21世纪高等院校电气信息类系列教材

微型计算机原理与接口技术

张荣标　等编著

机械工业出版社
CHINA MACHINE PRESS

21 世纪高等院校电气信息类系列教材

微型计算机原理与接口技术

张荣标　等编著

机 械 工 业 出 版 社

本书以 Intel 系列微处理器为背景，介绍了微型计算机原理与接口技术。全书以弄懂原理、掌握应用为编写宗旨，在内容安排上注重系统性、逻辑性、先进性与实用性。本书分三个部分：微型计算机原理部分（1、2、6、7、8 章），汇编语言程序设计部分（3、4、5 章），接口与应用部分（9、10、11、12 章）。根据 Intel 系列微处理器的向下兼容性，着重讲解了 16 位微型计算机的工作原理、指令系统、8086 汇编语言程序设计以及接口技术。考虑到目前 32 位 CPU 的广泛应用，又重点介绍了其代表芯片 80386 的工作原理，特别是 80386 的存储器管理技术。

为了便于读者自学，在内容安排方面除附有一定量的习题外，还增设了详细的习题例解。

本书可以作为高等院校电气信息类专业教材，也可作为计算机等级考试的参考教材，还可供从事微型机系统设计和应用的技术人员自学和参考。

图书在版编目（CIP）数据

微型计算机原理与接口技术/张荣标等编著. —北京：机械工业出版社，2005.1
（21 世纪高等院校电气信息类系列教材）
ISBN 7-111-16025-8

Ⅰ. 微 … Ⅱ. 张 … Ⅲ. ① 微型计算机 - 理论 - 高等学校 - 教材
② 微型计算机 - 接口 - 高等学校 - 教材 Ⅳ. TP36

中国版本图书馆 CIP 数据核字（2005）第 004096 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策　　划：胡毓坚
责任编辑：时　静
责任印制：石　冉

保定市印刷厂印刷·新华书店北京发行所发行

2005 年 1 月第 1 版·第 1 次印刷
787mm × 1092mm 1/16·27.5 印张·679 千字
0 001 —5 000 册
定价：36.00元

出 版 说 明

随着科学技术的不断进步，整个国家自动化水平和信息化水平的长足发展，社会对电气信息类人才的需求日益迫切、要求也更加严格。在教育部颁布的“普通高等学校本科专业目录”中，电气信息类（Electrical and Information Science and Technology）包括电气工程及其自动化、自动化、电子信息工程、通信工程、计算机科学与技术、电子科学与技术、生物医学工程等子专业。这些子专业的人才培养对社会需求、经济发展都有着非常重要的意义。

在电气信息类专业及学科迅速发展的同时，也给高等教育工作带来了许多新课题和新任务。在此情况下，只有将新知识、新技术、新领域逐渐融合到教学、实践环节中去，才能培养出优秀的科技人才。为了配合高等院校教学的需要，机械工业出版社组织了这套“21 世纪高等院校电气信息类系列教材”。

本套教材是在对电气信息类专业教育情况和教材情况调研与分析的基础上组织编写的，期间，与高等院校相关课程的主讲教师进行了广泛的交流和探讨，旨在构建体系完善、内容全面新颖、适合教学的专业教材。

本套教材涵盖多层面专业课程，定位准确，注重理论与实践、教学与教辅的结合，在语言描述上力求准确、清晰，适合各高等院校电气信息类专业学生使用。

机械工业出版社

前　言

微型计算机原理与接口技术是自动化、电气、电子信息以及其他电气信息类专业的一门重要专业基础课。随着微处理器技术的不断发展和用人单位对人才培养的更高要求，迫切需要一批适合新形势需要的相关教材。为此，本书作者参考现有教材，扬长避短，结合多年来一线教学的经验，并征求同行教师以及学生对微型计算机原理教材的要求，从教和学的角度出发，着手编写了本教材。与现有教材相比，本教材有如下特点：

1. 增设题解，便于自学

微型计算机原理与接口技术这门课内容多，课时少，除教师课堂上讲解外，学生必须花一定的时间复习和巩固已学过的知识。本书除编入一定量的习题外，还编入了习题例解，这在很大程度上减轻了该专业基础课的教学压力。

2. 面向实用，夯实基础

本教材侧重基础知识，用模型机讲解 CPU 的工作原理，以 8086 CPU 为背景，系统地讲解了 16 位微型计算机的工作原理。考虑到目前 32 位 CPU 的广泛应用，又重点介绍了其代表芯片 80386 的工作原理，特别是 80386 的存储器管理技术。这样，可以使学生从基本原理出发，把握先进技术。

3. 力求图示，方便理解

本教材尽可能采用图示的方法，让学生有一种感性认识。如介绍微型计算机系统时，采用实物图片，使学生对微机有一种实实在在的感觉，激发出对微型计算机原理学习的兴趣；在讲解指令的寻址过程中，采用示意图的方式，使学生一目了然。

4. 条理清晰，便于领会

本教材中通篇都贯穿了“条理清晰”这一特点，学生比较容易掌握要点。

5. 突出重点，详解难点

从学生实际应用出发，在掌握了必要的基础知识情况下，将重点放在汇编语言编程和接口技术的学习上，这些内容也是学生学习的难点。

全书由张荣标教授统稿，其中第 6 章由陆文昌副教授编写，其余各章由张荣标教授编写。书中的汇编语言程序已由作者的研究生冯友兵、李华、章云峰、陈相朝等同学在计算机上验证通过。本书还得到了赵德安教授、李岚博士的大力支持。同时对参与书稿录入和整理工作的硕士研究生们表示感谢。

为便于教学，系列教材中配备了白殿生编著的《微型计算机原理与接口技术学习指导》。有兴趣的读者可以参考。

由于作者水平有限，存在一些不足之处，恳请读者批评指正。

目　录

第 1 章　微型计算机基础

随着微处理器制造技术的不断发展，计算机的结构越来越复杂，功能越来越强大，性能越来越优越，计算机原理所涉及的内容也就越来越多。但是计算机基本原理没有改变，只要对计算机的基础知识有充分的了解，就能从容地面对计算机日新月异的变化。

本章着重学习计算机中的数制和码制，最简单计算机的结构及其工作原理。为了使读者对计算机有一个感性认识，本章从计算机的结构出发，介绍了目前流行的台式个人计算机结构以及模型机的结构和工作原理。

1.1　计算机中的数制与码制

数是客观事物的量在人们头脑中的反映，一个“量”相同的数可以用不同的计数制度来表示，这就形成了不同的数制。表达一个数的大小和正负的不同方法称作码制。

1.1.1　计算机中的数制

1. 数的位置表示法

数制是人们按某种进位规则进行计数的科学方法。位置表示法是表示数的常用方法。在数的位置表示法中，基数取值不同便可得到不同进位制的表达式。设待表示的数为 N，则

$$N = \sum_{i=-m}^{n-1} a_i X^i \tag{1-1}$$

式中，X 为基数；a_i 为系数（$0 < a_i < X-1$）；m 为小数位数；n 为整数位数。

在计算机中常用的数制有二进制、八进制、十六进制和十进制，相应的 X 可取值为 2，8，16，10。在计数或加法运算过程中，它们分别是逢二进一、逢八进一、逢十六进一和逢十进一。在减法运算过程中，它们分别是借一当二、借一当八、借一当十六和借一当十。在数的位置表示法中，它们的后缀分别是 B、Q、H 和 D（或省略不写）。

下面用例题来说明用二进制、八进制、十六进制表示数的结果（十进制数）。

【例 1-1】　（1）二进制数

$$10011.11B = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} = 19.75$$

（2）八进制数

$$7345.6Q = 7\times8^3 + 3\times8^2 + 4\times8^1 + 5\times8^0 + 6\times8^{-1} = 3813.75$$

（3）十六进制数

$$4AC6H = 4\times16^3 + 10\times16^2 + 12\times16^1 + 6\times16^0 = 19142$$

人们习惯使用的是十进制，而计算机中采用的基本数制是二进制（八进制和十六进制作为二进制的一种编写形式便于表示）。这是为什么呢？采用二进制数，不仅因为它只有 0 和 1 两个系数，还因为 0 和 1 用电路实现起来很方便。即数字电路中通常的两种稳态：逻辑器件的饱和与截止状态。相应地形成低电位和高电位，以此代表两个数码：0 和 1。计算机通

常用高电位代表 1，低电位代表 0。采用这样的电路精心设计的电路系统，既简单又快捷。

2. 数制之间的转换

(1) 任意进制数转换为十进制数

对二进制、八进制和十六进制以及任意进制数转换为十进制数可采用表达式（1-1）展开求和实现，详见例 1-1。

(2) 二进制、八进制和十六进制数之间转换

一位八进制数相当于三位二进制数；一位十六进制数相当于四位二进制数。它们之间的转换十分方便。

【例 1-2】 二进制数转换成八进制和十六进制数。

$$1101100101100011B = 154543Q$$
$$= D963H$$

(3) 十进制数转换为二进制数

当十进制数转换为二进制数时，需将整数部分和小数部分分开。整数常采用“除 2 取余法”，而小数则采用“乘 2 取整法”。这里要提及的是十进制小数并不是都能用有限的二进制小数精确地表示，此时要根据精度的要求来确定被转换的二进制位数。

1) 十进制整数转换为二进制整数。转换方法是除 2 取余，直到商等于零为止，逆序排列余数即可。对数值比较大的十进制数进行转换时，可采用先将十进制整数转换为十六进制整数，然后再将十六进制整数转换为二进制整数。十进制整数转换为十六进制整数的方法是除 16 取余，直到商等于零为止，逆序排列余数。

【例 1-3】 将十进制数 19，3910 分别转换为相对应的二进制数。

解：

```
2 | 19
2 |  9      商为9，余数为 1      低位
2 |  4      商为4，余数为 1
2 |  2      商为2，余数为 0
2 |  1      商为1，余数为 0
2 |  0      商为0，余数为 1      高位
```

```
16 | 3910
16 |  244    商为244，余数为 6      低位
16 |   15    商为 15，余数为 4
16 |    0    商为 0，余数为 15(F)   高位
```

转换结果分别为 19D = 10 011B；3910 D = F46H = 111101000110B。

2) 十进制小数转换为二进制小数。转换方法是将小数部分乘 2 取整，直到乘积的小数部分等于零为止（若永不为零则根据精度要求截取一定的位数），顺序排列每次乘积的整数部分即可。

【例 1-4】 将十进制数 19.8125 转换为相对应的二进制数。

解： 整数部分可由例 1-3 的结果得：

$$19D = 10011B$$

小数部分 0.8125D 的转换过程：

0.8125D × 2=1.625	得小数部分为	0.625	整数部分为	1	高位
0.625D × 2=1.25	得小数部分为	0.25	整数部分为	1	
0.25D × 2=0.5	得小数部分为	0.5	整数部分为	0	
0.5D × 2=1.0	得小数部分为	0	整数部分为	1	低位

转换结果为　　19.8125D = 10011.1101B

1.1.2　计算机中的码制及补码运算

一个数除了有量大小之分外还有正负的区别。为了处理数的符号问题，在计算机中引进了码制的概念。通常用二进制数的最高位来表示数的符号位。常用的码制有原码、补码、反码及偏移码。

1. 原码

用二进制数的最高位表示数的符号，通常规定以 0 表示正数，1 表示负数，其余各位表示数值本身，则称该二进制数为原码表示法。

设机器字长为 n，数 X 的原码为 $[X]_{原}$，则原码的定义如下：

$$[X]_{原}=\begin{cases}X & 0\leqslant X\leqslant 2^{n-1}-1\\ 2^{n-1}+|X| & -2^{n-1}-1\leqslant X\leqslant 0\end{cases} \tag{1-2}$$

【例 1-5】　设机器字长为 n = 8 时，试求 + 0、+ 6、+ 127、- 0、- 6、- 127 的原码。

解：　$[+0]_{原}=00000000$　　$[-0]_{原}=10000000$

$[+6]_{原}=00000110$　　$[-6]_{原}=10000110$

$[+127]_{原}=01111111$　　$[-127]_{原}=11111111$

由此可见，原码是把符号数值化了的数，在计算机中称为机器数。对正数来说，原码与相应的二进制数完全相同；对负数来说，二进制数的最高位一定是“1”，其余各位是该数的绝对值。零的原码表示有正零和负零之分。原码表示法最大优点是简单直观，但不便于加减运算。原码数的运算完全类同于正负数的笔算。比如两个正数相减，先比较两个数绝对值的大小，然后把绝对值大的减去绝对值小的，最后在结果前面加上原来绝对值较大的符号。因而处理过程非常繁琐，要求计算机的结构也极为复杂。

2. 反码

设机器字长为 n，数 X 的反码为 $[X]_{反}$，则反码的定义如下：

$$[X]_{反}=\begin{cases}X & 0\leqslant X\leqslant 2^{n-1}-1\\ 2^{n}-1+X & -2^{n-1}-1\leqslant X\leqslant 0\end{cases} \tag{1-3}$$

【例 1-6】　设机器字长为 n = 8 时，试求 + 0、+ 6、+ 127、- 0、- 6、- 127 的反码。

解：　$[+0]_{反}=00000000$　　$[-0]_{反}=11111111$

$[+6]_{反}=00000110$　　$[-6]_{反}=11111001$

$[+127]_{反}=01111111$　　$[-127]_{反}=10000000$

可以看出，正数的反码与相应的原码完全相同，负数的反码只需把相应的原码除符号位外其余各位按位求反即可。用反码来表示负数现已较少采用。

3. 补码

原码和反码都不便于运算，是否存在另一种数的表示法能便于运算呢？回答是肯定的，那就是补码表示法。在数的原码和反码表示法中，参加运算的数的符号是不能参加运算的，而在数的补码表示法中，参加运算的数的符号与数一样，也可以参加运算，并且使减法运算变成加法运算，省去一套减法电路。因而采用补码运算使计算机的结构大为简化。

补码为什么具有这种功能，为了说明补码的概念，可从时钟校准谈起。若现在是北京时间 1 点整，而时钟快了两小时，时针指在 3 点上，要将时钟校准有两种方法：一种是把时钟倒拨两小时，相当于作减法运算；另一种是顺时针拨 10 小时，相当于作加法运算，钟面上两种方法得到的结果是一样的。即：

$$3+10=1 \text{（时针经过 12 点时自动丢失一个数 12）}$$

相当于

$$3-2=3+(-2)=1$$

这里把减法运算变成了加法运算，其中 10 与 -2 到底有什么关系，自动丢失的一个数 12 又是什么，这是学习补码概念的关键。数学上把 12 这个数叫作“模”，10 是（-2）对模 12 的补码。这样，在模 12 的条件下，负数就可以转化为正数，而正负数相加也就可以转化为正数间的相加。

再考虑计算机运算的特点，计算机中的部件都有固定的位数，假定位数为 n，则计算机中最大的计数值（包括符号位）为 2^n-1，当大于等于 2^n 时同样会自动丢失一个数 2^n。因此，计算机中的负数可以表示成以 2^n 为模的补码。这样可以得到计算机中二进制补码的定义，即：

设机器字长为 n，数 X 的补码为 $[X]_{补}$，则补码的定义如下：

$$[X]_{补}=\begin{cases} X & 0\leqslant X\leqslant 2^{n-1}-1 \\ 2^n-|X| & -2^{n-1}\leqslant X<0 \end{cases} \tag{1-4}$$

【例 1-7】 设机器字长为 n=8 时，试求 +0、+6、+127、-0、-6、-127 的补码。

解： $[+0]_{补}=00000000$　　$[-0]_{补}=00000000$

$[+6]_{补}=00000110$　　$[-6]_{补}=11111010$

$[+127]_{补}=01111111$　　$[-127]_{补}=10000001$

可以看出，正数的补码与相应的原码完全相同，负数的补码只需把相应的原码除符号位外其余各位按位求反并在末位加 1 即可。

4. 偏移码

偏移码主要用于模/数转换过程中，若被转换数需参加运算，则仍要转换为补码。

设机器字长为 n，数 X 的移码为 $[X]_{移}$，则移码的定义如下：

$$[X]_{移}=2^{n-1}+X \tag{1-5}$$

【例 1-8】 设机器字长为 n=8 时，试求 -128、0、+127 的移码。

解： $[-128]_{移}=00000000$

$[0]_{移}=10000000$

$[+127]_{移}=11111111$

可以看出，移码与相应的补码在数轴上向右平移了 2^{n-1}，从而弥补了补码不直观的缺点，用移码表示数的大小可以说是一目了然。注意它仅是在数轴上平移了 2^{n-1} 个单位，使最

小的负数变为0，最大的正数变为最大数111....11。但是，这里的数2^{n-1}与补码中模的概念是不一样的。

5. 补码运算

在计算机中带符号二进制数通常采用补码形式表示，当两个二进制数进行补码加减运算时，有两个主要特点：一是可以使符号位与数一起参加运算；二是将两数相减变为减数变补后再与被减数相加来实现。让我们来看补码加、减法运算规则。

加法规则：$[X+Y]_{补}=[X]_{补}+[Y]_{补}$

减法规则：$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$

其中，$[-Y]_{补}$ 称作变补运算，可以用 $[Y]_{补}$ 再作一次求补运算即可得到。具体运算过程可以通过下面的两个例子来说明。

【例 1-9】 $X=64-12=52$ （字长为8位）

$[X]_{补}=[64]_{补}+[-12]_{补}$

$[64]_{补}=01000000B$ $[-12]_{补}=11110100B$

```
  01000000
+ 11110100
----------
1 00110100
```

自然丢失

由于字长为8位，最高有效位的进位自然丢失。其结果为$(52)_{10}$的补码。

【例 1-10】 $X=34-98=-64$ （字长为8位）

$[X]_{补}=[34]_{补}+[-98]_{补}$

$[34]_{补}=00100010B$ $[-98]_{补}=10011110B$

```
  00100010
+ 10011110
----------
  11000000
```

和的最高位是1，表示负数，数值部分由后七位按位取反再加1，即为11000000B，其结果为$(-64)_{10}$的补码。

可以看出，上述两例的计算结果是正确的。即使最高有效位的进位因字长的限制而被自动丢失（自动丢失一个称作模的数2^8），也并不影响结果的正确性。计算机中对有符号数的运算采用的就是补码运算。

可以看到，计算机中加、减运算采用补码，不仅十分简便，而且不要判断正负号，符号位一齐参加运算，自动得到正确的补码结果。

6. 溢出判别

当两个带符号位的二进制数进行补码运算时，若运算结果的绝对值超过运算装置的容量，数值部分便会发生溢出，占据符号位的位置，从而引起计算出错。这和补码运算过程中的正常溢出（符号位的进位）性质上是不同的。正常溢出是以2^n（n为二进制数的位数）为模的溢出，它被自然丢失，不影响结果的正确性（如例1-9）。但是，如果数值部分发生了溢出，就会影响结果的正确性。现举例说明如下：

$X=-34-98=-132$ （字长为8位）

$$[X]_{补} = [-34]_{补} + [-98]_{补}$$

$[-34]_{补} = 11011110B$　　$[-98]_{补} = 10011110B$

$$\begin{array}{rr} & 11011110 \\ + & 10011110 \\ \hline 1 & 01111100 \end{array}$$

丢失

显然，X = 124 这样的结果是错误的。

任何一种运算都不允许发生溢出，除非是仅利用溢出作为判断依据而不使用计算结果的情况。所以当溢出产生时，应使计算机停机或进入检查程序找出溢出原因，然后作相应处理。因此溢出判别是确保二进制补码运算结果正确的关键。

微型机中常用的溢出判别法是双高位判别法。在双高位判别法中有两个附加的符号，即 C_S 和 C_P，其具体定义如下：

C_S：如最高位（符号位）有进位，$C_S = 1$，否则，$C_S = 0$，它表征符号位的进位情况。

C_P：如次高位有进位，$C_P = 1$，否则，$C_P = 0$，它表征数值部分最高位的进位情况。

根据两个附加的符号，可得补码运算溢出的双高位判别法则：

当两数进行二进制补码加减过程中，若最高位进位 C_S 和次高位进位 C_P 相同（同为 0 或同为 1），则无溢出发生，若 C_S 和 C_P 相异，则有溢出发生，且 $C_S = 0$，$C_P = 1$ 为正溢出，$C_S = 1$，$C_P = 0$ 为负溢出。

在微型机中，常用“异或”线路来判别有无溢出产生，即：当 $C_S \oplus C_P = 1$ 时，表示有溢出产生，否则无溢出产生。

下面通过具体的例题来说明双高位判别法的应用过程。

【例 1-11】 试判别下列二进制补码运算溢出的情况（字长为 8 位）

(1) 92 + 105　　(2) (− 115) + (− 87)

(3) 35 + 55　　(4) (− 15) + (− 67)

(1) 解：

$$\begin{array}{rrrr} & 0101 & 1100 & 92 \\ + & 0110 & 1001 & 105 \\ \hline 0 & 1100 & 0101 \rightarrow & -59\ (\text{结果求补}) \end{array}$$

$C_S=0$　$C_P=1$　　正溢出，结果出错

可见两个正数相加，设微型计算机的字长为 n，若数值部分之和大于 2^{n-1}，则数值部分必有进位 $C_P = 1$，而符号位却无进位 $C_S = 0$。这种 C_SC_P 的状态为“01”时的溢出称为“正溢出”。

(2) 解：

$$\begin{array}{rrrr} & 1000 & 1101 & [-115]_{补} \\ + & 1010 & 1001 & [-87]_{补} \\ \hline 1 & 0011 & 0110 \rightarrow & +54 \end{array}$$

$C_S=1$　$C_P=0$　　负溢出，结果出错

可见两个负数相加，设微型计算机的字长为 n，若数值部分绝对值之和大于 2^{n-1}，则数值部分之和必小于 2^{n-1}，$C_P=0$，而符号位肯定有进位 $C_S=1$。这种 C_SC_P 的状态为“10”时的溢出称为“负溢出”。

(3) 解：

$$\begin{array}{rcll} & 0010 & 0011 & 35 \\ + & 0011 & 0111 & 55 \\ \hline & 0101 & 1010 \rightarrow & 90 \end{array}$$

$C_S=0$　$C_P=0$　　无溢出

可见两个正数相加，若和小于 2^{n-1}时，必有 $C_S=0$，$C_P=0$，则无溢出发生。

(4) 解：

$$\begin{array}{rcll} & 1111 & 0001 & [-15]_{补} \\ + & 1011 & 1101 & [-67]_{补} \\ \hline 1 & 1010 & 1110 \rightarrow & -82 \quad \text{(结果求补)} \end{array}$$

$C_S=1$　$C_P=1$

可见两个负数相加，若和的绝对值小于 2^{n-1}时，必有 $C_S=1$，$C_P=1$，则无溢出发生。

一个正数和一个负数相加，和肯定不溢出。此时，若和为正数，则 $C_S=1$，$C_P=1$；若和为负数，则 $C_S=0$，$C_P=0$。请读者自己验证。

1.1.3 计算机中的小数点问题

计算机中小数点的表示法有两种：定点表示法和浮点表示法。无论采用哪一种方法，数字本身是看不出小数点位置的，而是通过人-机“约定”来解决小数点的表示问题。

1. 定点表示法

小数点在数中的位置是固定不变的，通常有两种，即定点整数和定点小数。前者是将小数点固定在最低数位之后，后者是将小数点固定在最高数位之前。在对小数点位置做出选择之后，运算中的所有数均应统一为定点整数或定点小数，在运算中不再考虑小数点问题。

由于定点表示的数值范围有局限性，运算精度又很低，故实用意义不大，但对一些运算量不大或运算精度满足的场合这种方法比较直观简单，一般在智能仪器仪表中使用得比较多。

2. 浮点表示法

为了扩大数值范围，提高运算精度，计算机中大多采用浮点表示法。

将二进制数 N 表示成如下形式：

$$N=\pm S\times 2^{\pm J} \tag{1-6}$$

该表达式在计算机中表示为：

J_f	J	S_f	S

其中：S 称作尾数，表示全部的有效数字，一般以纯小数表示；

S_f 为尾符，即浮点数的符号；

J 为阶数，它与阶符一起来决定小数点的实际位置，用整数表示；

J_f 为阶符，即阶数符号。

下面通过一个实例来说明计算机中浮点数的具体表示方法。

【例 1-12】 若用一个 16 位二进制表示浮点数，其中阶符尾符各占一位，阶数占 5 位，尾数占 9 位，试写出 10110.101B 的具体格式。

解：设尾数以纯小数表示，则

$$10110.101B = 0.10110101 \times 2^5$$

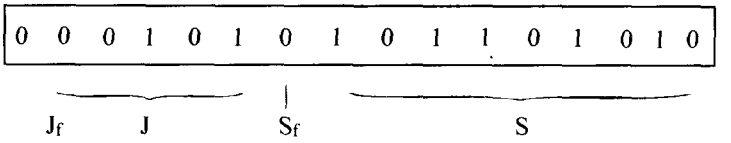
可得 $S = 101101010$

$S_f = 0$

$J = 00101$

$J_f = 0$

在计算机中的表示形式为：



浮点数应用中必须注意两个问题：

(1) 浮点数的规格化

规格化的浮点数可以保留最多的有效数字。浮点数规格表示结果如下：

对浮点二进制正数，其尾数数字部分的最高位必须是 1。

对浮点二进制负数，其尾数数字部分的最高位必须是 0。

(2) 浮点数的对阶原则

在运用浮点数进行加减时，两数的阶码必须取得一致，否则不能进行加减运算，对阶原则如下：

① 以大的阶码为准对阶。

② 对阶后数的大小不变（在精度允许范围内），对阶规则是：阶码每减少 1，尾数向左移一位，阶码每增加 1，尾数向右移一位。

浮点运算过程比较复杂。计算机上的浮点运算有两种实现方案：一种是配置浮点运算器（硬件法），另一种是编制浮点运算程序（软件法），但程序较为复杂。在高档微型机系统中多采用硬件法。如与 8086、80286、80386 CPU 相匹配的数学协处理器分别是 8087、80287 和 80387，对 80486 以上的微处理器，其数学协处理器已集成在 CPU 芯片内部。

1.1.4 计算机中信息的编码

计算机只能处理二进制数，而实际应用中送入计算机处理的原始数据大多是十进制数、字母、符号等信息，除了用二进制表示数据外，这些信息同样要用二进制编码来表示，这就是信息的编码。信息编码可分为十进制数的二进制编码、字符信息的编码和汉字编码。

1. 十进制数的二进制编码

十进制数不能直接送入计算机中参加运算，必须用二进制数为它编码，使其成为二－十

进制码，也称作 BCD 码（Binary Coded Decimal）。

用二进制数为十进制数编码，每一位十进制数需要由四位二进制数来表示。四位二进制数能编出 16 个码，其中 6 个码是多余的，放弃不用。由于这种多余性便产生了多种不同的 BCD 码。下边介绍常用的 3 种 BCD 码。

1）四位二进制数的权（对相应数位所赋的位值）分别为 8、4、2、1 的 BCD 码称为 8421 码，见表 1-1。它所表示的数值规律与二进制计数制相同，是最简单、最容易理解的一种 BCD 码。例如：

324.6 对应的 8421BCD 码是 0011 0010 0100. 0110

2）四位二进制数的权分别为 2、4、2、1 的 BCD 码，称为 2421 码，见表 1-1。该 BCD 码具有自补性质，即它的二进制编码的 1 补码是相应十进制数的 9 补码。例如：

724.6 对应的 2421BCD 码是 1101 0010 0100. 1100

3）将 8421 码加上 0011 就得到余 3 码，见表 1-1。余 3 码的特点是其十进制运算较为简单。余 3 码也是一种自补码，对各位取反就得到它的 9 补码。例如：

825.7 对应的余 3 码是 1011 0101 1000. 1010

表 1-1　3 种常用的 BCD 码

BCD 码	在不同码制中所对应的十进制值		
	8421 码制	2421 码制	余 3 码值
0000B	0	0	-
0001B	1	1	-
0010B	2	2	-
0011B	3	3	0
0100B	4	4	1
0101B	5	-	2
0110B	6	-	3
0111B	7	-	4
1000B	8	-	5
1001B	9	-	6
1010B	-	-	7
1011B	-	5	8
1100B	-	6	9
1101B	-	7	-
1110B	-	8	-
1111B	-	9	-

2. 字符信息的编码

字母、数字和符号等各种字符也必须按特定的规则用二进制编码才能在计算机中表示。在微型机中表示字符的常用码制是 ASCII 码，它是美国信息交换标准码（American Standard Code for Information Interchange），多用于输入/输出设备上。它能用 6 位、7 位或 8 位二进制数对字符编码。7 位 ASCII 码可表示 128 种字符，它包括 52 个大、小写字母、0～9 十个数字和控制符号（见表 1-2）。在计算机中用一个字节来表示一个 ASCII 码字符，最高位置 0。如字