# 单片机原理及应用

## （机械制造与控制专业）

14

中等职业教育国家规划教材

全国中等职业教育教材审定委员会审定

# 单片机原理及应用

（机械制造与控制专业）

主　　编　刘力群
副 主 编　郑红峰
参　　编　符　刚　罗建彬　孙志平
责任主审　罗圣国
审　　稿　刘兆英　刘　江

机 械 工 业 出 版 社

本书共十章，以MCS-51单片机为例，介绍单片机基础知识，单片机结构、指令系统和程序设计方法、输入输出、中断与定时器/计数器、串行接口、存储器扩展、并行接口扩展、A/D和D/A转换、常用外部设备接口以及应用实例。每章末有习题，附录中配有实验。

本书介绍单片机基本内容，注重应用，通俗易懂，循序渐进，适合中职非电类专业使用，也可用于高职学校，还可供有关工程技术人员参考。

**图书在版编目（CIP）数据**

单片机原理及应用：机械制造与控制专业/刘力群主编．—北京：机械工业出版社，2002.1

中等职业教育国家规划教材

ISBN 7-111-09743-2

Ⅰ.单… Ⅱ.刘… Ⅲ.单片微型计算机—专业学校—教材

Ⅳ.TP368.1

中国版本图书馆CIP数据核字（2001）第097321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：赵爱宁 版式设计：霍永明 责任校对：张 媛

封面设计：姚 毅 责任印制：路 琳

北京机工印刷厂印刷·新华书店北京发行所发行

2002年2月第1版·第1次印刷

787mm×1092mm 1/16·12印张·290千字

0 001--5 000册

定价：14.00元

# 前　言

本书是根据教育部“面向21世纪中等职业教育课程改革和教材建设规划”，以及依据其精神制定的“单片机原理及应用”教学大纲编写的国家规划教材。

全书共分十章，主要介绍单片微型计算机基础知识、MCS-51单片机结构、指令系统、程序设计方法、输入输出、中断与定时器/计数器、系统扩展方法以及常用接口芯片等，并列举了应用实例。每章末有习题，并在附录中编写了与教材内容相结合的实验。

本书以MCS-51系列单片机为例，对微机原理及应用进行了介绍。本书内容侧重于应用，原理部分只做一般性介绍，重点放在程序设计方法、接口技术等方面，对片内硬件和接口芯片，采用结合应用举例的方法介绍其外部性能和使用方法。

本书在编写过程中注重体现计算机处理问题的方法、技巧、如标志概念、堆栈概念、寻址方式、数据排序和模块化结构等，使读者在学习微机原理的同时学会用计算机的思维方式处理问题。

按教学大纲要求，本书中带有“*”部分为选修内容。

本书第一章、第二章、第十章的第一节由刘力群编写，第三章、第六章由符刚编写，第四章、第五章、第十章的第二、三、四节由郑红峰编写，第七章、第八章由罗建彬编写，第九章和实验由孙志平编写。本书由刘力群任主编，郑红峰任副主编。

全书由责任主审罗圣国，审稿刘兆英、刘江审阅；主审陶砂审阅了全书。参加审稿的还有曹振军、徐衡和马进中，他们对本书提出了很多宝贵意见和建议。在编写前期的准备工作中，曹振军、符刚做了大量工作。在此，编者对为本书做出贡献的老师表示衷心感谢。

由于编者水平有限，书中难免存在缺点和错误，敬请读者批评指正。

编　者

2001年7月

# 目　录

# 第一章　单片微型计算机基础知识

本章主要介绍单片微型计算机的发展、应用和特点，数制、码制、单片微型计算机的组成及其基本工作原理。通过对本章的学习，要学会进行二进制数、十六进制数和十进制数之间的相互转换，明确补码在单片微型计算机中的使用，了解单片微型计算机的基本结构和工作过程。

## 第一节　概　　述

### 一、单片微型计算机的发展概况

自从1946年世界上第一台电子计算机问世以来，计算机由于自身的优势得到了高速发展。大规模及超大规模集成电路的出现使计算机的体积迅速缩小，促使微型计算机应运而生。微型计算机由微处理器（运算器和控制器）、存储器和输入输出接口三部分组成，这三部分之间通过总线相互连接（图1-1）。微型计算机从1971年出现到现在，已经经历了从4位机到32位机的四代发展。

单片微型计算机简称单片机，是将一台计算机的主要器件集成在一个芯片上。它是微型计算机的一个发展分支，以其体积小、功能强的突出特点而迅速普及，目前正向着高集成度、大容量、多功能、高速度和低能耗方向发展。

### 二、单片机的特点

微型计算机具有运算速度快、精度高、方便灵活、适应范围广和可靠性高等特点。作为其分支的单片机，由于特殊的硬件结构和指令系统，还具有以下突出特点：

1）体积小，价格低，应用广。由于计算机的主要器件集中在一个芯片上，而且适合大规模生产，因此在体积、重量、价格上具有优势，便于在中小设备、廉价设备上使用，从而使计算机深入到过去无法进入的领域，拓宽了微型计算机的应用范围。

2）通用性、灵活性强。在改变单片机的控制对象时，可以基本上不动硬件，只需改变程序。另外，可以很方便地对其进行扩展。

3）可靠性高、抗干扰能力强。单片机的高集成度，避免了功能器件之间的连线焊接、插接，缩短了系统内部的信息传送距离，从而提高了可靠性和减少了外部干扰的影响，能够适用于工作环境较恶劣的场合。

4）实时控制能力强。实时控制又称过程控制，是指及时的检测设备、采集数据信息，并按最佳方案对设备进行自动调节和控制。单片机具有很强的逻辑操作、位处理和判断转移功能，运行速度快，特别适合于工业系统实时控制。

5）应用开发周期短。单片机结构简单，硬件组合、软件编程都很方便，又容易进行模拟试验，因此付诸实际应用快。

### 三、单片机的应用

单片机体积小、功能强等特点，决定了它必然在工业控制、智能化仪器、通信系统、信

息处理和家用电器等领域得到广泛应用。

(1) 工业控制　如过程控制自动化、数控机械设备、工业机器人等。用单片机进行过程实时控制，可提高自动化水平，减轻劳动强度，提高控制准确性，从而降低成本，提高产品质量。

(2) 智能化仪器　如色谱仪、齿轮精度检验仪类的各种工业检验、测量仪器、医疗器械等。单片机可用于仪器的数据处理、存储、测试、校准和自动诊断故障等，提高了仪器的精度和可靠性，并扩大了仪器功能。

(3) 家用电器　如自动控温冰箱、全自动洗衣机、智能电饭锅、电话机、空调器、智能玩具等。单片机用于家用电器，使这些产品使用更简捷、方便，提高了生活质量。

(4) 导航系统　如飞机导航、导弹制导控制、航天飞机的地面控制等。单片机可以进行飞行的数据计算、分析、传送及控制，使飞行更精确。

单片机的应用范围很广，计算机的终端（打印机，键盘等）脱机工作、通信设备的计算机通信网等等，几乎所有需要控制、检测、数据处理的设施上都可以有单片机的应用。

## 第二节　数制和码制

单片机是用数字电路进行数据处理的，数字电路作数据处理是用二进制数的进位计数制，而人们习惯于十进制数。因此，学习计算机，首先要掌握计算机用到的计数制和计数制之间的转换，以及数字与字符的编码方法和数的符号表示方法等。

### 一、数制及其转换

习惯用的十进制数，是 10 个不同的数字 0，1，2，…，9 依据“逢十进一”的规则进行计数和运算的。

十进制数中，各位数字表示的值与数字本身和数字所在的位置有关，如数 56，十位上的 5 表示 5 个 10，即 50 ($5\times10^1$)；个位上的 6 表示 6 个 1，即 6 ($6\times10^0$)。可以看出，每位数被赋以一定的位值，称为该位的“权”。十进制中的个，十，百，千，万…各位的“权”依次为 1，10，100，1000，10000…，即 $10^0$，$10^1$，$10^2$，$10^3$，$10^4$…。每个数位上的数字所表示的值为：数字与它所在位的“权”的乘积。因此，任何一个十进制数，如 456.32 可以按权展开成多项式为

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

写成通式为

$$\begin{aligned}N &= D_nD_{n-1}D_{n-2}\cdots D_1D_0D_{-1}D_{-2}\cdots D_{-m}\\ &= D_n\times10^n+D_{n-1}\times10^{n-1}+D_{n-2}\times10^{n-2}+\cdots\\ &\quad +D_1\times10^1+D_0\times10^0+D_{-1}\times10^{-1}+D_{-2}\times10^{-2}+\cdots\\ &\quad +D_{-m}\times10^{-m}\\ &= \sum_{i=-m}^{n}(D_i\times10^i)\end{aligned}$$

式中，$D_i$ 是第 i 位的数字；10 为基数（基数是相邻两数位的权之比）；n、m 为正整数。

计算机中要用到二进制数和十六进制数，除在运算中采用“逢二进一”和“逢十六进一”的规则外，其他与十进制数基本相同。

（一）二进制数及其转换

1. 二进制数的特点及运算

（1）二进制数的表示　二进制数有两个不同的数字 0、1，基数是 2，依据“逢二进一”的规则。例如，二进制数

$$101.01$$

为区别于其他进制的数，加后缀 B 写成 101.01B 或 $(101.01)_2$。二进制数同样可以按权展开成多项式，例如

$$\begin{aligned}(101.01)_2 &= 1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}\\ &= (5.25)_{10}\end{aligned}$$

十进制数的后缀为 D（通常省略）或写成 $(5.25)_{10}$。

（2）二进制数的主要特点　计算机中的数据处理均采用二进制数，主要因为二进制数有以下特点：

1）二进制数中只有 0 和 1 两个数字，很容易用电子元器件来实现，如电压的高电平和低电平、继电器的通和断、指示灯的亮和灭等等，这些电器元件的两个稳定状态可以表示二进制数字的 0 和 1。用这种简单、明显的稳定状态表示的数字，抗干扰能力强，可靠性高。

2）二进制数的运算简单，加法和乘法法则仅有以下 8 条

$$0+0=0 \qquad 0+1=1 \qquad 1+0=1 \qquad 1+1=10$$
$$0\times0=0 \qquad 0\times1=0 \qquad 1\times0=0 \qquad 1\times1=1$$

明显比十进制数简单得多，从而减化了计算机中运算部件的结构。

3）二进制数的两个数字 0 和 1 与逻辑运算变量值一致，因此，可以直接用二进制数进行逻辑运算。

（3）二进制数的运算　下面举例说明二进制数的运算规则。

1）加法：

$$\begin{array}{r} 1101 \\ +1011 \\ \hline 11000 \end{array} \qquad \begin{array}{r} 10001101 \\ +01011010 \\ \hline 11100111 \end{array}$$

2）减法：

$$\begin{array}{r} 1101 \\ -1010 \\ \hline 0011 \end{array} \qquad \begin{array}{r} 11010110 \\ -01011011 \\ \hline 01111011 \end{array}$$

3）乘法：

$$\begin{array}{r} 1101 \\ \times101 \\ \hline 1101 \\ 0000 \\ 1101 \\ \hline 1000001 \end{array}$$

4）除法：

$$
\begin{array}{r}
1011 \\
110\overline{)1000010} \\
110 \\
\hline
1001 \\
110 \\
\hline
110 \\
110 \\
\hline
0
\end{array}
$$

由以上可见，加法是按逢二进一的规则进位，减法按借一当二的规则借位。

2. 二进制数和十进制数之间的转换

(1) 二进制数转换成十进制数

方法：乘权求和法。即对二进制数按权展开求和得到十进制数。

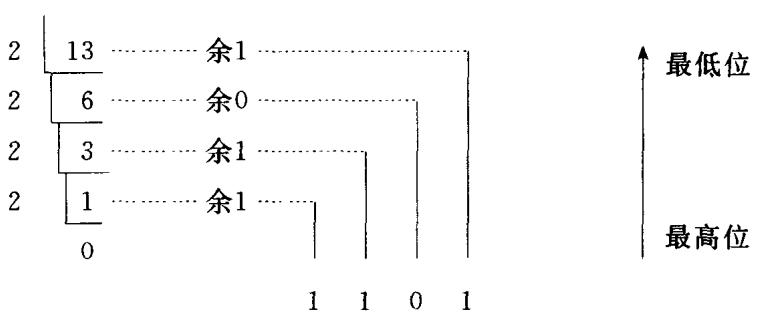
例：将 $(10101.01)_2$ 转换为十进制数

$$
\begin{aligned}
(10101.01)_2 &= 1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2} \\
&= (21.25)_{10}
\end{aligned}
$$

(2) 十进制整数转换成二进制整数

方法：除 2 取余逆排法。即将十进制整数不断用 2 除，直到商为 0 止，所得余数从后往前读，得到等值的二进制数。

例：将 $(13)_{10}$ 转换为二进制数




即 $(13)_{10}=(1101)_2$

(3) 十进制小数转换成二进制小数

方法：乘 2 取整顺排法。即将二进制小数不断乘 2，每次将整数取出从前向后排，直到积为 0 或达到要求精度为止。

例：将 $(0.375)_{10}$ 转换成二进制小数

$$
\begin{array}{rrr}
0.375 & 0.75 & 0.5 \\
\times\quad 2 & \times\quad 2 & \times\quad 2 \\
\hline
0.75 & 1.5 & 1.0 \\
\downarrow & \downarrow & \downarrow \\
0 & 1 & 1
\end{array}
$$

即 $(0.375)_{10}=(0.011)_2$

(二) 十六进制数及其转换

1. 十六进制数的表示

十六进制数有 16 个不同的数字和字母符号 0～9，A，B，C，D，E，F，分别表示十进制数的 0～15，基数为 16，依据“逢十六进一”的规则。例如

3AF.5E

十六进制数的后缀为 H，如上例为 3AF.5EH 或写成 $(3AF.5E)_{16}$。十六进制数也可以按权展开成多项式，如上例

$$(3AF.5E)_{16}=3\times16^2+10\times16^1+15\times16^0+5\times16^{-1}+14\times16^{-2}=(943.3671875)_{10}$$

2. 十六进制数的转换

(1) 十六进制数转换成二进制数

方法：一拉四法。即将每 1 位十六进制数用相应的 4 位二进制数替代。

例：将 $(3AF.5E)_{16}$转换成二进制数

| 3 | A | F | . | 5 | E |
|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | | ↓ | ↓ |
| 0011 | 1010 | 1111 | . | 0101 | 1110 |

即 $(3AF.5E)_{16}=(001110101111.01011110)_2$

(2) 二进制数转换成十六进制数

方法：四合一法。以小数点为界，分别向左右每 4 位一组由相应的 1 位十六进制数替代。

例：将 $(10111110110011.110001)_2$ 转换成十六进制数

| (00) 10 | 1111 | 1011 | 0011 | . | 1100 | 01 (00) |
|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | | ↓ | ↓ |
| 2 | F | B | 3 | . | C | 4 |

即 $(10111110110011.110001)_2=(2FB3.C4)_{16}$

转换时注意最后一组不足 4 位时，应用“0”补齐 4 位，否则易出错。

(3) 十六进制数与十进制数之间的转换　十六进制数与十进制数之间的转换可以采用二进制数与十进制数之间转换类似的方法，即十六进制数转换成十进制数采用“乘权求和法”；十进制数转换成十六进制数，整数部分采用“除 16 取余逆排法”；小数部分用“乘 16 取整顺排法”，小数部分通常有误差，一般转换到所要求的精度即可。

由于 16 进制数的基数为 16，比较大，做乘除法计算时繁琐，所以一般十六进制数与十进制数之间的转换是通过二进制数间接进行的，因为二进制数与十六进制数之间的转换十分方便，即先将十进制数转换成二进制数，再把二进制数转换成十六进制数；反之依然。

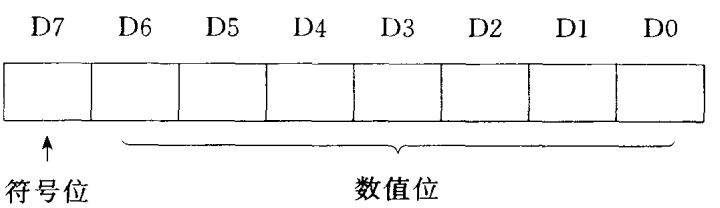
由于二进制数数值较大时，位数较长，不便于书写和记忆，而且容易出错，十六进制数位数少，读写方便，因此一般用十六进制数作为二进制数的缩写。十进制数、二进制数和十六进制数之间的对照关系见表 1-1。

**二、带符号数的表示法**

前面介绍的二进制数没有提及符号，称为无符号数。带有符号的数在数学运算中，表示一个数的正负，可以在数的前面冠以正负号。但是，计算机不能，因为计算机中所有信息都是用二进制数表示的，符号也不能例外。因此，在计算机中，通常用一个数的最高位作为符号位，用 0 表示正，1 表示负。例如，在 8 位计算机中，一个 8 位二进制数，它的最高位 D7 为符号位，后面的 7 位 D6～D0 为数值位，其格式如下：

表 1-1　十进制数、二进制数和十六进制数之间的对照关系

| 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 | 0 | 0 | 14 | 1110 | E |
| 1 | 1 | 1 | 15 | 1111 | F |
| 2 | 10 | 2 | 16 | 10000 | 10 |
| 3 | 11 | 3 | 17 | 10001 | 11 |
| 4 | 100 | 4 | ⋮ | ⋮ | ⋮ |
| 5 | 101 | 5 | 50 | 00110010 | 32 |
| 6 | 110 | 6 | 100 | 01100100 | 64 |
| 7 | 111 | 7 | 255 | 11111111 | FF |
| 8 | 1000 | 8 | ⋮ | ⋮ | ⋮ |
| 9 | 1001 | 9 | | | |
| 10 | 1010 | A | 0.5 | 0.1 | 0.8 |
| 11 | 1011 | B | 0.25 | 0.01 | 0.4 |
| 12 | 1100 | C | 0.125 | 0.001 | 0.2 |
| 13 | 1101 | D | 0.0625 | 0.0001 | 0.1 |




例如：① ＋107＝＋1101011B 为

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

符号位为 0。

② －107＝－1101011B 为

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

符号位为 1。

一个数在计算机中用二进制数表示的形式称为机器数，而反过来说，这个数称为机器数的真值。上例中，01101011B 和 11101011B 为机器数；＋1101011B 和－1101011B 为真值。

**三、原码、反码和补码**

在计算机中，机器数可以用三种方法表示：原码、反码和补码。

1. 原码

用原码表示带符号二进制数时，将最高位规定为符号位，其余位为原数值。例如

$$x=+1110011\text{B} \qquad [x]_t=01110011\text{B}$$

$$x=-1110011\text{B} \qquad [x]_t=11110011\text{B}$$

式中，$[x]_t$ 表示 $x$ 的原码，即机器数；$x$ 为机器数的真值。

数“0”的原码可以写成两种形式

$$[+0]_t=00000000\text{B}$$

$$[-0]_t=10000000\text{B}$$

原码表示方法简单，容易读，但运算时必须把符号与数值分开。如加法运算，先要判断两个数的符号，相同可直接相加；符号不同则不能相加，而要用绝对值大的数减去绝对值小的数，其差值为两数和，而符号取自绝对值大的。由此可见，原码计算很不方便，计算机在运算中一般不采用原码，从而引出了反码和补码。

2. 反码

反码也是带符号二进制数的一种表示方法。

正数的反码同原码，符号位为 0，其余位为原数值。

负数的反码，符号位为 1，其余各位为原数值按位取反，即 1 换成 0、0 换成 1。例如

$$x=+1110011\text{B} \qquad [x]_{\text{反}}=01110011\text{B}$$

$$x=-1110011\text{B} \qquad [x]_{\text{反}}=10001100\text{B}$$

式中，$[x]_{\text{反}}$ 表示 $x$ 的反码。

数“0”的反码也可以写成两种形式

$$[+0]_{\text{反}}=00000000\text{B}$$

$$[-0]_{\text{反}}=11111111\text{B}$$

3. 补码

补码是计算机中带符号数的常用表示方法。

正数的补码同原码，符号位为 0，其余位为原数值。

负数的补码，符号位为 1，其余各位为原数值按位取反后，再加 1，即反码加 1。例如

$$x=+1110011\text{B} \qquad [x]_{\text{补}}=01110011\text{B}$$

$$x=-1110011\text{B} \qquad [x]_{\text{补}}=10001101\text{B}$$

式中，$[x]_{\text{补}}$ 表示 $x$ 的补码。

数“0”的补码表示是唯一的，即

$$[0]_{\text{补}}=[+0]_{\text{补}}=[-0]_{\text{补}}=00000000\text{B}$$

带符号的 8 位二进制数对应的原码、反码、补码的表示及相应的十进制数对照见表 1-2。

**表 1-2　数的几种表示方法对照表**

| 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 |

综上所述可知：

1）三种编码的最高位为符号位，用0表示正、1表示负。

2）对于正数，三种编码的表示方法是一样的，即 $[x]_{原}=[x]_{反}=[x]_{补}$。

3）对于负数，三种编码的符号位均为1，但后面的数值部分不同。原码的数值部分与原数相同；反码的数值部分为原数按位取反；补码的数值部分为反码加1。

4）8位二进制数的原码、反码和补码所能表示的数值范围是不完全相同的，原码和反码为－127～＋127，0有两种表示；补码为－128～＋127，0只有一种表示。

另外还要说明以下几点：

1）采用补码后，符号位能与数值位一起参加运算。例如

已知：$x_1=+0011100B=+28$　　$x_2=-0010001B=-17$

求：$x_1+x_2$

解：$[x_1]_{补}=00011100B$　　$[x_2]_{补}=11101111B$

$[x_1]_{补}+[x_2]_{补}$ 为

$$\begin{array}{r} 00011100 \\ +11101111 \\ \hline \boxed{1}\ 00001011 \end{array}$$

自然丢失

即　　$[x_1]_{补}+[x_2]_{补}=00001011B=+11$

可见，符号在参加运算后结果是正确的。

2）采用补码运算后，结果也是补码，要得到真值还须转换。例如

已知：$x_1=-0011100B=-28$　　$x_2=+0010001B=+17$

求：$x_1+x_2$

解：$[x_1]_{补}=11100100B$　　$[x_2]_{补}=00010001B$

$[x_1]_{补}+[x_2]_{补}$ 为

$$\begin{array}{r} 11100100 \\ +00010001 \\ \hline 11110101 \end{array}$$

即结果的补码为11110101B，转换成原码为10001011B，真值为－0001011B＝－11。

3）负数补码求其原码时，符号位不变，数值位按位取反后加1，如上例。同样，由反码求其原码时，符号位不变，数值位按位取反。

4）计算机在做带符号的二进制数运算时，若结果超出了其表示范围，就会产生溢出。由表1-2可知，在8位二进制数中，最高位是符号位，后面7位表示数值，正数最大为1111111B，负数最大绝对值为10000000B，它们所对应的十进制数范围是＋127～－128。因此，当运算结果为大于127的正数或绝对值大于128的负数时，则会产生“溢出”而出错。例如

$$\begin{array}{rr} 01000001 & +65 \\ +)\ 01000100 & +)\ +68 \\ \hline 10000101 & -123 \end{array}$$

可见，结果本应为＋133，却成了－123。正是因为以上两数和大于＋127，使数值“溢

出”到符号位，结果出错。再如

$$\begin{array}{rr} 10000010 & -126 \\ +)\ 11111000 & +)\ \ -8 \\ \hline \boxed{1}\ 01111010 & +122 \end{array}$$

同样是因为两数的绝对值和超出了 128 而“溢出”出错。

然而，前面介绍的自然丢失不会出错，判断两者的方法为：

溢出：

① 符号位向前有进位，数值最高位向符号位无进位；

② 符号位向前无进位，数值最高位向符号位有进位。

自然丢失：

符号位向前有进位，数值最高位向符号位也有进位。

5）计算机不会区别带符号数和无符号数，只是按规定的法则进行计算。例如：两 8 位二进制数相加为

$$\begin{array}{r} 10000010 \\ +)\ 00011011 \\ \hline 10011101 \end{array}$$

作为无符号数：130+27=157，而作为带符号数则是（−126）+（+27）=−99。

此例说明，采用补码计算后，不管作为无符号数还是带符号数，运算结果都是正确的。因此，计算机用补码运算时，带符号数和无符号数是兼容的，只是操作者要知道自己做的是带符号数计算还是无符号数计算。

**四、BCD 码和字符的 ASCII 码**

（一）BCD 码

1. BCD（8421）码

计算机中使用二进制数代码操作，而人们习惯于十进制数，这样就使得我们要掌握并经常进行二进制数与十进制数之间的转换；数大时，既麻烦又费时。因此，需要一种比较适合十进制数的二进制编码方法，BCD 码就是此种性质的编码。用 4 位二进制代码表示 1 位十进制数，称为十进制数的二进制代码表示法，简称 BCD 码。

用 4 位二进制数可以写出 16 种组合，从 0000 至 1111。而要表示十进制数，用其中的 10 种组合即可，最常用的是按顺序取前 10 种组合。这种组合排列方便、对照直观、转换简单，称为 8421 码，也简称 BCD 码。BCD 码与十进制数、二进制数的对应关系见表 1-3。

**表 1-3 BCD 码与十进制数、二进制数的对应关系**

| 十进制数 | 二进制数 | 8421BCD 码 | 十进制数 | 二进制数 | 8421BCD 码 |
|---|---|---|---|---|---|
| 0 | 0000 | 0000 | 8 | 1000 | 1000 |
| 1 | 0001 | 0001 | 9 | 1001 | 1001 |
| 2 | 0010 | 0010 | 10 | 1010 | 0001 0000 |
| 3 | 0011 | 0011 | 11 | 1011 | 0001 0001 |
| 4 | 0100 | 0100 | 12 | 1100 | 0001 0010 |
| 5 | 0101 | 0101 | 13 | 1101 | 0001 0011 |
| 6 | 0110 | 0110 | 14 | 1110 | 0001 0100 |
| 7 | 0111 | 0111 | 15 | 1111 | 0001 0101 |

2. 十进制数与 BCD 码之间的转换

根据表 1-3 的对应关系进行转换。

(1) 例如将十进制数 8421 转换成 BCD 码：

$$\begin{array}{cccc} 8 & 4 & 2 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1000 & 0100 & 0010 & 0001 \end{array}$$

即 $8421=(1000010000100001)_{BCD}$

(2) 将 BCD 码 $(100101100011.0001)_{BCD}$转换成十进制数：

$$\begin{array}{cccccc} 1001 & 0110 & 0011 & . & 0001 \\ \downarrow & \downarrow & \downarrow & & \downarrow \\ 9 & 6 & 3 & . & 1 \end{array}$$

即 $(100101100011.0001)_{BCD}=963.1$

注意：BCD 码与真正的二进制数是不同的，它形式上为二进制数，实为十进制数。BCD 码是作为计算机用的二进制数与十进制数之间的一种过渡性编码使用的，以方便人机联系。

3. 二-十进制调整

计算机在进行 BCD 码运算时，是按照二进制数进位的，并不是“逢十进一”，因此运算结果可能会出错。

例如：

1) 求 BCD 码的 6+7

解：$6=(0110)_{BCD}$　　$7=(0111)_{BCD}$

$$\begin{array}{r} 0110 \\ +)\ 0111 \\ \hline 1101 \end{array}$$

BCD 码中没有 1101，因此结果出错。

2) 求 BCD 码 7+9

解：$7=(0111)_{BCD}$　　$9=(1001)_{BCD}$

$$\begin{array}{r} 0111 \\ +)\ 1001 \\ \hline 10000 \end{array}$$

$(10000)_{BCD}$为 10，而不是 16，因此结果出错。

以上计算结果出错的原因，都是没有按“逢十进一”的规则运算。因此，用 BCD 码运算，要想得到正确的结果，必须进行二-十进制调整，调整规则如下：

BCD 码 4 位（二进制数）一组，若相加结果大于 9，则对该 4 位进行“加 6 调整”。

如上两例中：

1) BCD 码的 6+7：加后得 1101，大于 9，要“加 6 调整”，则

$$\begin{array}{r} 1101 \\ +)\ 0110 \\ \hline 10011 \end{array}$$

得正确结果 $(10011)_{BCD}=13$。

2）BCD 码 7+9：加后得 10000，等于 10，也要“加 6 调整”，则

$$\begin{array}{r} 10000 \\ +)\ 0110 \\ \hline 10110 \end{array}$$

得正确结果 $(10110)_{BCD}=16$。

MCS-51 单片机对 BCD 码加法运算有专门的十进制调整指令(DA　A)，可以自动进行加 6 调整。

（二）ASC Ⅱ 码

计算机中除数字用二进制数表示外，字母和字符等都要用二进制数表示，即要用二进制数对字母和字符进行编码。现在最普遍使用的是 ASC Ⅱ 码，ASC Ⅱ 码是美国信息交换标准代码的英文缩写。它用 7 位二进制数对字符等进行编码，见附录 C。

## 第三节　单片机的基本组成及工作原理

### 一、单片机的基本组成

单片机由微处理器、存储器、输入输出接口以及连接这三部分的总线组成。其基本结构框图如图 1-1 所示。

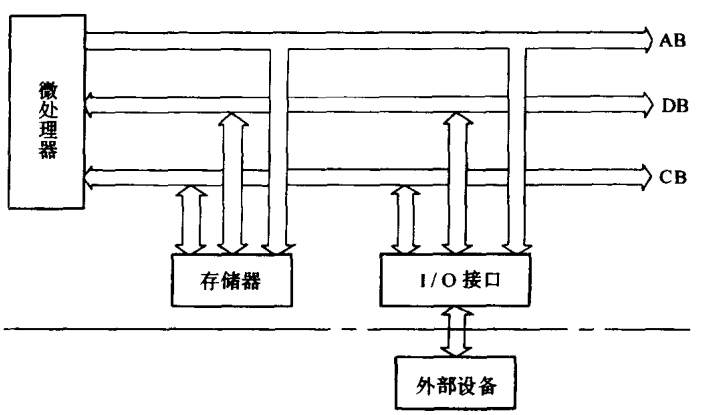



图 1-1　单片机的基本结构框图

（一）微处理器 CPU

微处理器 CPU 由运算器和控制器构成。运算器用来进行各种算术运算和逻辑运算。控制器是整个计算机的指挥中心，用来控制、协调计算机中各功能器件之间的工作。CPU 又称为中央处理单元，基本结构框图如图 1-2 所示。

1. 运算器

运算器包括算术逻辑单元 ALU、累加器 ACC 以及寄存器组等。算术逻辑单元 ALU 是进行各种算术（加法、减法等）运算和逻辑（“与”、“或”等）运算的部件。寄存器是临时存放数据的地方，每一个寄存器可以存放一个或两个 8 位二进制数。运算器在运算中需要进行提取待运算数据，送出中间结果和最终结果等操作。如果这些数据都要取自和送到外面的存储器，就要影响运算速度。因此，在运算器内部设置寄存器组，存放运算数据和中间结果，以使运算速度加快。累加器 ACC 是一个特殊功能寄存器。

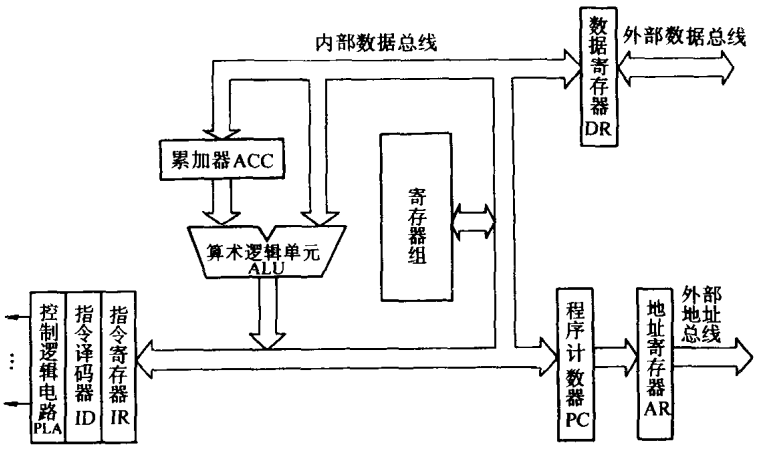



图 1-2　CPU 基本结构框图

在执行具体运算时，一般一个数据由累加器 ACC 提供，另一个数据来自某一个寄存器或存储器，经 ALU 运算后，结果一般放在累加器 ACC 中。

2. 控制器

控制器包括程序计数器 PC、指令寄存器 IR、指令译码器 ID 和控制逻辑电路 PLA 等。运算中，要从存有程序的存储器内依次取出指令来执行，程序计数器 PC 用来提供指令所在的位置（地址）。指令通过数据总线送入指令寄存器 IR 临时存放，指令译码器 ID 对指令寄存器 IR 的内容进行译码处理，确定要进行什么操作，译码后的命令通过控制逻辑电路 PLA 的多条线路发送到各个部件，指挥、协调整个计算机的工作。

图 1-2 中的数据寄存器 DR 和地址寄存器 AR，用来暂时存放数据和地址，并起着数据、地址在内部总线和外部总线之间传送中的缓冲和隔离作用。

（二）存储器

存储器是存放数据和程序（由一系列指令组成的运算步骤）的部件。它就像一个仓库，计算机开始运算之前，要把运算程序和原始数据送到存储器保存起来，运算中控制器从存储器得到指令，运算器也从中得到数据，运算结果一般还要送到存储器保存。

在存储器中，一般一位二进制数叫做一位，8 位为一个字节，每一个字节所占的存储空间为一个存储单元，如图 1-3 所示。8 位中最低位为 D0，最高位为 D7。

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

图 1-3　一个存储单元

一个存储单元存放一个字节的二进制数，图 1-3 中存有二进制数 01010101B。

存储器要存放大量的程序和数据，因此就有很多的存储单元。为区分不同的单元，就须对每个单元进行编号，这些编号称为存储单元的地址。存储器的结构如图 1-4 所示。存储单元的内容是 8 位二进制数编码的数据或指令，常用 2 位 16 进制数表示，地址编号是 16 位二进制数依次顺序排列，一般用 4 位 16 进制数来表示。

存储器的结构可用学生宿舍楼形象地加以比喻，学生宿舍楼相当于存储器，每一个房间相当于一个存储单元，房间的编号相当于存储单元的地址，房间内的 8 个床位相当于存储单