21世纪

高等院校计算机科学与技术规划教材

汇编语言程序设计

葛建梅　孙 海　邵珠富　等编著

中国水利水电出版社
www.waterpub.com.cn

21世纪高等院校计算机科学与技术规划教材

汇编语言程序设计

葛建梅　孙　海　邵珠富　等编著

中国水利水电出版社

内 容 提 要

本书主要以 Intel 8086 微处理器为背景，并兼顾 Intel 80x86 及 Pentium 微处理器，介绍了汇编语言程序设计的概念、原理、方法和技术。全书共分 10 章，主要内容包括：微机基础知识、寻址方式、8086 指令系统、80x86、Pentium 增强和扩展指令、程序设计方法、高级汇编技术、系统功能调用、输入输出程序设计和模块化程序设计及上机操作方法。每章配有适量习题，书后附有上机实验指导和习题答案。

本书结构清晰、内容丰富、实例恰当，突出了汇编语言程序设计的一般方法和技巧，方便教师教学和读者学习。可作为计算机专业及计算机相关专业本、专科“汇编语言程序设计”课程的教材，也可作为从事相关技术工作人员的参考书。

本书配有免费电子教案，读者可以从中国水利水电出版社网站上下载，网址为：http://www.waterpub.com.cn/softdown/。

图书在版编目（CIP）数据

汇编语言程序设计 / 葛建梅等编著. —北京：中国水利水电出版社，2005
（21 世纪高等院校计算机科学与技术规划教材）
ISBN 7-5084-2938-9

Ⅰ. 汇… Ⅱ. 葛… Ⅲ. 汇编语言－程序设计－高等学校－教材
Ⅳ. TP313

中国版本图书馆 CIP 数据核字（2005）第 081312 号

书　　名	汇编语言程序设计
作　　者	葛建梅　孙　海　邵珠富　等编著
出版　发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787mm×1092mm　16 开本　18.25 印张　448 千字
版　　次	2005 年 8 月第 1 版　2005 年 8 月第 1 次印刷
印　　数	0001—5000 册
定　　价	26.00 元

前　言

“汇编语言程序设计”是计算机专业的重要基础课，它不仅是微型计算机原理、操作系统、计算机接口技术等其他核心课程的先行课，而且对于训练学生掌握程序设计技术和程序调试技术均能起到重要作用。

汇编语言是用户能够利用计算机硬件特性，直接控制硬件的程序设计语言。利用汇编语言可以编写时间和空间效率较高的程序，计算机的一些系统程序就是使用汇编语言编写的。在某些领域，汇编语言仍然是必不可少的编程语言之一。由此决定了汇编语言程序设计是计算机专业及相关专业人员必须接受的专业基础训练之一，因此，作为计算机专业的学生，学习和掌握汇编语言程序设计方法是非常必要的。我们在总结多年教学实践经验的基础上，编写了这本《汇编语言程序设计》教程。

采用 Intel 80x86/Pentium 系列微处理器的微型计算机在国内得到了广泛使用，为了适应学生的认知规律，由浅入深、循序渐进地掌握汇编语言程序设计方法与技巧，本书主要以 Intel 8086 微处理器为基础和线索，系统地介绍了 Intel 8086 微处理器的特点、汇编语言程序结构、数据组织、简单的汇编语言程序设计、复杂的汇编程序设计和高级汇编技术。在详细介绍以上知识的基础上，在相应章节还扩充了适合 Intel 80x86/Pentium 系列微处理器的相关知识，如 Intel 80486 及 Pentium 微处理器的结构、存储管理、适用于 Intel 80x86/Pentium 系列微处理器的寻址方式、增强和扩展的指令等，为日后采用 Intel 80x86/Pentium 系列微机进行汇编语言程序设计奠定了坚实的基础。

本书共分 10 章。第 1 章是基础知识，概括地介绍了汇编语言和计算机中数的表示；第 2 章介绍了 Intel 8086、80486 及 Pentium 系列微处理器的结构及存储器的组成与原理；第 3 章详细介绍了寻址方式和指令系统中常用指令的格式、功能及使用方法，这些内容是利用汇编语言进行程序设计的基础；第 4 章主要讨论了汇编语句、伪指令和汇编语言的程序结构；第 5～8 章系统地介绍了顺序、分支、循环、子程序、串操作和高级汇编语言程序设计的方法，这是汇编语言程序设计的核心内容；第 9 章主要介绍了中断和输入/输出程序设计方法；第 10 章介绍了模块化程序设计方法。每章配有适量的习题，例题和习题均已调试通过。另外，为方便教学，本书还配有实验指导，共设有七个实验，给出了实验目的、要求和内容，并设置了设计和验证两种类型的实验题，以利读者在巩固书本知识的基础上，培养创新素质。

本书由葛建梅、孙海、邵珠富等编著。葛建梅编写了第 3 章、第 4 章，并负责全书的组织策划、修改补充和统稿定稿工作；孙海编写了第 1 章、第 8 章、第 9 章、第 10 章和 DEBUG 综合实验；邵珠富编写了第 5 章、第 7 章；刘艳编写了第 2 章和第 6 章；鲁静轩辅助编写了 3.7 节、4.6 节中的内容，并完成了本书全部程序的调试与验证；任冬梅编写了上机实验指导中的实验一到实验六和附录。苗巍、张淑英、尹健慧、张玲玲、王成喜、黄耀霖、薛京丽等同志参与了资料整理、讨论编写大纲工作。

由于编者水平有限，书中难免存在疏漏，敬请同行专家和广大读者指正。

编　者

2005 年 5 月

目　　录

第 1 章　基础知识

汇编语言是惟一能充分利用计算机硬件特征并能直接控制硬件的一种语言。本章将介绍利用汇编语言进行程序设计所需掌握的基本知识，包括：汇编语言及特点、数据表示等。

1.1　汇编语言程序设计概述

计算机已成为人们工作和学习的最主要工具之一，而计算机使用的程序语言的发展从面向过程到面向对象，现在又进一步发展成为面向组件，经历了非常曲折的发展过程。总的来说可以分成机器语言、汇编语言、高级语言、面向对象语言等等。许多人都学会了某种程序设计的高级语言（如 BASIC、FORTRAN、PASCAL、COBOL、C 等），使用它们编写程序，但是他们对计算机最基本的机器语言、汇编语言却了解不深。本书将详细介绍汇编语言。

1.1.1　机器语言

机器语言是第一代计算机语言，它全部由 0、1 代码组成，是能够直接被机器所接受的语言，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，因此，它的执行速度比较快，基本上充分发挥了计算机的速度性能。对于不同的计算机硬件（主要是 CPU），其机器语言一般是不相同的。每个计算机都有自己的指令集，所谓指令是指一种规定 CPU 执行某种特定操作的命令，也称为机器指令。通常一条指令对应一种基本操作，每台计算机的指令系统就是该机器的机器语言。用机器语言编写的程序，每条指令都是二进制形式的指令代码，由 0 和 1 组成，指令代码包括操作码和地址码两部分。

由于不同的计算机其机器语言有所不同，因此，针对一种计算机所编写的机器语言程序，一般不能在另一种计算机上运行。机器语言不容易记忆，程序编写难度大，调试修改烦琐，且不易移植，因此，现在几乎没有程序员这样编写程序了。但机器语言执行速度最快，它是一种面向机器的程序设计语言。

1.1.2　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，计算机严格按照程序员的要求去做，没有多余的操作。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，而不必关心机器指令的具体二进制编码。同时，把 CPU 内部的各种资源也符号化，使用该符号名也等于引用了对应的物理资源。这样，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，这就是汇编语言雏型。正是这种替代，使机器语言“符号化”，所以也称汇编语言是符号语言。这些具有一定含义的符号作为助记符，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。伪指令是在程序设计时所需要的一些辅助性说明指

令。汇编语言与特定类型的机器相对应，也是一种面向机器的语言。事实上，每一个计算机厂商都为自己的机器制定了一套机器码的“助记符”，即汇编语言指令系统。

由于汇编语言采用了助记符，因此，它比机器语言直观，容易记忆和理解，用汇编语言编写的程序也比机器语言程序易读、易检查、易修改。汇编语言与机器语言一般是一一对应的，因此，对于不同的计算机，针对同一问题所编写的汇编语言源程序是互不通用的。用汇编语言编写的程序执行效率比较高，但通用性与可移植性仍然比较差。

汇编语言是第二代程序设计语言，比机器语言前进了一步。但是，计算机不能直接识别用汇编语言编写的程序，必须由一种专门翻译程序将汇编语言程序翻译成机器语言程序，计算机才能执行。

1.1.3 为什么要学习和使用汇编语言

凡是学过一种程序设计高级语言的，都会有高级语言“易学好用”的感觉，这是因为这些语言的语句是面向数学语言或自然语言的，因此容易接受和掌握。相对来说，用汇编语言编写程序比用高级语言要困难些。既然如此，为什么至今还要学习和使用汇编语言呢?

1. 可以更深刻理解计算机的工作过程

学习和使用汇编语言可以从根本上认识、理解计算机的工作过程。因为计算机执行一个任务，归根到底就是执行一个计算机机器语言程序。通过用汇编语言编写程序，可以更清楚地了解计算机是怎样完成各种复杂工作的。在此基础上，程序设计人员更能充分地利用机器硬件的全部功能，发挥机器的长处。这就如同你编写的程序在计算机上运行时，它调动并控制着机器中每个部件和电路。

2. 许多领域和场合需要使用汇编语言

（1）与硬件资源密切相关的软件开发。现在的计算机系统中，某些功能仍然是靠汇编语言程序来实现的。例如机器自检、系统的初始化、实际的输入/输出设备的操作以及设备驱动程序等，仍然需要用汇编语言编写的程序来完成。

（2）要求执行效率高、反应快的领域。汇编语言程序的效率通常高于高级语言程序。这里的“效率”是指程序的目标代码的长短和程序运行的速度。所以在以节省内存空间和提高程序运行速度为重要指标的场合，如操作系统内核、实时工业控制、实时系统等，常常用汇编语言来编写程序。

（3）受存储容量限制的应用领域。工业控制、仪器仪表、家用电器的开发经常受到价格和体积的限制，因此使用的程序空间和容量受到限制，常常考虑使用汇编语言。

（4）其他场合。当系统性能出现瓶颈，或频繁使用子程序或程序段，没有适当的高级语言开发环境的场合，同样需要采用汇编语言。

1.2 进位计数制及其相互转换

在计算机内部常采用二进制形式表示数据，而日常生活人们熟悉的是十进制数，本节简单地介绍进位计数制及其之间相互转换的方法。

1.2.1 进位计数制

1. 数制的概念

按进位的原则进行计数叫进位计数制，简称数制。人们普遍习惯的进位计数制是十进制，

除十进制计数以外，还有钟表的六十进制，如 60 秒为 1 分钟，60 分钟为 1 小时，再如中国的“老秤”，一斤等于 16 两，是十六进制计数法。

每种数制都有其基数和各数位的位权。基数是指该数制中允许选用的基本数码的个数。十进制的每个数位上允许选用的数字是0、1、2、3、4、5、6、7、8、9，所以十进制的基数是10。一个数码处在数的不同位置时，它所代表的数值是不同的。如十进制数中，数字4在个位上表示4；在十位上表示40，即4×10；在百位上表示400，即4×10^2；在小数点后第一位上则表示0.4，即4×10^{-1}。可见每个数码所表示的数值等于该数码乘以一个与数码所在位置有关的常数，这个常数叫位权。位权的大小是以基数为底，数码所在位置的序号为指数的整数次幂。这样十进制数个位数位置上的位权为10^0，千位数位置上的位权为10^3，小数后第3位的位权为10^{-3}。例如，十进制数1548.3687可以表示成：

$1548.3687 = 1\times10^3+5\times10^2+4\times10^1+8\times10^0+3\times10^{-1}+6\times10^{-2}+8\times10^{-3}+7\times10^{-4}$

计算机的运算基础是二进制，在计算机中大都采用二进制计数制，而不使用人们习惯的十进制数，这是由二进制的特点决定的。二进制计数只有两个计数符号0和1，计算机中无论是数值型数据还是字符、声音、图像等非数值型数据都采用0和1形式的二进制码。

2. 常用的数制

在汇编语言中常用的数制有：十进制、二进制、八进制和十六进制。在计算机内部一切数据（包括数值、字符、指令等）的存储、处理和传送均采用二进制形式。二进制在计算机中是以器件的物理状态来表示的，这些器件具有两种不同的稳定状态（如低电平表示 0，高电平表示 1）且能相互转换，既简单又可靠。但二进制烦琐不便于书写，通常用八进制或十六进制来书写，为了适应人的习惯，数值型数据在输入输出设备上则采用人们十分熟悉的十进制。

（1）十进制（简记符为 D）。十进制数的基数为 10，十进制数中处于不同位置上的数字代表不同的值，与它对应的位权有关，十进制数的位权为 10^i，其中 i 代表数字在十进制数中的序号，例如，任一个十进制数 $a_na_{n-1}\cdots a_1\,a_0.a_{-1}\cdots a_{-m}$，则 i 的取值范围为 n,n-1, …,2,1,0,-1,-2, …-m，所以其中任一位 a_i 的值为 $a_i\times10^i$，这称为数的位权表示法。

如，十进制数 546.36 可表示成：$546.36 = 5\times10^2+4\times10^1+6\times10^0+3\times10^{-1}+6\times10^{-2}$ 。

一般地，任一个十进制数 $N = a_na_{n-1}\cdots a_1a_0.a_{-1}\cdots a_{-m}$ 可表示为：

$$\begin{aligned}N &= a_na_{n-1}\cdots a_1a_0.a_{-1}\cdots a_{-m}\\ &= a_n\times10^n + a_{n-1}\times10^{n-1}+\cdots+a_1\times10^1 + a_0\times10^0 + a_{-1}\times10^{-1}+\cdots+a_{-m}\times10^{-m}\\ &= \sum_{i=n}^{-m} a_i\times10^i\end{aligned}$$

其中 m，n 为正整数，n 为小数点左边的位数，m 为小数点右边的位数。a_i 取值为 0，1，2，3，4，5，6，7，8，9 之一。

（2）二进制（简记符为 B）。与十进制相似，二进制是用数字 0 和 1 表示数值，采用“逢二进一”计数原则的进位计数制。因此二进制数的基数为 2，二进制数中每一个数字的位权由 2 的幂次决定，即：2^i，其中 i 为数字在二进制数中的序号。

如，二进制数 101.11 可表示成：$1011.101 = 1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$ 。

一般地，任一个二进制数 $N = a_n\,a_{n-1}\cdots a_1\,a_0.a_{-1}\cdots a_{-m}$ 可表示为：

$N = a_na_{n-1}\cdots a_1a_0.a_{-1}\cdots a_{-m}$

$$= a_n\times2^n + a_{n-1}\times2^{n-1} +\cdots + a_1\times2^1 + a_0\times2^0 + a_{-1}\times2^{-1} +\cdots+a_{-m}\times2^{-m}$$

$$= \sum_{i=n}^{-m} a_i\times2^i$$

其中 m、n 为正整数，n 为小数点左边位数，m 为小数点右边位数。a_i 取值为 0 或 1。

（3）八进制（简记符为 Q）。八进制是用 0，1，2，3，4，5，6，7 八个数码表示数值，采用“逢八进一”计数原则的进位计数制。因此八进制数的基数为 8，每位数字的位权由 8 的幂次决定，即：8^i，i 为数字在八进制数中的序号。

任一个八进制数 $N = a_na_{n-1}\cdots a_1a_0.a_{-1}\cdots a_{-m}$ 可表示为：

$$N = a_n a_{n-1}\cdots a_1 a_0.a_{-1}\cdots a_{-m}$$

$$= a_n\times8^n+a_{n-1}\times8^{n-1}+\cdots+a_1\times8^1+a_0\times8^0+a_{-1}\times8^{-1}+\cdots+a_{-m}\times8^{-m}$$

$$= \sum_{i=n}^{-m} a_i\times8^i$$

其中 m，n 意义同前，a_i 取值范围为 0，1，2，3，4，5，6，7 中任一数字。

如：$(473.25)_8 = 4\times8^2+7\times8^1+3\times8^0+2\times8^{-1}+5\times8^{-2}$ 。

（4）十六进制（简记符为 H）。十六进制是用 0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F 十六个数码表示数值，采用“逢十六进一”计数原则的进位计数制。因此十六进制数的基数为 16，十六进制中每位数字的位权由 16 的幂次决定，即：16^i，i 为数字在十六进制数中的序号。

同理，任一个十六进制数 $N = a_na_{n-1}\cdots a_1a_0.a_{-1}\cdots a_{-m}$ 可表示为：

$$N = a_na_{n-1}\cdots a_1a_0.a_{-1}\cdots a_{-m}$$

$$= a_n\times16^n+a_{n-1}\times16^{n-1}+\cdots+a_1\times16^1+a_0\times16^0+a_{-1}\times16^{-1}+\cdots+a_{-m}\times16^{-m}$$

$$= \sum_{i=n}^{-m} a_i\times16^i$$

其中 m，n 意义同前，a_i 的取值范围为 0～9，A，B，C，D，E，F 中任一数字。

例如：$(4AF8.94B)_{16}=4\times16^3+A\times16^2 + F\times16^1+8\times16^0+9\times16^{-1}+4\times16^{-2}+B\times16^{-3}$ 。

1.2.2 各种数制间的相互转换

将数由一种数制转换成另一种数制称为数制间的转换。由于计算机采用二进制、八进制或十六进制，但对数值的输入输出常使用十进制。这就存在着十进制与非十进制之间相互转换的问题。

1. 非十进制数转换成十进制数

非十进制数转换成十进制数采用“位权法”，即把非十进制数写成各自的按权展开式，然后按十进制运算原则求和，其和值就是转换后对应的十进制数。

【例 1.1】将十六进制数 B2F 转换成十进制数。

$$(B2F)_{16}= B\times16^2 + 2\times16^1 + F\times16^0 = 11\times16^2 + 2\times16^1 + 15\times16^0$$

$$= 2816 + 32 + 15 = (2863)_{10}$$

2. 十进制数转换成非十进制数

将一个十进制数转换成非十进制数时，整数部分和小数部分的转换方法是不同的，需将

整数部分和小数部分分别转换，将两个转换结果结合起来就可以得到对应的非十进制数了。

（1）十进制整数转换成非十进制整数。将十进制整数转换为非十进制整数采用“除基取余法”，即：将十进制整数及此期间产生的商逐次除以需转换为数制的基数，直到商为零为止，并记下每一次相除所得到的余数，按从后往前的次序将各余数记作 $K_nK_{n-1}K_{n-2}\cdots K_0$，从而构成转换后对应的非十进制整数。

值得注意的是，第一次得到的余数为非十进制数的最低位，最后一次得到的余数为非十进制数的最高位。简言之，将十进制整数转换为非十进制整数的规则为：除基倒取余。

【例 1.2】将十进制整数 125 转换成对应的二进制整数。

		余数
2	125	
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

则得：$(125)_{10}=(1111101)_2$

【例 1.3】将十进制整数 125 转换成对应的十六进制整数。

		余数
16	125	
16	7	13（D）
	0	7

则得：$(125)_{10}=(7D)_{16}$

（2）十进制小数转换成非十进制小数。将十进制小数转换为非十进制小数采用“乘基取整法”，即：将十进制小数及此期间产生的积小数部分逐次乘以需转换为数制的基数，直到积的小数部分为零为止或达到一定精度为止，并记下每一次相乘所得到的整数部分，按照从前往后的次序，将各整数部分记作 $k_1\,k_2\cdots k_m$，从而构成转换后对应的非十进制小数。

【例 1.4】将十进制小数 0.467 转换成对应的二进制小数。

0.467	整数部分
× 2	
0.934	0
× 2	
1.868	1
0.868	
× 2	
1.736	1
0.736	
× 2	
1.472	1
……	

则得：$(0.467)_{10}=(0.0111\cdots)_2$

由上例可见，十进制小数并不是都能够用有限位的非十进制小数精确地表示。这时应根据精度要求转换到一定的位数为止，得到近似的非十进制小数。

如果一个十进制数既有整数部分，又有小数部分，则应将整数部分和小数部分分别进行转换，然后把两者相加便得到结果。

【例 1.5】将十进制数 125.625 转换成对应的二进制数。

因为$(125)_{10}=(1111101)_2$　　$(0.625)_{10}=(0.101)_2$

所以$(125.625)_{10}=(1111101.101)_2$

（3）二进制与八进制之间的转换。二进制的基数是 2，八进制的基数是 8，由于 8 是 2 的 3 次幂，即：$8=2^3$，因此，一位八进制数正好相当于三位二进制数；反之，三位二进制数可表示一位八进制数。所以若把二进制数转换为八进制数，只需以小数点为界，将整数部分从右向左每三位一组，最高一组不足三位时，在最左端添 0 补足三位，小数部分从左向右，每三位一组，最低一组不足三位时，在最右端添 0 补足三位，然后，将各组的三位二进制数转换为对应的一位八进制数，即得到转换后八进制数。反之，若将八进制数转换成二进制数，只要把每位八进制数用对应的三位二进制数表示，即可得到对应的二进制数。

【例 1.6】将二进制数 1101100111.10011 转换成对应的八进制数。

001 101 100 111 . 100 110

1　5　4　7　4　6

则得：$(1101100111.10011)_2=(1547.46)_8$

【例 1.7】将八进制数 576.32 转换成对应的二进制数。

$(576.32)_8$ = 101 111 110.011 010

则得：$(576.32)_8=(101111110.01101)_2$

（4）二进制与十六进制之间的转换。十六进制的基数是 16，由于 16 是 2 的 4 次幂，即：$16=2^4$，因此，一位十六进制数可用四位二进制数表示；若把二进制数转换为十六进制数，只需以小数点为界，将整数部分从右向左每四位一组，最高一组不足四位时，在最左端添 0 补足，小数部分从左向右按四位为一组，最低一组不足四位时，在最右端添 0 补足，然后，将各组的四位二进制数转换为对应的一位十六进制数，即得到转换后十六进制数。反之，若将十六进制数转换成二进制数，只要把每位十六进制数用对应的四位二进制数表示即可。

【例 1.8】将二进制数 1101100111.10111 转换成对应的十六进制数。

0011 0110 0111 . 1011 1000

3　6　7　B　8

则得：$(1101100111.10111)_2=(367.B8)_{16}$

【例 1.9】将十六进制数 5FD4.A3 转换成对应的二进制数。

$(5FD4.A3)_{16}$ = 0101 1111 1101 0100 . 1010 0011

则得：$(5FD4.A3)_{16}=(101111111010100.10100011)_2$

1.3　计算机中数的表示

计算机处理的数据分为数值型和非数值型两类。数值型数据具有量的含义，且有正负之

分、整数和小数之分；而非数值型数据是指输入到计算机中的所有信息，没有量的含义，如英文字母、数字符号 0～9、汉字、声音、图形、图像等，在计算机中这些数据是如何表示的呢？由于计算机采用二进制，也就是说计算机只识别 0 和 1 形式的代码，所以输入到计算机中任何数值型和非数值型数据都必须转换为二进制代码。这一节我们先讨论数值型数据的编码方法，在后面一节中我们再讨论非数值型数据的编码方法。

数值型数据是用二进制数来表示的，数值数据分为有符号数和无符号数，有符号数有正、负之分。通常，无符号数最高位表示数值，而有符号数最高位表示符号，规定：用“0”表示“+”号，用“1”表示“-”号。我们把这种连同数字与符号组合在一起的二进制数称为机器数，由机器数所表示的实际值称为真值。真值的符号用“+”或“-”表示。

例如：机器数为：00110010 ，其真值为：+0110010

有符号数可以用不同的码制来表示，即：原码、反码、补码、移码等，常用的是补码。

1.3.1　原码表示法

1. 原码定义

设字长为 n 位，最高位为符号位，正数的符号用“0”表示，负数的符号用“1”表示，数值部分 n-1 位用二进制真值的绝对值表示，这种表示法称为原码。

例如，设字长为 8，当 x = +1010 时，$[x]_{原}$ = 00001010，当 x = -1010 时，$[x]_{原}$ = 10001010

当 x = +0.1001 时，$[x]_{原}$ = 0.1001000，当 x = -0.0101 时，$[x]_{原}$ = 1.0101000

2. 原码的表示范围

定点整数原码的数值范围为：$-(2^{n-1}-1) \leqslant x \leqslant 2^{n-1}-1$。

定点小数原码的数值范围为：$-(1-2^{-(n-1)}) \leqslant x \leqslant 1-2^{-(n-1)}$。

原码表示简单，并易于和真值转换。但用原码进行加减运算时，却很麻烦。因此，加减运算时，机器数常常采用补码。

1.3.2　补码表示法

1. 补数概念

假设某科室共有人员 15 名，某日早上考勤时，发现有三名同志没有来，可计为-3，那么有 12 人出勤，这样 12 与-3 的绝对值之和为 15，则称-3 为 12 的补数，而 15 为模（mod15）。同样我们可以说-5 的补数是 10，-7 的补数是 8。

当模为 12 时，10-5 = 5，将-5 用其的补数 7 代替则有：

10-5 = 10+(-5)= 10+7 = 17 = 12+5 = 5（模 12）。

因为模为 12，结果 17 超 12 模 5 个，因此结果应该是 5，这样就与 10-5 的结果一样，因此就将减法变成了加法运算。将补数的概念用到计算机中，就出现了补码。

2. 补码的定义

补码表示法规定：正数的补码与原码相同，负数的补码是对该数的原码除符号位外各位取反，然后末位加 1。

例如，设字长为 8，当 x = +1000 时，$[x]_{补}$ = 00001000，当 x = -1001 时，$[x]_{补}$ = 11110111

当 x = 0.1011 时，$[x]_{补}$ = 0.1011000　当 x = -0.1110 时，$[x]_{补}$ = 1.0010000

3. 补码的表示范围

定点整数补码的数值范围为：$-2^{n-1} \leqslant x \leqslant 2^{n-1}-1$。

定点小数补码的数值范围为：$-1 \leqslant x \leqslant 1-2^{-(n-1)}$。

1.3.3 反码表示法

1. 反码的定义

反码表示法规定：正数的反码和原码相同，负数的反码是对该数的原码除符号位外各位取反，即“0”变“1”，“1”变“0”。

例如，设字长为 8，当 x = +1101 时，$[x]_{反} = 00001101$，当 x = -1101 时，$[x]_{反} = 11110010$

当 x = +0.1010 时，$[x]_{反} = 0.1010000$，当 x = -0.1001 时，$[x]_{反} = 1.0110111$

2. 反码的表示范围

定点整数反码的数值范围为：$-(2^{n-1}-1) \leqslant x \leqslant 2^{n-1}-1$。

定点小数反码的数值范围为：$-(1-2^{-(n-1)}) \leqslant x \leqslant 1-2^{-(n-1)}$。

1.3.4 移码表示法

1. 移码的定义

$[x]_{移} = 2^n + x \quad (2^n > x \geqslant -2^n)$

式中 x 为真值，n 为整数的位数。

移码是在真值上加一个常数 2^n，在数轴上移码所表示的范围恰好对应与真值在数轴上的范围向轴的正方向移动 2^n 个单元，由此得到移码名称。

例如，当 x = +1000101，$[x]_{移} = 2^7 + 1000101 = 11000101$。

x = -1000101，$[x]_{移} = 2^7 - 1000101 = 00111011$。

x = -1000101，$[x]_{补} = 2^8 - 1000101 = 10111011$。

可以看出同一真值的补码和移码只差最高位相反，如上例中移码最高位是 0，而其补码的最高位是 1。

2. 移码的表示范围

移码的表数范围是：$-2^n \leqslant x \leqslant 2^{n-1}$，与整数补码的表示范围相同。

1.3.5 补码的加法和减法运算

在计算机中，乘法运算可变成加法运算，除法运算可变为减法运算，因此加减法运算是计算机中最基本的运算，而加减运算普遍采用补码运算。

1. 补码加减运算的基本公式

补码加法的基本公式为：

$[A]_{补}+[B]_{补}=[A+B]_{补}$（整数 mod 2n+1；小数 mod 2）

即补码表示两个数在进行加法运算时，可以把符号位与数位同等处理，只要结果不超出机器能表示的数值范围，运算后的结果按 2n+1 取模（对于整数）；或按 2 取模（对于小数），就能得到本次加法的运算结果。

对于减法，因 A-B = A+(-B)，则$[A-B]_{补} = [A+(-B)]_{补}$

由补码加法基本公式可得：