计 算 机 系 列 教 材

汇编语言程序设计

主编 王先水 吴 蓓 章 玲

WUHAN UNIVERSITY PRESS
武汉大学出版社

计 算 机 系 列 教 材

汇编语言程序设计

主编 王先水 吴 蓓 章 玲
主审 刘永祥

WUHAN UNIVERSITY PRESS
武汉大学出版社

图书在版编目(CIP)数据

汇编语言程序设计/王先水,吴蓓,章玲主编.—武汉:武汉大学出版社,2012.1
计算机系列教材
ISBN 978-7-307-09422-2

Ⅰ.汇… Ⅱ.①王… ②吴… ③章… Ⅲ.汇编语言—程序设计—教材 Ⅳ.TP313

中国版本图书馆 CIP 数据核字(2011)第 282866 号

责任编辑:林 莉 责任校对:黄添生 版式:支 笛

出版发行:武汉大学出版社 (430072 武昌 珞珈山)
(电子邮件:cbs22@whu.edu.cn 网址:www.wdp.com.cn)
印刷:湖北金海印务有限公司
开本:787×1092 1/16 印张:20 字数:509 千字
版次:2012 年 1 月第 1 版 2012 年 1 月第 1 次印刷
ISBN 978-7-307-09422-2/TP·421 定价:35.00 元

前 言

“汇编语言程序设计”是工科院校计算机专业及相关专业一门重要的专业技术基础课程。在众多程序设计语言中，汇编语言属于低级语言，“低级”是指在面向用户方面；在面向机器方面，汇编语言是其他高级语言如 C++、JAVA 等无法比拟的。汇编语言可以充分发挥计算机的硬件特性，编写出时间和空间要求很高的程序。在实时控制场合，汇编语言更是无可替代。它借助计算机硬件知识，重点讲解汇编语言程序设计的方法与思想。本课程可以帮助学生掌握计算机硬件组成知识及汇编语言程序设计的方法，建立计算机体系结构的基本思想，为学生后续学习软件、硬件课程作好铺垫。

本书在编写过程中重视基础，循序渐进，内容精炼，重点突出，融入学科方法论内容和科学理念，反映计算机技术发展前沿，倡导理论联系实际和科学的思想方法，体现学科知识组织的层次结构。把握程序设计方法与思路，注重程序设计实践训练，引入典型的程序设计案例，将程序设计课程的学习融入案例的研究和解决过程中，以学生实际编程中解决问题的能力为突破口，注重程序设计算法的实现。从而使学生接受一次程序设计基本功的严格训练，培养学生良好的程序设计风格与严密的逻辑思维能力，提高学生分析问题、解决问题的能力及创新设计能力，为今后研制、开发各种计算机软件打下良好而坚实的基础。

本教材作为计算机系列教材之一，在内容的选择、概念的引入、案例设计与分析、文字描述等方面，都遵循面向应用、重视实践、方便教学的原则，符合人们实践—理论—实践的认知规律。以 Intel 8086/8088 系列微机为基础机型讲解汇编语言程序设计基础知识、8086/8088CPU 寻址方式及指令系统的基础上，详细讲解汇编语言程序设计的基本方法和基本思想。第 1、2 章提供学习汇编语言的基础知识，第 3、4 章重点讲解汇编语言程序设计的基本方法，第 5 章讲解汇编语言程序设计模块设计方法，第 6 章讲解汇编语言输入输出程序设计方法，第 7 章讲解文件的存储读写程序设计方法，第 8 章介绍汇编语言同 C++语言混合编程的基本方法。

本教材在编写过程中得到了中国地质大学江城学院计算机系领导和教师的大力支持。在系主任刘永祥教授的指导下，由王先水、吴蓓、辛玲三位教师共同编写完成。在编写过程中采用编者长期使用的讲稿，并参考了相关书籍，在此对相关作者表示诚挚的谢意。由于编者水平有限，书中难免存在疏漏，敬请同行专家批评指正。

作 者

2011 年 11 月

目 录

第1章 基础知识

【学习目标】

（1）汇编语言及认识汇编语言源程序。

（2）计算机中的数及字符的表示。

（3）8086 微处理器的内部结构、寄存器的功能、存储器的组织。

（4）Intel 80X86 系列微处理器简介。

1.1 汇编语言概述

汇编语言是面向机器的程序设计语言。在汇编语言中，用助记符代替操作码，用地址符号或标号代替地址码。这种用符号代替机器语言的二进制码，就把机器语言变成了汇编语言，因而汇编语言亦称为符号语言。使用汇编语言编写的程序，机器不能直接识别，需要一种程序将汇编语言翻译成机器语言，这种起翻译作用的程序叫汇编程序，汇编程序是系统软件中语言处理系统软件。汇编程序把汇编语言翻译成机器语言的过程称为汇编。

汇编语言是一种面向机器的语言，不同 CPU 的计算机，其汇编语言都不相同。要学习好汇编语言，首先应了解并掌握该汇编语言的计算机硬件结构、数据类型及其在计算机内的表示方法。本书从 8086/8088CPU 的硬件结构、寻址方式、指令系统、程序设计方法和输入输出等内容进行介绍。由于 8086/8088CPU 已被技术淘汰了，但目前奔腾系列微处理器具有向下兼容的特性，因此 8086/8088CPU 机器仍然是学习汇编语言的基础，学好该机器的汇编语言也很容易拓展到 80X86 的机器甚至更高档的机器。

汇编程序同汇编语言程序是两个不同的概念。在今后的学习过程中要认真去体会并加以理解。

1.1.1 汇编语言源程序

汇编语言源程序就是用汇编语言的语句编写的程序，它是不能被机器所识别的。可以用任何一种文本编辑器进行编辑但应保存为.ASM 文件类型。如纯文本编辑器 EDIT 编辑的汇编语言源程序（保存时属性应为.ASM 的源文件或汇编语言源程序的扩展名为.ASM）。汇编语言源程序要比用机器指令编写的程序容易理解和维护，但必须通过汇编程序对其汇编成机器能识别的机器指令。

首先我们一起来认识一个完整的汇编语言源程序。利用纯文本编辑如 EDIT 编写的程序内容如下：

```
DATA SEGMENT                ；定义数据段，段名是 DATA 用于存放程序中的数据
    BUF1 DB 5               ；定义变量 BUF1，其类型是字节，其值为 5
    BUF2 DB 2               ；定义变量 BUF2，其类型是字节，其值为 2
```

```
DATA ENDS                          ; 数据段结束
STACK SEGMENT STACK                ; 定义堆栈段，段名 STACK 用于保护程序中的数据
    DB 100 DUP(?)                  ; 堆栈段的大小为 200 个字节且每个单元初值为 0
STACK ENDS                         ; 堆栈段结束
CODE SEGMENT                       ; 定义代码段，段名为 CODE 用于存放程序中的代码
ASSUME DS:DATA,SS:STACK,CS:CODE    ; 建立程序中各段与段寄存器的联系使 CPU
                                     能访问
START:MOV AX,DATA                  ; START 是代码存放的起地址。将 DATA 段地址送 AX
    MOV DS,AX                      ; 将 AX 的内容送到段寄存器 DS 中
    MOV DL,BUF1                    ; 将变量 BUF1 的值传送到寄存器 DL 中
    MOV BL,BUF2                    ; 将变量 BUF2 的值传送到寄存器 BL 中
    ADD DL,BL                      ; 将寄存器 DL 的内容与 BL 的内容相加结果存在 DL 中
    ADD DL,30H                     ; 将寄存器 DL 中的数据加 30H 转换成 ASCII 码
    MOV AH,2                       ; 调用 DOS 系统 2 号功能，显示 DL 中的数据
    INT 21H
    MOV AH,4CH                     ; 调用 DOS 系统 4CH 号功能，退出 DOS
    INT 21H
CODE ENDS                          ; 代码段结束
    END START                      ; 程序从“START”始地址开始执行到此结束
```

该程序实现了一个计算 5+2 的和并在屏幕上打印其结果。该程序相对 C 语言程序来说较复杂，这是由汇编语言源程序的基本结构来体现的。实现功能要求的核心语句如下：

```
MOV DL,BUF1                        ; 将变量 BUF1 的值传送到寄存器 DL 中
MOV BL,BUF2                        ; 将变量 BUF2 的值传送到寄存器 BL 中
ADD DL,BL                          ; 将寄存器 DL 的内容与 BL 的内容相加结果存在 DL 中
ADD DL,30H                         ; 将寄存器 DL 中的数据加 30H 转换成 ASCII 码
MOV AH,2                           ; 调用 DOS 系统 2 号功能，显示 DL 中的数据
INT 21H
```

其余的是汇编语言源程序的基本框架语句。

1.1.2 机器语言

机器指令是 CPU 能直接识别并执行的指令，它的表现形式是二进制编码即 0 和 1 所组成的代码。机器指令通常由**操作码**和**操作数**两部分组成，**操作码**指出该指令所要完成的操作，即指令的功能，**操作数**指出参与运算的对象，以及运算结果所存放的位置等。

由于机器指令与 CPU 紧密相关，所以，不同种类的 CPU 所对应的机器指令也就不同，而且它们的指令系统往往相差很大。但对同一系列的 CPU 来说，为了满足各型号之间具有良好的兼容性，要求做到：新一代 CPU 的指令系统必须包括先前同系列 CPU 的指令系统。只有这样，先前开发出来的各类程序在新一代 CPU 上才能正常运行，即同类型的 CPU 具有兼容性。

机器语言是机器指令的集合。机器语言是用来直接描述机器指令、并遵循机器指令的规则。它是 CPU 能直接识别的唯一种语言，也就是说，CPU 能直接执行用机器语言描述的

程序。

1.1.3 汇编语言

虽然用机器语言编写程序有很高的要求和许多不便，但编写出来的程序执行效率高，CPU严格按照程序员的要求去做，没有多余的额外操作。所以，在保留“程序执行效率高”的前提下，人们就开始着手研究一种能大大改善程序可读性的编程方法。

为了改善机器指令的可读性，选用了一些能反映机器指令功能的单词或词组来代表该机器指令，而不再关心机器指令的具体二进制编码。与此同时，也把CPU内部的各种资源符号化，使用该符号名也等于引用了该具体的物理资源。

如此一来，令人难懂的二进制机器指令就可以用通俗易懂的、具有一定含义的符号指令来表示了，于是，汇编语言就有了雏形。现在，我们称这些具有一定含义的符号为助记符，用指令助记符、符号地址等组成的符号指令称为**汇编格式指令(或汇编指令)**。

汇编语言是汇编指令集、伪指令集和使用它们规则的统称。汇编指令集是CPU机器指令的符号化，如加法指令ADD。伪指令是在程序设计时所需要的一些辅助性说明指令，它不对应具体的机器指令，如段定义伪指令SEGMENT。有关汇编指令及伪指令在以后的章节中详细叙述。

1.1.4 高级语言

由于汇编语言依赖于硬件体系，且助记符量大难记，于是人们又发明了更加易用的所谓高级语言。在这种语言下，其语法和结构更类似普通英文，且由于远离对硬件的直接操作，使得一般人经过学习之后都可以编程。高级语言可以分为命令式语言（比如Fortran、Pascal、Cobol、C、C++、Basic、Ada、Java、C#）、函数式语言(如Lisp、Haskell、ML、Scheme)和逻辑式语言（例如Prolog）。

虽然各种语言属于不同的类型，但它们各自都不同程度地对其他类型的运算模式有所支持。

1.1.5 三种语言特点比较

1. 机器语言——面向机器的语言

机器语言是最底层的计算机语言。用机器语言编写的程序，计算机硬件可以直接识别。在用机器语言编写的程序中，每一条机器指令都是二进制形式的指令代码。对于不同的计算机硬件(主要是CPU)，其机器语言是不同的，因此，针对一种计算机所编写的机器语言程序不能在另一种计算机上运行。由于机器语言程序是直接针对计算机硬件所编写的，因此它的**执行效率比较高，能充分发挥计算机的速度性能**。但是，用机器语言编写程序的难度比较大，容易出错，而且程序的直观性比较差，也不容易移植。

2. 汇编语言——面向机器的语言

汇编语言与机器语言一般是一一对应的，因此，汇编语言也是与具体使用的计算机有关的。由于汇编语言采用了助记符，因此，它比机器语言直观，容易理解和记忆，但是，计算机仍不能直接识别用汇编语言编写的程序，需要用汇编程序对其进行汇编成机器能识别并能执行的机器指令，依赖于计算机硬件。

3. 高级语言——面向问题、面向对象的语言

高级语言就是算法语言，它不是面向机器的，而是面向问题的，不依赖于具体机器，具有良好的通用性。高级语言的表达方式接近于被描述的问题，又由于接近于自然语言和数学语言，从而易于为人们接受掌握和书写。高级语言的显著特点是独立于具体的计算机硬件，具有较好的通用性和可移植性。

1.2 计算机中数和字符的表示

计算机中的数据分为数值数据、非字符数据。在 8086 汇编语言中，常用的数值数据为二进制数、十进制数、十六进制数或 BCD 码。数值数据分为有符号数和无符号数两种，均以其补码表示。在计算机内，所有的数（含地址）均采用二进制形式，类型有字节、字、双字等。非字符数据主要有表示字符的 ASCII 码和表示汉字的变形国际码。对数值数据和非字符数据人们按约定用二进制数来表示，从而使计算机能识别并处理。

1.2.1 不同进制的数及相互间的转换

首先，我们要理解几个概念。

进位制：表示数时，仅用一位数码往往不够用，必须用进位计数的方法组成多位数码。多位数码每一位的构成以及从低位到高位的进位规则称为进位计数制，简称进位制。

基数：进位制的基数，就是在该进位制中可能用到的数码个数。

位权：在某一进位制的数中，每一位的大小都对应着该位上的数码乘上一个固定的数，这个固定的数就是这一位的权数。权数是一个幂。

1. 常用的几种数制

（1）十进制。数码为 0～9，基数是 10。运算规则：逢十进一，即 9+1=10。十进制数的权展开式，如

$(5555)_{10}=5\times10^3+5\times10^2+5\times10^1+5\times10^0$

10^3、10^2、10^1、10^0称为十进制的权，各数位的权是 10 的幂，同样的数码在不同的数位上代表的数值不同，任意一个十进制数都可以表示为各个数位上的数码与其对应的权的乘积之和，称**权展开式**。

又如：$(209.04)_{10}=2\times10^2+0\times10^1+9\times10^0+0\times10^{-1}+4\times10^{-2}$

（2）二进制。数码为 0、1，基数是 2。运算规则：逢二进一，即 1+1=10。二进制数的权展开式，如：

$(101.01)_2=1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$

二进制数只有 0 和 1 两个数码，它的每一位都可以用电子元件来实现，且运算规则简单，相应的运算电路也容易实现。

（3）八进制。数码为 0～7，基数是 8。运算规则：逢八进一，即 7+1=10。八进制数的权展开式，如：

$(207.04)_8=2\times8^2+0\times8^1+7\times8^0+0\times8^{-1}+4\times8^{-2}$

（4）十六进制。数码为 0～9、A～F，基数是 16。运算规则：逢十六进一，即 F+1=10。十六进制数的权展开式，如：

$(D8.A)_{16}=13\times16^1+8\times16^0+10\times16^{-1}$

2. 数制间的转换

同一个数可采用不同的计数体制来表示，那么各种数制表示的数是可以相互转换的。数制转换指一个数从一种进位制表示形式转换成等值的另一种进位制表示形式，其实质为权值转换。数制相互转换的原则为转换前后两个有理数的整数部分和小数部分必定分别相等。下面将介绍数制间的转换。

（1）二进制、八进制、十六进制数转换为十进制数。

转换规则：分别写出二进制、八进制、十六进制数按权展开式，数码和位权值的乘积称为加权系数。各位加权系数相加的结果便为对应的十进制数。如：

$(101.01)_2=1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}=(5.25)_{10}$

$(207.04)_8=2\times8^2+0\times8^1+7\times8^0+0\times8^{-1}+4\times8^{-2}=(135.0625)_{10}$

$(D8.A)_{16}=13\times16^1+8\times16^0+10\times16^{-1}=(216.625)_{10}$

（2）十进制数转换为二进制数。

整数和小数转换方法不同，因此必须分别进行转换，然后再将两部分转换结果合并得到完整的目标数制形式。

转换规则：整数部分采用除 2 取余法：先得到的余数为低位，后得到的余数为高位。小数部分采用乘 2 取整法：先得到的整数为高位，后得到的整数为低位。

【例题 1.1】将十进制数 44.375 转换为二进制数可分别采用整数部分除 2 取余法，小数部分乘 2 取整法，其求解过程如图 1-1、图 1-2 所示。

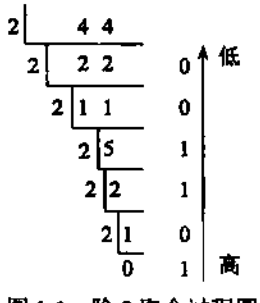

图 1-1 除 2 取余过程图

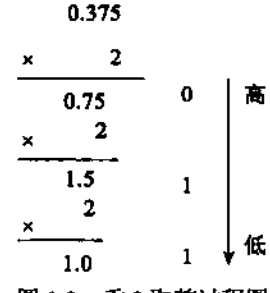

图 1-2 乘 2 取整过程图

所以：$(44.375)_{10}=(101100.011)_2$

同理：可采用同样的方法将十进制数转成八进制、十六进制数，但由于八进制和十六进制的基数较大，做乘除法不是很方便，因此需要将十进制转成八进制、十六进制数时，通常是将其先转成二进制，然后再将二进制转成八进制、十六进制数。

（3）二进制数转换为八进制、十六进制数。

①二进制数转换成八进制数。

将二进制数转换成八进制数时，首先从小数点开始，将二进制数的整数和小数部分每三位分为一组，不足三位的分别在整数的最高位前和小数的最低位后加“0”补足，然后每组用等值的八进制码替代，即得目的数。反之，则可将八进制数转换成二进制数。

如（11010111.0100111）$_2$=（?）$_8$得 （11010111.0100111）$_2$=（327.234）$_8$

②二进制数转换成十六进制数。

与上述相类似，由于十六进制基数 R=16=2^4，故必须用四位二进制数构成一位十六进

制数码，同样采用分组对应转换法，所不同的是此时每四位为一组，不足四位同样用“0”补足。

如（111011.10101）$_2$=（?）$_{16}$故有（111011.10101）$_2$=（3B.A8）$_{16}$

各种数制形式之间的转换方法中，最基本的是十进制与二进制之间的转变，八进制和十六进制可以借助二进制来实现相应的转换。

1.2.2 BCD 码

计算机系统只能识别 0 和 1，怎样才能表示更多的数码、符号、字母呢？用编码可以解决此问题。

用一定位数的二进制数来表示十进制数码、字母、符号等信息称为编码。

用以表示十进制数码、字母、符号等信息的一定位数的二进制数称为代码。

二十进制代码：用 4 位二进制数 $b_3b_2b_1b_0$ 来表示十进制数中的 0～9 十个数码，简称 BCD 码。

用四位自然二进制码中的前十个码字来表示十进制数码，因各位的权值依次为 8、4、2、1，故称 8421-BCD 码。十进制的 10 个数码对应的 BCD 码编码是：

十进制数码：	0	1	2	3	4	5	6	7	8	9
BCD 码编码：	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

将一个十进制数用 BCD 码表示，只要把十进制数的各位数字用对应的 BCD 码编码来表示即可。根据存储方式的不同，BCD 码分为两种：压缩存储的 BCD 码和非压缩存储的 BCD 码。压缩存储的 BCD 码是指用一个字节单元来存放两个十进制数；非压缩存储的 BCD 码是指用一个字节单元来存放一个十进制数且存放于低半字节而高半字节补 0。

值得注意的是，由于十进制数最大是 9，不存在十六进制数的 A-F。因此用 BCD 码表示十进制数时就不存在 1010-1111 这 6 个码，连续的数据在计算机中所存储的数值是不连续的。由于这种不连续性，BCD 码与二进制码是有本质区别的。

例如：10001001 BCD=89D

10001001 B=$2^7+2^3+2^0$=137D

【例题 1.2】画出十进数 1936 在计算机内存中以压缩的 BCD 码和非压缩 BCD 码存储示意图。

根据压缩 BCD 码和非压缩 BCD 码的要求可知：以压缩形式存储需要 2 个字节存储单元；以非压缩形式存储需要 4 个字节存储单元。其存储结构如图 1-3、图 1-4 所示。

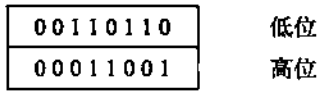

图 1-3 压缩 BCD 码存储结构

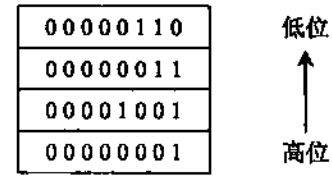

图 1-4 非压缩 BCD 码存储结构

1.2.3 ASCII 码

ASCII（American Standard Code for Information Interchange，美国信息互换标准代码）是基于拉丁字母的一套电脑编码系统。它主要用于显示现代英语和其他西欧语言。它是现今最通用的单字字节编码系统，并等同于国际标准 ISO/IEC 646。

ASCII 码使用指定的 7 位或 8 位二进制数组合来表示 128 或 256 种可能的字符。标准 ASCII 码也叫基础 ASCII 码，使用 7 位二进制数来表示所有的大写和小写字母，数字 0 到 9，标点符号，以及在美式英语中使用的特殊控制字符。其中：0～31 及 127(共 33 个)是控制字符或专用字符（其余为可显示字符），如控制符：LF（换行）、CR（回车）、FF（换页）、DEL（删除）、BS（退格）、BEL（振铃）等；通信专用字符：SOH（文头）、EOT（文尾）、ACK（确认）等；ASCII 值为 8、9、10 和 13 分别转换为退格、制表、换行和回车字符。它们并没有特定的图形显示，但会依不同的应用程序，而对文本显示有不同的影响。32～126（共 95 个）是字符（32sp 是空格），其中 48～57 为 0 到 9 十个阿拉伯数字；65～90 为 26 个大写英文字母，97～122 为 26 个小写英文字母，其余为一些标点符号、运算符号等。

同时还要注意，在标准 ASCII 中，其最高位（b_7）用作奇偶校验位。所谓奇偶校验，是指在代码传送过程中用来检验是否出现错误的一种方法，一般分奇校验和偶校验两种。奇校验规定：正确的代码一个字节中 1 的个数必须是奇数，若非奇数，则在最高位 b_7 添 1；偶校验规定：正确的代码一个字节中 1 的个数必须是偶数，若非偶数，则在最高位 b_7 添 1。

后 128 个称为扩展 ASCII 码，目前许多基于 x86 的系统都支持使用扩展(或“高”)ASCII。扩展 ASCII 码允许将每个字符的第 8 位用于确定附加的 128 个特殊符号字符、外来语字母和图形符号。

熟悉一些常用字符的 ASCII 码对学习汇编语言程序设计是十分必要的。每个字符的 ASCII 码用一个两位的十六进制数表示。

如：空格的 ASCII 码是 20H，数字 0～9 字符的 ASCII 码分别是 30H～39H，大写字母 A～Z 字符的 ASCII 码分别是 41H～5AH，小写字母 a～b 字符的 ASCII 码分别是 61H～7AH。其余字符的 ASCII 参照书附录 A。

1.2.4 原码、反码和补码

在计算机中，数值数据有两种表示法：定点表示法和浮点表示法。浮点表示法比定点表示法所表示的数的范围大、精度高。但因为 8086 微处理器处理的数据小数点位置是固定的，属定点数，则对浮点数的运算是与其配套的浮点部件来实现的。浮点部件具有浮点数值运算的功能并提供相应的指令系统。因此本教材只讨论定点数的运算即整数。对于有符号数一律采用其补码表示。

1. 原码表示法

原码表示法是机器数的一种简单的表示法。其符号位用 0 表示正号，用 1 表示负号，数值一般用二进制形式表示。设有一数为 X，则原码表示可记作 $[X]_{原}$。

例如，X_1= +1010110　X_2= −1001010

其原码记作：$[X_1]_{原}=[+1010110]_{原}=01010110$　$[X_2]_{原}=[-1001010]_{原}=11001010$

原码表示数的范围与二进制位数有关。当用 8 位二进制来表示小数原码时，其表示范围：最大值为 0.1111111，其真值约为（0.99）$_{10}$

最小值为 1.1111111，其真值约为（-0.99）$_{10}$

当用 8 位二进制来表示整数原码时，其表示范围：

最大值为 01111111，其真值为（127）$_{10}$

最小值为 11111111，其真值为（-127）$_{10}$

在原码表示法中，对 0 有两种表示形式：[+0]$_{原}$= 00000000　　[-0]$_{原}$=10000000

2. 补码表示法

机器数的补码可由原码得到。如果机器数是正数，则该机器数的补码与原码一样；如果机器数是负数，则该机器数的补码是对它的原码（除符号位外）各位取反，并在末位加 1 而得到的。设有一数 X，则 X 的补码表示记作 [X]$_{补}$。

例如，[X_1] =+1010110　　[X_2] =-1001010

[X_1]$_{原}$=01010110　[X_1]$_{补}$=0101011　即 [X_1]$_{原}$= [X_1]$_{补}$=01010110

[X_2]$_{原}$= 11001010　[X_2]$_{补}$=10110101+1＝10110110

补码表示数的范围与二进制位数有关。当采用 8 位二进制表示时，小数补码的表示范围：

最大为 0.1111111，其真值为（0.99）$_{10}$

最小为 1.0000000，其真值为（-1）$_{10}$

采用 8 位二进制表示时，整数补码的表示范围：

最大为 01111111，其真值为（127）$_{10}$

最小为 10000000，其真值为（-128）$_{10}$

在补码表示法中，0 只有一种表示形式：[+0]$_{补}$=00000000[-0]$_{补}$=11111111+1=00000000（由于受设备字长的限制，最后的进位丢失）所以有 [+0]$_{补}$= [-0]$_{补}$=00000000

3. 反码表示法

机器数的反码可由原码得到。如果机器数是正数，则该机器数的反码与原码一样；如果机器数是负数，则该机器数的反码是对它的原码（符号位除外）各位取反而得到的。设有一数 X，则 X 的反码表示记作 [X]$_{反}$。

例如：X_1= +1010110　X_2= -1001010

[X_1]$_{原}$=01010110　　[X_1]$_{反}$= [X_1]$_{原}$=01010110

[X_2]$_{原}$=11001010　[X_2]$_{反}$=10110101

反码通常作为求补过程的中间形式，即在一个负数的反码的末位上加 1，就得到了该负数的补码。

【例题 1.3】 已知 [X]$_{原}$=10011010，求 [X]$_{补}$。

分析如下：由 [X]$_{原}$求 [X]$_{补}$的原则是：若机器数为正数，则 [X]$_{原}$= [X]$_{补}$；若机器数为负数，则该机器数的补码可对它的原码（符号位除外）所有位求反，再在末位加 1 而得到。

现给定的机器数为负数，故有 [X]$_{补}$=[X]$_{反}$+1，即 [X]$_{原}$=10011010　[X]$_{反}$=11100101+1

[X]$_{补}$=11100110

【例题 1.4】 已知 [X]$_{补}$=11100110，求 [X]$_{原}$。

分析如下：对于机器数为正数，则 [X]$_{原}$= [X]$_{补}$；对于机器数为负数，则有 [X]$_{原}$= [[X]$_{补}$]$_{补}$。

现给定的为负数，故有：[X]$_{补}$=11100110　[[X]$_{补}$]$_{补}$= [[X]$_{补}$]$_{反}$+1=10011001+1

[[X]$_{补}$]$_{补}$=10011010= [X]$_{原}$