例 1.19　一个定点纯小数为 5AH。写出它的原码。

定点纯小数为 5AH，它表示为原码：00000000.01011010B。其真值为+0.0101101B。

例 1.20　用原码形式把十进制数－32.625 表示为一个字节的整数和一个字节的小数。

－32 的二进制为－100000B，0.625 的二进制为 0.101。因此，－32.625 的原码为

$$[X]_{原}=10100000.10100000B$$

其中最高位 1 是符号位。

2）浮点数

在十进制数中，常采用科学计数法来表示一个数据，如

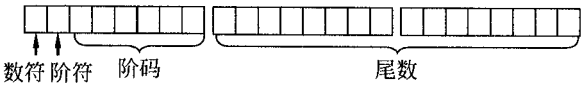
$$-5123.123\ 34=-0.512\ 312\ 334\times10^{+4}$$

由于十进制数的基为 10，可采用 4 部分来描述这个数据：符号为“－”（简称数符），有效位为 512312334（简称尾数），阶的符号为“＋”（简称阶符），阶的大小为 4（简称阶码）。因此，用数符、尾数、阶符、阶码 4 部分能正确地描述一个浮点数。显然，阶的大小不同，小数点的位置是浮动的，故称为浮点数。

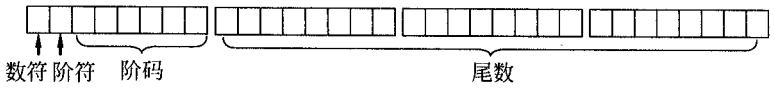
类似地，二进制数也可以表示成这种科学记数法的形式，一个二进制数的浮点表示为：

$$B=\pm S\times2^{\pm J}$$

S 为尾数，J 为阶码，它们均为整数。常见的有 3 字节浮点数和 4 字节浮点数，它们的格式如图 1.9 所示。图 1.9 中，第 1 个字节的最高位为尾数的符号位，即数符，也是整个浮点数的符号位；其余 7 位为阶数，它为带符号整数，常用补码表示。第 2 个字节之后为尾数，通常是纯小数，常用原码表示。4 字节浮点数相当于 C 语言中的单精度实型数据。关于浮点数的其他表示法和运算，请查阅有关计算机应用方面的文献。



(a) 3字节浮点数



(b) 4字节浮点数

图 1.9　浮点数格式

例 1.21　求十进制数－6 的 3 字节浮点数。

（1）把十进制数转换为二进制数：－6 转换为二进制为－110。

（2）求阶数和尾数。出于简便的目的，此处把阶码写成十进制数的形式，则：

$$-110=(0.110)\times2^{+3}$$

浮点数的第 1 个字节为 10000011，表示该数据为负，阶数为 0000011，即＋3，而后两个字节尾数为 11000000 00000000。尾数不足 8 位的，低位用 0 补齐。

（3）写成浮点数。－6 的 3 字节浮点数为 10000011 1100000000000000 转换成十六进制为 83 C0 00H。

例 1.22　求十进制数 31.25 的 3 字节浮点数。

（1）首先把十进制数转换为二进制数：31.25＝11111.01B。

(2) 确定阶数和尾数：11111.01=0.1111101×2^{+5}。

则第1个字节为00000101，尾数为11111010 00000000

(3) 31.25的3字节浮点数为00000101 11111010 00000000 B，转换成十六进制数为05 FA 00H

-0.125

例1.23 求十进制数−0.625的3字节浮点数。

(1) 把十进制数−0.625转换为二进制：−0.625=−0.001B。

(2) 确定阶数和尾数：−0.001=0.1×2^{-2}。

阶数−2的补码为1111110，因为该数为负数，则浮点数第1字节为11111110，尾数为10000000 00000000。

(3) −0.625的3字节浮点数为：11111110 10000000 00000000 B=FE 80 00H

-0.1

例1.24 求十进制数67.234的4字节浮点数。

(1) 把十进制数67.234转换为二进制：67.234 ≈1000011.00111011 11100111 0111。

(2) 确定阶数和尾数：

1000011.00111011 11100111 0111=0. 10000110 01110111 11001110 111×2^{+7}

因为该数为正数，则浮点数第1字节为00000111，4字节浮点数尾数为3个字节，因此尾数只取前24位，即10000110 01110111 11001110。

(3) 67.234的4字节浮点数为：

00000111 10000110 01110111 11001110 B=07 86 77 CEH

1.3.3 编码

由于计算机只能处理二进制数和二进制编码，因此，任何进入计算机的信息必须转化为二进制数或二进制编码。

1. 二进制编码

数码符号不仅可以用于记数表示数值的大小，而且可以用于表示特定的对象。如在日常生活中，电话号码、邮政编码、手机号码、身份证编号、学号等就是用0～9这10个十进制数码符号的组合来表示特定的对象，可以称为十进制代码。同样，由0和1组成的二进制数码不仅可以表示数值的大小，也可以用来表示特定的信息。这种具有特定含义的二进制数码称为二进制代码。建立这种代码与它表示的对象(如十进制数、字母、特定符号、逻辑值等)的一一对应关系的过程称为编码；将代码所表示的特定信息翻译出来称为译码，分别由编码器、译码器来实现。

计算机最重要的功能是处理信息，这些信息包括数值、文字、图形、符号、图像、声音以及模拟信号等，这些信息必须经过编码，转换为计算机能够识别和处理的二进制编码，才能被计算机存储备份、传送复制、加工分析、显示输出。

二进制编码是用预先规定的方法将数值、文字、图形、符号、图像、声音以及模拟信号等编成二进制的数码。如BCD码、ASCII码、GB2312码等标准编码，还有A/D转换、D/A转换数据与模拟信号之间的编码、字符显示的字型编码等。

2. 十进制数的4位二进制编码(BCD码)

十进制数的4位二进制编码就是用4位二进制数来表示0～9这10个十进制符号，简称为BCD码(Binary−Coded Decimal，BCD)。由于4位二进制数从0000～1111共有16种组合，而十进制只有10个数码符号，因此有很多种BCD码。如8421码、2421码等。常用

```
XCH   A, 2EH
```

这两种方法的区别是：第一种方法在程序存储器中需要 12 个单元存储指令代码，执行时需要 8 个机器周期，而后者只需要 9 个单元，执行时间为 5 个机器周期。

4. 访问程序存储器的数据传送指令

MCS-51 单片机的程序存储器主要用于存放程序指令代码(Code)，还可用来存放程序中需要的常数。这些常数存储在程序存储器的一个区域，由若干个连续单元构成，通常称为表或表格(Table)。因此，访问程序存储器的数据传送指令也叫查表(Lookup Table)，它的功能是从程序存储器某个单元读取一个字节的常数。指令有两种形式：

```
MOVC  A,@A+DPTR      ;[(A)+(DPTR)]→(A),常数所在存储单元的地址由 DPTR 和累加
                     ;器 A 的内容之和确定
MOVC  A,@A+PC        ;(PC)+1→(PC),CPU 取指令代码;[(A)+(PC)]→(A),常数所在
                     ;存储单元的地址由程序计数器 PC 和累加器 A 的内容之和确定
```

例 3.19　采用查表方法获取一个数 $x(0\leqslant x\leqslant 15)$的平方值。

首先在程序存储器中建立一个 $0\leqslant x\leqslant 15$ 的平方表，定义从 5000H 开始的连续 16 个单元中分别存有 0～15 的平方值。x 存放在累加器 A 中，x 取值为 00H～0FH。程序执行后，得到 x 的平方值在累加器 A 中。分别用上述两种指令设计查表程序。

(1) 采用“MOVC A,@A+DPTR”指令

```
MOV   DPTR,#5000H    ;表的首地址送 DPTR
MOVC  A,@A+DPTR      ;查表获得的值送累加器 A
RET                  ;返回
ORG   5000H          ;0≤x≤15 的平方表从 5000H 存放
  DB  00H            ;0²,DB: Define a byte,伪指令,在此定义 5000H 单元的内容是常
                     ;数而非指令代码,在程序中起说明作用
  DB  01H            ;1²存放在 5001H 单元
  DB  04H            ;2²存放在 5001H 单元
...
  DB  0E1H           ;15² = 225,存放在 500F1H 单元
```

5002H

500FH

说明：若(A)=00H，程序执行后，[(A)+(DPTR)]→(A)，即(00+5000)=(5000H)→(A)，则(A)=00H。若(A)=02H，则[(A)+(DPTR)]→(A)，即(5002H)→(A)，(A)=04H。

实际上，如果平方表放在程序存储器的其他地方，如存放在 3700H，只要在程序中修改表的首地址，“MOV DPTR,#3700H”，程序运行的结果是相同的。

(2) 采用“MOVC A,@A+PC”指令

由于这条指令执行时，其结果与 PC 有关，为了分析方便，把每条指令及其代码同时给出，程序从程序存储器的 1000H 单元开始存放，同样，x 存放在累加器 A 中，x 取值为 00H～0FH。程序执行后得到 x 的平方值在累加器 A 中。

【标号】	【指令】	【注释】	【单元地址】	【指令代码】
CHECKUP:	INC A	;(A)的内容+1	1000H	04
	MOVC A,@A+PC	;(PC)+1→(PC), ;[(A)+(PC)]→(A)	1001H	83
	RET	;返回	1002H	22
	DB 00H	;0²	1003H	00H

算结果在累加器 A 中；指令执行时影响标志位 Cy,AC,OV,P。指令有以下 4 种形式：

```
SUBB   A,#data        ;(A)-data-(Cy)→(A)
SUBB   A,Rn           ;(A)-(Rn)-(Cy)→(A)
SUBB   A,direct       ;(A)-(direct)-(Cy)→(A)
SUBB   A,@Ri          ;(A)-[(Ri)]-(Cy)→(A)
```

例 3.31 设累加器 A 的内容为 0C9H,寄存器 R2 的内容为 5AH,当前 Cy 的状态为 1,执行指令“SUBB A,R2”后,累加器 A 和标志位 Cy,AC,OV,P 的状态如何?

指令“SUBB A,R2”的执行过程如图 3.24 所示。累加器 A 的内容为 6EH,(Cy)=0,(AC)=1,(OV)=0,(P)=1。

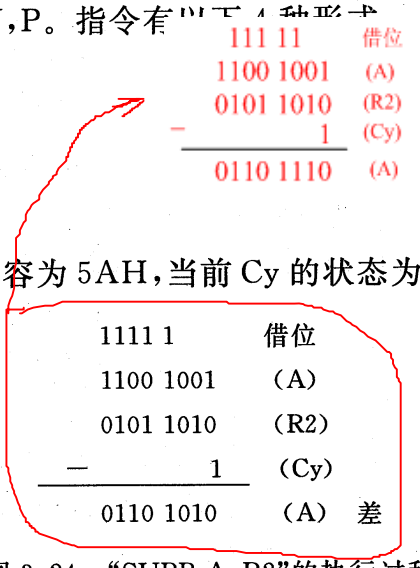


图 3.24 “SUBB A,R2”的执行过程

0C9H−5AH=6FH,而例 3.31 中累加器 A 的内容为 6EH,这是因为减法指令执行时,当前的进位位 Cy 参与了运算。在进行减法时,如果不能确定进位位 Cy 的状态,在应用减法指令时,必须对进位位 Cy 清零(用指令“CLR Cy ”或“CLR C”),以保证正确的运算结果。

例 3.32 已知 x 存放在 20H 单元,y 存放在 21H 单元,假设 $x \geqslant y$,求 $z=x-y$。

两个任意的 8 位二进制数 x,$y(x \geqslant y)$相减,结果会是几个字节呢? 因为 x 和 y 是无符号数,一个字节最大的二进制数为 0FFH,最小为 00H。当 x 是 0FFH、y 取 00H 时,$x-y$ 的结果为 0FFH。显然,给 z 分配一个单元足够了。程序如下:

```
MOV    A, 20H
CLR    C              ;借位位清零
SUBB   A, 21H
MOV    20H,  A        ;差存放在 20H 单元
```

2) 减 1 指令

减 1 指令的一般形式:

```
DEC    源             ;源操作数-1→源操作数
```

源操作数必须是一个寄存器或存储单元,有以下 4 种形式:

```
DEC    A              ;(A)-1→(A),该指令执行时,不影响标志 Cy,AC 和 OV
DEC    Rn             ;(Rn)-1→(Rn),n=0～7,该指令执行时,不影响标志位
DEC    direct         ;(direct)-1→(direct),该指令执行时,不影响标志位
DEC    @Ri            ;[(Ri)]-1→[(Ri)],i=0, 1,该指令执行时,不影响标志位
```

若原来寄存器或单元的内容为 00H,减 1 运算后,其内容变为 0FFH,即向下溢出。与 INC 类指令类似,对 I/O 口操作时,参与减 1 运算的是 I/O 口对应的寄存器的内容,而不是来自于引脚的状态,但最终的运算结果将从 I/O 口的引脚输出,并修改寄存器的内容。

例 3.33 设 R0 的内容为 7EH,内部 RAM 的 7DH 和 7EH 单元的内容分别为 00H 和 40H,P1 口寄存器的内容为 55H,执行下列指令后,R0,P1 和 7EH 单元的内容分别是多少?

```
DEC    @R0
DEC    R0
```

```
MOV   A, #0FFH
CLR   C
SUBB  A, 30H          ;求反码
ADD   A, #01H         ;求补码
MOV   30H,  A         ;存补码
```

例 3.56 已知单片机应用系统外部 RAM 的一个单元 80FDH,在应用程序中需要把该单元的第 5 位取反,第 6 位和第 4 位置 1,第 3 位和最低位清零。

80FDH 是单片机外部 RAM 的一个单元,要修改该单元的内容,必须把它的内容读入单片机,修改完成后,再写入原单元。用异或指令把第 5 位取反的异或码为 00100000B(20H);用与指令把第 3 位和最低位清零,相与码为 11110110B(0F6H);用或指令把第 6 位和第 4 位置 1,相或码为 01010000B(50H)。程序如下:

```
MOV   DPTR, #80FDH
MOVX  A,@DPTR         ;读 80FDH 单元的内容
XRL   A, #20H         ;第 5 位取反
ANL   A, #0F6H        ;第 3 位和最低位清零
ORL   A, #50H         ;第 6 位和第 4 位置 1
MOVX  @DPTR,A         ;输出
```

3.3.5 位操作指令

在硬件上,MCS-51 单片机的布尔处理是一个完整的系统,它包括位处理器、位寻址空间、可以位寻址的 I/O 口等。在位处理时,位累加器 C 借用了 PSW 的进位位 Cy,位寻址空间由内部 RAM 的 128 位(位地址 00H～7FH)和特殊功能寄存器区的可寻址位(位地址为 80H～FFH)提供。另外,所有的 I/O 口线都是可以位寻址的,每根口线可以当作独立的口使用。位操作指令支持对位的直接操作,包括位传送、位逻辑运算以及位控制转移指令,为逻辑处理提供了一种高效的方法,可使逻辑电路软件化,减少系统中元器件的数量,提高系统可靠性。本节介绍位传送和位运算指令,位控制转移指令将在 3.3.6 节介绍。

1. 位数据传送指令

```
MOV   C,bit           ;(bit)→(C)
MOV   bit,C           ;(C)→(bit)
```

位传送指令仅支持某 1 个指定位 bit 与布尔处理器 C 之间的状态传送,两个位之间不能直接进行状态传送,必须通过 C 来进行。

例 3.57 已知(Cy)=1,P3 口为输入,当前状态为 11000101B,P1 口为输出,先前写入的数据为 35H(00110101B),执行下列程序:

```
MOV   P1.3,  C
MOV   C,  P3.3
MOV   P1.2,  C
```

结果为:(Cy)=0,(P1)=39H (00111001B),P3 口状态保持不变。

例 3.68　x 和 y 以补码形式分别存储在 R0 和 R1 中，求：

$$y=\begin{cases}1 & x>0\\0 & x=0\\-1 & x<0\end{cases}$$

0 的补码为 00H，负数的补码最高位为 1，正数的补码最高位为 0，根据题意，程序的设计流程框图如图 3.48 所示。

```
            CJNE R0,   #00,   NON_ZERO      ;x 是否为 0
            MOV  R1,  #00                   ;x = 0,   y = 0
            RET
NON_ZERO:   MOV  A,   R0                    ;取 x
            ANL  A,   #10000000B            ;提取符号位信息
            CJNE A,   #80H,   GT0           ;x 是否大于 0
            MOV  R1,  #0FFH                 ;x = 0, y = -1, -1 的补码为 0FFH
            RET
    GT0:    MOV  R1,  #01H                  ;x = 0,y = 1,
            RET
```

3）以进位位 Cy 状态为判别条件的转移指令

（1）以 Cy 状态是 1 为判别条件的转移指令

```
JC   rel
```

该指令的指令代码为 2 字节，CPU 执行过程为：

① 取指令：(PC)+2→(PC)；

② 执行并获取目标地址：若(Cy)=1，则(PC) +rel→(PC)；若(Cy)=0，则顺序向下执行。

CPU 执行过程流程图如图 3.49(a) 所示。

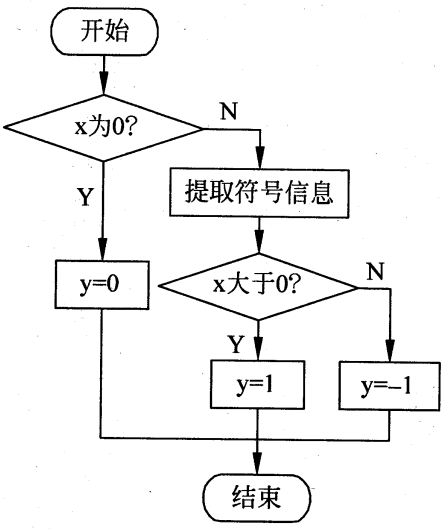


图 3.48　例 3.68 的程序的流程框图

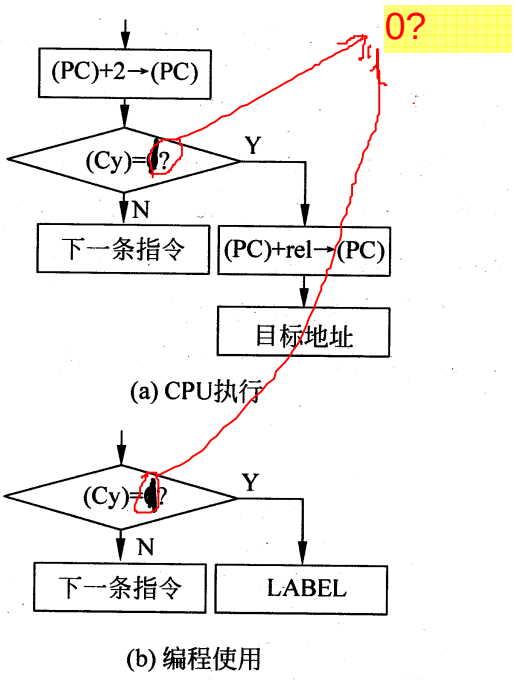


图 3.49　JC 指令的流程图

该指令在程序中的使用方式为:

```
JC   LABEL
```

编程使用方式的流程图如图 3.49(b)所示。

(2) 以 Cy 状态是 0 为判别条件的转移指令

```
JNC  rel
```

该指令的指令代码为 2 字节,CPU 执行过程为:

① 取指令:(PC)+2→(PC);

② 执行并获取目标地址:

若(Cy)=0,则(PC) +rel→(PC);

若(Cy)=1,则顺序向下执行。

CPU 执行过程流程图如图 3.50(a) 所示。

该指令在程序中的使用方式为:

```
JNC  LABEL
```

编程使用方式的流程图如图 3.50(b) 所示。

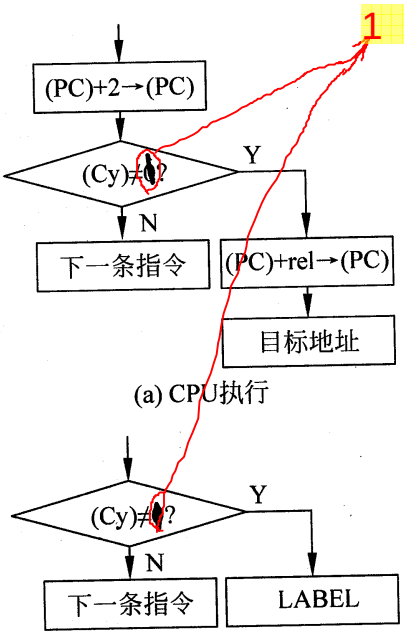


图 3.50 JNC 指令的流程图

以上两种以进位标志 Cy 的状态为判断条件,满足条件则转移到目标地址处。在程序设计时,建立判断条件的途径如下:

(1) 位传送:MOV C, bit。

(2) 算术运算(加、减法指令):ADD/ADDC/SUBB。

(3) 带进位移位的指令:RLC A,RRC A。

(4) 位逻辑运算:与、或运算。

例 3.69 比较两个 8 位二进制无符号数 x、y 的大小,并将大数存放在 MAX 单元,若相等,置标志位 F0 位为 1,否则,F0 清零。

根据题意,可采用两种方法比较 x、y 的大小,程序流程图如图 3.51 所示。设 x 和 y 分别存储在 20H 和 21H 单元。

程序一:采用减法比较大小。

```
        MOV   A,   20H          ;取 x
        CLR   C
        SUBB  A,   21H          ;减法
        JZ    EQU               ;差为 0,相等
        CLR   F0                ;不相等
        JNC   GRT               ;没有借位,x 大于 y
        MOV   MAX, 21H          ;y 大于 x,存大数
        RET                     ;返回
EQU:    SETB  F0                ;x 和 y 相等
GRT:    MOV   MAX, 20H          ;存大数
        RET
```

4. 空操作指令

```
NOP
```

该指令的指令代码为 1 字节，CPU 执行过程为：

取指令：(PC)＋1→(PC)

这是一条单字节指令，执行时间(指令周期)为 1 个机器周期(T_M)。该指令执行时，不做任何操作(即空操作)，仅将程序计数器 PC 的内容加 1，使 CPU 指向下一条指令继续执行程序。这条指令常用来产生一个机器周期的时间延迟。

例 3.75　一个能延时 1s 的软件延时子程序。假设系统的晶体振荡器频率为 6MHz。

```
DELAY:  MOV R2, #250          ;指令周期为 1TM
DELY1:  MOV R3, #250          ;1TM
DELY2:  NOP                   ;1TM
        NOP                   ;1TM
        NOP                   ;1TM
        NOP                   ;1TM
        NOP                   ;1TM
        NOP                   ;1TM
        DJNZ  R3,  DELY2      ;指令周期为 2TM
        DJNZ  R2,  DELY1      ;指令周期为 2TM
        RET                   ;指令周期为 2TM
```

由于晶振的频率为 6MHz，则 $T_M=2\mu s$，总延时时间为：

$$T=T_M+[250\times(2T_M+6\times T_M)+3T_M]\times 250+2T_M=1001ms\approx 1s$$

例 3.76　一个单片机应用系统采用串行方式给外围芯片发送数据，先发低位，每次传送一个字节，发送数据的时序如图 3.61 所示。请用编程模拟实现图 3.61 的时序。设片选信号 CS 为 P1.0，时钟信号 CLK 为 P1.1，数据输出 DAT 为 P1.2。

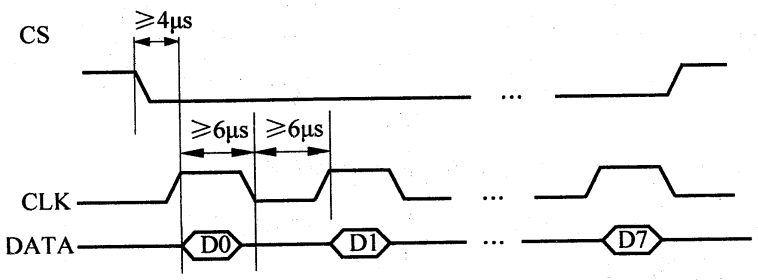


图 3.61　单片机应用系统发送数据的时序

发送数据的时序可以采用延时方法来实现。单片机执行指令需要时间，执行指令能起到延时的作用。在发送数据时，先从数据的最低位开始，因此采用“RRC A”指令把需要发送的位移入 Cy 中，然后由 Cy 传递给 P1.2(DAT)输出。下面是数据发送子程序时，待发送的数据放在累加器 A 中。应用系统晶振频率为 12MHz。

```
SENDBYTE:  CLR  CLK           ;初始化 CLK
           CLR DAT            ;初始化 DAT
           CLR CS             ;CS = 0,准备传送数据
           MOV R5, #08        ;发送数据位数
```

8. 在图 4.20 单片机应用系统中，A，B 两路检测信号分别从 P3.2($\overline{\text{INT0}}$)和 P3.3($\overline{\text{INT1}}$)引入单片机，通常情况下，当 A，B 为高电平时，表示系统工作正常，指示灯 L1 亮；当 A 出现低电平时，指示灯 L1 灭，L2 以 500ms 的间隔闪烁，除非 A 再次变为高电平，系统恢复正常。无论在什么情况下，只要 B 出现低电平，关闭指示灯 L1，L2 以 200ms 的间隔闪烁，同时蜂鸣器 BUZ 以 200ms 的间隔鸣叫，除非 B 再次变为高电平，系统恢复正常。采用中断方式实现以上监控功能。

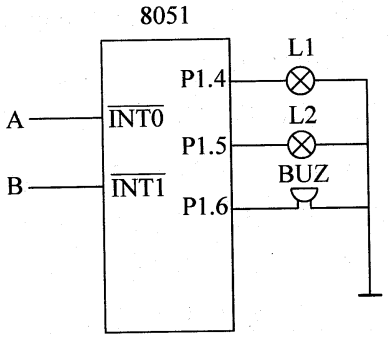


图 4.20　单片机应用系统原理图

当 GATE＝0 时，只要 TR0 为 1，计数器 TL0 就开始计数；

当 GATE＝1 时，仅当 TR0 为 1，且$\overline{\text{INT0}}$引脚输入状态为 1 时，计数器 TL0 才开始计数。

计数器开始工作时，8 位计数器 TL0 从初始值开始加 1 计数，当计数器各位全为 1 以后，计数器再计 1 次产生溢出，则 TF0 位被自动置 1，同时把 TH0 的内容装载到 TL0。

在方式 2 下，计数器计数范围是 1～256，而定时时间范围是 1～256 个机器周期。

1. 计数器工作模式

设初始值为 X，计数器计数 N 次后溢出，则 $X+N=2^8$，得到 $X=2^8-N$。因此，预先给计数器装入初始值 $X=2^8-N$，计数器计数 N 次后溢出，把 TF0 置 1。

2. 定时器工作模式

要求计数器定时后溢出。首先把 t_d 换算为机器周期的个数，$N=\frac{t_d}{T_M}$。设初始值为 X，则 $X+N=2^8$，得到 $X=2^8-N$。预先给计数器装入初始值 $X=2^8-N$，N 个机器周期后计数器溢出，TF0 被置 1，t_d 秒定时时间到。

方式 2 时，定时器/计数器的最大计数次数为 256，最大定时时间为 $256T_M$。

5.3.4　方式 3

当 M1M0 设置为 11 时，定时器/计数器 T0 的工作方式为方式 3。只有定时器/计数器 T0 有工作方式 3，定时器/计数器 T1 没有工作方式 3，如果把 T1 设置为方式 3，计数器将停止工作。

在工作方式 3 下，定时器/计数器 T0 被拆分成两个独立的 8 位计数器 TL0 和 TH0，其逻辑结构如图 5.7 所示。其中 TL0 既可以作为计数器使用，又可以作为定时器使用，它使用了定时器/计数器 T0 所有的控制及标志位：C/$\overline{\text{T}}$，GATE，TF0，TR0 以及外部控制信号输入引脚$\overline{\text{INT0}}$，作为计数器使用时，外部事件的计数输入信号由引脚 T0 输入。另一个 8 位

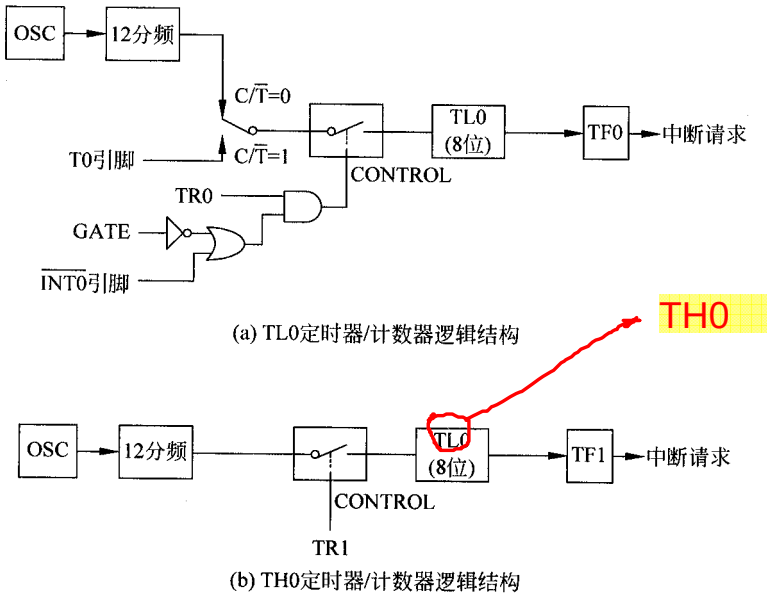


图 5.7　定时器/计数器 T0 工作方式 3 的逻辑结构

证通信双方采用相同的波特率和数据格式。

1. 方式 1 的应用

串行口方式 1 可实现点对点的通信。在数据通信之前，需要进行以下初始化编程：

(1) 确定定时器/计数器 T1 的工作方式，设置 TMOD。通常定时器/计数器 T1 设定为方式 2，定时模式。

(2) 根据波特率，计算定时器/计数器 T1 的计数初始值，分别装入 TH1 和 TL1。

(3) 启动定时器/计数器 T1，SETB TR1。

(4) 确定串行口工作方式，设置 SCON，接收时，把 REN 设置为 1。

(5) 如果采用中断方式，则开放 CPU 中断(EA=1)、允许串行口中断(ES=1)。

例 6.3　A、B 两台 MCS-51 单片机进行单工串行通信，A 机工作在发送状态，B 为接收状态，如图 6.20 所示。现将 A 机片内 RAM 从 30H 单元开始存储的 16B 的数据发送到 B 机，并存储在片内 RAM 的 20H 单元开始的区域。A、B 单片机的晶振频率均为 11.0592MHz，采用波特率为 9600bit/s。

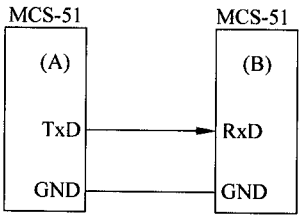


图 6.20　单片机的单工串行通信

定时器/计数器 T1 采用方式 2 的定时模式，下面给出采用查询方式的发送和接收程序。

(1) A 机发送程序：

```
TRANS:   MOV  TMOD, #20H        ;定时器 T1 方式 2,定时模式
         MOV  TH1, #0FDH        ;
         MOV  TL1, #0FDH
         MOV  SCON, #40H        ;设定串行口工作方式 1 发送,允许接收 REN = 1
         MOV  PCON, #00H        ;SMOD = 0
         SETB TR1
         MOV  R0, #30H          ;设发送数据的地址指针
         MOV  R2, #10H          ;设发送数据的长度
LOOP:    MOV  A, @R0            ;取发送数据送 A
         MOV  SBUF, A           ;启动发送
WAIT:    JBC  TI, LOOP1         ;是否发送完?
         SJMP WAIT
LOOP1:   INC  R0
         DJNZ R2, LOOP
         RET
```

(2) B 机接收程序：

```
RECEIVE: MOV  TMOD, #20H        ;定时器/计数器 T1 工作方式
         MOV  TH1, #0FDH
         MOV  TL1, #0FDH
         MOV  SCON, #50H        ;串行口方式 1、接收模式(REN = 1)
         MOV  PCON, #00H
         SETB TR1
         MOV  R0, #20H          ;设接收数据的地址指针
         MOV  R1, #10H          ;设接收数据的长度
LOOP:    JBC  RI, LOOP1         ;等待接收数据
         SJMP LOOP
```