姚燕南 原京亮 武子旸

《微型计算机原理》

学习指导书

西安电子科技大学出版社

《微型计算机原理》

学习指导书

姚燕南　原京亮　武子旸

西安电子科技大学出版社

1997

内容简介

本书作为《微型计算机原理》教材的学习辅导材料，对第1～10章作了要点、难点和重点分析。分析中适当增加了例题以加深理解，并对每章的习题和思考题作了示范性的解答。

《微型计算机原理》

学习指导书

姚燕南　原京亮　武予旸

责任编辑　叶德福

西安电子科技大学出版社出版发行

西安秦群印刷厂印刷

新华书店经销

开本 787×1092　1/16　印张 6 6/16　字数 143 千字

1997 年 10 月第 1 版　1997 年 10 月第 1 次印刷　印数 1—8 000

ISBN 7－5606－0539－7/TP・0267　　定价：6.50 元

前　言

作为电子工业部工科电子类规划教材《微型计算机原理》(姚燕南、薛钧义编，西安电子科技大学出版社 1994 年 6 月出版)一书受到广大授课教师和学生的喜爱，已陆续重印多次。为配合教学需要，应出版社要求，我们根据多年的教学实践编写了这本学习指导书。书中对各章节内容的要点、难点和重点作了较为详细的分析，并适当增加了一定份量的例题，也对每章的习题和思考题中的主要部分作了标准解题过程的示范，目的在于启迪学生的思维。希望通过对该书的阅读，能使学生牢固掌握教材内容，并学以致用。

本书第 1～4 章由姚燕南编写，第 5～8 章由原京亮编写，第 9、10 章由武子旸编写。并由姚燕南同志统编全稿。

本书在编写过程中得到薛钧义教授的热情指导和大力协助，在此表示衷心的感谢。由于编者水平有限，书中难免还存在一些缺点和错误，敬请广大读者批评指正。

编　者

于西安交通大学

1996.10.18

目　录

第 1 章　计算机中的数制与码制

1.1　学习要点

1.1.1　计算机中的数制

1. 几个重要概念

1）数制

数是客观事物的量在人们头脑中的反映，它的大小可以用不同的计数制度（简称数制）来表示。虽然形式不同，但数的量则是相等的。

2）数的位置表示法

采用数的位置表示法时，用来表示一个数的一组数字或符号中，其每个数字或符号表示的量不但决定于数字或符号本身，而且还决定于它们所在的位置（称为数位）。

3）权

采用数的位置表示法时，表示一个数的一组数字或符号中，每个数位所具有的位值称为权。

4）基数

采用数的位置表示法时，表示一个数的一组数字或符号中，相邻两位的高位权与低位权相比如果是常数，则此常数称为基数。

5）系数

采用数的位置表示法时，表示一个数的一组数字或符号中，各数位上的数字称为系数，其取值为小于基数的正整数。

2. 一个 n 位整数、m 位小数的数

对于一个 n 位整数、m 位小数的数

$$\underbrace{a_{n-1}\cdots a_i\cdots a_0}_{n\text{位整数}}\underset{\text{小数点}}{.}\underbrace{a_{-1}\cdots a_i\cdots a_{-m}}_{m\text{位小数}}$$

来说，若用 X 表示基数，则该数的值 N 为

$$N=\underbrace{a_{n-1}X^{n-1}+a_{n-2}X^{n-2}+\cdots+a_0}_{n\text{位整数的值}}+\underbrace{a_{-1}X^{-1}+\cdots+a_{-(m-1)}X^{-(m-1)}+a_{-m}X^{-m}}_{m\text{位小数的值}}$$

每个数位上的权是基数 X 的乘幂（幂指数大小从左至右依等差级数方式减少），每个数位的

值等于其权与系数之积。整个数的值等于各数位值的和。系数 $a_i((n-1)\geqslant i\geqslant -m)$ 可在 0，1，…，$X-1$，共 X 种数中任意取值。n、m 为幂指数，均为正整数，分别表示数的整数位数和小数位数。在基数为 X 的位置表示方法中，两数加减运算时采用“逢 X 进一”和“借一当 X”的规则。

3. 计算机技术中常用的进位制数

在数的位置表示法中，基数 X 的取值不同便得到不同进位制数的表达式。计算机技术中常用的进位制数有如下 4 种：

十进制数　　$(N)_{10}=\sum_{i=-m}^{n-1}a_i10^i$

二进制数　　$(N)_2=\sum_{i=-m}^{n-1}a_i2^i$

八进制数　　$(N)_8=\sum_{i=-m}^{n-1}a_i8^i$

十六进制数　　$(N)_{16}=\sum_{i=-m}^{n-1}a_i16^i$

由于二进制记数制各数位的系数只有 0 和 1 两种取值，用电路实现起来最为方便，且其运算也最为简单，因而它被作为计算机内部采用的计数制。这也就是说，当我们用计算机的机器语言编写程序时，必须采用二进制数。但是，书写机器语言时，由于二进制数书写起来显得冗长，常用八进制数和十六进制数。

众所周知，当我们采用计算机的高级语言或汇编语言编写程序时，不仅可以用二进制数，而且还可以采用十进制数、八进制数和十六进制数。这并不是说计算机硬件本身可以识别二进制以外的计数制，只是因为高级语言的编译程序和汇编语言的汇编程序均具有非二进制数与二进制数之间的转换程序而已。

采用汇编语言编写程序时，十进制数加后缀 D 或不加反缀，八进制数和十六进制数的后缀分别为 Q 和 H。这里特别要注意的是采用十六进制数时，系数 a_i 的取值范围是 0～15，其中 10～15 用 A～F 或 $\bar{0}$～$\bar{5}$来表示。

4. 各种进位制数之间的转换方法

十进制数是日常生活中最常用的计数法，计算机的输入数据和输出数据若能用十进制数表示，就显得特别直观和方便，尤其对那些不懂计算机内部结构的用户来说更为必要。然而，计算机内部是采用二进制数，人们书写程序时又多用八进制数和十六进制数。这样，作为计算机应用的高级技术人员必须学会这几种进位制数之间的转换方法。

1）八进制数、十六进制数与二进制数之间的转换方法

由于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。因而，八进制数、十六进制数与二进制数之间的相互转换非常简便、易学，具体方法见教材①

2）任意进位制数转换成十进制数

转换方法很多，教材上介绍了两种方法，其中第二种方法需将整数部分和小数部分分

① 本书所指教材均指由姚燕南、薛钧义编写的《微型计算机原理》(第三版)(1994 年西安电子科技大学出版社出版)。

开进行转换。这两种方法都是最常用的转换方法，也很简便、直观，读者不难掌握。

3）十进制数转换成任意进位制数

通常也需将整数部分和小数部分分开进行转换。教材上介绍的转换方法较为常用，且直观、方便，易于掌握。值得提醒读者的是，一个二进制小数能完全准确地转换成十进制小数，但一个十制数小数不一定能完全准确地转换成二进制小数。这是二进制计数制的一个缺点，今后编程应用时务必注意。

5. 二进制数的运算方法

由于本教材的重点在于介绍微型计算机的内部结构、它的指令系统、汇编语言编程方法及中断处理和输入输出方法，因而，熟悉和掌握二进制数的运算方法十分重要，这里重点是掌握好一位二进制数的四则运算规则和机理，然后便可方便地扩展到多位二进制数的运算。

1.1.2　计算机中的码制

在计算机中，用来表示一个数的一串数字或符号称为这个数的码或编码。表达一个数的大小和正负的不同编码方法称为码制。

1. 计算机中常用码制的定义

计算机中常用的码制有 4 种：原码、补码、反码及移码，定义如下：

（1）设 X 为 $n-1$ 位二进制正数，即

$$X = +X_{n-2}X_{n-3}\cdots X_1X_0$$

则

$$[X]_{原} = [X]_{补} = [X]_{反} = 0X_{n-2}X_{n-3}\cdots X_1X_0$$

其中最高位为符号位。

这就是说，正数的原码、补码及反码均为 n 位二进制数，且相同。其中最高位为符号位“0”，其余 $n-1$ 位即为 $n-1$ 位二进制正数的本身。

（2）设 X 为 $n-1$ 位二进制负数，即

$$X = -X_{n-2}X_{n-3}\cdots X_1X_0$$

则

$$[X]_{原} = 1X_{n-2}X_{n-3}\cdots X_1X_0 = 2^{n-1} + X_{n-2}X_{n-3}\cdots X_1X_0$$
$$= 2^{n-1} - X = 2^{n-1} + |X|$$

$$[X]_{补} = 2^n + X = 2^n - |X| = [X]_{反} + 1 = 2^{n-1} + \sum_{i=0}^{n-1}\overline{X}_i2^i + 1$$

$$[X]_{反} = 2^{n-1} + \overline{X}_{n-2}\overline{X}_{n-3}\cdots\overline{X}_1\overline{X}_0 = 2^{n-1} + \sum_{i=0}^{n-1}\overline{X}_i2^i$$

这就是说，$n-1$ 位二进制负数的原码为 n 位二进制数，最高位为符号位 1，其余 $n-1$ 位即为 $n-1$ 位二进制负数的数值部分（绝对值）。

$n-1$ 位二进制负数的补码为 n 位二进制数，它等于模 2^n 加上 $n-1$ 位二进制负数的真值（或说减去其真值的绝对值）。它也等于其原码除符号位不变（等于 1）外，其余各位按位求反，再在最低位加 1。要求读者熟练掌握求补码的方法。

$n-1$ 位二进制负数的反码为 n 位二进制数，它等于其原码除符号位不变（等于 1）外，各位全部变反。

（3）设 $n-1$ 位二进制数 X 为 0，则

$[+0]_{原}=000\cdots00$

$[-0]_{原}=100\cdots00$

$[+0]_{补}=2^n+00\cdots00=000\cdots00$

$[-0]_{补}=2^n-00\cdots00=000\cdots00$

$[+0]_{反}=000\cdots00$

$[-0]_{反}=111\cdots11$

以上均为 n 位二进制数，最高位为符号位。

这就是说，对数 0 其原码、反码均有正零和负零之分，而补码则无正零和负零之分。

(4) 移码的定义。

$$[X]_{移}=2^{n-1}+X\quad(2^{n-1}>X\geqslant-2^{n-1})$$

同一真值的移码和补码其数值部分完全相同，而符号位正好相反。最小的负数 -2^{n-1} 的移码为 000…00，最大的正数 $2^{n-1}-1$ 的移码为 111…11，$[+0]_{移}=[-0]_{移}=100\cdots00$。

2. 几种常用码制的优缺点比较

(1) 原码的主要优点是直观。但用原码作加减运算时，类似于笔算，处理过程非常繁琐，要求计算机的结构也极为复杂，因而，计算机中加减法运算不采用原码进行。原码表示法有时可用于乘除法运算。

(2) 补码的主要优缺点。补码的优点有二：

① 进行加减运算时，符号位可当作普通数字，与数值部分一起进行运算(若符号位有进位，则丢掉)，结果仍为补码形式。

② 两个补码数相减时，可以用减数(补码)变补(包括符号位在内变反加 1)与被减数相加来实现。这就是说，采用补码作加减法运算，计算机内部只要一个加法器即可，线路结构最简单。

补码的主要缺点是负数的表示形式不直观，与人们的习惯不一致，很容易判断出错。

(3) 反码表示法由于加减运算处理方法复杂，已不被采用。它多用来进行逻辑运算，以便完成一定的计算或控制功能。

(4) 移码克服了补码表示形式与人们习惯不一致的缺点，常用作 A/D 和 D/A 转换器的双极性编码，也可用在浮点数的阶码表示中。

3. 溢出判别

(1) 对带符号位的二进制数加减运算的溢出情况采用双高位判别法，具体判别方法见教材。

(2) 对不带符号位的二进制数，只有加法时才有溢出问题，即 2 个不带符号位数相加，若有进位，便表示有溢出。减法则无溢出问题。

4. 算术移位

最常用的是补码数进行算术移位。根据补码定义，正数移位时，无论左移还是右移，空缺位均补 0，负数移位时，左移最低位补 0；右移最高位补 1。

1.1.3　小数点问题

计算机中对小数点的处理有两种方法：定点法和浮点法。

1. 定点法

定点法表示一个带小数点的数时，参加运算的各原始数据小数点位置均一致，且由编程者事先根据具体情况规定好。可将小数点规定在整个字长的任何位置。由于小数点位置固定，加减运算时不需再考虑对位。

定点法的缺点是可表示的数值范围小，运算中为防止溢出，需事先考虑比例因子，较为麻烦。另外，用定点数作乘法时受字长影响会使精确度受限制。

2. 浮点法

用浮点法表示一个数时分尾数及阶码两部分。尾数通常为纯小数，尾数及阶码都有符号位。用浮点法表示数时，数的范围很大，运算中一般不需考虑溢出问题，这是浮点法的最大优点。用浮点法数进行四则运算均较麻烦，且费时。

(1) 浮点数加减运算时，尾数采用补码，阶码可用补码或移码表示。加减运算须先对阶(实则为小数点对位)，且应以大的阶为准对阶，然后才能进行加减运算。

浮点数加减法运算是较为重要的，读者应掌握对阶方法及浮点数和定点数的转换方法。

(2) 浮点数乘除法运算时，为避免损失精度，必须先将参加运算的数进行规格化处理，然后才能进行乘除法运算。要求读者掌握浮点数规格化的方法。

1.1.4　BCD 码及 ASCII 码

要求掌握常用 BCD 码的表示方法和特点。

ASCII 码在微机应用中十分重要，多用于输入/输出设备中，如屏幕显示代码等。要求读者熟悉数字、英文字母及常用符号(如回车 CR、换行 LF、空格 SP 及 $ 等)的 ASCII 码。

奇偶校验码在微机应用中也很重要，可增加微机运行的可靠性。如 IBM - PC 机采用的便是字长为 9 位的奇偶校验码。读者应熟悉奇校验码及偶校验码的定义及应用。

1.2　难点和重点

本章的难点和重点在于掌握好带符号数与不带符号数的补码运算。

1. $[Y]_{补}$、$[-Y]_{补}$及$[[Y]_{补}]_{补}$

$[Y]_{补}$的求法是将$[Y]_{原}$的符号位不变，其余各位变反加 1。

$[-Y]_{补}=[[Y]_{补}]_{变补}$，它的求法是将$[Y]_{补}$的各位(包括符号位在内)变反加 1。或先求出$[-Y]_{原}$，再求$[-Y]_{补}$。

$[[Y]_{补}]_{补}=[Y]_{原}$

例 1　设 $Y=-18$，则

$[Y]_{原}=10010010B$

$[Y]_{补}=11101110B$

$[-Y]_{补}=00010010B$($[-Y]_{原}=00010010B$，$\therefore [-Y]_{补}=00010010B$)

$[[Y]_{补}]_{补}=[11101110B]_{补}=10010010B=[Y]_{原}$

可见，$[[Y]_{补}]_{补}\neq[[Y]_{补}]_{变补}$

2. 补码的加减法运算

教材中已证明，采用补码表示法进行加减运算，不仅减法可变为加法运算，而且还可将带符号位数运算和不带符号位数运算统一起来。这就是说，不管参加运算的两个 n 位二进制数是带符号的补码形式，还是不带符号的数，对计算机来说，处理方法都是一样的。作加法时，直接相加即可；作减法时，用减数变补与被减数相加来实现。只不过两种情况下，运算结果的溢出判别和正负判别方法不同而已。对带符号位数来说，运算结果的溢出与否采用双高位判别法，正负情况采用最高位(符号位)来判别。对不带符号位数来说，只有加法才有可能发生溢出，以最高位是否有进位来判别。同时，对不带符号位数来说，只有减法结果才有正负之分，依是否有借位来判别。

例 2 设 $X=10000001B$，$Y=01001111B$

若将该两数视为不带符号位数，则 $X=129$，$Y=79$。$X-Y$ 及 $Y-X$ 的运算过程如下：

(1) $X-Y=129-79=50$

$X=10000001B$

$Y=01001111B$

$[-Y]_{补}=10110001B$

```
    1000 0001B     X
+   1011 0001B   [-Y]补
-------------------------
   10011 0010    [X-Y]补
```

由于减数变补与被减数相加时，最高位有进位，表示原码相减时，最高位无借位，结果为正，故

$$[X-Y]_{原}=[X-Y]_{补}=00110010B=50$$

因为是两个正整数相减，不可能发生溢出。

(2) $Y-X=79-129=-50$

$Y=01001111B$

$X=10000001B$

$[-X]_{补}=01111111B$

```
    01001111B     Y
+   01111111B   [-X]补
-------------------------
  (0)11001110B  [Y-X]补
```

由于减数变补与被减数相加时，最高位无进位，表示原码相减时最高位有借位，结果为负。故对$[Y-X]_{补}$再求补，得

$$[Y-X]_{原}=10110010B=-50$$

同样，对两个不带符号位的正整数相减，不可能发生溢出。

例 3 若将例 2 中 $X=10000001B$，$Y=01001111B$ 理解为带符号位的补码数，则 $[X]_{原}=11111111B=-127$，$Y=79$。$X-Y$ 及 $Y-X$ 的运算过程如下：

(1) $X-Y=-127-79=-206$，其绝对值>128，应有溢出发生。

$X=10000001B$

$Y=01001111B$

$[-Y]_{补}=10110001B$

$$\begin{array}{rl} & 10000001B \quad X \\ +& 10110001B \quad [-Y]_{补} \\ \hline & 100110010B \quad [X-Y]_{补} \end{array}$$

$C_S=1$ $C_P=0$，负溢出

根据双高位判别法，知该运算结果产生负溢出，结果不正确。

(2) $Y-X=79-(-127)=206$，其绝对值大于 127，应有溢出发生。

$Y=01001111B$

$X=10000001B$

$[-X]_{补}=01111111B$

$$\begin{array}{rl} & 01001111B \quad Y \\ +& 01111111B \quad [-X]_{补} \\ \hline & 011001110B \quad [Y-X]_{补} \end{array}$$

$C_S=0$ $C_P=1$，正溢出

由双高位判别法知运算结果产生正溢出，结果不正确。

比较例 2 及例 3 可看出对两个二进制数来说，不管它们表示的是不带符号位的数，还是带符号位的二进制补码数，其运算过程完全一样，即计算机将一视同仁，同样处理。只是结果的分析和处理不同而已。而这种对运算结果的不同分析和处理是由用户用程序来实现的。

例 4　对例 2 的两个数 $X=10000001B$ 及 $Y=0100111B$ 进行加法运算，且分析该两个数为不带符号位数和带符号位的补码数两种情况下的运算结果。

解：$X=10000001B$

$Y=01001111B$

$$\begin{array}{rl} & 10000001B \quad X \\ +& 01001111B \quad Y \\ \hline & 011010000B \quad [X+Y]_{补} \end{array}$$

$C_S=0$，$C_P=0$

(1) 当 X、Y 均为不带符号位的正整数时，$X+Y=129+79=206$，最高位无进位，故无溢出，事实上 $206<256$，确无溢出发生。

(2) 当 X、Y 均为带符号位的二进制补码数时，$X+Y=-127+79=-48$。对运算结果采用双高位判别法进行判断，由于 $C_S=0$，$C_P=0$，故无溢出发生。同时，运算结果的最高位(符号位)为 1，表示和为负数的补码，故对$[X+Y]_{补}$再求补得

$$[X+Y]_{原}=10110000B=-48$$

1.3　习题题解

本章第 1 和第 6 题的题解从略。

2. 将下列十进制数转换为二进制数：47；0.74；24.31；199。

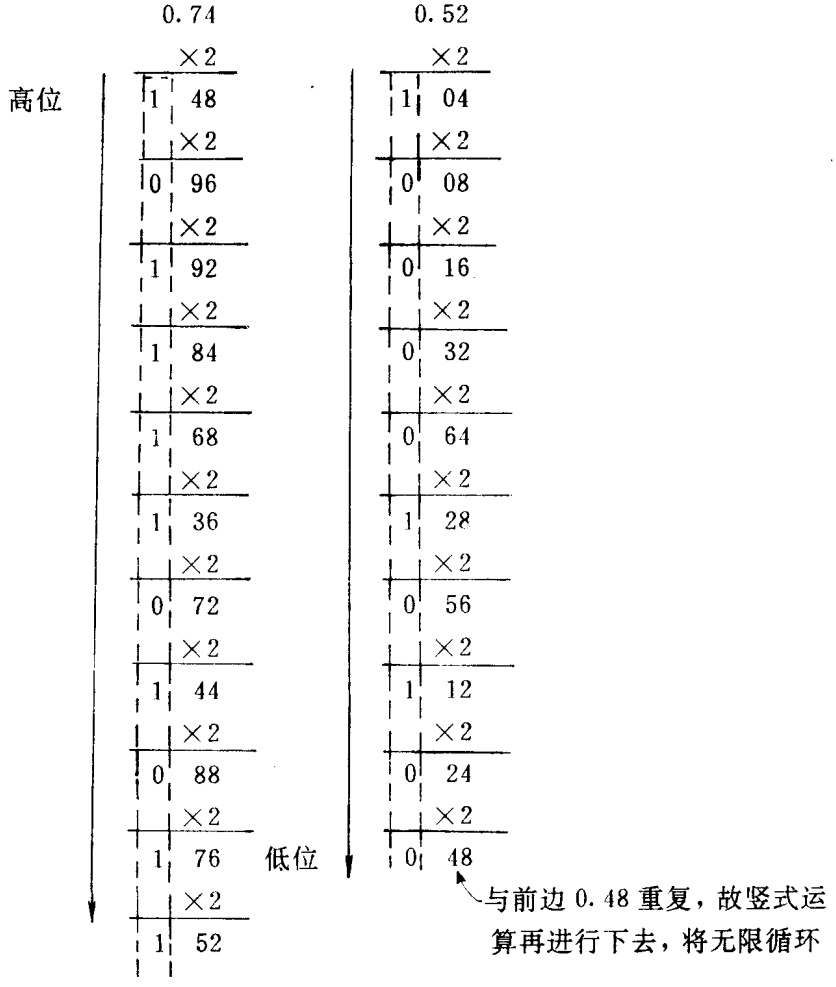
解：用竖式进行转换。

求$(47)_2$：

2 | 47
2 | 23　　余 1　低位
2 | 11　　余 1
2 | 5　　余 1
2 | 2　　余 1
2 | 1　　余 0　高位
0　　余 1

∴ 47＝101111B

求$(0.74)_2$：



$\therefore 0.74=0.1\underbrace{0111101011100001010001111010111000010100}_{\text{无限循环}}\cdots B$

求$(24.31)_2$：

$$
\begin{array}{r|l}
2 & 24 \\
2 & 12 \quad \text{余 } 0 \uparrow \text{低位} \\
2 & 6 \quad \text{余 } 0 \\
2 & 3 \quad \text{余 } 0 \\
2 & 1 \quad \text{余 } 1 \\
\multicolumn{2}{r}{0 \quad \text{余 } 1 \text{ 高位}}
\end{array}
$$

$$
\begin{array}{l}
0.31 \\
\times 2 \\
\hline
0.62 \quad \text{高位} \\
\times 2 \\
\hline
0.24 \\
\times 2 \quad \text{出现 0.48，竖式进行下去} \\
\hline
0.48 \quad \text{与 0.74 转换过程中 0.48} \\
\vdots \quad \text{后一样} \quad \text{低位}
\end{array}
$$

$\therefore 24.31=11000.010\underbrace{0111101011100001010001111010111000010100}_{\text{无限循环}}\cdots B$

求$(199)_2$：

$$
\begin{array}{r|l}
2 & 199 \\
2 & 99 \quad \text{余 } 1 \uparrow \text{低位} \\
2 & 49 \quad \text{余 } 1 \\
2 & 24 \quad \text{余 } 1 \\
2 & 12 \quad \text{余 } 0 \\
2 & 6 \quad \text{余 } 0 \\
2 & 3 \quad \text{余 } 0 \\
2 & 1 \quad \text{余 } 1 \\
\multicolumn{2}{r}{0 \quad \text{余 } 1 \text{ 高位}}
\end{array}
$$

$\therefore 199=11000111B$

3. 将下列二进制数转换为十进制数：

+101101.101B；−10110B；+10011B；−11011B

解：$101101.101B=1\times2^5+0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$
$+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$
$=32+8+4+1+0.5+0.125=45.625$

$-10110B=-\{[(1\times2+0)\times2+1]\times2+1\}\times2+0=-22$

$+10011B=+(1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0)=+19$

$-11011B=-(1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0)=-27$

4. 完成下列转换：

(1) 11011.101101B 转换成十六进制数；

(2) 374.971 转换成十六进制数；

(3) 8FH 转换成十进制数；

(4) BADH 转换成十进制数；

(5) 1979 转换成八进制数。

解：(1) 11011.101101B=0001 1011.1011 0100B=1B.B4H

（2）采用竖式转换：

整数部分

除数	被除数	余数	
16	374		
16	23	余 6	↑ 低位
16	1	余 7	
	0	余 1	高位

小数部分

	0.971
	×16
高位 15	536
	×16
8	576
	×16
9	216
	×16
3	456
	×16
7 ↓	296

	0.296
	×16
4	736
	×16
11	776
	×16
12	416
	×16
6	656
↓ 低位	×16
10	496
	⋮

∴ 374.971＝176.F89374BC6A…H

（3）8FH＝8×16＋15＝143

（4）BADH＝(11×16＋10)×16＋13＝2989

（5）列竖式转换：

除数	被除数	余数	
8	1979		
8	247	余 3	↑ 低位
8	30	余 7	
8	3	余 6	
	0	余 3	高位

∴ 1979＝3673Q

5. 设机器字长为 6 位，写出下列各数的原码、补码和反码：

10101　　11111　　10000

－10101　－11111　－10000

解：求解结果列于表 2－1。

表 2－1

真值	10101	－10101	11111	－11111	10000	－10000
原码	010101	110101	011111	111111	010000	110000
补码	010101	101011	011111	100001	010000	110000
反码	010101	101010	011111	100000	010000	101111

7. 设机器字长为 8 位，最高位为符号位，试对下列各算式进行二进制补码运算。

（1）16＋6＝？　　（2）8＋18＝？

（3）9＋(－7)＝？　　（4）－25＋6＝？

（5）8－18＝？　　（6）9－(－7)＝？

（7）16－6＝？　　（8）－25－6＝？