第 2 章　51 单片机的指令系统

☞教学要点

51 单片机指令系统的特点是不同的存储空间其寻址方式不同。指令是程序设计的基础。本章重点讲授寻址方式、传送指令、算术运算指令、逻辑运算指令、控制转移指令和位操作指令，要求掌握指令的功能、操作对象和结果，对标志位的影响要求熟记。

计算机通过执行程序完成人们指定的任务，程序由一条条指令构成。能为 CPU 识别并执行的指令的集合就是该 CPU 的指令系统。

51 单片机的指令系统中有数据传送交换类、算术运算类、逻辑运算与循环类、子程序调用与转移类、位操作类和 CPU 控制类等指令。它有如下 3 个特点。

① 指令执行时间快。大多数指令执行时间为 1 个机器周期，少数指令（45 条）为 2 个机器周期，仅乘、除指令为 4 个机器周期。

② 指令短。大多数为 1～2 字节，少数为 3 字节。

③ 具有丰富的位操作指令。可对内部数据 RAM 和特殊功能寄存器中的可寻址位进行多种形式的位操作。

单片机指令的这些特点使之具有极强的实时控制和数据运算功能。

51 单片机的指令格式为：

操作符　目的操作数，源操作数　　；注释

其中，操作符指明该指令完成什么操作，操作数指明该指令的操作对象。目的操作数和源操作数完成操作符规定的操作后，结果存放在目的操作数中。操作数可以是一个具体的数据，也可以是由寄存器或存储器提供的数据，这种提供操作数的方式称为寻址方式。注释可有可无，用分号“；”与指令隔开，用于编程者的说明。

下面先介绍指令系统中的寻址方式，然后分述各类指令，附录 A 中列有指令操作码表。在学习指令之前，把指令中的符号介绍如下。

Rn：当前工作寄存器组中的 R0～R7（其中，n＝0，1，…，7）。

Ri：当前工作寄存器组中的 R0，R1（ 其中，i＝0，1）。

dir：8 位直接字节地址（片内 RAM 和 SFR 地址）。

＃data：8 位立即数。

＃data16：16 位立即数。

addr16：16 位地址值。

addr11：11 位地址值。

bit：位地址（在位地址空间中）。

rel：相对偏移量（在相对转移指令中使用，为一字节补码）。

（）：用于注释中表示存储单元的内容。

2.1 寻 址 方 式

如前所述，寻址方式是指令中提供操作数的形式，它可以是操作数本身，也可以是操作数存放的位置(地址)。51 单片机中，存放数据的存储器空间有 4 种：内部 RAM、特殊功能寄存器 SFR、外部 RAM 和程序存储器 ROM。其中，除内部 RAM 和 SFR 统一编址外，外部 RAM 和程序存储器 ROM 是分开编址的。为了区别指令中操作数所处的地址空间，对于不同存储器中的数据操作，采用不同的寻址方式，这是 51 单片机在寻址方式上的一个显著特点。

2.1.1 立即寻址

指令中直接给出操作数的寻址方式称为立即寻址。在 51 单片机的指令系统中，立即数用一个前面加“#”号的 8 位数(#data，如#30H)或 16 位数(#data16，如#2052H)表示。指令的机器码中立即数在操作码(OP)之后，因此，立即寻址的指令多为 2 字节或 3 字节指令，机器码可由附录 A 查出，以下以传送指令为例说明。

【例 2-1】

```
MOV   A,#80H        ;80H→A,机器码 7480
MOV   DPTR,#2000H   ;2000H→DPTR,机器码为 902000
```

注意：16 位数的存放顺序不同于 8086(上例中的 2000H 在 8086 中为 0020 存放)。

2.1.2 直接寻址

指令中直接给出操作数所在的地址(dir)的寻址方式称为直接寻址。寻址对象为：①内部数据存储器 RAM，在指令中以直接地址表示；②特殊功能寄存器 SFR，在指令中用寄存器名表示。

直接寻址的指令码中应有直接地址字节，因此多为 2 字节或 3 字节指令，当指令中的两个操作数均为直接寻址时，指令码为 op $dir_{源}$ $dir_{目的}$。

【例 2-2】

```
MOV   A,25H         ;内部 RAM(25H)→A,机器码 E525
MOV   P0,#45H       ;45H→P0,P0 为直接寻址的 SFR,其地址为 80H(见表 1-2),机器码
                     为 758045
MOV   30H,20H       ;内部 RAM 的(20H)→(30H),机器码为 852030
```

2.1.3 寄存器寻址

以通用寄存器的内容为操作数的寻址方式称为寄存器寻址。

通用寄存器包括：A，B，DPTR，R_0～R_7。其中，B 仅在乘除法指令中为寄存器寻址，在其他指令中为直接寻址。A 既可以寄存器寻址，又可以直接寻址(此时写作 ACC)。直接寻址和寄存器寻址的差别在于，直接寻址是以操作数所在的字节地址(占 1 字节)出现在指令码中，寄存器寻址是寄存器编码出现在指令码中。由于可寄存器寻址的寄存器少、编码位数少(少于 3 位二进制数)，通常操作码和寄存器编码合用 1 字节，因此寄存器寻址的指令机器码短，执行快。然而除上面所指的几个寄存器外的特殊功能寄存器，一律为直接寻址。

【例 2-3】

```
MOV   A,R0          ;R0→A,A,R0 均为寄存器寻址,机器码为 E8,仅占 1 字节
MUL   AB            ;A*B→BA,A,B 为寄存器寻址,机器码为 A4
MOV   B,R0          ;R0→B,R0 为寄存器寻址,B 为直接寻址,查 MOV dir,Rn 指令,机器码
                     为 88F0,其中 F0 为 B 的字节地址(见表 1-2)
PUSH  ACC           ;A 的内容压入堆栈,机器码为 C0E0,其中 E0 为 A 的字节地址,A 为直接
                     寻址
```

ADD　A,ACC　　;A+A→A,A 和 ACC 同为累加器,A 为寄存器寻址,ACC 为直接寻址,因为指令只有 ADD　A,dir 形式,而没有 ADD　A,A 形式

2.1.4　寄存器间接寻址

以寄存器中的内容为地址,该地址中的内容为操作数的寻址方式称为寄存器间接寻址,简称寄存器间址。能够用于寄存器间址的寄存器有:R0、R1、DPTR,用前面加@表示。如@R0、@R1、@DPTR。

寄存器间接寻址的存储器空间包括内部数据存储器和外部数据存储器。由于内部数据存储器共有 128 字节,因此用一字节的 R0 或 R1 可间接寻址整个空间。而外部数据存储器最大可达 64KB,仅 R0 或 R1 无法寻址整个空间。因此,需由 P_2 端口提供外部 RAM 高 8 位地址,由 R0 或 R1 提供低 8 位地址,由此共同寻址 64KB 范围。也可用 16 位的 DPTR 作寄存器间接寻址 64KB 存储空间。

在指令中,当用@R0 或@R1 寻址时,究竟是对内部 RAM 还是对外部 RAM 寻址,取决于用 MOV 还是用 MOVX 作操作符。

注意:使用间接寻址指令前,需先给作间址的寄存器赋值。

【例 2-4】　MOV　@R0,A　　;A→以 R0 内容为地址的内部 RAM,机器码 F6
　　MOVX　A,@R1　　;外部 RAM(地址为 P2 R1)的内容→A,机器码 F7
　　MOVX　@DPTR,A　;A→以 DPTR 内容为地址的外部 RAM,机器码 F0

以上各例指令的操作分别如图 2-1～图 2-3 所示。

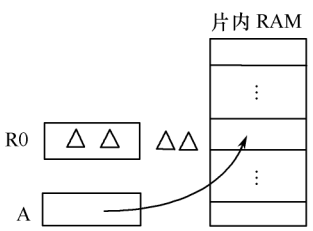

图 2-1　MOV　@R0,A

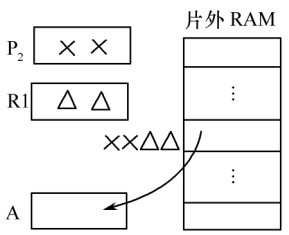

图 2-2　MOVX　A,@R1

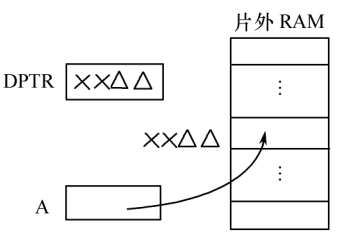

图 2-3　MOVX　@DPTR,A

2.1.5　变址寻址

由寄存器 DPTR 或 PC 中内容加上 A 累加器内容之和,形成操作数地址的寻址方式称为变址寻址。变址寻址只能对程序存储器中的数据作寻址操作。由于程序存储器是只读存储器,因此变址寻址操作只有读操作而无写操作。指令操作符采用 MOVC。

【例 2-5】　MOVC　A,@A+DPTR　　;(A+DPTR)→A
　　MOVC　A,@A+PC　　;(A+PC)→A

以上两例指令的操作如图 2-4 和图 2-5 所示。

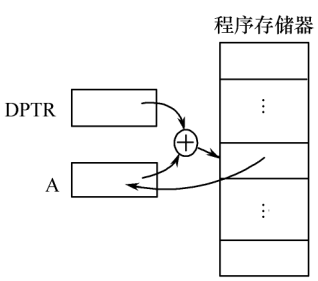

图 2-4　MOVC　A,@A+DPTR

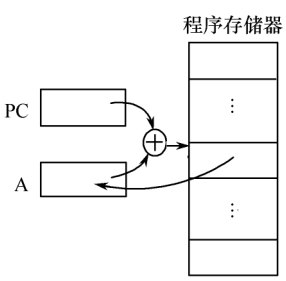

图 2-5　MOVC　A,@A+PC

2.1.6 相对寻址

以当前程序计数器 PC 的内容为基值，加上指令给出的一字节补码数（偏移量）形成新的 PC 值的寻址方式称为相对寻址。

相对寻址只用于修改 PC 之值，故主要用于实现程序的分支转移。如：SJMP 08H；当前 PC＋08H→PC（该指令为 2 字节，这里的当前 PC 是该指令的地址加 2），指令执行后，转移到地址为 PC＋08H（即指令地址＋0AH）处执行程序。

2.1.7 位寻址

对位地址中的内容作位操作的寻址方式称为位寻址。

由于单片机中只有内部 RAM 和特殊功能寄存器的部分单元有位地址，因此，位寻址只能对有位地址的这两个空间进行寻址操作。位寻址是一种直接寻址方式，由指令给出直接位地址。与直接寻址不同的是，位寻址给出的是位地址，而不是字节地址。

【例 2-6】

```
SETB  20H      ;1→20H 位，机器码为 D220
MOV   32H,C    ;进位位 CY（即指令中的 C）→32H 位，机器码为 9232
ORL   C,5AH    ;CY∨5AH 位→CY，机器码为 725A
```

从前面内容可知，51 单片机的寻址方式形式简单，类型少，容易掌握。但由于单片机严格的存储器空间分配，因此，指令中应根据操作数所在的存储空间选用不同的寻址方式。

在直接寻址的内部 RAM 和特殊功能寄存器的操作数中，特殊功能寄存器中的操作数常采用符号字节地址（如 PSW，TMOD，P_0，IE 等）和符号位地址（如 C，RS_0，EA，$P_{1.1}$ 等）的形式，而不用直接字节地址或直接位地址形式。对内部 RAM 则用直接字节地址或直接位地址，如 30H，50H 等。

各种寻址方式的适用范围在表 1-5 中归纳得很清楚，可以查阅。

2.2 数据传送与交换指令

这类指令共有 28 条，包括以 A、Rn、DPTR、直接地址单元、间接地址单元为操作数的指令；访问外部 RAM 的指令；读程序存储器的指令；数据交换指令及堆栈操作指令。

2.2.1 传送类指令

1. 内部 RAM 和 SFR 间的传送指令 MOV

图 2-6 示意了 MOV 的操作，图中，→表示单向传送，↔表示互相传送，箭头指向目的操作数。

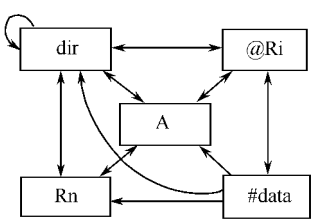

图 2-6 MOV 指令

（1）以 A 为目的操作数（4 条）

```
        ┌ Rn      ;Rn→A
        │ dir     ;dir→A
MOV  A, │ @Ri     ;(Ri) →A
        └ #data   ;#data→A
```

【例 2-7】 R1＝20H，(20H)＝55H，指令 MOV A，@R1 执行后，A＝55H。

(2) 以 Rn 为目的操作数(3 条)

```
MOV   Rn,{ A        ;A→Rn
         { dir      ;dir→Rn
         { #data    ;#data→Rn
```

【例 2-8】 (40H)=30H,指令 MOV　R7,40H 执行后,R7=30H。

(3) 以 DPTR 为目的操作数(1 条)

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

【例 2-9】 执行 MOV　DPTR,#0A123H 指令后,DPTR=A123H。

(4) 以直接地址为目的操作数(5 条)

```
MOV   dir,{ A       ;A→dir
          { Rn      ;Rn →dir
          { dir     ;dir→dir
          { @Ri     ;(Ri) →dir
          { #data   ;#data→dir
```

【例 2-10】 R0=50H,(50H)=10H,指令 MOV　35H,@R0 执行后,(35H)=10H。这一操作也可用指令 MOV　35H,50H 来完成。

(5) 以间接地址为目的操作数(3 条)

```
MOV   @Ri,{ A       ;A→(Ri)
          { dir     ;dir→(Ri)
          { #data   ;#data→(Ri)
```

【例 2-11】 R0=50H,执行指令 MOV　@R0,#67H 后,(50H)=67H。

2. 外部存储器和 A 累加器之间的传送

外部数据存储器及程序存储器只能和 A 之间进行传送,而不能与内部 RAM 和 SFR 之间进行传送,指令如图 2-7 所示。

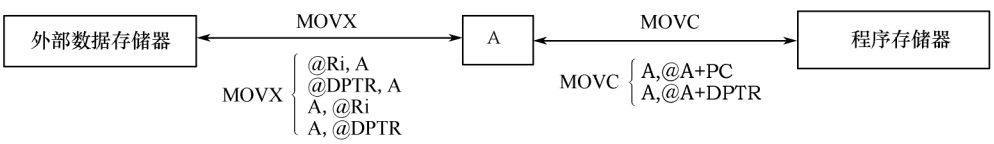

图 2-7　外部存储器的传送

(1) 访问外部数据存储器指令(4 条)

外部数据存储器可读可写,故和 A 可以互相传送,共有 4 条指令。

```
MOVX   @Ri,A        ;A→(P2Ri),其中 P2 指示高 8 位地址,Ri 指示低 8 位地址
MOVX   @DPTR,A      ;A→(DPTR)
MOVX   A,@Ri        ;(P2Ri) →A
MOVX   A,@DPTR      ;(DPTR) →A
```

【例 2-12】 将立即数 23H 送入外部 RAM 0FFFH 单元:

```
MOV   A,#23H
MOV   P2,#0FH
MOV   R1,#0FFH
MOVX  @R1,A
```

或

```
MOV   A,#23H
MOV   DPTR,#0FFFH
MOVX  @DPTR,A
```

(2) 访问程序存储器指令

程序存储器只能读,不能写,故只有两条读指令:

```
MOVC   A,@A+PC          ;(A+PC)→A
MOVC   A,@A+DPTR        ;(A+DPTR)→A
```

这两条指令常用于查表。

【例 2-13】 分析执行下列程序后 A=?

```
        MOV   A,#01H          ;A=01
        MOV   DPTR,#M2        ;M2 的地址送 DPTR
        MOVC  A,@A+DPTR       ;执行完该指令,A=(01+DPTR)=(1+M2)=77H
M1:     RET                   ;子程序返回指令
M2:     DB 66H,77H,88H,99H    ;定义字节数据
```

程序中 DB 为在程序存储器中定义字节伪指令,MOVC 指令把地址为(M2+1)单元的内容送到 A,因此该程序段执行结果:A=77H。

以上程序段也可用下列程序段代替:

```
        MOV   A,#02H          ;A=02
        MOVC  A,@A+PC         ;取完该指令 PC=M1,执行(2+M1)→A
M1:     RET                   ;子程序返回指令,为一字节指令
M2:     DB   66H,77H,88H,99H
```

由于在执行 MOVC 指令时,PC=M1,A+PC=2+M1,RET 指令占 1 字节,(2+M1)=77H,因此,执行 MOVC A,@A+PC 指令后,A=77H。

3. 堆栈操作

堆栈是用于存放暂时需要保留的信息的一片存储区(相当于储藏室),而这片存储区单元的地址必须用寄存器 SP 指示。在 51 单片机中,堆栈被安排在内部 RAM 区,保存信息和取出信息用如下指令:

(1) 入栈操作

```
PUSH   dir          ;SP+1→SP,(dir) →(SP)
```

(2) 出栈操作

```
POP    dir          ;(SP) →(dir),SP-1→SP
```

例如:

```
MOV   A,#90H
MOV   SP,#15H
PUSH  ACC           ; SP=16H,(16H)=90H(这里 A 为直接寻址,写为 ACC),操作见图 2-8(a)
POP   20H           ;(20H)=90H,SP=15H,操作见图 2-8(b)
```

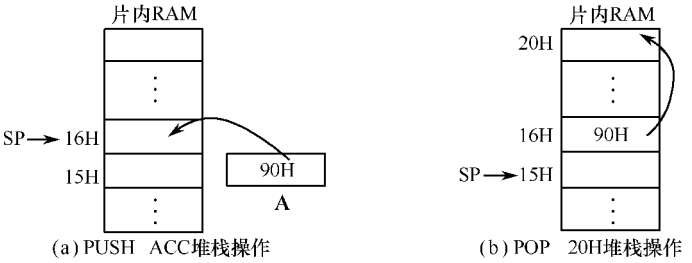

图 2-8 堆栈操作示意图

堆栈操作指令说明:

① 初始化时 SP=07H,若不重置 SP,数据将从 08H 单元开始压入内部 RAM,执行一次指令,只完成一字节数据操作。

② 入栈和出栈操作实际上是数据的传送，只不过每执行一次指令，SP 指示的地址会自动加 1 或减 1。压入堆栈时，SP 先加 1，数据再压入；弹出堆栈时，数据先弹出，SP 再减 1。这种地址的修改是单片机内部硬件自动完成的，不需要用户用 INC 或 DEC 指令完成。

③ 如果要恢复一系列原来堆栈中保存的信息，应遵从后进先出的原则。例如：

```
    ⋮
PUSH   ACC
PUSH   PSW
PUSH   30H          ;以上 3 条入栈保存 ACC、PSW、30H 单元的内容
    ⋮
POP    30H
POP    PSW
POP    ACC          ;以上 3 条还原 ACC、PSW、30H 单元的内容
    ⋮
```

2.2.2 交换指令

1. 字节交换指令(3 条)

$$\text{XCH}\quad A,\begin{cases} Rn & ;Rn \leftrightarrows A \\ dir & ;dir \leftrightarrows A \\ @Ri & ;(Ri) \leftrightarrows A \end{cases}$$

字节交换指令如图 2-9 所示。例如，A＝FFH，R1＝30H，(30H)＝87H，执行 XCH　A，@R1后，A＝87H，(30H)＝FFH。

2. 低半字节交换

XCHD　A，@Ri　　；$A_{0\sim3} \leftrightarrows (Ri)_{0\sim3}$

内部 RAM(Ri)的低 4 位和 A 的低 4 位交换。例如，A＝34H，(50H)＝96H，执行：

```
MOV   R1，#50H
XCHD  A，@R1
```

后，A＝36H，(50H)＝94H，操作如图 2-10 所示。

3. A 的高、低半字节交换

SWAP　A　　；$A_{0\sim3} \leftrightarrows A_{4\sim7}$

操作如图 2-11 所示。例如，A＝0FH，执行 SWAP　A 后，A＝F0H。

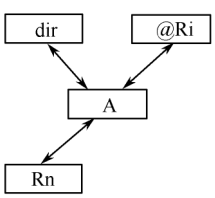

图 2-9　XCH 指令

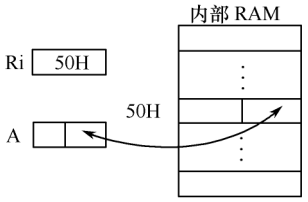

图 2-10　XCHD　A，@Ri 示意图

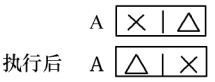

图 2-11　SWAP　A

2.3　算术运算和逻辑运算指令

51 单片机指令系统中算术运算有加、进位加(两数相加后还加进位位 CY)、借位减(两数相减后还减去借位位 CY)、加 1、减 1、乘、除指令；逻辑运算有与、或、异或指令。

2.3.1 算术运算和逻辑运算指令对标志位的影响

在 51 单片机的程序状态字 PSW 寄存器中，有 4 个测试标志位：P(奇偶)、OV(溢出)、CY(进位)、AC(辅助进位)。算术运算、逻辑运算指令对标志位的影响和 8086 微型计算机有所不同，归纳如下。

① P(奇偶)标志仅对 A 操作的指令有影响，凡是对 A 操作的指令(包括传送指令)都将 A 中“1”个数的奇偶性反映到 PSW 的 P 标志位上。即 A 中奇数个“1”，P=1；偶数个“1”，P=0。

② 传送指令、加 1、减 1 指令、逻辑运算指令不影响 CY、OV、AC 标志位。

③ 加、减运算指令影响 P、OV、CY、AC 4 个测试标志位，乘、除指令使 CY=0，当乘积大于 255，或除数为 0 时，OV =1。

具体指令对标志位的影响可参阅附录 A。标志位的状态是控制转移指令的条件，因此指令对标志位的影响应该记住。

2.3.2 以 A 为目的操作数的算术运算和逻辑运算指令

以 A 为目的操作数的算术运算和逻辑运算指令如图 2-12 所示。

加：	ADD	A，	@Ri
进位加：	ADDC		dir
借位减：	SUBB		Rn
与：	ANL		#data
或：	ORL		
异或：	XRL		

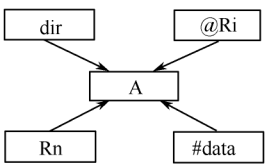

图 2-12 以 A 为目的操作数的算术运算和逻辑运算指令

以 A 为目的操作数指令中，每一类运算对应有 4 个源操作数，共计 24 条指令。对指令说明如下：

① ADDC 指令为进位加(即两数相加后还需加进位位 CY)、SUBB 指令为借位减(即两数相减后还需减去借位位 CY)。应注意的是，减指令只有带借位减，因此在多字节减法中，最低位字节做减法时，注意先清 CY。

② 逻辑运算是按位进行的，因此不影响 CY、OV、AC 标志位。

③ 逻辑运算指令常用于对数据位进行加工。两数逻辑运算的法则是：

与：有“0”则“0”；

或：有“1”则“1”；

异或：同为“0”，异为“1”；与“0”异或值不变；与“1”异或值变反。

例如，设 A=97H，CY=1。

执行 ADD A，#95H 后，A=97H+95H=2CH，标志位 CY=1，OV=1，P=1，AC=0。

执行 ADDC A，#95H 后，A=97H+95H+CY=2DH，标志位 CY=1，OV=1，P=1，AC=0。

例如，设 A=95H，CY=1。执行 SUBB A，#62H 后，A=95H−62H−CY=32H，标志位 CY=0，OV=1，P=1，AC=0。

例如，设 A=8BH，执行 ADD A，ACC 后，A=8BH+8BH=16H，CY=1。

注意：由于指令系统没有 ADD A，A 指令，源操作数 A 作直接寻址时写为 ACC，否则有的编译器会通不过。

例如，A=0FH，执行 ORL A，#80H 后，A=8FH。

例如，A=0FH，(20H)=76H，执行 ANL A，20H 后，A=06H。

2.3.3 以 dir 为目的操作数的逻辑运算指令

以 dir 为目的操作数仅有逻辑运算指令，如图 2-13 所示。

$$\left.\begin{array}{ll}\text{与} & \text{ANL}\\ \text{或} & \text{ORL}\\ \text{异或} & \text{XRL}\end{array}\right\}\text{dir},\quad\left\{\begin{array}{l}\text{A}\\ \#\text{data}\end{array}\right.$$

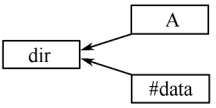

图 2-13 以 dir 为目的操作数的逻辑运算指令

由于每一类运算对应有两个源操作数，共计 6 条指令。

例如，P1＝0FH，执行 XRL P1，#0FFH 后，P1＝F0H（这里 P1 属于直接寻址 dir）。

例如，A＝9BH，(25H)＝39H，执行 ORL 25H，A 后，(25H)＝BBH。

以上 30 条指令这样列出是为了方便读者掌握和记忆。

2.3.4 加 1、减 1 指令

加 1 指令是内部 RAM 或寄存器自增 1 指令，减 1 指令是内部 RAM 或寄存器自减 1 指令，示意图如图 2-14 所示。

加 1 指令格式如下：

$$\text{INC}\left\{\begin{array}{ll}\text{A} & ;\text{A}+1\rightarrow\text{A}\\ @\text{Ri} & ;(\text{Ri})+1\rightarrow(\text{Ri})\\ \text{dir} & ;(\text{dir})+1\rightarrow(\text{dir})\\ \text{Rn} & ;\text{Rn}+1\rightarrow\text{Rn}\\ \text{DPTR} & ;\text{DPTR}+1\rightarrow\text{DPTR}(\text{见图 2-15})\end{array}\right.$$

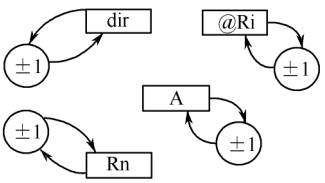

图 2-14 INC，DEC 指令

减 1 指令格式如下：

$$\text{DEC}\left\{\begin{array}{ll}\text{A} & ;\text{A}-1\rightarrow\text{A}\\ @\text{Ri} & ;(\text{Ri})-1\rightarrow(\text{Ri})\\ \text{dir} & ;(\text{dir})-1\rightarrow\text{dir}\\ \text{Rn} & ;\text{Rn}-1\rightarrow\text{Rn}\end{array}\right.$$

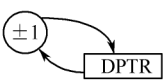

图 2-15 INC DPTR 指令

例如，(20H)＝55H，执行 INC 20H 后，(20H)＝56H。

例如，R7＝80H，执行 DEC R7 后，R7＝7FH。

2.3.5 十进制调整指令

计算机完成二进制数加法，其和也为二进制数，如果是十进制数相加（即 BCD 码相加）想得到十进制数的结果，就必须进行十进制调整（即 BCD 码调整）。调整指令如下：

```
DA  A      ;将 A 中二进制数相加和调整成 BCD 码
```

指令按下列原则进行调整：和低 4 位大于 9 或有半进位，则低 4 位加 6；如果和的高 4 位大于 9 或有进位，则高 4 位加 6。指令根据相加和及标志自行进行判断，因此，该指令应紧跟在加指令之后，至少在加指令和该指令之间不能有影响标志的指令。

DA A 指令只对一字节和进行调整，如为多字节相加，必须进行多次调整。此指令不能对减法结果进行调整。

【例 2-14】 完成 56＋17 的编程。

```
MOV  A,#56H        ;A 存放 BCD 码 56H
MOV  B,#17H        ;B 存放 BCD 码 17H
ADD  A,B           ;A=6DH
```

```
DA   A                    ;A=73H
SJMP   $
```

程序中 SJMP 为相对转移指令，$ 表示本指令首址，用循环执行本转移指令以实现动态停机操作，这是由于 51 单片机没有停机指令；如果不动态停机，将顺序执行后面随机代码而造成死机。

2.3.6 专对 A 的指令

A 取反：　CPL　A　　;$\overline{A}$→A
A清零：　CLR　A　　;0→A
A 左环移：　RL　A　　;见图 2-16
A 右环移：　RR　A　　;见图 2-16
A 左大环移：　RLC　A　　;见图 2-16
A 右大环移：　RRC　A　　;见图 2-16

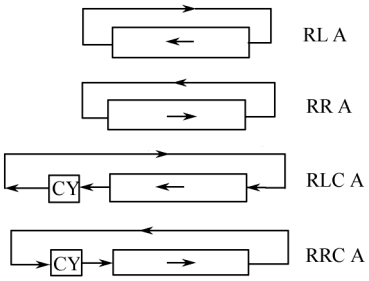

图 2-16　循环移位指令

循环移位指令通常用于位测试、位统计、乘 2、除 2 等操作。

例如，A=84H，执行 RL　A 指令后，A=09H。

例如，A=84H，CY=1，执行 RRC　A 指令后，A=C2H，CY=0。

2.3.7 乘、除法指令

1. 乘法指令

```
MUL   AB          ;A×B→BA
```

说明：本指令实现 8 位无符号乘法。A、B 各放一个 8 位乘数，指令执行后，16 位积的高 8 位在 B 中，低 8 位在 A 中。

例如，A=50H，B=A0H，指令 MUL　AB 执行后，A=00H，B=32H。

2. 除法指令

```
DIV   AB          ;A÷B→商在 A 中，余数在 B 中
```

说明：本指令实现两个 8 位无符号数除法。A 放被除数，B 放除数，指令执行后，A 中为商，B 中为余数。若除数 B=00H，则指令执行后，溢出标志 OV=1，且 A，B 内容不变。

例如，A=28H，B=12H，指令 DIV　AB 执行后，A=02H，B=04H。又如，A=08H，B=09H，指令 DIV　AB 执行后，A=00H，B=08H。

2.3.8 指令综合应用举例

【例 2-15】 编程将 21H 单元的低 3 位和 20H 单元中的低 5 位合并为一个字送 30H 单元，要求(21H)的低 3 位放在高位上。

```
MOV   30H,20H          ;(30H)=(20H)
ANL   30H,#1FH         ;保留低 5 位
MOV   A,21H            ;A=(21H)
SWAP  A                ;高低 4 位交换
RL    A                ;低 3 位变到高 3 位位置
ANL   A,#0E0H          ;保留高 3 位
ORL   30H,A            ;和(30H)的低 5 位合并
SJMP  $                ;动态停机
```

注意：SJMP 为转移指令，$ 为本条指令地址。SJMP　$ 即为无休止循环执行本指令，从而实现动态停机的效果。

【例 2-16】 把在 R4 和 R5 中的两字节数取补(高位在 R4 中)。

```
CLR  C                    ;CY 清零
MOV  A,R5
CPL  A
ADD  A,#01H               ;低位取反加 1
MOV  R5,A
MOV  A,R4
CPL  A                    ;高位取反
ADDC  A,#00H              ;加低位的进位
MOV  R4,A
SJMP  $
```

【例 2-17】 把 R7 中的无符号数扩大 10 倍。

```
MOV  A,R7
MOV  B,#0AH
MUL  AB
MOV  R7,A                 ;R7 存积的低位
MOV  R6,B                 ;R6 存积的高位
SJMP  $
```

【例 2-18】 把 R1R0 和 R3R2 中的两个 4 位 BCD 码数相加,结果送 R5R4 中,如有进位,则存于进位位 C 中。

```
CLR  C                    ;清进位
MOV  A,R0
ADD  A,R2                 ;低字节相加
DA  A                     ;十进制调整
MOV  R4,A
MOV  A,R1
ADDC  A,R3                ;高字节相加
DA  A                     ;十进制调整
MOV  R5,A
SJMP  $
```

2.4 控制转移指令

这一类指令的功能是改变指令的执行顺序,转到指令指示的新的 PC 地址执行。

51 单片机的控制转移指令有以下类型。

① 无条件转移:无须判断,执行该指令就转移到目的地址。

② 条件转移:需判断标志位是否满足条件,满足条件转移到目的地址执行,否则顺序执行。

③ 绝对转移:转移的目的地址用绝对地址指示,通常为无条件转移。

④ 相对转移:转移的目的地址用相对于当前 PC 的偏差(偏移量) 指示,通常为条件转移。

⑤ 长转移或长调用:目的地址距当前 PC 64KB 地址范围内。

⑥ 短转移或短调用:目的地址距当前 PC 2KB 地址范围内。

在实际编程中,无论哪种转移指令,转移的目的地址均是以符号地址表示的(如

SJMP　ABC，AJMP　LOOP…），转移的类型是由指令的操作符决定的，转移的目的地址的机器码是汇编器编译时自动填入的。

以上共 14 条指令，下面分别予以介绍。

2.4.1　调用程序和返回类指令

1. 长调用

LCALL　addr16　　；addr16→$PC_{0\sim15}$

说明：

① 该指令功能是：保护断点，即当前 PC（本指令的下一条指令的首地址）压入堆栈；子程序的入口地址 addr16 送 PC，转子程序执行。

② 本指令为 64KB 地址范围内的调子程序指令，子程序可在 64KB 地址空间的任意处。

③ 本指令的机器码为 3 字节 12 addr16。

2. 短调用

ACALL　addr11　　；addr11→$PC_{0\sim10}$

说明：

① 该指令的功能是：保护断点，即当前 PC 压入堆栈；addr11→$PC_{0\sim10}$，而 $PC_{11\sim15}$保持原值不变。

② 本指令为 2KB 地址范围的调子程序指令，子程序入口距当前 PC 不得超过 2KB 地址范围。

③ 本指令的机器码为 2 字节，设 addr11 的各位是 $a_{10}a_9a_8\cdots a_2a_1a_0$，则 ACALL 指令机器码为 $a_{10}a_9a_8$**10001**$a_7a_6a_5a_4a_3a_2a_1a_0$，其中 10001 是 ACALL 指令的操作码。

【例 2-19】　子程序调用指令 ACALL 在程序存储器中的首地址为 0100H，子程序入口地址为 0250H。试确定能否使用 ACALL 指令实现调用？如果能使用，确定该指令的机器码。

解　因为 ACALL 指令首地址在 0100H，而 ACALL 是 2 字节指令，所以下一条指令的首地址在 0102H。0102H 和 0250H 在同一 2KB 地址范围内，故可用 ACALL 调用。调用入口地址为 0250H，ACALL 指令的机器码形式为：010 <u>10001</u>01010000B＝5150H。

3. 子程序返回指令

① RET　　；从调用子程序返回

功能：从栈顶弹出断点到 PC，从子程序返回主程序。

② RETI　　；从中断服务程序返回

功能：从栈顶弹出断点到 PC，并恢复中断优先级状态触发器，从中断服务返回主程序。

2.4.2　转移指令

1. 无条件转移指令

（1）短转移

AJMP　addr11　　；addr11→$PC_{0\sim10}$

说明：

① 转移范围：本指令为 2KB 地址范围内的转移指令。对转移目的地址的要求与 ACALL 指令对子程序入口地址的要求相同。

② 机器码形式：本指令为 2 字节指令。设 addr11 的各位是 $a_{10}a_9a_8\cdots a_2a_1a_0$，则指令的机器码为 $a_{10}a_9a_8$**00001**$a_7a_6a_5a_4a_3a_2a_1a_0$。

【例 2-20】　绝对转移指令 AJMP 在程序存储器中的首地址为 2500H，要求转移到 2250H 地址处执行程序，试确定能否使用 AJMP 指令实现转移？如能实现，其指令的机器码是什么？

解　因为 AJMP 指令的首址为 2500H，其下一条指令的首址为 2502H，2502H 与转移目的

地址 2250H 在同一 2KB 地址范围内，故 AJMP 指令可实现程序的转移。指令的机器码：

0100000101010000B＝4150H

（2）长转移

LJMP　addr16　　；addr16→$PC_{0\sim15}$

说明：

① 本指令为 64KB 程序存储空间的全范围转移指令。转移地址可为 16 位地址中的任意值。

② 本指令为 3 字节指令 02 addr16。

（3）间接转移

JMP　@A＋DPTR　　；A＋DPTR→PC

【例 2-21】 A＝02H，DPTR＝2000H，指令 JMP　@A＋DPTR 执行后，PC＝2002H。也就是说，程序转移到 2002H 地址单元去执行。

【例 2-22】 现有一段程序如下：

```
        MOV   DPTR,#TABLE
        JMP   @A+DPTR
TABLE:  AJMP  PROC0
        AJMP  PROC1
        AJMP  PROC2
        ⋮
        AJMP  PROCn
```

根据 JMP　@A＋DPTR 指令的操作可知，当 A＝00H 时，程序转入到地址 PROC0 处执行；当 A＝02H 时，转到 PROC1 处执行……可见，这是一段多路转移程序，进入的路数由 A 确定。因为 AJMP 指令是 2 字节指令，所以 A 必须为偶数。

以上均为绝对转移指令，下面介绍相对转移指令。

（4）无条件相对转移

SJMP　rel　　；PC＋rel→PC，机器码为 80 rel

说明：

该指令的功能是将 $PC_{当前}$ 修改为 $PC_{目的}$，即转到目的地址执行程序。转移的偏移量 rel＝$PC_{目的}-PC_{当前}$，转移可以向前转（$PC_{目的}<PC_{当前}$），也可以向后转（$PC_{目的}>PC_{当前}$），rel 可能正也可能负，是 1 字节有符号数（用补码表示），因此 1 字节补码表示转移范围为－128～＋127。

2. 条件转移指令

条件转移指令均为相对转移指令，rel 的计算方法同 SJMP　rel 指令。

（1）累加器为零（非零）转移

JZ　rel　　；A＝0 转移（PC＋rel→PC），A≠0 程序顺序执行，机器码为 60 rel

JNZ　rel　　；A≠0 则转移（PC＋rel→PC），A＝0 程序顺序执行，机器码为 70 rel

（2）减一非零转移

DJNZ　Rn，rel　　；Rn－1→Rn，Rn ≠0，则转移（PC＋rel→PC）

　　；Rn＝0，程序顺序执行

DJNZ　dir，rel　　；(dir)－1→(dir)，(dir)≠0，则转移（PC＋rel→PC）

　　；(dir)＝0，程序顺序执行

说明：

① 本指令有自动减 1 功能。

② DJNZ　Rn，rel 是 2 字节指令，而 DJNZ　dir，rel 是 3 字节指令，所以在满足转移的条件

后，前者是 PC＋rel→PC，而后者是 PC＋rel→PC。

【例 2-23】 试说明以下一段程序运行后 A 中的结果。

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

根据程序可知，A＝10＋9＋8＋7＋6＋5＋4＋3＋2＋1＝55＝37H。

（3）比较转移

```
CJNE   A,dir,rel          ;A≠dir,转移(PC+rel→PC)
                          ;A=dir,程序顺序执行
CJNE   A,#data,rel        ;A≠#data,转移(PC+rel→PC)
                          ;A=#data,程序顺序执行
CJNE   Rn,#data,rel       ;Rn≠#data,转移(PC+rel→PC)
                          ;Rn=#data,程序顺序执行
CJNE   @Ri,#data,rel      ;(Ri)≠#data,转移(PC+rel→PC)
                          ;(Ri)=#data,程序顺序执行
```

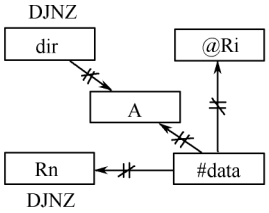

图 2-17　DJNZ、CJNE(≠)示意图

说明：

① CJNE 指令都是 3 字节指令，做两数相减操作，不回送结果，影响 CY 标志。

② 若第一操作数大于或等于第二操作数，则标志 CY＝0。若第一操作数小于第二操作数，则 CY＝1。这几条指令除实现两操作数相等与否的判断外，利用对 CY 的判断，还可完成两数大小的比较。

DJNZ、CJNE 指令的助记图如图 2-17 所示。

【例 2-24】 R7＝56H，指令 CJNE　R7，#34H，$＋08H 执行后，程序转移到放本条 CJNE 指令的首地址（$）加 08H 后的地址单元去执行。

【例 2-25】 编写程序，要求读 P_1 端口上的信息，若不为 55H，则程序等待，直到 P_1 端口为 55H 时，程序往下顺序执行。程序为：

```
MOV   A,#55H              ;A=55H
CJNE  A,P1,$              ;P1≠55H,则程序循环执行本指令
⋮
```

在实际编程中，转移的目的地址不管是 addr11、addr16，还是 rel，均是以符号地址表示的（如 SJMP　ABC，AJMP　LOOP…），转移的类型是通过指令的操作符来决定的。

【例 2-26】 51 单片机指令系统中，没有停机指令，通常用短转移指令 SJMP　$（$ 为本条指令的首地址）来实现动态停机的操作，试写出这条指令中机器码。

解　查附录 A，SJMP　rel 的指令码为 80rel。据题意，本条指令的首地址为 $，转移的目的地址也是 $，指令码字节数为 2，因此 $PC_{当前}$＝$＋2，则

$$rel = PC_{目的} - PC_{当前} = \$ - (\$ + 2) = -2 = FEH$$

所以 SJMP　$ 指令的机器码是 80FEH。

【例 2-27】 某程序中必须用比较转移指令 CJNE　A，#55H，LOOP，而目的地址 LOOP 和 PC 地址相隔甚远，超过－128～＋127 字节转移范围，应如何实现这类转移？

解　可以采用类似体育运动的 3 级跳远的方法，在距 PC 的－128～＋127 字节转移范围设

一条 LJMP　LOOP 指令，程序如下：

```
        CJNE A,#55H,LP
        ……
LP:     LJMP LOOP
        ……
LOOP:……
```

2.4.3　空操作指令

```
NOP    ;机器码   00
```

该指令经取指，译码后不进行任何操作(空操作)，而转到下一条指令，常用于生产一个机器周期的延时，或上机修改程序时作填充指令，以方便增减指令。

2.4.4　指令应用举例

【例 2-28】　将 A 累加器的低 4 位取反 4 次，高 4 位不变。每变换一次，从并行口 P_1 输出。

分析　因为异或运算的规则是一个数与“0”异或，该数不变；与“1”异或，该数变反。欲高 4 位不变，高 4 位与“0”异或；低 4 位取反，低 4 位与“1”异或，因此，A 和 0FH 异或可实现要求。4 次的计数可以采用加 1 计数，也可以采用减 1 计数，程序如下：

方法 1：加 1 计数。

```
        MOV   R0,#0          ;计数初值送 0
LL:     XRL   A,#0FH         ;高 4 位不变,低 4 位取反
        INC   R0             ;次数加 1
        MOV   P1,A           ;从 P1 输出
        CJNE  R0,#04,LL      ;不满 4 次循环
        RET
```

方法 2：减 1 计数。

```
        MOV   R0,#04H        ;计数初值送 4
LL:     XRL   A,#0FH
        MOV   P1,A
        DJNZ  R0,LL          ;次数减 1 不等于 0 循环
        RET
```

【例 2-29】　在内部 RAM 的 40H 地址单元中，有 1 字节符号数，编写求其绝对值后放回原单元的程序。程序如下：

```
        MOV   A,40H
        ANL   A,#80H
        JNZ   NEG            ;为负数转移
        SJMP  $              ;为正数,绝对值=原数,不改变原单元内容
NEG:    MOV   A,40H          ;为负数求补,得其绝对值
        CPL   A
        INC   A
        MOV   40H,A
        SJMP  $
```

有符号数在计算机中以补码形式存放，例如－5，存放在内部 RAM 中为 FBH，求补后得 5，即|－5|＝5。

2.5 位操作指令

51 单片机的特色之一就是具有丰富的位处理功能，在其硬件结构中有位处理机，包括有位累加器 C(即进位标志 CY)，有位存储器(即内部 RAM 和 SFR 的可寻址位)，还有对位操作的指令系统，使得开关量控制系统的设计变得十分方便。

在程序中，位地址的表达有多种方式：

- 用直接位地址表示，如 D4H；
- 用“·”操作符号表示，如 PSW.4，或 D0H.4；
- 用位名称表示，如 RS1；
- 用用户自定义名表示，如 ABC　BIT　D4H，其中，ABC 定义为 D4H 位的位名，BIT 为位定义伪指令。以上各例均表示 PSW.4 的 RS1 位。

位操作类指令的对象是 C 和直接位地址，由于 C 是位累加器，所以位逻辑运算指令的目的操作数只能是 C，这就是位操作指令的特点。下面将位操作的 17 条指令介绍如下。

1. 位清零、置 1、取反指令

CLR　C　　;0→CY

CLR　bit　　;0→bit

SETB　C　　;1→CY

SETB　bit　　;1→bit

CPL　C　　;$\overline{CY}$→CY

CPL　bit　　;$\overline{bit}$→ bit

2. 位与、位或指令

ANL　C,bit　　;CY∧(bit)→CY

ANL　C,/bit　　;CY∧($\overline{bit}$)→CY

ORL　C,bit　　;CY∨(bit)→CY

ORL　C,/bit　　;CY∨($\overline{bit}$)→CY

3. 位传送

MOV　C,bit　　;(bit)→CY

MOV　bit,C　　;CY→bit

4. 位转移

位转移根据位的值决定转移，均为相对转移指令，设 As 为下面各指令的首地址。

JC　rel　　;CY=1，则转移(PC+rel→PC)，否则程序顺序执行

JNC　rel　　;CY=0，则转移(PC+rel→PC)，否则程序顺序执行

JB　bit,rel　　;(bit)=1，则转移(PC+rel→PC)，否则程序顺序执行

JNB　bit,rel　　;(bit)=0，则转移(PC+rel→PC)，否则程序顺序执行

JBC　bit,rel　　;(bit)=1，则转移(PC+rel→PC)，且该位清零；否则程序顺序执行

【例 2-30】 用位操作指令实现 X=X0⊕X1，设 X0 为 $P_{1.0}$，X1 为 $P_{1.1}$，X 为 ACC.0。

解　方法 1：因位操作指令中无异或指令，依据 X=X0⊕X1=$\overline{X0}$X1+X0 $\overline{X1}$，用与、或指令完成，编程如下：

```
X   BIT   ACC.0
X0  BIT   P1.0
```

```
X1   BIT   P1.1          ;位定义
MOV   C,X0
ANL   C,/X1              ;C=X0∧X̄1
MOV   20H,C              ;暂存入 20H 单元
MOV   C,X1
ANL   C,/X0              ;C=X̄0∧X1
ORL   C,20H              ;C=X̄0X1+X0X̄1
MOV   X,C
SJMP   $
```

方法 2：根据异或规则，一数与“0”异或，其值不变；与“1”异或，其值变反，编程如下：

```
        MOV   C,X0
        JNB   X1,NCEX       ;X1=0,X=C=X0
        CPL   C
NCEX:   MOV   X,C           ;X1=1,X=C̄=X̄0
        SJMP   $
```

2.6 小　　结

① 51 单片机指令系统的特点是不同的存储空间寻址方式不同，适用的指令不同，必须进行区分。

② 指令是程序设计的基础，应重点掌握传送指令、算术运算指令、逻辑运算指令、控制转移指令和位操作指令，掌握指令的功能、操作的对象和结果，对标志位的影响，应要求熟记。

思考题与习题 2

2.1　51 单片机有哪几种寻址方式？适用于什么地址空间？用表格表示。

2.2　51 单片机的 PSW 程序状态字中无 ZERO(零)标志位，怎样判断某内部数据存储单元的内容是否为 0？

2.3　设 A=0，执行下列两条指令后，A 的内容是否相同，说明道理。

(1) MOVC　A,@A+DPTR

(2) MOVX　A,@DPTR

2.4　指出下列各指令中操作数的寻址方式：

指　　令	目的操作数寻址方式	源操作数寻址方式
ADD　A,40H		
PUSH　ACC		
MOV　B,　20H		
ANL　P1,#35H		
MOV　@R1,　PSW		
MOVC　A,@A+DPTR		
MOVX　@DPTR,A		

2.5　执行下列程序段

```
MOV　A,#56H
ADD　A,#74H
ADD　A,ACC
```

后,CY=_____,OV=_____,A=_____。

2.6　在错误的指令后面括号中打×。

MOV　@R1,#80H	(　)	MOV　R7,@R1	(　)
MOV　20H,@R0	(　)	MOV　R1,#0100H	(　)
CPL　R4	(　)	SETB　R7.0	(　)
MOV　20H,21H	(　)	ORL　A,R5	(　)
ANL　R1,#0FH	(　)	XRL　P1,#31H	(　)
MOVX　A,2000H	(　)	MOV　20H,@DPTR	(　)
MOV　A,DPTR	(　)	MOV　R1,R7	(　)
PUSH　DPTR	(　)	POP　30H	(　)
MOVC　A,@R1	(　)	MOVC　A,@DPTR	(　)
MOVX　@DPTR,#50H	(　)	RLC　B	(　)
ADDC　A,C	(　)	MOVC　@R1,A	(　)

2.7　设内部 RAM 中(59H)=50H,执行下列程序段:

```
MOV　A,59H
MOV　R0,A
MOV　A,#0
MOV　@R0,A
MOV　A,#25H
MOV　51H,A
MOV　52H,#70H
```

问 A=_____,　(50H)=_____,　(51H)=_____,　(52H)=_____。

2.8　设 SP=60H,内部 RAM 的(30H)=24H,(31H)=10H,在下列程序段注释的括号中填执行结果。

```
PUSH　30H　　;SP=(　),　(SP)=(　)
PUSH　31H　　;SP=(　),　(SP)=(　)
POP　DPL　　;SP=(　),　DPL=(　)
POP　DPH　　;SP=(　),　DPH=(　)
MOV　A,#00H
MOVX　@DPTR,A
```

最后执行结果是(　　)。

2.9　对下列程序中各条指令加注释,并分析程序运行的最后结果。

```
MOV　20H,#0A4H
MOV　A,#0D6H
MOV　R0,#20H
MOV　R2,#57H
ANL　A,R2
ORL　A,@R0
SWAP　A
CPL　A
ORL　20H,A
SJMP　$
```