第 3 章　MCS-51 单片机指令系统

在前两章的学习中，我们已经对单片微型计算机的内部结构和工作原理有了一个基本的了解。在此基础上，本章将进一步介绍指令的格式、分类和寻址方式，并以大量实例阐述 MCS-51 指令系统中每条指令的含义和特点，以便为汇编语言程序设计打下基础。

3.1　概　　述

本节主要论述指令格式、指令的三种表示形式、指令的字节数、指令的分类和指令系统综述 5 个问题，用以为本章后面各节的介绍准备条件。

3.1.1　指令格式

指令格式是指指令码的结构形式。通常，指令可以分为操作码和操作数两部分。其中，操作码部分比较简单，操作数部分则比较复杂，常常随计算机类型的不同而有较大差别。

在最原始的计算机中，操作数部分可以包括 4 部分地址，故称为 4 地址计算机。这种计算机的指令格式为：

操作码	第一操作数地址	第二操作数地址	结果操作数地址	下一条指令地址

其中，操作码字段用于指示机器执行何种操作，是加法操作还是减法操作，是数据传送还是数据移位操作，等等；“第一操作数地址”用于指示两个操作数中的第一操作数在内存中的地址；“第二操作数地址”可以使机器在内存中找到参加运算的第二个操作数；“结果操作数地址”用于存放操作结果；“下一条指令地址”指示机器按此地址取出下一条要执行指令的指令码。这种指令格式的缺点是指令码太长，严重影响了指令执行的速度。

MCS-51 单片机指令格式采用了地址压缩技术，它把操作数字段的 4 个地址压缩到一个，故称为单地址指令格式。指令的具体格式为：

操 作 码	操作数或操作数地址

其中，“操作数或操作数地址”字段相当于四地址机中的“第一操作数地址”字段；“第二操作数地址”和“结果操作数地址”合二为一，由累加器 A 充任，物理地址为 E0H，在操作码中隐含；“下一条指令地址”由程序计数器 PC 充当，PC 自动加“1”就能使 MCS-51 连续按序执行程序。因此，在指令执行前，用户通常必须安排一条传送指令，预先把第二操作数传送到累加器 A。这样，累加器 A 在指令执行后就可自动获得结果操作数。

3.1.2 指令的三种表示形式

指令是计算机用于控制各功能部件完成某一指定动作的指示和命令。指令不同，各功能部件所完成的动作也不一样，指令的功能也不相同。因此，根据题目要求，选用不同功能指令的有序组合就构成了程序。计算机执行不同的程序就可完成不同的运算任务。

指令的表示形式是识别指令的标志，也是人们用来编写和阅读程序的基础。通常，指令有二进制、十六进制和助记符三种表示形式，指令的这三种表示形式各有各的用处，是人们学习、掌握和使用好计算机的重要手段。

指令的二进制形式是一种可以直接为计算机识别和执行的形式，故又称为指令的机器码或汇编语言源程序的目标代码。指令的二进制形式具有难读、难写、难记忆和难修改等缺点，因此人们通常不用它来编写程序。指令的十六进制形式虽然读写方便，但仍不易为人们识别和修改，通常也不被用来编写程序，只是在某些场合（如实验室）才被用来作为输入程序的一种辅助手段。指令以这种十六进制代码输入机器以后，需要由常驻于机器内部的监控程序把它们翻译成二进制形式存入内存储器，而后才能为机器所识别和执行。指令的助记符形式又称为指令的汇编符形式或汇编语句形式，是一种由英文单词或缩写字母形象表征指令功能的形式。这种形式不仅易为人们识别和读写，而且记忆和交流极为方便，常常被人们用来进行程序设计，但编好和修改好的程序必须通过人工或机器把它们翻译成机器码形式才能被计算机执行。例如，如果累加器 A 中已有一个加数 10，那么，能够完成 10＋8 并把结果送入累加器 A 的加法指令的二进制形式为 00100100000001000B；指令的十六进制形式为 2408H；指令的助记符形式为：

```
ADD   A,   #08H   ;A←A+08H
```

其中，ADD 为操作码，指示进行加法操作；逗号右侧为源操作数或第一操作数；逗号左侧的累加器 A 在指令执行前为第二操作数寄存器，在指令执行后为结果操作数寄存器；分号的后面部分为注释，它并非指令的组成部分，只是用来标明相应指令的功能。

3.1.3 指令的字节数

在指令的二进制形式中，指令不同，指令的操作码和操作数也不相同。有些指令的操作码和操作数加起来只有 1 个字节，这种指令称为单字节指令；有些指令是双字节指令，操作码和操作数各占 1 个字节。同样道理，可以有 3 字节指令，4 字节指令，等等。

按照指令码的字节来分，MCS-51 单片机指令通常可以分为单字节、双字节和 3 字节指令三种。

1. 单字节指令（49 条）

单字节指令码只有一个字节，由 8 位二进制数组成。这类指令共有 49 条，占总指令数的 44%。通常，单字节指令又可分为两类：一类是无操作数的单字节指令；另一类是含有操作数寄存器编号的单字节指令。

（1）无操作数单字节指令　这类指令的指令码只有操作码字段，没有专门指示操作数的字段，操作数是隐含在操作码中的。例如，INC　DPTR 指令的二进制形式为

1	0	1	0	0	0	1	1

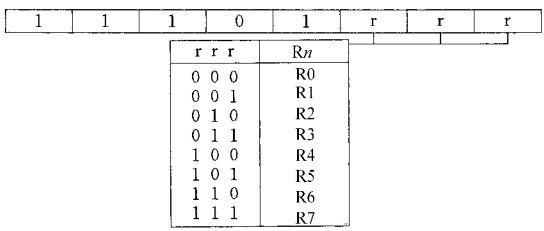
其中：8 位二进制数码均为操作码，DPTR 数据指针由操作码隐含。

(2) 含有操作数寄存器号的单字节指令　这类指令的指令码由操作码字段和专门用来指示操作数所在寄存器号的字段组成。

例如，8 位数传送指令

MOV　A，　R*n*　；A←R*n*

其中：*n* 的取值范围为 0～7。相应指令码格式如图 3-1 所示。



r r r	R*n*
0 0 0	R0
0 0 1	R1
0 1 0	R2
0 1 1	R3
1 0 0	R4
1 0 1	R5
1 1 0	R6
1 1 1	R7

图 3-1　MOV　A，　R*n* 指令的格式

图 3-1 中，r r r 3 位为源操作数所在的寄存器号，取值范围为 000B～111B；其余 5 位为操作码，目的操作数寄存器是累加器 A，由操作码字段隐含。

2. 双字节指令(46 条)

双字节指令含有两个字节，可以分别存放在两个存储单元中，操作码字节在前，操作数字节在后。操作数字节可以是立即数(即指令码中的数)，也可以是操作数所在的片内 RAM 地址。

例如，8 位数传送指令

MOV　A，　#data　；A←data

这条指令的含义是把指令码中第 2 字节 data 取出来存放到累加器 A 中，该指令的指令码为：

0	1	1	1	0	1	0	0
data							

其中：74H 为操作码，占 1 个字节；data 为源操作数，也占 1 个字节；累加器 A 是目的操作数寄存器，由操作码隐含。

3. 三字节指令(16 条)

这类指令的指令码的第 1 字节为操作码，第 2 和第 3 字节为操作数或操作数地址。由于有两个字节的操作数或操作数地址，故三字节指令共有如下 4 类：

操作码
data15～data8
data7～data0

例如，指令 MOV　DPTR，#data16

操作码
direct
data

例如，指令 MOV　direct，#data

操作码
data
direct(rel)

例如，指令 CJNZ A，#data，rel

操作码
addr15～addr8
addr7～addr0

例如，指令 LCALL　addr16

通常，指令字节数越少，指令执行速度越快，所占存储单元也就越少。因此，在程序设计中，应在可能的情况下注意选用指令字节数少的指令。

3.1.4　指令的分类

指令通常是按功能分类的，MCS-51 单片机按功能可以分为 5 类：数据传送指令、算术运算指令、逻辑操作和环移指令、控制转移指令和位操作指令等。

1. 数据传送指令(28 条)

这类指令共有 28 条，主要用于在单片机片内 RAM 和特殊功能寄存器 SFR 之间传送数据，也可以用于在单片机片内和片外存储单元之间传送数据。数据传送指令是把源地址中的操作数传送到目的地址(或目的寄存器)的指令，在该指令执行后源地址中的操作数不被破坏。源操作数有 8 位和 16 位之分，前者称为 8 位数传送指令，后者称为 16 位数传送指令。

交换指令也属于数据传送指令，是把两个地址单元中的内容相互交换。因此，这类指令中的操作数或操作数地址是互为“源操作数/源操作数地址”和“目的操作数/目的操作数地址”的。

2. 算术运算指令(24 条)

算术运算指令共有 24 条，用于对两个操作数进行加、减、乘、除等算术运算。在两个操作数中，一个应放在累加器 A 中，另一个可以放在某个寄存器或片内 RAM 单元中，也可以放在指令码的第 2 和第 3 字节中。指令执行后，运算结果便可保留在累加器 A 中，运算中产生的进位标志、奇偶标志和溢出标志等均可保留在 PSW 中。参加运算的两数可以是 8 位的，也可以是 16 位的。

3. 逻辑操作和环移指令(25 条)

这类指令包括逻辑操作和环移两类指令。逻辑操作指令用于对两个操作数进行逻辑乘、逻辑加、逻辑取反和异或等操作。大多数指令在执行前也需要把两个操作数中的一个预先放入累加器 A，操作结果也在累加器 A 中。环移指令可以对累加器 A 中的数进行环移。环移指令有左环移和右环移之分，也有带进位位 Cy 和不带进位位 Cy 之分。

4. 控制转移指令(17 条)

控制转移指令分为条件转移、无条件转移、调用和返回等指令,共 17 条。这类指令的共同特点是可以改变程序执行的流向,或者是使 CPU 转移到另一处地址执行,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 中地址为目标。

5. 位操作指令(17 条)

位操作指令又称布尔变量操作指令,共分为位传送、位置位、位运算和位控制转移指令 4 类。其中,位传送、位置位和位运算指令的操作数不是以字节为单位进行操作,而是以字节里某位中的内容为单位进行的;位控制转移指令不是以检测某个字节为条件而转移,而是以检测字节中的某一位的状态来转移的。

3.1.5 指令系统综述

指令的集合或全体称为指令系统。指令系统是微型计算机核心部件 CPU 的重要性能指标,是进行 CPU 内部电路设计的基础,也是计算机应用工作者共同关心的问题。因此,计算机类型不同,指令系统中每条指令的格式和功能也不相同。例如,MCS-48 中的 8048 有 90 条指令,MCS-96 中的 8096 单片机有 100 条指令,它们之间的指令是不相同的。但同一系列不同型号的计算机,其指令系统常常有不少是同宗相近的。例如,8022 指令系统是 8048 的一个子集,Z80-CPU 有 158 条基本指令,其中 78 条和 Intel 8080 在机器码一级上兼容。

MCS-51 单片机指令系统共有 111 条指令,可以实现 51 种基本操作。这 111 条指令的分类方法颇多,除可以按照指令功能和字节数分类外,还可以按照指令的机器周期数来分类。如果按照指令的机器周期数来分,MCS-51 系列单片机常可以分为单机器周期指令 57 条,双机器周期指令 52 条和四机器周期指令 2 条等。

1. 指令系统中所用符号的说明

MCS-51 指令系统中的所有指令如附录 C 所列。附表 C.1 中,除操作码字段采用了 42 种操作码助记符外,还在源操作数和目的操作数字段中使用了一些符号。这些符号的含义归纳如下:

(1) R*n*:工作寄存器,可以是 R0～R7 中的一个。

(2) #data:8 位立即数,实际使用时 data 应是 00H～FFH 中的一个。

(3) direct:8 位直接地址,实际使用时 direct 应该是 00H～FFH 中的一个,也可以是特殊功能寄存器 SFR 中的一个。

(4) @Ri:表示寄存器间接寻址,Ri 只能是 R0 或 R1。

(5) #data16:16 位立即数。

(6) @DPTR:表示以 DPTR 为数据指针的间接寻址,用于对外部 64K RAM/ROM 寻址。

(7) bit:位地址。

(8) addr11:11 位目标地址。

(9) addr16:16 位目标地址。

(10) rel:8 位带符号地址偏移量。

(11) $：当前指令的地址。

2. 指令对标志位的影响

MCS-51 指令分两类：一类指令执行后要影响到 PSW 中某些标志位的状态，即不论指令执行前标志位状态如何，指令执行时总按标志位的定义形成新的标志状态；另一类指令执行后不会影响到标志位的状态，标志位原来是什么状态，指令执行后也是这个状态。

不同的指令对标志位影响是不相同的，每条指令对标志位的影响如附录 C 中表 C.1 所示。其中，√表示对相应标志位有影响，×表示对相应标志位无影响。

3.2 寻址方式

在计算机中，寻找操作数的方法定义为指令的寻址方式。在执行指令时，CPU 首先要根据地址寻找参加运算的操作数，然后才能对操作数进行操作，操作结果还要根据地址存入相应存储单元或寄存器中。因此，计算机执行程序实际上是不断寻找操作数并进行操作的过程。通常，指令的寻址方式可以有多种，寻址方式越多，指令功能就越强。

在 MCS-51 单片机中，操作数的存放范围是很大的，可以放在片外ROM/RAM中，也可以放在片内 ROM/RAM 以及特殊功能寄存器 SFR 中。为了适应这一操作数范围内的寻址，MCS-51 的指令系统共使用了 7 种寻址方式，它们是：寄存器寻址、直接寻址、立即寻址、寄存器间址、变址寻址、相对寻址和位寻址。

3.2.1 寄存器寻址

这类指令所需操作数在 MCS-51 内部累加器 A、通用寄存器 B 和某个工作寄存器 R0～R7 等中，指令码内含有该操作数的寄存器号。例如，加 1 指令 INC Rn，含义是把 Rn 工作寄存器中的内容加 1，其指令格式为：

0	0	0	0	1	r	r	r

其中：r r r 三位二进制代码可以代表工作寄存器 R0～R7 中的任何一个，其编号如图 3-1 所示。若 r r r＝000B，则上述指令变为 INC R0，指令码为 08H，CPU 执行后可使 R0 中的内容加 1。由表 2-2 可知，R0 的物理地址由 PSW 中 RS1 和 RS0 的状态决定。因此，若设 RS1 RS0＝01B，则 R0 的物理地址必为 08H，MCS-51 单片机执行 INC R0 指令后，08H 单元中的内容可由原来的 24H 变为 25H，如图 3-2 所示。

寄存器寻址方式的指令很多，在 3.1.3 节的单字节指令中所述单字节指令 MOV A、Rn 的源操作数也属于寄存器寻址方式。

3.2.2 直接寻址

直接寻址指令的指令码中含有操作数地址，该地址通常可以是 8 位二进制数，常处于指令码中的第二或第三字节，机器执行它们时便可根据直接地址找到所需要的操作数。

在 MCS-51 单片机中，可以用于直接寻址的存储空间主要有片内 RAM 的低 128 字

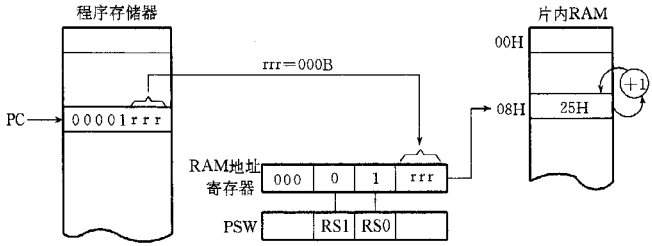


图 3-2　寄存器寻址示意图

节和特殊功能寄存器 SFR(8052 片内 RAM 的高 128 字节只能被间接寻址)。在这类寻址方式的指令中,直接地址通常采用 direct(或 addr11 或 addr16)表示。例如,8 位数传送指令 MOV A,direct 中的源操作数就是采用直接寻址的,其操作码为 E5 direct。但在这条指令的实际使用中或将它汇编成机器码时,direct 必须采用实际操作数的物理地址。例如,若用 3AH 代替上述指令中的 direct,则指令变为

```
MOV   A, 3AH   ;A←(3AH)
```

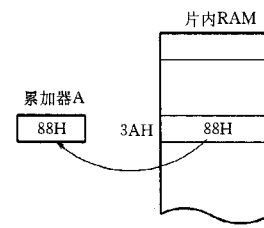


图 3-3　直接寻址示意图

该指令的操作码为 E5H,处在指令的第 1 字节,3AH 为直接寻址地址,处在指令的第 2 字节(见附录 C 中表 C.2 的指令表)。指令的含义是把 3AH 中的内容 88H 送入累加器 A 中,如图 3-3 所示。

直接寻址指令颇多,使用时特别容易混淆。为此,特提出以下三点注意事项:

(1) 指令助记符中的 direct 是操作数所在存储单元的物理地址,由两位十六进制数码表示。当直接寻址的地址为 SFR 中的某一个时,direct 既可以使用 SFR 的物理地址,也可以使用 SFR 的名称符号。但提倡使用后者,因为这可以增强所编程序的可读性。不过,在汇编时仍应将它翻译成物理地址才能为机器识别和执行。例如,如下两条指令:

```
MOV   A,   SP    ;A←SP
MOV   A,   81H   ;A←(81H)
```

它们的形式虽然不同,但汇编后的指令码是完全一样的,均为 E581H。在注释字段中,SP(Stack Pointer,堆栈指针)中的内容也可采用 81H 加圆括号后来表示。

(2) 在 MCS-51 指令系统中,累加器有 A、ACC 和 E0H 三种表示形式,分属于两种不同的寻址方法,但指令的执行效果是完全相同的。例如:

```
INC    A
INC    ACC
INC    0E0H
```

其中:第一条指令是寄存器寻址,指令码为 04H;第二条和第三条指令是直接寻址,指令码为 05E0H,第三条指令中的物理地址 E0H 前面要加 0(凡以字母 A～F 开头的十六进

制数均需加前导 0)。这三条指令的执行效果相同,都是使累加器 A 中的内容加 1。

(3) 在指令系统中,字节地址和位地址是有区别的。前者用 direct 表示,后者用 bit 表示。但在实际程序中,两者都要用十六进制数表示,因此使用中也容易混淆。例如:

```
MOV  A, 20H  ;A←(20H)
MOV  C, 20H  ;Cy←(20H)
```

在第一条指令中,由于目标寄存器是累加器 A,因此指令中的 20H 是字节地址 direct,汇编时应汇编成 E520H。第二条指令中由于目标寄存器是进位标志位 C(即 PSW.7),故它的 20H 属于位地址 bit,相应 20H 中的内容是指 24H 单元中的最低位 20H(见图 2-5)中的内容,汇编后的指令码为 A220H(见附录 C 中表 C.2)。显然,两条指令的含义和执行效果是完全不同的。

3.2.3 立即寻址

立即寻址指令的特点是指令码中直接含有所需寻找的操作数。该操作数称为立即数,可以是二进制 8 位,也可以是二进制 16 位,常处在指令码的第二和第三字节位置上。

在指令的汇编形式中,立即数通常使用 # data 或 # data16 表示,其中 # 是它区别于 direct(或 bit)的唯一标志,读者使用时务必不要疏忽。例如:

```
MOV  A, #3AH  ;A←3AH
MOV  A, 3AH   ;A←(3AH)
```

其中:第一条指令的源操作数是立即型寻址,3AH 作为一个 8 位二进制数传送到累加器 A 中,指令码为 743AH;第二条指令的源操作数是直接寻址,3AH 是作为地址看待的,指令的含义是把 3AH 中的内容送入累加器 A,指令码为 E53AH(见附录 C 中表 C.2)。

对于 16 位立即数指令,汇编时它的高 8 位应放在前面(即指令的第二字节位置),低 8 位放在后面(即指令的第三字节位置)。例如,如下指令的指令码为 901828H:

```
MOV  DPTR, #1828H  ;DPTR←1828H
```

3.2.4 寄存器间址

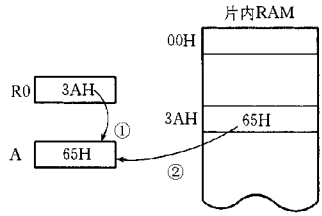
寄存器间址指令的特点是指令码中含有操作数地址的寄存器号。计算机执行这类指令时,它首先根据指令码中寄存器号找到所需要的操作数地址,再由操作数地址找到操作数,并完成相应操作。因此,寄存器间址实际上是一种二次寻找操作数地址的寻址方式。

在汇编形式的指令中,间址寄存器采用@Ri 或@DPTR 表示。其中,Ri 或者是 R0,或者是 R1(即:i 的取值为 0 或 1),@是它区别于寄存器寻址的标记。例如:

```
MOV  A, R0   ;A←R0
MOV  A, @R0  ;A←(R0)
```

其中:第一条指令是寄存器寻址,R0 中为操作数,指令码为 E8H;第二条指令是寄存器间

址，R0 中为操作数地址，不是操作数，指令码为 E6H。两条指令的含义是截然不同的：第一条指令执行后累加器 A 中为 3AH；第二条指令执行后累加器 A 中为操作数 65H，如图 3-4 所示。



注：①为第一条指令的执行效果
②为第二条指令的执行效果

图 3-4　寄存器间址寻址示意图

使用寄存器间址指令时，应注意如下两点：

(1) 寄存器间址指令可以拓宽单片机寻址范围。其中，@Ri 用于对片内 RAM 的寻址(8052 的地址范围为 00H～FFH，8031/8051 的地址范围为 00H～7FH)，也可以对片外 RAM 寻址(地址范围为 0000H～00FFH)；@DPTR 的寻址范围可以覆盖片外 ROM/RAM 的全部 64KB 区域。

(2) 寄存器间址指令不能用于特殊功能寄存器 SFR 的寻址。例如，如下程序是不能访问 SP 的：

```
MOV   R0, #81H
MOV   A, @R0
```

3.2.5　变址寻址

MCS-51 变址寻址指令具有以下 3 个特点：

(1) 指令操作码内隐含有作为基地址寄存器用的数据指针 DPTR 或程序计数器 PC，其中 DPTR 或 PC 中应预先存放有操作数的基地址。

(2) 指令操作码内也隐含有累加器 A，累加器 A 中应预先存放有被寻址操作数地址对基地址的偏移量，该地址偏移量应是一个 00H～FFH 范围内的无符号数。

(3) 在执行变址寻址指令时，单片机先把基地址(在 DPTR 或 PC 内)和地址偏移量(累加器 A 中)相加，以形成操作数的物理地址。

MCS-51 单片机有如下两条变址寻址指令：

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

第一条变址寻址指令是单字节指令，机器码为 83H。该指令执行时先使 PC 中当前值(机器码 83H 所在 ROM 单元地址)加 1，即取出指令码 83H，然后把这个加 1 后的 PC 中的地址与累加器 A 中的地址偏移量相加，从而取出该地址中操作数并传送到累加器 A 中。第二条指令执行过程和第一条指令类似，现举例加以说明。

[例 3.1]　已知片外 ROM 的 0302H 单元中有一常数 X，现欲把它取到累加器 A，请编写相应程序，并进行必要的分析。

解：根据变址寻址特点，基地址显然应取 0300H，地址偏移量为 02H。相应程序为：

```
MOV     DPTR, #0300H        ;DPTR←0300H
MOV     A, #02H             ;A←02H
MOVC    A, @A+DPTR          ;A←X
```

其中:第一条和第二条传送指令是为第三条变址寻址指令准备条件的。在第三条指令执行时,单片机先把 DPTR 中的 0300H 和累加器 A 中的 02H 相加后得到 0302H,然后到片外 ROM 中取出操作数 X 送到累加器 A。因此,累加器 A 具有双重作用,在指令执行前用来存放地址偏移量 02H,指令执行后的内容为目的操作数 X,指令执行过程如图 3-5 所示。

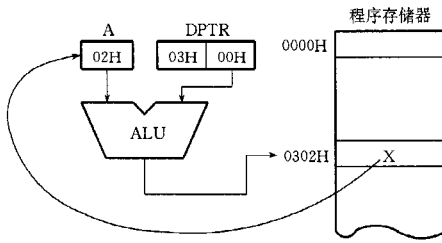


图 3-5 变址寻址示意图

应当指出两点:一是变址寻址指令的变址寻址区是程序存储器 ROM,而不是数据存储器 RAM;二是变址寻址是单字节双周期指令,CPU 执行这条指令前应预先在 DPTR 和累加器 A 中为该指令的执行准备条件。

3.2.6 相对寻址

相对寻址在相对转移指令中使用。相对转移指令的指令码中含有相对地址偏移量,相对地址偏移量是一个带符号的 8 位二进制补码,其取值范围为 $-128 \sim +127$(见表1-2)。

MCS-51 单片机有两类相对转移指令:一类是双字节相对转移指令,又称为短转移指令;另一类是三字节相对转移指令,又称长转移指令。不论是哪一类指令,单片机执行时总是把程序计数器 PC 中的**当前值**(即:从程序存储器中取出转移指令后的 PC 值)和指令码中的相对地址偏移量 rel 相加,以形成下一条要执行指令的地址。例如,现有如下双字节相对转移指令

```
2000H  8054H  SJMP  rel  ;PC←PC+2+rel
```

其中:80H 是该指令的操作码,放在 2000H 单元中,54H 是相对地址偏移量(相当于 rel),放在 2001H 单元。在执行这条指令时,单片机先从 2000H 和 2001H 单元取出指令码(程序计数器 PC 被加 1 两次从而变为 2002H),然后把程序计数器 PC 中的内容和 54H 相加,以形成目标地址 2056H,重新送回 PC。这样,当单片机再根据 PC 取指令执行时,程序就转到 2056H 处执行,指令的执行过程如图 3-6 所示。

相对转移指令特别有用,能生成浮动代码,深受用户青睐。

在使用中应注意以下两点:

(1) 在相对转移指令的执行中,当前 PC 值是指相对转移指令从程序存储器中取出来后的 PC 值,该值和地址偏移量相加便可形成目标转移地址,地址偏移量是一个带符号的