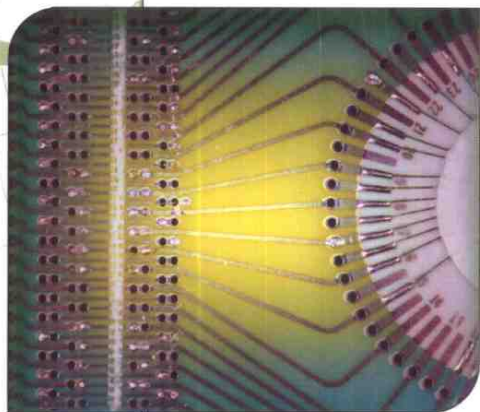




中等职业教育国家规划教材（电子与信息技术专业）
全国中等职业教育教材审定委员会审定

单片机原理与应用

专业主编 任德齐 主编 潘永雄
责任主审 吴锡龙 审稿 吕琴康 陈泉林



117

3 2 1 2

3 1

中等职业教育国家规划教材(电子与信息技术专业)

单片机原理与应用

专业主编 任德齐 主编 潘永雄
责任主审 吴锡龙 审稿 吕琴康 陈泉林



A0998680

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书以教育部下发的中等职业学校电子信息类专业《单片机原理与应用》课程教学大纲为依据,以 MCS-51 单片机原理及其应用为主线,系统地介绍了 MCS-51 系列单片机的内部结构、指令系统、资源及扩展、接口技术、单片机应用系统的硬件结构、开发过程及手段。在编写过程中,尽量避免过多地介绍程序设计方法和技巧,着重介绍系统硬件连接、调试技巧,注重典型性和代表性,以期达到举一反三的效果。内容力求新颖、全面、实用。

本书是单片机原理与应用的人门教材,可作为中等职业学校电子类专业“单片机原理与应用”课程的教材或教学参考书,亦可供从事单片机技术开发、应用的工程技术人员阅读。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

单片机原理与应用/潘永雄主编. —北京:电子工业出版社,2002.6

中等职业教育国家规划教材(电子与信息技术专业)

ISBN 7-5053-7197-5

I. 单… II. 潘… 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(2002)第 034545 号

责任编辑:徐晓光 特约编辑:吴浩源

印刷者:中国科学院印刷厂

出版发行:电子工业出版社 <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销:各地新华书店

开 本:787×1092 1/16 印张:15.75 字数:403,2 千字

版 次:2002 年 6 月第 1 版 2002 年 6 月第 1 次印刷

印 数:8 000 册 定价:19.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。
联系电话:(010)68279077

前 言

单片机技术作为计算机技术的一个重要分支,广泛应用于工业控制、智能化仪器仪表、家用电器,甚至电子玩具等各个领域,具有体积小、功能多、价格低廉、使用方便和系统设计灵活等优点。因此,越来越受到工程技术人员的重视,目前国内中等职业学校电子技术、电力技术、自动控制、计算机硬件等专业均开设“单片机原理与应用”课程。

本书以单片机在电子技术中的应用为主线,以需要掌握和使用单片机技术的中等职业学校有关专业学生作为主要的服务对象;从实用角度出发,力争用通俗易懂的语言,由浅入深,系统、详细地介绍 MCS-51 系列单片机系统的硬件结构、指令系统、程序设计方法、接口技术等方面的基本知识;结合典型应用实例,介绍单片机应用系统的开发过程和手段。在编写过程中,尽量避免过多地介绍程序设计方法和技巧,着重介绍系统的硬件组成及连接、系统调试及维修方法,注重典型性和代表性,以期达到举一反三的效果。内容力求新颖、全面、实用。

本书共分 7 章。考虑到前导课程中没有介绍计算机组成及工作原理等方面的教学内容,在第 1 章中先介绍计算机系统的基本结构、工作原理等基础知识,为后续内容的学习奠定基础;第 2 章全面、系统地介绍 MCS-51 单片机系统的内部结构、系统资源、存储器扩展方法,这是本书的重点;第 3 章简要介绍 MCS-51 指令系统、单片机应用程序结构和特点;第 4 章介绍 MCS-51 中断控制器、定时/计数器与串行接口电路的功能和使用方法;第 5 章介绍开关信号的输入/输出接口电路;第 6 章介绍模拟信号的输入/输出接口电路;第 7 章介绍单片机应用系统的设计规则、开发过程及手段、系统调试技巧和注意事项。

考虑到职业教育教材的特点,在附录中安排了较为详细的、可操作的实验内容。

本书由潘永雄主编。潘永雄编写了第 1~5 章、第 7 章部分内容和附录,第 6 章由李晓荃编写,第 7 章大部分内容由刘殊编写,李溪冰等也参与了本书部分章节的编写工作。

本书可以作为中等职业学校有关专业“单片机原理与应用”课程的教材或教学参考书,也可作为需要掌握和使用单片机技术的工程技术人员的参考资料。

由于编者水平有限,书中不当之处恳请读者批评、指正。

编 者
2002 年 1 月

中等职业教育国家规划教材出版说明

为了贯彻《中共中央国务院关于深化教育改革全面推进素质教育的决定》精神，落实《面向 21 世纪教育振兴行动计划》中提出的职业教育课程改革和教材建设规划，根据《中等职业教育国家规划教材申报、立项及管理意见》（教职成[2001]1 号）的精神，教育部组织力量对实现中等职业教育培养目标和保证基本教学规格起保障作用的德育课程、文化基础课程、专业技术基础课程和 80 个重点建设专业主干课程的教材进行了规划和编写，从 2001 年秋季开学起，国家规划教材将陆续提供给各类中等职业学校选用。

国家规划教材是根据教育部最新颁发的德育课程、文化基础课程、专业技术基础课程和 80 个重点建设专业主干课程的教学大纲（课程教学基本要求）编写，并经全国中等职业教育教材审定委员会审定。新教材全面贯彻素质教育思想，从社会发展对高素质劳动者和中初级专门人才需要的实际出发，注重对学生的创新精神和实践能力的培养。新教材在理论体系、组织结构和阐述方法等方面均作了一些新的尝试。新教材实行一纲多本，努力为教材选用提供比较和选择，满足不同学制、不同专业和不同办学条件的教学需要。

希望各地、各部门积极推广和选用国家规划教材，并在使用过程中，注意总结经验，及时提出修改意见和建议，使之不断完善和提高。

教育部职业教育与成人教育司

2001 年 10 月

目 录

第 1 章 基础知识	(1)
1.1 数制	(1)
1.1.1 二进制	(1)
1.1.2 二进制与十进制之间的转换	(2)
1.1.3 十六进制	(2)
1.1.4 二进制数与十六进制数之间的转换	(3)
1.1.5 二进制数和十六进制数的运算	(4)
1.2 码制	(5)
1.2.1 英文字符的表示方法——ASCII 码	(6)
1.2.2 BCD 码(二进制编码的十进制数)	(6)
1.2.3 计算机中带符号数的表示方法	(6)
1.3 计算机的基本认识	(8)
1.3.1 计算机的工作过程及其内部结构	(10)
1.3.2 指令及其指令系统	(14)
1.4 寻址方式	(20)
1.5 单片机及其发展概况	(23)
1.5.1 单片机及其特点	(24)
1.5.2 单片机技术现状及发展趋势	(24)
习题 1	(26)
第 2 章 MCS-51 单片机的结构	(28)
2.1 内部结构和引脚功能	(28)
2.1.1 内部结构	(28)
2.1.2 引脚功能	(29)
2.2 输入/输出(I/O)口	(32)
2.2.1 P1 口内部结构及使用	(32)
2.2.2 P0 口内部结构及使用	(33)
2.2.3 P2 口内部结构及使用	(34)
2.2.4 P3 口内部结构及使用	(34)
2.2.5 读锁存器和读引脚指令	(35)
2.3 存储器系统	(35)
2.3.1 程序存储器	(36)
2.3.2 片内数据存储器	(37)
2.3.3 外部数据存储器	(45)
2.4 MCS-51 外部存储器的连接	(45)
2.4.1 CPU 地址线与存储器地址线的连接	(46)

2.4.2 MCS-51 控制系统中程序存储器的连接	(48)
2.4.3 数据存储器的连接	(49)
* 2.5 操作时序	(53)
* 2.5.1 对外部程序存储器的读操作时序	(53)
* 2.5.2 外部数据存储器读写时序	(54)
2.6 复位及复位电路	(57)
2.6.1 CPU 内部复位电路	(57)
2.6.2 复位电路	(57)
* 2.7 掉电保护	(58)
* 2.7.1 CPU 内部数据存储器 and 特殊功能寄存器的保护	(59)
* 2.7.2 具有掉电数据保护的系统组成及连接	(60)
* 2.8 AT89C 系列单片机简介	(67)
* 2.8.1 AT89C51/52 系列的主要特点	(67)
* 2.8.2 AT89C1051/2051 的主要特点	(68)
* 2.8.3 AT89C 系列编程	(69)
习题 2	(69)
第 3 章 MCS-51 指令系统及汇编语言程序设计基础	(71)
3.1 MCS-51 指令系统	(71)
3.1.1 数据传送指令	(72)
3.1.2 算术运算指令	(78)
3.1.3 逻辑运算指令	(82)
3.1.4 位操作指令	(84)
3.1.5 控制及转移指令	(85)
3.2 汇编语言程序设计基础	(91)
3.2.1 汇编语言程序结构	(91)
3.2.2 汇编语言程序编辑与执行	(96)
* 3.2.3 对汇编语言程序的基本要求	(97)
习题 3	(99)
第 4 章 中断控制、定时/计数器与串行口	(101)
4.1 CPU 与外设通信方式概述	(101)
4.1.1 查询方式	(101)
4.1.2 中断通信方式	(101)
4.2 MCS-51 中断控制系统	(102)
4.2.1 中断源及标志	(103)
4.2.2 中断控制	(104)
4.2.3 中断响应过程及入口地址	(106)
4.2.4 中断初始化及中断服务程序结构	(107)
4.3 MCS-51 定时/计数器	(108)
4.3.1 定时/计数功能概述	(108)
4.3.2 定时/计数器 T0、T1 结构及控制	(108)

*4.3.3 定时/计数器 T2 结构及控制	(113)
4.3.4 定时/计数器应用	(116)
4.4 串行通信系统	(123)
4.4.1 串行通信概念	(123)
4.4.2 MCS-51 串行通信口控制及初始化	(125)
4.4.3 串行口工作方式及应用	(129)
4.4.4 RS232C 串行接口标准及应用	(134)
习题 4	(137)
第 5 章 数字信号输入/输出接口电路	(138)
5.1 I/O 资源及扩展	(138)
5.1.1 通过锁存器、触发器扩展 I/O 口	(139)
5.1.2 用 8255 可编程 I/O 芯片扩展 MCS-51 的 I/O 口	(141)
5.1.3 利用 8155/8156 可编程 I/O 芯片扩展 MCS-51 的 I/O 口	(147)
5.2 开关信号输入/输出方式	(153)
5.3 简单显示驱动电路	(154)
5.3.1 发光二极管	(154)
5.3.2 驱动电路	(155)
5.4 LED 数码管及其显示驱动电路	(156)
5.4.1 LED 数码管	(156)
5.4.2 LED 数码显示器接口电路	(157)
5.4.3 LED 点阵显示器及其接口电路	(164)
5.5 键盘电路	(167)
5.5.1 按键结构与按键电压波形	(168)
5.5.2 键盘电路形式	(169)
5.5.3 键盘按键编码	(170)
5.5.4 键盘监控方式	(171)
5.6 单片机与继电器接口电路	(176)
习题 5	(177)
第 6 章 A/D、D/A 转换电路	(179)
6.1 转换器基本原理	(179)
6.1.1 数模转换器 DAC	(179)
6.1.2 DAC 转换器的主要技术指标	(182)
6.1.3 模数转换器 ADC	(182)
6.1.4 A/D 转换器的性能指标	(186)
6.2 常用的 A/D 转换器与 MCS-51 单片机的接口	(186)
6.2.1 ADC0809	(186)
6.2.2 ADC0809 与 MCS-51 单片机的接口	(188)
* 6.2.3 MCS-51 单片机与 ADC574A 的接口	(190)
6.3 常用 D/A 转换器与 MCS-51 单片机的接口	(192)
6.3.1 D/A 转换器 DAC0832	(192)

6.3.2 DAC0832 与 MCS-51 的接口	(194)
* 6.3.3 MCS-51 与 12 位 D/A 转换器的连接	(196)
习题 6	(197)
第 7 章 单片机应用系统开发	(199)
7.1 单片机应用系统开发过程	(199)
7.1.1 总体设计	(200)
7.1.2 硬件设计	(202)
7.1.3 软件设计	(204)
7.1.4 资源分配	(204)
7.1.5 系统性能测试及评价	(205)
7.2 单片机开发工具及选择	(206)
7.3 系统可靠性设计	(208)
7.3.1 硬件可靠性设计	(209)
7.3.2 软件可靠性设计	(210)
7.3.3 系统自诊断技术	(211)
7.3.4 系统抗干扰性能	(212)
7.4 单片机应用系统的调试和维护	(215)
7.4.1 系统调试	(215)
7.4.2 系统维护与维修	(217)
习题 7	(219)
附录 实验	(220)
实验一 认识 8051 单片机及其开发系统(仿真机)	(220)
实验二 MCS-51 指令系统	(225)
实验三 MCS-51 指令系统综合练习	(227)
实验四 中断	(228)
实验五 显示器与定时中断	(234)
实验六 A/D 转换	(237)
实验七 D/A 转换	(238)
实验八 联机调试	(240)
参考文献	(241)

第1章 基础知识

1.1 数制

十进制是人们最熟悉也是最习惯的计数方式,但十进制要用0~9十个数码表示,在计算机中显得很不方便。计算机的电路基础是数字电路,而数字电路中的晶体管只有两个稳定状态,要么截止,要么饱和。截止时,集电极输出高电平,定义为“1”态;饱和时,集电极输出低电平,定义为“0”态。又如脉冲宽度内为“1”,脉冲间歇期为“0”等。这与二进制非常类似。一位二进制也有两个状态(0和1),这就是计算机中用二进制计数表示、运算、存储的原因,即二进制数是计算机系统能够认识、处理的惟一数制。二进制数不够直观,位数较长,不便记忆。向计算机输入数据时,往往直接输入人们习惯的十进制数,计算机将其转化为二进制数后再处理;计算机的处理结果也要转换为人们习惯的十进制数。因此,在计算机中,需要进行各种数制之间的转换。

1.1.1 二进制

二进制的优点是:只有两个数码,即0和1;逢二进1。

1位二进制数只能表示0和1两个状态;为了表示更多的状态,可用两位或两位以上的二进制数表示。二进制数的位数与它表示的状态数之间的关系如下:

1位二进制数,共有 2^1 (即2)个状态,分别编码为0、1。

2位二进制数,共有 2^2 (即4)个状态,分别编码为00、01、10、11。

4位二进制数,共有 2^4 (即16)个状态,分别编码为0000、0001、0010、0011、0100、0101、0110、0111、1000、1001、1010、1011、1100、1101、1110、1111。

8位二进制数,共有 2^8 (即256)个状态,分别编码为00000000、00000001、00000010、00000011、……、11111100、11111101、11111110、11111111。

在计算机系统中所用的寄存器、存储器的本质就是一组触发器。一个触发器,如RS、D型触发器等均有两个稳定的状态,即0态和1态,显然一个触发器可以存储一位二进制数。为了提高数据处理速度,在计算机中往往需要并行处理多位二进制数。习惯上,存储器中一个存储单元通常由8个触发器组成,即一个存储单元可以存放一个8位二进制数。8位二进制数称为一个字节(Byte),有256种状态,或者说可以表示256个符号。因此,存储器(包括内存储器和外存储器)容量单位常用字节(或千字节)表示,如某存储器的容量为640KB,即该存储器有 640×1024 个存储单元,每个存储单元的大小为一个字节。

10位二进制数,共有 2^{10} (即1024,在计算机中,“1024”习惯上称为1K)个状态,编码从0000000000~1111111111。

16位二进制数,共有 2^{16} (65536,即64K)个状态,编码从0000000000000000~1111111111111111。

有些微处理器,如大多数8位的微处理器就有16根地址线。由于每根地址线只有两种可

能状态,16根地址线相当于16位二进制数,最多可以寻址64K个存储单元。存储单元的大小一般是一个字节,对于具有16根地址线的微处理器来说,最多可以寻址64KB的存储空间,或者说寻址能力为64KB。

同理,对于20位二进制数,将有 2^{20} (1048576,即1024K,在计算机中习惯上称为1M)个状态,编码从00000000000000000000~11111111111111111111。

某些微处理器,如Intel 8088 CPU,就有20根地址线,因此该微处理器最多可以寻址1MB的存储空间。

为了不致引起混乱,二进制数用后缀字母B作标记,如2进制数1110记为1110B。

1.1.2 二进制与十进制之间的转换

众所周知,对于 n 位10进制数,可以表示为:

$$A_{n-1} \times 10^{n-1} + A_{n-2} \times 10^{n-2} + \cdots + A_2 \times 10^2 + A_1 \times 10 + A_0 + B_1 \times 10^{-1} + B_2 \times 10^{-2} + \cdots$$

例如,9876.54可以表示为 $9 \times 10^3 + 8 \times 10^2 + 7 \times 10 + 6 + 5 \times 10^{-1} + 4 \times 10^{-2}$ 。

同理, n 位2进制数,也可以表示为:

$$A_{n-1} \times 2^{n-1} + A_{n-2} \times 2^{n-2} + \cdots + A_2 \times 2^2 + A_1 \times 2 + A_0 + B_1 \times 2^{-1} + B_2 \times 2^{-2} + \cdots$$

例如,1101.01可以表示为 $1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 + 0 \times 2^{-1} + 1 \times 2^{-2}$,即十进制数13.25。

可见,将二进制数转换为十进制数不难,只要按上式展开即可求出对应的十进制数。

十进制数转换为二进制数时,可以按如下规律进行:

整数部分除以2所得的余数,就是对应二进制数的个位;其商再除以2所得的余数就是对应二进制数的“十”位,依此类推,即可获得对应二进制数的整数部分。

小数部分乘以2所得的整数就是对应二进制数小数部分的“十分”位,乘积中的小数部分再乘以2得到的整数就是对应二进制数小数部分的“百分”位。依此类推,即可求出所有的小数位。为了满足习惯,这里借用十进制中的“个”位、“十”位、“十分”位和“百分”位等称谓。

【例1.1】13.75的整数部分是13,小数部分是0.75,转换为二进制数的过程如下:

$$\begin{array}{r|l} 2 & 13 \quad 1 \\ \hline & 6 \quad 0 \\ 2 & 3 \quad 1 \\ \hline & 1 \end{array}$$

13相当于二进制数1101;而 $0.75 \times 2 = 1.5$,因此“十分”位为1; $0.5 \times 2 = 1.0$,因此“百分”位为1,即 $13.75 = 1101.11B$

1.1.3 十六进制

由于二进制数由一长串的0、1组成,位数太长,不便书写和记忆;另一方面,二进制数和十六进制数之间的换算非常方便、直观。为此,书写时常用十六进制数表示二进制数,但必须注意引入十六进制数的目的仅仅是为了便于书写和记忆,被计算机认识、处理的数据依然是二进制数,或者说二进制数是计算机系统中惟一存在的数制。

十六进制的特点是逢“十六进一”,具有16个数码,分别用0,1,2,……,9和A,B,C,D,E,F

表示。

一位十六进制数可以表示 16 种状态,编码从 0 ~ F;两位十六进制数可以表示 16^2 (即 256) 种状态,编码从 00 ~ FF;四位十六进制数可以表示 16^4 (65536, 即 64K) 种状态,编码从 0000 ~ FFFF;而八位十六进制数可以表示 16^8 (即 4096M) 种状态,编码从 00000000 ~ FFFFFFFF。

可见十六进制数位短,便于书写和记忆。

为了不致引起误解,十六进制数要加后缀字母 H,如 16 进制数“3F”记为“3FH”;而对于以字母开头的十六进制数,必须带有前缀 0(零),以示区别于一般字符串,如十六进制数 FE 记为“0FEH”。

与十进制数类似,对于 n 位 16 进制数,可以表示为:

$$A_{n-1} \times 16^{n-1} + A_{n-2} \times 16^{n-2} + \cdots + A_2 \times 16^2 + A_1 \times 16 + A_0 + B_1 \times 16^{-1} + B_2 \times 16^{-2} + \cdots$$

例如,98BF.5EH 可以表示为 $9 \times 16^3 + 8 \times 16^2 + B \times 16 + F + 5 \times 16^{-1} + E \times 16^{-2}$,即 39103.3671875。

1.1.4 二进制数与十六进制数之间的转换

如果在 $A_{n-1} \times 2^{n-1} + A_{n-2} \times 2^{n-2} + \cdots + A_2 \times 2^2 + A_1 \times 2 + A_0 + B_1 \times 2^{-1} + B_2 \times 2^{-2} + \cdots$ 式中,对于整数部分,从个位开始每四位分为一组;对于小数部分,从“十分”位开始,每四位分为一组,则

$$\begin{aligned} & A_{n-1} \times 2^{n-1} + A_{n-2} \times 2^{n-2} + \cdots + A_2 \times 2^2 + A_1 \times 2 + A_0 + B_1 \times 2^{-1} + B_2 \times 2^{-2} + \cdots \\ &= (A_{n-1} \times 2^3 + A_{n-2} \times 2^2 + A_{n-3} \times 2 + A_{n-4}) \times 2^{n-4} + (A_{n-5} \times 2^3 + A_{n-6} \times 2^2 + A_{n-7} \times 2^1 + \\ & A_{n-8}) \times 2^{n-8} + \cdots + (A_7 \times 2^3 + A_6 \times 2^2 + A_5 \times 2^1 + A_4) \times 2^4 + A_3 \times 2^3 + A_2 \times 2^2 + A_1 \times 2^1 + A_0 \\ &+ (B_0 \times 2^3 + B_1 \times 2^2 + B_2 \times 2^1 + B_3) \times 2^{-4} + \cdots + (B_{n-4} \times 2^3 + B_{n-3} \times 2^2 + B_{n-2} \times 2^1 + B_{n-1}) \\ &\times 2^{-(n-1)+4} \end{aligned}$$

上式与十六进制表示非常接近,括号内就是对应十六进制的数码,而 2^{n-4} 就对应位的权,如:

$$\begin{aligned} 10101010B &= (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) \times 2^4 + (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) \\ &= A \times 2^4 + A \\ &= A \times 16 + A \end{aligned}$$

因此,10101010B = 0AAH。

(1) 二进制数转换成十六进制数时,按如下规则进行:

对于二进制数的整数部分来说,从个位开始,每 4 位作为一组,划分整数部分(如果最后一组不足 4 位,可在前面补 1~3 个零);对于二进制数的小数部分来说,从“十分”位开始,每 4 位作为一组,划分小数部分(同样,当最后一组不足 4 位时,可在后面补 1~3 个零)。然后把每组中的 4 位二进制数用对应的十六进制数表示,即可获得相应的十六进制数。

$$\begin{aligned} & \text{如: } 1110010101.10101B \\ &= 0011 \ 1001 \ 0101.1010 \ 1000 \\ & \quad \quad \quad 3 \quad 9 \quad 5 \quad A \quad 8 \end{aligned}$$

即 1110010101.10101B = 395.A8H

(2) 十六进制数转换为二进制数时,按如下规则进行:

将十六进制数的整数部分和小数部分的每一位十六进制数码用对应的 4 位二进制数表示,删除整数部分前面和小数部分后面多余的零,即可获得对应的二进制数,如:

93FE.3A3H = 1001 0011 1111 1110.0011 1010 0011B

又如:3E.CH = 0011 1110.1100B

= 111110.11B(删除整数部分前面的零和小数部分后面的零)

可见,二进制数与十六进制数之间的转换非常方便,只要记住 4 位二进制数 0000 ~ 1111 与十六进制数 0 ~ F 之间的对应关系即可。下面是二进制数 0000 ~ 1111 对应的十六进制数和十进制数。

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

1.1.5 二进制数和十六进制数的运算

1. 二进制数的运算

(1) 二进制数的加法

二进制数的加法运算规则为:0+0=0;0+1=1;1+0=1(交换律);1+1=10(二进制中的“10”就是十进制的“2”)。例如:

$$\begin{array}{r} 10010110B \\ + 01110011B \\ \hline 100001001B \end{array}$$

(2) 二进制数的减法

二进制数向前借位时,为“10”,例如:

$$\begin{array}{r} 10010110B \\ - 01110011B \\ \hline 00100011B \end{array}$$

(3) 二进制数的乘法

二进制数的乘法运算规则为:0×0=0;0×1=0;1×0=0(交换律);1×1=1。例如:

$$\begin{array}{r} 10010110B \\ \times \quad 101B \\ \hline 10010110 \\ 00000000 \\ + 10010110 \\ \hline 1011101110B \end{array}$$

2. 十六进制数的运算

十六进制数的加法运算规则与十进制数的加法运算规则相同,如 $3H + 4H = 7H$; $7H + 4H = 0BH$ (结果是十进制数的 11,即十六进制数的 $0BH$); $8H + 7H = 0FH$ (结果是十进制数的 15,即十六进制数的 $0FH$); $8H + 9H = 11H$ (结果是十进制数的 17,即十六进制数的 $11H$),例如:

$$\begin{array}{r} 346AH \\ + 589CH \\ \hline 8D06H \end{array}$$

在上式中,个位 A(即 10)加 C(即 12),结果为 22,即十六进制数的“16”,向“十”位进 1,结果为 6;“十”位 $6 + 9 + 1$ (个位进位),结果为 16,即十六进制数中的“10”,向“百”位进 1,结果为 0;“百”位 $4 + 8 + 1$ (十位进位),结果为 13,即十六进制数中的“D”;“千”位 $3 + 5$,结果为 8。

$$\begin{array}{r} 746AH \\ - 589CH \\ \hline 1BCEH \end{array}$$

在上式中,个位向“十”位借 1 后,变成十六进制数的“1A”,即十进制数的 26,再减 C(即十进制数的 12),结果为 14,即十六进制数的“E”;“十”位原本是“6”,个位借 1 后变为“5”,“十”位再向“百”位借 1,变成十六进制数的“15”,即 21,减 9,结果为 12,即十六进制数的“C”;“百”位原本是“4”,“十”位借 1 后变为“3”,“百”位向千位借 1,变成十六进制数的“13”,即 19,减 8,结果为 11,即十六进制数的“B”。

$$\begin{array}{r} 746AH \\ \times \quad 9CH \\ \hline (进位)347 \\ 574F8 \\ (进位)235 \\ + \quad 417BA \\ \hline 46F098 \end{array}$$

可见,十六进制数的乘法与十进制数的乘法运算方法类似,但必须注意将十六进制乘法运算的中间结果转为十六进制数,例如 6×9 ,结果为十进制数的 54,应转化为十六进制数 $36H$ 。

1.2 码制

由于计算机内部所有的数据均采用二进制代码表示,但通过输入设备(如键盘)输入的信息和通过输出设备(如显示器、打印机)输出的信息却是多种多样的,既有字母、数字,又有各种控制字符,甚至汉字。为了方便,人们对计算机中常用的符号进行编码。当通过输入设备向计算机输入某个字符时,计算机自动将该字符转化为对应的二进制数,再进行处理;同时,计算机也将处理结果还原为对应的字符。于是,字符所对应的二进制数就称为该字符的编码。

1.2.1 英文字符的表示方法——ASCII 码

由于计算机只能处理二进制数,因此除了数值本身需要用二进制数形式表示外,字符,包括数码(如 0,1,2,3,4,5,6,7,8,9)、字母(如 A,B,C,D,⋯,X,Y,Z 及 a,b,c,d,⋯,x,y,z)、特殊符号(如 %,!,+,-,= 等)也必须用二进制数表示,即在计算机中需将数码、字母、特殊符号等代码化,以便于计算机识别、存储和处理。

英文属于典型的拼音文字,由字母、数字、特殊符号等组合而成,但这些字母、数字、特殊符号的数目毕竟有限,不过百余个。7 位二进制数可以表示 128 种状态,如果每种状态代表特定的字母或数字,则 7 位二进制数可以表示 128 个字符。例如:可用 0110000B 表示数字 0;0110001B 表示数字 1;1000001B 表示大写字母 A 等。目前,在计算机系统中普遍采用美国标准信息交换代码(American Standard Code for Information Interchange II,简称 ASCII 码)。

在计算机系统中,存储单元的长度通常为 8 位二进制数(即一个字节)。为了存取方便,规定一个存储单元存放一个 ASCII 码,其中低 7 位表示字母本身的编码,第 8 位(bit7)用作奇偶校验位或规定为零(通常如此)。因此,也可以认为 ASCII 码的长度为 8 位(但 bit7 为 0)。128 个字符对于某些特殊应用来说,可能不够,因此就采用 8 位的 ASCII,即扩展 ASCII 码(共有 256 个代码)。其中前 128 个(高位为 0)编码用于表示基本的 ASCII 码,基本 ASCII 码主要用于表示数字、英文字母(大、小写)、标点符号、控制字符等;后 128(高位为 1)个编码用于表示扩展的 ASCII 码,扩展 ASCII 用于表示一些特殊的符号,如希腊字母等。

1.2.2 BCD 码(二进制编码的十进制数)

十进制毕竟是人们最习惯的记数方式,在向计算机输入数据时,常用十进制数输入;但计算机只认识二进制数,因此每一位十进制数必须用二进制数表示。一位十进制数包含 0~9 十个数码,必须用 4 位二进制数表示,这样就需要确定 0~9 与 4 位二进制数 0000B~1111B 之间的对应关系,其中较常用的 8421 BCD 码规定了十进制数 0~9 与 4 位二进制数编码之间的对应关系:

十进制数	8421 BCD 码	十进制数	8421 BCD 码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

注:在 BCD 码中,不使用 1010(0AH)~1111(0FH)。

例如:4567.89 的 BCD 码为:0100 0101 0110 0111.1000 1001(每一位十进制数用相应的 4 位二进制数表示即可)。

尽管 BCD 码比较直观,但 BCD 码与二进制数之间的转换并不方便,需要转换成十进制数后,才能转换为二进制数,反之亦然。

1.2.3 计算机中带符号数的表示方法

在计算机中,对于带符号数来说,一般用最高位表示数的正、负。对于正数,最高位定义为

“0”，对于负数，最高位为“1”，例如：

56H 可以表示 0 1010110 (对于 8 位二进制数来说，b7 位表示数的正负，b6 ~ b0 表示数的绝对值)；-56H 可以表示 1 1010110。

0256H 可以表示 0 000 0010 0101 0110 (对于 16 位二进制数来说，b15 位表示数的正负，b14 ~ b0 表示数的绝对值)；-0256H 可以表示 1 000 0010 0101 0110。

1. 原码

对于带符号数来说，用最高位表示带符号数的正负，其余各位表示该数的绝对值，这种表示方法称为原码表示法。

2. 反码

带符号数也可以用反码表示，反码与原码的关系是：

正数的反码与原码相同，如 $[56H]_{反} = [56H]_{原} = 0\ 1010110B$ 。

负数的反码等于对应正数的原码按位求反。因此，求 -56H 反码的过程如下：

对应正数 56H 的原码为 0 1010110；按位求反后为 1 0101001，即 -56H 的反码为：10101001。或者说，正数的反码与原码相同，而负数的反码是对应负数原码除符号位外，绝对值部分按位取反。

3. 补码

在计算机内，带符号数并不用原码或反码表示，而是用补码表示的，引入“原码”、“反码”的目的是为了方便理解补码的概念。不用原码表示的原因是：用原码表示时，“0”的原码并不惟一，0 的原码可以表示为 0 0000000 (即 +0)，也可以表示为 1 0000000 (即 -0)，这会造成混乱；再就是用原码表示时，减法并不能转化为加法运算，反码也存在类似问题。在计算机中，带符号数用补码表示后，减法可以转化为加法运算，例如：

$56H - 23H = 56H - 23H + 100H$ (100H 是 8 位二进制数能表示的最大数，加上 100H 后，对计算结果没有影响，原因是 8 位二进制数无法存放 100H 中的“1”，即 b8 位)

$$= 56H + 100H - 23H$$

$$= 56H + 0DDH$$

$$= \boxed{1} 33H$$

$$= 33H \text{ (由于 8 位二进制数不能存放 b8 位, 结果 133H 中最高位“1”自然丢失)}$$

显然，56H - 23H 的结果与 56H + 0DDH 相同，即引入补码后，减法可以用加法来完成。可见，在 8 位二进制中，23H 与 0DDH 互为补码。

补码的定义如下：

正数的补码与反码、原码相同；

负数的补码等于它的反码末位加 1。因此，求 -23H 补码的过程如下：

对应正数 23H 的原码为 0 0100011；按位求反后为 1 1011100，即 -23H 的反码；反码末位加 1 后为 1 1011101，即 -23H 的补码为 1 1011101 (相当于无符号数的 0DDH)。

可见，用补码表示时，最高位为 0 时，表示该数为正数，数值部分就是真值；而最高位为 1 时，为负数，数值部分并不是它的真值，必须再求补后，才得到该数的绝对值，如上例中的

-23H 的补码为 1 1011101,按位取反后为 00100010,加 1 后为 00100011,即 23H。

对于 8 位二进制数来说,补码表示的范围是 $-128 \sim +127$;对于 16 位二进制数来说,补码表示的范围是 $-32768 \sim +32767$ 。

例如,求 16 位二进制数中 -23H 的补码过程如下:

对应正数 23H 的原码为 0 000 0000 0010 0011,按位取反后为 1 111 1111 1101 1100;末位加 1 后为 1 111 1111 1101 1101(即 16 位二进制数 -23H 的补码,相当于无符号数的 0FFDDH)。可见,对于同一个数,作为 8 位二进制数的补码和作为 16 位二进制数的补码不同。

1.3 计算机的基本认识

为了理解计算机的硬件组成、工作原理及过程,先来看算盘计算如下代数式的过程:

$$12 \times 34 + 56 \div 7 - 8 =$$

首先要有算盘作为计算工具,在计算机里用“运算器”(简称为 ALU,即算术逻辑运算单元)作为计算工具,由它承担算术运算和逻辑运算。因为在计算机里,除了需要对数据进行加、减、乘、除四则运算外,还需要进行逻辑与、或、非、异或等逻辑运算。其次需要纸和笔记录算式、计算步骤、中间结果及最终结果。在计算机中,起到纸和笔作用的器件是存储器和寄存器(寄存器在 CPU,即中央处理器内,特点是存取速度快,但数量少,用于存放中间结果,而存储器由成千上万个存储单元组成,容量大;与寄存器相比,存取速度慢一些,常用于存放数据、运算步骤,即指令)。在计算上述算式时,先计算 12×34 ,并把中间结果记录下来;然后再计算 $56 \div 7$,再记录中间结果;把上述两步中间结果相加,并记录下来;再减 8。以上计算步骤由人脑控制,如果改用计算机,可用计算机指令写出如下的计算步骤:

MOV B, #34	;将乘数 34 传送到 CPU 的内部寄存器 B
MOV A, #12	;将被乘数 12 传送到 CPU 的内部寄存器 A
MUL AB	;计算 12×34 ,结果的高 8 位存放在寄存器 B 中,低 8 位存放在寄存器 A 中
MOV R2, A	
MOV R3, B	;将中间结果暂时保存到寄存器 R2、R3 中
MOV B, #7	;将除数 7 传送到 CPU 的内部寄存器 B
MOV A, #56	;将被除数 56 传送到 CPU 的内部寄存器 A
DIV AB	;计算 $56 \div 7$,商存放在寄存器 A 中,余数存放在寄存器 B 中
ADD A, R2	;低 8 位相加
MOV R2, A	;把结果暂存到 R2 中
MOV A, R3	;把高 8 位传送到 A 中
ADDC A, #00	;低 8 位相加时可能产生进位,用 ADC 指令将 A 与 00 相加,可将低 8 位相加;产生的进位加到高 8 位中
MOV R3, A	;把结果暂存到 R3 中
MOV A, R2	;把低 8 位传送到 A 中
CLR C	;清进位标志
SUBB A, #08	;低 8 位减 8
MOV R2, A	;把结果传送到 R2 中
MOV A, R3	;把高 8 位传送到 A 中
SUBB A, #00	; $12 \times 34 + 56 \div 7$ 获得的中间结果减 8 时,低 8 位可能产生借位,因此需要用 SUBB 指令将高 8 位与 00 相减