后再通过指令系统的学习去掌握汇编语言程序设计的方法。

1.1.1 电子计算机的基本组成

计算机由硬件子系统和软件子系统两大部分组成。

所谓硬件子系统(简称硬件系统)系指构成计算机系统的物理实体或物理装置。它由运算器、控制器、存储器、输入输出设备等部件构成，如图 1.1 所示。

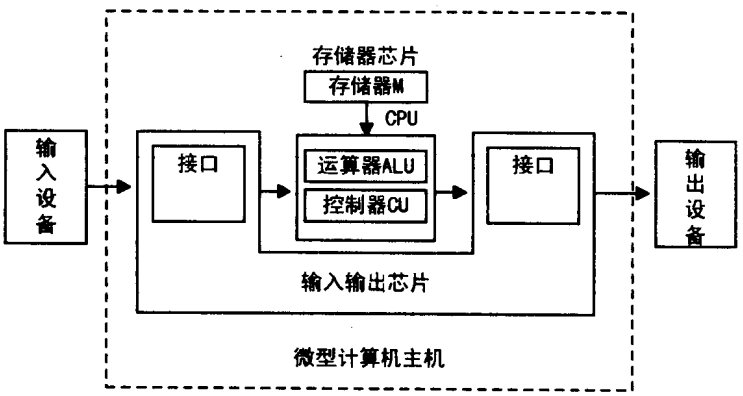

图 1.1 计算机硬件组成

在微型计算机中，将运算器和控制器集成在一片大规模集成电路中，称之为微处理器(或中央处理机 CPU)。

输入设备是计算机从外部世界获取信息的入口，常用的输入设备有键盘、鼠标器等。输出设备是计算机向外部世界输出信息的出口，它把运算的结果或各种信息以数字、字符、图形等形式表示出来。常用的输出设备有打印机、CRT 显示器和绘图仪等。

存储器是计算机存放数据和程序的部件。计算机的存储器分为内存储器和外存储器两大类。内存储器主要采用半导体存储器，其中可以随机进行读和写的存储器称为随机存储器(RAM)，只能读出不能写入信息的存储器称为只读存储器(ROM)。内存储器是主机的一部分，而外存储器如磁盘(硬磁盘和软磁盘)、磁带机、光盘等属于输入输出设备。

输入输出(I/O)芯片是计算机与输入输出设备之间交换数据信息、控制信息、状态信息的接口。

运算器是计算机实现各种算术运算和逻辑运算的部件。

控制器是计算机控制指挥的中心，它的功能是识别翻译指令代码，安排操作次序并向计算机各部分发出适当的控制信号，以便执行机器指令，使计算机能自动地、协调一致地工作。

计算机的软件子系统(简称软件系统)是指计算机系统所用的各种程序的集合，包括系统软件、程序设计语言及应用软件等。

1.1.2　微机硬件系统结构

所谓硬件系统的结构是指由各部件构成系统时的连接方式。一种典型的微型计算机(简称微机)硬件系统结构如图 1.2 所示。

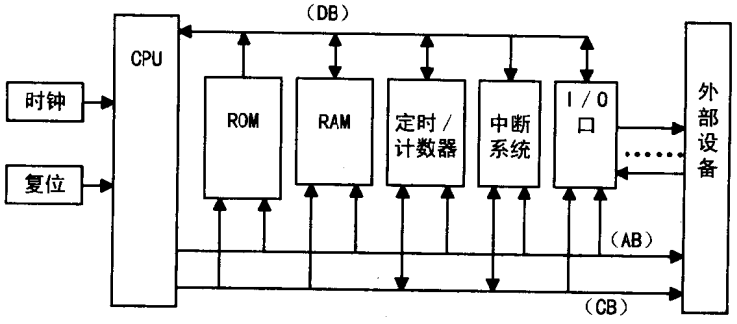

图 1.2　微型计算机系统结构

这种硬件系统结构是单总线的系统结构。总线是进行信息传递的公共信号线，单总线由地址总线 AB(Address Bus)、数据总线 DB(Data Bus)和控制总线 CB(Control Bus)组成。系统中各部件均连接在总线上。这种系统结构简单，易于扩充，所以目前绝大多数微机硬件系统均采用这种结构。

1.1.3 存储器组织

1. 基本概念

存储器用来存放数据和程序。在计算机内部，数据和程序都以二进制代码的形式来表示。

在微机中，一般用 bit 来表示一位二进制位，用 8 位二进制代码作为一个字节 Byte(即 1 Byte＝8 bit)，用 16 位二进制位或两个字节组成一个字 Word(即 1 Word＝2 Byte＝16 bit)，将两个字组成一个双字 Double Word 或 DW。

如果用字表示一个数，称为一个数据字；若用字来表示一条指令，称为指令字。数据字和指令字也可以用双倍字长或多位字长表示。

微机的数据总线宽度有 8 位、16 位和 32 位等，8086/8088 CPU 的数据总线宽度为 16 位，而 80386、80486 的数据总线宽度则为 32 位。

微机的存储器可以拥有很多存储单元，存放在存储单元中的内容为数据或指令代码。每个存储单元都具有唯一的编号，这个编号就是该存储单元的地址。

2. 存储器的组织

现假设存储器由 256 个存储单元组成，字长为 8 位，其结构图如图 1.3 所示。

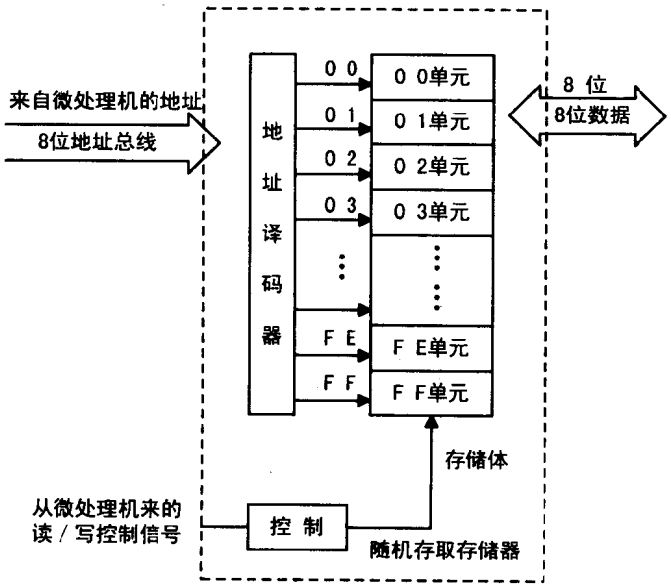

图 1.3 随机存取存储器结构简图

这种存储器既可以随机地向任意一个存储单元写入数据也可以从中读出数据，因而被称为随机存取存储器。它由存储体、地址译码器和控制部件组成。地址译码器接收从地址总线来的地址码，对其译码后选中相应的存储单元。为了能选中地址为 00H～FFH(十六进制表示)的 256 个存储单元中的某一个，应向地址译码器送入 8 位地址代码。控制部件用来控制存储器的读、写操作。

3. 存储器的读写工作过程

从存储器读出信息的过程如图 1.4 所示。

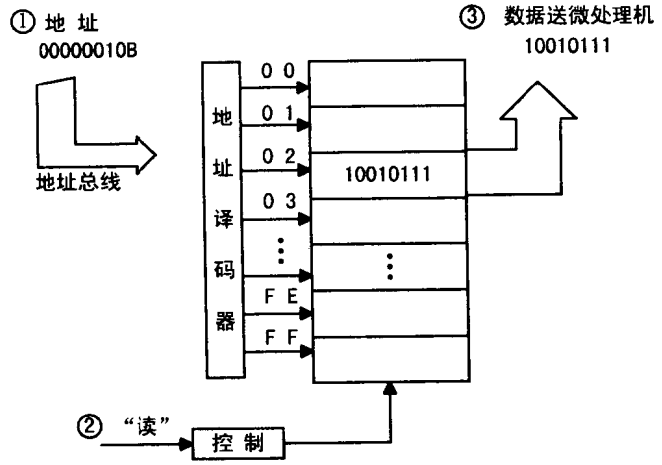

图 1.4　从存储器读出信息的过程

存储器的地址译码器接收从地址总线送来的被读单元的地址代码。当微处理器发来的读控制信号到达控制部件后，便从被选中的存储单元中读出 8 位二进制信息，送至数据总线并经数据总线送至微处理器。

向存储器写入信息的过程如图 1.5 所示。

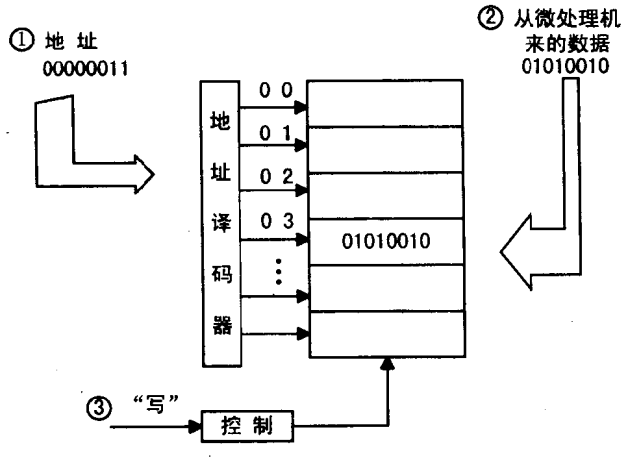

图 1.5　向存储器写入信息的过程

写入操作是将数据总线上的数据存入被选中的存储单元中去。这时，存储器地址译码器从地址总线接收地址代码，微处理器将数据送到数据总线后，接着发出写控制信号，将数据写入指定的存储单元中。

1.2 计算机中的数制与码制

1.2.1 计算机中采用的数制

数制也称为进位计数制。日常生活中，人们习惯采用十进制数进行计算，而计算机内部的信息则是用二进制代码来表示的。

1. 十进制数的特点

① 它有10个不同的符号(或称为元素、系数)，即0，1，2，3，4，5，6，7，8，9。

② 它是逢10进位的，因为一位十进制数能用0～9中的一个符号表示，对于大于等于10的数，就要用多位十进制数来表示。例如一个十进制数为1234.56，其中的"4"代表个位数4，而"3"所代表的数已不是"3"本身，它代表的是30或3×10^1。因此，我们可以把该数表示成如下形式：

$$1234.56 = 1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0 + 5\times10^{-1} + 6\times10^{-2}$$

一般地说，任意一个十进制数N都可表示为

$$N = \pm[K_n\times10^n + K_{n-1}\times10^{n-1} + \cdots + K_0\times10^0 + K_{-1}\times10^{-1} + \cdots + K_{-m}\times10^{-m}] = \pm\sum_{i=-m}^{n}[K_i\times10^i]$$

其中K_i=0，1，2，…，9。

2. 数的一般表示法

对不同的数制，N可表示为：

$$N = \pm\sum_{i=-m}^{n}[K_i\times R^i]$$

R称为进位制的基数。在十进制中，R=10；在二进制中，R=2；在十六进制中，R=16。

R^i称为以R为基数的第i位的位权，显然，不同进位制中的位权是不一样的(即与R有关)。同一数制中不同的位的权也不相同(即与i值有关)。

K_i称为第i位的系数，在以R为基数时，K_i=0，1，2，…，R−1。

例如：R=2，K_i=0，1

R=10，K_i=0，1，2，…，9

R=16，K_i=0，1，2，…，15

常用的几种进位计数制的表示方法见表1.1。

3. 各种数制间的转换

(1) 十进制与其它计数制之间的转换

十进制数转换二进制数，其整数转换与小数转换的方法是不一样的，下面分别叙述之。

十进制整数转换成二进制整数采用除2取余法。因为偶数能被2整除，因而对应的二进制数的最低位应为0；奇数不能被2整除，因而对应的二进制数的最低位为1。这里出现的0和1恰恰就是该数用2整除后的余数。

表 1.1　常用进位计数制数的表示方法

十进制数	二进制数	八进制数	十六进制数
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
⋮	⋮	⋮	⋮

例 1.1　将十进制数 25 转换成二进制数。

解

$$
\begin{array}{r|l|l}
2 & 25 & \text{余数} \\
\hline
2 & 12 & 1=K_0 \\
\hline
2 & 6 & 0=K_1 \\
\hline
2 & 3 & 0=K_2 \\
\hline
2 & 1 & 1=K_3 \\
\hline
 & 0 & 1=K_4
\end{array}
$$

$(25)_{10}=(K_4\,K_3\,K_2\,K_1\,K_0)_2=(1\ 1\ 0\ 0\ 1)_2$

此处，$(25)_{10}$表示十进制数 25，$(11001)_2$ 表示二进制数 11001。

除 2 取余法是将$(25)_{10}$这个数不断地除以 2，除尽，即余数为 0，$K_i=0$；除不尽，即余

数为1，$K_i=1$。第一次的余数为转换后的二进制数的最低位，最后一次的余数为转换后的二进制数的最高位。

以上结果验证如下：

$$(11001)_2 = 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0$$
$$= 16 + 8 + 0 + 0 + 1 = (25)_{10}$$

把十进制纯小数转换成二进制纯小数，采用小数部分乘2取整法。因为大于等于0.5以上的十进制小数，转换成的二进制纯小数的最高位为1，即$(0.1)_2=(0.5)_{10}$，而乘2即得整数1。反之，小于0.5的十进制小数，对应的二进制纯小数的最高位为0，而小于0.5的数乘2，所得乘积的整数部分为0。

例1.2 将十进制纯小数0.3125转换成二进制数。

解

0.3125		
×2	乘积整数位	
0.625	0	即 $K_{-1}=0$
×2		
1.250	1	即 $K_{-2}=1$
0.25		
×2		
0.5	0	即 $K_{-3}=0$
×2		
1.0	1	即 $K_{-4}=1$

故$(0.3125)_{10}=(0.K_{-1}K_{-2}K_{-3}K_{-4})_2=(0.0101)_2$。

以上结果验算如下：

$$(0.0101)_2 = 0\times 2^{-1} + 1\times 2^{-2} + 0\times 2^{-3} + 1\times 2^{-4}$$
$$= 0 + 0.25 + 0 + 0.0625$$
$$= (0.3125)_{10}$$

将十进制混合小数转换成二进制数时，可采用整数、小数部分分别转换的办法。它们的逆运算，即二进制数转换成十进制数，正如例1.1、例1.2的验算那样，按通用表达式展开求和即可得到。

不难验证，将十进制整数转换成八进制整数，可用“除8取余法”来实现，而将八进制数转换成十进制数，亦按通用表达式展开求和得到。

同样，将十进制整数转换成十六进制整数，可用“除16取余法”，十进制纯小数转换成十六进制纯小数，可用“乘16取整法”实现，而十六进制数转换成十进制数，亦按通用表达式展开求和得到。

由于八进制及十六进制与二进制数之间存在着特殊的简单关系，即1位八进制数对应3位二进制数，1位十六进制数对应4位二进制数，所以将它们转换成十进制数时，可以先转换成二进制数，然后再将二进制数转换成十进制数。

(2) 二进制与八进制之间转换

因为 1 位八进制数可用 3 位二进制数表示，故二进制数转换为八进制数，可以小数点为界，整数部分从右向左每 3 位为一组，不足 3 位则左边加零；小数部分从左向右每 3 位为一组，不足 3 位则右边补零，然后按二进制与八进制的关系直接转换。例如：

$$(1111101.011100010)_2 = (001\ 111\ 101.011\ 100\ 010)_2$$
$$= (175.342)_8$$

由八进制数到二进制数的转换是上述过程的逆运算，即将八进制数每一位展开为 3 位二进制数进行。例如：

$$(503.061)_8 = (101\ 000\ 011.000\ 110\ 001)_2$$

(3) 二进制与十六进制之间转换

十六进制数与二进制数之间也存在着类似八进制与二进制之间的简单而又直接的转换方法：以小数点为界，整数部分从右向左，每 4 位为一组，不足 4 位则左边加零，小数部分从左向右，每 4 位为一组，不足 4 位则右边补零，然后按二进制与十六进制之间的对应关系直接转换。例如：

$$(11010101000.10110011111)_2 = (0110\ 1010\ 1000.1011\ 0011\ 1110)_2$$
$$= (6A8.B3E)_{16}$$

同样，由十六进制转换为二进制的例子如下：

$$(E80.5EC)_{16} = (1110\ 1000\ 0000.0101\ 1110\ 1100)_2$$

1.2.2　计算机中数和字符的编码

1. 机器数和真值

一个数若不考虑它的符号，即为无符号数。一个数的前面冠以“+”或“−”的符号，表示该数为正数或负数，这样的数称为带符号数。为了与机器中表示的数相区别，因此把这种正负号表示的数称之为真值。

当然，计算机中所有的数都用 0 或 1 表示，机器并不认识正负号，只有将正负号也用 0 或 1 编码，机器才能识别。通常，符号位为 0 表示正，为 1 表示负。这种表示数的方法，称为带符号数的表示方法。请看以下两个例子：

0 1011011　　1 1011011

↓　　↓

符号位 数值部分　　符号位 数值部分

以上两数中的最高位为符号位。符号位为 0 的数对应的真值为$(+91)_{10}$，符号位为 1 的数对应的真值为$(-91)_{10}$。符号本身并不是数值的一部分，通常不参加加减乘除等运算。数在机器中的表示形式，称为机器数，机器数与真值之间存在着一一对应的关系。

若机器数的最高位不是符号位，也是数值的一部分，则称为无符号的机器数。对 8 位二进制数来讲，无符号的机器数表示范围为$(0\sim255)_{10}$，带符号的机器数表示范围与它采用的表示方法有关，通常有原码、反码和补码 3 种表示方法。

2. 原码、反码、补码

(1) 原码表示法

将一个数的真值的符号用0或1表示，数值部分保留不变就得到该数的原码。

例如：$X=1001010$，因 X 是正数，所以，$[X]_{原码}=01001010$。又如：$X=-1001010$，因 X 是负数，所以，$[X]_{原码}=11001010$。

一般可将原码定义为

$$[X]_{原}=\begin{cases} X, & \text{当 } 0\leqslant X<2^{n-1} \\ 2^{n-1}-X, & \text{当 } -2^{n-1}<X\leqslant 0 \end{cases} \tag{1.1}$$

其中，n 为二进制数的位数，其模为 2^n。$n=4$，$2^4=16$，即模为 2^4 时，所能表示的最大的数要小于 2^4，即15。在带符号位的情况下，由于符号位占1位，所以正数的表示范围最大为+7，负数的表示范围最小为−7。当 $n=8$ 时，原码表示的数值范围为−127～+127。其中的零有正零和负零两种情况，机器中均按零的原码来处理。

$[+0]_{原}=0\underbrace{0000000}$

↓

符号位 $n-1$ 个0

$[-0]_{原}=1\underbrace{0000000}$

↓

符号位 $n-1$ 个0

一般采用正零表示法表示零的原码。

(2) 补码表示法

两个正数相加时，其和为正数；两个负数相加时，其和为负数。而当一个正数与一个负数相加时，必须做减法，而和的符号位由绝对值较大的那个数的符号决定。为了确定其和的符号，必须判断两个数的绝对值的大小，这将使求和的过程及控制线路变得很复杂。当采用补码表示法时，正负数的加减运算被转化为单纯的补码相加运算，从而使问题得以简化。

在日常生活中，补码的例子是很多的，如时钟的校对就是其中一例。设标准时间是6点整，而时针现在指向10点整。若要将时针拨到6点整，可有两种拨法：

(1) 10−4=6，采用倒拨的办法实现，将时针倒拨4小时。

(2) 10+8=6，采用顺拨的办法实现，将时针顺拨8小时，时针指向6点整。

这儿倒拨与顺拨的效果是相同的，也即是互补的。在时钟以12为模的条件下，−4与+8是互补的。

用数学表达式可写成：

$$-4=+8(\text{mod } 12)$$

也称−4与+8对模12是同余的。所谓同余就是−4和+8被12整除时，其余数是相同的。

当然，时间为6点整与时间为18点整，时针的位置是相同的，因此也可以表示为：

$$6=18(\text{mod } 12)$$

6与18对模12是同余的。

由于在模12情况下，超过12的数只能用12以内的同余数表示，故18用6表示。因