# 第3章 80C51的指令系统和程序设计

**学习目的**

(1) 了解机器语言、汇编语言和高级语言的特点,汇编语言程序的设计步骤。

(2) 理解 80C51 的寻址方式及相应的寻址空间。

(3) 熟练掌握 80C51 的 111 条指令的应用方法和功能。

(4) 掌握汇编语言指令的基本格式,熟悉机器语言指令的格式。

(5) 掌握汇编语言程序的设计思想和设计方法。

(6) 理解子程序的特点和设计中应注意的问题。

**学习重点和难点**

(1) 80C51 的寻址方式及相应的寻址空间。

(2) 80C51 的指令系统。

(3) 汇编语言程序的设计思想和设计方法。

(4) 子程序设计。

## 3.1 指 令 概 述

指令是 CPU 完成某种操作的命令,指令系统是指计算机能够完成各种功能的指令的集合。总体来说,计算机的指令越丰富、寻址方式越多,则其总体功能就越强。80C51 的指令系统共有 111 条。从指令的字节数来说,单字节指令 49 条,双字节指令 45 条,三字节指令 17 条;从指令的执行时间来说,单周期指令 64 条,双周期指令 45 条,4 周期指令 2 条;从指令中所含操作数的多少来说,无操作数指令 3 条,单操作数指令 35 条,双操作数指令 69 条,三操作数指令 4 条。

### 3.1.1 指令分类

80C51 的指令系统按照功能的不同可分为 5 大类。

(1) 数据传送和交换类指令(29 条):包括内部 8 位数据传送指令 15 条,内部 16 位数据传送指令 1 条,外部数据传送指令 4 条,交换和查表类指令 9 条。

(2) 算术运算类指令(24 条)：包括加法指令 8 条，减法指令 4 条，乘法指令 1 条，除法指令 1 条，加一减一指令 9 条，BCD 码调整指令 1 条。

(3) 逻辑运算类指令(24 条)：包括逻辑运算指令 20 条，循环移位类指令 4 条。

(4) 控制转移类指令(17 条)：包括无条件转移类指令 4 条，条件转移类指令 8 条，调用和返回类指令 5 条。

(5) 位操作类指令(17 条)：包括位传送指令 2 条，位置位和位清零指令 4 条，位运算指令 6 条，位转移指令 3 条，判 CY 标志指令 2 条。

### 3.1.2　指令的格式

对于 80C51 系统，汇编语言的指令格式为：

[标号：]　操作码　　操作数或操作数地址　　[；注释]

对指令格式中各项的说明如下所示。

#### 1. 标号

- 标号是程序员根据编程需要给指令设定的符号地址，可有可无。
- 标号由 1～8 个字符组成，第 1 个字符必须是英文字母，而不能是数字或其他符号。
- 标号后必须用冒号。
- 在程序中，不可以重复使用。

#### 2. 操作码

操作码表示指令的操作种类，规定了指令的具体操作。例如：ADD(加法操作)和 MOV(数据传送操作)。

#### 3. 操作数或操作数地址

操作数或操作数地址，表示参加运算的数据或数据的存放地址，如果操作数是以它的地址的形式给出，就要先找到这个地址才能找到需要的操作数。操作数和操作数之间必须用逗号分开。操作数一般有以下几种形式。

(1) 没有操作数项，操作数隐含在操作码中，如 RET 指令。

(2) 只有一个操作数，如 RL　A 指令。

(3) 有两个操作数，如 MOV　A，＃0FFH 指令，操作数之间以逗号相隔，其中：0FFH 项称为源操作数，累加器 A 项称为目的操作数。

(4) 有三个操作数，如 CJNE　A，＃00H，NEXT 指令，操作数之间也以逗号相隔。

#### 4. 注释

注释是对指令的解释说明，用以提高程序的可读性；注释前必须以“；”和指令分开，注释可有可无。

### 3.1.3 指令中的符号意义说明

在描述 80C51 指令系统的功能时，约定了一些用于说明操作数及操作数存放方式的符号，这些符号的意义如表 3-1 所示。

**表 3-1 80C51 指令中的符号意义说明**

| 符　号 | 意　义 |
|---|---|
| Rn | 表示当前选定寄存器组的工作寄存器 R0～R7 |
| Ri | 表示作为间接寻址的地址指针 R0～R1 |
| data | 表示 8 位立即数，即 00H～FFH |
| data16 | 表示 16 位立即数，即 0000H～FFFFH |
| addr16 | 表示 16 位地址，用于 64KB 范围内寻址 |
| addr11 | 表示 11 位地址，用于 2KB 范围内寻址 |
| direct | 8 位直接地址，可以是内部 RAM 区的某一单元或某一专用功能寄存器的地址 |
| rel | 带符号的 8 位偏移量(－128～＋127) |
| bit | 位寻址区的直接寻址位 |
| (x) | x 地址单元中的内容，或 x 作为间接寻址寄存器时所指单元的内容 |
| @ | 间接地址符号 |
| / | 在位操作指令中，表示取反运算 |
| ← | 将 ← 后面的内容传送到前面去 |

## 3.2 寻 址 方 式

从指令格式可以发现，一条指令最关键的地方就是操作数部分，特别是当操作数不是直接给出，而是间接给出时，CPU 就要先行寻找操作数。由于操作数可以用不同的方法给出来，所以寻找操作数地址的方式也就不同。寻找操作数地址的方式，称为寻址方式。在 80C51 单片机中有 7 种寻址方式。

(1) 立即数寻址：要在数据前加“＃”号，代表这个数是立即数。

(2) 直接寻址：在直接寻址中，操作数的地址是直接给出，可以是 20H、30H 等。

(3) 寄存器寻址：寄存器寻址的过程与直接寻址一模一样，唯一不同的就是，操作数这时不是放在某一个单元，而是放在某一个寄存器当中。

(4) 寄存器间接寻址：在寄存器间接寻址中，操作数的地址是通过某一个寄存器间接得到的。

(5) 变址寻址：在变址寻址中，操作数的地址是由两个寄存器中的内容相加得到的。

(6) 相对寻址：操作数是一个相对距离值。

(7) 位寻址：直接对单片机中的位进行操作。

### 3.2.1　立即寻址

“立即寻址”是指在指令中直接给出参与操作的数据(立即数)的寻址方式,立即数前加“#”。这种寻址方式主要用于对特殊功能寄存器和对指定存储单元赋予初始值。

例如：MOV A,#30H

其功能是将30H这个立即数传送给累加器ACC,在这里称30H为立即数。指令的机器代码为74H、30H,双字节指令。指令的执行过程如图3-1(a)所示。再例如：MOV R0,#12H,其作用是对R0赋予初始值为12H。

在80C51单片机中,当要对片外的RAM和I/O端口进行访问时,或进行查表操作时,通常要对DPTR赋值。

例如：MOV DPTR,#3456H

其功能是将3456H这个立即数传送给数据指针DPTR,其中高字节34H送DPH,低字节56H送DPL。指令的机器代码为90H、34H、56H,三字节指令。指令的执行过程如图3-1(b)所示。

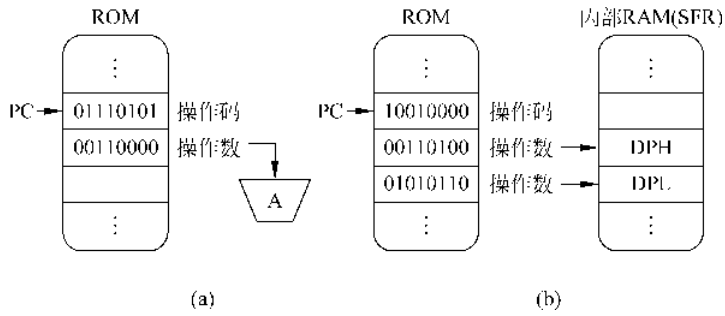


图3-1　立即寻址示意图

此外,在80C51单片机的立即寻址中允许使用符号地址。假设程序中有一表格：

```
TABLE: DB 00H,01H,02H,…,0FFH
```

则可以用MOV DPTR,#TABLE指令获得表格的首地址。

### 3.2.2　直接寻址

“直接寻址”是在指令中直接给出操作数所在存储单元的地址号,该地址指示了参与操作的数据所在的字节地址或位地址。这种寻址方式主要用于对特殊功能寄存器和内部RAM(低128B)的访问。

例如：MOV A,30H

其功能是将内部RAM 30H单元的内容送累加器A中,指令的机器代码为E5H、30H,双字节指令。指令的执行过程如图3-2所示。

在直接寻址方式中,可以直接用符号地址代替。

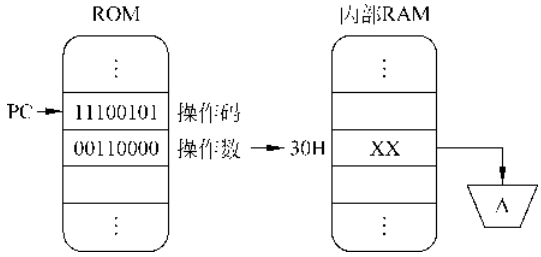


图 3-2　直接寻址示意图

例如：MOV A,P1

表示将 P1 口(地址为 90H)的数据送累加器 A。

### 3.2.3　寄存器寻址

“寄存器寻址”是在指令中给出存放操作数的寄存器名称(Rn、A、B、DPTR 等),被寻址寄存器中的内容就是操作数。由于这种寻址是在 CPU 内部的访问,所以运算速度最快。

例如：MOV A,R1

其功能是将 R1(默认为内部 RAM 01H 单元)的内容送累加器 A 中,指令的机器代码为 E9H,单字节指令。指令的执行过程如图 3-3 所示。

**说明**：对于指令 MOV A,Rn

指令的机器代码为 11101xxxB,其中 xxx 与 n 的取值(0～7)对应。

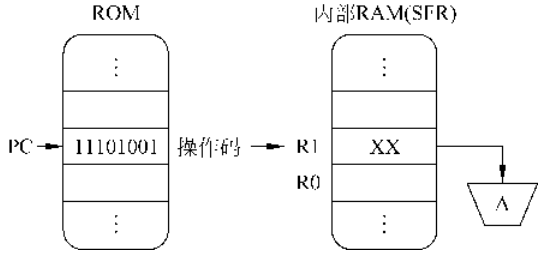


图 3-3　寄存器寻址示意图

### 3.2.4　寄存器间接寻址

“寄存器间接寻址”是以指令中指定寄存器(R0、R1、DPTR)的内容作为操作数的地址,再以该地址对应单元中的内容作为操作数。为了区别于寄存器寻址,因此在寄存器间接寻址中的寄存器名称前加地址符号“@”。在寄存器间接寻址中,当访问内部 RAM 低 128B 空间、或者访问外部 RAM 的 256B 空间时(地址范围为 0000H～00FFH),用当前组工作寄存器 R0 或 R1 作地址指针,而当访问外部 RAM 的整个 64KB 空间时,用 DPTR 作地址指针。

例如：MOV A,@R1

其功能是以当前组工作寄存器 R1 的内容(设：40H)作为操作数的地址，然后将内部 RAM 40H 单元的内容送累加器 A 中，指令的机器代码为 E7H，单字节指令。指令的执行过程如图 3-4 所示。

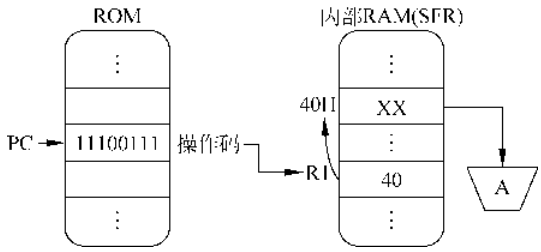


图 3-4　寄存器间接寻址示意图

## 3.2.5　变址寻址

“变址寻址”是将基址寄存器(DPTR 和 PC)与变址寄存器(A)的内容相加，形成一个 16 位结果，作为操作数的地址，实现对程序存储器的访问。

例如：　MOVC A,@A + DPTR

其功能是把 DPTR 中的内容作为基地址，把累加器 A 中的内容作为地址偏移量，两者相加后得到一个 16 位地址，然后把与该地址对应的 ROM 单元中的内容送累加器 A。指令的机器代码为 93H，单字节指令。指令的执行过程如图 3-5 所示。

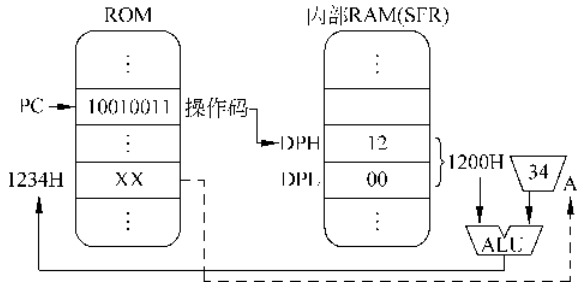


图 3-5　变址寻址示意图

同样寻址方式的指令还有两条：

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

该类指令常用于编写查表程序。

## 3.2.6　相对寻址

“相对寻址”是以程序计数器 PC 的当前值与指令中给定的 8 位偏移量相加，其结果作为跳转指令的目的地址。这种寻址方式常用于相对转移指令中。

指令：SJMP rel

含义为当程序执行到上述语句时，在当前语句位置的基础上向前或向后跳转 rel 中指定的位移量。

例如：SJMP 25H

其功能是当程序执行到该语句时，在当前语句位置的基础上向前跳转 25H 的位移量，继续执行程序。指令的机器代码为 80H 和 25H，双字节指令。指令的执行过程如图 3-6 所示。

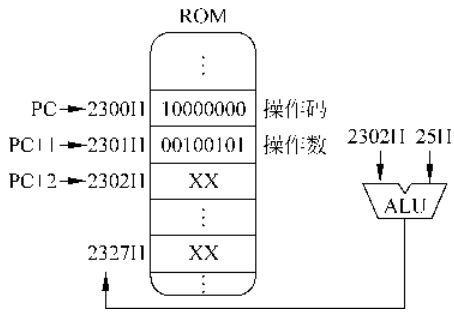


图 3-6 相对寻址示意图

### 3.2.7 位寻址

“位寻址”是指按二进制位(bit)进行的寻址，可寻址位包括片内 RAM 的 20H～2FH 共 16 字节区域的 128 位和部分特殊功能寄存器的相关位。位寻址是直接寻址方式的一种，其特点是对 8 位二进制数中的某一位的地址进行操作。

可位寻址的位地址的表示形式如下所示。

(1) 直接使用位地址形式。

例如：MOV 00H, C ；(00H)←(Cy)

其中 00H 是片内 RAM 中 20H 地址单元的第 0 位。

(2) 字节地址加位序号的形式。

例如：MOV 20H.0, C ；(20H.0)←(Cy)

其中 20H.0 是片内 RAM 中 20H 地址单元的第 0 位。

(3) 位的符号地址(位名称)的形式。对于部分特殊功能寄存器，其各位均有一个特定的名字，所以可以用它们的位名称来访问该位。

例如：ANL C, P ；(C)← (C)∧(P)

其中 P 是 PSW 的第 0 位，C 是 PSW 的第 7 位。

(4) 字节符号地址(字节名称)加位序号的形式。对于部分特殊功能寄存器(如状态标志寄存器 PSW)，还可以用其字节名称加位序号形式来访问某一位。

例如：CPL PSW.6 ；(AC)← ( / PSW.6 )

其中 PSW.6 表示该位是 PSW 的第 6 位。

例如：
```
SETB   C          ；将专用寄存器 PSW 中的 Cy 位置为 1
CLR    P1.0       ；将单片机的 P1.0 清“0”
SETB   3CH        ；将内部 RAM27H 的第 4 位置“1”
```

## 3.3 80C51 的指令系统

指令系统是计算机所固有的，是表明计算机性能特性的重要指标；同时它也是汇编语言程序设计的基础。80C51 的指令系统按照功能的不同可分为 5 大类：数据传送和交换类、算术运算类、逻辑运算类、控制转移类和位操作类。学习指令系统时，应注意以下几个

方面。

- 指令的格式和功能。
- 操作码的含义，操作数的表示方法。
- 寻址方式，源操作数和目的操作数的范围。
- 对标志位的影响。
- 指令的适用范围。
- 正确估算指令的字节数。

一般地，操作码占 1 字节；在操作数中，直接地址 direct 占 1 字节、#data 占 1 字节、#data16 占两字节；操作数中的 A、B、R0～R7、@Ri、DPTR、@ A+DPTR、@ A+PC 等均隐含在操作码中，下面分别予以说明。

### 3.3.1　数据传送类指令

数据传送类指令是把源操作数传送到指定目的操作数。指令执行后，源操作数的内容不变，而目的操作数的内容被修改；若要求在传送时不丢失目的操作数，则用交换传送类指令。

数据传送类指令说明：

- 数据传送类指令对程序状态字 PSW 的 Cy、Ac、OV 位不产生影响。
- 数据传送类指令寻址范围：累加器 A、片内 RAM、SFR 和片外 RAM。
- 数据传送类指令功能：(目的地址)←(源地址)。

#### 1. 8 位数据传送指令(15 条)

8 位数据传送指令是在 80C51 内部 RAM 和特殊功能寄存器 SFR 间的传送，指令助记符为“MOV”，源操作数的寻址方式可以是立即寻址、直接寻址、寄存器寻址和寄存器间接寻址。

(1) 以累加器 Acc 为目的地址的传送指令。

其功能是将源操作数的内容送入累加器 Acc。指令的表现形式如下所示。

```
MOV  A,#data          ; A ← data
MOV  A,direct         ; A ←(direct)
MOV  A,Rn             ; A←(Rn)
MOV  A,@Ri            ; A←((Ri))
```

**【例 3-1】** 以累加器 Acc 为目的地址的传送指令应用，设(R1)＝50H，(45H)＝20H，(50H)＝10H。

```
MOV A,#45H            ; 立即寻址，将 8 位立即数 45H 送入累加器，(A) = 45H
MOV A,45H             ; 直接寻址，将 RAM 中 45H 单元的内容送入累加器，(A) = 20H
MOV A,R1              ; 寄存器寻址，将 R1 的内容送入累加器，(A) = 50H
MOV A,@R1             ; 寄存器间接寻址，将 R1 指示的内存单元 50H 中的内容送累加器，(A) = 10
```

(2) 以直接地址为目的地址的传送指令。

其功能是将源操作数的内容送入片内 RAM 存储单元。指令的表现形式如下所示。

```
MOV direct,#data        ; direct ← data
MOV direct1,direct2     ; direct1 ←(direct2)
MOV direct,A            ; direct ←(A)
MOV direct,@Ri          ; direct ←((Ri))
MOV direct,Rn           ; direct ←(Rn)
```

**【例 3-2】** 以直接地址为目的地址的传送指令应用，设(13H)＝15H，(30H)＝11H，(A)＝12H，(R2)＝13H。

```
MOV 20H,#23H      ; 立即寻址，将 8 位立即数 23H 送内部 RAM 20H 单元，(20H) = 23H
MOV 20H,30H       ; 直接寻址，将内部 30H 单元的内容送内部 RAM 20H 单元，(20H) = 11H
MOV 20H,A         ; 寄存器寻址，将累加器 A 的内容送内部 RAM 20H 单元，(20H) = 12H
MOV 20H,R2        ; 寄存器寻址，将 R2 的内容送入内部 RAM 20H 单元，(20H) = 13H
MOV 20H,@R2       ; 寄存器间接寻址，将 R2 指示的内存单元 13H 中的内容送入内部 RAM 20H 单元，(20H)
```

(3) 以通用寄存器 R*n* 为目的地址的传送指令。

其功能是将源操作数的内容送入当前工作寄存器区的 R0～R7 中的某一个寄存器。指令的表现形式如下所示。

```
MOV   Rn,A          ; Rn ← (A)
MOV   Rn,direct     ; Rn←(direct)
MOV   Rn,#data      ; Rn← data
```

**【例 3-3】** 以通用寄存器 R*n* 为目的地址的传送指令应用，设(A)＝26H，(30H)＝35H。

```
MOV R1,A            ; 寄存器寻址，将累加器 A 的内容送 R1，(R1) = 26H
MOV R2,30H          ; 直接寻址，将内部 30H 单元的内容送 R2，(R2) = 35H
MOV R5,#30H         ; 立即寻址，将 8 位立即数 30H 送 R5，(R5) = 30H
```

(4) 以通用寄存器间接地址为目的操作数的传送指令。

其功能是将源操作数的内容送入以 R0 或 R1 为地址指针的内部 RAM 单元中。指令的表现形式如下所示。

```
MOV  @Ri,A          ; (Ri) ← (A)
MOV  @Ri,direct     ; (Ri) ←(direct)
MOV  @Ri,#data      ; (Ri) ← data
```

**【例 3-4】** 以通用寄存器间接地址为目的操作数的传送指令应用，设(R0)＝30H，(R1)＝50H，(40H)＝22H，(A)＝66H。

```
MOV @R0,A           ; 寄存器寻址，将累加器 A 的内容送 R0 指示的内存单元，(30H) = 66H
MOV @R1,40H         ; 直接寻址，将内部 40H 单元的内容送 R1 指示的内存单元，(50H) = 22H
MOV @R1,#40H        ; 立即寻址，将 8 位立即数 40H 送 R1 指示的内存单元，(50H) = 40H
```

综合上述内容可以看出：内部数据传送指令的传送关系如图 3-7 所示。

### 2. 16 位数据传送指令(1 条)

当需要对片外的 RAM 单元或 I/O 端口进行访问时，或进行查表操作时，必须将 16 位地址赋给地址指针 DPTR，这就必须使用 16 位数据传送指令，这也是 80C51 指令系统中唯一的一条 16 位数据传送指令。

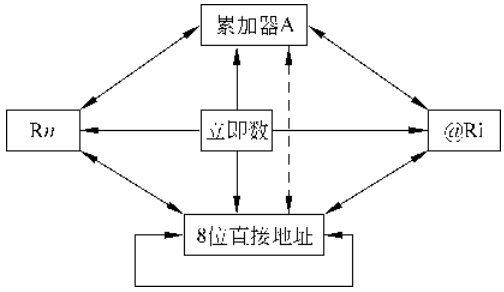


图 3-7　MOV 指令应用关系图

指令格式为：MOV　DPTR,＃data16　；属于立即寻址方式

例如：MOV DPTR,＃2345H　；将 16 位立即数 2345H 送入地址指针 DPTR,(DPH) = 23H,(DPL) = 45H

### 3. 外部数据传送指令(4 条)

当 80C51 CPU 与外部数据存储器或 I/O 端口之间进行数据传送时，只能通过累加器 Acc 进行。其指令助记符为：MOVX，其中的 X 表示外部(External)，指令的表现形式如下所示。

```
MOVX   A, @DPTR       ; A ←((DPTR))
MOVX   @DPTR,A        ; (DPTR)←(A)
MOVX   A,@Ri          ; A ←((Ri))
MOVX   @Ri,A          ; (Ri)←(A)
```

**注意：**

① 使用 R*i* 时，只能访问低 8 位地址为 00H～FFH 的地址段。

② 使用 DPTR 时，能访问 0000H ～ FFFFH 的地址段。

**【例 3-5】** 将外部 RAM 的 4000H 单元初始化设置为 0。

```
MOV A,＃0
MOV DPTR,＃4000H
MOVX @DPTR,A
```

**【例 3-6】** 从外部设备的 3000H 端口读入数据，并将该数据送当前工作寄存器 R3。

```
MOV DPTR,＃3000H
MOVX A,@DPTR
MOV R3,A
```

### 4. 查表指令(2 条)

查表指令也称为 ROM 数据传送指令，功能是实现从程序存储器 ROM 中读取数据。其指令助记符为：MOVC，其中的 C 表示代码(Code)，指令的表现形式如下所示。

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

**说明：**MOVC A, @A＋PC 指令是单字节指令，其功能是以程序计数器 PC 的当前值

(下一条指令的起始地址)作为基地址,以累加器 A 中的内容作为偏移地址量,两者相加后得到一个 16 位地址,然后把与该地址对应的 ROM 单元中的内容送累加器 A。该指令的优点是不改变 PC 的状态,仅根据累加器 A 的内容即可读取表格中的内容;缺点是表格只能存放在该查表指令后面的 256B 范围内,因此表格只能被一段程序所用。

**【例 3-7】** 设在程序存储器 ROM 中存放了字符 0～4 的 ASCII 码表,通过查表找出字符 3 的 ASCII 码送入当前工作寄存器 R1。

```
    MOV A,#3              ;机器码 74H 和 03H,占 2B
    ADD A,#1              ;偏移量修整,机器码 24H 和 01H,占 2B
    MOVC A,@A+PC          ;查表,机器码 83H,占 1B
    MOV R1,A              ;机器码 F9H,占 1B
ASC:  DB 30H,31H,32H,33H,34H
```

**说明**:设上述程序段的存放起点在 1000H,由于从查表指令到数据表的首地址之间的距离为 1B,所以在执行查表指令前必须有偏移量修整,ASCII 码表的起点地址为 1006H。

从例 3-7 可以看出:执行 MOVC A,@A+PC 指令后,PC 的值不发生变化,仍指向下一条指令。该指令的执行过程可分为以下几个步骤。

(1) 将所查表格数据的位置号(即数据在表格中的位置号,如上例中的 33H 的位置号是 3)送累加器 A。

(2) 计算偏移量(rel):查表指令到数据表首地址之间的距离。

(3) 偏移量(rel)=表格首地址-PC 指针的当前值=表格首地址-(MOVC 指令所在的地址+1),再将 rel 作为修整量,加入到累加器 A。

(4) 执行查表指令 MOVC A,@A+PC,将查表的结果送回到累加器 A。

MOVC A, @A+DPTR 指令也是单字节指令,其功能是以 DPTR 作为基址寄存器(存放表格的首地址),以累加器 A 中的内容作为偏移地址量,两者相加后得到一个 16 位地址,然后把与该地址对应的 ROM 单元中的内容送累加器 A。该指令的优点是执行结果只和地址指令 DPTR 及累加器 A 的内容有关,与该指令在程序中的位置和表格的存放位置无关,因此表格的大小和位置可以在 64KB 程序存储器中任意安排,且同一表格可供多个程序块使用。

**【例 3-8】** 用 MOVC A, @A+DPTR 指令解例 3-7。

```
    MOV A,#3              ;机器码 74H 和 03H,占 2B
    MOV DPTR,#ASC         ;或取表格首地址,占 3B
    MOVC A,@A+DPTR        ;查表,机器码 83H,占 1B
    MOV R1,A              ;机器码 F9H,占 1B
    …
ASC: DB 30H,31H,32H,33H,34H
```

从例 3-8 可以看出:执行 MOVC A,@A+DPTR 指令后,DPTR 的值不发生变化。其执行过程可分为 3 个步骤。

(1) 将所查表格数据的位置号(即数据在表格中的位置号,如上例中的 33H 的位置号是 3)送累加器 A。

(2) 表格首地址送 DPTR。

(3) 执行查表指令 MOVC A,@A+DPTR,将查表的结果送回到累加器 A。