维普资讯 http://www.cqvip.com

初学者园地

手把手教你学单片机(六)

[套件供应]

■周兴华

控制转移类指令学习

单片机有一定的智能作用,主要是控制转移类指令的功劳。这一类指令的功能主要是控制程序从原顺序执行地址转移到其他指令地址上(需要改变程序运行方向,或者需要调用子程序,或需要从子程序中返回)。由于该类指令用于控制程序的走向,所以其作用区间必然是程序存贮器空间,此时都需要改变程序计数器 PC 的内容,控制转移类指令可实现这一要求。MCS-51 指令系统中的控制程序转移类指令,它们是无条件转移和条件转移、绝对转移和相对转移、长转移和短转移,还有调用和返回指令等。这类指令多数不影响程序状态标志寄存器。

1.无条件转移指令:

无条件转移指令功能是,当程序执行无条件转移指令时,程序就无条件地转移到该指令所提供的地址去。

AJMP addr11(绝对无条件转移指令) 这是两字节指令,指令中包含 addr11 共 11 位地址码,转移的目标地址必须和 AJMP 指令的下一条指令首字节位于程序存储器的同一 2KB 区内。绝对转移指令仅为 2 个字节指令,却能提供 2K 范围的转移空间,它比相对转移指令的转移范围大得多。但是要求 AJMP 指令的转移目标地址和 PC+2 的地址处于同一 2KB 区域内,故受一定的限制。

LJMP addr16(长转移指令) 长转移指令是 3 字节指令,这条指令执行时把指令操作数提供的 16 位目标地址 a15~a0 装入 PC 中,即 PC=a15~a0。所以用长转移指令可以跳到 64KB 程序存储器的任何位置。

SJMP rel (短转移指令) 短转移指令是两字节指令,首字节为操作码,第二字节为相对偏移量。它是一条无条件相对转移指令,转移的目标地址为:目标地址=源地址+2+rel。源地址是 SJMP 指令操作码所在的地址,相对偏移量 rel 是一个用补码表示的 8 位带符号数,转移范围为-128~+127 共 256 个单元,即从(PC-126)~(PC+129),因此转移目标地址可以在 SJMP 指令的下条指令首字节前 128 个字节和后 127 个字节之间。

JMP @A+DPTR(间接转移指令) 这条指令的功能是把累加器 A 中的 8 位无符号数与数据指针 DPTR 的 16 位数相加,相加之和作为下一条指令的地址送入 PC 中,不改变 A 和 DPTR 的内容,也不影响标志。间接转移指令采用变址方式实现无条件转移,其特点是转移地址可以在程序运行中加以改变。例如,当把 DPTR 作为基地址且确定时,根据 A 的不同值就可以实现多分支转移,故一条指令可完成多条条件判断转移指令功能,这种功能称为散转功能,所以间接转移指令又称为散转指令。

2.条件转移指令:

条件转移指令是依某种特定条件转移的指令。条件满足时转移(相当于执行一条相对转移指令),条件不满足时则按顺序执行下面一条指令。MCS-51 的条件转移指令非常丰富,包括累加器判零转移、判位(bit)状态转移、比较转移和循环转移共四组。

JZ rel (累加器判零转移) 若 A 为 0,程序跳转至 PC+rel 处执行;若 A 不为 0,则程序顺序执行。

JNZ rel (累加器判非零转移) 若 A 不为 0,程序跳转至 PC+rel 处执行;若 A 为 0,则程序顺序执行。

JB bit,rel (判位为 1 状态转移) 若位(bit)为 1,程序跳转至 PC+rel 处执行;若位(bit)为 0,则程序顺序执行。

JNB bit,rel(判位非 1 状态转移) 若位(bit)为 0,程序跳转至 PC+rel 处执行;若位(bit)为 1,则程序顺序执行。

JBC bit,rel 若位(bit)为 1,将位清 0,程序跳转至 PC+rel 处执行;若位(bit)为 0,则程序顺序执行。

JC bit,rel 若进位位 CY 为 1,程序跳转至 PC+rel 处执行;若 CY 为 0,则程序顺序执行。

JNC bit,rel 若进位位 CY 为 0,程序跳转至 PC+rel 处执行;若 CY 为 1,则程序顺序执行。

3.比较转移指令:

比较转移指令的功能是比较前两个无符号操作数的大小。若不相等,则转移,否则顺序往下执行。如果第一个操作数大于或等于第二个操作数,则 CY 清 0,否则 CY 置 1。指令执行结果不影响其它标志位和所有的操作数。这组指令为 3 字节指令,因此转移目标地址应是 PC+3 以后再加偏移量 rel 所得的 PC 的值。即:目标地址=源地址+3+rel。源地址是比较转移指令所在位置的首字节地址。

CJNE A,direct,rel 若(direct)<A 中内容,则程序跳转至 PC+rel 处,CY=0;若(direct)>A 中内容,则程序也跳转至 PC+rel 处,CY=1;若(direct)=A 中内容,则程

维普资讯 http://www.cqvip.com

序顺序执行,CY=0。换言之,当(direct)≠A 中内容,则程序跳转至 PC+rel 处,CY 是 0 或是 1 要看(direct)与 A 中内容的大小。

CJNE A,#data,rel 若 data<A 中内容,则程序跳转至 PC+rel 处,CY=0;若 data>A 中内容,则程序也跳转至 PC+rel 处,CY=1;若 data=A 中内容,则程序顺序执行,CY=0。换言之,当 data≠A 中内容,则程序跳转至 PC+rel 处,CY 是 0 或是 1 要看 data 与 A 中内容的大小。

CJNE Rn,#data,rel 若 data<Rn 中内容,则程序跳转至 PC+rel 处,CY=0;若 data>Rn 中内容,则程序也跳转至 PC+rel 处,CY=1;若 data=Rn 中内容,则程序顺序执行,CY=0。换言之,当 data≠Rn 中内容,则程序跳转至 PC+rel 处,CY 是 0 或是 1 要看 data 与 Rn 中内容的大小。

CJNE @Ri,#data,rel 若 data<以 Ri 中内容为地址的另一单元内容,则程序跳转至 PC+rel 处,CY=0;若 data>以 Ri 中内容为地址的另一单元内容,则程序也跳转至 PC+rel 处,CY=1;若 data=以 Ri 中内容为地址的另一单元内容中内容,则程序顺序执行,CY=0。换言之,当 data≠(Ri),则程序跳转至 PC+rel 处,CY 是 0 或是 1 要看 data 与(Ri)的大小。

4. 循环转移指令:

DJNZ Rn,rel(寄存器 Rn 减 1 不为 0 循环转移指令) 该指令是把 Rn 中内容减 1,结果送回到 Rn 中去。如果结果不为 0 则转移,为 0 顺序进行。

DJNZ direct ,rel (直接寻址单元 direct 减 1 不为 0 循环转移指令) 该指令是把 direct 中内容减 1,结果送回到 direct 中去。如果结果不为 0 则转移,为 0 顺序进行。

5.子程序调用及返回指令:

ACALL addr11(绝对调用指令) 绝对调用指令 ACALL 上是一条两字节指令,该指令提供了 11 位目标地址 addr11,产生调用地址的方法和绝对转移指令 AJMP 产生转移地址的方法相同,ACALL 是在同一 2K 区范围内调用子程序的指令。指令执行过程是:执行 ACALL 指令时,PC+2 后获得了下一条指令的地址,然后把 PC 的当前值压栈 (栈指针 SP 加 1,PCL 进栈,SP 再加 1,PCH 进栈)。最后把 PC 的高 5 位和指令给出的 11 位地址 addrll 连接组成 16 位目标地址(PC15~11a10~a0),并作为子程序入口地址送入 PC 中,使 CPU 转向执行子程序。因此,所调用的子程序入口地址必须和 ACALL 指令下一条指令的第一个字节在同一个 2KB 区域的程序存储器空间。

LCALL addr16 (长调用指令) 长调用指令 LCALL 是一条可以在 64KB 程序存储器内调用于程序的指令,它是三字节指令。指令执行过程是:把 PC 加 3 获得的下一条指令的地址进栈(先压入低字节,后压入高字节)。进栈操作使 SP 加 1 两次。接着把指令的第二和第三字节(a15~8,a7~0)分别装入 PC 的高位和低位字节中,然后从该地址 addr16(a15~0)开始执行子程序。

RET(子程序返回指令) 这条返回指令的功能是从堆栈中取出断点地址,送给 PC,并从断点处开始继续执行程序。RET 应放在一般子程序的末尾。

RETI(中断返回指令) 这条返回指令的功能也是从堆栈中取出断点地址,送给 PC,并从断点处开始继续执行程序。RETI 也应放在中断服务子程序的末尾。在执行 RETI 指令时,还将清除 MCS-51 中断响应时所置位的优先级状态触发器,开放中断逻辑,使得已申请的较低级中断源可以响应,但必须在 RETI 指令执行完之后,至少要再执行一条指令才能响应这个中断。

下面开始做实验,具体体验这些指令在程序中的作用。

在 S2 板上做一个实验,通电后右边三个数码管显示 000,按下(任何)一个按键,寄存器 R0 从 0 起递加(1、2…9→1、2…),根据 R0 的内容,程序散转到 PR1、PR2…PR9 子程序执行,右边三个数码管分别显示 111、222…999→111、222…。

在我的文档中建立一个文件目录(S10),然后建立一个 S10.uv2 的工程项目,最后建立源程序文件(S10.asm)。

输入下面的程序:

```
序号:1          ORG 0000H;
     2          LJMP MAIN;
     3          ORG 030H;
     4    MAIN: MOV P0,#0C0H;
     5          MOV P1,#0C0H;
     6          MOV P2,#0C0H;
     7          MOV R0,#00H;
     8    ST:   MOV P3,#0FH;
     9          MOV A,P3;
     10         CJNE A,#0FH,F1;
     11         ACALL DEL;
     12         AJMP ST;
     13   F1:   ACALL DEL;
     14         CJNE A,#0FH,F2;
     15         AJMP ST;
     16   F2:   INC R0;
     17         CJNE R0,#0AH,F3;
     18         MOV R0,#00H;
     19   F3:   MOV DPTR,#JPTAB;
     20         MOV A,R0;
     21         CLR C;
     22         RLC A;
     23         JNC NADD;
     24         INC DPH;
     25   NADD: JMP @A+DPTR;
     26   JPTAB: NOP;
     27         NOP;
     28         AJMP PR1;
     29         AJMP PR2;
     30         AJMP PR3;
     31         AJMP PR4;
     32         AJMP PR5;
     33         AJMP PR6;
     34         AJMP PR7;
```

维普资讯 http://www.cqvip.com


```
35          AJMP PR8;
36          AJMP PR9;
37    DEL:  MOV R7,#014H;
38    DEL1: MOV R6,#0FFH;
39    DEL2: MOV R5,#01FH;
40    DEL3: DJNZ R5,DEL3;
41          DJNZ R6,DEL2;
42          DJNZ R7,DEL1;
43          RET;
44    PR1:  MOV P0,#0F9H;
45          MOV P1,#0F9H;
46          MOV P2,#0F9H;
47          ACALL DEL;
48          AJMP ST;
49    PR2:  MOV P0,#0A4H;
50          MOV P1,#0A4H;
51          MOV P2,#0A4H;
52          ACALL DEL;
53          AJMP ST;
54    PR3:  MOV P0,#0B0H;
55          MOV P1,#0B0H;
56          MOV P2,#0B0H;
57          ACALL DEL;
58          AJMP ST;
59    PR4:  MOV P0,#99H;
60          MOV P1,#99H;
61          MOV P2,#99H;
62          ACALL DEL;
63          AJMP ST;
64    PR5:  MOV P0,#92H;
65          MOV P1,#92H;
66          MOV P2,#92H;
67          ACALL DEL;
68          AJMP ST;
69    PR6:  MOV P0,#82H;
70          MOV P1,#82H;
71          MOV P2,#82H;
72          ACALL DEL;
73          AJMP ST;
74    PR7:  MOV P0,#0F8H;
75          MOV P1,#0F8H;
76          MOV P2,#0F8H;
77          ACALL DEL;
78          AJMP ST;
79    PR8:  MOV P0,#80H;
80          MOV P1,#80H;
81          MOV P2,#80H;
82          ACALL DEL;
83          AJMP ST;
84    PR9:  MOV P0,#90H;
85          MOV P1,#90H;
86          MOV P2,#90H;
87          ACALL DEL;
88          AJMP ST;
89    END
```

编译通过后，将其烧录到89C51芯片中，将芯片插入到S2型试验板上，通电运行后，右边三个数码管显示000。按一下S1~S12按键中的任意一个，右边三个数码管显示111；再按一下，显示变为222；…按第9下，显示变为999。按第10下起，显示又从111起开始循环。

下面我们对程序进行详细分析解释。

序号1(程序解释，以下同)：程序开始。
序号2：跳转到MAIN主程序处。
序号3：主程序MAIN从地址0030H开始。
序号4~6：P0~P2口（右边三位LED数码管）输出显示000。
序号7：寄存器R0清零。
序号8：P3口置0FH。
序号9：将P3口的状态读至累加器A中。
序号10：判有无按键输入。若A的内容不等于0FH（说明有按键输入），程序跳转至F1处；若A的内容等于0FH(说明无按键输入)，程序顺序执行。
序号11：调用延时子程序，维持数码管点亮。
序号12：程序跳转至ST处。
序号13：调用延时子程序，避开按键的抖动期后再判。
序号14：再判有无按键输入。若A的内容不等于0FH，程序跳转至F2处；若A的内容等于0FH，程序顺序执行。
序号15：程序跳转至ST处。
序号16：寄存器R0内容加1。
序号17：如R0内容不等于0AH(10)，跳转至F3；否则，向下执行。
序号18：寄存器R0清零。
序号19：取直接转移地址表的首地址送DPTR。
序号20：寄存器R0内容送累加器A。
序号21：清除进位位CY。
序号22：累加器A的内容左移一位(相当于x2)。
序号23：判断是否有进位？无进位转NADD；有进位向下执行。
序号24：有进位，DPH加1。
序号25：转向形成散转地址。根据A的内容，跳转至PR1、PR2…。
序号26~36：直接转移地址表。
序号37~43：延时子程序。
序号44~46：P0~P2口显示111。
序号47：调用延时子程序，维持点亮。
序号48：跳转至ST。
序号49~51：P0~P2口显示222。
序号52：调用延时子程序，维持点亮。
序号53：跳转至ST。
序号54~56：P0~P2口显示333。
序号57：调用延时子程序，维持点亮。
序号58：跳转至ST。
序号59~61：P0~P2口显示444。
序号62：调用延时子程序，维持点亮。
序号63：跳转至ST。
序号64~66：P0~P2口显示555。
序号67：调用延时子程序，维持点亮。
序号68：跳转至ST。
序号69~71：P0~P2口显示666。
序号72：调用延时子程序，维持点亮。
序号73：跳转至ST。
序号74~76：P0~P2口显示777。
序号77：调用延时子程序，维持点亮。
序号78：跳转至ST。
序号79~81：P0~P2口显示888。
序号82：调用延时子程序，维持点亮。
序号83：跳转至ST。
序号84~86：P0~P2口显示999。
序号87：调用延时子程序，维持点亮。
序号88：跳转至ST。
序号89：程序结束。

总结归纳：这段程序基本概括了讲座(六)所学的指令。开始时进行初始化，将右边三位数码管(P0~P2)显示为0，同时清除R0。随后向P3送数0FH，准备读取并判断有无按键输入。无键输入，则反复循环判读。有键输入，R0内容增1(如增加到10，则清除后重来)。然后再将直接转移地址表的首地址送16位数据指针DPTR中。同时将R0内容送累加器A（由于随后的转移指令AJMP PR1~9为双字节长度，故需将A乘2进行修正)，其后用JMP@A+DPTR指令进行变址寻址。由于R0中的内容从1起开始执行散转，故在直接散转地址表的开始处安排两条单字节指令NOP进行修正(若R0中的内容从0起开始执行散转则无此必要)。随即程序依R0内容不同(1~9)散转至PR1~PR9处执行。通过对上面指令的学习并结合对这段程序理解，读者会基本了解散转程序的结构。

(下一讲继续介绍指令的学习及实验)。

需要单片机学习资料及器材的读者的读者请见上期本文的配文广告。 ◀