高等学校
工科电子类 规划教材

罗万钧 冯子纲 赵正德

汇编语言程序设计

HUIBIAN
YUYAN
CHENGXUSHEJI

电子科技大学出版社

汇编语言程序设计

罗万钧　冯子纲　赵正德

电子科技大学出版社

.1994.

[川]新登字 016 号

内 容 简 介

本书首次以单片机指令系统为主，讲解汇编语言程序设计的基本理论和方法，该书取材新颖、实用性强、系统性好。

全书共分十二章，前五章内容为 MCS—51 系列单片机的指令系统和程序设计方法，第六章介绍 MCS—96 系列单片机的指令系统和应用实例，第七至十一章系统地介绍 MASM—86 宏汇编语言程序设计。

本书是全国大专计算机应用专业统编教材，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。

汇编语言程序设计

罗万钧　冯子纲　赵正德

*

电子科技大学出版社出版

（成都建设北路二段四号）　邮编 610054

四川建筑印刷厂胶印

四川省新华书店经销

*

开本 787×1092　1/16　印张 25.625　字数 618 千字

版次 1994 年 6 月第一版　印次 1994 年 6 月第一次印刷

印数 1—5000 册

中国标准书号 ISBN 7-81043-043-2/TP・26

定价：19.80 元

出 版 说 明

根据国务院关于高等学校教材工作的规定，我部承担了全国高等学校和中等专业学校工科电子类专业教材的编审、出版的组织工作。由于各有关院校及参与编审工作的广大教师共同努力，有关出版社的紧密配合，从1978～1990年，已编审、出版了三个轮次教材，及时供给高等校和中等专业学校教学使用。

为了使工科电子类专业教材能更好地适应“三个面向”的需要，贯彻国家教委《高等教育“八五”期间教材建设规划纲要》的精神，“以全面提高教材质量水平为中心，保证重点教材，保持教材相对稳定，适当扩大教材品种，逐步完善教材配套”，作为“八五”期间工科电子类专业教材建设工作的指导思想，组织我部所属的九个高等学校教材编审委员会和四个中等专业学校教学指导委员会，在总结前三轮教材工作的基础上，根据教育形势的发展和教学改革的需要，制定了1991～1995年的“八五”（第四轮）教材编审出版规划。列入规划的，以主要专业主干课程教材及其辅助教材为主的教材约300多种，这批教材的评选推荐和编审工作，由各编委会或教学指导委员会组织进行。

这些教材的书稿，其一是从通过教学实践、师生反映较好的讲义中经院校推荐、由编审委员会（小组）评选择优产生出来的，其二是在认真遴选主编人的条件下进行约编的，其三是经过质量调查在前几轮组织编写出版的教材中修编的。广大编审者、各编审委员会（小组）、教学指导委员会和有关出版社，为保证教材的出版和提高教材的质量，做出了不懈的努力。

限于水平和经验，这批教材的编审、出版工作还可能有缺点和不足之处，希望使用教材的单位，广大教师和同学积极提出批评和建议，共同为不断提高工科电子类专业教材的质量而努力。

电子工业部电子类专业教材办公室

前　言

本教材系按电子工业部的工科电子类专业教材 1991～1995 年编审出版规划，由全国大专计算机专业教材编审委员会基础课教材编审小组征稿并推荐出版。责任编委为潘道才。

本教材由苏州市广播电视职工大学罗万钧副教授担任主编，上海第二工业大学江庚和教授担任主审。

本课程的参考学时数为 80 学时，其主要内容为两大部分，第一部分为 MCS—51 及 MCS—96 系列的单片机汇编语言程序设计，并且其中以 MCS—51 为重点。因为它的指令系统规模较小，汇编语言程序编写比较简单，并且又可以通过 IBM—PC 个人计算机进行汇编语言编写，汇编、调试和运行的全面训练，为以后学习指令系统比较庞大，汇编语言比较系统，但又比较复杂的、具有结构化程序设计特点的 8086/8088 汇编语言程序打下基础。第二部分为 8086/8088 汇编语言程序设计。两部分学时各占一半左右。两部分以相同的次序介绍两者的指令系统、汇编语言程序设计的基本约定和基本方法，然后介绍汇编语言源程序的开发工具和开发方法，最后介绍数据处理及接口程序设计的常用方法和技巧。由于以上两种指令系统的汇编语言取代了以往的 Z80 CPU 指令系统，因而从根本上更新了教材的内容，使之能尽量反映出计算机的最新技术成果，使教材有更强的实用性，具有相对稳定的使用周期。在使用本教材时，首先要结合各校的具体情况，选择两种指令系统的汇编语言并以一种指令系统为主，也可以只选择其中的一种指令系统汇编语言进行教学。但是本教材本身是两种指令系统汇编语言的有机结合，因此可以节省教学时间，深入浅出地学习，达到事半功倍的效果。本教材强调了实践性环节的训练，编写了相应的实验指导和练习思考题，介绍了汇编源程序的开发工具和开发过程，希望教学中能通过具体的开发实践的训练，在较短的时间内比较熟练地掌握汇编语言程序设计应用开发的方法和技巧，从而改变过去仅仅通过上机实习，单纯验证课堂理论的局面。

本教材由苏州市广播电视职工大学罗万钧副教授编写了其中的第二章至第五章以及第九章和第十一章，由该校的冯子纲副教授编写了其中的第一章、第六章至第十章，由上海科技专科学校的赵正德讲师编写了其中的第十二章，冯子纲统编了全稿。参加审阅工作的还有华东工学院的袁承恩副教授和苏州市商业职工大学的曹宝光副教授，他们都为本书提出了许多宝贵意见，谨此表示诚挚的谢意。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。

编　者

目　录

第一章　计算机基础

汇编语言是面向机器的语言，它与机器的指令系统有关，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，计算机执行程序特别是执行指令的过程，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　计算机的基本结构与组成

一、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称物理装置。它由运算器 控制器、存储器、输入输出接口等部件构成主机（简称计算机）。主机配以输入输出设备，便构成了计算机的硬件系统，如图 1-1 所示。

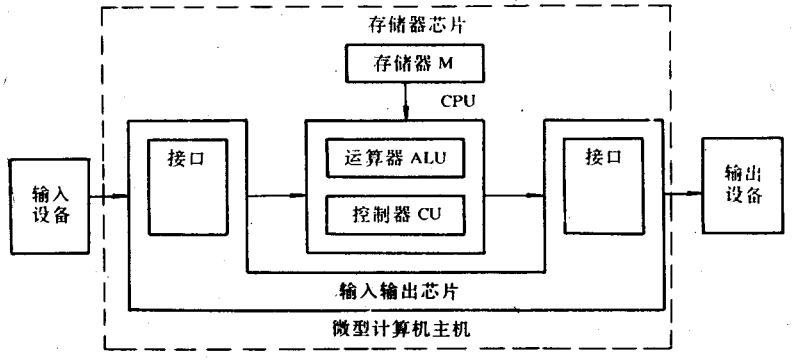


图 1-1　计算机硬件组成

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）。将主机集成在一片大规模集成电路的芯片上，又称为单片机。

图中，输入设备是人与计算机交往的入口。常用的输入设备有键盘、光电输入机等。

输出设备：计算机与人交往的输出窗口，它把运算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、阴极射线管（CRT）显示器和绘图仪等。

存储器：存储数据和程序的部件。计算机的存储器分为内存储器和外存储器两大类别。内存储器主要采用半导体存储器，它包括随机存取存储器（RAM）和只读存储器（ROM，可编程 ROM 即 PROM，可用紫外线光擦除的 PROM 即 EPROM，以及可用电改写的 PROM 即

EEPROM或 E^2 PROM)。内存储器是主机的一部分;而外存储器如磁盘、磁带机和磁鼓等,则属于输入输出设备,有时也称为外部设备或外设。

输入输出芯片(I/O 芯片)是计算机与输入输出设备之间的接口。

运算器:计算机对各种信息进行算术运算和逻辑运算的主要部件。

控制器:是计算机的控制指挥中心,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。

运算器、控制器集成在一块芯片时,称为中央处理器(CPU)或称为微处理器(MPU)。

计算机的软件子系统(简称软件系统)系指计算机系统所使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。

没有软件系统的计算机系统(称为裸机系统),是没有什么用途的。当然,没有硬件系统,软件系统也就无立足之地了。现代的计算机硬件系统和软件系统之间的分界线并不明显,总的趋势是两者统一融合,在发展上互相促进。

二、微机硬件系统结构

由于本书介绍的单片机 MCS—51 以及 8086/8088 指令系统的汇编语言都是微型计算机的汇编语言,所以我们主要介绍微机硬件系统的结构。

所谓微机硬件系统的结构系指由各部件构成系统的连接方式。一种典型的微机硬件系统结构如图 1-2 所示。

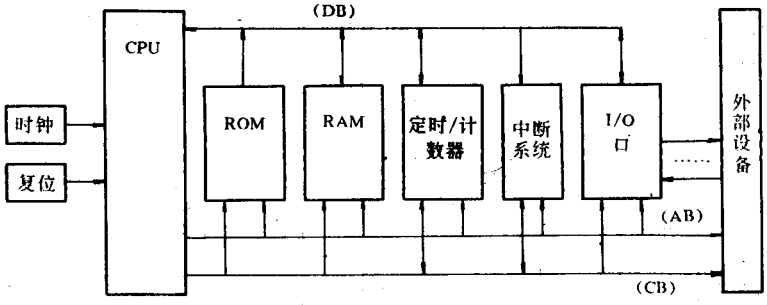


图 1-2 微型计算机系统结构

这种硬件系统结构是单总线(或称系统总线)系统结构。单总线是一族用来进行信息传递的公共信号线,它由地址总线(AB)、数据总线(DB)、控制总线(CB)组成。系统中各部件均挂在单总线上。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。

由于单片机是一个不带外部设备的微型计算机,可看成是一个不带监控程序、没有显示器及键盘的单板机,如图 1-3 所示。

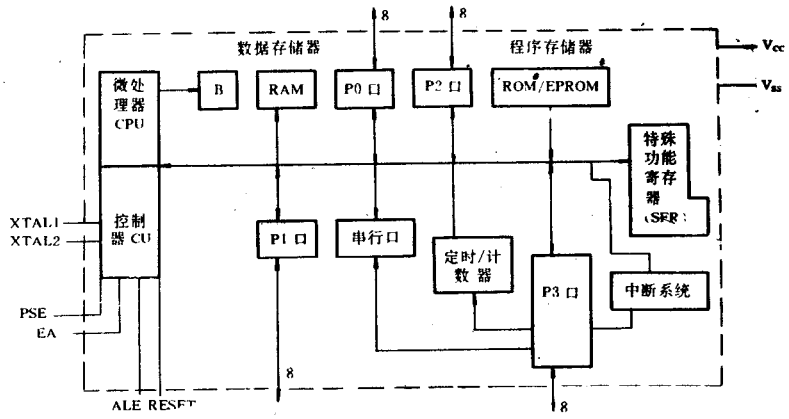


图 1-3　MCS—51 单片机内部结构

由图 1-3 可见，MCS—51 单片机片内结构也属于单总线结构。

三、存储器组织

(一)　基本概念

存储器用来存放数据和程序。在计算机内部，数据和程序都用二进制代码的形式来表示。

在微机中，一般用 8 位二进制代码作为一个字节(即(1Byte＝8 bit)。用 16 位二进制位或两个字节组成一个字，即 1 Word＝2 Byte＝16 bit，将两个字组成一个双字，即 1 Doubleword ＝2 Word＝32 bit。

如果用字表示一个数，称为数据字；用字表示一条指令，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。

微机的字长多为 8 位和 16 位，高档微机的字长可达 32 位。例如 MCS—51 单片机的字长为 8 位，而 MCS—96 单片机，8086/8088 CPU 的字长为 16 位，80286 的字长亦为 16 位，而 80386 及 80486 的字长则为 32 位。

一个存储器可包括很多存储单元，存储单元的内容为数据或指令，存储单元的编号称为地址。

(二)　存储器组织

现假设存储器由 256 个字组成，字长为 8 位，其结构如图 1-4 所示。

存储器由 256 个单元组成，每个单元存储一个字节，这种规格的存储器，通常称为 256 ×8 位的读写存储器。

随机存取存储器由存储体、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，经译码器译码选中相应的存储单元，便从其中读出信息或写入信息。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和写操作。

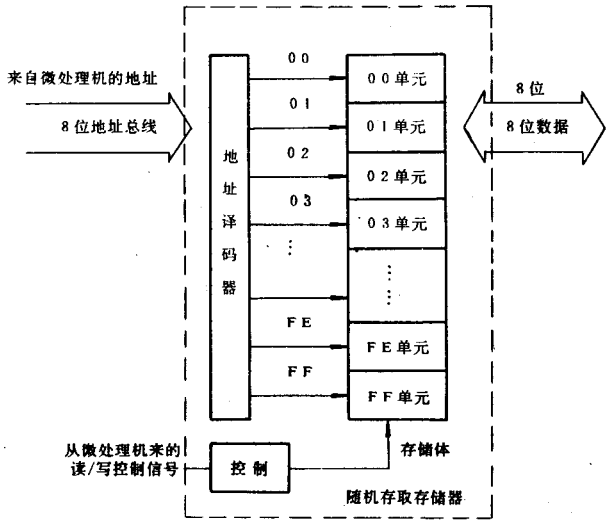


图 1-4　随机存取存储器结构简图

（三）　读写工作过程

从存储器读出信息的过程如图 1-5 所示。

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

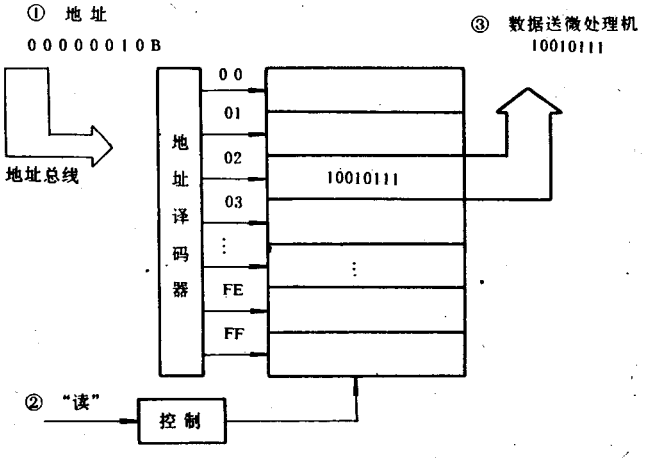


图 1-5　从存储器读出信息的过程

向存储器写入的过程如图 1-6 所示。

写入操作是将数据总线上的数据存入被选中的单元中去的。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上，由微处理器来的写控制信号，将数据写入指定的单元中。

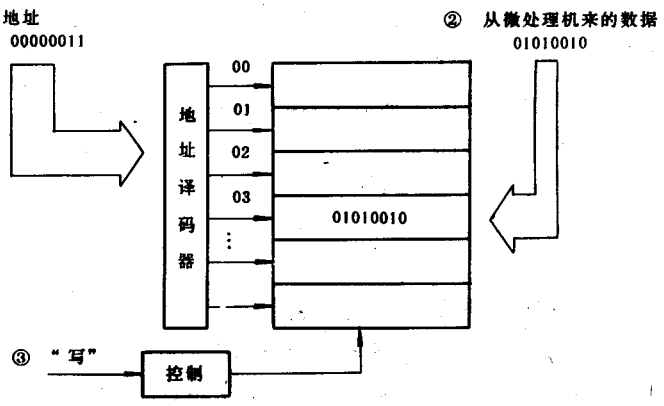


图 1-6 向存储器写入信息的过程

§1.2 计算机中的数制与码制

一、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以二进制代码来表示的。

（一） 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，因为一位十进制数只能用 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"，由于它处于十位的位置上，因而它所代表的数已不是"3"本身，它代表的是 30 或 3×10^1。同理；其中的"5"，代表的是 5×10^{-1}。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，都可以表示为

$$N=\pm[K_n\times10^n+K_{n-1}\times10^{n-1}+\cdots+K_1\times10^1+K_0\times10^0+K_{-1}\times10^{-1}+K_{-2}\times10^{-2}+\cdots+K_{-m}\times10^{-m}]$$

$$=\pm\sum_{i=-m}^{n}[K_i\times10^i]$$

其中，K_i 为 0,1,2,3,4,5,6,7,8,9。

（二） 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，在十进制数中，R＝10；在二进制中，R＝2；八进制中，R＝8；十六进制中，R＝16 等等。

R^i 称为第 i 位的位权，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 R 为基数时,$K_i=0,1,\cdots,R-1$。

例如:R=2, $K_i=0,1$

R=8, $K_i=0,1,2,\cdots,7$

R=10, $K_i=0,1,2,\cdots,9$

R=16, $K_i=0,1,2,\cdots,15$

常用的几种进位计数制的表示方法见表 1-1。

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	C
14	1110	16	E
15	1111	17	F
16	10000	20	10
⋮	⋮	⋮	⋮

(三) 各种数制间的转换

1. 十进制与其它计数制之间的转换

在十进制数与二进制数之间转换中,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。

例 1 将十进制数 26 转换成二进制数。

解

```
2 | 26        余数
  2 | 13      0 = K0
    2 | 6     1 = K1
      2 | 3   0 = K2
        2 | 1 1 = K3
          0   1 = K4
```

于是$(26)_{10}=(K_4K_3K_2K_1K_0)=(11010)_2$，其中，$(X)_n$ 表示 X 是 n 进制数。

除 2 取余法是将$(26)_{10}$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，$K_i=1$。

现在我们再验算一下：

$$(11010)_2=1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0$$
$$=16+8+0+2+0=(26)_{10}$$

把十进制纯小数转换成二进制的纯小数采用小数部分乘 2 取整法。

例 2　将十进制纯小数 0.625 转换成二进制数。

解

$$\begin{array}{rl} & 0.625 \\ \times) & \quad 2 \\ \hline & 1.250 \qquad \text{整数位为 1，即 } K_{-1}=1 \\ & 0.25 \\ \times) & \quad 2 \\ \hline & 0.50 \qquad \text{整数位为 0，即 } K_{-2}=0 \\ & 0.5 \\ \times) & \quad 2 \\ \hline & 1.0 \qquad \text{整数位为 1，即 } K_{-3}=1 \end{array}$$

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

将小数反复乘 2，若所得新数的整数部分为 1，则相应位为 1；若整数部分为 0，则相应位为 0，从小数的高位逐次向低位进行，直到满足精度要求为止。

验算：$(0.101)_2=1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=(0.625)_{10}$

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、例 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；十进制纯小数可用“乘 8 取整”法；而八进制数转换成十进制数，按用通用表达式展开求和即得。例如：

$$(1503.54)_8=1\times8^3+5\times8^2+0\times8^1+3\times8^0+5\times8^{-1}+4\times8^{-2}$$
$$=512+320+0+3+0.625+0.0625=(835.6875)_{10}$$

同样；十进制整数转换成十六进制整数可用“除 16 取余”法，十进制小数转换成十六进制小数可用“乘 16 取整”法。而十六进制数换成十进制数可用通用表达式展开求和。

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的关系，所以他们和十进制数之间的转换，可以先转换成二进制数，然后再采用二进制数与十制数的转换方法即得。

2. 二进制与八进制之间转换

因为八进制数可用 3 位二进制数表示，故二进制数转换为八进制数，可以以小数点为界，整数部分自右向左每 3 位为一组，不够位数则前加零；小数部分自左向右每 3 位为一组，不够位数则后加零，然后按二进制与八进制的规律直接转换。例如：

$$(1101001.0100111)_2=(001101001.010011100)_2=(151.234)_8$$

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：

$$(653.503)_8=(110101011.101000011)_2$$

3. 二进制与十六进制之间的转换

因为十六进制数可用 4 位二进制数表示，所以在十六进制与二进制之间也存在类似八进制与二进制之间的简单而又直接的转换规律，即以小数点为界，整数部分自右向左，每 4 位为一组，不够的位用前面加零补足；小数部分自左向右，每 4 位为一组，不够的位用后面加零补足，然后按二进制与十六进制之间的规律直接转换即可。例如：

$$(16D.44)_{16}=(000101101101.01000100)_2$$

$$(100101111.101000011)_2=(12F.A18)_{16}$$

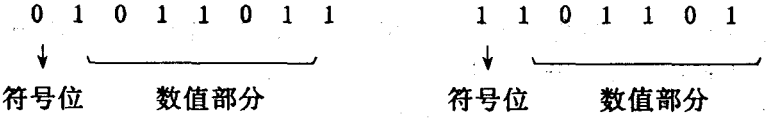
二、计算机中的码制

(一) 机器数和真值

一个数若不考虑它的符号，即为无符号数。具体的数显然有正负，用数学符号“+”或“−”表示的数，如 N=+91 或 N=−91。并把这种用“+”，“−”号表示的数，称为真值。

但是，计算机中所有的数都用 0 或 1 进行编码，机器并不认识正负号，只有将正负号也用 0 或 1 进行编码，机器才能识别。通常在符号位用“0”表示正，用“1”表示负。这种表示数的方法，称为带符号数的表示方法。

在机器中带符号数的表示方法为：



以上两数，符号位为 0 的表示$(+91)_{10}$，符号位为 1 的表示$(-91)_{10}$。一个数在机器中的表示形式，称为机器数。机器数和它的真值之间存在着一一对应的关系。

对于无符号数在机器中的表示方法，其不同之处是：它没有符号位，全部二进制位均用于表示数的大小。对八位二进制数来讲，表示范围为$(0\sim255)_{10}$。

(二) 原码、补码、反码

原码、补码和反码是带符号数的机器数的表示方法。

1. 原码表示法

前面介绍的带符号数在机器中的表示方法，实际上就是原码表示法。只要将一个数的真值的符号部分用 0 或 1 表示，即得到该数的原码。例如 X=+1001010，因 X>0，所以其原码$[X]_{原}=01001010$。正数原码的符号位用 0 表示。又如 X=−1001010，因 X<0，所以其原码$[X]_{原}=11001010$。负数原码的符号位用 1 表示。由上面原码的表示形式，可将原码定义为

$$[X]_{原}=\begin{cases}X & ; \quad 当\ 0\leqslant X<2^{n-1}\\ 2^{n-1}-X & ; \quad 当\ -2^{n-1}<X\leqslant 0\end{cases}$$

其中，n 为二进制数的位数，其模为 2^n，n=4 时，$2^4=16$，即模 mod 2^n 为 16。若 4 位全部用来表示数，其范围为 0～15，即 $0\sim2^n-1$。由于符号位占用 1 位，所以其范围将减少，X 是正数时，其范围为+0～+7，即 $0\leqslant X<2^{4-1}$或 $0\leqslant X<2^3$。当 X 为负数时，其范围为 0～−7，即 $-2^{4-1}<X\leqslant0$ 或 $-2^3<X\leqslant0$。当 =8 时，原码表示的数其范围为 −127～+127，即

$(11111111)_2 \sim (01111111)_2$。

其中的零有正零和负零两种情况，机器中均作零处理：

$$[+0]_{原}=0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0$$

符号位　　n－1个0

$$[-0]_{原}=1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0$$

符号位　　n－1个0

一般采用正零表示法。

2. 补码表示法

我们知道两个正数相加，其和为正数，两个负数相加，其和为负数，而当一个正数与一个负数相加时，必须做减法，和的符号位与绝对值较大的那个数的符号相同。为了确定和的符号，必须判断两个数的绝对值的大小，这样将使求和的过程及控制线路变得更加复杂。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，从而使问题的解决变得简单。

在日常生活中，补码的例子是很多的，时钟的校对就是其中一例。设标准时间是6点正，而时针指向10点正。要将时针拨到6点正，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。

这儿倒拨与顺拨的效果是相同的，是互相补充的，即在时钟以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。

显然，时间为6点正与时间为18点正，时针的位置是相同的，当然也可以表示为：

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

6与18对模12是同余的。

但在模12情况下，18也只能用6点表示。因此，对正数讲，其补码就是它自身，但对负数讲，例如12＋(－4)＝8，即模加负数的结果比模小，因而负数可以找到一个比模小的同余数，它就是其补码。至此，我们可以对模2^n系统的二进制补码定义如下：

$$[X]_{补}=\begin{cases} X & ;\quad 0\leqslant X<2^{n-1} \\ 2^n+X & ;\quad -2^{n-1}<X\leqslant 0 \end{cases}$$

其中，n为二进制数的位数，当n＝8时，补码表示的数的范围是－128～＋127。

由于正数的补码就是它自身，不用另求；而对负数的补码，则需通过减法来求得，这是很不方便的。例如：

$X=(+1000000)_2$，因$X>0$，所以$[X]_{补}=(01000000)_2$。

$X=(-0001010)_2$，因$X<0$，所以$[X]_{补}=2^8-(0001010)_2=(11110110)_2$。

对负数的补码，其最高位是符号位，取值为1，而具体的数值位(1110110)恰好为其原码值(0001010)按位求反再加1，即$[X]_{补}=11110101+1=11110110$。