微机原理和接口技术

第六讲 指令系统与汇编程序4

提 纲

1. 指令系统概述
2. 寻址方式
3. 数据传送类指令
4. 算术运算类指令
5. 逻辑运算类指令
6. 控制转移类指令
7. 位操作指令
8. 查表指令的应用
9. 堆栈操作指令的应用
10. 十进制调整指令的应用
11. 逻辑指令与字节状态操作
12. 转移指令的应用

提 纲

7. 位操作指令

问答：

问题： 编程实现双字节无符号数相加，被加数放在内部RAM的20H和21H（低字节在前），加数放在2AH和2BH，结果送回20H和21H。

问答：

- 编程实现双字节无符号数相加，被加数放在内部RAM的20H和21H（低字节在前），加数放在2AH和2BH，结果送回20H和21H。

- 程序如下：

```
START：MOV R0, #20H      ;被加数的首地址
       MOV R1, #2AH      ;加数的首地址
       MOV A, @R0
       ADD A, @R1
       MOV @R0, A        ;和低字节保存到20H单元
       INC R0            ;R0指向被加数的高字节
       INC R1            ;R1指向加数的高字节
       MOV A, @R0
       ADDC A, @R1
       MOV @R0, A        ;和高字节保存到21H单元
```

- **程序执行之后，双字节相加的进位状态保存在Cy中。**

问答：

问题：下述指令执行后，问SP、A、B的内容分别是多少？

```
                ORG   0100H
0100H           MOV   SP, #40H
0103H           MOV   A, #30H
0105H           LCALL 0500H
0108H           ADD   A, #10H
010AH           MOV   B, A
010CH L1:       SJMP  L1

                ORG    0500H
0500H           MOV   DPTR, #010AH
0503H           PUSH  DPL
0504H           PUSH  DPH
0505H                 RET
```

问答：

解答：下述指令执行后，问SP、A、B的内容分别是多少？

```
                ORG   0100H
0100H           MOV   SP, #40H
0103H           MOV   A, #30H
0105H           LCALL 0500H
0108H           ADD   A, #10H
010AH           MOV   B, A
010CH L1:       SJMP  L1

                ORG    0500H
0500H           MOV   DPTR, #010AH
0503H           PUSH  DPL
0504H           PUSH  DPH
0505H                 RET
```

子程序中的2次PUSH，修改了保存有断点地址的堆栈栈顶，使得REI指令自动恢复的断点是2次PUSH的内容，所以子程序返回到了010AH的地址。

结果： （SP）=42H，（A）=30H，（B）=30H

位操作类指令

- 8051MCU具有一个功能齐全的位处理器，进位标志**Cy**为位累加位，可以执行多种“位”操作。所有位操作均采用直接寻址方式，寻址位空间为8051MCU的位寻址空间。17条位操作类指令可分为4组。

  - 位数据传送指令：2条
  - 位状态设置指令：6条
  - 位逻辑运算指令：4条
  - 位转移指令：5条

位操作类指令

- **1.位数据传送指令(2条)**

MOV C, bit ; C ← （bit）

MOV bit，C ;（bit）← C

➢ **功能：**实现位累加器（C）和其它位地址之间的数据传递。

bit表示可位寻址的各位地址（8051MCU中，共有211位）

例：

MOV P1.0, C ; 将C中的状态送到P1.0引脚上去

MOV C, P1.0 ; 将P1.0的状态送给C

位操作类指令

- **2.位状态设置指令(6条)**
- **(1)位清零指令：将C或指定位清零，助记符：CLR**(Clear)

```
CLR C              ; 使C=0
CLR bit            ; 使指定位地址中的值等于0
 例：CLR P1.0              ; 使P1.0变为0
```

(2)位置1指令：将C或指定位置1，助记符：SETB(Set Bit)

```
SETB C             ; 使C=1
SETB bit           ; 使指定位地址中的值等于1
 例：SETB  P1.7   ; 使P1.7变为1
```

位操作类指令

- **2.位状态设置指令(6条)**
- **(3)位取反指令：将C或指定位取反，助记符：CPL**(Complement)

```
CPL C        ;使C等于原来的相反的值，由1变为0，或由0变为1。
CPL bit      ;使指定的位等于原来相反的值，由0变为1或由1变为0。
```

- **3.位逻辑运算指令(4条)**
- **(1)位与指令**

```
ANL C, bit  ;C与指定的位地址的值相与，结果送回C
ANL C,/bit  ;先将指定的位地址中的值取出后取反，再和C相与，
            ；结果送回C
```

注意：指定的位地址中的值本身并不发生变化

位操作类指令

- **3.位逻辑运算指令(4条)**

(2)位或指令：

将指定位(bit)的内容或指定位内容取反后(原内容不变)与C的内容进行逻辑或运算。结果仍存于C中。

ORL C,bit

ORL C,/bit

位操作类指令

4.位转移指令(5条)

位转移指令的功能分别是判断Cy或bit是1还是0，符合条件转移到目的地址，否则继续顺序执行，共5条指令。

```
JC   rel          ;C=1, 跳到标号处，执行程序；C=0,不跳,顺序执行
                  ;（Jump if the Carry flag is set）
JNC  rel          ;C=0, 跳;C=1,不跳，顺序执行
                  ;（Jump if Not Carry）
JB  bit，rel      ;(bit)=1, 跳;(bit)=0,不跳（Jump if the Bit is set）
JNB  bit，rel     ;(bit)=0, 跳;(bit)=1,不跳
                  ;（Jump if the Bit is Not set），
JBC  bit，rel     ;(bit)=1,跳，并使(bit)←0;(bit)=0，不跳
                  ;（Jump if the Bit is set and Clear the bit）
```

位操作类指令

- **5.位操作指令举例**
- **例3-12：**设计程序实现P1.1和P1.2内容的互换。
- MOV　　C, P1.1　　;C ← P1.1
- MOV　　00H, C　　;（00H）← Cy
- MOV　　C, P1.2　　;Cy ← P1.2
- MOV　　P1.1, C　　;P1.1 ← Cy
- MOV　　C, 00H　　;Cy ← （00H）
- MOV　　P1.2, C　　;P1.2 ← Cy
- **例3-13：**设C=0，P0口的内容为00111010B。若执行以下指令后，分析C和P0的内容。
- CPL　P0.0
- CPL　C
- 执行结果：C＝1，P0.0=1，即（P0）=00111011B

提 纲

8. 查表指令的应用

查表指令

- 查表是指根据已知变量在表格中查找目标值的过程。在微控制器应用中，经常会遇到查表操作，如LED数码管显示中查找BCD对应的段码；微控制器系统中非线性补偿中查找修正系数以及数值换算等。
- 查表方法使得程序结构简单、执行速度快等优点。
- 8051MCU有近程查表和远程查表两条查表指令。

查表指令

1.近程查表指令

MOVC A,@A+PC ;（PC）←（PC）+1

; (A) ← ((A)+(PC)) ； A中内容看作无符号数

- 基址寄存器PC是下条指令首地址，即执行完查表指令后的PC，称为当前PC；PC值不可改变。
- 变址寄存器A是下条指令到常数表格中被访问字节的偏移量，范围是 0-255。

- **该指令只能查找本指令后256B范围内的数据表格，故称为近程查表。**

- **应用时比较麻烦，建议多实用远程查表指令**

查表指令

2.远程查表指令

MOVC A,@A+DPTR ;(A) ← ((A)+(DPTR))，A中内容看作无符号数

- 基址寄存器DPTR是常数，指向数据表格的首地址；
- 变址寄存器A为表格首址到被访问数据的地址偏移量；
- DPTR 、 A都可以改变，A的范围是0-255；DPTR的范围是0000H-0FFFFH。
- **该指令可以查找存放在64KB范围内的数据表格，故称为远程查表指令。**

查表指令

3.查表指令举例

- **例3-14：**设R3中的值小于等于0FH。分别使用远程查表指令和近程查表指令，查出R3的平方值，存回到R3中。

- （1）远程查表方式
- ORG 0100H
- SUB1:**MOV DPTR, #TABLE**
- MOV A, R3
- MOVC A, @A+DPTR
- MOV R3, A
- SJMP $
- ORG 0150H
- **TABLE: DB** 00, 01,04,09,16,25,36,49,
- 64,51H,64H,121,144,0A9H,
- 0C4H,0E1H
- END

程序说明

假设R3中的值为4

- 远程查表：DPTR指向数据表头的符号地址TABLE，待查数据（R3）送入A，则查表指令从TABLE+4这个该单元取出的数据就是4的平方值16。

查表指令

（2）近程查表方式

```
        ORG     0100H
SUB1:   MOV     A, R3
0101H   ADD     A,#REL    ;REL=3
0103H   MOVC    A, @A+PC
0104H   MOV     R3, A
0105H   SJMP    $

TABLE:  DB 00, 01,04,09,16,25,36,49,
(0107H)    64,51H,64H,121,144,0A9H,
           0C4H,0E1H
        END
```

REL=3：修正值是MOVC指令后一条指令首址，与表头地址的间隔。

程序说明
假设R3中的值为4

- 近程查表：当前PC指向下一条指令首址（PC=0104H），待查数据（R3）送入A（（A）=4），此时若不对A进行修正，则（A+PC）=0108H，从0108H单元取数得到01H，结果错误。**对于近程查表指令，通常需要对A进行修正，修正方法是加上一个修正值REL，REL是当前PC到数据表首址的间隔字节数。**本例中，间隔的2条指令共3字节，所以REL=3。修正后（A）=7，则将从010BH单元取数，得到16，结果正确。

提 纲

9. 堆栈操作指令的应用

堆栈操作指令的应用

- 堆栈指令PUSH和POP常用于**子程序、中断服务程序中的现场保护与恢复**，且要**成对使用**，不然无法使子程序和中断程序正常返回，而使微控制器系统崩溃。

使用堆栈操作指令时，应注意：

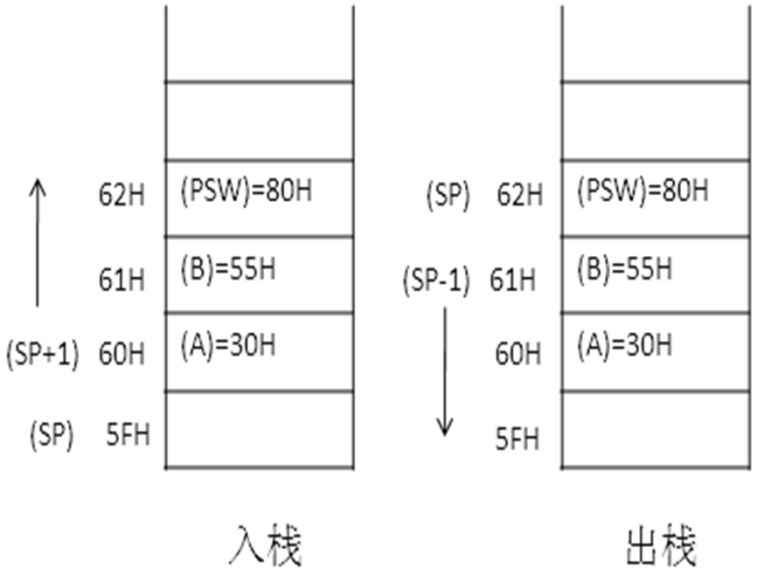
- 通过对SP重新赋值，可以改变堆栈区域；堆栈区域的设置应考虑堆栈的深度需求。
- 采用直接寻址方式，如PUSH R0和PUSH R1要写成PUSH 00H和PUSH 01H。不然汇编程序会报错。
- 堆栈的操作服从“先进后出”、“后进先出”规则，在子程序、中断服务程序中用于保护现场和恢复现场时，要注意入栈和出栈的次序。

堆栈操作指令的应用

1.堆栈指令的使用

例：主程序和子程序SUB1均用到寄存器A、B，子程序中的指令会影响PSW，因此子程序要对A、B、PSW的内容进行保护，并在返回前进行恢复。

```
SUB1:   PUSH   ACC   ；A的内容压入堆栈
        PUSH   B     ；B的内容压入堆栈
        PUSH   PSW   ；PSW的内容压入堆栈
               ……
        POP    PSW   ；从堆栈恢复PSW的内容
        POP    B     ；从堆栈恢复B的内容
        POP    ACC   ；从堆栈恢复A的内容
        RET
```



堆栈操作指令的应用

问题：简述下列程序段完成的功能，程序执行后SP指针指向哪里？

```
        MOV    SP, #2FH
        MOV    DPTR,#2000H
        MOV    R7, #50H
LOOP:   MOVX   A, @DPTR
        INC    DPTR
        PUSH   ACC

        DJNZ   R7, LOOP
        SJMP   $
```

堆栈操作指令的应用

解答：简述下列程序段完成的功能，程序执行后SP指针指向哪里？

```
        MOV    SP, #2FH          ;确定堆栈空间，从30H开始
        MOV    DPTR,#2000H       ;外部RAM首址
        MOV    R7, #50H          ;取数的个数
LOOP:   MOVX   A, @DPTR          ;从外部RAM取一个数
        INC    DPTR
        PUSH   ACC               ;取来的数压入堆栈，
                                 ; (SP)←(SP)+1,((SP))←(A)
        DJNZ   R7, LOOP
        SJMP   $
```

- 程序功能：将外部RAM 2000H开始的50H个数据传送到内部RAM 的30H开始的50H 个单元中。程序执行后SP指针指向7FH。

Thank you!