高等院校计算机应用规划教材

微型计算机原理与汇编语言程序设计

WEIXING JISUANJI YUANLI YU
HUIBIAN YUYAN CHENGXU SHEJI

刘爱荣　马耀锋　主编

化学工业出版社

◆ 高等院校计算机应用规划教材 ◆

微型计算机原理与汇编语言程序设计

WEIXING JISUANJI YUANLI YU
HUIBIAN YUYAN CHENGXU SHEJI

刘爱荣　马耀锋　主编
孔国利　王振成　副主编

化学工业出版社
·北京·

全书共分8章。根据课堂教学和实践的需要，详细介绍了计算机中的信息表示及运算、80x86内部结构、微处理器的存储器组织、内存储器，介绍了汇编指令系统和汇编语言编程基础、微型计算机存储器系统、汇编语言程序设计技巧和中断程序设计。运用大量综合性实例对各种关键技术进行了深入浅出的分析。此外，每一章节配有练习题，附录配有实训内容。

本书适合作为理工类计算机、电子、通信及自动控制等相关专业本科微机原理、汇编语言以及作为它们的组合课程的教材，也可以作为理工类高职高专教材或工程技术人员参考书。

图书在版编目（CIP）数据

微型计算机原理与汇编语言程序设计/刘爱荣，马耀锋主编. —北京：化学工业出版社，2015.6
高等院校计算机应用规划教材
ISBN 978-7-122-23678-4

Ⅰ.①微… Ⅱ.①刘… ②马… Ⅲ.①微型计算机-高等学校-教材②汇编语言-程序设计-高等学校-教材
Ⅳ.①TP36②TP313

中国版本图书馆CIP数据核字（2015）第079261号

责任编辑：廉 静 李翠翠
责任校对：宋 玮 装帧设计：韩 飞

出版发行：化学工业出版社（北京市东城区青年湖南街13号 邮政编码100011）
印 装：大厂聚鑫印刷有限责任公司
787mm×1092mm 1/16 印张16½ 字数425千字 2015年7月北京第1版第1次印刷

购书咨询：010-64518888（传真：010-64519686） 售后服务：010-64518899
网 址：http://www.cip.com.cn
凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定 价：36.00元

前　言

微型计算机原理与汇编语言是计算机专业的专业基础课程，也是电子、通信及自动控制等相关专业计算机技术课程的专业基础课。本书以80X86系列微型计算机为基础，以emu8086集成开发环境为汇编上机实验环境，重点介绍Intel8086指令系统。本书的写作特点为采用实例驱动教学的方法，以丰富的示例和实例为依托展开教学和学习；在附录D中布置了实验任务模块，通过多层次的实验训练来加强各章内容的学习理解、融会贯通。

全书结构清晰，内容丰富，例题多样，练习和习题针对性强。所有程序都经过运行验证，习题和测验附有答案。与本书配套有多媒体PPT课件、书中的例题程序及习题答案等，可登录化学工业出版社教材资源网www.cipedu.com.cn下载。

本书由微型计算机基础知识、微型计算机原理和汇编语言等部分组成。主要内容有：微机硬件基础、8086/8088CPU的内部结构和指令系统、汇编语言及编程技巧，内存的存储原理及与CPU的连接，I/O读写方式及中断应用编程，实用附录等。本教材融多位老师的教学经验，重点突出，详略有序，图表丰富，实用性强。

本书在选材上注重应用、重点突出。书中给出了一定的设计实例，希望能对读者迅速掌握微型计算机原理与汇编语言程序设计有所帮助。

全书由刘爱荣、马耀锋担任主编，刘爱荣、马耀锋、孔国利、张璐璐、刘佳、王振成、王欣参与编写工作。其中第1章由王欣编写，第2章和附录B由张璐璐编写，第3章和第4章的4.1节由刘爱荣编写；第5、7章由马耀锋编写，第6章由王振成编写，第8章、第4章的4.2节和4.3节由孔国利编写，附录A、D和附录C由刘佳编写，全书由中州大学刘爱荣统稿、定稿。另外，解放军信息工程大学王志新教授对此书的编写提出了宝贵意见，在此深表感谢。

在编写本书的过程中参考了相关文献，在此向这些文献的作者深表感谢！由于微型计算机是一门发展迅速的新技术，加上作者水平有限，书中难免疏漏、不妥之处，恳请专家和广大读者批评指正。联系信箱：zgdxlar@sina.com。

编者

2015年2月

目 录

第1章 计算机中的信息表示及运算

本章导读

- 掌握数据的表示及其转换
- 重点掌握数值型数据的表示方法
- 重点掌握二进制数的逻辑运算和算术运算
- 了解文字信息的编码及表示

什么是汇编语言？为什么要学习汇编语言？为什么要用十六进制数？怎样区别计算机中数的含义？

学习汇编语言，重要的是掌握如何通过汇编指令和程序来控制计算机各个组成部件工作完成一系列任务。因此学习汇编语言与学习高级语言的不同之处是要学习如何深入到计算机的内部进行控制，包括控制 CPU 的数据寄存器、地址寄存器、标志寄存器；对存储器的读写操作，对输入设备和输出设备的访问等具体操作。

学会了汇编语言，就能够在 CPU 的寄存器级上进行控制和操作，掌握直接对计算机硬件编程的方法，从而对计算机系统有更深刻的认识，分析问题的角度就会处在更深的层面。而高级语言采用英语单词表示程序语句，使用通常的数学表达式以及专门的语法规则编写程序。通过高级语言，不懂计算机原理和结构的人也可编写出程序来，但是他们有可能只知其一，不知其二。用高级语言编写的程序对计算机的控制不如用汇编语言编写的程序对机器的控制来得更直接、更有效和迅速。

本章介绍学习汇编语言所需的基本知识，并通过具体的例子为读者建立起汇编语言的初步概念。

1.1 计算机基本概念

众所周知，计算机以二进制数为基础。那么控制计算机工作的机器指令就由二进制数构成，而机器指令的集合称之为机器语言。如果想让计算机工作，就要写出一系列二进制的机器码。计算机获得这些机器指令后，立即而迅速地完成相应的任务。例如计算 Z＝35＋27，写成机器指令为：

二进制表示	十六进制表示
101110000010001100000000	B82300
000001010001101100000000	051B00
101000110000010000000000	A30400

可以看出，机器指令非常难记。在上例中二进制数用了 24 位，写成十六进制数表示也要用 6 位。将这种用二进制表示的机器指令改为便于记忆和理解的助记符表示形式。助记符采用英文单词的缩写表示，容易理解记忆。上例计算用指令助记符表示如下：

```
MOV  AX, 35
ADD  AX, 27
MOV  Z, AX
```

其中，MOV 代表传送，第 1 条指令表示把 35 传送给 AX 寄存器；ADD 代表加法，该指令表示将 AX 中的值与 27 相加，结果再放到 AX 中；最后一条指令表示将 AX 中的计算结果放入 Z 存储单元中。这样的指令就好理解了，可以容易地将计算程序编写出来。

这些助记符就是汇编指令，汇编指令的集合构成了汇编语言。汇编语言是一种符号化的机器语言。汇编语言既便于程序员编写程序，又保留了机器语言可直接而迅速地控制机器的长处。可以说，汇编语言是直接控制计算机工作的最简便的语言。

但是用汇编语言编写程序时要遵循一定的语法规则，这些规则与高级语言相似，程序员要按照规则编写程序以便于翻译程序进行翻译。对于汇编语言而言，程序员编写的程序称为源程序，翻译程序是一种称为汇编程序的系统软件，“翻译”的过程称为汇编。这是规范的叫法。

汇编语言有三种指令形式：汇编指令、伪指令和宏指令。其中，汇编指令可以翻译成二进制的机器指令代码，而伪指令和宏指令不能翻译成机器指令，它们是在汇编期间为汇编程序提供相关信息使用的。总而言之，用户编写的汇编源程序中必须要有汇编指令和伪指令，宏指令可根据需要设定。

在汇编语言中，涉及的基本概念有：数的表示、寄存器、存储单元、指令格式、语法规则等。要想掌握汇编语言的概念和汇编语言程序设计方法，就要先学习和掌握这些基础知识。

1.2 计算机中数据的表示及其转换

计算机中数据的概念是广义的，是指计算机所能处理的信息的总和。信息有多种形式，如一个人的自然信息，可以用姓名、性别、身高、出生日期等来表示。还可以用图形同时显示其相貌。计算机要处理这些信息，首先要将这些信息转换成适合计算机处理的数据形式。

计算机中有两类数据信息：数值数据和非数值数据。数值数据即平常所说的有大小、正负含义的量，非数值数据包括字母、数字、通用符号、控制符号、汉字、图形、图像和声音等各种信息，但在计算机内部所有信息都要转换成 0 和 1 表示的二进制编码。在计算机中，数可以用二进制、十六进制、十进制、八进制等表示。日常生活中习惯使用十进制，而编写汇编语言程序时经常要用到的是十六进制、二进制数据。因此，三种进制之间的相互转换应该熟练掌握；同时还要使自己尽快习惯用十六进制来思考。由于计算机中常用的进制是二进制、十六进制、十进制，本书主要介绍这三种进制数。

1.2.1　计算机中信息的表示

目前的电子计算机是一种电子设备，只认识电信号，即：电平高低、电路通断、晶体管的导通与截止、电子开关的开与关。用 0 和 1 表示一个二进制位（电信号的两个状态），计算机中的任何信息都是 0 与 1 的特定组合。

信息表示的单位：

- bit（比特）：一个二进制位。
- Byte（字节）：8 个一个二进制位。
- Word（字）：16 位机中是 2 个字节。

1.2.2　数制表示

凡是按进位的方式计数的数制叫做进位计数制，简称进位制。数据无论使用哪种进位制都涉及到基数（Radix）与各数位的“权”（weight），所谓某进位制的基数是指该进位制中允许选用的基本数码的个数。

计算机汇编语言中常用的计数制有十进制数（Decimal）、二进制数（Binary）、十六进制数（Hexadecimal）和八进制数（Octal）。

1. 十进制数

最常用的十进制数，每个数位上允许选用 0，1，2，……，9 共 10 个不同数码中的 1 个，因而十进制数的基数为 10，每位计满 10 时向高位进 1。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有关的常数，这个常数就叫“位权”，简称权。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。它的运算规则是“逢十进一，借一为十”。在书写二进制时，为了区别，在数据后面紧跟一个字母 B。例如：

$236.75D=2*10^2+3*10^1+6*10^0+7*10^{-1}+5*10^{-2}$

2. 二进制数

二进制只有两个记数符号 0 和 1，它逢二进一，二进制数由排列起来的 0 和 1 组成。各位的权值，从位序号 0 向左数，依次为 1，2，4，8 等等，从位序号为 −1 向右数，依次为 1/2，1/4，1/8 等等，一个二进制所表示的实际值按如下公式计算：

例如：

$1011.11B=1*2^3+0*2^2+1*2^1+1*2^0+1*2^{-1}+1*2^{-2}$

3. 八进制数

八进制数是 0 到 7 这八个数码组成，它的基数为 8，各位的权值为 8^i。八二进制数的运算规则是“逢八进一，借一为八”。在书写时，为了区别，在数据后面紧跟一个字母 Q。例如，1234Q、7654Q、54Q 等都是八进制。

4. 十六进制数

十六进制数是由 0 到 9 和 A，B，C，D，E 和 F（小写也可以，分别代表 10 到 15）这十六个数码组成的，它的基数为 16，各位的权值为 16^i。它的运算规则是“逢十六进一，借一为十六”。

在书写时，为了区别，在数据后面紧跟一个字母 H。当十六进制数的第一个字符是字母时，在第一个字符之前必须添加一个 0。例如，100H、56EFH、0FFH、0ABCDH 等都是十六进制数。加 0 是为了和寄存器相区别，例如，AH 可以理解为寄存器，只有前面加 0

为0AH才理解为16进制数。

1.2.3 数制的转换

1. 将十进制数转换成二进制数

整数的转换采用“除2取余”法。把要转换的十进制数的整数部分不断除以2并在右边记下余数，然后以此类推，直到最后的商为0，最后将余数倒着排列即得结果。

小数的转换采用“乘2取整”法。小数的转换正好与整数的转换相反，将小数部分不断地乘以2在右边记下成绩整数部分，依此类推，直到最后小数部分为0或结果已达到精度要求为止。先取出来的整数作高位，后取出的整数作低位，顺序排列即可。

例如，将205转换成二进制整数，如图1-1所示。

```
2 | 205
2 | 102   ……1(最低有效位)
2 |  51   ……0
2 |  25   ……1
2 |  12   ……1
2 |   6   ……0
2 |   3   ……0
2 |   1   ……1
      0   ……1(最高有效位)
故：205=11001101B
```

图1-1 205转换成二进制整数

```
将347转换成十六进制数
16 | 347
16 |  20   ……7
16 |   1   ……4
       0   ……1
故：347=147H
```

图1-2 347转换成十六进制数

例如，将十进制数 125.6875D转换成二进制数。

125.6875D=1111101.1011B

2. 将十进制数转换成八/十六进制数

方法如上，只不过乘/除数为8或16而已。转换过程如图1-2所示。

例如：

100D=64H

65535=0FFFFH

255=0FFH

3. 二/八/十六进制数转换成十进制数

转换方法是采用位置记数法把它们写成相应的幂运算形式，然后求和即可得到结果。例如：

$572.34Q=5*8^2+7*8+2+3*8^{-1}+4*8^{-2}=378.4375D$

4. 二进制数与八进制数的转换

因为$2^3=8$，所以二进制数转换成八进制数只需将二进制数从小数点开始每3位转成一位八进制数，整数部分由最低位开始划分，小数部分从最高位开始划分。例如：

101111010.011100B=572.34Q

八进制数转换成二进制数只需将每一位八进制数用三位二进制数表示，小数点位置不变。例如：

175.54Q=001 111 101.101 100B=1111101.1011B

5. 二进制数与十六进制数的转换

至于十六进制数与二进制数的转换和八进制类似只需将每3位一组改为每4位一组即

可。例如：

101111010.011111B＝17A.7CH

1.3　数值型数据的表示方法

一个数值型的数据既有大小，又可能有正负；即可能是整数，又可能是小数。而在计算机中，数的所有信息都要通过二进制编码的形式表示出来。本小节介绍数值型数据是如何在计算中表示出来的。

数据是计算机处理的对象，数据表示是计算机运算的基础。在选择数据的表示方式时，需要考虑以下几个因素：①要表示的数的类型（小数、整数、实数和复数）；②可能遇到的数值范围；③数值精确度；④数据存储和处理所需要的硬件代价。

1.3.1　机器数的概念及其特点

机器数是指一个数值数据在计算机中的二进制表示形式，即一个数值数据的机内编码。如前所述，一个数值型数据的所有信息都要以二进制编码的形式表示出来，因此，机器数有以下的特点。

1. 符号位数码化

数的符号位在计算机中也要以二进制编码的形式表示出来。通常用的方法是用一位二进制的“0”来代表“＋”，用“1”来代表“－”。通常这个符号放在二进制数最高位，称为符号位。这样机器中数的符号就被数值化了，符号数值化是机器数的一个特点。

2. 小数点位置确定

小数点在一个机器中的位置通常是事先约定隐含在一个固定位置上，而不再占据机器数中的数位。约定的具体方法在后面会有介绍。小数点位置用一定方式约定，这是机器数的又一个特点。

3. 每个机器数对应唯一的真值

机器数所代表的实际值叫真值。由于机器数的编码中包含有除数值大小之外的信息，因此，一个机器数形式上的值就不等于其实际的数值大小。但某种编码规则下一个机器数应该代表一个唯一的实际值。

4. 机器数表示数的范围和精度受机器字长的限制

在计算机中作为整体进行传输和参与运算的一个二进制串称为字。每个字中包含的二进制位数称为字长，即字长是 CPU 一次能够并行处理二进制数据位数。一般来说，字长越长，机器数所能表示的二进制数的范围就越大，表示数的精度就越高。

1.3.2　数值型数据的表示形式

数值型数据多带有小数，小数点在计算机中通常有两种形式：一种是约定所有数值数据的小数点隐含在某个固定位置上，称为定点表示法；另一种小数点的位置可以浮动，称为浮点表示法。

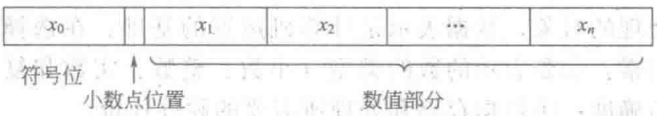
计算机常用的数据表示格式有两种：定点格式和浮点格式。一般来说，定点格式容许的数值范围有限，但要求的处理硬件比较简单。而浮点格式容许的数值范围很大，但所要求的

处理硬件比较复杂。

1. 定点数表示法（fixed-point）

所谓定点格式，即约定机器中所有数据的小数点位置是固定不变的。在计算机中通常采用两种简单的约定：将小数点的位置固定在数据的最高位之前，或者是固定在最低位之后。一般常称前者为定点小数，后者为定点整数。

• 定点小数是纯小数，约定的小数点位置在符号位之后、有效数值部分最高位之前。若数据 x 的形式为 $x=x_0.x_1x_2\cdots x_n$（其中 x_0 为符号位，$x_1\sim x_n$ 是数值的有效部分，也称为尾数，x_1 为最高有效位），则在计算机中的表示形式为：



一般说来，如果最末位 $x_n=1$，前面各位都为 0，则数的绝对值最小，即 $|x|_{min}=2^{-n}$。如果各位均为 1，则数的绝对值最大，即 $|x|_{max}=1-2^{-n}$。所以定点小数的表示范围是：

$$2^{-n}\leqslant|x|\leqslant1-2^{-n}$$

• 定点整数是纯整数，约定的小数点位置在有效数值部分最低位之后。若数据 x 的形式为 $x=x_0x_1x_2\cdots x_n$（其中 x_0 为符号位，$x_1\sim x_n$ 是尾数，x_n 为最低有效位），则在计算机中的表示形式为：

定点整数的表示范围是：

$$1\leqslant|x|\leqslant2^n-1$$

当数据小于定点数能表示的最小值时，计算机将它们作 0 处理，称为下溢；大于定点数能表示的最大值时，计算机将无法表示，称为上溢，上溢和下溢统称为溢出。

计算机采用定点数表示时，对于既有整数又有小数的原始数据，需要设定一个比例因子，数据按其缩小成定点小数或扩大成定点整数再参加运算，运算结果，根据比例因子，还原成实际数值。若比例因子选择不当，往往会使运算结果产生溢出或降低数据的有效精度。

用定点数进行运算处理的计算机被称为定点机。

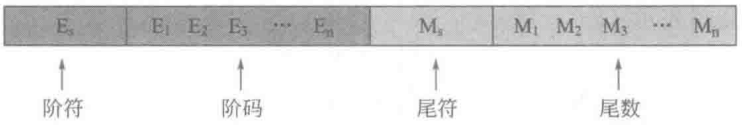
2. 浮点数表示法（floating-point number）

与科学计数法相似，任意一个 J 进制数 N，总可以写成

$$N=J^E\times M$$

式中 M 称为数 N 的尾数（mantissa），是一个纯小数；E 为数 N 的阶码（exponent），是一个整数，J 称为比例因子 J^E 的底数。这种表示方法相当于数的小数点位置随比例因子的不同而在一定范围内可以自由浮动，所以称为浮点表示法。

底数是事先约定好的（常取 2），在计算机中不出现。在机器中表示一个浮点数时，一是要给出尾数，用定点小数形式表示。尾数部分给出有效数字的位数，因而决定了浮点数的表示精度。二是要给出阶码，用整数形式表示，阶码指明小数点在数据中的位置，因而决定了浮点数的表示范围。浮点数也要有符号位。因此一个机器浮点数应当由阶码和尾数及其符号位组成：



其中 E_S 表示阶码的符号，占一位，$E_1 \sim E_n$ 为阶码值，占 n 位，尾符是数 N 的符号，也要占一位。当底数取 2 时，二进制数 N 的小数点每右移一位，阶码减小 1，相应尾数右移一位；反之，小数点每左移一位，阶码加 1，相应尾数左移一位。

若不对浮点数的表示作出明确规定，同一个浮点数的表示就不是唯一的。例如 11.01 也可以表示成 0.011012^{-3}，0.1101×2^{-2} 等等。为了提高数据的表示精度，当尾数的值不为 0 时，其绝对值应大于等于 0.5，即尾数域的最高有效位应为 1，否则要以修改阶码同时左右移小数点的方法，使其变成这一要求的表示形式，这称为浮点数的规格化表示。

当一个浮点数的尾数为 0 时，不论其阶码为何值，或者当阶码的值遇到比它能表示的最小值还小时，不管其尾数为何值，计算机都把该浮点数看成 0 值，称为机器零。

浮点数所表示的范围比定点数大。假设机器中的数由 8 位二进制数表示（包括符号位）：在定点机中这 8 位全部用来表示有效数字（包括符号）；在浮点机中若阶符、阶码占 3 位，尾符、尾数占 5 位，在此情况下，若只考虑正数值，定点机小数表示的数的范围是 0.0000000 到 0.1111111，相当于十进制数的 0 到 127/128，而浮点机所能表示的数的范围则是 $2^{-11}\times0.0001$ 到 $2^{11}\times0.1111$，相当于十进制数的 1/128 到 7.5。显然，都用 8 位，浮点机能表示的数的范围比定点机大得多。

尽管浮点表示能扩大数据的表示范围，但浮点机在运算过程中，仍会出现溢出现象。下面以阶码占 3 位，尾数占 5 位（各包括 1 位符号位）为例，来讨论这个问题。图 1-3 给出了相应的规格化浮点数的数值表示范围。

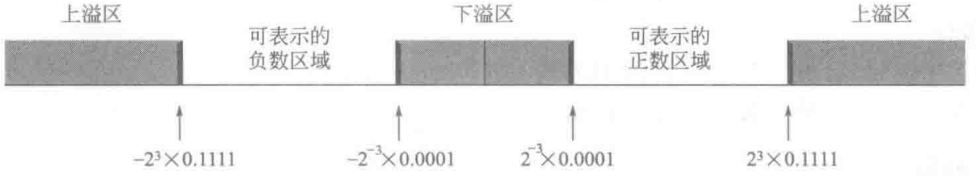


图 1-3　规格化浮点数分布示意图

图 1-3 中，“可表示的负数区域”和“可表示的正数区域”及“0”，是机器可表示的数据区域；上溢区是数据绝对值太大，机器无法表示的区域；下溢区是数据绝对值太小，机器无法表示的区域。若运算结果落在上溢区，就产生了溢出错误，使得结果不能被正确表示，要停止机器运行，进行溢出处理。若运算结果落在下溢区，也不能正确表示之，机器当 0 处理，称为机器零。

一般来说，增加尾数的位数，将增加可表示区域数据点的密度，从而提高了数据的精度；增加阶码的位数，能增大可表示的数据区域。

1.3.3　有符号数的表示法

有符号数在计算机中的表示有原码表示法、反码表示法、补码表示法，它们表示的范围和计算机的字长有关。

1. 原码

原码表示最高位为 0 表示正数，为 1 表示负数，其余位表示数的绝对值。对于整数，原

码可以用下面的公式表示：

$$[X]_{原}=\begin{cases}X & 当\ 0\leqslant X<2^n\\ 2^n-X=2^n+|X| & 当\ 0\geqslant X>-2^n\end{cases}$$

对于小数，原码用下面的公式表示：

$$[X]_{原}=\begin{cases}X & 当\ 1>X\geqslant 0\\ 1-X=1+|X| & 当\ 0\geqslant X>-1\end{cases}$$

例如，X=0.1001B，则 $[X]_{原}=0.1001B$

X=−0.1001B，则 $[X]_{原}=1.1001B$

X=−107D，则 $[X]_{原}=1001011B$

X=45，则 $[X]_{原}=00101101B$

X=0.625D，则 $[X]_{原}=0.10100000B$

原码的表示法简单易懂，但它的加法运算复杂。例如，当两个数相加时，如果同号则相加，异号则两数相减。

2. 反码

正数的反码表示和原码相同，最高位为 0 表示正数，其余位表示数的绝对值。负数的反码表示可以在其原码表示的基础上保持符号位不变，其余位求反。

对于整数，反码用下面的公式表示：

$$[X]_{反}=\begin{cases}X & 当\ 0\leqslant X<2^n\\ (2^{n+1}-1)+X & 当\ 0\geqslant X>-2^n\end{cases}$$

对于小数，反码用下面的公式表示：

$$[X]_{反}=\begin{cases}X & 当\ 0\leqslant X<2^n\\ (2-2^{-n})+X & 当\ 0\geqslant X>-1\end{cases}$$

例如：

若 X=−107，则 $[X]_{反}=10010100B$

若 X=−0.5，则 $[X]_{反}=1.0111111B$

3. 补码

在汇编语言中，数值都看成是补码。补码作为机器数之一，有着重要的作用。有了补码，减法就可变为加法，除法就可用乘法实现，而这些都离不开我们对补码的认识。在用补码做运算时，你首先要知道这个数是正数还是负数。对于二进制数和十六进制数要一眼就看出该数的符号；而对于运算的结果，要能判断出结果是否溢出，结果是否有进位；表示成十进制是多少……，对于这些问题的判断，首先要改变自己的思维方式，换十进制思维为十六进制思维，锻炼自己观察二进制数和十六进制数的能力。当然，在实际操作中，并不需要人来作出判断，计算机自己就完成了。

正数的补码表示和原码相同，最高位为 0 表示正数，其余位表示数的绝对值。负数的补码表示可以在其原码表示的基础上保持符号位不变，其余位求反末位加 1。

对于整数，补码用下面的公式表示：

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

对于小数，反码用下面的公式表示：

$$[X]_{补}=\begin{cases}X & 当\ 0\leqslant X<2^n\\ 2+X=2-|X| & 当\ 0\geqslant X>-1\end{cases}$$