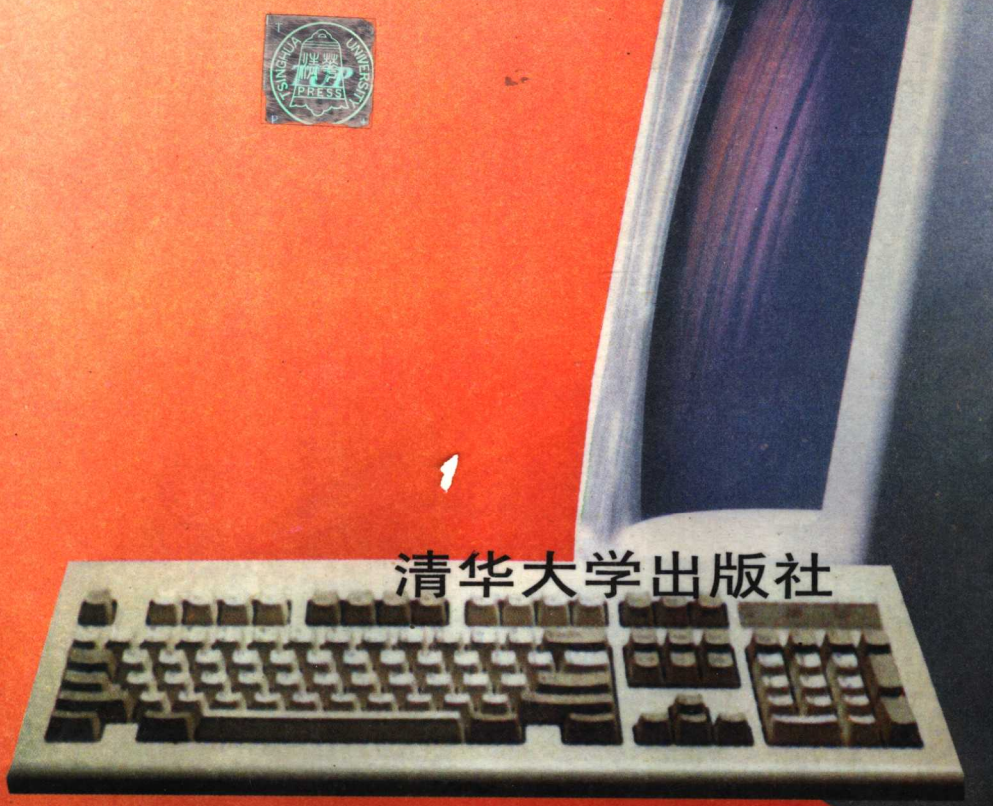
清华大学
计算机基础教育系列教材

张菊鹏等 编著

计算机硬件技术基础



清华大学出版社

清华大学计算机基础教育系列教材

计算机硬件技术基础

张莉鹏等 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书是清华大学非电专业"计算机硬件技术基础"课程的教材。书中首先介绍了微型计算机的基础知识；分析了 8086/8088 微处理器的寻址方式、指令系统、汇编语言程序设计和操作时序；然后以 IBM PC/XT 为核心阐述了微型计算机系统中有关硬件的基础知识，逐一讲述了 RAM 与 ROM 存储器、计数器/定时器 8253、中断控制器 8259A、并行接口 8255A 和串行接口 8251 等芯片的原理与应用；最后介绍了 D/A 和 A/D、微型计算机应用开发及一些常用外设的原理。通过对上述基本知识的学习和总结，可使读者对微型计算机系统有一个完整和全面的了解。

本书是清华大学计算机基础教育的系列教材之一，适于作为大专院校非计算机类各专业学生的教材，也可作为成人高等教育的培训教材及广大科技工作者的自学参考书。

图书在版编目(CIP)数据

计算机硬件技术基础/张菊鹏等编著．—北京：清华大学出版社，1996
ISBN 7-302-02346-8

Ⅰ．计… Ⅱ．张… Ⅲ．微型计算机-硬件-高等学校-教材 Ⅳ．TP303

中国版本图书馆 CIP 数据核字(96)第 20765 号

出版者：清华大学出版社（北京清华大学校内，邮编 100084）
印刷者：北京丰华印刷厂
发行者：新华书店总店北京科技发行所
开　本：787×1092 1/16　**印张**：26 5/8　**字数**：660 千字
版　次：1997 年 1 月 第 1 版　1997 年 1 月 第 1 次印刷
书　号：ISBN 7-302-02346-8/TP・1167
印　数：0001—8000
定　价：22.80 元

前　言

计算机基础教育是面向非计算机类各专业学生的计算机教育。与其它传统的基础课(如数学、物理、化学、外语等)一样,计算机教育已成为大学本科生基础教学的重要组成部分。

计算机基础教育大致可分为三个层次:即"计算机文化基础"、"计算机技术基础"和"计算机应用基础"。

"计算机文化基础"课程是为了培养人们的"计算机意识",使人们具备必要的计算机基础知识,掌握计算机的基本操作技能,以便于在未来信息化社会中更好地工作、学习和生活。

"计算机技术基础"课程则是为同学们毕业后在各自专业领域从事一些计算机的应用开发工作,为今后进一步学习计算机软、硬件知识与技术打下一个较为全面的基础。

而"计算机应用基础"课程则是针对当前计算机的主要应用领域,将那些通用的、具有普遍意义的内容传授给学生,使他们初步掌握计算机应用中一些必要的知识、方法、工具和技能。

本书是上述第二层次——"计算机技术基础"中有关硬件方面的适用教材。

本书以 8086/8088 CPU 的 16 位微型计算机系统 IBM PC/XT 作为基础,介绍 CPU 的结构、指令系统、存储器及输入/输出接口电路。这一方面是因为对初学者来说,学习微型计算机硬件原理以此为起点较易入门;另一方面 IBM PC/XT 曾经是最具有代表性的主流机型,其设计思想、体系结构、接口芯片的安排及信号关系等仍被高等微型计算机设计者在设计时作为参考因素。由于 Intel 80X86 系列机具有兼容性,因此了解它的工作原理后,有利于掌握微型计算机及其系统的概念,并为进一步学习和应用 32 位微型计算机打下良好的基础。

本书是编者以清华大学非电专业"计算机硬件技术基础"课的讲稿为基础而编著的教材。讲课学时安排为 48 至 64 学时。内容取材上注意做到少而精;叙述方法上力图由浅入深、循序渐进;章节安排上尽量使之独立成章,供讲授时选择,以适应不同读者的需要。第十章及第一、九章部分内容是针对非计算机专业的特点而增加的自学参考知识。

本书第一、二、四、九章由张菊鹏编写,第三、七、十章由沈永林编写,第五、六、八章由李芙英编写,全书由张菊鹏修改定稿。由于编者水平有限,书中难免存在错误及不妥之处,敬请读者提出宝贵意见。

编　者

于 1996 年 2 月

目　录

第一章　微型计算机基础知识

第一节　计算机中的数和编码系统

一、常用进位计数制

在计算机中，数是用二进制表示的，由一串“0”或“1”的二进制数构成的代码是计算机唯一能识别的机器语言。但用二进制数来表示一个较大的数时，既长又不好记，为了阅读和书写方便，故计算机中也广泛采用十六进制数。人们习惯于十进制数，故通过输入设备送入计算机的数，以及计算机通过输出设备送出的计算结果，一般都是用十进制表示的。

下面简单介绍一下这几种计数制。

（一）十进制数

十进制数共有 10 个数字符号，为 0～9，逢十进位。

一个十进制数 D 可表示为：

$$\begin{aligned} D &= D_{n-1} \times 10^{n-1} + D_{n-2} \times 10^{n-2} + \cdots + D_0 \times 10^0 + \\ &\quad D_{-1} \times 10^{-1} + \cdots + D_{-m} \times 10^{-m} \\ &= \sum_{i=-m}^{n-1} D_i \times 10^i \end{aligned}$$

其中，D_i 表示第 i 位的数码 0～9 中的 1 个；m 和 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；10 为十进制数的基数，10^i 称为十进制数的权。

例　十进制数 6543.82 可表示为：

$$6543.82 = 6\times10^3 + 5\times10^2 + 4\times10^1 + 3\times10^0 + 8\times10^{-1} + 2\times10^{-2}$$

（二）二进制数

只有两个不同的数字符号 0 和 1，逢二进位。

一个二进制数 B 可表示为：

$$\begin{aligned} (B)_2 &= B_{n-1} \times 2^{n-1} + B_{n-2} \times 2^{n-2} + \cdots + B_0 \times 2^0 + \\ &\quad B_{-1} \times 2^{-1} + \cdots + B_{-m} \times 2^{-m} \\ &= \sum_{i=-m}^{n-1} B_i \times 2^i \end{aligned}$$

其中，B_i 只能取 1 或 0；2 为基数，2^i 称为二进制数的权；m 和 n 的含意同十进制表达式。于是一个二进制数可用它的权展开式来表示。

例　$$\begin{aligned} (10110.011)_2 &= 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^{-2} + 1 \times 2^{-3} \\ &= (22.375)_{10} \end{aligned}$$

（三）十六进制数

共有 16 个数字符号。用 0～9 和 A～F 来表示，逢十六进位。

一个十六进制数 H 可表示为：

$$(H)_{16}=H_{n-1}\times 16^{n-1}+H_{n-2}\times 16^{n-2}+\cdots+H_0\times 16^0+H_{-1}\times 16^{-1}+\cdots+H_{-m}\times 16^{-m}$$

$$=\sum_{i=-m}^{n-1}H_i\times 16^i$$

其中，H_i 取值在 0～9 和 A～F 的范围内；16 为基数，16^i 称为十六进制数的权；m 和 n 的含意同十进制数表达式。

同样，对于一个十六进制数，亦可用它的权展开式来表示。

例

$$(64)_{16}=6\times 16^1+4\times 16^0=(100)_{10}$$

$$(FFFF)_{16}=15\times 16^3+15\times 16^2+15\times 16^1+15\times 16^0$$

$$=(65535)_{10}$$

目前，绝大部分微型计算机的字长是 4 的整数倍，故广泛采用十六进制数来表示一个数，但机器中仍是采用二进制数。不过二进制数和十六进制数之间有一种特殊关系，即 $2^4=16$，这就带来一个很大的方便：一位十六进制数可以用四位二进制数来表示，且它们之间的关系既直接又唯一。

十进制数，二进制数及十六进制数之间的关系如表 1-1 所示。

表 1-1　各种数制的对照表

十进制数	二进制数	十六进制数	十进制数	二进制数	十六进制数
0	0000	0	9	1001	9
1	0001	1	10	1010	A
2	0010	2	11	1011	B
3	0011	3	12	1100	C
4	0100	4	13	1101	D
5	0101	5	14	1110	E
6	0110	6	15	1111	F
7	0111	7	16	10000	10
8	1000	8			

为了区别这三种常用数制，可在数的右下角用数字来注明数制，或在数字后面加一字母来表示。一般用字母 B(binary)表示二进制数制；用字母 D(decimal)表示十进制数制；用 H(hexadecimal)表示十六进制数制。

二、各种进位制数之间的转换

（一）二进制数、十六进制数转换为十进制数

采用上述进位制各自的权展开式即可得到对应的十进制数。

例 $1101.101B = 1\times2^3+1\times2^2+1\times2^0+1\times2^{-1}+1\times2^{-3}$

$=13.625$

$0BF4H = 11\times16^2+15\times16^1+4\times16^0=3060$

（二）十进制数转换为二进制数、十六进制数

1. 将十进制数转换为二进制数

转换方法：将整数部分不断地用 2 去除，直至商为 0 为止。记下各次所得余数，以最后余数为最高数位，依次排列，即得到所转换的二进制数的整数部分；将十进制数的小数部分不断地用 2 乘，直至满足所要求的精度或小数部分等于零为止。把每次乘积的整数部分记下，以最初的整数为最高数位，依次排列，即得到所转换的二进制数的小数部分。

例 将十进制数 117.625 转换为二进制数

	余数		积的整数部分
117/2=58	$b_0=1$（最低位）	0.625	
58/2=29	$b_1=0$	× 2	$b_{-1}=1$（最高位）
29/2=14	$b_2=1$	1.250	
14/2=7	$b_3=0$	× 2	
7/2=3	$b_4=1$	0.500	$b_{-2}=0$
3/2=1	$b_5=1$	× 2	
1/2=0	$b_6=1$（最高位）	1.000	$b_{-3}=1$（最低位）

所以 $(117.625)_{10}=(1110101.101)_2$

2. 将十进制数转换为十六进制数

转换方法 1：类似于十进制数转换为二进制数的方法，但将整数部分不断地用 16 除，并记下余数；将小数部分不断地用 16 乘，记下乘积的整数部分。

例 将十进制数 222.6875 转换为十六进制数

	余数		积的整数部分
222/16=13	$H_0=14$（最低位）	0.6875	
13/16=0	$H_1=13$（最高位）	× 16	
		11.0000	$H_{-1}=11$

所以 $(222.6875)_{10}=(DE.B)_{16}$

转换方法 2：先将十进制数转换为二进制数，再将二进制数转换为十六进制数，因后者的转换非常容易。

（三）二进制数与十六进制数之间的相互转换

1. 二进制数转换为十六进制数

前面已述，一位十六进制数可用四位二进制数来表示，故将二进制数转换为十六进制数的方法很简单，只要将整数部分从小数点往左，每 4 位一组，最左面一组不足 4 位时在前面补 0；小数部分从小数点往右，每 4 位一组，最右面不足 4 位时从后面补 0。这样分组后，将每

组的四位二进制数用相应的十六进制数表示即可。

例 二进制数 1111011110.10111 可如下分组：

$$\underbrace{0011}_{3}\ \underbrace{1101}_{D}\ \underbrace{1110}_{E}.\underbrace{1011}_{B}\ \underbrace{1000}_{8}$$

所以 1111011110.10111B＝3DE.B8H

2. 十六进制数转换为二进制数

只要将每位十六进制数用对应的二进制数表示即得。

例 DE.BH 可表示为

$$\underbrace{1101}_{D}\ \underbrace{1110.}_{E.}\ \underbrace{1011}_{B}$$

所以 DE.BH ＝ 11011110.1011B

三、二进制数的运算

二进制数的运算规则与十进制数类似，但因二进制数只有 0 和 1 两个数，故运算规则比十进制数要简单得多。

（一）加法运算

二进制数的加法规则：

① 0＋0＝0

② 0＋1＝1＋0＝1

③ 1＋1＝10　有进位

④ 1＋1＋1＝11　有进位

例 有两数 1010 和 1111 相加，其加法过程为：

进　位	11
被加数	1010
加　数	＋1111
	11001

（二）减法运算

二进制数的减法规则：

① 0－0＝0

② 1－0＝1

③ 1－1＝0

④ 0－1＝1　有借位

例 10101010 减去 00110100。减的过程如下：

借　位	11101
被减数	10101010
减　数	－00110100
	01110110

（三）乘法运算

二进制的乘法规则为：

① 0×0=0

② 0×1=0

③ 1×0=0

④ 1×1=1

从乘法规则可知，只有当两个 1 相乘时，其积才为 1，其它情况下乘积均为 0，比起十进制乘法运算规则要简单多了。

例 1100 与 1001 相乘。

我们仿照十进制乘法的过程，从低位开始乘起，但遵循二进制的乘法规则，过程如下：

```
    1100        被乘数
  × 1001        乘  数
    1100   ┐
   0000    ├    部分积
  0000     ┘
 1100
1101100         乘  积
```

从这个实例可以看出其操作过程：从乘数低位开始，用乘数每一位去乘被乘数，若乘数位为 1，相乘所得的中间结果（称为部分积）即为被乘数，故只需将被乘数照抄；若乘数为 0，则所得中间结果为 0。要注意将每次中间结果的最后一位与相应的乘数位对齐。最后将这些部分积加起来，其结果即为所得乘积。

在机器中，二进制乘法是利用加法器的加操作和移位操作来实现的。设先将部分积置为零，过程步骤为：

乘数	被乘数	部分积
1001	1100	0000
→乘数为1，加被乘数至部分积上		+1100
被乘数左移1位	11000	1100
→乘数为0，不加被乘数		
被乘数左移1位	110000	
→乘数为0，不加被乘数		
被乘数左移1位	1100000	
→乘数为1，加被乘数（已左		1100
移后的）至部分积上		+1100000
得乘积		1101100

也可以从乘数高位乘起，所得结果亦同，只是要注意将每次中间结果的最高位与相应的乘数位对齐，仍以上面乘法为例：

$$
\begin{array}{r}
1100 \\
\times 1001 \\
\hline
1100 \\
0000 \\
0000 \\
1100 \\
\hline
1101100
\end{array}
$$

在机器中，这种从乘数高位乘起的方法，可采用部分积右移的办法来实现，先将部分积置为零，过程步骤为：

乘数	被乘数	部分积
1001	1100	0000
→乘数为1，加被乘数		+1100
部分积右移1位		1100
→乘数为0，不加被乘数		01100
部分积右移1位		001100
→乘数为0，不加被乘数		
部分积右移1位		0001100
→乘数为1，加被乘数		+1100
得乘积		1101100

这种运算方法的优点是，虽是两个 n 位相乘，乘积为 2n 位，但只要 n 个加法器，而前一种运算方法，由于 2n 位都可能要有相加的操作，因此需要 2n 个加法器。

（四）除法运算

与十进制除法类似，步骤如下：

① 从被除数最高位开始检查，找到大于除数的位数，找到这一位时，商记为 1，并将选定的被除数减除数，得余数；

② 然后把被除数的下一位移到余数上，将余数减除数，若够减，商为 1；若不够减，商为 0。

③ 重复步骤②，直至把被除数的所有位都下移完为止。

例

$$
\begin{array}{r}
\quad 1001 \\
110\overline{)\,111011} \\
\quad 110 \\
\hline
\quad 1011 \\
\quad 110 \\
\hline
\quad 101
\end{array}
$$

结果：商为 1001，余数为 101。继续往下除，商为小数，若除不尽，根据精度要求，取足位数即可。

（五）“与”运算

规则为按位进行“与”运算；进行“与”操作的两位均为 1，则“与”的结果为 1；两位中有一

位为 0，则“与”的结果为 0。

“与”运算又称逻辑乘，一般用符号“∧”或“·”来表示。

例

$$\begin{array}{r} 11011001 \\ \wedge\ 00110100 \\ \hline 00010000 \end{array}$$

（六）“或”运算

规则为按位进行“或”运算；进行“或”操作的两位中有一位为 1，则“或”的结果为 1；两位均为 0，则“或”的结果为 0。

“或”运算又称逻辑加，一般用符号“∨”或“+”来表示。

例

$$\begin{array}{r} 11011001 \\ \vee\ 00110101 \\ \hline 11111101 \end{array}$$

（七）“异或”运算

规则为按位进行“异或”运算；进行“异或”操作的两位不相同时（即一位为 1，另一位为 0），则“异或”结果为 1，两位相同时（即都为 1，或都为 0），则“异或”结果为 0。

“异或”运算一般用符号“⊕”表示。

例

$$\begin{array}{r} 11011001 \\ \oplus 00110100 \\ \hline 11101101 \end{array}$$

四、计算机中带符号数的表示

上面提到的二进制数，均未涉及符号问题，故是一种无符号数的表示。但数是有正有负的，通常用符号“+”表示正，用符号“−”表示负，那么在计算机中数的正、负如何来表示呢？

我们知道在计算机中二进制数 1 或 0 是由器件的两种不同的稳定状态（当然这两种不同的稳定状态能按要求相互转换），即高、低电平来表示的，所以数的正、负号也只能由这两种不同的状态来表示，也就是说数的符号在计算机中也数码化了。通常规定一个数的最高位代表符号，该位为“0”表示正，该位为“1”表示负。以 8 位字长为例，D_7 为符号位，$D_6 \sim D_0$ 为数字位，若字长为 16 位，D_{15} 为符号位，$D_{14} \sim D_0$ 为数字位。

例 +0110100　　代表十进制数为+52

−0110100　　代表十进制数为−52

而在计算机中

00110100　　代表+52

10110100　　代表−52

为了区别这两种表示形式，我们把已经数码化了的带符号数称为机器数，如 00110100 和 10110100 就代表机器数，而把原来的数值称为机器数的真值。如例子中的+0110100 和

−0110100 就是真值。

为了运算方便，机器数有三种表示法，即原码、反码和补码。下面分别加以讨论。

（一）原码

正数的符号位用 0 表示，负数的符号位用 1 表示，其余各位表示数值本身，这样表示的机器数就称为原码。

1. 正数表示法

例 X＝＋52

$$[X]_{原}= \underset{\text{符号位}}{0}\ \underbrace{0110100}_{\text{二进制数值}}$$

2. 负数表示法

例 X＝－52

$$[X]_{原}= \underset{\text{符号位}}{1}\ \underbrace{0110100}_{\text{二进制数值}}$$

3. 数 0 表示法

0 在原码中有两种表示法，即＋0 和－0：

$$X=+0 \qquad [X]_{原}=0\ 0000000$$

$$X=-0 \qquad [X]_{原}=1\ 0000000$$

（二）反码

1. 正数表示法

正数的反码表示与原码相同，即符号位用 0 表示正，其余位为数值位。

例 X＝＋0110100

$$[X]_{反}=[X]_{原}=00110100$$

2. 负数表示法

负数的反码表示：除符号位不变外(即仍为 1)，其余各位按位取反，即得到它的反码。

例 X＝－0110100

$$[X]_{反}=11001011$$

3. 数 0 表示法

根据原码中的＋0 和－0 可得到两种表示法：

$$X=+0, \qquad [X]_{反}=00000000$$

$$X=-0, \qquad [X]_{反}=11111111$$

（三）补码

1. 正数表示法

其补码表示与原码相同。

例 X＝＋0110100

$$[X]_{补}=[X]_{原}=00110100$$

2. 负数表示法

负数的补码定义为：$[X]_{补}=[X]_{反}+1$。即一个负数的补码等于它的反码加 1。注意此处加 1 是在最低位加 1。

例 X＝－0110100，由定义：

$$[X]_{补}=[X]_{反}+1=11001011+1=11001100$$

同理

$$[-2]_{补}=[-2]_{反}+1=11111101+1=11111110$$

$$[-127]_{补}=[-127]_{反}+1=10000000+1=10000001$$

3. 数 0 表示法

$$X=+0,\quad [+0]_{补}=00000000$$

$$X=-0,[-0]_{补}=[-0]_{反}+1=11111111+1=00000000$$

对 8 位字长来说，最高位的进位被舍掉，故对 8 位字长来说，其$[+0]_{补}=[-0]_{补}=$00000000，即 8 位补码中只有一个 0。

4. 8 位补码中的一个特殊数 10000000

此数在 8 位补码中定义为－128。这样，8 位数的补码范围为－128～＋127。

有关 8 位二进制数的原码、反码及补码表示请见表 1-2。

表 1-2　8 位二进制数的原码、反码及补码表示

二进制数码表示	无符号二进制数	原　码	补　码	反　码
00000000	0	＋0	＋0	＋0
00000001	1	＋1	＋1	＋1
00000010	2	＋2	＋2	＋2
⋮	⋮	⋮	⋮	⋮
01111100	124	＋124	＋124	＋124
01111101	125	＋125	＋125	＋125
01111110	126	＋126	＋126	＋126
01111111	127	＋127	＋127	＋127
10000000	128	－0	－128	－127
10000001	129	－1	－127	－126
10000010	130	－2	－126	－125
⋮	⋮	⋮	⋮	⋮
11111100	252	－124	－4	－3

续表

二进制数码表示	无符号二进制数	原　码	补　码	反　码
11111101	253	−125	−3	−2
11111110	254	−126	−2	−1
11111111	255	−127	−1	−0

下面举例说明已知一个数的补码，如何求出对应的真值。

对于一个用补码表示的8位二进制数，当其符号位为“0”(代表正数)时，其余7位即为此数的二进制数值。而当符号位为“1”(代表负数)时，其余7位并不是此数的二进制数值，而是要将此7位按位取反，再在最低位加1，才是此数的二进制数值。

例　$[X]_{补}=00101110$，符号为“0”，故其真值

$$X=+0101110=+46$$

例　$[X]_{补}=11010010$，其真值X不等于−82。因符号位为“1”，其余7位应按位求反，再在最低位加1，故真值

$$X=-(0101101+1)=-0101110=-46$$

为什么要引进补码的概念呢？在前面介绍二进制的算术运算，即加、减、乘、除时，对乘法采用加法和移位来完成，对除法则采用减法和移位来完成，故加、减、乘、除运算归结为加、减和移位三种操作来完成。但在计算机中为了节省设备，一般只设置加法器而无减法器，这就需要将减法运算转化为加法运算，从而使在计算机中的二进制四则运算最终变成加法和移位两种操作。引进补码运算就是用来解决将减法运算转化为加法运算的。

例　96−20。为了避免进行减法运算，可写成：

$$96-20=96+(-20)$$

利用公式：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}\qquad(模\ 2^n)$$

$$[X-Y]_{补}=[X+(-Y)]_{补}=[X]_{补}+[-Y]_{补}$$

故

$$[96-20]_{补}=[96+(-20)]_{补}=[96]_{补}+[-20]_{补}$$

$$[+96]_{补}=01100000$$

$$[-20]_{补}=11101100$$

这样一来

$$\begin{array}{r} 01100000 \\ -00010100 \\ \hline 01001100 \end{array} \xrightarrow{\text{转换成}} \begin{array}{r} 01100000 \\ +\ 11101100 \\ \hline \boxed{1}\ 01001100 \end{array}$$

↓

自然丢失

在字长为8位的机器中，从第8位向上的进位是自然丢失的，故本例中做减法运算的结果与用补码做加法运算的结果是相同的，都是十进制数76。

例　20−96=20+(−96)

$[+20]_{补}=00010100$

$[-96]_{补}=10100000$

则补码相加：