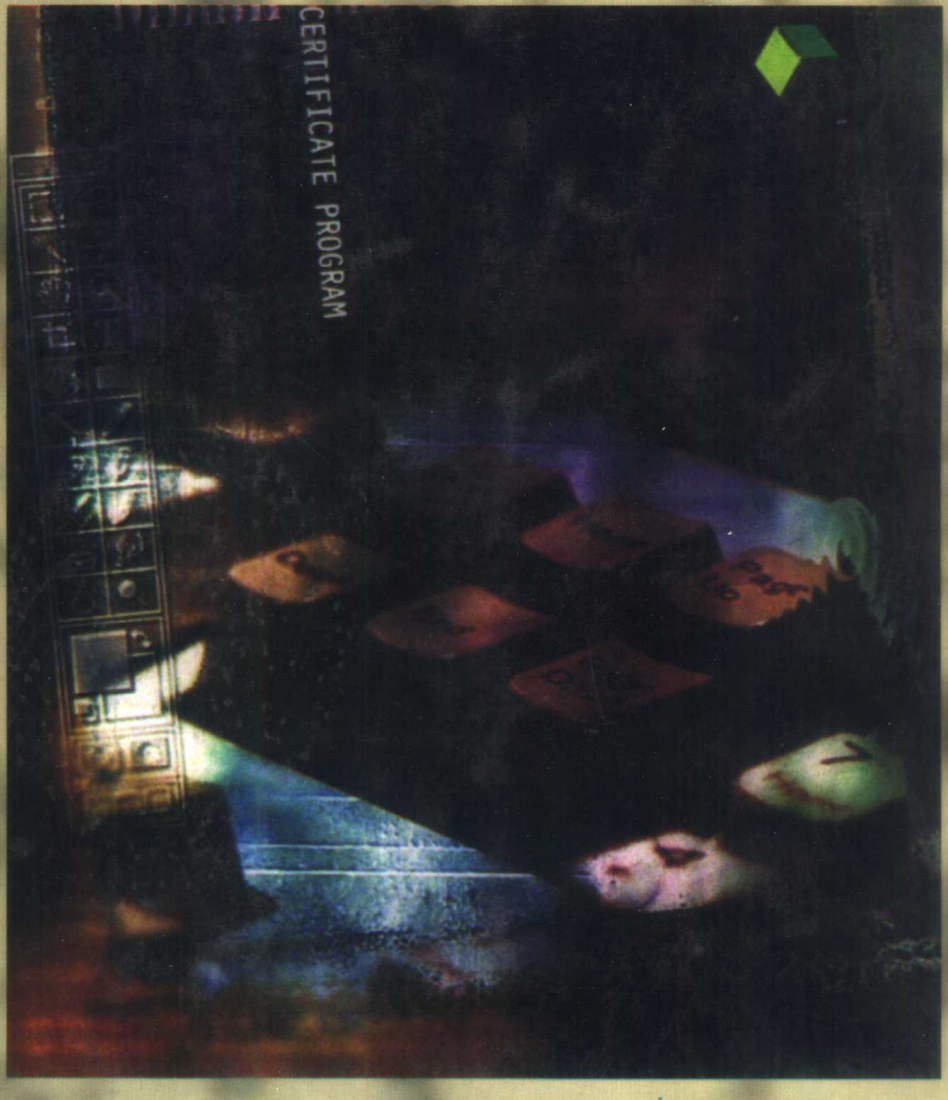
高校计算机教材丛书

单片微型计算机原理及应用

张毅坤 陈善久 裘雪红 编著



西安电子科技大学出版社

高校计算机教材丛书

单片微型计算机原理及应用

张毅坤　陈善久　裘雪红　编著

西安电子科技大学出版社

2000

内 容 简 介

本书较为系统、全面地叙述了MCS—51系列单片微型计算机的基本原理、结构、指令系统、汇编语言程序设计、应用系统扩展、输入/输出技术以及常用接口芯片的原理与应用，并简要介绍了单片微型计算机系统的设计、开发、调试的原则、步骤及方法，同时对国内几种常见单片微型计算机的类型与性能也作了介绍。

本书从教学与工程应用的角度出发，力求概念准确，由浅入深，内容充实，既有重点，又有扩展。为便于读者理解与掌握本书的内容，每章均配有大量的例子与习题。本书可作为大专院校有关专业师生及自学人员的教科书，也可供从事计算机应用方面的工程技术人员阅读、参考。

图书在版编目(CIP)数据

单片微型计算机原理及应用/张毅坤等编著. —西安：西安电子科技大学出版社，1998.8
ISBN 7-5606-0620-2

Ⅰ.单… Ⅱ.张… Ⅲ.单片微型计算机-基本知识 Ⅳ.TP368.1

中国版本图书馆CIP数据核字(98)第12148号

责任编辑 徐德源 陈其昌
出版发行 西安电子科技大学出版社
(西安市太白南路2号)
邮 编 710071
电 话 (029)8227828
经 销 新华书店
印 刷 西安兰翔印刷厂
版 次 1998年9月第1版
2000年3月第2次印刷
开 本 787毫米×1092毫米 1/16 印张 22
字 数 523千字
印 数 4 001～8 000册
定 价 22.00元
ISBN 7-5606-0620-2/TP·0312

前　言

随着大规模集成电路的发展，组成微型计算机的各功能部件：中央处理器、存储器、串/并行输入输出接口、定时器/计数器、中断控制器，以及许多特殊功能单元，如：A/D、D/A 转换器、高速输入输出部件、DMA、浮点运算等已集成在一块半导体晶体芯片上，构成一完整的微型计算机——单片机。由于它具有功能强、体积小、功耗低、价格便宜、工作可靠、使用方便等特点，因此特别适合于工业控制或与控制有关的数据处理系统，愈来愈广泛地应用于自动控制、智能化仪器、仪表、数据采集、军工产品以及家用电器等各个领域。由于其结构及应用特点，不同于通用的微型计算机，因此，学习单片机原理及应用是非常必要的。

本书是依据陕西省教育委员会于 1997 年修订的《陕西省高校非计算机专业学生计算机应用知识与应用能力等级考试三级(偏硬)考试大纲》的要求编写的，共分 8 章。既考虑了大纲要求的计算机基础知识，又以 MSC－51 系列单片机为基础，全面、系统地介绍了单片机的基本结构、指令系统、汇编语言程序设计、系统扩展、接口技术，以及常用芯片的应用，同时还简要介绍了单片机系统的设计与开发步骤及国内几种常见的单片机类型与性能。全书力求概念准确、由浅入深、既有重点、又有扩展，并配以大量例题、习题及思考题，便于课堂教学与自学，使读者通过本书的学习，为今后的工作打下一个坚实的基础。

本书第 1、2、3 章及附录 A、B 由张毅坤编写，第 4、5、6 章由陈善久编写，第 7、8 章由裘雪红编写。全书由张毅坤统稿。

本书从编写大纲到最后审定，都得到了西安电子科技大学李伯成教授的热情指导与帮助，并且提出了许多宝贵的意见，在此谨向李伯成教授表示衷心的感谢。

由于编者水平有限，加之时间仓促，缺点和错误在所难免，敬请读者不吝指正。

编　者

1997 年 10 月于西安

目　录

第 1 章 预备知识(数制与码制)

1.1 进位计数制及各计数制间的转换

数制是人们对事物数量计数的一种统计规律。在日常生活中最常用的是十进制，但在计算机中，由于其电气元件最易实现的是两种稳定状态：器件的“开”与“关”；电平的“高”与“低”。因此，采用二进制数的“0”和“1”可以很方便地表示机内的数据运算与存储。在编程时，为了方便阅读和书写，人们还经常用八进制数或十六进制来表示二进制数。虽然一个数可以用不同计数制形式表示它的大小，但该数的量值则是相等的。

1.1.1 进位计数制

当进位计数制采用位置表示法时，同一数字在不同的数位所代表的数值是不同的。每一种进位计数应包含两个基本的因素：

(1) 基数 R(Radix)：它代表计数制中所用到的数码个数。如：二进制计数中用到 0 和 1 两个数码；而八进制计数中用到 0～7 共八个数码。一般地说，基数为 R 的计数制(简称 R 进制)中，包含 0、1、…、$R-1$ 个数码，进位规律为“逢 R 进 1”。

(2) 位权 W(Weight)：进位计数制中，某个数位的值是由这一位的数码值乘以处在这一位的固定常数决定的，通常把这一固定常数称之为位权值，简称位权。各位的位权是以 R 为底的幂。如十进制数基数 $R=10$，则个位、十位、百位上的位权分别为 10^0，10^1，10^2。

一个 R 进制数 N，可以用以下两种形式表示：

(1) 并列表示法，或称位置计数法：

$$(N)_R=(K_{n-1}K_{n-2}\cdots K_1K_0K_{-1}K_{-2}\cdots K_{-m})_R$$

(2) 多项式表示法，或称以权展开式：

$$(N)_R=K_{n-1}R^{n-1}+K_{n-2}R^{n-2}+\cdots+K_1R^1+K_0R^0+K_{-1}R^{-1}+\cdots+K_{-m}R^{-m}$$

$$=\sum_{i=n-1}^{-m}K_iR^i$$

其中：m、n 为正整数，n 代表整数部分的位数；m 代表小数部分的位数；K_i 代表 R 进制中的任一个数码，$0\leqslant K_i\leqslant R-1$。

1. 二进制数

二进制数，$R=2$，K_i 取 0 或 1，进位规律为“逢 2 进 1”。任一个二进制数 N 可表示为：

$$(N)_2=K_{n-1}2^{n-1}+K_{n-2}2^{n-2}+\cdots+K_1 2^1+K_0 2^0+K_{-1}2^{-1}+\cdots+K_{-m}2^{-m} \quad (1-1)$$

例如：$(1001.101)_2=1\times2^3+0\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

2. 八进制数

八进制，$R=8$，K_i 可取 0～7 共 8 个数码中的任意 1 个，进位规律为“逢 8 进 1”。任意一个八进制数 N 可以表示为：

$$(N)_8=K_{n-1}8^{n-1}+K_{n-2}8^{n-2}+\cdots+K_1 8^1+K_0 8^0+K_{-1}8^{-1}+\cdots+K_{-m}8^{-m} \quad (1-2)$$

例如：$(246.12)_8=2\times8^2+4\times8^1+6\times8^0+1\times8^{-1}+2\times8^{-2}$

3. 十六进制数

十六进制数，$R=16$，K_i 可取 0～15 共 16 个数码中的任一个，但 10～15 分别用 A、B、C、D、E、F 表示，进位规律为“逢 16 进 1”。任意一个十六进制数 N 可表示为：

$$(N)_{16}=K_{n-1}16^{n-1}+K_{n-2}16^{n-2}+\cdots+K_1 16^1+K_0 16^0+K_{-1}16^{-1}+\cdots+K_{-m}16^{-m} \quad (1-3)$$

例如：$(2D07.A)_{16}=2\times16^3+13\times16^2+0\times16^1+7\times16^0+10\times16^{-1}$

注意：在实际应用中，如果一个十六进制数最高位数字为字母(A～F)，则字母前必须加一个数字 0，以便与变量名相区别。

表 1－1 给出了以上 3 种进制数与十进制数的对应关系。为避免混淆，除用 $(N)_R$ 的方法区分不同进制数外，还常用数字后加字母作为标注。其中字母 B(Binary) 表示二进制数；字母 Q(Octal 的缩写为字母 O，为区别数字 0 故写成 Q)表示八进制数；字母 D(Decimal)或不加字母表示十进制数；字母 H(Hexadecimal)表示十六进制数。

表 1－1 二、八、十、十六进制数码对应表

十进制	二进制	八进制	十六进制
0	0000 B	0 Q	0 H
1	0001 B	1 Q	1 H
2	0010 B	2 Q	2 H
3	0011 B	3 Q	3 H
4	0100 B	4 Q	4 H
5	0101 B	5 Q	5 H
6	0110 B	6 Q	6 H
7	0111 B	7 Q	7 H
8	1000 B	10 Q	8 H
9	1001 B	11 Q	9 H
10	1010 B	12 Q	0A H
11	1011 B	13 Q	0B H
12	1100 B	14 Q	0C H
13	1101 B	15 Q	0D H
14	1110 B	16 Q	0E H
15	1111 B	17 Q	0F H
16	10000 B	20 Q	10 H

1.1.2 各种进制数间的相互转换

1. 各种进制数转换成十进制数

各种进制数转换成十进制数的方法是：将各进制数先按权展成多项式，再利用十进制运算法则求和，即可得到该数对应的十进制数。

例 1：将数 1001.101B，246.12Q，2D07.AH 转换为十进制数。

$1001.101B = 1\times2^3+0\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

$=8+1+0.5+0.125=9.625$

$246.12Q = 2\times8^2+4\times8^1+6\times8^0+1\times8^{-1}+2\times8^{-2}$

$=128+32+6+0.125+0.03125=166.15625$

$2D07.AH = 2\times16^3+13\times16^2+0\times16^1+7\times16^0+10\times16^{-1}$

$=8192+3328+7+0.625=11527.625$

2. 十进制数转换为二、八、十六进制数

任一十进制数 N 转换成 q 进制数，先将整数部分与小数部分分为两部分，并分别进行转换，然后再用小数点将这两部分连接起来。

1）整数部分转换

整数部分转换步骤为：

第 1 步：用 q 去除 N 的整数部分，得到商和余数，记余数为 q 进制整数的最低位数码 K_0；

第 2 步：再用 q 去除得到的商，求出新的商和余数，余数又作为 q 进制整数的次低位数码 K_1；

第 3 步：再用 q 去除得到的新商，再求出相应的商和余数，余数作为 q 进制整数的下一位数码 K_i；

第 4 步：重复第 3 步，直至商为零，整数转换结束。此时，余数作为转换后 q 进制整数的最高位数码 K_{n-1}。

例 2：将 168 转换成二、八、十六进制数。

2 ⌊168

2 ⌊84　余数 0，$K_0=0$

2 ⌊42　余数 0，$K_1=0$

2 ⌊21　余数 0，$K_2=0$

2 ⌊10　余数 1，$K_3=1$

2 ⌊5　余数 0，$K_4=0$

2 ⌊2　余数 1，$K_5=1$

2 ⌊1　余数 0，$K_6=0$

0　余数 1，$K_7=1$

168＝10101000B

8 ⌊168

8 ⌊21　余数 0，$K_0=0$

8 ⌊2　余数 5，$K_1=5$

0　余数 2，$K_2=2$

168＝250Q

16 ⌊168

16 ⌊10　余数 8，$K_0=8$

0　余数 10，$K_1=A$

168＝A8H

2）小数部分转换

小数部分转换步骤为：

第 1 步：用 q 去乘 N 的纯小数部分，记下乘积的整数部分，作为 q 进制小数的第 1 个数码 K_{-1}；

第 2 步：再用 q 去乘上次积的纯小数部分，得到新乘积的整数部分，记为 q 进制小数的次位数码 K_{-i}；

第 3 步：重复第 2 步，直至乘积的小数部分为零，或者达到所需要的精度位数为止。此时，乘积的整数位作为 q 进制小数位的数码 K_{-m}。

例 3：将 0.686 转换成二、八、十六进制数(用小数点后 5 位表示)。

$0.686\times2=1.372$ $K_{-1}=1$　$0.686\times8=5.488$ $K_{-1}=5$　$0.686\times16=10.976$ $K_{-1}=A$

$0.372\times2=0.744$ $K_{-2}=0$　$0.488\times8=3.904$ $K_{-2}=3$　$0.976\times16=15.616$ $K_{-2}=F$

$0.744\times2=1.488$ $K_{-3}=1$　$0.904\times8=7.232$ $K_{-3}=7$　$0.616\times16=9.856$ $K_{-3}=9$

$0.488\times2=0.976$ $K_{-4}=0$　$0.232\times8=1.856$ $K_{-4}=1$　$0.856\times16=13.696$ $K_{-4}=D$

$0.976\times2=1.952$ $K_{-5}=1$　$0.856\times8=6.848$ $K_{-5}=6$　$0.696\times16=11.136$ $K_{-5}=B$

$0.686\approx0.10101B$　　$0.686\approx0.53716Q$　　$0.686\approx0.AF9DBH$

例 4：将 168.686 转换为二、八、十六进制数。

根据例 2、例 3 可得：

$168.686\approx10101000.10101B$

$168.686\approx250.53716Q$

$168.686\approx A8.AF9DBH$

从以上例子可以看出，二进制表示的数愈精确，所需的数位就愈多，这样，不利于书写和记忆，而且容易出错。另外，若用同样数位表示数，则八、十六进制数所表示数的精度较高。所以在汇编语言编程中常用八进制或十六进制数作为二进制数的缩码，来书写和记忆二进制数，便于人—机信息交换。在 MCS－51 系列单片机编程中，通常采用十六进制数。

3. 二进制数与八进制数之间的相互转换

由于 $2^3=8$，故可采用“合 3 为 1”的原则，即从小数点开始分别向左、右两边各以 3 位为 1 组进行二—八换算；若不足 3 位的以 0 补足，便可将二进制数转换为八进制数。

例 5：将 1111011.0101B 转换为八进制数。

解：根据“合 3 为 1”和不足 3 位以 0 补足的原则，将此二进制数书写为：

001　111　011　.　010　100

1　7　3　.　2　4

因此，其结果为 1111011.0101B＝173.24Q。

反之，采用“1 分为 3”的原则，每位八进制数用 3 位二进制数表示，便可将八进制数转换为二进制数。

例 6：将 1357.246Q 转换成二进制数。

解：根据“1 分为 3”的原则，可将该十进制数书写为：

1　3　5　7　.　2　4　6

001　011　101　111　.　010　100　110

其结果为 1357.246Q＝1011101111.01010011B。

4. 二进制数与十六进制数之间的相互转换

由于 $2^4=16$，故可采用“合 4 为 1”的原则，从小数点开始分别向左、右两边各以 4 位为 1 组进行二—十六换算；若不足 4 位以 0 补足，便可将二进制数转换为十六进制数。

例 7：将 1101000101011.001111B 转换成十六进制数。

解：根据“合 4 为 1”的原则，可将该二进制数书写为：

0001 1010 0010 1011 . 0011 1100

1 A 2 B . 3 C

其结果为 1101000101011.001111B=1A2B.3CH。

反之，采用“1 分为 4”的原则，每位十六进制数用 4 位二进制数表示，便可将十六进制数转换为二进制数。

例 8：将 4D5E.6FH 转换成二进制数。

解：根据“1 分为 4”的原则，可将该十六进制数书写为：

4 D 5 E . 6 F

0100 1101 0101 1110 . 0110 1111

其结果为 4D5E.6FH=100110101011110.01101111B。

1.2 二进制数的运算

1.2.1 二进制数的算术运算

二进制数不仅物理上容易实现，而且算术运算也比较简单，其加、减法遵循“逢 2 进 1”、“借 1 当 2”的原则。

以下通过 4 个例子说明二进制数的加、减、乘、除运算过程。

1. 二进制加法

1 位二进制数的加法规则为：

0+0=0 0+1=1 1+0=1 1+1=10（有进位）

例 1：求 11001010B+11101B。

解：

被加数		11001010
加数		11101
进位	+	00110000
和		11100111

则 11001010B+11101B=11100111B。

由此可见，两个二进制数相加时，每 1 位有 3 个数参与运算(本位被加数、加数、低位进位)，从而得到本位和以及向高位的进位。

2. 二进制减法

1 位二进制数减法规则为：

1−0=1 1−1=0 0−0=0 0−1=1（有借位）

例 2：求 10101010B－10101B。

解：

被减数		10101010
减数		10101
借位	－）	00101010
差		10010101

则 10101010B－10101B＝10010101B。

可见，二进制减法与加法类似，每 1 位有 3 个数参与运算（本位被减数、减数、低位借位），从而得到本位的差以及向高位的借位。

3. 二进制乘法

1 位二进制乘法规则为：

$0\times0=0$　　$0\times1=0$　　$1\times0=0$　　$1\times1=1$

例 3：求 110011B×1011B。

解：

被乘数		110011
乘数	×）	1011
		110011
		110011
		000000
	＋）	110011
积		1000110001

则 110011B×1011B＝1000110001B。

由运算过程可以看出，二进制数乘法与十进制数乘法相类似，可用乘数的每 1 位去乘被乘数，乘得的中间结果的最低有效位与相应的乘数位对齐，若乘数位为 1，则中间结果为被乘数；若乘数位为 0，则中间结果为 0，最后把所有中间结果同时相加即可得到乘积。显然，这种算法计算机实现时很不方便。对于没有乘法指令的微型计算机来说，常采用比较、相加、与部分积右移相结合的方法进行编程来实现乘法运算。

4. 二进制除法

二进制除法的运算过程类似于十进制除法的运算过程。

例 4：求 100100B÷101B。

解：

```
      000111
   ┌────────
101 │ 100100
      101
   ────────
      1000
       101
   ────────
        110
        101
   ────────
          1
```

则 100100B÷101B＝111B，余 1B。

二进制数除法是二进制数乘法的逆运算，在没有除法指令的微型计算机中，常采用比

较、相减、余数左移相结合的方法进行编程来实现除法运算。

由于 MCS－51 系列单片机指令系统中包含有加、减、乘、除指令，因此给用户编程带来了许多方便，同时也提高了机器的运算效率。

1.2.2 二进制数的逻辑运算

1. “与”运算(AND)

“与”运算又称逻辑乘，运算符为·或∧。“与”运算的规则如下：

$0\cdot 0=0 \quad 0\cdot 1=1\cdot 0=0 \quad 1\cdot 1=1$

例 5：若二进制数 $X=10101111B$，$Y=01011110B$，求 $X\cdot Y$。

```
  10101111
∧ 01011110
----------
  00001110
```

则 $X\cdot Y=00001110B$。

由此可见，若要保留 X 中的某位，就可在 Y 中某对应位用 1 与其进行“与”运算；若要 X 中某位为 0，只要使 Y 中对应位为 0 与其进行“与”运算即可。所以，在计算机控制或运算中，往往用“与”逻辑屏蔽某些位(即：使某些位为 0)或保留某些位不变。

2. “或”运算(OR)

“或”运算又称逻辑加，运算符为＋或∨。“或”运算的规则如下：

$0+0=0 \quad 0+1=1+0=1 \quad 1+1=1$

例 6：若二进制数 $X=10101111B$，$Y=01011110B$，求 $X+Y$。

```
  10101111
∨ 01011110
----------
  11111111
```

则 $X+Y=11111111B$。

由此可见，“或”运算与算术加的区别仅仅在于不产生进位。无论 X 中某位为 0 或 1，只要 Y 中某位为 1，就将该位置 1；Y 中某位为 0，保持该位结果不变。因此，在计算机中通常利用“或”逻辑给某些位置 1。

3. “非”运算(NOT)

“非”运算又称逻辑非，如变量 A 的“非”运算记作 $\overline{A}$。“非”运算的规则如下：

$\overline{1}=0 \quad \overline{0}=1$

例 7：若二进制数 $A=10101111B$，求 $\overline{A}$。

$$\overline{A}=\overline{10101111}B=01010000B$$

由此可见，逻辑“非”可使 A 中各位结果均发生反变化，即 0 变 1，1 变 0。

4. “异或”运算(XOR)

“异或”运算的运算符为∀或⊕，其运算规则如下：

$0\oplus 0=0 \quad 0\oplus 1=1\oplus 0=1 \quad 1\oplus 1=0$

例 8：若二进制数 $X=10101111B$，$Y=01011110B$，求 $X\oplus Y$。

$$\begin{array}{rl} & 10101111 \\ \forall & 01011110 \\ \hline & 11110001 \end{array}$$

则 $X \oplus Y = 11110001B$。

由此可见，X 与 Y 中两个数值相同位异或的结果为 0，否则为 1。利用 $X \oplus Y$ 就可实现 X 清 0；利用异或运算还可检验两个数是否相等，即：若 $X \oplus Y = 0$，则说明 $X = Y$。

以上二进制数的算术、逻辑运算规则奠定了计算机中数据运算与控制的基础。

1.3 带符号数的表示方法——原码、反码、补码

1.3.1 机器数与真值

在 1.2.1、1.2.2 节中讨论的二进制数运算均为无符号数运算，但实际的数值是带有符号的，既可能是正数，也可能是负数，前者符号用“+”号表示，后者符号用“−”号表示，运算的结果也可能是正数，也可能是负数。于是在计算机中就存在着如何表示正、负数的问题。

由于计算机只能识别 0 和 1，因此，在计算机中通常把一个二进制数的最高位作为符号位，以表示数值的正与负(若用 8 位表示一个数，则 D7 位为符号位；若用 16 位表示一个数，则 D15 位为符号位)，并用 0 表示“+”；用 1 表示“−”。

例如：$N_1 = +1011$，$N_2 = -1011$ 在计算机中用 8 位二进制数可分别表示为：

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	1	1
符号	数值部分						

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	1	1
符号	数值部分						

把原来的二进制数连同符号位一起作为一个新数，该数在计算机中的表示形式称之为机器数，而把原来二进制数的数值称为机器数的真值。计算机中表示机器数的方法有 3 种，即原码、反码和补码。

1.3.2 原码、补码与反码

1. 原码

正数的符号位用 0 表示，负数的符号位用 1 表示，数值部分用真值的绝对值来表示的二进制机器数称之为原码，用$[X]_{原}$表示。

(1) 正数的原码。若真值为正数 $X = +K_{n-2}K_{n-3}\cdots K_1K_0$(即 $n-1$ 位二进制正数)，

则
$$[X]_{原} = 0K_{n-2}K_{n-3}\cdots K_1K_0 \tag{1-4}$$

(2) 负数的原码。若真值为负数 $X = -K_{n-2}K_{n-3}\cdots K_1K_0$(即 $n-1$ 位二进制负数)，

则
$$\begin{aligned}[X]_{原} &= 1K_{n-2}K_{n-3}\cdots K_1K_0 \\ &= 2^{n-1} + K_{n-2}K_{n-3}\cdots K_1K_0\end{aligned}$$

$$=2^{n-1}-(-K_{n-2}K_{n-3}\cdots K_1K_0)=2^{n-1}-X \qquad (1-5)$$

例如：+115 和－115 在计算机中(设机器字长为 8 位)，其原码可分别表示为：

$$[+115]_{原}=01110011B;\ [-115]_{原}=11110011B$$

(3) 零的原码。若真值为零，则原码有两种表示法：

$$[+0]_{原}=000\cdots00$$

$$[-0]_{原}=100\cdots00$$

由此可得原码与真值的关系为

$$[X]_{原}=\begin{cases}X, & 0\leqslant X<2^n\\ 2^{n-1}-X, & -2^n<X\leqslant 0\end{cases} \qquad (1-6)$$

原码表示法的优点为直观、简单易懂、与真值的转换方便；其缺点为做加、减法不方便。如两个同号数相减或两个异号数相加(实际上是做减法)，运算时先要判断两数的绝对值大小，然后由绝对值大的数减去绝对值小的数，最后再给出适当的符号。其结果使计算机的运算时间增长，控制线路也变得复杂。因此，为了克服原码运算的缺点，简化计算机结构，提高运行速度，在计算机运算中引入了补码的概念，使正、负数的加法和减法运算简化为单一的加法运算。

2. *补码与反码*

1) 补码的概念

在日常生活中有许多"补"数的事例。如钟表，假设标准时间为 6 点整，而某钟表却指在 9 点，若要把表拨准，可以有两种拨法，一种是倒拨 3 小时，即 9－3＝6；另一种是顺拨 9 小时，即 9＋9＝6。尽管将表针倒拨或顺拨不同的时数，但却得到相同的结果，即 9－3 与 9＋9 是等价的。这是因为钟表采用 12 小时进位，超过 12 就从头算起，即：9＋9＝12＋6，该 12 称之为模(mod)。

模(mod)为一个系统的量程或此系统所能表示的最大数，它会自然丢掉，如：

$$9-3=9+9=12+6\rightarrow 6 \quad (\text{mod}12 \quad \text{自然丢掉})$$

通常称＋9 是－3 在模为 12 时的补数。于是，引入补数后使减法运算变为加法运算。

例如：　　$11-7=11+5\rightarrow 4 \quad (\text{mod } 12)$

＋5 是－7 在模为 12 时的补数，减 7 与加 5 的效果是一样的。

一般情况下，任一整数 X，在模为 K 时的补数可用下式表示：

$$[X]_{补数}=X+K \ (\text{mod } K)$$

$$=\begin{cases}X & 0\leqslant X<K\\ K-|X| & -K\leqslant X\leqslant 0\end{cases} \qquad (1-7)$$

在计算机运算与存储时，用二进制数所表示数据的位数，即字长总是有限的。设字长为 n，当两数相加求和时，如果 n 位的最高位产生进位，该位就会丢掉，这正是在模的意义下相加的概念，丢掉的进位即为模 2^n。所以，以 2^n 为模的补数，称之为 2 的补码，简称补码。

由补码的概念引伸，当用 n 位二进制数表示整数 X (1 位为符号位，$n-1$ 位为数值位)、模为 2^n 时，数 X 的补码可表示为：

$$[X]_{补}=\begin{cases}X & 0\leqslant X<2^{n-1}\\ 2^n+X & -2^{n-1}\leqslant X\leqslant 0\end{cases} \quad (\text{mod } 2^n) \qquad (1-8)$$