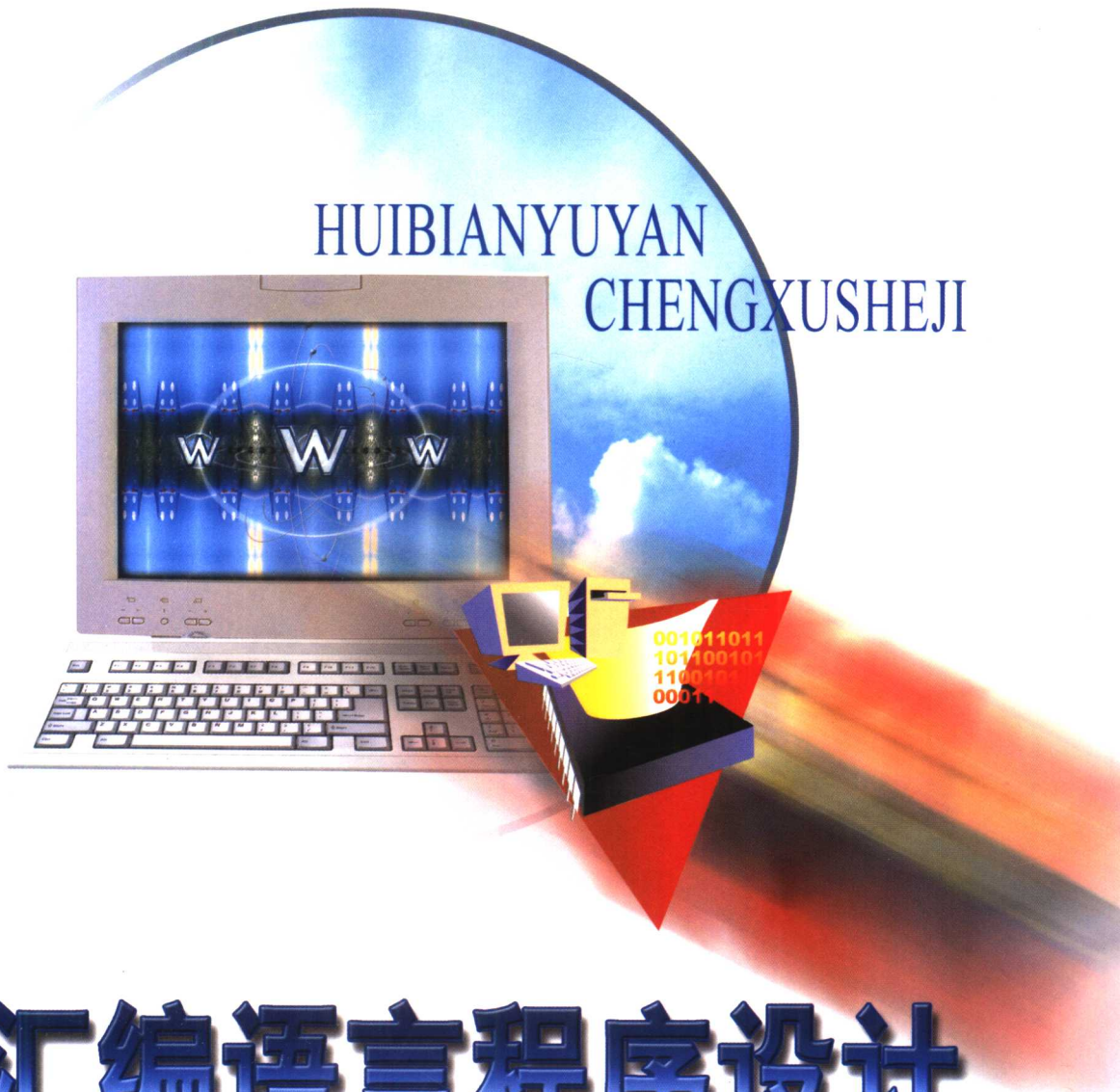
十五规划计算机系列教材




# 汇编语言程序设计

主　编：刘　念
副主编：赵美虹　李顺增
崔广才　潘广贞

兵器工业出版社

# 汇编语言程序设计

主　编　刘　念
副主编　赵美虹　李顺增
　　崔广才　潘广贞

兵器工业出版社

## 内 容 简 介

本书共分 10 章，第 1 章介绍汇编语言的基础知识，第 2 章介绍 8086 计算机的基本结构，第 3 章介绍寻址方式和 8086 指令系统，第 4 章介绍汇编程序的基本框架以及程序开发过程，第 5 章讲述汇编语言源程序的开发以及开发环境，第 6 章主要介绍程序设计的基本步骤和结构化程序设计的基本结构，第 7 章介绍输入/输出程序设计的基本方法和中断调用方法，第 8 章重点讲述 DOS 功能调用和 BIOS 调用方法，第 9 章简要介绍了彩色图形程序设计，第 10 章介绍了发声系统的程序设计。本书以 8086CPU 为背景，由浅入深地介绍了使用汇编语言及程序设计的方法，通过本书的学习，可以帮助读者初步了解计算机的结构和掌握程序设计技巧。

本书可供高等院校计算机以及相关专业的师生作为教材使用，也可供工程技术人员作为参考书使用。

**图书在版编目（CIP）数据**

汇编语言程序设计 / 刘念主编．—北京：兵器工业出版社，2004.8
ISBN 7-80172-283-3

Ⅰ. 汇… Ⅱ. 刘… Ⅲ. 汇编语言－程序设计 Ⅳ. TP313

中国版本图书馆 CIP 数据核字（2004）第 076937 号

出版发行：兵器工业出版社　　封面设计：李　晖
责任编辑：莫丽珠　　责任校对：郭　芳
邮编社址：100089　北京市海淀区车道沟 10 号　　责任印制：王京华
经　　销：各地新书店　　开　　本：787×1092　1/16
印　　刷：北京市登峰印刷厂　　印　　张：19.5
版　　次：2006 年 2 月第 1 版第 2 次印刷　　字　　数：498 千字
印　　数：1851－3850　　定　　价：36.00 元

# 计算机系列教材编辑工作委员会

# 前　言

随着计算机技术的发展和各种软件开发平台的不断涌现，目前完全用汇编语言实现的软件系统已经极为罕见。汇编语言也难以胜任大型软件系统的开发。然而，汇编语言的最大特点就是能充分发挥计算机硬件的特性，汇编语言可以实现高级语言难以胜任甚至无法完成的任务。尤其适合用于对时空效率要求较高，与机器硬件密切相关的软件的开发，例如：操作系统的部分核心代码，要求能快速响应的实时系统等。

汇编语言与程序设计作为高等院校计算机及相关专业本科生的技术基础课，是学习操作系统和编译原理等课程的基础，同时，掌握汇编语言知识，对于提高程序设计水平，加深对计算机系统的理解也是非常重要的。

全书共分 10 章，第 1 章介绍了学习汇编语言所需的基础知识，主要是数制以及数制之间的转换和位的基本运算；第 2 章介绍了 8086 计算机的基本结构，重点是 CPU 中寄存器的结构；第 3 章详细介绍了寻址方式和 8086 指令系统，并给出了大量的指令使用实例；第 4 章介绍了汇编程序的基本框架以及程序开发过程，以汇编语句的格式为主，分别介绍了汇编语言的运算符和操作符以及伪指令；第 5 章结合具体实例，讲述了汇编语言源程序的开发以及开发环境，使学生学习完前段内容后就可进行上机练习；第 6 章主要介绍了程序设计的基本步骤，介绍了顺序、分支、循环、过程、宏汇编技术的程序设计方法和技巧；第 7 章介绍了输入/输出程序设计的基本方法和中断调用方法；第 8 章重点讲述 DOS 功能调用和 BIOS 调用方法，使读者对常用外设的编程方法有一定的了解；第 9 章和第 10 章属于提高性的要求，分别介绍了彩色图形和发声系统的程序设计。

通过本书的学习，读者可以系统地学习 8086 汇编语言程序设计的基本技术，这一部分内容自成体系，无需其他先修知识，由于时间仓促，书中难免会有错处，敬请各位读者多多指出，我们将表示由衷的谢意。

编　者

2004 年 6 月

# 目　录

# 第1章 基础知识

本章介绍学习汇编语言程序设计所必须具备的基本知识，主要包括汇编语言的基本概念及计算机中数据的表示方法。通过本章的学习，读者应了解什么是汇编语言、汇编语言的特点和意义、数据的组织（字节、字和双字）、带符号数的二进制补码表示、BCD 码以及基本位操作等。尤其要深刻理解：对于一个二进制数，其具体含义依赖于使用者的解释。

## 1.1 汇编语言概述

汇编语言是机器语言的符号表示形式。在汇编语言出现之前，计算机使用机器语言来控制计算机的各种动作。因为构成计算机硬件本身的各个部件是基于二值逻辑的，这些部件只能识别 0 和 1 两个状态，计算机能直接识别并进行处理的只有 0、1 组成的二进制代码。其功能就是记忆、传输和加工二进制信息 0 或 1。所谓机器语言，就是用“0”和“1”所组成的一串二进制数所表示的命令或数据，机器的硬件可以直接识别和执行，不需要进行翻译。机器语言的特点是命令代码效率高，但不容易记忆，不利于推广和使用；程序员借助机器语言编程时，要用一串由 0 和 1 所组成的二进制数值表示指令和地址，不但费时费力，而且容易出错。汇编语言将机器语言指令和地址符号化，程序员只需要记住符号名并用其编程，汇编器负责把汇编程序翻译成机器指令和正确的地址数值。

### 1.1.1 汇编语言

汇编语言是为了克服机器语言难以记忆、表达和阅读的缺点，人们将机器语言中的指令符号化，以直观、便于记忆的符号来表示指令，这些符号被称作指令助记符（Mnemonic）。例如，用 ADD 表示加法指令，MOV 表示传送指令，MUL 表示乘法指令等。

以助记符描述的指令称作汇编格式指令或符号指令，通常简称指令。指令和伪指令的集合及其程序设计规则便构成了汇编语言。伪指令的概念将在第 3 章介绍。用汇编语言编写的程序就是汇编语言源程序。

利用汇编语言，计算表达式 67 * 15 – 10 的程序代码如下：

```
MOV     AL，67
MOV     BL，15
MUL     BL
SUB     AX，10
```

显然，用汇编语言编写的程序要比机器代码更容易理解。

然而，汇编语言仅仅是机器语言的符号化，每条汇编语言指令均对应惟一的机器指令，因而与机器语言并无本质区别，即具有机器语言“与机器的密切相关性”等特点，只是在《

直观和记忆方面有了改进。

### 1.1.2 高级语言

虽然汇编语言较机器语言在记忆和直观等方面有了很大的改进，但并无本质上的飞跃。人们迫切希望有一种接近自然语言或数学表达形式的程序设计语言，使程序设计工作能避开与机器硬件相关的细节，而着重于解决问题的算法本身，因此便产生了高级语言。例如，可以在程序中直接使用表达式 $36 \times 68 + 10$。目前，常用的高级语言有数十种，如 Pascal、C、C++、Basic、Java 等。

高级语言在程序设计的简易性与代码的可移植性等方面有了质的提高。当然，用高级语言编写的源程序必须经过编译和连接，将其转换为可执行程序或借助于解释程序方可运行。

### 1.1.3 汇编语言的优势

高级语言简单、易学且开发效率高，而汇编语言复杂、难懂、开发效率低。那么，是否就意味着没有必要学习和使用汇编语言了呢？对于这一问题，也存在着不同看法。支持使用汇编语言的观点认为，汇编语言具有如下优势：

● 用汇编语言容易得到高时空效率的程序。由于汇编语言本质上就是机器语言，可直接、有效地控制计算机硬件，因而与高级语言相比，容易得到运行速度快，执行代码短、占用内存空间少的高时空效率的目标程序。

● 用汇编语言能设计出高级语言无法实现的程序。正是由于与机器的密切相关性，使得汇编语言能充分利用计算机的硬件特性，编写出与硬件紧密相关而高级语言又无法实现的程序来。

尽管如此，汇编语言还是在某些方面具有高级语言无法比拟的独特优势。因此，汇编语言的意义可归纳为如下四个方面：

1. 速度：对于同一个问题，用汇编语言设计出的程序能达到“运行速度最快”。
2. 空间：对于同一个问题，用汇编语言设计出的程序能达到“占用空间最少”。
3. 功能：汇编语言可以实现高级语言难以胜任甚至不能完成的任务。
4. 知识：掌握汇编语言知识，有助于加深对计算机系统特别是程序执行逻辑的理解，有助于写出更好的高级语言程序来。很难想像，一个没有汇编语言知识的程序员能写出高质量的程序来。至少对于计算机专业人员来说，汇编语言是非常重要的。

当然，我们不能期望用汇编语言开发大型的应用软件系统，而应尽可能扬长避短。汇编语言正是由于其“面向机器”的特点，广泛用于高性能软件的开发中，例如，操作系统的核心代码要求有快速响应的实时系统等。

虽然汇编语言不具有通用性，不同类型 CPU 的指令系统可能有较大差异，但其原理和方法是具有普遍性的。只要熟练掌握一种汇编语言，再学习其他汇编语言是相当容易的。

## 1.2 数据的表示

### 1.2.1 数与数制

数制是指用一组固定的数字符号和统一的规则表示数的方法。基数和权是数制要涉及到

的两个基本概念。数制中，每个数位（数字位置）所用到的不同数字的个数叫做基数。常用的十进制，是采用0～9这10个数字表示的，它的基数是10。在一个数中，数字在不同的数位所代表的数值是不同的，每个数字所表示的数值等于它本身乘以与所在数位有关的常数，通常称这个常数为位权，简称权。例如，十进制数个位的位权是1，十位的位权是10，百位的位权是100，千位的位权是1000等。相邻两位权的比值就等于基数。一个数的数值大小就等于它的各位数码乘以相应位权的总和。

1. 数制

计算机常用的数制有二进制、八进制、十进制、十六进制。

1）十进制

由十个数字0～9组成，基数为10，小数点左边从右至左其各位的位权依次是：$10^0$，$10^1$，$10^2$，$10^3$，…；小数点右边从左至右其各位的位权依次是：$10^{-1}$，$10^{-2}$，$10^{-3}$，…。

例如，十进制数678.5可以表示为

$$678.5 = 6 \times 10^2 + 7 \times 10^1 + 8 \times 10^0 + 5 \times 10^{-1}$$

十进制数值的表示方法是在0～9数字序列后可跟一个字母D或不跟D，例如，128D或128。十进制实数的表示分为底数与指数两部分，正号“+”可有可无，指数部分写在E之后，若省略E，则底数部分应有小数点。例如，1E－32，3.1415。

2）二进制

由数字0和1组成，基数为2。小数点左边从右至左其各位的位权依次是：$2^0$，$2^1$，$2^2$，…；小数点右边从左至右其各位的位权依次是：$2^{-1}$，$2^{-2}$，$2^{-3}$，…。表示方法是在0、1序列后跟一个字母B，例如，110B、1100011B。

3）八进制

计算机中经常使用的八进制和十六进制可用于解决二进制数书写长且不易阅读的问题。八进制数由0～7共8个数字组成，基数为8，小数点左边从右至左其各位的位权依次是：$8^0$，$8^1$，$8^2$，…；小数点右边从左至右其各位的位权依次是：$8^{-1}$，$8^{-2}$，$8^{-3}$，…。表示方法是在0～7组成的数后跟一个字母O或Q。例如，1477O或1477Q。

4）十六进制

由16个数字符号（数字0～9、符号A、B、C、D、E、F）组成，16为基数。小数点左边从右至左其各位的位权依次是：$16^0$，$16^1$，$16^2$，…；小数点右边从左至右其各位的位权依次是：$16^{-1}$，$16^{-2}$，$16^{-3}$，…。表示方法是在0～9的数字或A～E的字母序列之后跟一个字母H，但必须以数字打头，例如，40490EH、0C0000H。

2. 数制的转换

1）十六进制数转换为二进制数

不论是十六进制的整数还是小数，只要把每一位十六进制的数用相应的4位二进制数代替，就可以转换为二进制数。

**【例1－1】**将$(3AB)_{16}$转换为二进制数。

$$\begin{array}{ccc} 3 & A & B \\ | & | & | \\ 0011 & 1010 & 1011 \end{array}$$

$$(3AB)_{16} = (0011\ \ 1010\ \ 1011)_2 = (11\ \ 1010\ \ 1011)_2$$

**【例 1-2】** 将 $(0.7A53)_{16}$ 转换为二进制数

$$\begin{array}{cccc} 7 & A & 5 & 3 \\ | & | & | & | \\ 0111 & 1010 & 0101 & 0011 \end{array}$$

$$(0.7A53)_{16} = (0.0111\ \ 1010\ \ 0101\ \ 0011)_2$$

2）二进制数转换为十六进制数

二进制的整数部分由小数点向左，每 4 位一分，最前面不足 4 位的在前面补 0；小数部分由小数点向右，每 4 位一分，最后不足 4 位的后面补 0。然后把每 4 位二进制数用相应的十六进制数代替，即可转换为十六进制数。

**【例 1-3】** 将 $(1101111100011.100101111)_2$ 转换为十六进制数。

$$\begin{array}{ccccccc} 0001 & 1011 & 1110 & 0011 & 1001 & 0111 & 1000 \\ | & | & | & | & | & | & | \\ 1 & B & E & 3 & 9 & 7 & 8 \end{array}$$

$$(1101111100011.100101111)_2 = (1BE3.978)_{16}$$

总之，数在机器中是用二进制表示的，但是，一个二进制数书写起来太长，且容易出错。大部分微型机的字长是 4 位、8 位、16 位或 32 位的，都是 4 的整数倍，在书写时用十六进制来表示。一个字节就可以用两位十六进制数表示，两个字节可以用 4 位十六进制表示，书写、阅读方便且不容易出错。

3）十进制整数转换成二进制整数

一般采用的方法是除 2 取余法。

具体做法为：将十进制数除以 2，得到一个商和一个余数；再将商除以 2，又得到一个商和一个余数；继续这一过程，直到商等于 0 为止。每次得到的余数（必定是 0 或 1）就是对应二进制数的各位数字。

**【例 1-4】** 将十进制数 97 转换成二进制数。其过程如下：

| | | | |
|---|---|---|---|
| 2 | 97 | | 即 $A_0=1$ |
| 2 | 48 | | 即 $A_1=0$ |
| 2 | 24 | | 即 $A_2=0$ |
| 2 | 12 | | 即 $A_3=0$ |
| 2 | 6 | | 即 $A_4=0$ |
| 2 | 3 | | 即 $A_5=1$ |
| 2 | 1 | 余数为 1 | 即 $A_6=1$ |
| 商为 | 0 | 余数为 0 | 结束 |

最后结果为

$$(97)_{10} = (A_6A_5A_4A_3A_2A_1A_0)_2 = (1100001)_2$$

4）十进制小数转换成二进制小数

一般的方法为乘 2 取整方法。

具体做法为：用 2 乘以十进制小数，得到整数和小数部分；再用 2 乘以小数部分，又得到一个整数和一个小数部分；继续这一过程，直到余下的小数部分为 0 或满足精度要求为

止。最后将每次得到的整数部分（必定是0或1）按先后顺序从左到右排列即得到所对应的二进制小数。

【例1-5】将十进制小数0.6875转换成二进制小数。其过程如下：

0.6875

× 2

1.3750　整数部分为1，即 $A_{?1}=1$

0.3750　余下的小数部分

× 2

0.7500　整数部分为0，即 $A_{?2}=0$

0.7500　余下的小数部分

× 2

1.5000　整数部分为1，即 $A_{?3}=1$

0.5000　余下的小数部分

× 2

1.0000　整数部分为1，即 $A_{?4}=1$

0.0000　余下的小数部分为0，结束

最后结果为

$$(0.6875)_{10}=(0.A_{?1}A_{?2}A_{?3}A_{?4})_2=(0.1011)_2$$

为了将一个既有整数部分又有小数部分的十进制数转换成二进制数，可以将其整数部分和小数部分分别进行转换，然后再组合起来。

### 1.2.2　计算机中的数据表示

计算机只能识别二进制数，因此数、字符等在计算机中都是使用二进制形式表示的。数可以分为带符号数和无符号数。

1. 带符号数的表示法

1）机器数与真值

上面提到的二进制数，没有提到符号问题，故是一种无符号数的表示。数总有正负之分，那么符号在计算机中又是如何表示的呢？机器中通常用一个数的最高位表示符号位。字长为8位的数，用最高位D7表示符号位，余下的7位D6～D0为数字位，符号位为“0”表示正，为“1”表示负。如：

$$X=(0101\quad 1011)_2=+91$$

而

$$X=(1101\quad 1011)_2=-91$$

这样连同一个符号位在一起的一个数，称为机器数，它的数值称为机器数的真值。

为了运算方便，即统一加减运算，机器中的负数有3种表示法——原码、反码和补码。

2）原码

若正数的符号位用“0”表示，负数的符号位用“1”表示，这种表示法就称为原码。

$X=+105,[X]=0\quad 1101001$

$X=-105,[X]=1\quad 1101001$

符号位　数值

其中，字长为8位的数，其最高位为符号位，后面7位是数值；若字长为16位，则后面的15位是数值。用原码表示时，+105和-105它们的数值位相同，而符号位不同。

原码表示简单易懂，与真值的转换方便。但是，两个异号数相加或两个同号数相减，就要做减法，难以实现加减法的统一。反码和补码的引进，就是为了把减法运算转换为加法运算。8位二进制的原码如表1-1所示。

3）反码

当一个带符号数由反码表示时，最高位为符号位。当符号位为0（即正数）时，后面的七位为数值部分；当符号位为1（即负数）时，后面几位表示的不是此负数的数值，一定要把它们按位取反，才能得到它的数值。例如，一个反码表示的数为

$$10010100$$

这是一个负数，它不表示 $(-20)_{10}$，而表示的是

$$\begin{aligned}-1101011 &= -(1\times2^6+1\times2^5+1\times2^3+1\times2^1+1)\\ &= -(64+32+8+3)=(-107)_{10}\end{aligned}$$

$$[(+4)]_{反}=0\quad\underline{0000100}$$

符号位　二进制数值

$$[(+31)]_{反}=0\quad\underline{0011111}$$

符号位　二进制数值

$$[(+127)]_{反}=0\quad\underline{1111111}$$

符号位　二进制数值

而负数的反码是它的正数连同符号位按位取反而形成的。如：

$$[(+4)]_{反}=0\ 0000100,[(-4)]_{反}=1\ 1111011$$

$$[(+31)]_{反}=0\ 0011111,[(-31)]_{反}=1\ 1100000$$

$$[(+127)]_{反}=0\ 1111111,[(-127)]_{反}=1\ 0000000$$

$$[(+0)]_{反}=0\ 0000000,[(-0)]_{反}=1\ 1111111$$

负数反码的表示与原码表示有很大的区别：最高位仍为符号位，“0”表示正数，“1”表示负数，但数值位就不同了。如：

$$[(-127)]_{反}=1\qquad 0000000$$

符号位　取反=数值

8位二进制数的反码表示如表1-2所示。

4）补码

正数的补码表示与原码相同，即最高位为符号位，“0”表示正数，其余位为数值。如：

$[(+4)]_{补}=0$ 0000100

符号位 二进制数值

$[(+31)]_{补}=0$ 0011111

符号位 二进制数值

$[(+127)]_{补}=0$ 1111111

符号位 二进制数值

反码的特点有以下3点：

（1）“0”有两种表示法。

（2）8位二进制反码所能表示的数值范围为 $-127 \sim +127$。

（3）负数的补码是先求其反码，再在最后位（即最低位）加1。如：

$[(+4)]_{原}=00000100,[(-4)]_{反}=11111011,[(-4)]_{补}=11111100,$

$[(+31)]_{原}=00011111,[(-31)]_{反}=11100000,[(-31)]_{补}=11100001,$

$[(+127)]_{原}=01111111,[(-127)]_{反}=10000000,[(-127)]_{补}=10000001,$

$[(+0)]_{原}=00000000,[(-0)]_{反}=11111111,[(-0)]_{补}=00000000$

8位带符号位的补码表示如表1-1所示，有以下特点：

（1）$[+0]_{补}=[-0]_{补}=00000000$。

（2）8位二进制补码所能表示的数值为 $-128 \sim +127$。

（3）一个用补码表示的二进制数，最高位为符号位，当符号位为“0”（即正数）时，其余七位即为此数的二进制值；当符号位为“1”（即负数）时，其余几位不是此数的数值，其数值获取的方法是将其按位取反，并在最低位加1。

**【例1-6】** $[X]_{补}=10010100$，它不等于 $(-20)_{10}$，它的数值为由0010100按位取反得1101011，然后再加1为1101100。即

$$X=-1101100=-(1\times2^6+1\times2^5+1\times2^3+1\times2^2)$$
$$=-(64+32+8+4)=(-108)_{10}$$

当负数采用补码表示时，就可以把减法转换为加法。

**【例1-7】** $X=64-10=64+(-10)$

$$[X]_{补}=[64]_{补}+[-10]_{补}$$
$$+64=[64]_{补}=01000000$$
$$+10=00001010$$
$$[-10]_{补}=11110110$$

**【例1-8】** $34-68=34+(-68)$

$$[+34]_{补}=00100010$$
$$[+68]_{补}=01000100$$
$$[-68]_{补}=10111100$$

于是