MCS—51

单片微机原理与应用

（下）

复旦大学计算机系微机开发应用研究室编
复旦科教仪器厂印
八六年六月

下 册 目 录

第十章　程序设计

§1　程序设计方法

本章将结合MCS－51指令的特点，介绍一些常用程序的编制方，希望这一章能对用户使用MCS－51微计算机的指令编制程序之平有所提高。

一、查表程序

在用户的控制系统中，查表程序是必不可少的一种程序，它对于户的缩短程序，提高编程效益有着相当的用处。什么是查表程序呢？表所用的表格构造怎样？在下面首先介绍表格的由来以及构造。

查表，也就是说根据某个x，进入表格中寻找y，使y＝f(x)

x有二种表示方法

·规则表示方法

x的值为0，1，2，3……n

y的值为y_0，y_1，y_2，y_3……y_n

现给出相应的x值，要求得相应的y值，（当然，y值可能是字节，二字节，三字节……。也就是说，y_0…y_n要么均为一字节，么均为二字节，要么均为三字节……。表格中不允许y_i为2字节，y_j为一字节，这样就使表格的构造较为简单，编制查表程序也较方便。

这种表格的查找方法如下：设进入口A中是x的取值数）

```
DPTR        表首
A * y所含的字节数==>A
 A   保存                          取出y_i所包含
MOVC  A，  @A+DPTR                 的所有字节的内容
 A   恢复
DPTR+1==>DPTR
MOVC   A，@A+DPTR
   ⋮
返回
```

表首地址：DB y_{01}
y_{02}
⋮
y_{0m} } 设 y_0 拥有 m 个字节内容

y_{11}
y_{12}
⋮
y_{1m} } y_1 拥有 m 个字节内容

⋮
⋮

y_{n1}
y_{n2}
⋮
y_{nm} } y_n 拥有 m 个字节内容

例如：现在完成一个控制装置的实现，根据最后得出的数字（0，1，2，…9），输出相应的控制字，每个控制字由二个字节组成，对应关系如下：

0	1	2	3	4	5	6
75FE	F087	FF09	559E	5597	789A	A430

7	8	9
1110	8754	FFEE

下面是一查表子程序，根据给出的数字（存放在20H单元中）查表所得的结果存放于22H，23H二单元。

```
MOV   DPTR,#TABEL     ;表首⇒DPTR
MOV   A,20H           ;x⇒A
RL    A               ;A*2⇒A
MOV   20H,A           ;A保存
```

```
        MOVC    A,@A+DPTR       ;查表，方位结果=>A
        MOV     22H,A           ;方位结果=>22H
        INC     DPTR            ;取该栏的下一单元的
        MOV     A, 20H            结果
        MOVC    A,@A+DPTR
        MOV     23H,A           ; 低位结果=>23H
        RET
TABEL:  DB      75H
        DB      0FEH            0   对应的控制字
        DB      0F0H
        DB      87H             1   对应的控制字
        DB      0FFH
        DB      09H             2   对应的控制字
        DB      55H
        DB      9EH             3   对应的控制字
        DB      55H
        DB      97H             4   对应的控制字
        DB      78H
        DB      9AH             5   对应的控制字
        DB      0A4H
        DB      30H             6   对应的控制字
        DB      11H
        DB      10              7   对应的控制字
        DB      87H
        DB      54H             8   对应的控制字
        DB      0FFH
        DB      0EEH            9   对应的控制字
```

上述的查表法有什么局限呢？一个很大的局限在于表格所占的单元数不能超过256个，对于表格的长度大于256个单元的表格，

我们利用DPH，DPL的加法特性进行查表。

设在上例中，它们的对应关系存在512对，表格的长度为1024，所以说，最后所得的数字在0～511之间，这个数据占据二个单元20H，21H，查表结果存放于22H，23H中。

```
        MOV   DPTR, #TABEL      ; 表首=> DPTR
        MOV   A,21H             ; 20H'21H'*2=>20H21H
        CLR   C
        RLC   A
        MOV   21H, A
        MOV   A,20H
        RLC   A
        MOV   20H, A
        MOV   A,21H             ; 20H'21H'+DPTR'=>
                                  DPTR
        ADD   A,DPL
        MOV   DPL,A
        MOV   A,20H
        ADDC  A,DPH
        MOV   DPH,A
        CLR   A                 ; A<=0
        MOVC  A,@A+DPTR         ; @DPTR =>A(程序存贮器)
        MOV   22H, A              结果=> 22H
        INC   DPTR              ; DPTR'+1=>DPTR
        CLR   A                 ; A <= 0
        MOVC  A,@A+DPTR         ; @DPTR =>A(程序存贮器)
        MOV   23H, A            ; 结果=> 23H
        RET                     ; 返回
TABEL:  DB    75H
        DB    0FEH              0  对应的控制字
```

```
DB      0F0H
DB      87H        } 1  对应的控制字
  ⋮
  ⋮
  ⋮
DB      0AAH
DB      0CCH       } 511 对应的控制字
```

·非规则变量x

上述表格的构成均为存在规律变量x，那么对应的y也相应易于寻找，如果存在非规则变量，即x是非〔0…n〕中的所有数，而是对于某些正整数i，f(x)无定义，也就是说，在〔0……n〕区域中，只有在部分正整数与f(x)才有定义，那么表格的构造方式如下：

```
DB      x低        } X
DB      x高        }

DB      y低        } y=f(x)
DB      y高        }
```

表格的每一项由四个单元组成，第一单元存放x的低八位，第二单元存放x的高八位，第三单元存放f(x)的低八位，第四单元存放f(x)的高八位

对应关系如下

```
x:      0000    01F5    0A087  ……  0FF07H
          ↕       ↕       ↕            ↕
y=f(x): 7789H   7FE0H   89FEH  ……  0E986H
```

共存在n个对应关系

进入下段子程序以前，20H，21H存放着给出的x值，要求查出对应的值，结果放入22H、23H中，表格末地址十1存放于24H 25H单元中。

```
        MOV   DPTR, #TABEL     ;表首=> DPTR
RE:     CLR   A
        MOVC  A,@A+DPTR        ;@DPTR=>A(程序存贮器)
        CJNE  A,21H,NEXT1      ;x低位相等吗?
        INC   DPTR             ;相符,准备查高位
        CLR   A
        MOVC  A,@A+DPTR
        CJNE  A,20H,NEXT2      ;高位相等吗?
        INC   DPTR             ;相等,取出相应的控制字
        CLR   A
        MOVC  A,@A+DPTR
        MOV   23H, A
        INC   DPTR
        CLR   A
        MOVC  A,@A+DPTR
        MOV   23H,A
        CLR   A                ;0 => A
        RET                      查到正确返回
NEXT1:  INC   DPTR             ;这一项不是
NEXT2:  INC   DPTR               DPTR+4=> DPTR
        INC   DPTR
        INC   DPTR
        MOV   A,25H            ;到表底吗?
        CJNE  A,DPL,RE
        MOV   A,24H
        CJNE  A,DPH,RE
        MOV   A,#0FFH          ;到, 0FFH => A
        RET                      查不到返回
TABEL:  DB    00H  }
        DB    00H  }  0000
```

```
DB   89H
DB   77H        0000对应的控制字
DB   F5H
DB   01H        01F5
DB   0E0H
DB   7FH        01F5H对应的控制字
DB   87H
DB   0A0H       0A087
DB   FEH
DB   89H        0A087对应的控制字
 .        .
 .        .
 .        .
 .        .
 .        .
DB   07H
DB   0FFH       0FF07
DB   86H
DB   0E9H
```

二、分支程序

分支程序的意义在于根据某一单元的内容，分别转入处理程序0，处理程序1，……

下面的程序，是完成根据20H单元内容转入不同的处理程序。

20H'=0	转处理程序0	program0
20H'=1	转处理程序1	program1
20H'=2	转处理程序2	program2

```
                .
                .
                .
                .
20H'=nH     转处理程序n      programn
```

程序如下：

```
        MOV     DPTR, # TABEL       ;表首 => DPTR
        MOV     A, 20H              ; 20H'*2=> A
        CLR     C
        RLC     A
        JNC     NADD                ;有否进位
        INC     DPH                 ;有,DPH'+1=>DPH
NADD:   AJMP    @A+DPTR             ;散转
TABEL:  AJMP    program0
        AJMP    program1
        AJMP    program2
                .
                .
                .
        AJMP    programn
```

这样的分支程序有些局限，鉴于AJMP的范围，这就要求所有的处理程序入口program0，program1，program2，……programn和分支程序段均位于2K范围之内。为了避免这种情况，可以把AJMP换为LJMP。

程序如下：

```
        MOV     DPTR, # TABEL   ; 表首 => DPTR
        MOV     A, 20H            20H'*3=>BA
        MOV     B, # 03H        ; B'+DPH' =>DPH
        MUL     AB
        PUSH    A
        MOV     A,B
        ADD     A,DPH
```

```
        MOV     DPH,A
        POP     A
        JMP     @ A+DPTR                散转
TABEL:  LJMP    program 0
        LJMP    program 1
        LJMP    program 2
          .
          .
          .
        LJMP    program n
```

当散转点超过256个时，即n＞255时，20H一个单元已存放不下这个散转数，该数必须存放在二个单元中。下面程序20H 21H存放着散转数，根据其内容散转：

```
        MOV     DPTR, #TABEL    ;表首=>DPTR
        MOV     A, 20H          ;20H'21H'*3=>BA
        MOV     B, # 03A         B'+DPH'=>DPH
        MUL     AB
        ADD     A, DPH
        MOV     DPH,A
        MOV     A,21H
        MOV     B,#03H
        MUL     AB
        PUSH    A
        MOV     A,B
        ADD     A,DPH
        MOV     DPH,A
        JMP     @ A+DPTR
TABEL:  LJMP    program 0       ;散转
        LJMP    program 1
        LJMP    program 2
          ⋮
```

LJMP　　program n

利用上述方法可以进行多达64K表格的散转。

如果不利用JMP @A+DPTR这条指令来实现散转，那么利用RET 这条指令也能实现同样目标。

例：设20H、21H中存放着散转数，也就是说，根据20H，21H内容散转。其中20H中存放着高八位，21H存放着散转数的低八位。表格的构造如下。

第i项　DB programi入口地址低八位
　　DB programi入口地址高八位　} programi入口地址

程序如下：

```
        MOV     DPTR, #TABEL    ;表首=> DPTR
        MOV     A, 21H          ; 20H'21H'*2=>
                                  20H 21H
        CLR     C
        RLC     A
        MOV     21H, A
        MOV     A, 20H
        RLC     A
        MOV     20H, A
        MOV     A, 21H          ; 20H'21H'+DPTR'
        ADD     A, DPL            => DPTR
        MOV     DPL, A
        MOV     A, 20H
        ADDC    A, DPH
        MOV     DPH, A
        CLR     A               ; A清零
        MOVC    A, @A+DPTR      ; @DPTR=>A（程序
                                  存贮器）
        PUSH    A               入口地址低八位进栈
```

```
        CLR     A
        INC     DPTR
        MOVC    A,@ A+DPTR
        PUSH    A                       入口地址高八位进栈
        RET                             退栈，返回，散转到
                                        入口地址
TABEL   DB      program0入口地址低八位  }
        DB      program0入口地址高八位  } program 0
        DB      program1入口地址低八位  }
        DB      program1入口地址高八位  } program 1
                  .
                  .
                  .
                  .
        DB      program n入口地址低八位 }
        DB      program n入口地址高八位 } program n
```

三、循环程序

在程序中，往往要求某一段程序重复执行多次，这时候就可用循环程序结构，这有助于程序的缩短，一个循环结构由以下三部分组成。

1．循环体：就是要求某一段程序重复执行的程序段部分。

2．循环结束条件：在循环程序中必须给出循环结束的条件。常见的循环是计数循环，循环了一定次数后就结束循环。

3．循环初态：用于循环过程的工作单元。在循环开始时往往要置以初态。即分别赋其一个初始值。

例：多进制乘法

x，y_n，y_{n-1}，……y_0均为8位二进制的数，要完成

$x*(y_n 2^{n\cdot 8}+y_{n-1}\cdot 2^{(n-1)\cdot 8}+\cdots\cdots+y_0)$这一相乘的功能，进入此子程序以前$R_0$存放的是指向乘数的地址。如下所示

$$@R_n \rightarrow y_0$$
$$@(R_0'+1) \rightarrow y_1$$
$$\vdots$$
$$@(R_0'+n) \rightarrow y_n$$

R，指出存放结果的单元首址（先低后高），20H中存放着n＋1这个数。24H单元中存放着x程序如下：

```
        MOV   R7, 20H          ; 把@R1所指的单元共
        INC   R7                 20H'+1 个均清 0
SET0:   MOV   @R1, #00H
        INC   R1
        DJNZ  R7, SET0
        SETB  C                ;   恢复R1
        MOV   A,R1
        SUBB  A,20H
        MOV   R1, A
RE:     MOV   B,24H            ; x*y1+@R1'=>@(R1+1)
        MOV   A,@R0
        MUL   AB
        ADD   A,@R1
        MOV   @R1,A
        INC   R0               ; R0'+1 => A
        INC   R1               ; R1'+1 => A
        MOV   A, B
        ADDC  @R1, A
        MOV   @R1, A
        DJNZ  20H, RE          ; 循环结束吗?
        RET                    ; 返回
```

循环体

四、逻辑操作

MCS—51 给用户提供丰富的位操作功能，用户可以使用这些位操作指令，实现原由硬件所实现的功能，不过存在一个缺点，这就是软件所需的时间较长，也就是所需延时较多。

例：用软件实现下式：

$$y = x_0x_1x_2 + \overline{x}_0x_1x_2 + x_0\overline{x}_1x_2x_3 + \overline{x}_4x_5x_6x_7 + x_4x_5x_6 + x_4\overline{x}_5x_6x_7$$

程序如下：

```
MOV   DPTR, #ADDRESS    ;设：通过8155的口输入
MOVX  A,@DPTR             通过p1.0输出,ADDRESS
MOV   C, A.0              是8155 A口地址
ANL   C, A.1            ; 完成 x0·x1·x2 => 00H
ANL   C, A.2
MOV   00H, C
MOV   C, A.0            ; 完成 x0·x̄1·x̄2 => C
ANL   C, /A.1
MOV   01H,C             ; x0·x̄1暂存01H中
ANL   C, /A.2
ORL   C, 00H            ; 完成 x0x1x2+x0x1x̄2
                          => C
MOV   00H, C            ; C暂存
MOV   C, A.2            ; 完成 x0x1x2x3 → C
ANL   C, 01H
ANL   C, A.3
ORL   C, /00H           ; 完成 x0x1x2+x0x1x̄2
MOV   00H,C               +x0x̄1x2x̄3 => 00H
MOV   C,A.7             ; 完成 x̄4x̄5x̄6x̄7 => C
ANL   C,/A.6
ANL   C,/A.5
```

```
ANL     C,/A.4
ORL     C, 00H          ;完成 x0x1x2+x0x̄1x̄2
MOV     00H,C             +x0x̄1x2x3+x4x5x6x7
                          => 00H
MOV     C, A.4          ;完成 x4x5x6 => C
ANL     C, A.6
MOV     01H,C
ANL     C,A.5
ORL     C,00H           ;完成 x0x1x2+x0x̄1x̄2
MOV     00H,C             +x0x1x2x3+x4x5x6x7
MOV     C,A.7             +x4x5x6 => C
ANL     C,01H
ANL     C,/A.5
ORL     C,00H           ;完成y式的计算结果在C
MOV     P1.0,C          ;输出
```

从上段程序可以看到无论多么复杂的逻辑代码式均可化为软件实现。不过略为费时。

五、二进制小数转换成10进制小数的方法

10进制小数的表示方法为

$$y_0\cdot10^{-1}+y_1\cdot10^{-2}+y_2\cdot10^{-3}+\cdots$$

其中 y_i 均为十进制数

二进制小数的十进制表示方法

$$x_0\cdot2^{-1}+x_1\cdot2^{-2}+x_2\cdot2^{-3}+\cdots\cdots$$

其中 x_i 均为二进制数

已知一个数的二进制表示方法，要求其十进制的表示方法，也就

是说，已知x_i，要求y_i，即

$$x_0\cdot 2^{-1}+x_1\cdot 2^{-2}+x_2\cdot 2^{-3}\cdots=y_0\cdot 10^{-1}+y_1\cdot 10^{-2}+y_2\cdot 10^{-3}+\cdots\cdots$$

从这个式子可以看出，要求得y_0，只要式子两边乘上10，式子右边$=y_0+y_1 10^{-1}+y_2 10^{-2}+\cdots\cdots$

式子左边$=(x_0\cdot 2^{-1}+x_1\cdot 2^{-2}+x_2\cdot 2^{-3}+\cdots)\times 10$

根据整数=整数，小数等于小数的原理，得知，y_0实际上等于$(x_0\cdot 2^{-1}+x_1\cdot 2^{-2}+\cdots\cdots)\times 10$的整数部分，以此类推，可以求得所有的$y_i$。

例：设20H、21H两单元中存放着二进制小数。20H中存放着高八位小数。21H中存放着低八位小数。现把这些小数转换成十进制数，形成压缩的BCD码存放在30H、31H、32H中。程序如下：

```
        MOV     R0, #30H        ;置R0=30H（指针）
        MOV     30H, #00H       ;置30H31H32H内容为0
        MOV     31H, #00H
        MOV     32H, #00H
        MOV     23H, #06H       ;形成6位十进制小数
RE:     MOV     A, 21H          ; 20H'21H'*0A =>
        MOV     B, #0AH           B 20H 21H
        MUL     AB
        MOV     21H, A
        MOV     R7, B
        MOV     A, 20H
        MOV     B, #0AH
        MUL     AB
        ADD     A, B7
        MOV     20H, A
```