3.8 控制转移指令

通常情况下，程序的执行是顺序进行的，但也可以根据需要改变程序的执行顺序，这种情况称作程序转移。 控制程序的转移要利用转移指令。转移类指令的共同特点是可以改变程序执行的顺序，使CPU转移到另一处执行，或者是继续顺序地执行。无论是哪一类指令，执行后都以改变程序计数器PC中的值为目标。

转移类指令分为四类：无条件转移、条件转移、调用指令及返回指令，共计有21条指令，另外还有一条NOP指令。除NOP指令执行时间为一个机器周期外，其它转移指令的执行时间都是两个机器周期。

助记符18种：AJMP、LJMP、SJMP、JMP；JZ、JNZ、CJNE、DJNZ；JC、JNC、JB、JNB、JBC；ACALL、LCALL；RET、RETI；NOP。

有了丰富的控制转移类指令，就能很方便地实现程序的向前、 向后跳转，并根据条件分支运行、 循环运行、 调用子程序等。

3.8 控制转移指令

3.8.1 无条件转移指令

不规定条件的程序转移称之为无条件转移。 51系列共有4条无条件转移指令，它们提供了不同的转移范围和寻址方式。执行指令后程序的执行顺序是必须转移的。

一、短跳转指令（Absolute Jump）

AJMP addr11；PC←（PC）+2，（PC）10～0←addr11

该指令执行时，先将PC的内容加2（这是PC指向的是AJMP的下一条指令），然后把指令中11位地址码传送到PC10~0，而PC15~11保持原内容不变。需要注意的是，由于AJMP是双字节指令，当程序转移时PC的内容加2，因此转移的目标地址应与AJMP下相邻指令第一字节地址在同一双字节范围。本指令不影响标志位。

在目标地址的11位中，前3位为页地址，后8位为页内地址（每页含256个单元）。当前PC的高5位（即下条指令的存储地址的高5位）可以确定32个2KB段之一。所以，AJMP指令的转移范围为包含AJMP下条指令在内的2KB区间。

3.8 控制转移指令

例如，程序存储器的2070H地址单元有绝对转移指令：

2070H AJMP 16AH（00101101010B）

程序计数器PC当前=PC+2=2070H+02H=2072H（0010 0000 0111 0010），取PC当前的高5位00100和指令给出的11位地址00101101010最后形成的目的地址为：0010 0001 0110 1010B=216AH。

二、相对转移指令（Short Jump）

SJMP rel ；PC ←（PC）+ 2，PC ←（PC）+ rel

第一字节为操作码，第二字节为相对偏移量 rel，rel 是一个带符号的偏移字节数（2的补码），取值范围为+127～-128（00H～7FH对应表示0～+127，80H～FFH对应表示-128～-1）。负数表示反向转移，正数表示正向转移。 转移范围为当前PC值的-128～+127范围内，共256个单元。

3.8 控制转移指令

rel 可以是一个转移目标地址的标号，由汇编程序在汇编过程中自动计算偏移地址，并填入指令代码中。在手工汇编时，可用转移目标地址减转移指令所在的源地址，再减转移指令字节数2得到偏移字节数rel。

若偏移量rel取值为FEH（-2的补码），则目标地址等于源地址，相当于动态停机，程序终止在这条指令上，停机指令在调试程序时很有用。51系列单片机没有专用的停机指令，若要求动态停机可用SJMP指令来实现：

HERE：SJMP　HERE；　动态停机

或写成HERE：SJMP　$　；“$”表示本指令首字节所在单元的地址，使用它可省略标号。

例　若标号“NEWADD”表示转移目标地址0123H，PC的当前值为0100H。执行指令 SJMP NEWADD 后，程序将转向0123H 处执行（此时re l= 0123H－（0100＋2）= 21H）。

3.8 控制转移指令

三、长跳转指令（Long Jump）

LJMP addr16 ； PC ← addr16

第一字节为操作码，该指令执行时，将指令的第二、三字节地址码分别装入指令计数器PC的高8位和低8位中，程序无条件地转移到指定的目标地址去执行。不影响任何标志。

LJMP提供的是16位地址，因此程序可以转向64KB的程序存储器地址空间的任何单元，因此称之为“长转移”。

例如，若标号“NEWADD”表示转移目标地址1234H。执行指令 LJMP NEWADD 时，两字节的目标地址将装入PC中，使程序转向目标地址 1234H 处运行。

四、基寄存器加变址寄存器间接转移指令（散转指令）

JMP @A+DPTR ； PC ←（PC）+ 1，PC ←（A）+（DPTR）

功能：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，其和作为下一条指令的地址送入PC。其特点是转移地址可以在程序运行中加以改变。

3.8 控制转移指令

该指令具有散转功能，可以代替许多判别跳转指令。该指令执行时对标志位无影响。

例 有一段程序如下：

```
        MOV   DPTR，#TABLE
        JMP   @A+DPTR
TABLE：AJMP   ROUT0
        AJMP   ROUT1
        AJMP   ROUT2
        AJMP   ROUT3
```

当（A）=00H时，程序将转到 ROUT0处执行；当（A）=02H时，程序将转到 ROUT1处执行；其余类推。

3.8 控制转移指令

3.8.2 条件转移指令（判跳指令）

条件转移指令是当某种条件满足时，程序转移执行；条件不满足时，程序仍按原来顺序继续执行。 条件转移的条件可以是上一条指令或者更前一条指令的执行结果（常体现在标志位上），也可以是条件转移指令本身包含的某种运算结果。 本类指令有13条。

一、测试条件符合转移指令

1）累加器判零/非零转移指令（Jump on Zero/Not Zero 2条）

助记符　　转移条件

JZ　rel　；（A）=0，则（PC）←（PC）+2+rel 转移

；（A）≠0，则（PC）←（PC）+2

JNZ　rel　；（A）≠0，则（PC）←（PC）+2+rel 转移

；（A）=0，则（PC）←（PC）+2

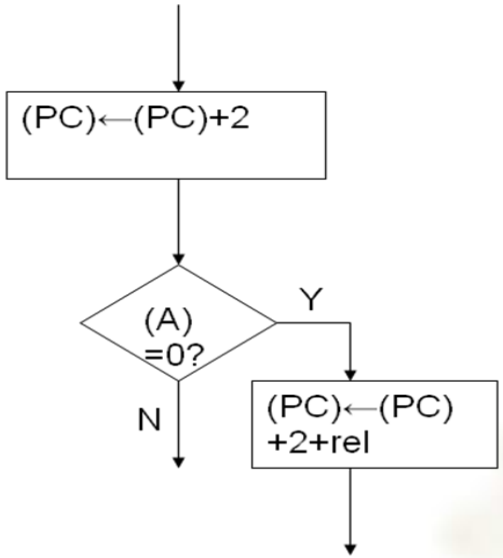
功能：对累加器A的内容为 0 和不为 0 进行检测并转移。当不满足各自的条件时，程序继续往下执行。当各自的条件满足时，程序转向指定的目标地址。目标地址的计算与SJMP指令情况相同。指令执行时对标志位无影响。

3.8 控制转移指令

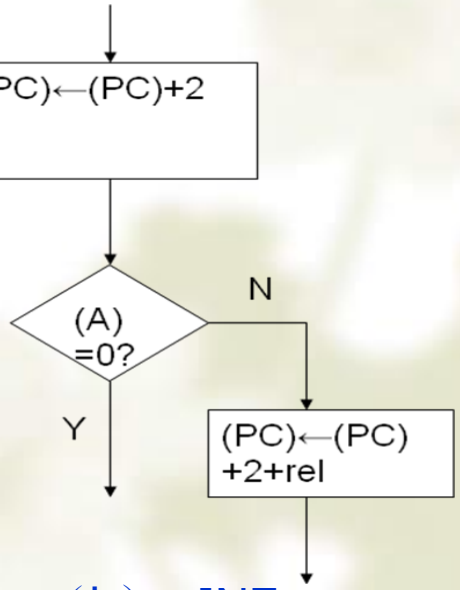
rel=目标地址-PC的当前值

注意：差值的最高两位必须为00H或FFH，否则超出偏移量允许范围。rel取低两位。偏移量rel是用补码形式表示的带符号的8位数，程序转移的目标地址为当前PC值的后128 B与前 127 B之间。

指令不改变原累加器内容，不影响标志位。转移的目标地址在以下一条指令的起始地址为中心的256个字节范围之内（－128～+127）。当条件满足时，（PC）←（PC）+2+ rel，其中（PC）为该条件转移指令的第一个字节的地址。



（a） JZ



（b） JNZ

3.8 控制转移指令

例 若累加器A原始内容为00H，则：

JNZ L1 ；由于A的内容为00H，所以程序往下执行

INC A ；

JNZ L2 ；由于A的内容已不为0，所以程序转向L2 处执行

2）位条件转移

位条件转移指令是以进位标志CY或者位地址bit的内容作为是否转移的条件，共有 5 条指令。

- 以C状态为条件的转移指令（进位/无进位转移指令 Jump on Carry/Not Carry 2条）

JC rel ；（CY）=1，则（PC）←（PC）+2+rel 转移

；（CY）=0，则（PC）←（PC）+2

JNC rel ；（CY）= 0，则（PC）←（PC）+2+rel 转移

；（CY）= 1，则（PC）←（PC）+2

功能：对进位标志位CY进行检测，当（CY）=1（第一条指令）或（CY）=0（第二条指令），程序转向PC当前值（+2后）与rel之和的目标地址去执行，否则程序将顺序执行。转移的范围是-128～+127 B。双字节双周期指令。

3.8 控制转移指令

例 设（C）＝0，执行指令：

```
JC  LABEL1     ；（CY）=0，则程序顺序往下执行
CPL  C         ；（CY）=1，程序转LABEL2
JC  LABEL2
```

执行结果：进位位取反变为1，程序转向LABEL2单元执行。

例 设（C）＝1，执行指令：

```
JNC   LABEL1
CLR   C
JNC   LABEL2
```

执行结果：进位位清为0，程序转向LABEL2单元执行。

这两条指令常和比较条件转移指令CJNE一起使用，先由CJNE指令判别两个操作数是否相等，若相等就顺序执行；若不相等则依据两个操作数的大小置位或清零CY，再由JC 或JNC指令根据CY的值决定如何进一步分支，从而形成三分支的控制模式（详见3.9程序设计方法）。

3.8 控制转移指令

- 以位状态为条件的转移指令（位测试指令 Jump on Bit/Not Bit 3条）

JB bit，rel ；（bit）=1，则（PC）←（PC）+3+rel 转移

；（bit）=0，则（PC）←（PC）+3

JNB bit，rel ；（bit）=0，则（PC）←（PC）+3+rel 转移

；（bit）=1，则（PC）←（PC）+3

JBC bit，rel ;（bit）=1，则（PC）←（PC）+3+rel 转移且（bit）←0

;（bit）=0，则（PC）←（PC）+3

功能：对指定位bit进行检测，当（bit）=1（第一和第三条指令）或（bit）=0（第二条指令），程序转向PC当前值（+2后）与rel之和的目标地址去执行（相当于一条相对转移指令），否则程序将顺序执行。对于第三条指令，当条件满足时（指定位为1），还具有将该指定位清0的功能，一条指令起到了两条指令的作用。转移的范围是-128～+127 B。操作不影响标志位。三字节双周期转移指令。

3.8 控制转移指令

例 设累加器 A中的内容为FEH（11111110B），

JB ACC.0，LABEL1 ；ACC.0=0，程序顺序往下执行

JB ACC.1，LABEL2 ；ACC.1=1，转LABEL2单元执行

例 设累加器A中的内容为FEH（11111110B），

JNB ACC.1，LABEL1 ；ACC.1=1，程序顺序往下执行

JNB ACC.0，LABEL2 ；ACC.0=0，程序转向LABEL2单元执行

例 设累加器A中的内容为7FH（01111111B），执行指令：

JBC ACC.7，LABEL1 ；ACC.7=0，程序顺序往下执行

JBC ACC.6，LABEL2 ；ACC.6=1，程序转向LABEL2单元执行，

；并将ACC.6位清为0。

执行结果：（A）=3FH（00111111B）。

3.8 控制转移指令

二、比较不相等转移指令（Compare Jump on Not Equal 4条）

数值比较转移指令把两个操作数进行比较，比较结果作为条件来控制程序转移。比较转移指令共有 4 条，其一般格式为：

CJNE目的操作数，源操作数，rel

这组指令是先对两个规定的操作数进行比较，根据比较的结果来决定是否转移到目的地址。具体如下：

CJNE A，direct，rel

CJNE A，#data，rel

CJNE Rn，#data，rel

CJNE @Ri，#data，rel

这组指令的功能是比较两个操作数的大小，则转移。如果第一操作数小于第二操作数，则置位进位标志CY，否则，清“0”CY。不影响任何一个操作数的内容。

3.8 控制转移指令

这 4 条指令的含义分别为：

第 1 条指令：累加器内容与立即数比较，不等则转移；

第 2 条指令：累加器内容与内部RAM（包括特殊功能寄存器）内容比较，不等则转移；

第 3 条指令：内部RAM内容与立即数比较，不等则转移；

第 4 条指令：工作寄存器内容与立即数比较，不等则转移。

以上 4 条指令的差别仅在于操作数的寻址方式不同，均完成以下操作：

若目的操作数=源操作数，则（PC）←（PC）+3，进位标志位清“0”

（CY）=0。

若目的操作数>源操作数，则（PC）←（PC）+3 +rel，进位标志位清“0”

（CY）=0。

若目的操作数<源操作数，则（PC）←（PC）+3 +rel，进位标志位置“1”

（CY）=1。

说明：

（1）比较不相等转移指令为3字节、3操作数相对转移指令。程序转移的范围是从（PC）+3为起始的-128B～+127B的单元地址。

3.8 控制转移指令

（2）指令执行过程中的比较操作实际上是减法操作，但不保存两数之差，产生CY标志。

（3）若参加比较的两个操作数X和Y是无符号数，则可以直接根据指令执行后产生的CY来判断两个操作数的大小。若（CY）=0，则X≥Y；若（CY）=1，则X＜Y。

（4）当参加比较的两个操作数X和Y是有符号数补码时：

若X＞0且Y＜0，则X＞Y；

若X＜0且Y＞0，则X＜Y；

若X＞0且Y＞0（或X＜0且Y＜0）时，则需对比较条件转移中产生的CY值进一步判断。若（CY）=0，则X＞Y； 若（CY）=1，则X＜Y。

例 当 P1 口输入为 3AH时，程序继续进行，否则等待，直至P1口出现3AH。 参考程序如下：

```
        MOV  A，＃3AH    ；立即数3A送A
WAIT：CJNE  A，  P1，WAIT ；（P1）≠3AH，则等待
```

3.8 控制转移指令

三、减1不为0转移指令（Decrease Jump on Not Zero 2条）

把减1与条件转移两种功能结合在一起的指令。共两条：

- 寄存器减1条件转移指令

  DJNZ Rn，rel

2字节指令，其功能为：寄存器内容减1，如所得结果为0，则程序顺序执行，如没有减到0，则程序转移。具体表示如下：

Rn←（Rn）-1；若（Rn）≠0，则PC←（PC）+2+rel

；若（Rn）=0，则PC←（PC）+2

- 直接寻址单元减1条件转移指令

  DJNZ direct，rel

3字节指令，其功能为：直接寻址单元内容减1，如所得结果为0，则程序顺序执行；如没有减到0，则程序转移。具体表示如下：

direct←（direct）-1；若（direct）≠0，则PC←（PC）+3+rel

；若（direct）=0，则PC←（PC）+3

3.8 控制转移指令

要点分析：

（1）这组指令把源操作数减1，结果送回源操作数中去。如果结果不为0，则转移到规定的地址单元，否则顺序执行。转移的目标地址是在以PC当前值为中心的-128～+127 的范围内。 如果源操作数内容为 00H， 则执行该组指令后，结果为FFH，但不影响任何状态标志。

（2）这两条指令主要用于控制程序循环。如预先把寄存器或内部RAM单元赋值循环次数，则利用减1条件转移指令，以减1后是否为0作为转移条件，即可实现按次数控制循环。

例 有一段程序如下：

```
        MOV  23H，#0AH
        CLR   A
LOOPX：ADD   A，23H
        DJNZ  23H，LOOPX
        SJMP  $
```

该程序执行后：

（A）=10+9+8+7+6+5+4+3+2+1

=37H

例 从P1.7引脚输出5个方波。

```
    MOV  R2，#10；5个方波，10个状态
LOP：CPL  P1.7 ；P1.7状态变反
        DJNZ  R2 ，LOP
```

3.8 控制转移指令

条件转移指令综合举例

例3-24 已知20H中有一无符号数X，若它小于50，则转向LOOP1执行；若它等于50，则转向LOOP2执行；若它大于50，则转向LOOP3执行，试编写相应程序。

解：程序如下：

```
        MOV   A，20H              ；（A）←X
        CJNE  A，#50，COMP  ；若X≠50，则转移到COMP，产生CY 标志
        SJMP   LOOP2              ；若X＝50，则转移到LOOP2
COMP：JNC    LOOP3              ；若X＞50，则转移到LOOP3
LOOP1：…                  ；LOOP1程序段
LOOP2：…                  ；LOOP2程序段
LOOP3：…                  ；LOOP3程序段
        END
```

3.8 控制转移指令

例 3-25 已知内部RAM的M1和M2单元中各有一个无符号8位二进制数。试编程比较它们的大小，并把大数送到MAX单元。

解：程序如下：

```
        MOV   A，M1        ；（A）←（M1）
        CJNE  A，M2，LOOP ；若（A）≠（M2），则转移到LOOP，产生CY标志
LOOP：  JNC   LOOP1        ；若（A）≥（M2），则转移到LOOP1
        MOV   A，M2        ；若（A）<（M2），则（A）←（M2）
LOOP1： MOV   MAX，A       ；大数→ MAX
```

思考： 送小数到MAX单元如何实现？

3.8 控制转移指令

例 3-26 编写程序将内部RAM从DATA单元开始的10个无符号数相加，相加结果送SUM单元保存（假设结果不超过8位二进制数）。

解：程序如下：

```
        MOV   R0，#0AH      ；设置循环次数
        MOV   R1，#DATA     ；R1作地址指针，指向数据块首地址
        CLR   A             ；A清零
LOOP：ADD   A，@R1          ；加一个数
        INC    R1           ；修改指针，指向下一个数
        DJNZ  R0， LOOP     ；R0减1不为0，继续循环
        MOV   SUM，A        ；存10个数相加的和
```

3.8 控制转移指令

例 3-27 编写程序，统计片内RAM 30H单元开始的20个带符号数中负数的个数，结果存入50H单元。

解：程序如下：

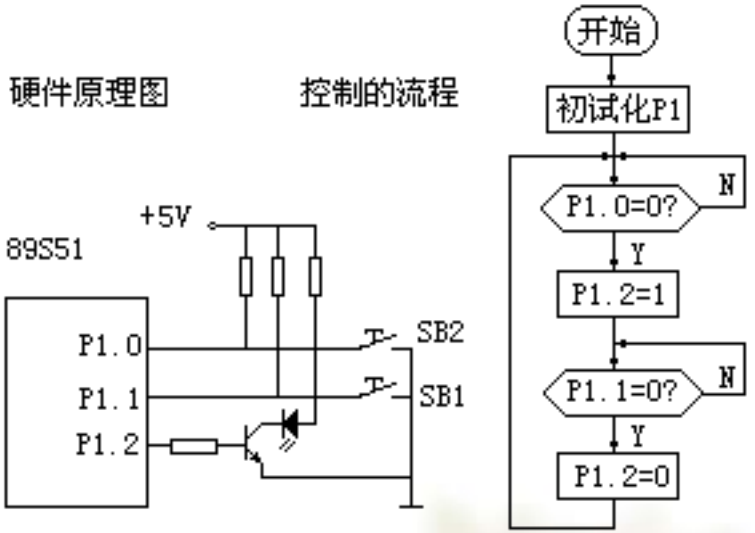
```
        MOV  R7，#20    ；循环次数存R7
        MOV  R3，#0     ；计数变量清0
        MOV  R0，#30H   ；数据单元首地址存R0
LOOP：  MOV  A，@R0     ；取数据至A
        RLC   A         ；带进位向左循环移1位
        JNC   L1        ；CY=0（非负数）转L1
        INC   R3        ；CY=1，负数，统计，（R3）←（R3）+1
L1：    INC   R0        ；修改R0，取下一个数
        DJNZ  R7，LOOP；（R7）←（R7）-1，若（R7）≠0继续循环
        MOV  50H，R3
        SJMP  $
```

思考： 1.判定负数有无另外方法？

2. 统计正数、零和非零数的个数如何实现？

3.8 控制转移指令

例3-28 利用P1.0、P1.1作为外接发光二极管的启停按钮，P1.2作为外接发光二极管端，试编制控制程序。



解：程序如下：

```
START：MOV   P1，#03H；  P1口作输入时，端口锁存器先置1
  WT1：JB    P1.0，WT1
       SETB  P1.2
  WT2：JB    P1.1，WT2
       CLR   P1.2
       SJMP  WT1
```

3.8 控制转移指令

例 3-29 将外部数据RAM的一个数据块传送到内部数据RAM，两者的首址分别为DATA1 和 DATA2，遇到传送的数据为零时停止。

解: 外部RAM向内部RAM的数据传送一定要以累加器A作为过渡，利用判零条件转移正好可以判别是否要继续传送或者终止。 完成数据传送的参考程序如下：

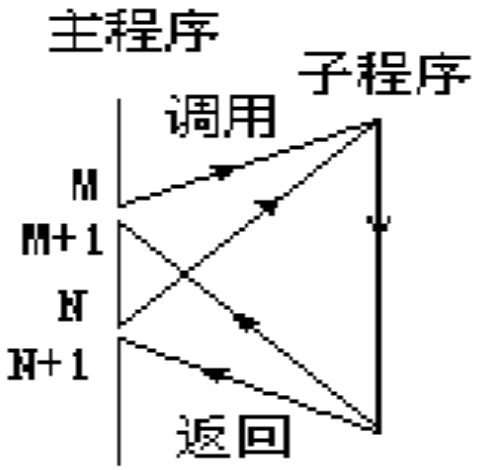
```
        MOV   R0，＃DATA1   ；外部数据块首址送R0
        MOV   R1，＃DATA2   ；内部数据块首址送R1
LOOP：  MOVX   A，@R0       ；取外部RAM数据入A
HERE：  JZ  HERE            ；数据为零则终止传送
        MOV @R1，  A        ；数据传送至内部RAM单元
        INC  R0             ；修改地址指针，指向下一数据地址
        INC  R1
        SJMP  LOOP          ；循环取数
```

3.8 控制转移指令

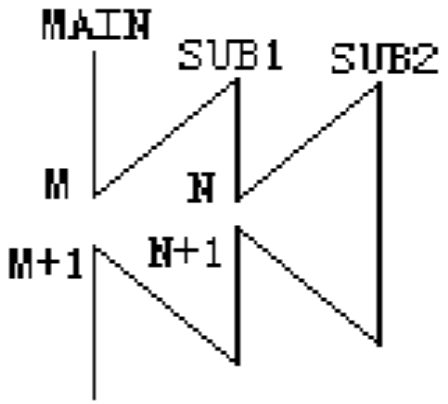
3.8.3 调用和返回指令

子程序结构，即把重复的程序段编写为一个子程序，通过主程序调用而使用它。减少了编程工作量，缩短了程序的长度。当主程序需要调用子程序时用调用指令，将程序转向子程序的入口地址。而在子程序的最后安排一条子程序返回指令，以便执行完子程序后能返回主程序继续执行。按两者的关系有多次调用和子程序嵌套两种调用情况。

执行调用指令时，CPU自动将当前PC值（该值也称断点地址）压入堆栈中，并自动将子程序入口地址送入程序计数器PC中；当执行返回指令时，CPU自动把堆栈中的断点地址恢复到程序计数器PC中。



1）二次调用



2）二级子程序嵌套

3.8 控制转移指令

一、短调用指令（Absolute Call 1条）

ACALL addr11 ；（PC）←（PC）+2

；（SP）←（SP）+1，（SP）←PC7～PC0

；（SP）←（SP）+1，（SP）←PC15～PC8

；PC10～0←addr11

所调用的子程序的首地址必须与ACALL后面指令的第1个字节在同一个2 KB区域内，其构造目的地址是在PC+2的基础上，以指令提供的11位地址取代PC的低11位，而PC的高5位不变。

例 若（SP）=07H，标号“XADD”表示的实际地址为0345H，PC的当前值为0123H。执行指令 ACALL XADD 后，（PC）+2=0125H，其低8位的25H压入堆栈的08H单元，其高8位的01H压入堆栈的09H单元。（PC）=0345H，程序转向目标地址0345H 处执行。

例 设（SP）＝60H，（PC）＝0123H，子程序SUBRTN的首地址为0456H。执行指令 ACALL SUBRTN

执行结果为（PC）+2=0123H＋2=0125H→（PC），将（PC）=0125H压入堆栈：25H压入（SP）+1=61H，01H压入（SP）+l=62H，(SP）=62H。addr11→PC10～0，PC=0456H。

3.8 控制转移指令

二、长调用指令（Long Call 1条）

LCALL addr16 ；（PC）←（PC）+3

；（SP）←（SP）+1，（SP）←PC7～PC0

；（SP）←（SP）+1，（SP）←PC15～PC8

；（PC）←addr16

执行时：

（1）（PC）+ 3 →（PC），并压入堆栈，先压入PC的低8位，后压入PC的高8位。

（2）addr16 → PC，获得子程序起始地址；

（3）可调用64KB地址范围内的任意子程序。

功能：指令执行后，断点进栈保存，addr16作为子程序起始地址，编程时可用标号代替。

例 设（SP）＝60H，（PC）＝0123H，子程序SUBRTN的首地址为3456H。 执行指令：LCALL SUBRTN

执行结果：（PC）+3=0123H+3=0126H→（PC）。将（PC）压入堆栈：26H压入（SP）+l=61H中，01H压入（SP）+1=62H中，此时（SP）=62H。（PC）=3456H，执行子程序。

3.8 控制转移指令

例 设（SP）=30H，标号为SUB1 的子程序首址在2500H，（PC）=3000H。执行指令：3000H：LCALL SUB1；

结果为（SP）=32H，（31H）=03H，（32H）=30H，（PC）=2500H。

调用指令与转移指令的异同：

（1）两条指令可以实现子程序的短调用和长调用。目标地址的形成方式与AJMP和LJMP相似。LCALL和ACALL指令类似于转移指令LJMP和AJMP，不同之处在于转移指令不保存返回地址，而子程序调用指令在转向目的地址的同时，必须保留返回地址（也称为断点地址），以便执行返回指令时回到主程序断点的位置。通常采用堆栈技术保存断点地址，这样可以允许多重子程序调用，即在子程序中再次调用子程序。子程序调用指令在转移前要把执行完该指令的PC内容自动压入堆栈后，才将addr16（或addr11）送往PC， 即把子程序的入口地址装入PC。

（2）LCALL与LJMP一样提供 16 位地址，可调用 64 KB 范围内所指定的子程序。 由于该指令为三字节指令，所以执行该指令时首先把（PC）+3→（PC），以获得下一条指令地址，并把此时PC内容压入堆栈（先压入低字节，后压入高字节）作为返回地址，堆栈指针SP加2指向栈顶，然后把目的地址 addr16 装入PC。 执行该指令不影响标志位。

3.8 控制转移指令

（3）ACALL与AJMP一样提供 11 位目的地址。 由于该指令为两字节指令，所以执行该指令时（PC）+2→ (PC)以获得下一条指令的地址，并把该地址压入堆栈作为返回地址。该指令可寻址 2 KB, 只能在与PC同一2 KB 的范围内调用子程序。 执行该指令不影响标志位。

（4）堆栈是内部RAM中一片存储区，采用先进后出的原则存取数据，调用时保护断点的工作由调用指令完成，调用后恢复断点的工作由返回指令完成。

（5）ACALL指令执行时，被调用的子程序的首址必须设在包含当前指令（即调用指令的下一条指令）的第一个字节在内的2K字节范围内的程序存储器中。LCALL指令执行时，被调用的子程序的首址可以设在64K字节范围内的程序存储器空间的任何位置。

3.8 控制转移指令

三、返回指令（2条）

返回指令能自动恢复断点，将原压入堆栈的PC值弹回到PC中，保证回到断点处继续执行主程序。返回指令必须用在子程序或中断服务子程序的末尾。

（1）子程序的返回RET （Return）

RET ；PC15～PC8←（SP），（SP）←（SP）-1

；PC7～PC0←（SP），（SP）←（SP）-1

功能：执行时表示结束子程序，返回主程序ACALL或LCALL的下一条指令（即断点地址），继续往下执行。将栈顶的断点地址送入PC（先PCH，后PCL），并把栈指针减2。

例 设（SP）=62H，RAM中的（62H）=0lH，（61H）=26H。执行指令：RET，（SP）=60H，（PC）=0126H。

（2）中断返回指令RETI（Return for Interrupt）

RETI ；PC15～PC8←（SP），（SP）←（SP）-1

；PC7～PC0←（SP），（SP）←（SP）-1

3.8 控制转移指令

功能：1）将堆栈顶部2字节的内容送到PC中，从中断服务程序返回中断时保护的断点处继续执行程序（类似RET功能）；

2） 清除内部相应的中断状态寄存器。

该指令用于中断服务程序的末尾，即中断服务程序必须以RETI为结束指令。CPU执行RETI指令后至少再执行一条指令，才能响应新的中断请求。可用来实现单片微机的单步操作。

与RET指令不同之处：

RET指令的功能是从堆栈中弹出由调用指令压入堆栈保护的断点地址，并送入指令计数器PC，从而结束子程序的执行，程序返回到断点处继续执行。

RETI指令是专用于中断服务程序返回的指令，除正确返回中断断点处执行主程序以外，并有清除内部相应的中断状态寄存器（以保证正确的中断逻辑）的功能。

例 设（SP）=62H，中断时断点是0123H，RAM中的（62H）=0lH，（61H）=23H。

执行指令RETI结果为：（SP）=60H，PC=0123H。

程序回到断点0123H处继续执行。清除内部相应的中断状态寄存器。

3.8 控制转移指令

四、空操作指令（1条）

NOP ；（PC）←（PC）+1

说明：空操作指令也算一条控制指令，即控制CPU不作任何操作，只消耗一个机器周期的时间。空操作指令是单字节指令，因此执行后PC加1，时间延续一个机器周期。不影响任何标志，故称为空操作指令。在程序中加上几条NOP指令用于设计延时程序，拼凑精确延时时间或产生程序等待等。