第 3 章 指令系统与汇编语言程序设计

CHAPTER 3

指令系统是一套控制单片机执行操作的编码，它是单片机能直接识别的命令。指令系统在很大程度上决定了单片机的功能和使用是否方便灵活。指令系统对于用户来说也是十分重要的，只有详细了解了单片机的指令功能，才能编写出高效的软件程序。本章介绍 8051 单片机的指令系统。

3.1 指令助记符和字节数

指令本身是一组二进制数代码，记忆起来很不方便，为了便于记忆，将这些代码用具有一定含义的指令助记符来表示。助记符一般采用有关英文单词的缩写，这样就容易理解和记忆单片机的各种指令了。下面是两条分别用代码形式和助记符形式书写的指令：

```
十六进制代码     助记符              功能
   740A       MOV A,#0AH    ;将十六进制数 0AH 放入累加器 A 中
   2414       ADD A,#14H    ;累加器 A 中的内容与十六进制数 14H 相加,结果放在累加器 A 中
```

尽管采用助记符后，书写的字符增多了，但由于增强了可读性，使用时会觉得更方便。采用助记符和其他一些符号来编写的指令程序，称为汇编语言源程序，汇编语言源程序经过汇编之后即可得到可执行的机器代码目标程序。

一条指令通常由两部分组成：操作码和操作数。操作码用来规定这条指令完成什么操作，例如做加减运算，还是数据传送等。操作数则表示这条指令所完成的操作对象，即是对谁进行操作。操作数可以直接是一个数，或者是一个数所在的内存地址。

操作码和操作数都是二进制代码。在 8051 单片机中，8 位二进制数为 1 字节，指令是由指令字节组成的。对于不同的指令，指令的字节数不相同。8051 单片机有单字节指令、双字节指令或三字节指令。

单字节指令中既包含操作码的信息，也包含操作数的信息。这可能有两种情况，一种是指令的含义和对象都很明确，不必再用 1 字节来表示操作数。例如数据指针加一指令：INC DPTR，由于操作的内容和对象都很明确，故不必再加操作数字节，其指令码为：

10100011

另一种情况是用1字节中的几位来表示操作数或操作数所在的位置。例如从工作寄存器向累加器A传送数据的指令：MOV A,Rn,其中Rn可以是8个工作寄存器R0～R7中的一个,在指令码中分出3位来表示这8个工作寄存器,用其余各位表示操作码的作用,指令码为：

11101rrr

其中最低3位码用来表示从哪个寄存器取数,故1字节也就够了。8051单片机共有49条单字节指令。

双字节指令一般是用1字节表示操作码,再用1字节表示操作数或操作数的地址。这时操作数或其地址就是一个8位的二进制数,因此必须专门用1字节来表示。例如8位二进制数传送到累加器A的指令：MOV A,#data,其中#data表示8位二进制数,也叫立即数,这就是双节指令,其指令码为：

01110100	#data

双字节指令的第二字节,也可以是操作数所在的地址。8051单片机共有45条双字节指令。

三字节指令则是1字节的操作码,2字节的操作数。操作数可以是数据,也可以是地址,因此可能有如下4种情况：

操作码	立即数	立即数
操作码	地址	立即数
操作码	立即数	地址
操作码	地址	地址

8051单片机共有17条三字节指令,只占全部指令的15%。一般而言,指令的字节数越少,则其执行速度越快,从这个角度来说,8051单片机的指令系统是比较合理的。

3.2　寻址方式


视频讲解

所谓寻址,就是寻找操作数的地址。在用汇编语言编程时,数据的存放、传送、运算都要通过指令来完成,编程者必须自始至终十分清楚操作数的位置,以及如何将它们传送到适当的寄存器去运算。因此,如何从各个存放操作数的区域去寻找和提取操作数就变得十分重要。寻址方式就是通过确定操作数所在的地址把操作数提取出来的方法。

在8051单片机中,有7种寻址方式：寄存器寻址、直接寻址、立即寻址、寄存器间接寻址、变址寻址、相对寻址、位寻址。下面分别进行说明。

3.2.1　寄存器寻址

寄存器寻址就是以通用寄存器的内容作为操作数,在指令的助记符中直接以寄存器的

名字来表示操作数的位置。在8051单片机中,没有专门的通用硬件寄存器,而是把内部数据RAM区中00H～1FH地址单元作为工作寄存器使用,共有32个地址单元,分成4组,每组8个工作寄存器,命名为R0～R7,每次可以使用其中一组。当以R0～R7来表示操作数时,就属于寄存器寻址方式。例如:

```
MOV A,R0
ADD A,R0
```

前一条指令是将R0寄存器的内容传送到累加器A中,后一条指令则是对A和R0的内容作加法运算。

特殊功能寄存器B也可当作通用寄存器使用,但用B表示操作数地址的指令不属于寄存器寻址,而是属于下面所讲的直接寻址。

3.2.2 直接寻址

在指令中直接给出操作数地址,就属于直接寻址方式。在这种方式中,指令的操作数部分直接是操作数的地址。8051单片机中,用直接寻址方式可以访问片内数据RAM中DATA空间的00H～7FH共128字节单元以及所有的特殊功能寄存器。在指令助记符中,直接寻址的地址可用2位十六进制数表示。对于特殊功能寄存器,还可用它们各自的名称符号来表示,以增加程序的可读性。例如:

```
MOV A,3AH
```

就属于直接寻址,其中3AH所表示的就是内部RAM地址,这条指令的功能是将内部RAM中3AH字节单元的内容传送到累加器A,该指令的功能如图3.1所示。

图3.1 直接寻址操作

3.2.3 立即寻址

若指令的操作数是一个8位或16位二进制数,就称为立即寻址,指令中的操作数称为立即操作数。由于8位立即数和直接地址都是8位二进制数(或2位十六进制数),为区分起见,在立即数前面冠以“#”号,如#3AH表示立即数3AH,而直接写3AH则表示RAM区中地址为3AH的字节单元。例如:

```
MOV A,#3AH
MOV A,3AH
```

前一条指令为立即寻址,执行后累加器A中的内容变为3AH;后一条指令为直接寻址,执行后累加器A中的内容变为RAM区中地址为3AH字节单元的内容。在8051单片机中,只有一条16位立即数指令:

```
MOV DPTR,#data16
```

其功能是将16位立即数送往数据指针寄存器。由于是16位立即数,需要2字节表示,因此这是一条三字节的指令,即1字节指令码,2字节立即数,指令格式如下:

10010000	立即数高8位	立即数低8位

3.2.4　寄存器间接寻址

若以寄存器的名称间接给出操作数的地址，则称为寄存器间接寻址。在这种寻址方式下，指令中工作寄存器的内容不是操作数，而是操作数的地址。指令执行时，先通过工作寄存器的内容取得操作数地址，再到此地址所规定的存储单元取得操作数。

8051 单片机可采用寄存器间接寻址方式访问全部 256 字节的片内 RAM 地址单元 00H～FFH(即 IDATA 空间)，也可访问 64KB 的外部 RAM(即 XDATA 空间)，但是这种寻址方式不能访问特殊功能寄存器。只能采用工作寄存器 R0、R1 或数据指针寄存器 DPTR 来进行间接寻址，在寄存器 R0、R1 或 DPTR 名称前面加一个符号@来表示寄存器间接寻址。例如：

```
MOV A,@R0
```

该指令的功能如图 3.2 所示。指令执行之前 R0 寄存器的内容 3AH 是操作数的地址，内部 RAM 中地址为 3AH 单元的内容 65H 才是操作数，执行后，累加器 A 中的内容变为 65H。若采用寄存器寻址指令：

```
MOV A,R0
```

则执行后累加器 A 中的内容变为 3AH。对这两类指令的差别和用法，一定要区分清楚，正确使用。

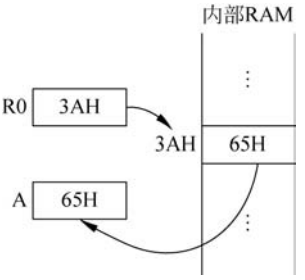


图 3.2　寄存器间接寻址操作

3.2.5　变址寻址

变址寻址是以某个寄存器的内容为基本地址，然后在这个基址上加上一定偏移量，才是真正的操作数地址。8051 单片机没有专门的变址寄存器，而是采用数据指针 DPTR 或程序计数器指针 PC 的内容为基本地址，地址偏移量则是累加器 A 中的内容，将基址与偏移量相加，即以 DPTR 或者 PC 的内容与 A 的内容之和作为实际的操作数地址。8051 单片机采用变址寻址方式可以访问 64KB 的 ROM 程序存储器(即 CODE 空间)。例如：

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

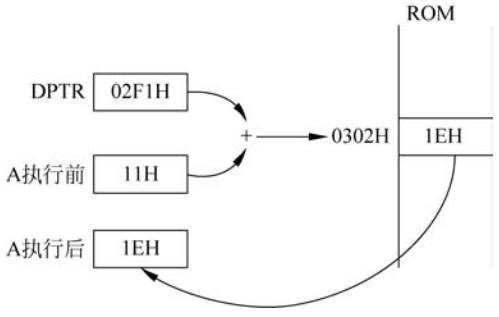


图 3.3　变址寻址操作

该指令的功能如图 3.3 所示。指令执行前(A)＝11H，(DPTR)＝02F1H，故实际操作数的地址应为 02F1H＋11H＝0302H。指令执行后将程序存储器 ROM 中 0302H 单元的内容 1EH 传送到累加器 A。需要注意的是，虽然在变址寻址时采用数据指针 DPTR 作为基址寄存器，但变址寻址的区域都是程序存储器 ROM 而不是数据存储器 RAM，另外尽管变址寻址方式的指令助记符和指令操作都较为复杂，但却是 1 字节指令。

3.2.6 相对寻址

8051单片机设有直接转移指令和相对转移指令。相对转移指令需要采用相对寻址方式,此时指令的操作数部分给出的是地址的相对偏移量。在指令中以rel表示相对偏移量,rel为一个带符号的常数,可正也可以负,若rel值为负数,则应用补码表示。一般将相对转移指令本身所在的地址称为源地址,转移后的地址称为目的地址,它们的关系为:

目的地址=源地址+指令字节数+rel

例如:

```
SJMP rel
```

该指令的功能如图3.4所示。这条指令的机器码为80,rel,共2字节。设该指令所在的源地址为2000H,rel的值为54H,则转移后的目的地址为:2000H+02+54H=2056H。

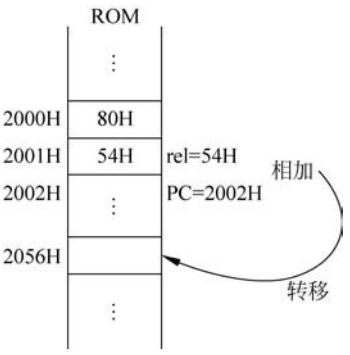


图3.4 相对寻址操作

3.2.7 位寻址

采用位寻址方式的指令,其操作数是8位二进制数中的某一位,在指令中要给出位地址,位地址可以是片内RAM中可位寻址区20H~2FH(即BDATA区)某字节单元的某一位,或者是可位寻址特殊功能寄存器的某一位。表3.1给出了可位寻址特殊功能寄存器及其位地址。

表3.1 可以位寻址的特殊功能寄存器

特殊功能寄存器	单元地址	表示符号	位地址
P0	80H	P0.0~P0.7	80H~87H
TCON	88H	TCON.0~TCON.7	88H~8FH
P1	90H	P1.0~P1.7	90H~97H
SCON	98H	SCON.0~SCON.7	98H~9FH
P2	A0H	P2.0~P2.7	A0H~A7H
IE	A8H	IE.0~IE.7	A8H~AFH
P3	B0H	P3.0~P3.7	B0H~B7H
IP	B8H	IP.0~IP.7	B8H~BFH
PSW	D0H	PSW.0~PSW.7	D0H~D7H
ACC	E0H	ACC.0~ACC.7	E0H~E7H
B	F0	B.0~B.7	F0H~F7H

位地址可采用以下几种方式表示。

(1) 直接用位地址00H~FFH来表示,如20H字节单元的0~7位可表示为20H~27H。

(2) 采用第n字节单元第n位来表示,如25H.5,表示25H字节单元的第5位。

(3) 对于特殊功能寄存器可以用寄存器名加位数的表示方法,如PSW.7,也可以直接

用其特殊功能位名称，如 CY。

(4) 用汇编语言中的伪指令定义。例如：

```
SETB PSW.7
SETB CY
```

这 2 条指令具有相同的功能，都是将程序状态字 PSW 的最高位置“1”。

3.3　指令分类详解


视频讲解

8051 单片机共有 111 条指令，按指令功能可分为算术运算指令、逻辑运算指令、数据传送指令、控制转移指令及位操作指令 5 大类。

3.3.1　算术运算指令

算术运算指令包括加、减、乘、除法指令，加法指令又分为普通加法指令、带进位加法指令和加 1 指令。

1. 普通加法指令

```
ADD  A,Rn              ;Rn(n=0～7)为工作寄存器
ADD  A,direct          ;direct 为直接地址单元
ADD  A,@Ri             ;Ri(i=0～1)为工作寄存器
ADD  A,#data           ;#data 为立即数
```

这组指令的功能是将累加器 A 的内容与第二操作数的内容相加，结果送回到累加器 A 中。在执行加法的过程中，如果位 7 有进位，则置“1”进位标志 CY，否则 CY 清“0”。如果位 3 有进位，则置“1”辅助进位标志 AC，否则 AC 清“0”。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，则置“1”溢出标志 OV，否则 OV 清“0”。

2. 带进位加法指令

```
ADDC  A,Rn             ;Rn(n=0～7)为工作寄存器
ADDC  A,direct         ;direct 为直接地址单元
ADDC  A,@Ri            ;Ri(i=0～1)为工作寄存器
ADDC  A,#data          ;#data 为立即数
```

这组指令的功能与普通加法指令类似，唯一的不同之处是在执行加法时，还要将上一次进位标志 CY 的内容也一起加进去。对于标志位的影响与普通加法指令相同。

3. 加 1 指令

```
INC  A
INC  Rn                ;Rn(n=0～7)为工作寄存器
INC  direct            ;direct 为直接地址单元
INC  @Ri               ;Ri(i=0～1)为工作寄存器
INC  DPTR              ;DPTR 为 16 位数据指针寄存器
```

这组指令的功能是将所指出操作数的内容加 1，如果原来的内容为 0FFH，则加 1 后将产生上溢出，使操作数的内容变成 00H，但不影响任何标志。指令 INC　DPTR 是对 16 位的数据指针寄存器 DPTR 执行加 1 操作，指令执行时，先对数据指针的低 8 位 DPL 的内容加 1，当产生上溢出时就对数据指针的高 8 位 DPH 加 1，但不影响任何标志。

4. 十进制调整

```
DA  A
```

这条指令的功能是对累加器 A 中内容进行 BCD 码调整,通常用于 BCD 码运算程序中,使 A 中的运算结果为两位 BCD 码数。该指令的执行过程如图 3.5 所示。

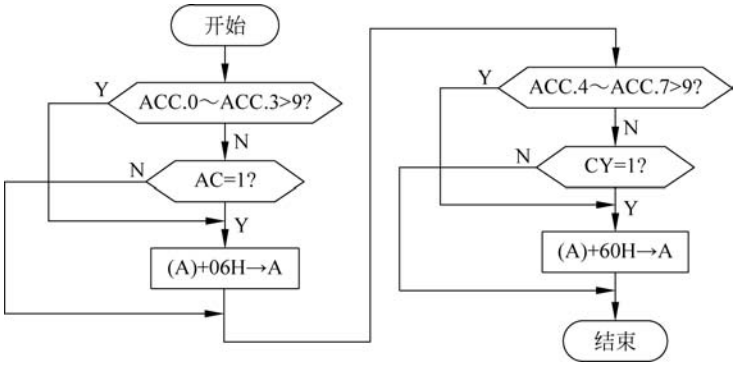


图 3.5 "DA A"指令的执行过程

5. 带进位减法指令

```
SUBB  A,Rn              ;Rn(n = 0～7)为工作寄存器
SUBB  A,direct          ;direct 为直接地址单元
SUBB  A,@Ri             ;Ri(i = 0～1)为工作寄存器
SUBB  A,#data           ;#data 为立即数
```

这组指令的功能是将累加器 A 的内容与第二操作数的内容相减,同时还要减去上一次进位标志 CY 的内容,结果送回到累加器 A 中。在执行减法的过程中,如果位 7 有借位,则当前进位标志 CY 置"1",否则 CY 清"0"。如果位 3 有借位,则置"1"辅助进位标志 AC,否则 AC 清"0"。如果位 6 有借位而位 7 没有借位,或者位 7 有借位而位 6 没有借位,则溢出标志 OV 置"1",否则 OV 清"0"。

6. 减 1 指令

```
DEC  A
DEC  Rn                 ;Rn(n = 0～7)为工作寄存器
DEC  direct             ;direct 为直接地址单元
DEC  @Ri                ;Ri(i = 0～1)为工作寄存器
```

这组指令的功能是将所指出操作数的内容减 1,如果原来的内容为 00H,则减 1 后将产生下溢出,使操作数的内容变成 0FFH,但不影响任何标志。

7. 单字节乘法指令

```
MUL  AB
```

这条指令的功能是将累加器 A 中的 8 位无符号整数与寄存器 B 中的 8 位无符号整数相乘,乘积为 16 位整数。乘积的低 8 位存放在累加器 A 中,高 8 位存放在寄存器 B 中。如果乘积大于 255(0FFH),则溢出标志 OV 置"1",否则 OV 清"0"。进位标志总是被清"0"。

8. 单字节除法指令

```
DIV  AB
```

这条指令的功能是将累加器 A 中的 8 位无符号整数除以寄存器 B 中的 8 位无符号整数，所得商的整数部分存放在累加器 A 中，余数部分存放在寄存器 B 中，清“0”进位标志 CY 和溢出标志 OV。如果原来 B 中的内容为 0(被 0 除)，则执行除法后 A 和 B 中的内容不定，并置“1”溢出标志 OV，在任何情况下，进位标志总是被清“0”。

3.3.2　逻辑运算指令

逻辑运算指令分为简单逻辑指令、逻辑与指令、逻辑或指令以及逻辑异或指令。

1. 简单逻辑指令

```
CLR  A                ;对累加器 A 清"0"
CPL  A                ;对累加器 A 的内容求反
RL   A                ;累加器 A 的内容向左环移一位
RLC  A                ;累加器 A 的内容带进位位 CY 向左环移一位
RR   A                ;累加器 A 的内容向右环移一位
RRC  A                ;累加器 A 的内容带进位位 CY 向右环移一位
SWAP A                ;将累加器 A 的高半字节(A.7～A.4)与低半字节(A.3～A.0)交换
```

这组指令的功能是直接对累加器 A 的内容进行简单逻辑操作，结果仍在累加器 A 中。

2. 逻辑与指令

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

这组指令的功能是将两个操作数的内容按位进行逻辑与运算，结果送入累加器 A 或由 direct 所指出的内部 RAM 单元。

3. 逻辑或指令

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

这组指令的功能是将两个操作数的内容按位进行逻辑或运算，结果送入累加器 A 或由 direct 所指出的内部 RAM 单元。

4. 逻辑异或指令

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

这组指令的功能是将两个操作数的内容按位进行逻辑异或运算，结果送入累加器 A 或由 direct 所指出的内部 RAM 单元。

3.3.3 数据传送指令

8051单片机的存储器空间可分为如下3部分,即

```
ROM存储器(CODE空间):                    0000H～FFFFH
片内RAM存储器(IDATA空间):                00H～FFH
片外RAM存储器/扩展I/O端口(XDATA空间):    0000H～FFFFH
```

指令对哪一个存储器空间进行操作是由指令的操作码和寻址方式确定的。对于程序存储器ROM只能通过变址寻址方式采用MOVC指令访问,对于特殊功能寄存器只能采用直接寻址和位寻址方式,不能采用间接寻址方式;对于8051单片机片内RAM的高128字节则只能采用寄存器间接寻址方式,而片内RAM的低128字节则既能间接寻址,也能直接寻址;片外RAM存储器/扩展I/O端口只能通过间接寻址方式用MOVX指令访问。

1. 数据传送到累加器A的指令

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

这组指令的功能是把源操作数的内容送入累加器A。

2. 数据传送到工作寄存器Rn的指令

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

这组指令的功能是把源操作数的内容送入当前工作寄存器区中的某一个寄存器R0～R7。

3. 数据传送到内部RAM单元或特殊功能寄存器SFR的指令

```
MOV  direct,A
MOV  direct,Rn               ;n=0～7
MOV  direct,direct
MOV  direct,@Ri              ;i=0或1
MOV  direct,#data
MOV  @Ri,A                   ;i=0或1
MOV  @Ri,direct              ;i=0或1
MOV  @Ri,#data               ;i=0或1
MOV  DPTR,#data16
```

这组指令的功能是把源操作数的内容送入指定的片内RAM单元或特殊功能寄存器。最后一条指令的功能是将16位数据送入数据指针寄存器DPTR。

4. 堆栈操作指令

```
PUSH direct                  ;进栈
POP  direct                  ;出栈
```

在8051单片机的特殊功能寄存器中有一个堆栈指针寄存器SP,进栈指令的功能是首先将堆栈指针SP的内容加1,然后将直接地址所指出的内容送入SP指出的内部RAM单元。出栈指令的功能是将SP所指出的内部RAM单元的内容送入由直接地址所指出的字节单元,同时将栈指针SP的内容减1。

5. 累加器 A 与外部数据存储器之间的数据传送指令

```
MOVX  A,@DPTR              ;((DPTR))→A
MOVX  A,@Ri                ;((P2Ri))→A,i = 0 或 1
MOVX  @DPTR,A              ;(A)→(DPTR)
MOVX  @Ri,A                ;(A)→(P2Ri)
```

这组指令的功能是在累加器 A 与片外数据存储器 RAM 或扩展 I/O 端口之间进行数据传送。

6. 查表指令

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

这是两条很有用的查表指令，它们可用来查找存放在程序存储器中的常数表格。其中第一条指令是以程序计数器 PC 作为基址寄存器，累加器 A 的内容作为无符号数偏移量与 PC 的内容(下一条指令的起始地址)相加，得到一个 16 位的地址，并将该地址指出的程序存储器单元的内容送入累加器 A。这条指令的优点是不改变特殊功能寄存器和 PC 的状态，只要根据 A 中的内容就可以取出表格中的常数；缺点是表格只能放在该条查表指令后面的 256 个单元之中，表格大小受到限制，而且表格只能被一段程序所利用。

第二条指令是以数据指针寄存器 DPTR 作为基址寄存器，累加器 A 的内容作为无符号数偏移量与 DPTR 的内容相加，得到一个 16 位的地址，并将该地址指出的程序存储器单元的内容送入累加器 A。这条查表指令的执行结果只与 DPTR 和累加器 A 的内容有关，而与该条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 64KB 的程序存储器中任意安排，并且一个表格可以为各个程序块所公用。

7. 字节交换指令

```
XCH   A,Rn                 ;n = 0～7
XCH   A,direct
XCH   A,@Ri                ;i = 0 或 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。

8. 半字节交换指令

```
XCHD A,@Ri                 ;i = 0 或 1
```

这条指令的功能是将累加器 A 的低 4 位内容和 R(i)所指出的内部 RAM 单元的低 4 位内容相互交换。

3.3.4　控制转移指令

1. 无条件短跳转指令

```
AJMP  addr11
```

这是 2KB 范围内的无条件跳转指令，它把程序存储器划分为 32 个区，每个区为 2KB，转移的目标地址必须与 AJMP 后面一条指令的第一字节在同一个 2KB 的范围之内(即转移目标地址必须与 AJMP 下一条指令的地址 A15～A11 相同)，否则将引起混乱。该指令执行时先将 PC 的内容加 2，然后将 11 位地址送入 PC.10～PC.0，而 PC.15～PC.11 保持不变。

2. 相对转移指令

```
SJMP  rel
```

这是一条无条件跳转指令,执行时在(PC)+2 后,把指令中有符号偏移量 rel 加到 PC 上,计算出偏移地址。因此,转移的目标地址可以在这条指令前 128 字节到后 127 字节之间。

3. 长跳转指令

```
LJMP  addr16
```

这条指令执行时把指令的第二字节和第三字节分别装入 PC 的高 8 位和低 8 位字节中,无条件地转向指定的地址。转移的目标地址可以在 64KB 程序存储器地址空间的任何地方。

4. 散转指令

```
JMP  @A+DPTR
```

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 中的 16 位数相加,结果作为下一条指令的地址送入 PC,不改变累加器 A 和数据指针 DPTR 的内容,也不影响标志。

5. 条件转移指令

```
JZ   rel                ;(A)=0 时转移
JNZ  rel                ;(A)≠0 时转移
JC   rel                ;CY=1 时转移
JNC  rel                ;CY=0 时转移
JB   bit,rel            ;(bit)=1 时转移
JNB  bit,rel            ;(bit)=0 时转移
JBC  bit,rel            ;(bit)=1 时转移,并清"0"bit 位
```

条件转移指令是当满足某一特定条件时执行转移操作的指令。条件满足时转移(相当于一条相对转移指令),条件不满足时则顺序执行下面一条指令。转移的目的地址在以下一条指令的起始地址为中心的 256 字节范围之内(-128～+127)。当条件满足时,把 PC 的值加到下一条指令的第一字节地址,再把有符号的相对偏移量 rel 加到 PC 上,计算出转移地址。

6. 比较不相等转移指令

8051 单片机没有专门的比较指令,但是提供了如下 4 条比较不相等转移指令:

```
CJNE  A,    direct,rel
CJNE  A,    #data,  rel
CJNE  Rn,   #data,  rel      ;n=0～7
CJNE  @Ri,  #data,  rel      ;i=0 或 1
```

这组指令的功能是比较前面两个操作数的大小,如果它们的值不相等则转移。把 PC 的值加到下一条指令的起始地址后,再把指令最后一字节的有符号相对偏移量加到 PC 上,计算出转移地址。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 CY 置"1",否则 CY 清"0"。不影响任何一个操作数的内容。

7. 减 1 不为 0 转移指令

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

```
DJNZ  direct,rel
```

这组指令把源操作数(Rn、direct)的内容减 1,并将结果回送到源操作数中去。如果相减的结果不为 0 则转移到由相对偏移量 rel 计算得到的目的地址。

8051 单片机提供了两条子程序调用指令,即短调用和长调用指令。

8. 短调用指令

```
ACALL  addr11
```

这是一条 2KB 范围内的子程序调用指令。执行时先把 PC 的值加 2 获得下一条指令的地址,然后把获得的 16 位地址压进堆栈(PCL 先进栈 PCH 后进栈),并将堆栈指针 SP 的值加 2,最后把 PC 值的高 5 位与指令提供的 11 位地址 addr11 相连接(PC15～PC11,a10～a0),形成子程序的入口地址并送入 PC,使程序转向执行子程序。所调用的子程序的起始地址必须在与 ACALL 指令后面一条指令的第一字节在同一个 2KB 区域的程序存储器中。

9. 长调用指令

```
LCALL  addr16
```

这条指令无条件地调用位于 16 位地址 addr16 处的子程序。它把 PC 的值加 3 以获得下一条指令的地址并将其压入堆栈(先低位字节后高位字节),同时把 SP 的值加 2,接着把指令的第二字节和第三字节(A15～A8,A7～A0)分别装入 PC 的高 8 位和低 8 位字节中,然后从 PC 所指出的地址开始执行程序。LCALL 指令可以调用 64KB 范围内程序存储器中的任何一个子程序,不影响任何标志。

10. 子程序返回指令

```
RET
```

这条指令的功能是从堆栈中弹出 PC 的高 8 位和低 8 位字节,同时把 SP 的值减 2,并从 PC 指向的地址开始继续执行程序,不影响任何标志。

11. 中断返回指令

```
RETI
```

这条指令的功能与 RET 指令相似,不同的是它还清“0”单片机的内部中断状态标志。

12. 空操作指令

```
NOP
```

这条指令只完成(PC)+1 操作,而不执行任何其他操作。

3.3.5　位操作指令

8051 单片机内部 RAM 中有一个位寻址区,还有一些特殊功能寄存器也可以位寻址,为此提供了丰富的位操作指令。

1. 位数据传送指令

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

这组指令的功能是把由源操作数指出的位变量送到目的操作数指定的位单元去。其中,一个操作数必须为进位标志,另一个操作数可以是任何可寻址位。

2. 位变量修改指令

```
CLR  C                    ;0→CY
CLR  bit                  ;0→bit
CPL  C                    ;对 CY 的内容取反
CPL  bit                  ;对 bit 位取反
SETB C                    ;"1"→CY
SETB bit                  ;"1"→bit
```

这组指令对操作数所指出的位进行清“0”、取反、置“1”的操作,不影响其他标志。

3. 位变量逻辑与指令

```
ANL  C,bit
ANL  C,/bit
```

这组指令的功能是将进位标志与指定的位变量(或位变量的取反值)相“与”,结果送到进位标志,不影响别的标志。

4. 位变量逻辑或指令

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

这组指令的功能是将进位标志与指定的位变量(或位变量的取反值)相“或”,结果送到进位标志,不影响别的标志。

附录 A 按指令功能列出了 8051 的全部指令。

3.4 汇编语言程序设计

前面介绍了 8051 单片机的指令系统,实际应用中将这些指令按需要有序地排列成一段完整的程序,就可以完成某一特定的任务。通常把这种程序称为汇编语言源程序,它主要由指令助记符和一些汇编伪指令组成,而把可以直接在计算机上运行的机器语言程序称为目标代码,由汇编语言源程序转换为目标代码的过程称为“汇编”,可以通过查附录 A 的指令表将汇编语言源程序中的指令逐条翻译为机器代码,实际上现在已经有许多在个人计算机上运行的专门“汇编程序”(如 ASM51 等),可以很方便地将汇编语言源程序转换成目标代码。


视频讲解

8051 单片机汇编语言程序由若干指令行组成,一般格式:

```
[标号:]操作码,[操作数][;注释]
```

其中,各段意义说明如下:

① “标号”是可选项,它可用来表示程序的地址。

② “操作码”是 8051 单片机的指令助记符。

③ “操作数”是可选项,它依赖于不同的 8051 指令,有些指令不需要操作数,有些指令则需要 1～3 个操作数,操作数可以是数字、符号或地址。十进制数以字符“D”为后缀,十六进制数以字符“H”为后缀,八进制数以字符“O”为后缀,二进制数以字符“B”为后缀,省略后缀时则默认为十进制数。立即数的前面须冠以符号“#”。

④ “注释”也是可选项,它是为理解程序含义而加上的文字解释,注释文字前面必须有一个分号。

在对汇编语言源程序进行汇编时，8051 指令行将被转换为一一对应的目标代码，它们可以被单片机 CPU 执行。另外，汇编语言源程序中还包含一些不能被单片机 CPU 执行的指令，称为汇编伪指令，它们仅提供汇编控制信息，用于在汇编过程中执行一些特殊操作，而不会被转换为目标代码。下面介绍一些常用的汇编伪指令。

1. 设置起始地址 ORG

一般格式：

```
ORG nnnn
```

其中，nnnn 为 4 位十六进制数，表示程序的起始地址。ORG 伪指令总是出现在每段程序的开始处，用于对该段程序在程序存储器中进行定位。需要注意的是，由 ORG 设置的程序空间地址应从小到大，并且不能重复。例如：

```
ORG 1000H
MAIN: MOV A,20H
```

表示该段程序在程序存储器中的起始地址为 1000H，换句话说，这里标号“MAIN”所代表的就是地址值 1000H。

2. 定义字节 DB

一般格式：

```
[标号：] DB 项或项表
```

其中，“项或项表”是单字节数据，或多个由逗号隔开的单字节数据，它们可以是数值，也可以是用引号括起来的 ASCII 字符串。DB 伪指令的功能是将项或项表的数据存入由标号（地址）开始的连续存储器单元之中。例如：

```
ORG  1000H
SEG1: DB 53H,78H
SEG2: DB'THIS IS A TEST'
```

注意，项或项表若为数值，其范围为 00H～FFH，若为字符串，其长度不能超过 80 个字符。

3. 定义字 DW

一般格式：

```
[标号：] DW 项或项表
```

DW 的基本含义与 DB 相似，不同之处在于 DW 用于定义 16 位数据。例如：

```
ORG  1000H
TABLE: DW 1234H,78H
```

4. 保留存储器空间 DS

一般格式：

```
[标号：] DS 表达式
```

DS 伪指令的功能是从标号指定的存储器地址开始，保留由表达式的值规定的存储器空间单元。例如：

```
ORG  1000H
TEMP: DS 10
```

本例表示从 TEMP 地址（1000H）开始保留 10 个连续的存储器单元。

5. 为标号赋值 EQU

一般格式：

```
字符名 EQU 表达式
```

EQU 伪指令的功能是将表达式的值赋给“字符名”。“字符名”一旦赋值之后，它的值在整个程序中就不能再改变。注意，这里“字符名”与标号不同，它后面没有冒号。例如：

```
PPAGE EQU 9000H
EN    EQU 1
```

6. 源程序结束 END

一般格式：

```
END
```

END 是一个程序结束标志，通常放在汇编语言源程序的结尾。

汇编语言程序设计，一般有主程序和子程序之分。主程序又称为前台程序，它通常是一个无穷循环；子程序又称为后台程序，它可以是各种功能子程序，也可以是中断服务子程序。在主程序中完成单片机系统的初始化，如内存单元清“0”、开放中断等。子程序一般完成某个具体任务，如数据采集、存储和运算等。一般在前台主程序的循环体中根据需要不断调用各种后台功能子程序，从而完成单片机应用系统规定的任务。

下面给出几个常用汇编语言设计的例子。

【例 3-1】 用程序实现 $c=a^2+b^2$，假设 a、b、c 分别存放于单片机片内 RAM 的 30H、31H、32H 三个单元。主程序通过调用子程序“SQR”用查表方式分别求得 a^2 和 b^2 的值，然后进行相加得到最后的 c 值。

主程序如下：

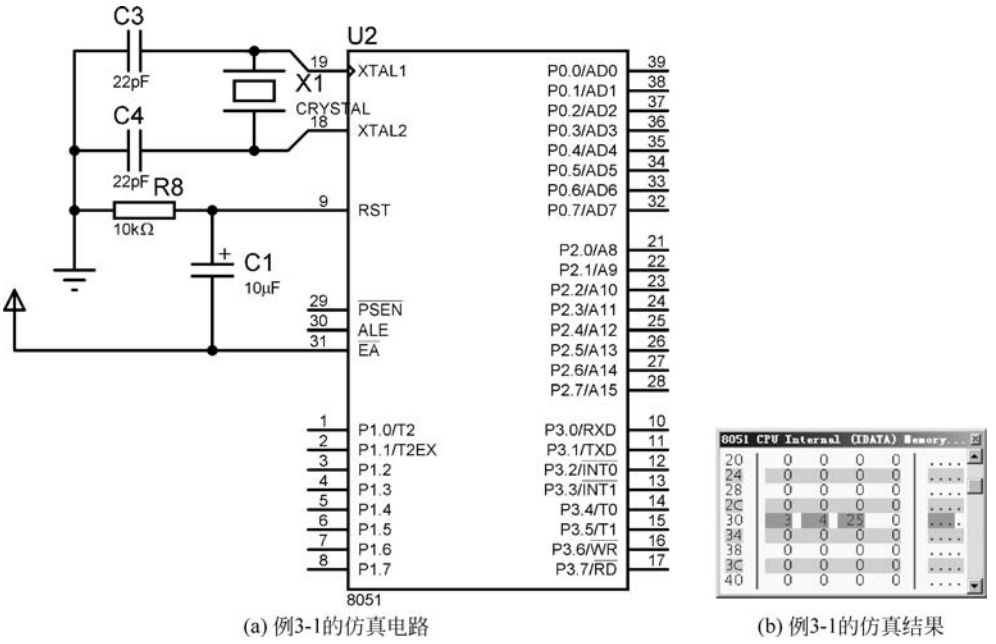
```
    ORG 0000H                  ; 程序的复位入口
START: LJMPMAIN
    ORG 0030H                  ; 主程序入口
MAIN: MOV 30H,#03              ; a=3
      MOV 31H,#04              ; b=4
      MOV A,30H                ; 取得 a 值
      LCALL SQR                ; 调查表子程序
      MOV R1,A                 ; a² 暂存于 R1 中
      MOV A,31H                ; 取得 b 值
      LCALL SQR                ; 调查表子程序
      ADD A,R1                 ; 计算 a² + b²
      MOV 32H,A                ; 存结果
WAIT: SJMP $                   ; 循环,等待
```

查表子程序如下：

```
    ORG0F00H
SQR: MOV DPTR,#TAB
     MOVC A,@A+DPTR            ; 查表求得平方值
     RET                       ; 子程序返回
TAB: DB 0,1,4,9,16             ; 平方表
     DB 25,36,49,64,81
     END                       ; 程序结束
```

这是一个包含了主程序、子程序以及汇编伪指令的完整汇编语言程序例子，一般应用

程序都可以参照这个方式编写，先编写一个主程序框架，再编写各个功能子程序。为了调用方便，应对子程序的入口和出口条件做尽可能详细的说明，并根据需要对主程序和子程序分别定位到适当的存储器地址，最后在主程序中通过调用子程序来完成所要求的任务。采用 Proteus 仿真本程序十分容易，仿真电路如图 3.6(a)所示，先将例 3-1 添加为仿真源程序并编译链接，然后将生成的 HEX 文件装入 8051 单片机，运行到标号“WAIT”处暂停，此时再打开 8051 单片机内部存储器窗口，观察到片内 30H、31H、32H 单元的内容如图 3.6(b)所示。



(a) 例3-1的仿真电路　　(b) 例3-1的仿真结果

图 3.6　例 3-1 仿真

【例 3-2】 利用循环实现软件延时 10ms 子程序。若单片机的晶振为 6MHz，则一个机器周期为 2μs，子程序的入口条件为：(R0)＝延时毫秒数，(R1)＝1ms 预定值。出口条件为：定时时间到，返回。

```
        ORG 1000H                                    机器周期数
DELAY: MOV R0,#10       ; 延时 10ms 值→R0              1
DL2:   MOV R1,#MT       ; 1ms 预定值→R1                1
DL1:   NOP              ; 延时 1 个机器周期            1
       NOP              ; 延时 1 个机器周期            1
       DJNZ R1,DL1      ; 1ms 延时循环                 2
       DJNZ R0,DL2      ; 10ms 延时循环                2
       RET              ; 延时结束,返回                2
```

这是一个双重循环程序，内循环的预定值 MT 尚需计算。因为各条指令执行时所需要的机器周期数是确定的，预定延时时间也已经给定 1ms，故 MT 的值可以这样确定：

$$(1+1+2)\times 2\times \mathrm{MT}=1000\mu s$$

$$\mathrm{MT}=125=7\mathrm{DH}$$

将 7DH 代替程序中的 MT，即可以实现 10ms 的延时。上面计算中仅考虑了内循环的执行

时间,若考虑其他指令的影响,该子程序的精确延时时间应为:

$$(1+2)\times 2+((1+2)\times 2+(1+1+2)\times 2\times 125)\times 10=10\,066\mu s$$

【例 3-3】 双字节数取补子程序。将(R4R5)中的双字节数取补,结果送 R4R5。

```
CMPT: MOV A,R5
      CPL A
      ADD A,#1
      MOV R5,A
      MOV A,R4
      CPL A
      ADDC A,#0
      MOV R4,A
      RET
```

【例 3-4】 双字节原码数左移一位子程序。将(R2R3)左移一位,结果送 R2R3,不改变符号位,不考虑溢出。

```
DRL1: MOV A,R3
      CLR C
      RLC A
      MOV R3,A
      MOV A,R2
      RLC A
      MOV ACC.7,C          ; 恢复符号位
      MOV R2,A
      RET
```

【例 3-5】 双字节原码右移一位子程序。将(R2R3)右移一位,结果送 R2R3,不改变符号位。

```
DRR1: MOV A,R2
      MOV C,ACC.7          ; 保护符号位
      CLR ACC.7            ; 移入 0
      RRC A
      MOV R2,A
      MOV A,R3
      RRC A
      MOV R3,A
      RET
```

【例 3-6】 双字节补码右移一位子程序。将(R2R3)右移一位,结果送 R2R3,不改变符号位。

```
CRR1: MOV A,R2
      MOV C,ACC.7     ; 保护符号位
      RRC A           ; 移入符号位
      MOV R2,A
      MOV A,R3
      RRC A
      MOV R3,A
      RET
```

补码表示的数可以直接相加,所以双字节无符号数加减程序也适用于补码的加减法。

【例 3-7】 双字节无符号数加法子程序。将(R2R3)和(R6R7)两个无符号数相加,结果送 R4R5。

```
NADD: MOV A,R3
      ADD A,R7
      MOV R5,A
      MOV A,R2
      ADDC A,R6
      MOV R4,A
      RET
```

【例 3-8】 双字节无符号数减法子程序。将(R2R3)和(R6R7)两个双字节数相减,结果送 R4R5。

```
NSUB1: MOV A,R3
       CLR C
       SUBB A,R7
       MOV R5,A
       MOV A,R2
       SUBB A,R6
       MOV R4,A
       RET
```

二进制数的乘法运算可以仿照十进制数。下面介绍一种利用单字节乘法指令来实现的多字节乘法。

因为(R2R3)×(R6R7)=((R2)×(R6))×2^{16}+((R2)×(R7)+(R3)×(R6))×2^{8}+(R3)×(R7),从而可以得到图 3.7 所示的算法。

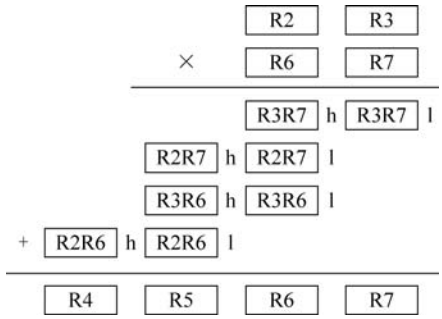


图 3.7　双字节二进制数快速乘法

【例 3-9】 无符号双字节快速乘法。将(R2R3)和(R6R7)两个双字节无符号数相乘,结果送 R4R5R6R7。

```
QMUL: MOV A,R3
      MOV B,R7
      MUL AB              ; R3 × R7
      XCH A,R7            ; R7 = (R3 × R7)L
      MOV R5,B            ; R5 = (R3 × R7)H
      MOV B,R2
      MUL AB              ; R2 × R7
      ADD A,R5
      MOV R4,A
      CLR A
      ADDC A,B
      MOV R5,A            ; R5 = (R2 × R7)H
      MOV A,R6
      MOV B,R3
```

```
        MUL AB              ; R3×R6
        ADD A,R4
        XCH A,R6
        XCH A,B
        ADDC A,R5
        MOV R5,A
        MOV F0,C            ; 暂存 CY
        MOV A,R2            ; R2×R6
        MUL AB
        ADD A,R5
        MOV R5,A
        CLR A
        MOV ACC.0,C
        MOV C,F0            ; 加以前加法的进位
        ADDC A,B
        MOV R4,A
        RET
```

二进制数除法也可以采用类似于人工手算除法的方法来实现。首先对被除数高位和除数进行比较,如果被除数高位大于除数,则商位为1,并从被除数减去除数,形成一个部分余数;如果被除数高位小于除数,商位为0,且不执行减法。接着把部分余数左移一位,并与除数再次进行比较。如此循环直至被除数的所有位都处理完为止。一般商如果为 n 位,则需循环 n 次。这种除法先比较被除数和除数的大小,根据比较结果确定商为1或0,并且当商为1时才执行减法,故称为比较法。一般情况下,如果除数和商均为双字节,则被除数为4字节,如果被除数的高两字节大于或等于除数,则商不能用双字节表示,此时为溢出,所以在除法之前先检验是否会发生溢出,如果溢出则置溢出标志不执行除法。

【例 3-10】 将(R2R3R4R5)和(R6R7)中两个无符号数相除,结果商送R4R5,余数送R2R3。

```
NDIV1: MOV A,R3             ; 先比较是否发生溢出
       CLR C
       SUBB A,R7
       MOV A,R2
       SUBB A,R6
       JNC NDVE1
       MOV B,#16            ; 无溢出,执行除法
       NDVL1: CLR C         ; 执行左移1位,移入为0
       MOV A,R5
       RLC A
       MOV R5,A
       MOV A,R4
       RLC A
       MOV R4,A
       MOV A,R3
       RLC A
       MOV R3,A
       XCH A,R2
       RLC A
       XCH A,R2
       MOV F0,C             ; 保存移出的最高位
       CLR C
       SUBB A,R7            ; 比较部分余数与除数
       MOV R1,A
```

```
        MOV A,R2
        SUBB A,R6
        JB F0,NDVM1
        JC NDVD1
NDVM1: MOV R2,A          ; 执行减法(回送减法结果)
        MOV A,R1
        MOV R3,A
        INC R5           ; 商为 1
NDVD1: DJNZ B,NDVL1      ; 循环 16 次
        CLR F0           ; 正常出口
        RET
NDVE1: SETB F0           ; 溢出
        RET
```

复习思考题

1. 8051 单片机指令系统有哪几种寻址方式?

2. 写出下列指令的寻址方式。

(1) JZ　20H

(2) MOV A,R2

(3) MOV　DPTR,＃4012H

(4) MOV　A,@R0

(5) MOVC　A,@A＋PC

(6) MOV　C,20H

(7) MOV A,20H

3. 已知 A＝7AH R0＝30H,(30H)＝A6H,PSW＝81H,写出以下各条指令执行之后的结果。

(1) XCH A,R0

(2) XCH A,30H

(3) XCH A,@R0

(4) XCHD A,@R0

(5) SWAP A

(6) ADD A,R0

(7) ADD A,30H

(8) ADD A,＃30H

(9) ADDC A,30H

(10) SUBB A,30H

(11) SUBB A,＃30H

(12) DA A

(13) RL A

(14) RLC A

(15) CJNE A,＃30H,00H

(16) CJNE A,30H,00H

4. 指出以下哪些指令是不存在的,并改用其他指令(或 n 条指令)来实现预期的指令。

(1) MOV 20H,30H

(2) MOV R1,R2

(3) MOV @R3,20H

(4) MOV DPH,30H

(5) MOV C,PSW.1

(6) MOVX R2 @DPTR

(7) XCH R1,R2

5. 设 A＝83H,R0＝17H,(17H)＝34H,问执行以下指令后,A 等于多少?

```
ANL A,＃17H
ORL 17H,A
XRL A,@R0
CPL A
```

6. 若 SP＝26H,PC＝2346H,标号 LABEL 所在的地址为 3466H,问执行长调用指令

LCALL LABEL 后,堆栈指针和堆栈的内容发生什么变化? PC 值等于多少?

7. 若已知 A=76H,PSW=81H,转移指令所在地址为 2080H,当执行以下指令后,程序是否发生转移? PC 值等于多少?

(1) JNZ 12H
(2) JNC 34H
(3) JB P,66H
(4) JBC AC,78H
(5) CJNE A,#50H,9AH
(6) DJNZ PSW,0BCH

8. 若已知 40H 单元的内容为 08H,下列程序执行之后 40H 单元的内容变为多少?

```
MOV  R1,#40H
MOV  A,@R1
RL   A
MOV  R0,A
RL   A
RL   A
ADD  A,R0
MOV  @R1,A
```

9. 试编写程序,将片外 8000H 开始的 16 个连续单元清“0”。

10. 按下面要求编程。

$$(51H)=\begin{cases}-1, & \text{若}(50H)\leqslant 20H\\ 0, & \text{若 } 20H<(50H)<40H\\ -1, & \text{若}(50H)\geqslant 40H\end{cases}$$

11. 试编写查表程序求 0～8 内整数的平方。

12. 有一个 16 位无符号二进制原码数存放于 50H、51H 单元,编程实现全部二进制数左移一位。

13. 两个 16 位有符号二进制原码数分别存放于 30H、31H 单元和 40H、41H 单元,编写用子程序调用方式实现这两个有符号二进制原码数相乘的程序。