学习任务一

51 系列单片机（AT89S51）基础知识

学习目标

任务说明

单片机是一种集成电路芯片，是采用超大规模集成电路技术把具有数据处理能力的中央处理器（CPU）、随机存储器（RAM）、只读存储器（ROM）、多种 I/O 端口和中断系统、定时器/计数器等（有的单片机还包括显示驱动电路、脉宽调制电路、模拟多路转换器、A/D 转换器等电路）集成到一块硅片上构成的一个小而完善的微型计算机系统，在工业控制领域具有广泛的应用。

在本学习任务中，主要学习单片机的发展历史、结构组成、存储结构、输入输出设备及单片机编程语言等。通过实验使学生加深对单片机编程语言的理解，掌握单片机编程的基本思路和流程以及其运行和控制的基本规律。

知识和能力要求

知识要求

（1）**了解单片机的发展历史和应用范围**；
（2）**掌握 AT89S51 的结构组成**；
（3）**熟悉单片机的存储结构**；
（4）**熟悉单片机的输入/输出（I/O）端口**；
（5）**掌握单片机编程语言**。

能力要求

（1）**能够根据控制需要连接相对简单的单片机外围电路**；
（2）**能够读懂简单的单片机控制程序**。

任务准备

一、单片机概述

无论规模大小、性能高低，计算机的硬件系统都由运算器、存储器、输入设备、输出设备以及控制器等单元组成。在通用计算机中，这些单元被分成若干块独立的芯片，通过电路连接构成一台完整的计算机。而单片机技术则将这些单元全部集成到一块集成电路中，即一块芯片就构成了一个完整的计算机系统。

1. 单片机简介

单片微型计算机简称单片机，是典型的嵌入式微控制器(micro controller unit)，单片机芯片常用英文母的缩写 MCU 表示。单片机又称单片微控制器，它不是完成某一个逻辑功能的芯片，而是把一个计算机系统集成到一个芯片上。单片机由运算器、控制器、存储器、输入输出设备构成，相当于一个微型的计算机(最小系统)。和计算机相比，单片机只缺少了外围设备。单片机的体积小、质量轻、价格便宜，为学习、应用和开发提供了便利条件。同时，学习使用单片机是了解计算机原理与结构的最佳选择，它最早被用在工业控制领域。

单片机由芯片内仅有 CPU 的专用处理器发展而来。最早的设计理念是通过将大量外围设备和 CPU 集成在一个芯片中，使计算机系统更小，更容易集成进复杂的而对体积要求严格的控制设备当中。

2. 单片机的发展及应用

20 世纪 70 年代，美国仙童半导体公司(Fairchild Semiconductor)首先推出了第一款单片机 F-8，随后 Intel 公司推出了 MCS-48 单片机系列，其他一些公司，如 Motorola、Zilog 等也先后推出了自己的单片机，取得了一定的成果，这是单片机的起步与探索阶段。总体来说，这一阶段的单片机性能较弱，属于中低档产品。随着集成技术的提高以及 CMOS 技术的发展，单片机的性能也随之改善，高性能的 8 位单片机相继问世。1980 年，Intel 公司推出了 8 位高档 MCS-51 系列单片机，性能得到很大的提高，应用领域也大为扩展，这是单片机的完善阶段。1983 年，Intel 公司推出了 16 位 MCS-96 系列单片机，加入了更多的外围接口，如模/数转换器(ADC)、看门狗(watch dog timer，WDT)、脉宽调制器(PWM)等，其他一些公司也相继推出了各自的高性能单片机系统。随后许多用在高端单片机上的技术被下移到 8 位单片机上，这些单片机内部一般都有非常丰富的外围接口，强化了智能控制器的特征，这是 8 位单片机与 16 位单片机的推出阶段。随着科学技术的进步，早期的 8 位中低档单片机逐渐被淘汰，但 8 位单片机并没有消失，尤其是以 80C51 为内核的单片机，不仅没有消失，还呈现出快速发展的趋势。

近年来，Intel、Motorola 等公司又先后推出了性能更为优越的 32 位单片机，单片机的应用达到了一个更新的层次。

单片机的型号有 8031、8051、80C51、80C52、8751、89S51 等，那么这些型号的单片机有什么区别呢？在这里简单介绍一下。

8031/8051/8751 是 Intel 公司早期的产品。

8031 片内不带程序存储器 ROM，使用时用户需外接程序存储器和一片逻辑电路 74LS373(74LS373 为 8 位地址锁存器)，外接的程序存储器多为 EPROM(一种断电后仍能

保留数据的计算机储存芯片，即非易失性的芯片。它是一组浮栅晶体管，被一个提供比电子电路常用电压更高电压的电子器件分别编程。一旦编程完成后，EPROM 只能用强紫外线照射来擦除。通过封装顶部能看见硅片的透明窗口，很容易识别 EPROM，这个窗口同时用来进行紫外线擦除)的 2764 系列。用户若想对写入到 EPROM 中的程序进行修改，必须先用一种特殊的紫外线灯将其照射擦除，之后再写入。写入到外接程序存储器的程序代码没有保密性可言。

8051 片内有 4 KB 的 ROM，无须外接外存储器和 74LS373，更能体现“单片”的简练。但是用户自编的程序无法烧写到其 ROM 中，只有将程序交芯片厂烧写，并且是一次性的，之后不能改写其内容。

8751 与 8051 基本一样，但 8751 片内有 4 KB 的 EPROM，用户可以将自己编写的程序写入单片机的 EPROM 中进行现场实验与应用，EPROM 的改写同样需要用紫外线灯照射一定时间擦除后再烧写。

由于上述类型的单片机应用较早，影响很大，已成为事实上的工业标准。后来很多芯片厂商以各种方式与 Intel 公司合作，也推出了同类型的单片机，如同一种单片机的多个版本一样，虽都在不断地改变制造工艺，但内核却一样，也就是说这类单片机指令系统完全兼容，绝大多数管脚也兼容，在使用上基本可以直接互换。人们统称这些与 8051 内核相同的单片机为“51 系列单片机”。

在众多的 51 系列单片机中，ATMEL 公司的 AT89C51、AT89S51 更实用，因它不但和 8051 指令、管脚完全兼容，而且其片内的 4 KB 程序存储器是 Flash 工艺的，对于这种工艺的存储器，用户可以用电的方式对其瞬间擦除、改写，一般专为 ATMEL AT89××做的编程器均带有这些功能。显而易见，这种单片机对开发设备的要求很低，开发时间也大大缩短。写入单片机内的程序还可以进行加密，这又很好地保护了用户的劳动成果。

二、51 系列单片机(AT89S51)的结构组成

1. 单片机的内部组成

AT89S51 把作为控制应用所必需的基本功能部件都集成在一个尺寸有限的集成电路芯片上。AT89S51 单片机片内硬件组成结构如图 1-1 所示。

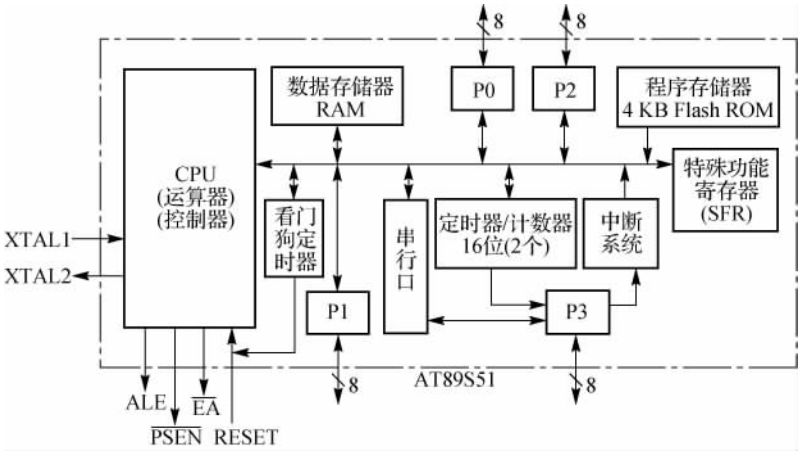


图 1-1　AT89S51 单片机片内硬件组成结构

1)微处理器(CPU)

8位的CPU,与通用CPU基本相同,同样包括了运算器和控制器两大部分,还有面向控制的位处理功能。

2)数据存储器(Flash RAM)

数据存储器片内为128 B(52子系列为256 B),片外最多可扩展64 KB。片内128 B的RAM以高速RAM的形式集成,可加快单片机运行的速度和降低功耗。

3)程序存储器(ROM)

片内集成有4 KB(AT89S52为8 KB,AT89C55为20 KB)的Flash存储器,如片内容量不够,片外可外扩至64 KB。

4)中断系统

中断系统具有6个中断源,常用5个,这些中断源可分为2级中断优先权。

5)定时器/计数器

片内有2个16位定时器/计数器(52子系列有3个),具有4种工作方式。

6)看门狗定时器(WDT)

片内有1个WDT,当CPU由于干扰使程序陷入死循环或跑飞状态时,WDT可使程序恢复正常运行。

7)串行口

片内有1个全双工异步串行口,具有4种工作方式。可进行串行通信,扩展并行I/O端口,还可与多个单片机构成多机系统。

8)P1口、P2口、P3口、P0口

P1口、P2口、P3口、P0口为4个8位并行I/O端口。

9)特殊功能寄存器(SFR)

片内共有26个特殊功能寄存器,对片内各功能部件进行管理、控制和监视,是各个功能部件的控制寄存器和状态寄存器,映射在片内RAM区80H～FFH的地址区间内。

2.单片机的引脚功能

AT89S51的外形引脚如图1-2所示。

AT89S51与51系列中各种型号芯片的引脚互相兼容。目前多采用40只引脚双列直插式的封装形式。下面按照引脚序号对单片机各个引脚做一个总体的介绍。1号～8号引脚为P1口(P1.0～P1.7),在串行编程和校验时,MOSI/P1.5、MISO/P1.6和SCK/P1.7分别是串行数据输入、输出和移位脉冲引脚,可以实现在线编程功能,即AT89S51芯片可以在PCB上直接下载程序,无须将芯片取下来放到编程器去写程序,从而提高使用的灵活性;9号引脚为复位引脚;10号～17号引脚为P3口(P3.0～P3.7);18号、19号引脚为时钟引脚,外接晶体振荡器;20号引脚为电源接地;21号～28号引脚为P2口(P2.0～P2.7);29号引脚为片外程序存储器读选通信号;30号引脚为地址锁存和对片内Flash编程双功能;31号引脚具有外部程序存储器访问允许控制端和对片内Flash编程双功能;32号～39号引脚为P0口(P0.0～P0.7);40号引脚为电源信号。

引脚按其功能可分为如下三类。

(1)电源及时钟引脚:V_{CC}、V_{SS};XTAL1、XTAL2。

(2)控制引脚:$\overline{PSEN}$、ALE/$\overline{PROG}$、$\overline{EA}$/V_{PP}、RST(RESET)。

(3)I/O 端口引脚:P0、P1、P2、P3,为 4 个 8 位 I/O 端口。

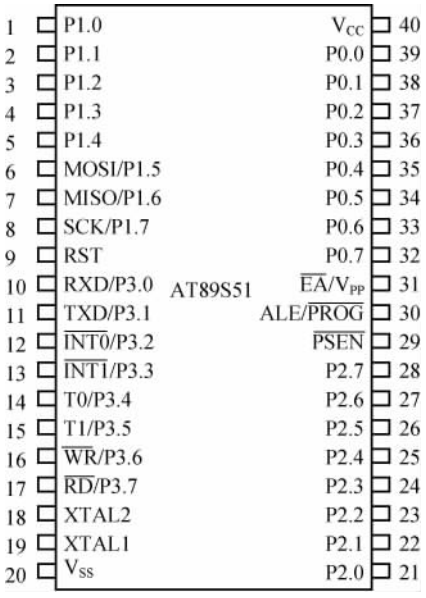


图 1-2　AT89S51 双列直插封装形式引脚

其中几个比较重要的引脚功能介绍如下。

1)电源引脚

(1)V_{CC}(40 脚):+5 V 电源。

(2)V_{SS}(20 脚):接地线。

2)时钟引脚

(1)XTAL1(19 脚)。片内振荡器反相放大器和时钟发生器的输入端。用片内振荡器时,该引脚接外部石英晶体和微调电容。外接时钟源时,该引脚接外部时钟振荡器的信号。

(2)XTAL2(18 脚)。片内振荡器反相放大器的输出端。当使用片内振荡器时,该引脚连接外部石英晶体和微调电容。当使用外部时钟源时,该引脚悬空。

3)控制引脚

(1)RST(RESET,9 脚)。复位信号输入端,高电平有效。在此引脚加上持续时间大于两个机器周期的高电平,可使单片机复位。正常工作时,此引脚电平应≤0.5 V。当看门狗定时器溢出输出时,该引脚将输出长达 96 个时钟振荡周期的高电平。

(2)$\overline{EA}/V_{PP}$(enable address/voltage pulse of programing,31 脚)。

①$\overline{EA}$为引脚第一功能,即外部程序存储器访问允许控制端。

$\overline{EA}$=1。在 PC 值不超出 0FFFH(即不超出片内 4 KB Flash 存储器的地址范围)时,单片机读片内程序存储器中的程序;而当 PC 值超出 0FFFH(即超出片内 4 KB Flash 地址范围)时,将自动转向读取片外 60 KB(1000H～FFFFH)程序存储器空间中的程序。

$\overline{EA}$=0。只读取外部程序存储器中的内容,读取的地址范围为 0000H～FFFFH,片内的 4 KB Flash 程序存储器不起作用。

②V_{PP}为引脚第二功能,即对片内 Flash 进行编程时,V_{PP}引脚接编程电压。

温馨提示:以上 6 个引脚接线正确后,就形成了单片机最小系统,就可以保证单片机正

常工作了。

3. 单片机的外围电路

1)时钟电路及时序

时钟电路用于产生 AT89S51 工作时所必需的控制信号。AT89S51 单片机的内部电路在时钟信号的控制下,严格按时序执行指令。执行指令时,CPU 首先到程序存储器中取出需要执行的指令操作码,然后译码,并由时序电路产生一系列控制信号完成指令所规定的操作。CPU 发的时序信号有两类,一类用于控制片内各个功能部件,用户无须了解;另一类用于控制片外存储器或 I/O 端口,这部分时序对于分析、设计硬件接口电路至关重要。

时钟频率直接影响单片机的速度,时钟电路的质量也直接影响单片机系统的稳定性。常用的时钟电路有两种方式,一种是内部时钟方式,另一种是外部时钟方式。

(1)内部时钟方式。AT89S51 内部有一个用于构成振荡器的高增益反相放大器,输入端为芯片引脚 XTAL1,输出端为引脚 XTAL2。这两个引脚跨接石英晶体振荡器和微调电容,构成一个稳定的自激振荡器, AT89S51 内部时钟方式的电路如图 1-3 所示。

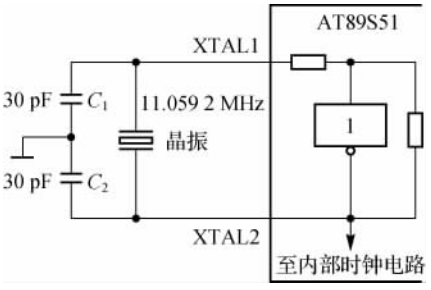


图 1-3　AT89S51 的内部时钟方式电路

电容 C_1 和 C_2 的典型值通常选择为 30 pF。电容的大小会影响振荡器频率高低、振荡器的稳定性和起振的快速性。晶振频率范围通常是 1.2～12 MHz。晶振频率越高,单片机的运行速度就越快。但反之,运行速度快对存储器的速度要求就高,对印制电路板的工艺要求也高,即要求线间的寄生电容要小。晶体和电容应尽可能与单片机芯片靠近,以减少寄生电容,保证振荡器稳定、可靠地工作。为提高温度稳定性,可采用温度稳定性能好的电容。AT89S51 常选 6 MHz 或 12 MHz 的石英晶体。随着集成电路制造工艺技术的发展,单片机的时钟频率也在逐步提高,AT89S51 和 AT89S52 芯片的时钟最高频率已达 33 MHz。

(2)外部时钟方式。使用现成的外部振荡器产生脉冲信号,常用于多片 AT89S51 同时工作,以便于多片单片机之间的同步。外部时钟源直接接到 XTAL1 端,XTAL2 端悬空,外部时钟方式的电路如图 1-4 所示。

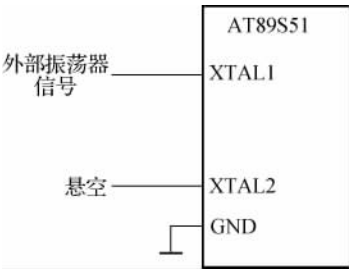


图 1-4　AT89S51 的外部时钟方式电路

2)时钟周期、机器周期与指令周期

单片机执行的指令均是在 CPU 控制器的时序控制电路的控制下进行的,各种指令时序均与时钟周期相关。

(1)时钟周期。时钟周期是时钟控制信号的基本时间单位。若晶振频率为 f_{osc},则时钟周期 $T_{osc}=1/f_{osc}$。例如,$f_{osc}=6$ MHz,则 $T_{osc}=1/6$ μs。

(2)机器周期。CPU 完成一个基本操作所需时间为机器周期。单片机中常把执行一条指令的过程分为几个机器周期。每个机器周期完成一个基本操作,如取出指令、读或写数据等。

1 个机器周期包括 12 个时钟周期,分 6 个状态:S1～S6。每个状态又分两拍:P1 和 P2。因此,一个机器周期中的 12 个时钟周期表示为 S1P1、S1P2、S2P1、S2P2、……、S6P2,如图 1-5 所示。

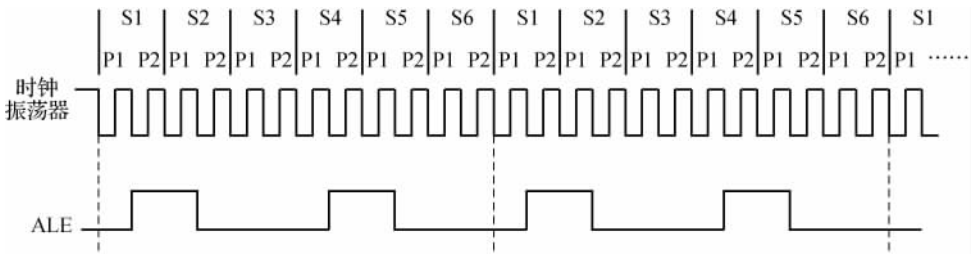


图 1-5　AT89S51 的机器周期

(3)指令周期。指令周期是指执行一条指令所需的时间。对于简单的单字节指令,取出指令立即执行,只需一个机器周期的时间。而有些复杂的指令,如转移、乘、除指令则需两个或多个机器周期。从指令执行时间看,单字节和双字节指令一般为单机器周期和双机器周期,三字节指令都是双机器周期,只有乘、除指令占用 4 个机器周期。

3)复位操作和复位电路

(1)复位操作。复位操作即单片机的初始化操作,给复位脚 RST 加上大于 2 个机器周期(即 24 个时钟振荡周期)的高电平就可以使 AT89S51 复位。复位时,PC 初始化为 0000H,程序从 0000H 单元开始执行。

除系统的正常初始化外,当程序出错(如程序跑飞)或操作错误使系统处于死锁状态时,需按复位键使 RST 脚为高电平,使 AT89S51 摆脱“跑飞”或“死锁”状态而重新启动程序。

复位操作还对其他一些寄存器有影响,这些寄存器复位时的状态见表 1-1。

表 1-1　复位时片内各寄存器的状态

寄 存 器	复位状态	寄 存 器	复位状态
PC	0000H	TMOD	00H
ACC	00H	TCON	00H
B	00H	TH0	00H
SP	07H	TL0	00H

续表

寄 存 器	复位状态	寄 存 器	复位状态
DPTR	0000H	TH1	00H
P0～P3	FFH	TL1	00H
IP	XXX00000B	SCON	00H
IE	0XX00000B	SBUF	XXXXXXXXB
DP0L	00H	PCON	0XXX0000B
DP1L	00H	AUXR	XXXX0XX0B
DP1H	00H	AUXR1	XXXXXXX0B
WDTRST	XXXXXXXXB	—	—

(2)复位电路。复位电路分为上电自动复位和按键复位两种。上电自动复位是给电容 C 充电加给 RST 引脚一个短的高电平信号,此信号随着 V_{CC} 对电容 C 的充电过程而逐渐回落,即 RST 引脚上的高电平持续时间取决于电容 C 充电时间。为保证系统可靠复位,RST 引脚上的高电平必须维持足够长的时间,如图 1-6 所示。除了上电复位外,有时还需要按键复位,按键复位电路如图 1-7 所示。在实际应用中,为了控制方便,提高可靠性,单片机的复位电路往往包含以上两种复位方式。

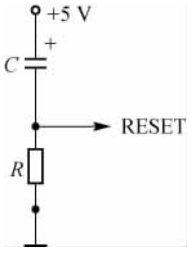


图 1-6　上电复位电路

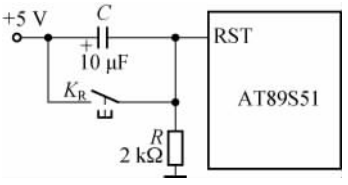


图 1-7　按键复位电路

三、单片机的存储器结构

AT89S51 单片机的存储器组织结构与一般微机不同,一般微机通常是程序和数据共用一个存储空间,属于 Von Neumann 结构(冯·诺依曼型结构),AT89S51 单片机把程序存储器空间和数据存储器相互分离开来,属于 Harvard 结构(哈佛型结构)。

AT89S51 单片机的存储器组织分 3 个不同的存储地址空间:64 KB 的程序存储器地址空间(包括片内 ROM 和片外 ROM),64 KB 的外部数据存储器地址空间,256 B 的内部数据存储器地址空间(其中 128 B 由特殊功能寄存器占用)。汇编语言对这 3 个不同的存储器空间进行数据传送时,必须分别采用 3 种不同形式的指令。AT89S51 单片机存储器组织结构如图 1-8 所示。

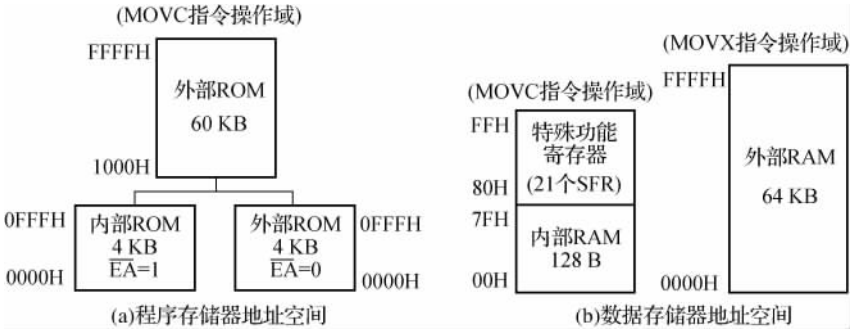


图 1-8　AT89S51 单片机存储器组织结构

1. 单片机的数据存储器 RAM

RAM 用于存放运算的中间结果，用作缓冲和数据暂存，以及设置特征标志等。

AT89S51 片内有 256 B 的 RAM 空间，片外有 64 KB 的 RAM 空间，两个存储空间独立寻址。

1) 内部 RAM

低 128 字节地址空间(00H～7FH)为内部 RAM 区，是供用户使用的数据存储器单元，作为处理问题的数据缓冲器。高 128 字节地址空间(80H～FFH)为特殊功能寄存器区(SFR 区)，共 21 个特殊功能寄存器，也就是 128 个字节单元中只有 21 个字节单元能够被用户使用。内部 RAM 存储空间小，仅用 8 位地址寻址，但存取速度比外部 RAM 快。内部 RAM 低 128 字节单元的划分如图 1-9 所示。内部 RAM 低 128 字节单元按用途可分为 3 个区域。

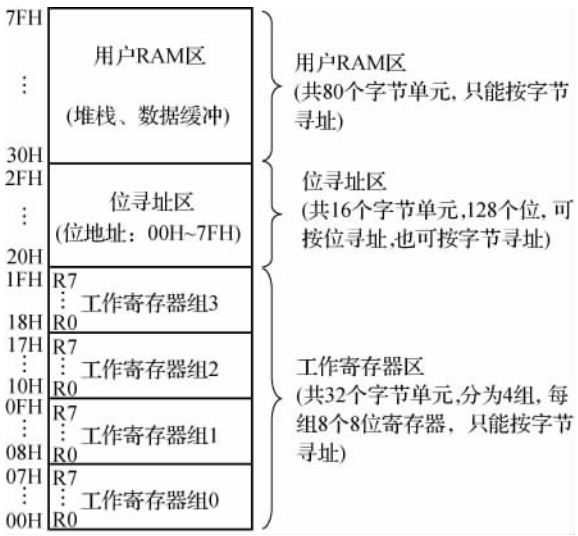


图 1-9　内部 RAM 低 128 字节单元的划分

(1)工作寄存器区。内部 RAM 的 00H～1FH 为工作寄存器区，共 32 个字节，分为 4 组，每组为 8 个 8 位寄存器(R0～R7)。在任一时刻，CPU 只能使用其中的一组寄存器，当前程序使用的工作寄存器组是由程序状态字 PSW 的 RS0、RS1 位来选择的。工作寄存器选择方式见表 1-2。

表 1-2　工作寄存器选择方式

PSW.4(RS1)	PSW.3(RS0)	当前使用的工作寄存器组
0	0	第 0 组(00H～07H)
0	1	第 1 组(08H～0FH)
1	0	第 2 组(10H～17H)
1	1	第 3 组(18H～1FH)

(2)位寻址区。内部 RAM 的 20H～2FH 字节为可位寻址区域,这 16 个字节共 128 位,每一位都有一个位地址,位编址为 00H～7FH,用户可用程序对它们直接进行清零、置位、取反和测试等操作。位寻址区的 RAM 单元也可按字节寻址,作为一般的数据缓冲器使用。

“位”有两种表示方式:以“位地址”的形式,如位寻址区的最后一个位是 7FH;以“存储单元地址＋位”的形式表示,如位寻址区的最后一个位表示为“2FH.7”。

内部 RAM 位寻址区的位地址见表 1-3。

表 1-3　内部 RAM 位寻址区位地址

单元地址	D7	D6	D5	D4	D3	D2	D1	D0
2FH	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H
2EH	77H	76H	75H	74H	73H	72H	71H	70H
2DH	6FH	6EH	6DH	6CH	6BH	6AH	69H	68H
2CH	67H	66H	65H	64H	63H	62H	61H	60H
2BH	5FH	5EH	5DH	5CH	5BH	5AH	59H	58H
2AH	57H	56H	55H	54H	53H	52H	51H	50H
29H	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H
28H	47H	46H	45H	44H	43H	42H	41H	40H
27H	3FH	3EH	3DH	3CH	3BH	3AH	39H	38H
26H	37H	36H	35H	34H	33H	32H	31H	30H
25H	2FH	2EH	2DH	2CH	2BH	2AH	29H	28H
24H	27H	26H	25H	24H	23H	22H	21H	20H
23H	1FH	1EH	1DH	1CH	1BH	1AH	19H	18H
22H	17H	16H	15H	14H	13H	12H	11H	10H
21H	0FH	0EH	0DH	0CH	0BH	0AH	09H	08H
20H	07H	06H	05H	04H	03H	02H	01H	00H

(3)用户 RAM 区。内部 RAM 的 30H～7FH 字节为用户 RAM 区,即通用数据缓冲区,共 80 个单元,作为一般数据缓冲使用。52 子系列的用户 RAM 区为 30H～FFH 范围内的 208 个字节。对于用户 RAM 区,只能以存储单元的形式来使用,没有其他任何规定和限制。一般把堆栈开辟在此区中。

2)单片机的特殊功能寄存器 SFR

单片机采用特殊功能寄存器集中控制各功能部件。特殊功能寄存器映射在片内 RAM 的 80H～FFH 区域中,共 26 个。有些还可以进行位寻址,位地址见表 1-4。

与 AT89C51 相比,AT89S51 新增 5 个 SFR——DP1L、DP1H、AUXR、AUXR1 和 WDTRST,已在表 1-4 中列出。

凡是可位寻址的 SFR,字节地址末位只能是 0H 或 08H。另外,若读/写未定义单元,将得到一个不确定的随机数。特殊功能寄存器 SFR 名称及其分布见表 1-4。

表 1-4　特殊功能寄存器 SFR 名称及其分布

序　号	特殊功能寄存器符号	名　称	字节地址	位 地 址	复 位 值
1	P0	P0 口	80H	80H～87H	FFH
2	SP	堆栈指针	81H	—	07H
3	DP0L	数据指针 DPTR0 低字节	82H	—	00H
4	DP0H	数据指针 DPTR0 高字节	83H	—	00H
5	DP1L	数据指针 DPTR1 低字节	84H	—	00H
6	DP1H	数据指针 DPTR1 高字节	85H	—	00H
7	PCON	电源控制寄存器	87H	—	00H
8	TCON	定时器/计数器控制寄存器	88H	—	0XXX0000B
9	TMOD	定时器/计数器方式控制	89H	88H～8FH	00H
10	TL0	定时器/计数器 0(低字节)	8AH	—	00H
11	TL1	定时器/计数器 1(低字节)	8BH	—	00H
12	TH0	定时器/计数器 0(高字节)	8CH	—	00H
13	TH1	定时器/计数器 1(高字节)	8DH	—	00H
14	AUXR	辅助寄存器	8EH	—	XXX00XX0B
15	P1	P1 口寄存器	90H	90H～97H	FFH
16	SCON	串行控制寄存器	98H	98H～9FH	00H
17	SBUF	串行发送数据缓冲器	99H	—	XXXXXXXXB
18	P2	P2 口寄存器	A0H	A0H～A7H	FFH
19	AUXR1	辅助寄存器 1	A2H	—	XXXXXXX0B
20	WDTRST	看门狗复位寄存器	A6H	—	XXXXXXXXB
21	IE	中断允许控制寄存器	A8H	A8H～AFH	0XX00000B
22	P3	P3 口寄存器	B0H	B0H～B7H	FFH
23	IP	中断优先级控制寄存器	B8H	B8H～BFH	XX000000B
24	PSW	程序状态字寄存器	D0H	D0H～D7H	00H
25	A 或(ACC)	累加器	E0H	E0H～E7H	00H
26	B	B 寄存器	F0H	F0H～F7H	00H

常用的特殊功能寄存器如下。

(1)ACC,累加器,通常用 A 表示。ACC 不是一个加法器。之所以取名加法器,是因为在运算器运算时,其中一个加数一定是在 ACC 中的。累加器自身带有全零标志 Z,若 A=0,则 Z=1;若 A≠0,则 Z=0。该标志常用作程序分支转移的判断条件。

(2)B,B 寄存器。在做乘、除法时存放乘数或除数,不做乘、除法时,使用比较随意。

(3)AUXR,辅助寄存器,它的各位功能见表 1-5。

表 1-5　AUXR 各位功能

D7	D6	D5	D4	D3	D2	D1	D0
—	—	—	WDIDLE	DISRTO	—	—	DISALE

AUXR 各位功能说明如下。

DISALE:ALE 的禁止/允许位。当 DISALE=0 时,ALE 有效,发出脉冲。当 DISALE=1 时,ALE 仅在执行 MOVC 和 MOVX 指令时有效,不访问外部存储器时,ALE 不输出脉冲信号。

DISRTO:禁止/允许 WDT 溢出时的复位输出。当 DISRTO=0 时,WDT 溢出时,在 RST 引脚输出一个高电平脉冲。当 DISRTO=1 时,RST 引脚仅为输入脚。

WDIDLE:WDT 在空闲模式下的禁止/允许位。当 DISRTO=0 时,WDT 在空闲模式下继续计数。当 DISRTO=1 时,WDT 在空闲模式下暂停计数。

(4)PSW。程序状态字。程序状态字内存放了 CPU 工作时的很多状态,为此我们可以了解 CPU 的当前状态,并做出相应的处理,它的各位功能见表 1-6。

表 1-6　PSW 各位功能

D7	D6	D5	D4	D3	D2	D1	D0
CY	AC	F0	RS1	RS0	OV	—	P

PSW 各位功能的说明如下。

CY:进位标志。8051 中的运算器是一种 8 位的运算器,我们知道,8 位运算器只能表示 0～255,如果做加法,两数相加可能会超过 255,这样最高位就会丢失,造成运算的错误,进位标志就可防止此类错误发生。有进、借位时,CY=1;无进、借位时,CY=0。例如,78H+97H(01111000+10010111)。

AC:辅助进位标志,辅助进、借位(高半字节与低半字节间的进、借位)。例如,57H+3AH(01010111+00111010)。

F0:用户标志位,由用户(编程人员)决定什么时候用,什么时候不用。

RS1、RS0:工作寄存器组选择位。

OV:溢出标志位。运算结果按补码运算理解。有溢出时,OV=1;无溢出时,OV=0。

P:奇偶校验位。奇偶校验位是用来表示 ALU 运算结果中二进制数位“1”的个数的奇偶性的。运算结果有奇数个 1,P=1;运算结果有偶数个 1,P=0。例如,某运算结果是 78H(01111000),显然 1 的个数为偶数,所以 P=0。

(5)数据指针 DPTR0 和 DPTR1。双数据指针寄存器,便于访问数据存储器。DPTR0

为 AT89C51 单片机原有的数据指针。DPTR1 为新增加的数据指针。

AUXR1 的 DPS 位用于选择两个数据指针。当 DPS＝0 时，选用 DPTR0；当 DPS＝1 时，选用 DPTR1。数据指针可作为一个 16 位寄存器来用，也可作为两个独立的 8 位寄存器 DP0H(或 DP1H)和 DP0L(或 DP1L)来用。

(6)AUXR1。辅助寄存器，它的各位功能见表 1-7。

表 1-7　AUXR1 各位功能

D7	D6	D5	D4	D3	D2	D1	D0
—	—	—	—	—	—	—	DPS

表中 DPS 为数据指针寄存器选择位。DPS＝0 时，选择数据指针寄存器 DPTR0。DPS＝1 时，选择数据指针寄存器 DPTR1。

(7)SP 堆栈指针。

日常生活中，我们都注意到过这样的现象，家里洗的碗，一只一只摞起来，最早放上去的放在最下面，而最晚放上去的放在最上面，在取的时候正好相反，先从最上面取，这种现象我们用一句话来概括："先进后出，后进先出。"生活中此类现象比比皆是，┆　工地上堆放的砖头、材料，仓库里放的货物，都是"先进后出，后进先出"，这实际是一种存取物品的规则，我们称之为"堆栈"。

在单片机中，我们也可以在 RAM 中构造一个区域用来存放数据，这个区域存放数据的规则就是"先进后出，后进先出"，我们称之为"堆栈"。为什么需要这样来存放数据呢？存储器本身不是可以按地址来存放数据吗？因为知道了地址的确就可以知道里面的内容，但如果我们需要存放的是一批数据，需要知道每一个数据的地址，如果我们让数据一个接一个地放置，那么只要知道第一个数据所在地址单元就可以了，所以利用堆栈方法来放数据可以简化操作。

单片机中能存放数据的区域有限，我们不能够专门分配一块地方做堆栈，所以就在内存(RAM)中开辟一块地方用于堆栈。因为 51 系列单片机是一种通用的单片机，各人的实际需求各不相同，所以 51 系列单片机中堆栈的位置是可以变化的，它们由编程人员来定，而这种变化就体现在 SP 中值的变化。

(8)WDT。看门狗定时器。WDT 包含一个 14 位计数器和看门狗定时器复位寄存器——WDTRST。当 CPU 由于受到干扰而使程序陷入死循环或跑飞状态时，WDT 提供了一种使程序恢复正常运行的有效手段。有关 WDT 在抗干扰设计中的应用以及低功耗模式下运行的状态，请参阅相关资料。

2. 单片机的程序存储器 ROM

ROM 用于存放程序及表格常数，读取 ROM 的指令为"MOVC"。AT89C51 片内有 4 KB 的 ROM，外部可用 16 位地址线扩展到最大 64 KB 的 ROM 空间。片内 ROM 和外部扩展 ROM 是统一编址的。

当芯片引脚 EA 为高电平时，程序计数器 PC 在 0000H～0FFFH(4 KB)地址时从内部 ROM 取指令，超过 4 KB 时，CPU 自动转向外部 ROM 执行程序。

如果 EA 为低电平(接地)，则所有取指令操作均在外部 ROM 中进行，这时外部扩展的 ROM 从 0000H 开始编址。0000H 单元是复位入口，单片机复位后，CPU 总是从 0000H 单

元开始执行程序。0000H～0002H 单元安排一条无条件转移指令，使之转向主程序的入口地址。

ROM 中的特定单元(0003H～002AH)共 40 个，均匀地分为 5 段，是给系统默认使用的，分别作为 5 个中断源的中断地址区。

四、单片机的输入/输出(I/O)端口

I/O 端口就是输入/输出端口。AT89S51 单片机拥有 4 个 8 位并行 I/O 端口，即 P0、P1、P2 和 P3，每个端口都是 8 位准双向口，共占 32 根引脚，每一条 I/O 线都能独立地用作输入或输出，每个端口都包括一个锁存器(即特殊功能寄存器 P0～P3)、一个输出驱动器和输入缓冲器，用作输出数据时可以锁存，用作输入数据时可以缓冲。

1. P0 口(P0.0～P0.7)

P0 口是一个双功能的 8 位并行端口，字节地址为 80H，位地址为 80H～87H。端口的各位具有完全相同但又相互独立的电路结构，P0 口某一位的位电路结构如图 1-10 所示。

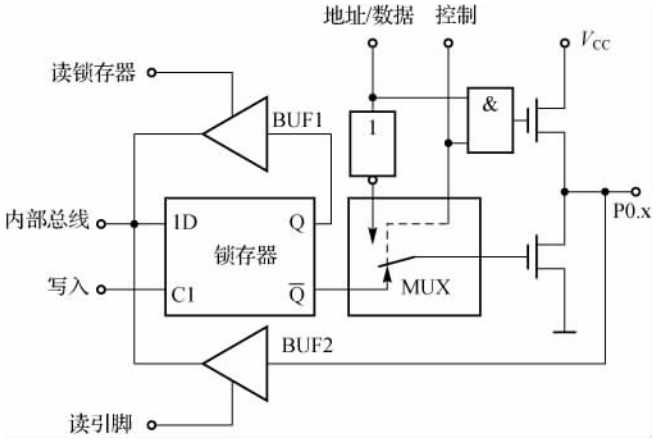


图 1-10　P0 口某一位的位电路结构

P0 口的工作过程分析如下。

1)P0 口用作地址/数据总线

外扩存储器或 I/O 时，P0 口作为单片机系统复用的地址/数据总线使用。当作为地址或数据输出时，控制信号为 1，硬件自动使转接开关 MUX 打向上面，接通反相器的输出，同时使与门处于开启状态。当输出的地址/数据信息为 1 时，与门输出为 1，上方的场效应管导通，下方的场效应管截止，P0.x 引脚输出为 1；当输出的地址/数据信息为 0 时，上方的场效应管截止，下方的场效应管导通，P0.x 引脚输出为 0。输出电路是上、下两个场效应管形成的推拉式结构，大大提高了负载能力，上方的场效应管此时起到内部上拉电阻的作用。

当 P0 口作为数据输入时，仅从外部存储器(或 I/O)读入信息，对应的控制信号为 0，MUX 接通锁存器的$\overline{Q}$端。

由于 P0 口作为地址/数据复用方式访问外部存储器时，CPU 自动向 P0 口写入 0FFH，使下方场效应管截止，上方场效应管由于控制信号为 0 也截止，从而保证数据信息的高阻抗

输入，从外部存储器输入的数据信息直接由 P0.x 引脚通过输入缓冲器 BUF2 进入内部总线。

具有高阻抗输入的 I/O 端口应具有高电平、低电平和高阻抗 3 种状态的端口。因此，P0 口作为地址/数据总线使用时是一个真正的双向端口，简称双向口。

2)P0 口用作通用 I/O 端口

当 P0 口不作为系统的地址/数据总线使用时，此时 P0 口也可作为通用的 I/O 端口使用。作通用的 I/O 端口时，对应的控制信号为 0，MUX 打向下面，接通锁存器的$\overline{Q}$端，与门输出为 0，上方场效应管截止，形成的 P0 口输出电路为漏极开路输出。

(1)P0 口作输出口使用时，来自 CPU 的“写”脉冲加在 D 锁存器的 CP 端，内部总线上的数据写入 D 锁存器，并由引脚 P0.x 输出。当 D 锁存器为 1 时，$\overline{Q}$端为 0，下方场效应管截止，输出为漏极开路，此时，必须外接上拉电阻才能有高电平输出；当 D 锁存器为 0 时，下方场效应管导通，P0 口输出为低电平。

(2)P0 口作输入口使用时，有两种读入方式：“读锁存器”和“读引脚”。当 CPU 发出“读锁存器”指令时，锁存器的状态由 Q 端经上方的三态缓冲器 BUF1 进入内部总线；当 CPU 发出“读引脚”指令时，锁存器的输出状态＝1(即$\overline{Q}$端为 0)，而使下方场效应管截止，引脚的状态经下方的三态缓冲器 BUF2 进入内部总线。

2. P1 口(P1.0～P1.7)

P1 口为单功能的 I/O 端口，字节地址为 90H，位地址为 90H～97H。P1 口某一位的位电路结构如图 1-11 所示。

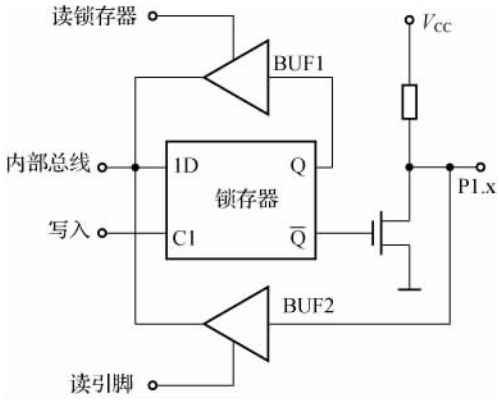


图 1-11　P1 口某一位的位电路结构

P1 口的工作过程分析如下。

P1 口只能作为通用的 I/O 端口使用。

(1)P1 口作输出口使用时，若 CPU 输出 1，Q＝1，$\overline{Q}$＝0，场效应管截止，P1 口引脚的输出为 1；若 CPU 输出 0，Q＝0，$\overline{Q}$＝1，场效应管导通，P1 口引脚的输出为 0。

(2)P1 口作为输入口使用时，分为“读锁存器”和“读引脚”两种方式。“读锁存器”时，锁存器的输出端 Q 的状态经输入缓冲器 BUF1 进入内部总线；“读引脚”时，先向锁存器写 1，使场效应管截止，P1.x 引脚上的电平经输入缓冲器 BUF2 进入内部总线。

(3)P1 口由于内部上拉电阻，无高阻抗输入状态，故为准双向口。P1 口“读引脚”输入

时，必须先向锁存器写入 1。

3. P2 口(P2.0～P2.7)

P2 口为双功能口，字节地址为 A0H，位地址为 A0H～A7H。P2 口某一位的位电路结构如图 1-12 所示。

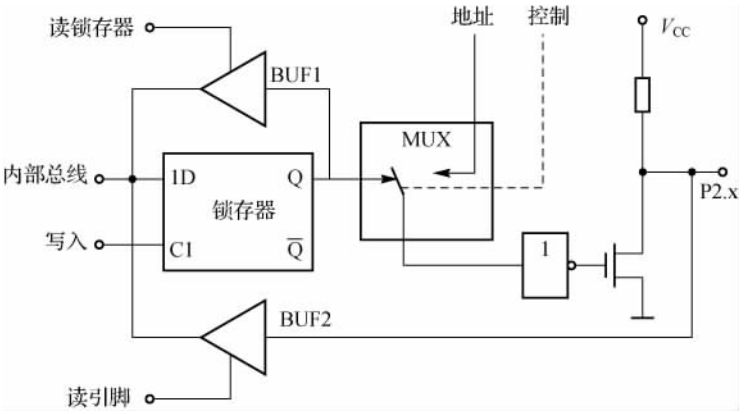


图 1-12　P2 口某一位的位电路结构

P2 口的工作过程分析如下。

1)P2 口用作地址总线

P2 口用作地址总线时，在控制信号作用下，MUX 与“地址”接通。当“地址”线为 0 时，场效应管导通，P2 口引脚输出为 0；当“地址”线为 1 时，场效应管截止，P2 口引脚输出 1。

2)P2 口用作通用 I/O 端口

P2 口用作通用 I/O 端口时，在内部控制信号作用下，MUX 与锁存器的 Q 端接通。CPU 输出 1 时，Q＝1，场效应管截止，P2.x 引脚输出 1；CPU 输出 0 时，Q＝0，场效应管导通，P2.x 引脚输出 0。P2 口用作输入时，分“读锁存器”和“读引脚”两种方式：“读锁存器”时，Q 端信号经输入缓冲器 BUF1 进入内部总线；“读引脚”时，先向锁存器写 1，使场效应管截止，P2.x 引脚上的电平经输入缓冲器 BUF2 进入内部总线。

3)P2 口的特点

P2 口作为地址输出线时，P2 口高 8 位地址与 P0 口输出的低 8 位地址寻址 64 KB 地址空间。作为通用 I/O 端口时，P2 口为准双向口。功能与 P1 口一样。一般情况下，P2 口大多作为高 8 位地址总线口使用，这时就不能再作为通用 I/O 端口使用了。

4. P3 口(P3.0～P3.7)

由于引脚数目有限，在 P3 口增加了第二功能。每一位都可以分别定义为第二输入功能或第二输出功能。P3 口字节地址为 B0H，位地址 B0H～B7H。P3 口某一位的位电路结构如图 1-13 所示。

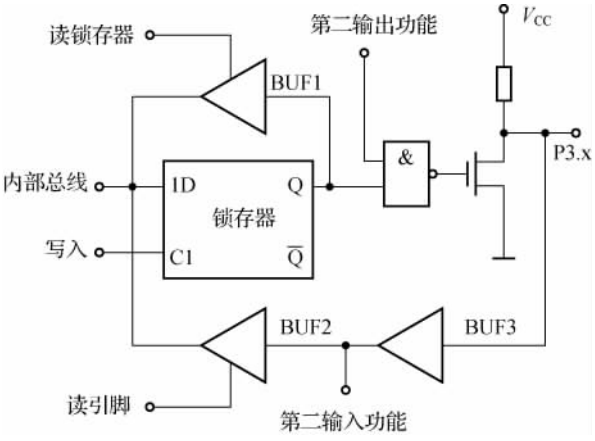


图 1-13　P3 口某一位的位电路结构

P3 口的工作过程分析如下。

1)P3 口用作第二输入/输出功能

当选择第二输出功能时,该位的锁存器需要置 1,K　非门为开启状态。当第二输出为 1 时,场效应管截止,P3.x 引脚输出为 1;当第二输出为 0 时,场效应管导通,P3.x 引脚输出为 0。

当选择第二输入功能时,该位的锁存器和第二输出功能端均应置 1,保证场效应管截止,P3.x 引脚的信息由输入缓冲器 BUF3 的输出获得。

2)P3 口用作第一功能——通用 I/O 端口

P3 口用作第一功能通用输出时,第二输出功能端应保持高电平,与非门开启。CPU 输出 1 时,Q=1,场效应管截止,P3.x 引脚输出为 1;CPU 输出 0 时,Q=0,场效应管导通,P3.x 引脚输出为 0。用作第一功能通用输入时,P3.x 位的输出锁存器和第二输出功能均应置 1,场效应管截止,P3.x 引脚信息通过输入 BUF3 和 BUF2 进入内部总线,完成“读引脚”操作。

当 P3 口用作第一功能通用输入时,也可执行“读锁存器”操作,此时信息由 Q 端进入并经过缓冲器 BUF1 进入内部总线。

3)P3 口的特点

P3 口内部有上拉电阻,无高阻抗输入态为准双向口。P3 口作为第二功能的输出/输入或第一功能通用输入,均须将相应位的锁存器置 1。实际应用中,由于复位后 P3 口锁存器自动置 1,满足第二功能所需的条件,所以不需任何设置工作就可以进入第二功能操作。当某位不作为第二功能用时,可作为第一功能通用 I/O 使用。引脚输入部分有两个缓冲器,第二功能的输入信号取自缓冲器 BUF3 的输出端,第一功能的输入信号取自缓冲器 BUF2 的输出端。

P3 口的第二功能定义见表 1-8。

表 1-8　P3 口的第二功能定义

引　脚	第二功能	功能说明
P3.0	RXD	串行数据输入口
P3.1	TXD	串行数据输出口
P3.2	$\overline{\text{INT0}}$	外部终端 0 输入
P3.3	$\overline{\text{INT1}}$	外部终端 1 输入
P3.4	T0	定时器 0 外部计数输入
P3.5	T1	定时器 1 外部计数输入
P3.6	$\overline{\text{WR}}$	外部数据存储器写选通输出
P3.7	$\overline{\text{RD}}$	外部数据存储器读选通输出

综上所述，P0 口可作为总线口，为双向口。作为通用的 I/O 端口使用时，为准双向口，这时需加上拉电阻。P1 口、P2 口、P3 口均为准双向口。

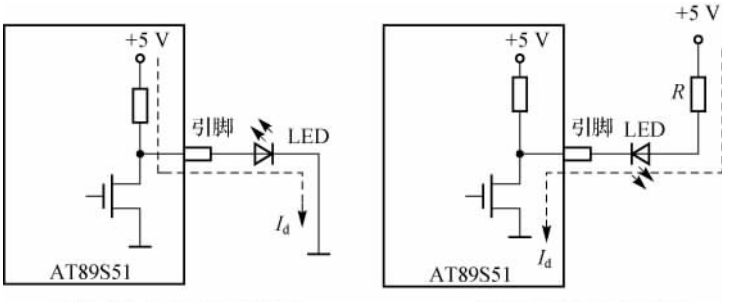
注意　准双向口仅有两个状态，而 P0 口作为总线使用，口线内没有上拉电阻，处于高阻“悬浮”态，故 P0 口为双向三态 I/O 端口。另外，准双向口作通用 I/O 的输入口使用时，一定要向该口先写入“1”。

5. I/O 端口的使用

下面讨论 P1～P3 口与 LED 发光二极管的驱动连接问题。P0 口与 P1、P2、P3 口相比，P0 口的驱动能力较大，每位可驱动 8 个 LSTTL 输入，而 P1、P2、P3 口的每一位的驱动能力只有 P0 口的一半。当 P0 口某位为高电平时，可提供 400 μA 的电流；当 P0 口某位为低电平(0.45 V)时，可提供 3.2 mA 的灌电流。如低电平允许提高，灌电流可相应加大，所以任何一个口要想获得较大的驱动能力，只能用低电平输出。

例如，使用单片机的并行口 P1～P3 直接驱动发光二极管，其电路如图 1-14 所示。P1～P3 内部有约 30 kΩ 的上拉电阻。

如高电平输出，则强行从 P1、P2 和 P3 口出的电流 I_d 会造成单片机端口的损坏，如图 1-14(a)所示。如端口引脚为低电平，能使电流 I_d 从单片机外部流入内部，则将大大增加流过的电流值，如图 1-14(b)所示。所以当 P1～P3 口驱动 LED 发光二极管时，应该采用低电平驱动。



(a)不恰当的连接：高电平驱动　　(b)恰当的连接：低电平驱动

图 1-14　发光二极管与 AT89S51 并行口的直接连接

五、单片机编程语言

语言是人类最重要的交际工具，是人们进行沟通交流的各种表达符号。人与计算机之间如果想进行交流，也要有一套独特的方式，即计算机语言。计算机语言分高级语言、汇编语言和机器语言，其中机器语言由二进制数(或十六进制数)构成，计算机(包括单片机)可以直接识别；而高级语言和汇编语言必须通过编译软件编译成机器语言后计算机才能识别。单片机一般采用汇编语言进行编程，因为汇编语言编译效率高，非常适合程序直接控制硬件的场合。由于汇编语言程序要安排运算或控制每一个细节，这就使得编写汇编语言程序比较繁琐复杂，现在也常用高级语言中的 C 语言进行单片机程序编写。

(一)单片机指令概述

在单片机系统中，我们把这种与计算机之间交流的语言称之为指令，指令是 CPU 按照人们的意图来完成某种操作的命令，它以英文名称或缩写形式作为助记符。指令不止一条，AT89S51 单片机使用 MCS-51 指令系统。掌握汇编语言指令是程序设计的基础。下面对指令系统进行简单的介绍。

AT89S51 拥有简明、易掌握、效率较高的指令系统——复杂指令集。指令按所占字节来分有：单字节指令 49 条，双字节指令 45 条，三字节指令 17 条。指令按执行时间来分有：1 个机器周期(12 个时钟振荡周期)的指令 64 条，2 个机器周期的指令 45 条，4 个机器周期——乘、除指令。

对于振荡频率为 12 MHz 的晶体振荡器，每个机器周期为 1 μs。AT89S51 的一大特点是在硬件结构中有一个位处理机，即一个处理位变量的指令子集。

下面来介绍单片机指令中的一些重要的概念。

指令格式：指令的表示方法。

指令通常由两部分组成：操作码和操作数。

操作码：指令进行的操作。

操作数：指令操作的对象，可能是一具体数据，也可能是指出到哪里取得数据的地址或符号。

指令长度不同，格式也就不同，具体如下。

①单字节指令：操作码和操作数同在一个字节中。

②双字节指令：一个字节为操作码，另一个字节是操作数。

③三字节指令：操作码占一个字节，操作数占两个字节。

1. 单片机寻址方式

寻址方式是指在指令中说明操作数所在地址的方法。一般来说，寻址方式越多，功能就越强，灵活性则越大，指令系统就越复杂。寻址方式所要解决的主要问题就是如何在整个存储器和寄存器的寻址空间内快速地找到指定的地址单元。下面介绍指令系统 7 种寻址方式。

1)寄存器寻址方式

指令中的操作数为某一寄存器的内容。例如，`MOV A,Rn ;(Rn)→A,n=0～7` 表示把 Rn 中的源操作数送入到累加器 A 中。由于指令指定了从寄存器 Rn 中取得源操作数，所以称为寄存器寻址方式。

2)直接寻址方式

指令中直接给出操作数的单元地址,该单元地址中的内容就是操作数,直接的操作数单元地址用“direct”表示。例如,MOV A,direct,“direct”为操作数的单元地址。具体指令为MOV A,40H,表示把内部 RAM 40H 单元(direct)的内容传送到 A。指令中源操作数(右边的操作数)采用的是直接寻址方式。

指令中两个操作数都可由直接寻址方式给出。例如,MOV direct1,direct2,具体指令为 MOV 42H,62H,表示把片内 RAM 中 62H 单元的内容送到片内 RAM 中的 42H 单元中。

直接寻址是访问片内所有特殊功能寄存器的唯一寻址方式。

3)寄存器间接寻址方式

寄存器中存的是操作数地址,即先从寄存器中找到操作数的地址,再按该地址找到操作数。

为了区别寄存器寻址和寄存器间接寻址,在寄存器间接寻址方式中,应在寄存器名称前面加前缀标志“@”。例如,MOV A,@Ri ;i=0 或 1。其中 Ri 的内容为 40H,即把内部 RAM 40H 地址单元中的内容传送给 A。

4)立即数寻址方式

直接在指令中给出操作数,也称立即数。为了与直接寻址指令中的直接地址加以区别,需在操作数前加前缀标志“#”。例如,MOV A,#40H。第一个字节是操作码,第二字节是立即数,就是放在程序存储器内的常数。

5)基址寄存器加变址寄存器间接寻址方式

以 DPTR 或 PC 作为基址寄存器,以累加器 A 作为变址寄存器,以两者内容相加形成的 16 位地址作为目的地址进行寻址。例如,MOVC A,@A+DPTR。其中,(A)=05H,(DPTR)=0400H,指令执行结果是把程序存储器 0405H 单元的内容传送给 A。

本寻址方式的指令有以下 3 条。

```
MOVC A,@A+DPTR
MOVC A,@A+PC
JMP A,@A+DPTR
```

前两条指令适用于读程序存储器中固定的数据。例如,将固定的、按一定顺序排列的表格存放在程序存储器中,在程序运行中由 A 的动态参量来确定读取对应的表格参数。

第 3 条为散转指令,A 中内容为程序运行后的动态结果,可根据 A 中不同内容实现跳向不同程序入口的跳转。

6)相对寻址方式

相对寻址是以该转移指令的地址(PC 值)加上它的字节数,再加上相对偏移量(rel),形成新的转移目的地址,从而将程序转移到该目的地址。转移的目的地址用下式计算。

目的地址=转移指令所在的地址+转移指令字节数+rel

其中,偏移量 rel 是带符号 8 位二进制补码数,其值为-128~+127。

程序转移范围是以转移指令的下条指令首地址为基准地址,相对偏移为-128~+127。例如,LJMP rel。

程序要转移到该指令的 PC 值加 3 再加上 rel 的目的地址处。编写程序时,只须在转移指令中直接写要转向的地址标号即可。例如,LJMP LOOP,“LOOP”为目的地址标号。汇编时,由汇编程序自动计算和填入偏移量,但手工汇编时偏移量的值由手工计算。

7)位寻址方式

对内部 RAM 和特殊功能寄存器具有位寻址功能的某位内容进行置 1 和清零操作。位地址一般以直接位地址给出,位地址符号为“bit”。例如,MOV C,bit,具体指令为 MOV C,40H,表示把位地址为 40H 的值送到进位标志位 C。

由于 AT89S51 具有位处理功能,可直接对数据位方便地实现置 1、清零、求反、传送、判跳和逻辑运算等操作,为测控系统的应用提供了最佳代码和速度,增强了实时性。

上面介绍了 7 种寻址方式,现有一个问题:当一条指令给定后如何来确定该指令的寻址方式?例如,MOV A,#40H 属于立即数寻址还是寄存器寻址?这要看以哪个操作数作为参照系。操作数分为源操作数和目的操作数,对于源操作数“#40H”来说,是“立即数寻址”方式,但对目的操作数“A”来说,属于“寄存器寻址”方式。一般而言,寻址方式指的是源操作数,所以此例为立即数寻址方式。指令系统中的 7 种寻址方式及其寻址空间见表 1-9。

表 1-9　7 种寻址方式及其寻址空间

序　号	寻址方式	寻址空间
1	寄存器寻址	R0～R7、A、B、C(位)、DPTR 等
2	直接寻址	内部 128 字节 RAM、特殊功能寄存器
3	寄存器间接寻址	片内数据存储器、片外数据存储器
4	立即数寻址	程序存储器中的立即数
5	基址寄存器加变址寄存器间接寻址	读程序存储器固定数据和程序散转
6	相对寻址	程序存储器相对转移
7	位寻址	内部 RAM 中的可寻址位、SFR 中的可寻址位

2. 单片机指令分类

AT89S51 共 111 条指令,按功能分为 5 类:数据传送类(28 条)、算术运算类(24 条)、逻辑操作类(25 条)、控制转移类(17 条)、位操作类(17 条)。指令用到的符号见表 1-10。

表 1-10　指令用到的符号

符　号	含　义
Rn	当前寄存器区的 8 个工作寄存器 R0～R7(n=0～7)
Ri	当前寄存器区中作为间接寻址寄存器的 2 个寄存器 R0、R1(i=0,1)
direct	直接地址,即 8 位内部数据存储器单元或特殊功能寄存器的地址
#data	指令中的 8 位立即数
#data16	指令中的 16 位立即数
rel	偏移量,8 位的带符号补码数
DPTR	数据指针,可用作 16 位数据存储器单元地址的寄存器

续表

符　号	含　义
bit	内部 RAM 或特殊功能寄存器中的直接寻址位
C 或 CY	进位标志位或位处理机中的累加器
addr11	11 位目的地址
addr16	16 位目的地址
@	间接寻址寄存器前缀，如@Ri，@A＋DPTR
(x)	表示 x 地址单元或寄存器中的内容
((x))	表示以 x 单元或寄存器中的内容作为间接寻址单元的内容
→	箭头右边的内容被箭头左边的内容所取代

(二)数据传送指令

数据传送类指令使用最频繁，一般数据传送类指令的助记符为“MOV”，通用格式如下。

MOV ＜目的操作数＞，＜源操作数＞

数据传送类指令是把源操作数传送到目的操作数。指令执行之后，源操作数不改变，目的操作数修改为源操作数。所以此类指令虽说称为“传送”类操作却属“复制”性质，而不是“搬家”。本类指令不影响标志位 CY、AC 和 OV，但不包括奇偶标志位 P。

1. 以累加器为目的操作数的指令

以累加器为目的操作数的指令如下。

```
MOV A,Rn              ;(Rn)→A,n=0～7
MOV A,@ Ri            ;((Ri))→A,i=0,1
MOV A,direct          ;(direct)→A
MOV A,#data           ;#data→A
```

把源操作数内容送累加器 A，源操作数有寄存器寻址、直接寻址、间接寻址和立即数寻址等方式，例如，

```
MOV A,R6              ;(R6)→A,寄存器寻址
MOV A,@R0             ;((R0))→A,间接寻址
MOV A,70H             ;(70H)→A,直接寻址
MOV A,#78H            ;78H→A,立即数寻址
```

2. 以 Rn 为目的操作数的指令

以 Rn 为目的操作数的指令如下。

```
MOV Rn ,A             ;(A)→Rn ,n=0～7
MOV Rn ,direct        ;(direct)→Rn ,n=0～7
MOV Rn ,#data         ;#data→Rn ,n=0～7
```

把源操作数送入当前寄存器区的 R0～R7 中的某一寄存器。

3. 以直接地址 direct 为目的操作数的指令

以直接地址 direct 为目的操作数的指令如下。

```
MOV direct,A          ;(A)→direct
```

```
MOV direct,Rn          ;(Rn)→direct,n=0～7
MOV direct1,direct2    ;(direct2)→direct1
MOV direct,@Ri         ;((Ri))→direct,i=0,1
MOV direct,#data       ;#data→direct
```

把源操作数送入直接地址指定的存储单元。direct 指的是内部 RAM 或 SFR 地址。

4. 以寄存器间接地址为目的操作数的指令

以寄存器间接地址为目的操作数的指令如下。

```
MOV  @Ri,A             ;(A)→((Ri)), i=0,1
MOV  @Ri,direct        ;(direct)→((Ri)),i=0,1
MOV  @Ri,#data         ;#data→((Ri)), i=0,1
```

把源操作数内容送入 R0 或 R1 指定的存储单元中。

5. 16 位数传送指令

16 位数传送指令如下。

```
MOV DPTR,#data16       ;#data16→DPTR
```

把 16 位立即数送入 DPTR,用来设置数据存储器的地址指针。例如,

```
MOV DPTR,#1234H
```

执行完了之后 DPH 中的值为 12H,DPL 中的值为 34H。

如果我们分别向 DPH 和 DPL 送数,则结果也一样。如下面两条指令

```
MOV DPH,#12H
MOV DPL,#34H
```

相当于执行了

```
MOV DPTR,#1234H
```

AT89S51 有两个 DPTR,通过设置特殊功能寄存器 AUXR1 中的 DPS 位来选择。当 DPS=1 时,指令中的 DPTR 为 DPTR1,DPTR0 被屏蔽,反之亦然。

DPTR 为 16 位的数据指针,分为 DPH 和 DPL,操作十分灵活方便。设有两个 DPTR 后,就可避免频繁地出入堆栈操作。

课堂练习:写出下列每条指令的寻址方式以及执行后的结果。

```
MOV  23H,#30H
MOV  12H,#34H
MOV  R0,#23H
MOV  R7,12H
MOV  R1,#12H
MOV  A,@R0
MOV  34H,@R1
MOV  45H,34H
MOV  DPTR,#6712H
```

```
MOV  12H,DPH
MOV  R0,DPL
MOV  A,@R0
```

6. 堆栈操作指令

内部 RAM 中设定一个后进先出(last in first out,LIFO)的区域,称为堆栈。在特殊功能寄存器中有一个堆栈指针 SP,指示堆栈的栈顶位置。堆栈操作有进栈和出栈两种,因此,在指令系统中相应有两条堆栈操作指令。

1)进栈指令 PUSH direct

首先将栈指针 SP 加 1,然后把 direct 中的内容送到 SP 指示的内部 RAM 单元中。

例如,当(SP)=60H,(A)=30H,(B)=70H 时,执行下列指令。

```
PUSH ACC          ;(SP)+1=61H→SP,(A)→61H
PUSH B            ;(SP)+1=62H→SP,(B)→62H
```

结果为(61H)=30H,(62H)=70H,(SP)=62H。

2)出栈指令 POP direct

将 SP 指示的栈顶单元的内容送入 direct 字节中,然后 SP 减 1。

例如,当(SP)=62H,(62H)=70H,(61H)=30H 时,执行下列指令。

```
POP DPH           ;((SP))→DPH,(SP)-1→SP
POP DPL           ;((SP))→DPL,(SP)-1→SP
```

结果为(DPTR)=7030H,(SP)=60H。

7. 累加器 A 与外部数据存储器 RAM/IO 传送指令

累加器 A 与外部数据存储器 RAM/IO 传送指令如下。

```
MOVX A,@DPTR      ;((DPTR))→A,读外部 RAM/IO
MOVX A,@Ri        ;((Ri))→A,读外部 RAM/IO
MOVX  @DPTR,A     ;(A)→((DPTR)),写外部 RAM/IO
MOVX  @Ri,A       ;(A)→((Ri)),写外部 RAM/IO
```

MOV 的后面加“X”,表示访问的是片外 RAM 或 I/O 端口。执行前两条指令时,$\overline{RD}$(P3.7)有效;执行后两条指令时,$\overline{WR}$(P3.6)有效。

8. 查表指令

查表指令是仅有的两条读程序存储器中表格数据的指令。由于程序存储器只读不写,因此传送为单向,从程序存储器中读出数据到 A 中。两条查表指令均采用基址寄存器加变址寄存器间接寻址方式。

1)MOVC A,@A+PC

以 PC 作为基址寄存器,A 的内容(无符号数)和 PC 的当前值(下一条指令的起始地址)相加后得到一个新的 16 位地址,把该地址的内容送到 A。例如,当(A)=30H 时,执行地址 1000H 处的指令。

```
1000H:MOVC A,@A+PC
```

该指令占用一个字节,下一条指令的地址为 1001H,(PC)=1001H 再加上 A 中的 30H,得 1031H,结果把程序存储器中 1031H 的内容送入累加器 A。

该指令的优点是不改变特殊功能寄存器及 PC 的状态,根据 A 的内容就可以取出表格中的常数;缺点是表格只能存放在该条查表指令所在地址的+256 个单元之内,表格大小受到限制,且表格只能被一段程序所用。

2)MOVC A,@A+DPTR

DPTR 为基址寄存器,A 的内容(无符号数)和 DPTR 的内容相加得到一个 16 位地址,把由该地址指定的程序存储器单元的内容送到累加器 A。例如,(DPTR)=8100H,(A)=40H,执行下列指令。

```
MOVC A,@A+DPTR
```

将程序存储器中 8140H 单元内容送入 A 中。

本指令执行结果只与指针 DPTR 及累加器 A 的内容有关,与该指令存放的地址及常数表格存放的地址无关,因此表格的大小和位置可以在 64 KB 程序存储器空间中任意安排,一个表格可以为各个程序块公用。两条指令的助记符都是在 MOV 的后面加“C”,是 CODE 的第一个字母,即表示程序存储器中的代码。执行上述两条指令时,单片机的$\overline{\text{PSEN}}$引脚信号(程序存储器)有效,这一点读者要牢记。

9. 字节交换指令

字节交换指令如下。

```
XCH A,Rn              ;(A)↔(Rn),n=0～7
XCH A,direct          ;(A)↔(direct)
XCH A,@Ri             ;(A)↔(( Ri)),i=0,1
```

这组指令的功能是将累加器 A 的内容和源操作数的内容相互交换。源操作数有寄存器寻址、直接寻址和寄存器间接寻址等方式。例如,(A)=80H,(R7)=08H,(40H)=F0H,(R0)=30H,(30H)=0FH,执行下列指令。

```
XCH A,R7              ;(A)↔(R7)
XCH A,40H             ;(A)↔(40H)
XCH A,@R0             ;(A)↔((R0))
```

结果为(A)=0FH,(R7)=80H,(40H)=08H,(30H)=F0H。

10. 低半字节交换指令

低半字节交换指令格式如下。

```
XCHD A,@Ri
```

累加器的低 4 位与内部 RAM 低 4 位交换。例如,(R0)=60H,(60H)=3EH,(A)=59H,执行完“XCHD A,@R0”指令,则(A)=5EH,(60H)=39H。

(三)算术运算指令

指令系统中,有单字节的加、减、乘、除法指令,算术运算功能比较强。算术运算指令都是针对 8 位二进制无符号数的,如要进行带符号或多字节二进制数运算,需编写具体的运算程序,通过执行程序实现。算术运算的结果将使 PSW 的进位(CY)、辅助进位(AC)、溢出(OV)3 种标志位置 1 或清零。但增 1 和减 1 指令不影响这些标志。

1. 不带进位加法指令

不带进位加法指令有以下 4 条指令。

```
ADD A,Rn              ;(A)+(Rn)→A,n=0～7
```

```
ADD A,direct        ;(A)+(direct)→A
ADD A,@Ri           ;(A)+((Ri))→A,i=0,1
ADD A,#data         ;(A)+#data→A
```

8 位加法指令的一个加数总是来自累加器 A，而另一个加数可由寄存器寻址、直接寻址、寄存器间接寻址和立即数寻址等不同的寻址方式得到。加的结果总是放在累加器 A 中。使用本指令时，要注意累加器 A 中的运算结果对各个标志位的影响。

2. 带进位加法指令

带进位加法指令的特点是进位标志位 CY 参加运算，3 个数相加。其 4 条指令如下。

```
ADDC A,Rn           ;(A)+(Rn)+C→A,n=0～7
ADDC A,direct       ;(A)+(direct)+C→A
ADDC A,@Ri          ;(A)+((Ri))+C→A,i=0,1
ADDC A,#data        ;(A)+#data+C→A
```

如果位 7 有进位，则进位标志 CY 置 1，否则 CY 清零；如果位 3 有进位，则辅助进位标志 AC 置 1，否则 AC 清零；如果位 6 有进位而位 7 没有进位，或者位 7 有进位而位 6 没有进位，则溢出标志 OV 置 1，否则标志 OV 清零。

3. 增 1 指令

增 1 指令有以下 5 条指令。

```
INC A
INC Rn              ;n=0～7
INC direct
INC  @Ri            ;i=0,1
INC DPTR
```

把指令中所指出的变量增 1，且不影响 PSW 中的任何标志。指令“INC DPTR”为 16 位数增 1 指令。首先对低 8 位指针 DPL 执行加 1，当溢出时，就对 DPH 的内容进行加 1，不影响进位标志 CY。

4. 十进制调整指令

用于对 BCD 码加法运算结果的内容修正，指令格式如下。

```
DA  A
```

十进制调整指令是对压缩的 BCD 码（一个字节存放 2 位 BCD 码）的加法结果进行十进制调整。两个 BCD 码按二进制相加之后，必须经本指令的调整才能得到正确的压缩 BCD 码的和数。

十进制调整指令的用法详见学习任务四中秒表的制作的知识链接部分。

5. 带借位的减法指令

带借位的减法指令有以下 4 条指令。

```
SUBB A,Rn           ;(A)-(Rn)-CY→A,n=0～7
SUBB A,direct       ;(A)-(direct)-CY→A
SUBB A,@Ri          ;(A)-((Ri))-CY→A, i=0,1
SUBB A,#data        ;(A)-#data-CY→A
```

从 A 的内容减去指定变量和进位标志 CY 的值，结果存在 A 中。如果位 7 需借位则

CY 置 1,否则 CY 清零;如果位 3 需借位则 AC 置 1,否则 AC 清零;如果位 6 借位而位 7 不借位,或者位 7 借位而位 6 不借位,则溢出标志位 OV 置 1,否则 OV 清零。

6. 减 1 指令

减 1 指令有以下 4 条指令。

```
DEC A                ;(A)－1→A
DEC Rn               ;(Rn)－1→Rn,n＝0～7
DEC direct           ;(direct)－1→direct
DEC  @Ri             ;((Ri))－1→(Ri),i＝0,1
```

减 1 指令的功能是指定的变量减 1。若原来为 00H,减 1 后下溢为 FFH,不影响标志位(P 标志除外)。

7. 乘法指令

乘法指令格式如下。

```
MUL AB               ;A×B→BA
```

乘法积的低字节在累加器 A 中,高字节在 B 中。如果积大于 255,则 OV 置 1,否则 OV 清 0。CY 标志总是清零。

8. 除法指令

除法指令格式如下。

```
DIV AB               ;A/B→A(商),余数→B
```

商(为整数)存放在 A 中,余数存放在 B 中,且 CY 和溢出标志位 OV 清零。如果 B 的内容为 0(即除数为 0),则存放结果的 A、B 中的内容不定,并溢出标志位 OV 置 1。例如,(A)＝FBH,(B)＝12H,执行以下指令。

```
DIV AB
```

结果为(A)＝0DH,(B)＝11H,CY＝0,OV＝0。

(四)逻辑操作指令

1. 累加器 A 清零指令

累加器 A 清零指令格式如下。

```
CLR A
```

累加器 A 清零,不影响 CY、AC、OV 等标志位。

2. 累加器 A 求反指令

累加器 A 求反指令格式如下。

```
CPL A
```

将累加器 A 的内容按位逻辑取反,不影响标志位。

3. 左环移指令

左环移指令格式如下。

```
RL A
```

左环移指令的功能是 A 向左循环移位,位 7 循环移入位 0,不影响标志位,如图 1-15 所示。

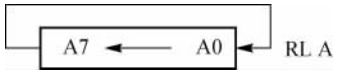


图 1-15　左环移操作

4. 带进位左环移指令

带进位左环移指令格式如下。

```
RLC A
```

带进位左环移指令的功能是将累加器 A 的内容和进位标志位 CY 一起向左环移一位，如图 1-16 所示。

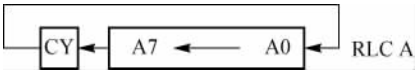


图 1-16　带进位左环移操作

5. 右环移指令

右环移指令格式如下。

```
RR A
```

右环移指令的功能是 A 的内容向右环移一位不影响其他标志位，如图 1-17 所示。

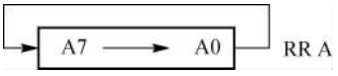


图 1-17　右环移操作

6. 带进位右环移指令

带进位右环移指令格式如下。

```
RRC A
```

带进位右环移指令的功能是将 A 的内容和进位标志 CY 一起向右环移一位，如图 1-18 所示。

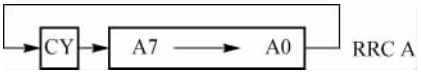


图 1-18　带进位右环移操作

7. 累加器半字节交换指令

累加器半字节交换指令格式如下。

```
SWAP A
```

累加器半字节交换指令的功能是将累加器 A 的高半字节(ACC.7～ACC.4)和低半字节(ACC.3～ACC.0)互换。例如，(A)= 95H，执行以下指令。

```
SWAP A
```

结果为 (A)=59H。

8. 逻辑与指令

逻辑与指令如下。

```
ANL A,Rn              ;(A)^(Rn)→A,n=0～7
ANL A,direct          ;(A)^(direct)→A
ANL A,#data           ;(A)^#data→A
ANL A,@Ri             ;(A)^((Ri))→A,i=0～1
ANL direct,A          ;(direct)^(A)→direct
```

```
ANL direct,#data        ;(direct)^#data→direct
```

逻辑与指令的功能是在指定的变量之间以位为基础进行“逻辑与”操作，结果存放到目的变量所在的寄存器或存储器中。例如，(A)=07H，(R0)=0FDH，执行以下指令。

```
ANL A,R0
```

运算式为

$$\begin{array}{cccccccc} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ \wedge & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ \hline & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}$$

结果为(A)=05H。

9. 逻辑或指令

逻辑或指令如下。

```
ORL A,Rn                ;(A)∨(Rn)→A ,n=0～7
ORL A,direct            ;(A)∨(direct)→A
ORL A,#data             ;(A)∨#data→A
ORL A,@Ri               ;(A)∨((Ri))→A,i=0,1
ORL direct,A            ;(direct)∨(A)→direct
ORL direct,#data        ;(direct)∨#data→direct
```

逻辑或指令的功能是在所指定的变量之间执行位的“逻辑或”操作，结果存到目的变量寄存器或存储器中。

10. 逻辑异或指令

逻辑异或指令如下。

```
XRL A,Rn                ;(A)⊕(Rn)→A,n=0～7
XRL A,direct            ;(A)⊕(direct)→A
XRL A,@Ri               ;(A)⊕((Ri))→A,i=0,1
XRL A,#data             ;(A)⊕#data→A
XRL direct,A            ;(direct)⊕(A)→direct
XRL direct,#data        ;(direct)⊕#data→direct
```

逻辑异或指令的功能是在所指定的变量之间执行以位的“逻辑异或”操作，结果存到目的变量寄存器或存储器中。

(五)跳转与循环指令

1. 长转移指令

长转移指令格式如下。

```
LJMP addr16
```

执行指令时，把转移的目的地址，即指令的第 2 和第 3 字节分别装入 PC 的高位和低位字节中，无条件地转向 addr16 指定的目的地址——64 KB 程序存储器地址空间的任何位置。

2. 相对转移指令

相对转移指令格式如下。

```
SJMP rel
```

无条件转移，rel 为相对偏移量，是一个单字节的带符号 8 位二进制补码数，因此程序转

移是双向的。rel 如为正，向地址增大的方向转移；rel 如为负，向地址减小的方向转移。

执行时，在 PC 加 2(本指令为 2 B)之后，把指令的有符号的偏移量 rel 加到 PC 上，并计算出目的地址。

编程时，只须写上目的地址标号，相对偏移量由汇编程序自动计算。例如，

```
LOOP:MOV A,R6
     ……
     SJMP LOOP
     ……
```

汇编时，跳到 LOOP 处的偏移量由汇编程序自动计算和填入。

3. 绝对转移指令

绝对转移指令格式如下。

```
AJMP addr11
```

绝对转移指令提供 11 位地址 A10～A0(即 addr11)，其中 A10～A8 位于第 1 字节的高 3 位，A7～A0 在第 2 字节。操作码只占第 1 字节的低 5 位。指令构造转移目的地址：执行本指令，PC 加 2，然后把指令中的 11 位无符号整数地址 addr11(A10～A0)送入“PC. 10～PC. 0”，“PC. 15～PC. 11”保持不变，形成新的 16 位转移目的地址。

需要注意的是，目标地址必须与 AJMP 指令的下一条指令首地址的高 5 位地址码 A15～A11 相同，否则将混乱。所以，绝对转移指令是 2 KB 范围内的无条件跳转指令。

4. 间接跳转指令

间接跳转指令格式如下。

```
JMP  @A+DPTR
```

间接跳转指令为单字节转移指令，目的地址由 A 中 8 位无符号数与 DPTR 的 16 位无符号数内容之和来确定。以 DPTR 内容为基址，A 的内容作为变址。给 A 赋予不同值，即可实现多分支转移。

5. 条件转移指令

执行条件转移指令时，如条件满足，则转移；不满足，则顺序执行下一指令。转移目的地址在以下一条指令首地址为中心的 256 B 范围内(－128～＋127)。

```
JZ rel          ;如果累加器内容为 0,则执行转移
JNZ rel         ;如果累加器内容非 0,则执行转移
```

6. 比较不相等转移指令

比较不相等转移指令如下。

```
CJNE A,direct,rel
CJNE A,#data,rel
CJNE Rn,#data,rel
CJNE  @Ri,#data,rel
```

比较前两个操作数大小，如果值不相等则转移，并转向目的地址。

注意　CJNE 指令只是比较两个数的大小,不是做真正的减法。也就是说,不真正保留两数的差值,所以也把 CJNE 称为逻辑减法,这与减法指令 SUBB 是不同的。虽不真减,但是却会影响 PSW 中的 CY 位。如果第一操作数(无符号整数)小于第二操作数(无符号整数),则进位标志位 CY 置 1,否则 CY 清零。该指令的执行不影响任何一个操作数的内容。

7. 减 1 不为 0 转移指令

减 1 不为 0 转移指令把减 1 与条件转移两种功能合在一起。此种指令有两种形式。

```
DJNZ Rn,rel    ;n=0～7
DJNZ direct,rel
```

此指令用于控制程序循环。预先装入循环次数,以减 1 后是否为 0 作为转移条件,即实现按次数控制循环。

8. 调用子程序指令

1)长调用指令

长调用指令格式如下。

```
LCALL addr16
```

长调用指令可调用 64 KB 范围内程序存储器中的任何一个子程序。执行时,先把 PC 加 3 获得下一条指令的地址(断点地址),并压入堆栈(先低位字节,后高位字节),堆栈指针加 2。

接着把指令的第 2 和第 3 字节(A15～A8,A7～A0)分别装入 PC 的高位和低位字节中,然后从 PC 指定的地址开始执行程序。执行后不影响任何标志位。

2)绝对调用指令

绝对调用指令格式如下。

```
ACALL addr11
```

与 AJMP 指令类似,绝对调用指令为 2 KB 范围内的调用子程序的指令。子程序地址必须与 ACALL 指令下一条指令的 16 位首地址中的高 5 位地址相同,否则将混乱。

9. 子程序的返回指令

子程序的返回指令格式如下。

```
RET
```

执行本指令时,

(SP)→PCH,然后(SP)－1→SP

(SP)→PCL,然后(SP)－1→SP

其功能是从堆栈中退出 PC 的高 8 位和低 8 位字节,把栈指针减 2,从 PC 值处开始继续执行程序,不影响任何标志位。

10. 中断返回指令

中断返回指令格式如下。

```
RETI
```

与 RET 指令相似,不同之处为该指令清除了中断响应时被置 1 的内部中断优先级寄存器的中断优先级状态。

11. 空操作指令

空操作指令格式如下。

```
NOP
```

空操作指令不进行任何操作，耗一个机器周期时间，执行(PC)＋1→PC 操作。

(六)布尔指令(位操作指令)

单片机内部有一个位处理机，其对应位操作指令。

1. 数据位传送指令

数据位传送指令格式如下。

```
MOV C,bit
MOV bit,C
```

数据位传送指令把源操作数指定的位变量送到目的操作数指定处。一个操作数必须为进位标志，另一个可以是任何直接寻址位，不影响其他寄存器或标志位。

例如

```
MOV C,06H          ;(20H).6→CY
```

06H 是位地址，20H 是内部 RAM 字节地址。06H 是内部 RAM 20H 字节位 6 的位地址。

2. 位变量修改指令

位变量修改指令格式如下。

```
CLR C              ;CY 位清零
CLR bit            ;bit 位清零
CPL C              ;CY 位求反
CPL bit            ;bit 位求反
SETB C             ;CY 位置 1
SETB bit           ;bit 位置 1
```

这组指令将操作数指定的位清零、求反、置 1，不影响其他标志位。例如，

```
CLR C               ;CY 位清零
CLR 27H             ;0→(27H)位
CPL 08H             ;(08H)位取反
SETB P1.7           ;P1.7 位置 1
```

3. 位变量逻辑与指令

位变量逻辑与指令格式如下。

```
ANL C,bit           ;bit∧CY→CY
ANL C,/bit          ;/bit∧CY→CY
```

第 2 条指令先对直接寻址位求反，然后与进位标志位 CY 进行“逻辑与”运算，结果送回到位累加器中。

4. 位变量逻辑或指令

位变量逻辑或指令格式如下。

```
ORL C,bit
ORL C,/bit
```

第 1 条指令是直接寻址位与进位标志位 CY(位累加器)进行“逻辑或”运算,结果送回到进位标志位中。

第 2 条指令先对直接寻址位求反,然后与位累加器(进位标志位)进行“逻辑或”运算,结果送回到进位标志位中。

5. 条件转移类指令

条件转移指令格式如下。

```
JC rel              ;如进位标志位 CY=1,则转移
JNC rel             ;如进位标志位 CY=0,则转移
JB bit,rel          ;如直接寻址位=1,则转移
JNB bit,rel         ;如直接寻址位=0,则转移
JBC bit,rel         ;如直接寻址位=1,转移,并把寻址位清零
```

(七)单片机伪指令

单片机汇编语言程序设计中,除了使用指令系统规定的指令外,还要用到一些伪指令。伪指令又称为指示性指令,具有和指令类似的形式,但汇编时伪指令并不产生可执行的目标代码,只是对汇编过程进行某种控制或提供某些汇编信息。下面对常用的伪指令作简单介绍。

1. DS:预留存储区命令

格式:[标号:] DS　表达式值

其功能是从指定地址开始,定义一个存储区,以备源程序使用。

存储区预留的存储单元数由表达式的值决定。

```
TMP: DS  1
```

从标号 TMP 地址处开始保留 1 个存储单元(字节)。

2. BIT:定义位命令

格式:字符名称　BIT　位地址

其功能用于给字符名称定义位地址。例如,

```
SPK BIT P3.7
```

经定义后,允许在指令中用 SPK 代替 P3.7。

3. USING:通知汇编器使用的是哪一个工作寄存器组

格式:USING　表达式(值必须为 0～3,默认值为 0)。例如,

```
USING  0
```

使用第 0 组工作寄存器。

4. ORG:用来改变汇编器的计数器,从而设定一个新的程序起始地址

格式:ORG　表达式

表达式必须是绝对或简单再定位表达式。例如,

```
ORG  0000H
AJMP MAIN
```

设定 MAIN 程序的起始地址为 0000H。

5. END:用来控制汇编结束

在每个汇编程序的最后一行必须有一条 END 指令,并且 END 指令只能出现一次。

6. EQU：用于将一个数值或寄存器名赋给一个指定的符号名

格式：符号名 EQU 表达式

或　　符号名 EQU 寄存器名

表达式必须是一个简单再定位表达式。

用 EQU 指令赋值以后的字符名，可以用作数据地址、代码地址、位地址或者直接当作一个立即数使用。例如，

```
LIMIT EQU 1200
COUNT EQU R5
```

7. DATA：用于将一个内部 RAM 的地址赋给指定的符号名

格式：符号名　DATA　表达式

数值表达式的值为 0～255，表达式必须是一个简单再定位表达式。例如，

```
PORT1  DATA  40H
```

8. DB：以表达式的值的字节形式初始化代码空间

格式：[标号：]　DB　表达式表

表达式中可包含符号、字符串和表达式等项，各个项之间用逗号隔开，字符串应用引号括起来。括号内的标号是可选项，如果使用了标号，则标号的值将是表达式表中第 1 字节的地址。DB 指令必须位于 CODE 段之内，否则将会发生错误。例如，

```
TABLE:
DB  0C0H,0F9H,0A4H
TABLE1:
DB  "WEINA"
```

9. XDATA：数据地址赋值伪指令

格式：符号名　XDATA　表达式

将表达式的值或某个特定汇编符号定义为一个制定的符号名，可以先使用后定义，并且用于双字节数据定义。例如，

```
DELAY XDATA  0356H
LCALL DELAY                ;执行指令后，程序转到 0356H 单元执行
```

（八）关于 AT89S51 指令部分的说明

1. 关于操作数的字节地址和位地址的区分问题

如何区别指令中出现的字节变量和位变量？例如，指令“MOV C，40H”和指令“MOV A，40H”中源操作数“40H”都是以直接地址形式给出的，“40H”是字节地址还是位地址？对于助记符相同指令，观察操作数就可看出。显然前条指令中的“40H”肯定是位地址，因为目的操作数 C 是位变量；后条指令的“40H”是字节地址，因为目的操作数 A 是字节变量。

2. 关于累加器 A 与 ACC 的书写问题

累加器可写成 A 或 ACC，区别是什么？

ACC 汇编后的机器码必有一个字节的操作数是累加器的字节地址 E0H，A 汇编后则隐含在指令操作码中。例如，“INC A”的机器码，查表是 04H。如写成“INC ACC”后，则成了“INC direct”的格式，再查表，对应机器码为“05H E0H”。

在对累加器 A 直接寻址和累加器 A 的某一位寻址要用 ACC,不能写成 A。例如,指令“POP ACC”不能写成“POP A”,指令“SETB ACC.0”不能写成“SETB A.0”。

3. 书写 2 位十六进制数据前要加 0

经常遇到必须在某些数据或地址的前面多填一个前导 0,否则汇编就通不过。这是汇编语言的严格性和规范性的体现。

由于部分十六进制数是用字母来表示的,而程序内的标号也常用字母表示,为了将标号和数据区分开,几乎所有的汇编语言都规定,凡是以字母开头(对十六进制数而言,就是 A～F 开头)的数字量,应当在前面添加一个数字 0。至于地址量,它也是数据量的一种,前面也应该添加 0。例如,

```
MOV A,#0F0H        ;“F0”以字母开头的数据量
MOV A,0F0H         ;“F0”以字母开头的地址量
```

如不加前导 0,就会把字母开头的数据量当作标号来处理,从而出错而不能通过汇编。

任务　单片机功能体验——LED 灯的“眨眼”节奏控制

工作内容及要求

本任务通过单片机对接在 P1.0 口上的一只发光二极管 LED 进行闪烁控制。控制过程为上电后发光二极管 LED 灯点亮,持续点亮一段时间后,LED 灯熄灭,熄灭相同的时间后再点亮……这样周而复始地进行下去,形成“眨眼睛”的效果。通过实训体验单片机控制外围设备的方法,了解单片机硬件系统和软件指令系统协调工作的过程,激发学生学习单片机应用技术的兴趣。

任务分析

程序设计是单片机开发最重要的工作,而程序在执行过程中常常需要完成延时的功能。例如,在交通灯的控制程序中,需要控制红灯亮的时间持续 30 s,就可以通过延时程序来完成。延时程序是如何实现的呢?在单片机编程里并没有真正的延时指令,从前面介绍过的机器周期和指令周期的概念中,我们知道单片机每执行一条指令都需要一定的时间,所以要达到延时的效果,可以让单片机不断地执行没有具体实际意义的指令或循环重复某种操作,从而达到延时的效果。

知识链接

一、简单的单片机时间控制

1. 空操作指令 NOP

空操作指令功能只是让单片机执行没有意义的操作,消耗一个机器周期。

2. 循环转移指令 DJNZ

循环转移指令功能是将第一个数进行减 1 并判断是否为 0，不为 0 则转移到指定地点，为 0 则往下执行。例如，

```
KK：DJNZ R7,KK
```

将寄存器 R7 的内容减 1 并判断寄存器 R7 里的内容减 1 后是否为 0，如果不为 0 则转移到地址标号为 KK 的地方；如果为 0 则执行下一条指令。如果晶体振荡频率为12 MHz，每个机器周期为 1 μs，执行这条指令需要 2 个机器周期。又如，

```
      MOV R7,#250
KK：  DJNZ R7,KK
```

执行本段程序之后，大约延时 250×2＝500 μs＝0.5 ms。

3. 利用定时器延时

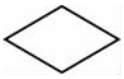
上面的定时方式误差较大，若想精确定时，则要利用单片机的定时器 T0 或 T1 来完成。详见“学习任务五中的单片机定时/计数器应用——交通灯控制系统设计”部分。

利用以上三种操作方式的组合就可以比较精确地编写出所需要的延时程序。

二、程序流程图

流程图就是用箭头线将一些规定的图形符号（如半圆弧形框、矩形框、菱形框等）有机地连接起来的图形。这些半圆弧形框、矩形框和菱形框与文字符号相配合用来表示实现某一特定功能或求解某一问题的步骤。利用流程图可以将复杂的工作条理化、抽象的思路形象化。流程图中常用的图形符号见表 1-11。

表 1-11　流程图中常用的图形符号

符　号	名　称	意　义
	端点、终止框	表示程序的开始或结束
	处理框	表示一段程序的功能或处理过程
	判断框	表示条件判断，以决定程序的流向
	流程线	指示程序执行的流向
	连接、分页符	当流程图在一页画不下需要分页时，使用该符号表示相关流程图之间的连接

常见的流程图结构说明如图 1-19 所示。

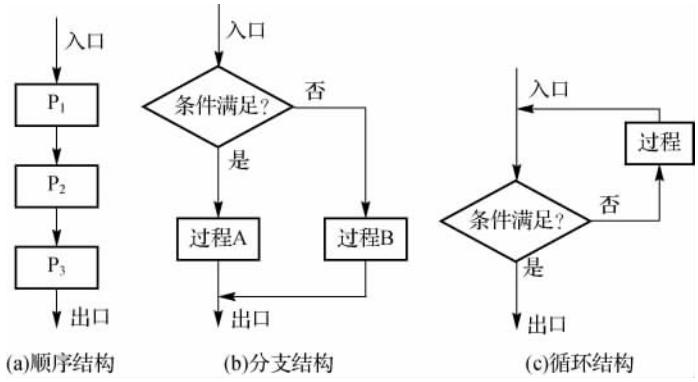


图 1-19　常见的流程图结构说明

实训模块　“眨眼”的 LED 灯电路设计

一、硬件电路原理图的设计

1. 电路设计思路及控制要求

“眨眼”的 LED 灯的具体控制原理是采用典型的单片机芯片 AT89S51 进行控制的。利用单片机的 P1 端口中的某位外接一只发光二极管(LED),发光二极管负极接 P1.0,正极通过限流电阻接电源。要求单片机 P1.0 引脚所控制的 LED 实现“眨眼”的效果。当 P1.0=0 时,对应的 LED 灯就会被点亮;相反,当 P1.0=1 时,对应的 LED 灯就会被熄灭。

2. 硬件电路原理图

根据上述的控制要求,所设计出的电路原理图如图 1-20 所示。

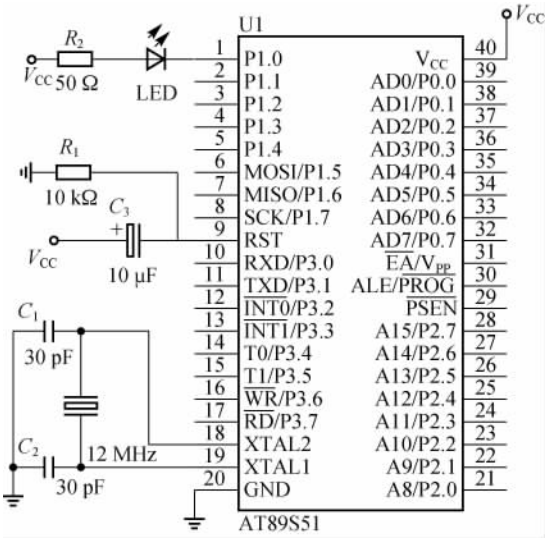


图 1-20　“眨眼”的 LED 灯电路原理图

二、会“眨眼”的 LED 系统程序设计

1. 主程序流程图

根据程序设计思路，画出程序流程如图 1-21 所示。

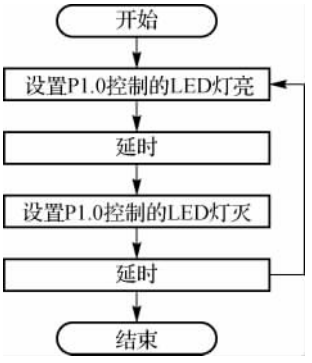


图 1-21 “眨眼”的 LED 灯程序流程图

2. 采用字节操作进行 LED 灯的控制

参考程序（采用字节操作）编写如下，程序仿真图如图 1-22 所示。

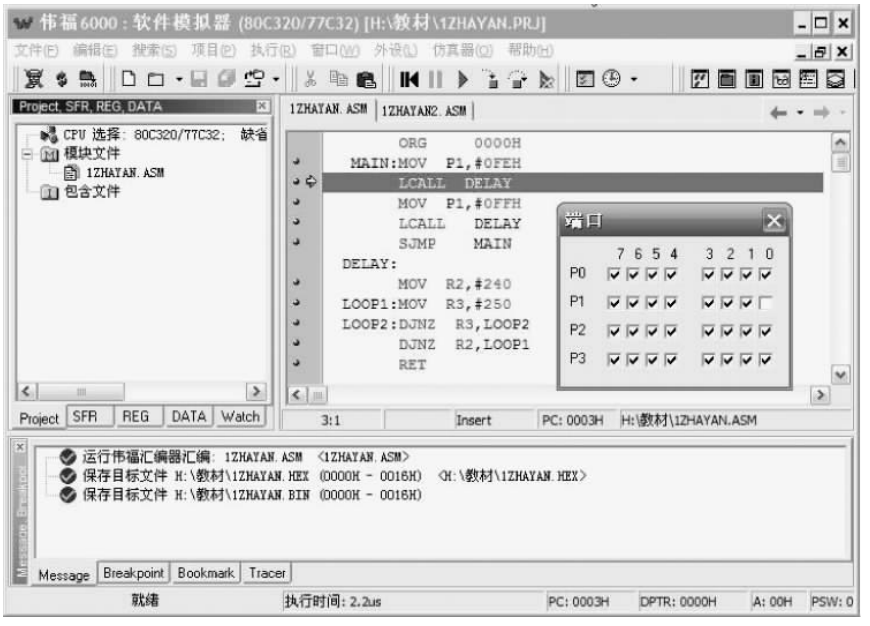


图 1-22 “眨眼”的 LED 灯程序仿真图（字节操作）

```
        ORG  0000H        ;将程序从地址 0000H 处开始
MAIN:   MOV P1,#0FEH      ;(P1)=1111,1110B,发光二极管点亮
        LCALL DELAY       ;调用延时子程序
        MOV P1,#0FFH      ;(P1)=1111,1111B,发光二极管熄灭
        LCALL DELAY       ;调用延时子程序
```

```
        SJMP  MAIN          ;程序返回到 MAIN 处,重复“眨眼”过程
DELAY:  MOV  R2,#240        ;延时子程序
LOOP1:  MOV  R3,#250
LOOP2:  DJNZ  R3,LOOP2
        DJNZ  R2,LOOP1
        RET                 ;子程序调用返回
        END                 ;汇编结束
```

3. 采用位操作进行 LED 灯的控制

参考程序(采用位操作)编写如下,程序仿真图如图 1-23 所示。

```
        ORG  0000H          ;将程序从地址 0000H 处开始
MAIN:   CLR  P1.0           ;(P1.0)=0 时,发光二极管点亮
        LCALL  DELAY        ;调用延时子程序
        SETB  P1.0          ;(P1.0)=1 时,发光二极管熄灭
        LCALL  DELAY        ;调用延时子程序
        SJMP  MAIN          ;程序返回到 MAIN 处,重复“眨眼”过程
DELAY:  MOV R6,#240         ;延时子程序
LOOP1:  MOV R7,#250
LOOP2:  DJNZ R7,LOOP2
        DJNZ R6,LOOP1
        RET                 ;子程序调用返回
        END                 ;汇编结束
```

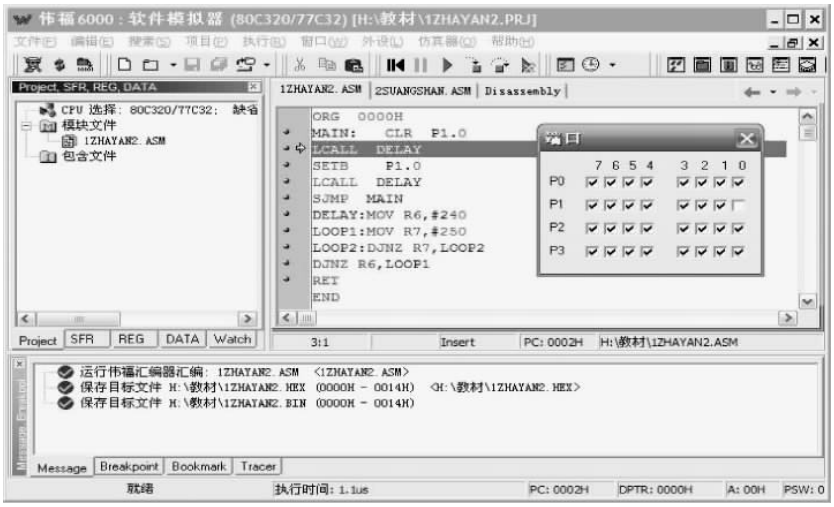


图 1-23　“眨眼”的 LED 灯程序仿真图(位操作)

注意 以上两段汇编程序中调用子程序的目的是实现延时效果，其原因在于单片机的处理速度太快。如果在发光二极管由亮到灭的切换过程中不进行延时处理，由于人的眼睛有视觉差，肉眼所观察到的效果是这个发光二极管始终保持亮的状态。另外，上述两段程序中的子程序延时必须超过一定时间，否则也同样会产生持续亮的效果。

4. 延时程序参数的变化对“眨眼”节奏的影响

改变延时子程序 DELAY 中的 R6 或 R7 的参数，观察二极管“眨眼”节奏的变化情况。注意 R6 和 R7 的值不能超过 255 或 0FFH，若想增加延时时间，可以再加一层延时循环。

三、“眨眼”电路的实训效果图

“眨眼”电路的实训效果图如图 1-24 所示。

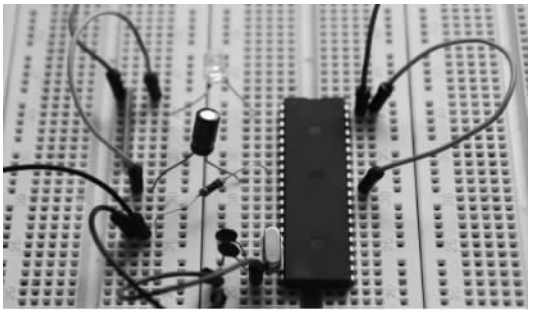


图 1-24 “眨眼”电路的实训效果图

评价与考核

课程名称：单片机应用技术	授课地点：	
学习任务：51 系列单片机(AT89S51)基础知识	授课教师：	授课学时：
课程性质：理实一体课程	综合评分：	

知识掌握情况评分(50 分)

序　号	知识考核点	教师评价	配　分	实际得分
1	单片机的发展历史和应用范围		5	
2	AT89S51 的结构组成		10	
3	单片机的存储结构		10	
4	单片机的输入/输出(I/O)		10	
5	单片机编程语言		15	

工作任务完成情况评分(50 分)

序　　号	能力操作考核点	教师评价	配　　分	实际得分
1	能够根据控制需要连接相对简单的单片机外围电路		15	
2	能够读懂简单的单片机控制程序		15	
3	能够编制简单的程序		20	

扣分情况(20 分)

序　　号	违规内容	配　　分	实际扣分
1	学习中玩手机、打游戏	5	
2	课上吃东西	5	
3	课上打电话	5	
4	其他扰乱课堂秩序的行为	5	

小　　结

本学习任务详细介绍了单片机的存储结构、端口结构、指令语言、程序设计流程等基础知识，结合任务实施，介绍了单片机指令系统的构成、指令的寻址方式及各类指令的格式、功能和使用方法。

51 系列单片机共有 7 种寻址方式，即立即寻址、直接寻址、寄存器寻址、寄存器间接寻址、变址寻址、相对寻址和位寻址。寻址方式的不同主要表现为取操作数的方法和寻址范围。

51 系列单片机指令按功能分为 5 大类 111 条，由于条数比较多，初学时不宜死记硬背。对于具体的指令，应结合寻址方式找出规律来记忆。累加器 A 是在指令中出现最频繁的一个特殊功能寄存器，除了位操作类指令和 DJNZ 指令与它无关，其余指令组中都有它。算术运算指令执行时通常会对进位标志 CY、半进位标志 AC、溢出标志 OV 及奇偶校验标志 P 产生影响，其余各类指令执行时一般不影响标志位，涉及累加器 A 或进位标志位 CY 的指令操作会影响 P 标志位及 CY 标志位。

本学习任务最后还增加了一个单片机体验实训项目，目的是让学生初步感受单片机软硬件是如何配合工作的，以提高学生对单片机应用的认知水平，增强其学习单片机知识的兴趣。

问题与思考

1. 什么是单片微型计算机？
2. 单片机的发展经历了哪几个阶段？

3. 单片机可分为几个系列？简述每个系列的主要特性。

4. 简述单片机技术发展的趋势。

5. 单片机具有哪些突出优点？举例说明单片机的应用领域。

6. MCS-51 单片机内部包含哪些主要逻辑功能部件？

7. 说明程序计数器 PC 和堆栈指针 SP 的作用。复位后 PC 和 SP 各为何值？

8. 程序状态字寄存器 PSW 的作用是什么？其中状态标志有哪几位？它们的含义分别是什么？

9. 什么是堆栈？堆栈有何作用？为什么要对堆栈指针 SP 重新赋值？SP 的初值应如何设定？

10. 开机复位后，CPU 使用的是哪组工作寄存器？它们的地址如何？CPU 如何指定和改变当前工作寄存器组？

11. MCS-51 的时钟周期、机器周期、指令周期是如何定义的？当振荡频率为 12 MHz 时，一个机器周期为多少微秒？

12. MCS-51 单片机的引脚有哪些？各有哪些功能？

13. MCS-51 的片外程序存储器和片外数据存储器共处同一地址空间为什么不会发生总线冲突？

14. 简述 MCS-51 内部数据存储器的存储空间分配。

15. 位地址和字节地址有何区别？位地址 20H 具体在内存中什么位置？

16. 8051 的 4 个 I/O 端口作用是什么？8051 的片外三总线是如何分配的？

17. 汇编语句是由 4 个部分(字段)构成的，简述各部分的含义。

18. 举例说明 MCS-51 单片机的 7 种寻址方式，以及各寻址方式的寻址空间。

19. 设内部 RAM 中 3AH 单元的内容为 50H，当执行下列程序段后，寄存器 A、R0 和内部 RAM 50H，51H 单元的内容为何值？

```
MOV A,3AH
MOV R0,A
MOV A,#00H
MOV @R0,A
MOV A,#25H
MOV 51H,A
```

20. 设堆栈指针 SP 中的内容为 60H，内部 RAM 30H 和 31H 单元的内容分别为 27H 和 1AH，执行下列程序段后，61H，62H，30H，31H，DPTR 及 SP 中的内容将有何变化？

```
PUSH 30H
PUSH 31H
POP DPL
POP DPH
MOV 30H,#00H
MOV 31H,#0FFH
```

21. 设(A)＝30H，(R1)＝23H，(30H)＝05H。执行下列两条指令后，累加器 A 和 R1 以及内部 RAM 30H 单元的内容各为何值？

XCH A,R1

XCHD A,@R1

22. 设(A)＝01010101B,(R5)＝10101010B,分别写出执行下列指令后的结果。

ANL A, R5

ORL A, R5

XRL A, R5

23. 编程将数据存储器中以 2A00H 为首地址的 100 个连续单元清零。

24. 从 2030H 单元开始,存有 100 个有符号数,要求把它传送到从 20BOH 开始的存储区中,但负数不传送,试编写程序。

25. 若晶振为 6 MHz,试编写延时 100 ms、1 s 的子程序。