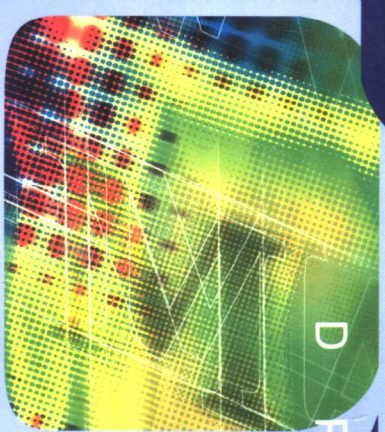
21世纪普通高等教育规划教材

单片机原理及其应用

陈立周　陈宇　编



DAN PIAN JI YUAN LI JI QI YING YONG

机械工业出版社
CHINA MACHINE PRESS

21世纪普通高等教育规划教材

单片机原理及其应用

陈立周　陈　宇　编
张国安　主审

机械工业出版社

本书是根据普通高等院校机电类的教学计划，以及对“单片机原理及其应用”课程的基本要求而编写的教材。内容包括单片机的基本原理、8051系列单片机的结构、MCS-51指令系统、编程技巧、存储器的扩展方法、中断、并口、串口、定时/计数器的结构与原理、功能器件的应用、C语言编程，以及单片机控制系统的硬件设计、软件调试等。由于本课程是实践性较强的课程，所以在内容上既注意讲述有关单片机的基础理论，也注意介绍在开发应用中会遇到的实际问题。

为适应近年来单片机技术的发展，本书强调功能器件的原理与应用、存储器与接口的串行扩展技术、对PC的串行通信、KEIL C51软件的使用以及有关操作调试方面的内容，同时有不少应用实例，以提高学生开发单片机应用系统的能力。

本书可作为普通高等院校本科机电类专业有关“单片机原理及应用”、“单片机控制系统”、“单片机接口”之类课程的教材，也可以供高职、高专、电大作为教学参考书或教材使用。此外，也可供从事单片机控制系统开发工作的工程技术人员学习参考。

图书在版编目（CIP）数据

单片机原理及其应用/陈立周，陈宇编 .—北京：机械工业出版社，2006.7

21世纪普通高等教育规划教材

ISBN 7-111-19240-0

Ⅰ.单 … Ⅱ.①陈 …②陈 … Ⅲ.单片微型计算机－高等学校－教材 Ⅳ.TP368.1

中国版本图书馆CIP数据核字（2006）第055841号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

责任编辑：贡克勤 版式设计：冉晓华 责任校对：董纪丽

封面设计：陈 沛 责任印制：杨 曦

北京机工印刷厂印刷

2006年8月第1版·第1次印刷

184mm×260mm·15.75印张·384千字

定价：22.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68326294

编辑热线电话（010）88379727

封面无防伪标均为盗版

前　言

为适应我国高等教育的发展需要，我们根据有关高等学校应用型本科的教学计划和培养目标，以及本科生就业岗位对从事单片机控制系统设计人员的要求，特编写此书。

“单片机原理及其应用”是一门实践性较强的课程，既要加强基础知识和基本理论，又要提高其实用性。所以本书一方面着重于基本原理的讲述，另一方面根据近年来单片机技术的发展、片内功能的加强、开发软件和开发设备的更新换代以及新的应用器件的不断出现，力求能反映这些领域的发展与变化，使得在原理与应用、深度与广度方面，能够更好适应本科层次的要求。

考虑到教学时数的限制，在内容上力求精简，使得教师能在规定学时内教完主要内容。所增加的内容，着重于实际应用，由浅入深，多用实例，以便于同学自学。可能有的专业学生没有学过C语言，所以在介绍C51编程的第九章中，增加了有关C语言的基础知识方面的内容，同时把C51编程单独列为一章，以便教师可以根据各专业的授课时数和学生的具体情况选择使用。

本书由福建工程学院陈立周编写了第1～7章，陈宇编写了第8～10章。全书由福建工程学院张国安教授担任主审。该教材初稿，经张国安教授、郑忠钦教授、余力副教授、雷伍讲师、祁建平高级讲师、蔡文培讲师、王亚林讲师进行审阅，他们在审稿中，对全书提出许多宝贵意见，在此向他们表示深切的感谢，并对福建工程学院计算机与信息科学系的教师和工作人员在审稿中提供的支持和帮助，表示深切的谢意。

书中难免还会存在问题和错误，敬请使用本书的老师和同学给予批评指正。编者电子信箱为chenlz@fjut．edu．cn

编　者

目　录

第一章　单片机的基础知识

第一节　不同进位计数制及其互换

电子计算机包括单片机都是一种处理数据信息的机器，因此在学习计算机之前有必要先了解一下有关数的知识。

在人们日常生活中，都习惯于使用十进制，但在数字电路和电子计算机内部，由于只能通过电位高低表示 1 和 0 两个数码，所以计算机中不用十进制而用二进制。这样，在使用计算机的时候，就存在常用的十进制与计算机中使用的二进制进行互换的问题，又由于用二进制表示一个数，所用的数码长，不但书写和阅读不方便，而且容易出错，所以书写时又常把二进制转换为十六进制。在开始学习微型电子计算机的时候，首先要熟练掌握这三种进位计数制的互换。

任何一种进位计数制，其位数多少决定于所表示的数的大小，位数越多，所表示的数值也越大，考虑到本书所讲的单片机是一种字长为 8 位的计算机，所以下面讨论二进制时都取 8 位为一个字节，超过 8 位则用两个或两个以上字节表示。

一、二进制与十六进制数的互换

一个 8 位二进制数，可以写成 2 位的十六进制数。所以两种进位制的数进行互换时，可以把每 4 位的二进制数划为一组，然后对每一组进行相应的变换，例如二进制转换为十六进制可写成

0110　1110
6　　E

把十六进制转换为二进制可写成

4　　B
0100　1011

在几种进制混用的场合为了不使二进制、十六进制或十进制数相混淆，规定在二进制数的后面加上符号 B 如 0110 0110B，在十六进制数的后面加上符号 H，如 6EH、4BH 等；也可以不在后面加 H，而在前面加 $ 或 0X，如 $ 5E、$ 4B、0X5E、0X4B 等。如果数的前后都没有符号，按习惯就认为是十进制数。

如果被转换的是一个小数，则分组时应以小数点为准，整数从小数点开始，从右向左每 4 位划为一组，小数部分则从小数点开始，从左向右也以 4 位为一组，如果最后一组不足 4 位，可以用零补齐。以数 1101100.11011B 为例，数的前后都要补 0，即

0110　1100　1101　1000
6　　C.　　D　　8

二、二进制与十进制数的互换

对于二进制整数，各位数的权可以用底数为 2 的 $n-1$ 次幂来确定，n 表示该数的位数，即第 1 位的权为 $2^0=1$，第 2 位的权为 $2^1=2$，……。若已知一个二进制数为 10101010B，对

应的十进制值应为 170，计算过程如下：

$$10101010B = 1\times2^7 + 0\times2^6 + 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 170$$

对于二进制小数，其小数点以后各位的权，可以用底数为 2 的负 n 次幂来确定，n 同样表示位数，即从小数点向右算起，第 1 位的权为 $2^{-1}=0.5$，第 2 位的权为 $2^{-2}=0.25$，……。例如求 11001100.00110011B 的十进制值，则

$$11001100.00110011B = 1\times2^7 + 1\times2^6 + 1\times2^3 + 1\times2^2 + 1\times2^{-3} + 1\times2^{-4} + 1\times2^{-7} + 1\times2^{-8}$$
$$= 204.19921875$$

反过来；要将十进制整数转换为二进制数，可以采用逐次除以 2 余数反序排列的方法，所谓反序排列；指第 1 次除以 2 的余数排在最低位。以十进制数 25 为例，逐次除以 2 列式如下

$$25 \div 2 = 12\cdots\cdots\text{余 } 1$$
$$12 \div 2 = 6 \cdots\cdots\text{余 } 0$$
$$6 \div 2 = 3 \cdots\cdots\text{余 } 0$$
$$3 \div 2 = 1 \cdots\cdots\text{余 } 1$$
$$1 \div 2 = 0 \cdots\cdots\text{余 } 1$$

由于 8 位微型计算机习惯将二进制数写成 8 位，可得

$$25 = 00011001B$$

如果二进制数不超过 8 位，即十进制数不超过 255，也可以不必列式，直接用口算转换。

要把十进制小数转换为二进制数，可以采用小数部分逐次乘 2，每次乘积若产生整数则将整数个位（即所谓溢出位）按正序排列，小数部分继续乘 2。以 33.6875 为例，其整数部分

$$33 = 00100001B$$

其小数部分

$$0.6875\times2 = 1.375\cdots\cdots\text{小数点左边整数为 } 1$$
$$0.375\times2 = 0.75 \quad\cdots\cdots\text{小数点左边整数为 } 0$$
$$0.75\times2 = 1.5 \quad\cdots\cdots\text{小数点左边整数为 } 1$$
$$0.5\times2 = 1 \quad\cdots\cdots\text{小数点左边整数为 } 1$$

可得出

$$33.6875 = 00100001.10110000B$$

三、十进制与十六进制数的互换

我们已经掌握了十进制与二进制的互换以及二进制与十六进制的互换，因此要把十进制数转换为十六进制数，可以先转换成二进制数，再改写成十六进制数。反之，十六进制数也可以先改成二进制数，再转换成十进制数。

十六进制数要直接转换为十进制数也可以按各位的权求出该数对应的十进制数值。整数部分的权等于底数为 16 的 $n-1$ 次幂，（n 为位数），即第 1 位的权为 $16^0=1$，第 2 位的权为 $16^1=16$，依次类推。例如十六进制的数为 8A71H，可求出

$$8A71H = 8\times16^3 + 10\times16^2 + 7\times16^1 + 1\times16^0 = 35441$$

对于十六进制的小数，其小数点以后各位的权，同样可以用底数为 16 的负 n 次幂来确定，n 表示位数，即从小数点向右算起，第 1 位的权为 $16^{-1}=0.0625$，第 2 位的权为 $16^{-2}=0.00390625$，……。例如某个小数为 0.4AC9H，转换为十进制值的计算过程为

$$0.4AC9H = 4 \times 16^{-1} + 10 \times 16^{-2} + 12 \times 16^{-3} + 9 \times 16^{-4} = 0.2921295$$

反过来，要把十进制转换为十六进制数，其方法与十进制数转换为二进制相似，即整数部分采用逐次除以 16 余数反序排列的方法。小数部分则采用逐次乘 16 溢出数正序排列的方法。例如将 13562 十进制数转换为十六进制：

$13562 \div 16 = 847$ ……余 10（记作 0AH）
$847 \div 16 = 52$ ……余 15（记作 0FH）
$52 \div 16 = 3$ ……余 4
$3 \div 16 = 0$ ……余 3

可得

$$13562 = 34FAH$$

在书写十六进制数时，若打头的数为 A ~ F，则应在 A ~ F 之前再加一个 0，以表示这是一个数而不是其他符号。

十进制小数转换为十六进制小数，同样采用小数部分逐次乘 16，每次乘积若产生整数，则将所得（即所谓溢出位）按正序排列，例如十进制小数 0.359375 转换为十六进制数

$0.359375 \times 16 = 5.75$ ……小数点左边整数为 5
$0.75 \times 16 = 12.0$ ……小数点左边整数为 C

可得

$$0.359375 = 0.5CH$$

第二节　带符号的二进制数

一、带符号二进制数与不带符号二进制数的区别

在数学运算中，表示一个数的正负，可以在数的前面冠以正号或负号。但计算机只能辨认 0 或 1 两个数码，不能辨认其他符号，因此在字长为 8 位的二进制数中，将最高位规定为符号位，最高位为 0，表示该数为正，最高位为 1，表示该数为负。例如数 01101111B 表示 +1101111B，而数 11101111B，则表示 -1101111B。这种将最高位定为符号位的二进制数，称为带符号的二进制数。对于 8 位字长的带符号二进制数来说，表示数值的仅有 7 位，所能表示的范围为 +127 ~ -127。考虑到 10000000 是 -128 的补码，所以补码表示范围可扩大为 +127 ~ -128。

应该注意，同样一个二进制数，它既可以是无符号数，也可以是带符号数。例如二进制数 11001100B，既可以是无符号数 204，也可以是带符号数 -76，到底它是 204 还是 -76，取决于事先约定，仅从数的本身无法判别它属于什么数。

例如下面要介绍的相对转移指令中的偏移量，约定必须使用带符号数，因此出现在偏移量中的二进制数，总是一个带符号数，在填写偏移量时，应使用带符号数，而程序中的地址值，约定为无符号数，对应使用无符号数。

总之，一个数是带符号的还是无符号的二进制数，从数的本身是无法区别的，只能根据它出现的场合，以及该场合约定使用什么数才能区分它们。

二、带符号数的表示方法

上面讲过，一个带符号数，它的最高位是符号位，其余表示数值。这种表示方式称为原

码，实际上，带符号的二进制数，除了原码表示法之外，还有反码与补码，下面分别加以介绍。

（一）原码（True form）

用原码表示一个带符号二进制数，其最高位为符号位，其余表示数值，例如

$$x = +1010101\text{B} \quad [x]_{t.f} = 01010101\text{B}$$

$$x = -1010101\text{B} \quad [x]_{t.f} = 11010101\text{B}$$

式中，x 为真值，$[x]_{t.f}$表示真值 x 的原码，对于数 0，其原码可以表示为

$$[+0]_{t.f} = 00000000\text{B}$$

$$[-0]_{t.f} = 10000000\text{B}$$

（二）反码（One’s complement）

反码也是带符号数的一种表示法，它同样规定最高位为符号位，其余则要看是正数还是负数。对于正数，其余各位表示数值，也就是正数的反码与原码相同。对于负数，其余各位应将 1 换成 0，将 0 换成 1，即所谓逐位取反，例如

$$x = +1010101\text{B} \quad [x]_{o.c} = 01010101\text{B}$$

$$x = -1010101\text{B} \quad [x]_{o.c} = 10101010\text{B}$$

式中，$[x]_{o.c}$表示真值 x 的反码。对于数 0 的反码，也有两种形式：

$$[+0]_{o.c} = 00000000$$

$$[-0]_{o.c} = 11111111$$

（三）补码（Two’s complement）

补码仍然把最高位定为符号位，对于正数，其余各位表示数值，可见正数的补码、反码与原码完全相同，对于负数，则其余各位逐位取反后再加 1，简称为取反加 1，例如：

$$x = +1010101\text{B} \quad [x]_{t.c} = 01010101\text{B}$$

$$x = -1010101\text{B} \quad [x]_{t.c} = 10101011\text{B}$$

式中，$[x]_{t.c}$为真值 x 的补码。

补码这个概念与某个具体计数器的最大容量有关，以常用的 8 位二进制数为例，扣除最高位作为符号位外，计数的最大容量为 7 位数。当计数器计至 128 时，最高位产生溢出，又由于计数器只有 7 位，溢出的数就被丢弃。这个被丢弃的值就是最大容量值，又称为模，用符号 Mod 表示。对于模等于 128 的 7 位计数器，0 与 128 的数码相同，或者说 0 与 128 等价。即

$$0 = 00000000\text{B}$$

$$128 = \boxed{1}\,0000000\text{B}$$

框中的 1 就是被丢弃的数。可见，若模为 M，则 a 与 $a + M$ 等价，例如模为 128 时，1 与 129 等价，2 与 130 等价。因为不计及溢出数，计数器内的示值是相同的，即

$$a = a + M \text{（Mod 为 } M\text{）}$$

再把这个概念推广到负数领域，例如 $a = (-2)$，代入上式有

$$(-2) = (-2) + 128 = 126 \text{（}M \text{ 为 } 128\text{）}$$

即模为 128 时，（－2）与 126 等价，或者说这两个数互为补数，同样，可推出（－3）

的补码为 125，（－4）的补码为 124。

为了进一步理解这个概念，可以用时钟做为例子，时钟的钟面最大示数为 12。时钟走到 12 点就等于 0 点，数 12 就被自动丢弃。可见时钟的钟面是一个模为 12 的计数器。时钟的时针向前拨 3 个字（$a=3$），跟向前拨 15 个字（$a+M=3+12=15$）都停在同一位置上，也就是说 +3 与 +15 等价。

如果将时钟向后拨定义为负数，那么时钟向后拨 3 个字（$a=(-3)$），跟时针向前拨 9 个字（$a=a+M=(-3)+12=9$）都停在相同位置上，也就是 -3 与 9 等价。因为对于模为 12 的钟面来讲，-3 可以用其补码 9 来表示。

要求得一个数的补码，可以利用公式 $a=a+M$（Mod = M），也可以采用正数补码等于原码，负数补码为取反加 1 的方法求得（注意：取反加 1 不包括符号位）。反过来，要从补码求原码，同样用取反加 1，表 1-1 是 8 位带符号二进制数的原码、反码、补码对照表，注意表中的

$$[+0]_{补}=00000000B$$

$$[-0]_{补}=00000000B$$

而 10000000B 不是（－0）的补码，而是（－128）的补码。

表 1-1　8 位带符号二进制数原码、反码、补码对照表

十进制数	二进制数	原码	反码	补码
+0	+0000000	00000000	00000000	00000000
+1	+0000001	00000001	00000001	00000001
+2	+0000010	00000010	00000010	00000010
⋮	⋮	⋮	⋮	⋮
+126	+1111110	01111110	01111110	01111110
+127	+1111111	01111111	01111111	01111111
-0	-0000000	10000000	11111111	00000000
-1	-0000001	10000001	11111110	11111111
-2	-0000010	10000010	11111101	11111110
⋮	⋮	⋮	⋮	⋮
-126	-1111110	11111110	10000001	10000010
-127	-1111111	11111111	10000000	10000001
-128	-10000000	无法表示	无法表示	10000000

三、带符号二进制数的运算

二进制数和十进制数的运算规则基本相同，所不同的仅仅是前者逢 2 进 1，后者逢 10 进 1，借位时，从高位借 1 到低位，前者当 2，后者当 10。但如果是带符号的二进制数，则情况比较复杂，因为带符号二进制数有三种表示方法，其中反码用得较少，下面主要介绍原码与补码运算中应注意的问题。

原码运算时，首先要把符号与数值分开。例如两数相加，先要判断两数的符号，如果同号，可以做加法，如果异号，实际要做减法，减后的差作为两数之和，和数的符号与绝对值较大的数的符号相同。两数相减也是一样，也要先判断符号，然后决定是相加还是相减，还

要根据两数的大小与符号决定两数之差的符号。

补码运算不存在符号与数值分开的问题，而且加法运算就一定是相加，减法运算就一定是相减，因此在计算机中对带符号的数进行加减时，最好使用补码。

设有 x、y 两个数，用补码表示如下：

$$x = 10011111B(-97\text{ 的补码})$$

$$y = 00001000B(+8\text{ 的补码})$$

若求 $x+y$ 之和，可不用考虑两数的符号，直接相加，得出的和为 $x+y=10100111B$（-89 的补码），可见直接相加结果必定是正确的。

若求 $x-y$ 之差，也可以直接相减，即

$$\begin{array}{rl} x = 10011111B & (-97\text{ 的补码}) \\ -y = 00001000B & (+8\text{ 的补码}) \\ \hline x-y = 10010111B & (-105\text{ 的补码}) \end{array}$$

若求 $y-x$ 之差，同样也用减法即

$$\begin{array}{rl} y = 00001000B & (+8\text{ 的补码}) \\ -x = 10011111B & (-97\text{ 的补码}) \\ \hline y-x = \boxed{1}01101001B & (+105\text{ 的补码}) \end{array}$$

也就是说做减法时，不论两数符号如何，其相减结果不论是数值还是符号都将是正确的。

在上述 $y-x$ 算式中，最高位发生进位，只是因为在字长为 8 位的计算机中，若运算结果并未超出补码的记数容量（$-128 \sim +127$），这时的进位被视为自然丢弃，计算机在运算中，这种自然丢弃并不影响结果的正确。

但要注意，如果得数超过 8 位补码所允许的表示范围（即超出 $+127 \sim -128$），则其进位称之为溢出。溢出与自然丢弃是两种不同的概念。判别属于哪一种，则要看第 7 位与第 8 位的进位情况，如果第 7 位和第 8 位同时产生进位，即所谓双进位，则这种进位属于允许的自然丢弃。如果只有第 7 位或者只有第 8 位产生进位，即只有单进位，则这种进位属于溢出，溢出表示其数值超出计算机字长所能表示的范围，运算结果必然是错误的，因而也是不允许的。

例 1-1 求下列两组 x、y 之和。

$$\begin{array}{lrl} x = +1100100B & [x]_{t.c} = 01100100B & [+100]_{t.c} \\ y = +1000011B & +[y]_{t.c} = 01000011B & +[+67]_{t.c} \\ \hline & [x+y]_{t.c} = 10100111B & [-89]_{t.c} \\ x = -1111000B & [x]_{t.c} = 10001000B & [-120]_{t.c} \\ y = -0011000B & +[y]_{t.c} = 11101000B & +[-24]_{t.c} \\ \hline & [x+y]_{t.c} = \boxed{1}01110000B & [+112]_{t.c} \end{array}$$

例中第 1 组，只有第 7 位产生进位，第 8 位没有进位。在第 2 组中只有第 8 位产生进位，第 7 位没有进位。都是单进位，都属于溢出，运算答案 -89 和 $+112$ 显然都是错误的。

“溢出”是带符号二进制数进行加减运算时因数值进位影响到符号位而产生的一种结果。无符号数的第 8 位不是符号，所以不使用溢出这个概念。

对于无符号数还应注意一点：当两个无符号数相减时，不允许用小的数去减大的数，因为数值小的数减去数值大的数，差一定是负数。无符号数的前提是没有符号，显然也不允许有负数，如果这样做，减的结果也必然是错误的。

综上所述，在原码运算中虽然可以把减法运算化为加法运算，例如将 $x-y$ 化为 $x+(-y)$，但因为原码运算要考虑两个数的符号，正数与负数相加实际还要做减法，所以这种转换没有什么实际意义。但如果这两个数是用补码表示，即 $(x)_{t.c}-(y)_{t.c}=(x)_{t.c}+(-y)_{t.c}$，由于补码运算时无须考虑两个数的符号，故加法运算就是相加，真正把减法运算转化为加法运算，这就是为什么计算机转移指令中的偏移量计算采用补码的原因。

第三节　BCD 码及文字符号代码

一个二进制数，可以表示一个无符号数，也可以表示一个带符号数，而且可以根据事先约定代表一个文字、一个符号或者代表一个特定的内容。当它代表文字或符号时称为代码，例如 00100100B，作为数，它表示的是无符号数 24H 或是带符号数 +24H。如果事先约定，它又能代表符号 $。所谓约定，可以按标准约定，也可以自行约定，下面介绍两种按标准约定的常见文字符号代码。

一、BCD 码（Binary coded decimal）

BCD 码是一种以二进制形式表示十进制数的编码，又称二-十进制码，它貌似二进制，实际是十进制数。

BCD 码以 4 位为一组，选用 0000B ~ 1001B 十种状态，代表 0 ~ 9 共 10 个数，舍弃其余的 6 种状态。当 BCD 码与十进制数进行互换时，可以按 4 位一组，逐组进行互换。

例 1-2　将 84.7 转换为 BCD 码。

8　　4　　.　　7　　0

1000　0100　.　0111　0000

例 1-3　将 BCD 码 10010100.01110010 转换为十进制数。

1001　0100　.　0111　0010

9　　4　　.　　7　　2

要将一个二进制数转换为 BCD 码，通常先将它转换为十进制数，然后再转换为 BCD 码。同样将 BCD 转换为二进制数，也是先转换成十进制数，再转换为二进制数。

例 1-4　将二进制数 11110011B 转换为 BCD 码。

$$11110011B = 1\times2^7+1\times2^6+1\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0=243$$

十进制数为　　2　　4　　3

BCD 码为　　0010　0100　0011

即二进制数 11110011B 的 BCD 码为 $(001001000011)_{BCD}$，或按习惯写成两个字节即等于 $(00000010\ 01000011)_{BCD}$。

例 1-5　将 BCD 码 $(10001001)_{BCD}$ 转换为二进制数。

1000　1001

十进制数为　8　9

二进制数为 = 01011001B

BCD 码的低 4 位向高 4 位进位要遵循逢 10 进 1 的原则，这与二进制显然不同，二进制的低 4 位要向高 4 位进位，必须遵循逢 16 进 1。因此两个 BCD 码相加。可以按二进制数的相加方法，但要以 4 位为一组，逐组相加，凡相加后的和大于 9 者，还应进行加 6 修正。

例 1-6　将 BCD 码 $(01011000)_{BCD}$ 与 $(01101001)_{BCD}$ 相加。

```
  0101  1000
+ 0110  1001
-----------
  1100  0001
进位1←┘
```

由于第一组的和为 17，大于 9，除进位 1 外，余数还应加 6 修正。第二组和为 12，也大于 9，也应加 6 修正，即上式的和应予以修正。

```
  1100  0001          检验   58
+ 0110  0110---加 6 修正    + 69
-----------------------------
0001  0010  0111---正确值    127
```

两个 BCD 码相减，凡低 4 位有向高 4 位借位者，都要进行减 6 修正，无借位的当然无需修正。

例 1-7　将 BCD 码 $(10000101)_{BCD}$ 与 $(00101000)_{BCD}$ 相减。

```
  1000  0101
- 0010  1000
-----------
  0101  1101
借位1←┘
```

```
  0101  1101          检验  85
-       0110              - 28
-----------------------------
  0101  0111              57
```

二、ASCII 码

ASCII 码是美国信息交换标准代码的简称，由 American Standard Code for Information Interchang 的第一个字母组成。

计算机只能辨认或存储 0 和 1 两个数码，也就是计算机内部只能使用二进制数，但在编制计算机程序或信息时，还会碰到许多文字符号，这就需要把文字符号改成一串二进制代码，即把文字符号数码化，现在国际通用的文字符号代码是 ASCII 码。

ASCII 码共 128 个，用 00000000 ~ 01111111 代表，其中英文大小写字母共 52 个，0 ~ 9 数字 10 个，常用书写符号（如!、%等等）和常用运算符号（如 + 、 - 、 < 、 > 等）32 个，另外有控制符号 34 个，共计 128 个。例如英文大写字母 A 的编码为 01000001，或写成十六进制 41H，数码 5 的编码为 00110101 或写成十六进制 35H。ASCII 码表见书后附录。

ASCII 码实际只占用一个字节的 7 位，余下的一个最高位可作为奇偶校验位。

第四节　单片机系统的组成

计算机由中央处理单元（简称 CPU）、存储器、输入输出接口及外围设备所组成。如果把中央处理单元集成在一小块芯片上，这种芯片就称为微处理器（Microprocesser），由微处理器和相应存储器、输入输出接口所组成的计算机则称为微型计算机。如果再进一步把微处理器、存储器及某些输入输出接口也一起集成在一个芯片上，这种芯片则称之为单片机。可见微型计算机和单片机其组成都是一样的，只是结构不同而已。

单片机也有的称之为单片微型计算机（Single Chip Microcomputer）、微控制器（Micro Controller Unit，简称 MCU）以及嵌入式系统等等。但“单片机”这种名称已经为国内大部分人所接受，所以本书统一称为“单片机”。

计算机除硬件外，还必须配上相应软件才能运行，加上软件后的计算机称为计算机系统。单片机也一样，单片机芯片虽然已经包含了微处理器、存储器及某些输入输出接口等硬件，但要运行也还需要软件和必需的外围设备，可见一个完整的单片机系统也是由硬件和软件所组成。

一、单片机系统的硬件

硬件是构成单片机系统的所有电子、机械和磁性的部件和设备，包括中央处理单元、存储器、外围设备与输入输出接口。它的组成如图 1-1 所示。

（一）中央处理单元（CPU）

中央处理单元是计算机的核心，用于控制和运算，它的内部由运算器、控制器和片内时钟振荡器等单元电路组成。运算器是 CPU 进行运算的部件，运算内容包括算术运算、逻辑运算及移位操作等。

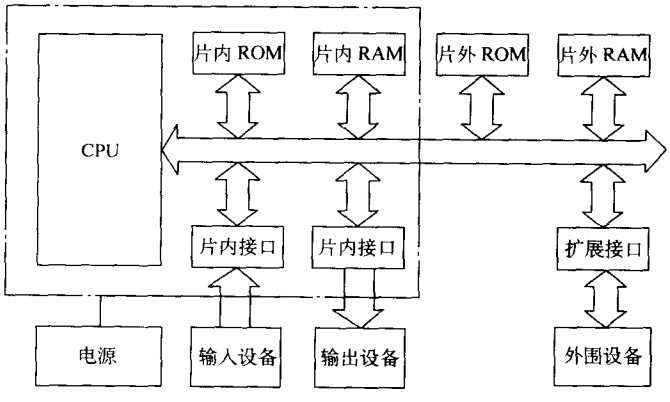


图 1-1　单片机系统的硬件组成

控制器是计算机的指挥中心，计算机是根据程序存储和程序控制的原理进行操作的。因此工作前需要输入程序，然后从程序中逐条取出指令，经指令译码分析后，发出执行命令。执行相应的操作并做好取出下一条指令的准备。控制器就是用来指挥这些操作，以便按一定的顺序，使运算工作快速而又有条不紊地进行。

（二）存储器

存储器是计算机的重要组成部分，它用来存储程序和数据。单片机系统一般使用半导体存储器，以便与 CPU 的工作速度相匹配。半导体存储器按存取方式和使用功能，又分为随机存取存储器（Random Access Memory，简称 RAM）和只读存储器（Read Only Memory，简称 ROM）两大类。通常 RAM 用于存储数据，ROM 用于存储固定的程序和数据。现在多数单片机的 ROM 和 RAM 都做在片内，不够用时才在片外扩展。

（三）接口与外围设备

外围设备简称外设，单片机系统使用的外设包括输入数据用的键盘、乒乓开关及各种传感器，以及输出或显示运算结果用的数码管、显示器、微型打印机等。

输入输出设备与主机的连接电路称为接口，简称 I/O 接口。接口是主机与外设之间的连接部件，设置它的目的是为了实现外设与总线的隔离。因为计算机要与众多的存储器和外围设备相联系，这种联系不能采用一一连线的办法，而是全部接在总线上。计算机总线包括数据总线 DB（Data Bus）、地址总线 AB（Address Bus）和控制总线（Control Bus），分别用于传送数据、地址和控制信号。既然所有外设都接在同一个总线上，某一时刻 CPU 又只能与一个存储单元或一个外围设备间传送信息，因此外设与总线就不能直接相通，而要通过一个 I/O 接口，这样，CPU 就可以在某一时刻通过地址总线选通一个接口，把该接口的外设与总线相连，其余外设接口处于阻断状态，以达到与总线隔离的目的，避免工作时的相互干扰。

隔离一般用三态门组成，三态门是一种可控的门电路，只有控制端使能时，输出端才受输入端控制。图 1-2a 是高电平控制的三态门，只有控制端为高电平时输出端才受输入端控制。与图 a 相反，图 1-2b 在控制端为低电平时输出端才受输入端控制，当输出端不受输入端控制时，其输出端呈高阻状态，相当于三态门将总线与外设隔离。由低电平控制的三态门真值表如表 1-2 所示。图 1-3 是由三态门构成的 I/O 接口。

表 1-2　三态门真值表

输入端电平	控制端电平	输出端电平
0	1	高阻态
1	1	高阻态
0	0	0
1	0	1

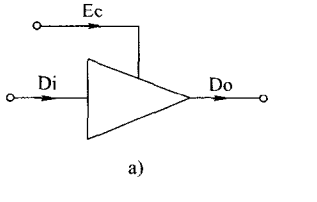


图 1-2　三态门

a）高电平控制三态门　b）低电平控制三态门

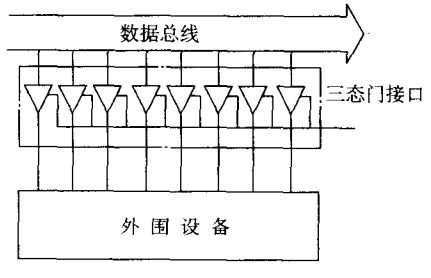


图 1-3　由三态门构成的 I/O 接口

接口除了隔离功能外，有的接口还需具有锁存或变换功能。锁存可以采用 D 触发器，因为 D 触发器一经触发，它的状态可以保持不变直到下一次触发为止。图 1-4 是用 D 触发器作为 I/O 接口的连接图。和三态门一样，它也是通过地址总线选通，被选通的锁存器，可以把 CPU 送来的数据存放在接口中，保持不变直至下一次选通。

二、单片机系统的软件

软件是各种程序及数据的总称，它以数字形式存储在硬件之中，要单片机完成某项任务，首先要把操作步骤按照单片机所能理解的语言编成程序，并通过编程器把程序连同原始数据存入（或称写入）单片机的 ROM，然后在程序控制下，自动进行各种操作和运算，需要时再通过输出设备将运算结果显示或打印出来。

软件以数码形式依附于硬件之中，它可以被替代或破坏。软件可以用不同的语言编写，

即机器语言、汇编语言和高级语言。

（一）机器语言

机器语言是指以二进制数码表示的指令和数据的一种语言。它可以直接为单片机的 CPU 所识别，所以称之为机器语言。由于不同单片机指令系统的代码都不相同，用一种指令系统编制的程序，放在另一种单片机或微处理器上就无法运行，所以机器语言是一种面向机器的语言。以 67（十六进制为 43H）与 58（十六进制为 3AH）相加为例，用 MCS-51 指令系统编写的机器语言程序为

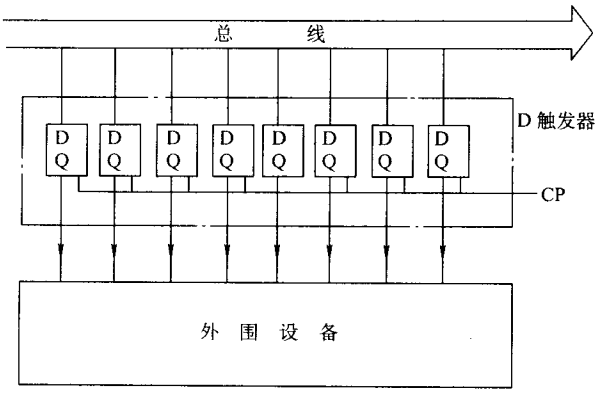


图 1-4　由 D 触发器构成的 I/O 接口

74　43　24　3A

这个程序本身只是一串机器码又称为机器指令，若要检查程序是否正确，除非对机器码十分熟悉，否则查起来将十分困难，但是我们又不得不用它，因为它是惟一能被机器所识别的一种语言，用其他语言编写的程序最后也要转换成机器指令，才能送入 CPU 执行运算，所以它又是一种必不可少的语言。

（二）汇编语言

汇编语言是以助记符代替机器指令的一种语言，每一条助记符都对应一条机器指令，目的是为了使写出来的指令容易阅读而且直观易记。每一条指令助记符写成一行，包括操作符和操作数，例如上面讲的将 67 和 58 相加的例子，写成汇编语言程序为

汇编语言	对应机器码
MOV　A，#43H	74　43
ADD　A，#3AH	24　3A

指令中的 MOV、ADD 称为操作符，其中 MOV 表示进行传送操作，ADD 表示进行加法操作，指令中“A，#43H”和“A，#3AH”是操作数，第一行表示将数 43H 传送到寄存器 A，第二行表示将 A 的内容与数 3AH 相加，并将结果存于 A。

用汇编语言写成的程序，主要优点是易记易读。但它只能称为汇编语言源程序，还要转换成用机器语言表示的目标程序才能使用。由于汇编语言的每条指令总是与相应的机器指令对应，所以汇编语言仍然是面向机器的语言。

应该注意，汇编程序与汇编语言程序是两个完全不同的概念，汇编语言程序是泛指用汇编语言编写出来的程序，而汇编程序是指一个专用程序，它专门用来把以汇编语言形式写出来的源程序转换为机器语言形式表示的目标程序。

（三）高级语言

高级语言是一种面向过程的语言，所谓面向过程，就是说这种语言只考虑解题的过程，只有在细节的地方才考虑使用的是什么机器，所使用的词和语句都尽量采用常用的单词、数学符号和表达式，用起来比较方便，如 BASIC、C、PASCAL 等语言。上述加法例子如用 BASIC 语言编写则为

A = 67 + 58