

算术运算指令学习

MCS-51 具有较强的加、减、乘、除四则运算指令及加 1、减 1 和 2-10 进制调整指令。加、减、乘、除指令会影响程序状态字 PSW, 只有加 1 和减 1 指令不影响 PSW。

1. 加法指令

ADD A, #data 表示把立即数 data 与累加器 A 的内容相加, 相加的结果送累加器 A。

ADD A, direct 表示把直接寻址单元 direct 中的内容与累加器 A 的内容相加, 相加的结果送累加器 A。

ADD A, @Ri 表示将寄存器 Ri 中内容作为地址的单元内容(寄存器间接寻址单元)与累加器 A 的内容相加, 相加的结果送累加器 A。

ADD A, Rn 表示将寄存器 Rn 中的内容与累加器 A 的内容相加, 相加的结果送累加器 A。

上述指令, 把源字节变量(立即数、直接、间接地址单元, 工作寄存器内容)与累加器相加, 结果保存在累加器中, 影响标志 AC、CY、OV、P。

2. 带进位加法指令

ADDC A, #data 表示把立即数 data 与累加器 A 的内容及进位标志 CY 相加, 相加的结果送累加器 A。

ADDC A, direct 表示把直接寻址单元 direct 中的内容与累加器 A 的内容及进位标志 CY 相加, 相加的结果送累加器 A。

ADDC A, @Ri 表示将寄存器 Ri 中内容作为地址的单元内容(寄存器间接寻址单元)与累加器 A 的内容及进位标志 CY 相加, 相加的结果送累加器 A。

ADDC A, Rn 表示将寄存器 Rn 中的内容与累加器 A 的内容及进位标志 CY 相加, 相加的结果送累加器 A。

3. 带借位减法指令

SUBB A, #data 表示累加器 A 中

的内容减立即数 data 及借位标志 CY, 结果送累加器 A。

SUBB A, direct 表示累加器 A 中的内容减直接寻址单元 direct 中的内容及借位标志 CY, 结果送累加器 A。

SUBB A, @Ri 表示累加器 A 中的内容减寄存器间接寻址单元中的内容及借位标志 CY, 结果送累加器 A。

SUBB A, Rn 表示累加器 A 中的内容减寄存器 Rn 中的内容及借位标志 CY, 结果送累加器 A。

上述减法指令寻址方式, 执行过程与加法指令类似, 只须把加操作改为减操作即可。

在加法中 CY=1 表示有进位, CY=0 表示无进位, 在减法中 CY=1 则表示有借位, CY=0 表示无借位。

4. 乘法指令

MUL AB 该指令把累加器 A 和寄存器 B 中的 8 位无符号整数相乘, 16 位乘积的低字节在累加器 A 中, 高字节在寄存器 B 中, 如果积大于 255(0FFH), 则使溢出标志位 OV 置 1, 否则清 0, 运算结果总使进位标志 CY 清 0。

5. 除法指令

DIV AB 指令把累加器 A 中的 8 位无符号整数除以寄存器 B 中 8 位无符号整数, 所得商放在累加器 A 中, 余数存在寄存器 B 中, 标志位 CY 和 OV 均清零。若除数(B 中内容)为 00H, 则执行后结果为不定值, 并置位溢出标志 OV, 在任何情况下, 进位标志 CY 总清零。

6. 加 1 指令

INC A 表示累加器 A 中的内容加 1, 结果送累加器 A。

INC direct 表示直接寻址单元 direct 中的内容加 1, 结果送回原单元中。

INC @Ri 表示寄存器间接寻址单元中的内容加 1, 结果送回原单元中。

INC Rn 表示寄存器 Rn 中的内容加 1, 结果送回 Rn 中。

INC DPTR 表示数据指针 DPTR 中的内容加 1, 结果送回 DPTR 中。

INC 指令把所指出的变量加 1, 结果仍送回原地址单元, 运算结果不影响任何标志位。指令共使用三种寻址方式: 寄存器寻址、直接寻址、寄存器间接寻址。

7. 减 1 指令

DEC A 表示累加器 A 中的内容减 1, 结果送累加器 A。

DEC direct 表示直接寻址单元 direct 中的内容减 1, 结果送回原单元中。

DEC @Ri 表示寄存器间接寻址单元中的内容减 1, 结果送回原单元中。

DEC Rn 表示寄存器 Rn 中的内容减 1, 结果送回 Rn 中。

上述指令将指定变量减 1, 结果仍存在原指定单元, 不影响任何标志位。有三种寻址方式: 寄存器寻址、直接寻址、寄存器间接寻址。

8. 2-10 进制调整指令

DA A 本指令是对 2-10 进制的加法进行调整的指令。两个压缩型 BCD 码按二进制数相加, 必须经过本条指令调整后才能得到压缩型的 BCD 码和数。由于指令要利用 AC、CY 等标志位才能起到正确的调整作用, 因此它必须跟在加法(ADD、ADDC)指令后面才能使用。指令的操作过程为: 若相加后累加器低 4 位大于 9 或半进位位 AC=1, 则加 06H 修正; 若累加器高 4 位大于 9 或进位位 CY=1, 则加 60H 修正; 若两者同时发生或高 4 位虽等于 9 但低 4 位修正有进位, 则应进行加 66 修正。

好了, 介绍了一下指令, 可能初学的人又觉得枯燥了。不急, 下面开始做实验, 实验做完后再请你回头仔细地反复想想这些指令和实验之间的关系, 多琢磨琢磨, 这样理解吸收得就快。

现在我们做一个加法实验, 要求实现 52H、FCH 两数相加。在我的文档中建立一个文件目录(如 S3), 然后建立一个 S3.uv2 的工程项目, 最后建立源程序文件(如 S3.asm)。

输入下面的程序:

```

序号:1  ORG 0000H;
2  LJMP MAIN;
3  ORG 030H;
4  MAIN: ACALL DEL;
5  MOV A,#052H;
6  MOV R0,#0FCH;
7  ADD A,R0;
8  NOP;
9  MOV P1,A;
10 DEL: MOV R7,#0FFH;
11 DEL1:MOV R6,#0FFH;
12 DEL2: MOV R5,#01FH;
13 DEL3:DJNZ R5,DEL3;
14 DJNZ R6,DEL2;
15 DJNZ R7,DEL1;
16 RET;
17 END

```

编译通过后,将其烧录到 89C51 芯片中,将芯片插入到 S1 型 LED 输出试验板上,在 S1 实验板上通电运行后,P1 口的输出为 01001110 (0 代表 LED 亮),那么输出结果正确吗?让我们做一下二进制加法,立即数 52H=01010010B,立即数 FCH=11111100B,相加后为:

```

01010010B
11111100B
+ -----
1 01001110 B

```

转换成 16 进制后即 14EH。其中的 1 为进位,在 P1 口上是无法观察的(P1 口只有 8 位输出)。打开模拟仿真界面进行软件仿真,从左边的寄存器窗口可看到 PSW,点一下前面的+号,展开后可看到进位位 CY=1。

下面我们解释程序。

序号 1(程序解释,以下同):程序开始。

序号 2:跳转到 MAIN 主程序处。

序号 3:主程序 MAIN 从地址 0030H 开始。

序号 4:延时一会儿,做好观察准备。

序号 5:将立即数 52H 传送给累加器 A。

序号 6:将立即数 FCH 传送给寄存器 R0。

序号 7:将寄存器 R0 中的内容与累加器 A 的内容相加,相加的结果送累加器 A。

序号 8:空操作(稍等一下)。

序号 9:将累加器 A 中的内容传送给 P1 口。

序号 10~16:延时子程序。

序号 17:程序结束。

再做一个乘法指令实验,要求实现两个数 FFH、03H 相乘,被乘数放 A,乘数放 B,乘法运算结果的低 8 位放 A,高 8 位放 B。若积大于 255 时,结果溢出,OV 位置 1。建立一个文件目录(如 S4),然后建立一个 S4.uv2 的工程项目,最后建立源程序文件(S4.asm)。

输入以下程序:

```

序号: 1  ORG 0000H;
2  LJMP MAIN;
3  ORG 030H;
4  MAIN:ACALL DEL;
5  MOV A,#0FFH;
6  MOV B,#03H;
7  MUL AB;
8  MOV P0,A;
9  MOV P1,B;
10 DEL: MOV R7,#0FFH;
11 DEL1:MOV R6,#0FFH;
12 DEL2: MOV R5,#01FH;
13 DEL3:DJNZ R5,DEL3;
14 DJNZ R6,DEL2;
15 DJNZ R7,DEL1;
16 RET;
17 END

```

编译通过后,将其烧录到 89C51 芯片中,将芯片插入到 S1 型 LED 输出试验板上,在 S1 实验板上通电运行后,P0 口的输出为 11111101 (即 P0.1 的发光点亮),P1 口的输出为 00000010 (即除 P1.1 的发光管熄灭外其它七个发光管均点亮)。说明执行的结果是 A 内容为 FDH,B 的内容为 02H,即 FFH*03H=02FDH。这个结果对吗?FFH=255,03H=3,255*3=765,而 02FDH=2*162+15*16+13=765,结果正确。

我们解释一下程序。

序号 1(程序解释,以下同):程序开始。

序号 2:跳转到 MAIN 主程序处。

序号 3:主程序 MAIN 从地址 0030H 开始。

序号 4:延时一会儿,做好观察准备。

序号 5:将立即数 FFH 传送给累加器 A。

序号 6:将立即数 03H 传送给寄存器 B。

序号 7:进行乘法运算。结果是 16 位乘积的低字节在累加器 A 中,高字节在寄存器 B 中,如果积大于 255(OFFH),

则使溢出标志位 OV 置 1,否则清 0,运算结果总使进位标志 CY 清 0。

序号 8:将累加器 A 中的内容传送给 P0 口观察。

序号 9:将寄存器 B 中的内容传送给 P1 口观察。

序号 10~16:延时子程序。

序号 17:程序结束。

进行软件仿真时,从寄存器窗口中展开 PSW 后可看到溢出位 OV=1。

下来做加 1 指令的实验,让 P1 口的 8 个 LED 发光管模拟二进制的加法运算。还是老样子做(反复练习有助于记牢)在我的文档中建立一个文件目录(S4),然后建立一个 S4.uv2 的工程项目,最后建立源程序文件(S4.asm)。

输入以下程序:

```

序号: 1  ORG 0000H;
2  LJMP MAIN;
3  ORG 030H;
4  MAIN: MOV A,#00H;
5  PLAY: MOV P1,A;
6  ACALL DEL;
7  INC A;
8  AJMP PLAY;
9  DEL: MOV R7,#0FFH;
10 DEL1: MOV R6,#0FFH;
11 DEL2: MOV R5,#01FH;
12 DEL3: DJNZ R5,DEL3;
13 DJNZ R6,DEL2;
14 DJNZ R7,DEL1;
15 RET;
16 END

```

编译通过后,将其烧录到 89C51 芯片中,将芯片插入到 S1 型 LED 输出试验板上,在 S1 实验板上通电运行后,P1 口的输出从 00000000 (8 个 LED 均点亮)起按二进制做加法,→00000001→00000010→……最后为 11111111 (8 个 LED 均熄灭),然后重复循环。

我们对程序进行解释。

序号 1(程序解释,以下同):程序开始。

序号 2:跳转到 MAIN 主程序处。

序号 3:主程序 MAIN 从地址 0030H 开始。

序号 4:累加器 A 清零。

序号 5:将累加器 A 中的内容传送给 P1 口观察。

序号 6:调用延时子程序,便于观察清楚。

序号 7:累加器 A 中的内容加 1,结

果送回累加器 A。

序号 8: 跳转到标号 PLAY 处进行循环运行。

序号 9~15: 延时子程序。

序号 16: 程序结束。

最后我们会体会一下 2~10 进制调整指令的作用。在我的文档中建立一个文件目录(S5), 然后建立一个 S5.uv2 的工程项目, 最后建立源程序文件(S5.asm)。

输入以下程序:

```

序号: 1  ORG 0000H
      2  LJMP MAIN;
      3  ORG 030H;
      4  MAIN:20H,#00H;
      5  GOON:MOV A,20H;
      6  ANL A,#0FH;
      7  MOV DPTR,#TAB;
      8  MOVC A,@A+DPTR;
      9  MOV P0,A;
     10  MOV A,20H;
     11  SWAP A;
     12  ANL A,#0FH;
     13  MOVC A,@A+DPTR;
     14  MOV P1,A;
     15  ACALL DEL;
     16  INC 20H;
     17  AJMP GOON;
     18  DEL: MOV R7,#014H;
     19  DEL1:MOV R6,#0FFH;
     20  DEL2: MOV R5,#01FH;
     21  DEL3:DJNZ R5,DEL3;
     22  DJNZ R6,DEL2;
     23  DJNZ R7,DEL1;
     24  RET;
     25  ORG 0100H;
     26  TAB: DB 0C0H,0F9H,0A4H,
           0B0H,099H,092H,082H,0F8H
     27  DB 080H,090H,088H,083H,
           0C6H,0A1H,086H,08EH
     28  END

```

编译通过后, 将其烧录到 89C51 芯片中, 将芯片插入到 S2 型 LED 数码管试验板上, 通电后右边两个数码管从 00 起开始加法计数, P0(个位)加到 9 后, 再加一次就变成 A, 而不是我们习惯的 0。它一直要到 F(15)后, 再加一次才变成 0。即做的是 16 进制加法。同理, 整个两位数码管计数到 99 后下一次并不显示 00, 而是要到 FF 后才计数到 00。这种计数方法, 在实用上有些不便。如一台频率计总是按 10 进制进行计数显示, 不然除观察外还要换算, 非常麻烦。那么有什么办法呢? 我们再做一个实验看看。不过先得将这个程序的指令先解释清楚。

序号 1(程序解释, 以下同): 程序开始。

序号 2: 跳转到 MAIN 主程序处。

序号 3: 主程序 MAIN 从地址 0030H 开始。

序号 4: 将立即数 00H 传送给 20H 单元中。

序号 5: 将 20H 单元中的内容传送给累加器 A。

序号 6: 累加器 A 中的内容与立即数 0FH 相“与”, 即结果是将 A 中内容的高 4 位置 0, 保留低 4 位的内容。这个方法的技术用语叫屏蔽高 4 位。

序号 7: 将数据表格的首地址(0100H)存入 16 位数据地址指针 DPTR 中。

序号 8: 将累加器 A 中内容与 DPTR 中内容相加, 得到的结果作为另一个固定存储单元的地址, 将该单元中的内容取出后传送给累加器 A。显然, 如果 DPTR 中放一常数, 而 A 中为可变量, 则可进行变址寻址, 技术上常用作查表。

序号 9: 将累加器 A 中内容传送给 P0 输出口, 点亮“个”位数码管。

序号 10: 再将 20H 单元中的内容传送给累加器 A。

序号 11: 将累加器 A 中内容的高 4 位和低 4 位互相交换。

序号 12: 屏蔽 A 中高 4 位。

序号 13: 查表。

序号 14: 将累加器 A 中内容传送给 P1 输出口, 点亮“十”位数码管。

序号 15: 调用延时子程序, 便于观察。

序号 16: 20H 单元内容加 1。

序号 17: 跳转到标号 GOON 处继续执行。

序号 18~24: 延时子程序。

序号 25: 数据表格的首地址为 0100H。

序号 26~27: 数据表格内容。

序号 28: 程序结束。

再做一个实验, 建立一个文件目录(S6), 然后建立一个 S6.uv2 的工程项目, 最后建立源程序文件(S6.asm)。

输入以下程序:

```

序号: 1  ORG 0000H
      2  LJMP MAIN;
      3  ORG 030H;
      4  MAIN: 20H,#00H;

```

```

5  MOV A,20H;
6  GOON: CLR C;
7  ANL A,#0FH;
8  MOV DPTR,#TAB;
9  MOVC A,@A+DPTR;
10 MOV P0,A;
11 MOV A,20H;
12 SWAP A;
13 ANL A,#0FH;
14 MOVC A,@A+DPTR;
15 MOV P1,A;
16 ACALL DEL;
17 MOV A,20H;
18 INC A;
19 DA A;
20 MOV 20H, A;
21 AJMP GOON;
22 DEL: MOV R7,#014H;
23 DEL1:MOV R6,#0FFH;
24 DEL2: MOV R5,#01FH;
25 DEL3:DJNZ R5,DEL3;
26 DJNZ R6,DEL2;
27 DJNZ R7,DEL1;
28 RET;
29 ORG 0100H;
30 TAB: DB 0C0H,0F9H,0A4H,
       0B0H,099H,092H,082H,0F8H
31 DB 080H,090H,088H,083H,0C6H,
       0A1H,086H,08EH
32 END

```

编译通过后, 将其烧录到 89C51 芯片中, 将芯片插入到 S2 型 LED 数码管试验板上, 通电后右边两个数码管从 00 起开始加法计数, P0(个位)加到 9 后, 再加一次就变成 0。变为 10 进制加法。整个两位数码管计数到 99 后下一次显示 00, 然后又开始新一轮的加法计数。整个计数过程完全按 10 进制进行。哈哈真灵, 问题解决了。看看程序解释中怎么说。

序号 1(程序解释, 以下同): 程序开始。

序号 2: 跳转到 MAIN 主程序处。

序号 3: 主程序 MAIN 从地址 0030H 开始。

序号 4: 将立即数 00H 传送给 20H 单元中。

序号 5: 将 20H 单元中的内容传送给累加器 A。

序号 6: 进位位 CY 置 0。

序号 7: 屏蔽累加器 A 中高 4 位。

序号 8: 将数据表格的首地址(0100H)存入 16 位数据地址指针 DPTR 中。

序号 9: 查表。

序号 10: 将累加器 A 中内容传送给 P0 输出口, 点亮“个”位数码管。

序号 11: 再将 20H 单元中的内容传

(下转 24 页)

于(!=)。前四种关系运算符(>,<,<=,>=)的优先级相同,后两种关系运算符(=,!=)的优先级相同,前四种优先级高于后两种。

与算术运算符的优先级相比,关系运算符的优先级低于算术运算符,但却高于赋值运算符(=)。

例6 unsigned char function(unsigned char x,unsigned char y)

```
{
if(x>=y)    return  x ;
else        return  y ;
}
```

例6中函数function实现的功能是返回参量x,y中的大者,如果关系运算(x>=y)为真(1),则返回x,否则返回y。

八、C51中的逻辑运算

C51支持三种逻辑运算符:逻辑与(&&)、逻辑或(|)|)、逻辑非(!)。逻辑与(&&)和逻辑或(|)|)是双目运算符,要求有两个运算对象,逻辑非(!)是单目运算符,只要求一个运算对象。

只有当逻辑与(&&)的两个运算对象的值同时为真时,逻辑与(&&)的结果才为真;当逻辑或(|)|)的两个运算对象的值有一个为真时,逻辑或(|)|)的值就为真。

例7 unsigned char function(unsigned char

```
x ,unsigned char y ,unsigned char z )
{
if( (x>=y) && (x > z) ) return100 ;
lse return 0 ;
}
```

例7中只有当(x>=y)和(x > z)同时为真时,才返回100,否则返回0。

九、C51中的位操作

C51提供了丰富的位操作运算,如下表所示:

表4 C51的位操作运算符

位操作功能	运算符号	运算规则
按位与	&	参加运算的两个运算对象,若两者相应的位都为1,则该位结果值为1,否则为0。
按位或		参加运算的两个运算对象,若两者相应的位中,只要有一个为1,则该位结果值就为1。
按位异或	^	参加运算的两个运算对象,若两者相应的位值相同,则结果为0;若两者相应的位值不同,该位结果为1。
按位取反	~	是一个单目运算符,用来对一个二进制数按位取反,即1变0,0变1。
位左移	<<	将一个二进制数左移若干位,移位后空白位补0,移出的位舍弃。
位右移	>>	将一个二进制数右移若干位,移位后空白位补0,移出的位舍弃。

例8 main()

```
{
unsigned char x = 0x66,y= 0x02 ,z;
z = x << 3; /*16进制的0x66相当于二进制01100110,左移三位后的结果*/
/*果为:00110000,即16进制的0*30。也就是z=0x30。*/
z = x&y ;/*x=0x66,相当于二进制的01100110。*/
/*y=0x02,相当于二进制的00000010。*/
/*相与后的结果,即z=00000010即
```

0x02。*/

}

十、自增运算、自减运算及复合运算

自增运算符(++的作用)是使变量的值自动加1;自减运算符(--的作用)是使变量的值自动减1。对于变量i:

i++; 使用i之后再使i的值加1。

++i; 先使i的值加1,再使用i。

i--; 使用i之后再使i的值减1。

--i; 先使i的值减1,再使用i。

复合运算符是所有的二目运算符与赋值运算符(=)组合在一起的复合赋值运算符,C51支持10种复合赋值运算。即:+=, -=, *=, /=, %=, <<=, >>=, &=, ^=, |=。采用复合赋值运算的目的是简化程序,提高C程序的编译效率。

a+=b; 相当于 a=a+b;

a*=b; 相当于 a=a*b;

(上接22页) 送给累加器A。

序号12:交换累加器A中的高、低4位。

序号13:屏蔽A中高4位。

序号14:查表。

序号15:将累加器A中内容传送给P1输出口,点亮“十”位数码管。

序号16:调用延时子程序,便于观察。

序号17:20H单元中的内容传送给累加器A。

序号18:累加器A内容加1。

序号19:2-10进制调整。这条指令是实现10进制计数的关键,好好理解透,看一下上面的指令学习中的解释。需要指出的是,这条指令一定要跟在加法指令后(如本例中的INC A),否则不起作用。

序号20:累加器A中的内容传送给20H单元。

序号21:跳转到标号GOON处继续执行。

序号22~28:延时子程序。

序号29:数据表格的首地址为0100H。

序号30~31:数据表格内容。

序号32:程序结束。

细心的读者可能会问,序号6的CLR C指令有什么用呢?开始时进位CY=0,当计数过100后,CY=1,如果此后再做DA A 2-10进制调整,则会出错。如计数到101(当然“百”位数1是看不到的),此时进行DA A调整后,则A中的内容不会是01,而是另一个数(出错啦)。所以在程序转到GOON时,每次做进位位清零,则不会出错。这两个关于2-10进制调整与否对比的程序如不经过实验是很难得到强烈的感性认识,所以说单片机学习是一门实践性很强的技术课,大家只有跟着讲座多做实验,多练习,才能真正学会编程技术。

(下一讲继续介绍指令的学习及实验)。

配文优惠邮购(每次邮费保价费12元,汇款时一定要附上联系电话):Keil C51 Windows集成开发环境(已汉化光盘,邮购代号:K1):46元。TOP851多功能编程器(邮购代号:B1):400元。LED输出试验板(邮购代号:S1):90元。LED数码管输出试验板(邮购代号:S2):140元。5V高稳定专用稳压电源(邮购代号:D1):35元。邮购时只需在附言栏中写明邮购代号及数量即可。邮购地址:201103

上海市闵行区莲花路2151弄57号201室 联系人:吕超亚

电话:021-64066571

E-mail:zcxh2151@sohu.com

邮购短消息查询:13044152947

另:湖南湘潭市湖南工程学院南院218信箱一读者姓名不详,邮包被退二次,请速与吕超亚联系。