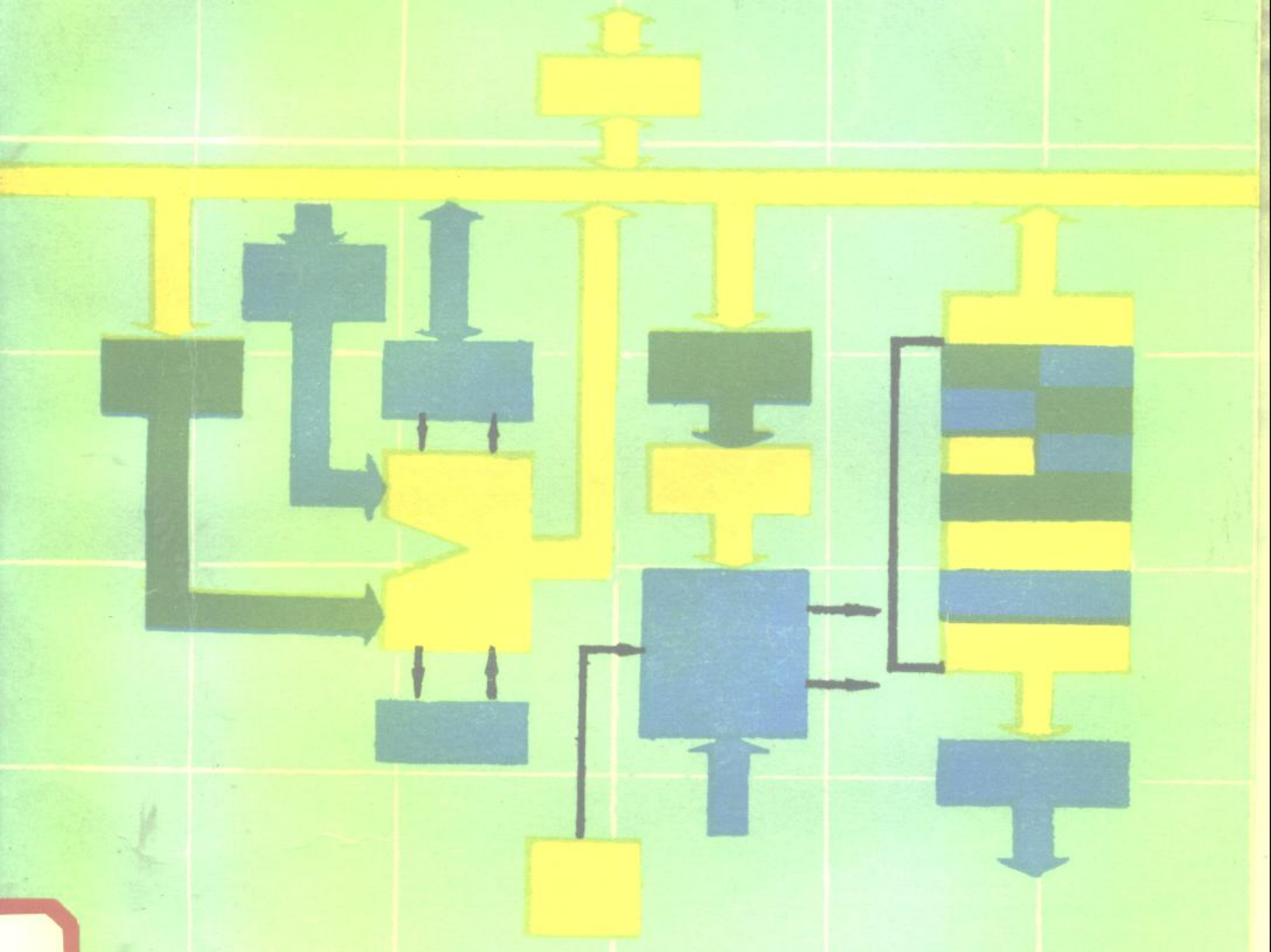
单片微型计算机原理与接口

阮家栋 何 政 主编



中国纺织大学出版社

单片微型计算机原理与接口

阮家栋　何　政　　主编

中国纺织大学出版社

内 容 提 要

本书主要根据上海高校计算机应用知识与应用能力等级考试三级（偏硬）考试大纲而编写的，同时考虑到单片机应用系统的特点，增加了考试大纲以外的内容。全书共分十章。第一章介绍微型计算机基础知识，第二、七章介绍MCS-51单片机的组成及工作原理，第三、四章介绍MCS-51指令系统和汇编语言程序设计，第五章介绍存贮器，第六章介绍输入/输出，第八章介绍常用I/O接口芯片，第九章介绍常用外设及其接口电路，第十章介绍单片机应用系统的开发方法。本书可作大专院校理工科学生“微机原理”的教材及工程技术人员的参考用书。

责任编辑　吴川灵
封面设计　赵　需

单片微型计算机原理与接口

阮家栋　何　政　主编

中国纺织大学出版社出版
（上海市延安西路1882号）

新华书店经销　江苏丹阳兴华印刷厂印刷

开本 787×1092　1/16　印张 17.25　字数 425 千字
1996年1月第1版　1996年1月第1次印刷

印数：0001—4000

ISBN 7-81038-063-X/TP·02　定价：19.80元

前　言

超大规模集成电路技术的发展把CPU、ROM、RAM及I/O接口集成在一个芯片上，组成了单片微型计算机。单片机由于体积小、功耗低、指令功能强、运行速度快、价格便宜等特点，被广泛地用于工业控制、计算机外设、智能仪表和家用电器中。单片机登上了计算机舞台，进入了工矿企业、科研单位、机关和高等学校。许多高校把单片微机作为“微机原理”课程的教学机型。上海普通高校计算机应用知识与应用能力等级考试三级(偏硬)的考试大纲把单片微机列入典型的微型计算机系统之一。本书在编写时参照“考纲”的要求，同时考虑到单片微机应用的特点，增加了部分“考纲”以外的内容。

全书共分十章。第一章介绍微型计算机基础知识。这一章是为非电类专业又未接触过计算机硬件的读者编写的。第二、七章介绍MCS-51单片机的组成及工作原理。第三、四章介绍MCS-51单片机指令系统和汇编语言程序设计。第五章介绍存贮器组成、工作原理及存贮器与CPU的接口。第六章介绍输入/输出和I/O接口的基本工作原理。第八章介绍常用I/O接口芯片，包括并行接口芯片、串行接口芯片、定时器/计数器芯片和A/D、D/A芯片。第九章介绍单片机应用系统中常用的输入/输出设备及其接口电路。第十章介绍单片机应用系统的开发方法。本书可作为大专院校理工科学生“微机原理”课程的教材，也可供从事单片微机开发应用的工程技术人员参考。

本书由阮家栋、何政主编，参加编写的有：阮家栋(第一、二、七、十章)、徐博铭(第三章)、徐安(第四章)、曹敏年(第五章)、王力生(第六章)、何政(第八章)、金家佑(第九章)。

本书在编写出版的过程中得到了上海铁道大学吴守箴教授和上海大学俞丽和教授的大力支持和帮助，在此表示衷心的感谢。

作者都希望把正确、无误的作品奉献给读者，但由于学识所限，本书错误和不妥之处在所难免，敬请读者批评指正。

编　者

1995年5月

目　录

1 微型计算机基础

1971年Intel公司首先推出了4位微处理器4004，接着又生产了8位微处理器8008。70年代中期，Intel公司、Motorola公司和Zilog公司分别推出了8位微处理器系列8085、M6800和Z80。70年代末、80年代初，这三家公司又推出了16位微处理器8086、M68000和Z8000。之后，Intel公司更是孤军突进，相继推出了80286、80386、80486和Pentium(586)，主宰了微机世界。微处理器把计算机中的中央处理单元(CPU)集成在一个芯片上，使计算机体积大大缩小。与此同时半导体存贮器和各种接口芯片也得到了飞速发展。这样，把CPU和ROM、RAM及接口芯片根据一定的规则连接起来，就组成了一台微型计算机。

把CPU和一定容量的ROM、RAM及输入/输出接口集成在一个芯片上，就形成了单片计算机(Single Chip Computer)。目前，单片机已广泛地用于工业控制、外设控制和智能仪表中，它的潜在能力已越来越为人们瞩目。

1976年9月Intel公司首先推出了MCS-48系列单片机。内存寻址范围小于4KB，片内ROM小于2KB，RAM小于128字节，只有并行I/O口，无串行I/O口。

1980年又推出了高性能的MCS-51系列8位单片机，寻址能力达到64KB，片内ROM可达4KB，RAM达256字节，除并行接口外还有串行I/O口，指令功能也更强。

1.1 计算机中的数和编码

计算机中，组成计算机的各种器件都基于半导体管的导通或截止两种状态，或者是低(0V)、高(+5V)两种电平。我们把其中的一种状态定义为“0”；另一种定义为“1”，于是基本的数字符号也只有两个：0和1。因此，目前在计算机中都采用二进制数。

1.1.1 数制

一个数字符号所表示的数值与它所在的位置有关。如一个十进制数4049.8，前一个4表示4000，而后一个4表示40。我们可以用如下的方法来表示4049.8这个数：

$$4049.8 = 4\times10^{3} + 0\times10^{2} + 4\times10^{1} + 9\times10^{0} + 8\times10^{-1}$$

等号左边的表示方法叫做并列表示法；等号右边的表示方法叫做多项式表示法。在多项式表示法中，10^{3}、10^{2}、…、10^{-1}等叫做“权”(Weight)。所以这种表示法又叫按权展开式。把这种表示方法一般化：

设K_i为其中的某一位($0 \leqslant K_i \leqslant 9$)，整数部分有$n$位，小数部分有$m$位，则：

$$
\begin{aligned}
(N)_{10} &= K_{n-1}K_{n-2}\cdots K_i\cdots K_1K_0.K_{-1}\cdots K_{-m} \\
&= K_{n-1}\cdot 10^{n-1}+K_{n-2}\cdot 10^{n-2}+\cdots+K_i\cdot 10^i \\
&\quad +\cdots+K_1\cdot 10^1+K_0\cdot 10^0+K_{-1}\cdot 10^{-1}+\cdots+K_{-m}\cdot 10^{-m} \\
&= \sum_{i=-m}^{n-1} K_i\cdot 10^i
\end{aligned}
$$

这里可能出现的数字符号有10个：0、1、……9。逢十进一。这里的“十”，我们称之为基数(Base或Radix)。

同样，我们也可以把二、十六等作为基数，则相应可以得到二进制和十六进制。

二进制中，可能出现的数字符号有2个：0和1，逢二进一。它的多项式表示为：

$$(N)_2=\sum_{i=-m}^{n-1} K_i\cdot 2^i \quad (K_i=0,1)$$

十六进制中，可能出现的数字符号有16个：0、1、……9、A、B、C、D、E、F，逢16进一。它的多项式表示为：

$$(N)_{16}=\sum_{i=-m}^{n-1} K_i\cdot(16)^i \quad (0\le K_i\le F)$$

一般地说，R进制的数，可能出现的数字符号有R个：0、1、……(R-1)，逢R进一。它的多项式表示为：

$$(N)_R=\sum_{i=-m}^{n-1} K_i\cdot R^i \qquad (0\le K_i\le R-1)$$

事实上，在R进制中，R这个符号是不出现的，因为逢R进一，所以R应该是“10”。十进制中的“十”，二进制中的“二”和十六进制中的“十六”都应表示为“10”。

在计算机书籍和程序中，二进制、十进制和十六进制都有可能出现，为避免混淆，通常在数的后面加一个字母B、D或H，分别表示二进制，十进制和十六进制。如：

$$1001B=(1001)_2$$
$$9824D=(9824)_{10}$$
$$26FEH=(26FE)_{16}$$

计算机采用二进制数，不只因为自然界中具有两个稳定状态的物理现象较多，还由于二进制的运算特别简单。其加法和乘法的运算规则为：

加法　0 + 0=0　0 + 1=1 + 0=1　1 + 1=10

乘法　0 × 0=1 × 0=0 × 1=0　1 × 1=1

实现上述运算的电子线路也比较简单。

计算机采用二进制数，而日常生活中人们习惯于十进制数，所以必须寻找一种十进制数与二进制数之间的转换方法。

十进制转换成二进制时，整数部分和小数部分要分别转换。

整数部分，先用2去除整数，以后用2逐次去除所得的商，直到商得0为止，依次得到的各个余数，即为从低位到高位的二进制整数。这一方法叫“除2取余法”。现举例说明如下：

例 1-1 将十进制数 57 转换成二进制数。

2|57　　余数
2|28 …… 1　　低位
2|14 …… 0
2|7 …… 0
2|3 …… 1
2|1 …… 1
0 …… 1　　高位

$(57)_{10}=(111001)_2$

小数部分，逐次用2乘小数部分，依次得到的整数部分即为从高位到低位的二进制小数。

例 1-2 将十进制数0.6875转换成二进制数。

	整数部分	
$0.6875\times 2=1.3750$	1	高位
$0.375\times 2=0.750$	0	
$0.75\times 2=1.50$	1	
$0.5\times 2=1.0$	1	低位

$(0.6875)_{10}=(0.1011)_2$

对于既有整数部分又有小数部分的十进制数，只要将整数部分和小数部分分别转换成二进制数然后相加即可。值得注意的是，有限位的十进制小数转换成二进制时可能是无限的。如:

$(0.8)_{10}=(0.1100011001100\cdots\cdots)_2$

遇到这种情况，一般根据精度要求取相应的位数。

在二进制转换成为十进制时，只要将二进制数的各位，乘上相应的权，然后相加即可。

例 1-3 将二进制数 1101.11 转换成十进制数。

$$\begin{aligned}(1101.11)_2 &= (1\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0+1\times 2^{-1}+1\times 2^{-2})_{10}\\ &= (8+4+0+1+0.5+0.25)_{10}\\ &= (13.75)_{10}\end{aligned}$$

由于二进制数的书写，输入和显示都不太方便，所以经常用一位十六进制数来表示4位二进制数。二进制数和十六进制数之间的转换是非常方便的。

在二进制转换成十六进制时，只要将二进制数的整数部分自右向左4位一组，小数部分自左向右4位一组，不足4位时用0填补，每组用相应的十六进制数替换即可。如:

$$(1101000101011)_2=(345.6)_{16}$$
$$(101010101111)_2=(2AB.C)_{16}$$
$$(1001011111101)_2=(12FE.8)_{16}$$

相反的转换也如此，只要把每位十六进制数用相应的4位二进制数表示，并去掉无效的0。

在计算机中，数制的转换都可以由程序自动完成。

1.1.2 带符号数的表示方法

任何一个数都由数值和符号(即正、负)两部分组成。在计算机中，为处理符号，对于数有各种不同的表示方法，称为码制。

在数学中，我们用“+”表示正数，“-”号表示负数。这种方法自然也可以用于二进制数。如正的1011可表示为+1011，负的1011可表示为-1011。然而，在计算机中，机器并不认识“+”号和“-”，它只认识“0”和“1”，所以就用“0”表示“+”；“1”表示“-”。于是+1011表示为01011，而-1011表示为11011。这种有符号数在机器中的表示形式就称为机器数，而原来的数则称为机器数的真值。

机器数有三种表示方法：原码、反码和补码。

1 原码

将真值中的“+”用“0”表示，“-”用“1”表示，所得机器数就是原码。在计算机中，我们常常把小数点固定在最高数位(MSB)的前面，所以+1011的原码为0.1011，而-1011的原码为1.1011。写成一般表达式为：

$$[x]_{原}=\begin{cases}x & x\geq 0\\ 1-x & x\leq 0\end{cases}$$

当x=0时，其原码有两种表示方法：

$[+0]_{原}=0.0000$

$[-0]_{原}=1.0000$

在进行原码的加、减运算时，两个同号数相加，符号不变，尾数相加；若是异号数相加或同号数相减，则先判别其绝对值的大小，然后大数减去小数，最后判定符号。

在机器中，要判别两个数的大小，就必须先做一次减法，否则无从知道数的大小。要设计一个这样的减法线路是很复杂的。为此提出了反码和补码。

2 反码

在用反码表示时，正数的反码同原码；负数的反码，符号为1，数字部分变反。写成表达式为：

$$[x]_{反}=\begin{cases}x & x\geq 0\\ 2+x-2^{-n} & x\leq 0\end{cases}\quad(\text{n为二进制的位数})$$

当x=0时，$[+0]_{反}=0.0000$

$[-0]_{反}=1.1111$

反码运算时，先把变量用反码表示，然后进行反码加法，符号当作一个数一起参加运算。符号位产生进位时，这一进位加到末位上去。

例 1-4 已知：x=0.1010 y=0.0110

求：x-y

解：$[x]_{反}=0.1010$ $[-y]_{反}=1.1001$

$$\begin{array}{r}[x]_{反}=\ 0.1010\\ +)[-y]_{反}=\ 1.1001\\ \hline [x-y]_{反}=\boxed{1}0.0011\end{array}\longrightarrow 0.0100\quad x-y=0.0100$$

反码中，0的表示不是唯一的，补码则克服了这一缺点。

3 补码

正数的补码同原码。负数的补码，符号位不变(仍为1)。数字部分变反，末位加1。写成表达式为：

$[x]_{补}=x+2\quad(\mathrm{mod}2)$

当$x \geqslant 0$时，$[x]_{补}=x+2=x \pmod 2$

当$x \leqslant 0$时，$[x]_{补}=x+2=2-|x| \pmod 2$

对于0，+0和-0的补码都是0.0000。

在补码运算时，先把变量连同它的符号一起变补，然后进行补码加法，符号当作一位数一起参加运算，符号位出现的进位丢失，所得的结果以补码形式出现。若结果为负数，再经一次变补，以求原码；若为正数，补码同原码。

例1-5 已知：$x=0.0101$ $y=0.1011$

求：(1)$x-y$ (2)$y-x$

解：(1) $[x]_{补}=0.0101$ $[-y]_{补}=1.0101$

$$\begin{array}{r} [x]_{补}= 0.0101 \\ +)[-y]_{补}= 1.0101 \\ \hline [x-y]_{补}= 1.1010 \end{array} \qquad x-y=-0.0110$$

(2) $[y]_{补}=0.1011$ $[-x]_{补}=1.1011$

$$\begin{array}{r} [y]_{补}= 0.1011 \\ +)[-x]_{补}= 1.1011 \\ \hline [y-x]_{补}=\boxed{1}0.0110 \end{array} \qquad y-x=+0.0110$$

丢失

由此可知，补码表示法可变减法运算为加法运算。

1.1.3 定点数与浮点数

在计算机中表示小数点位置的方法通常有两种：一是小数点的位置固定，这种方法叫定点法；另一种方法小数点的位置不固定，叫浮点法。

在定点法中，小数点位置通常放在最高数位之前，符号位之后。如16位字长的二进制数，它的小数点位置在D_{14}之前，D_{15}之后。

浮点表示时，小数点的位置可任意移动。如11.011可写成：

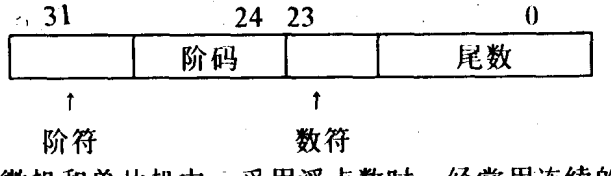
$$11.011=110.11\times 2^{-1}=1101.1\times 2^{-2}=0.11011\times 2^{2}$$

它的普遍形式是：

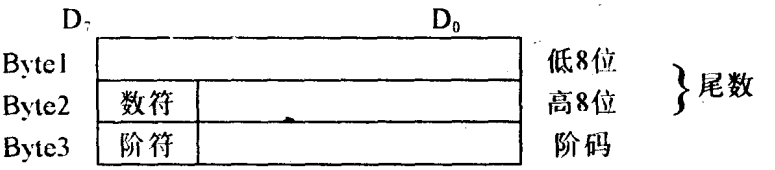
$$2^{E}\times F$$

其中，E叫做阶码，F叫做尾数。

在计算机中，阶码和尾数是分两部分表示的。若字长为32位，可用8位表示阶码，24位表示尾数。阶码和尾数中各有1位表示符号。



在8位微机和单片机中，采用浮点数时，经常用连续的3个字节表示一个浮点数。它们分别表示尾数的低8位、尾数的高8位和阶码。尾数的最高位和阶码的最高位分别表示数符和阶符。



大多数计算机都规定尾数是纯小数。但是尽管如此，一个数的浮点表示仍将是多样的。为此，规定尾数部分小数点后的第一位必须为1而不能为0。这样，一个数的浮点表示就是唯一的了。这种表示方法叫做浮点数的规格化表示。

1.1.4 数和字符的代码表示

1 十进制数的二进制编码

目前的电子数字计算机都是在二进制的基础上组织起来的，但是在日常生活中有时还必须用十进制数来表示。比如前面讲到的0.8，如果表示货币的8角，就不能用二进制的0.11001100……元来表示。在这种情况下，可用二进制数对十进制数进行编码，即二-十编码或BCD码(Binary Coded Decimal Code)。

BCD码有多种表示方法，常用的编码方案是8421码(NBCD码)。它的编码原则是十进制数的每一位用4位二进制数来表示。且4个数位上的权符合二进制的位值法则，即从左到右分别是8、4、2、1，所以叫8421码(见表1-1)。

表1-1 NBCD码

十进制数	BCD码
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

尚有1010、1011、1100、1101、1110和1111这6个编码舍去不用。在用BCD码进行运算时，上述6个编码属于非法码。大多数微机和单片机中都有十进制调整指令，在遇到非法码时，可用十进制调整指令处理。

2 ASCII码

ASCII码是目前国际上普遍采用的美国标准信息交换码（American Standard Code for Information Interchange），它用7位二进制编码表示数字0～9、英文字母及其他可打印字符。在计算机中通常用一个字节表示一个ASCII码。如30H～39H表示数字0～9，41H～54H表示大写字母A～Z等。七位二进制数可以表示128个编码，除用于可打印字符外，尚有一些表示打印机及CRT屏幕的控制。

在通信过程中，bit7经常用作奇偶校验。有些机器用bit7=1表示一些特殊图形符号，而标准的ASCII字符，bit7=0。当一个字符或图形用8位二进制数表示时，也可以用第9位作奇偶校验位。

1.1.5 算术运算

通过前面的学习，我们已经知道，计算机中最基本的运算方法是二进制补码的运算。如果是无符号数或BCD码，也可以不化成补码，而直接进行无符号数或BCD码的运算。至于高等数学中的微积分和初等函数的运算，通常是由子程序将它们逐步化解成二进制的四则运算。从硬件的角度看，计算机中的运算器，只要能进行加法，移位和变补（二进制数字位变反和末位加1），就可以完成所有数值运算。也有一些计算机，为加快运算速度，具有进行乘法和除法的运算部件。

这一节我们介绍计算机中的算术运算方法。

1 无符号数的运算

无符号数有两种基本运算：加法和减法。利用加法和减法，就可以进行乘法、除法和其它数值运算。

(1) 二进制加法

二进制加法的运算规则为：

0 + 0 = 0

0 + 1 = 1 + 0 = 1

1 + 1 = 0　　进位 1

1 + 1 + 1 = 1　进位 1

设有两数0101和0111相加，则加法过程如下：

```
    110    (进位)
   0101    (被加数)
 + 0111    (加数)
 ------
   1100    (和)
```

两个数相加时，每一位有3个数参加运算：被加数，加数和来自低位的进位，其结果是本位和以及向高位的进位。在进行二进制运算时，必须注意二进制数的长度（即二进制数的位数），其和不能超过规定的长度，否则会产生“溢出”，不能得到正确的结果。如果规定字长为8位，两个无符号二进制数 10110101（相当于十进制数181）和10001100（相当于十进制数140）相加：

```
  01111000    (进位)
  10110101    (被加数)
+ 10001100    (加数)
 ---------
 [1]01000001  (和)
  ↓
 丢失
```

其结果为01000001（相当于十进制数65），这显然是错误的。其原因是最高位产生的进位被丢失了，相当于丢失了一个十进制数256。

(2) 二进制减法

二进制减法的运算规则为：

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = 1　　向高位借 1

0 - 1 - 1 = 0　　向高位借 1

若有两数11000100和00100101相减，则减法过程如下：

```
  11000100  (被减数)
  00100101  (减数)
  01111110  (借位)
  10011111  (差)
```

与加法类似，两个数相减时，每一位有3个数参加运算：被减数、减数和来自低位的借位，其结果是本位差以及向高位的借位。

(3) 二进制乘法

二进制乘法的运算规则为：

0 × 0 = 0 × 1 = 1 × 0 = 0

1 × 1 = 1

二进制的乘法与十进制类似，如下所示：

```
    1101      (被乘数)
  × 1011      (乘数)
  ------
    1101
   1101
  0000
 1101
--------
 10001111     (积)
```

在计算机中实际的做法是采用部分积右移的方法，即当乘数为1时，加被乘数，部分积右移一位；乘数为0时，不加被乘数，部分积仍右移一位。如下所示：

乘数	被乘数	部分积
1011	1101	0000
乘数最低位为1，加被乘数		+1101
		1101
部分积右移一位		01101
右数第二位为1，加被乘数		+1101
		100111
部分积右移一位		100111
右数第三位为0，不加被乘数		
部分积仍右移一位		100111
乘数最高位为1，加被乘数		+1101
		10001111
部分积右移一位		10001111

其结果与前述相同，两个n位二进制数相乘，其积为$2n$位。

(4) 二进制除法