第3章

STC15单片机的指令系统

【学习目标】

- 掌握STC51系列单片机的寻址方式；
- 掌握数据传送指令的格式、功能和使用方法；
- 掌握算术运算类指令的格式、功能和使用方法；
- 掌握逻辑运算与移位类指令的格式、功能和使用方法；
- 掌握控制转移类指令的格式、功能和使用方法；
- 掌握位操作指令的格式、功能和使用方法。

【学习指导】

学习每一条指令时，应注意每条指令的格式、功能、操作数的寻址方式，以及对状态标志位的影响；可多看实例，并自己应用相关指令编写源程序有助于掌握指令的使用。

3.1 指令系统概述

指令是控制微控制器执行各类操作的控制命令。指令系统是特定处理器所能执行的全部指令的集合。不同的处理器有不同的指令系统，这是由该处理器厂商设计芯片时定义的。STC15单片机的指令系统，与经典51单片机的指令系统是全兼容的，包含111条指令。学习指令系统是掌握单片机的基础，是应用单片机的基本工具，是必须掌握的重要知识。

3.1.1 STC15单片机指令格式

微处理器的指令分为两种形式，一种是二进制代码形式的指令，例如二进制代码11101000，是将单片机片内R0寄存器的内容送往A累加器。这种二进制代码形式的指令，是CPU唯一能识别和处理的指令，称为机器指令。对于人来说，这种指令难以记忆和理解，因此将这些二进制代码用类似于英文的字符表示，这就是助记符的指令形式，又称为汇

编指令。例如,上述机器指令的助记符形式是:MOV A,R0,汇编指令和机器指令是简单的一一对应关系,这大大改善了指令的可读性和可理解性。但若要机器执行,必须先翻译成二进制代码表示的机器指令才行,这种翻译过程称为汇编,可以由计算机软件来自动完成。

STC15 单片机指令由操作码和操作数两部分组成。操作码表明该指令执行什么操作,它反映了指令的功能,对于汇编指令,操作码表现为指令功能的英文缩写。例如上面列举的例子中的 MOV,为数据传送指令的操作码。

操作数则表明在什么数据上执行该操作,也就是操作的对象。操作数可能是一个具体的常数,也可能是保存数据的存储器单元地址或寄存器。不同功能的指令,操作对象形式不同。绝大部分指令都有操作数,有的指令有一个操作数,有的指令有两个操作数。对于指令中的两个操作数,往往称一个操作数为源操作数,另一个为目的操作数。例如,上述 MOV A,R0,是传送类指令,将 R0 的内容传送至 A 累加器,R0 是源操作数,A 是目的操作数。一般在书写格式上是目的操作数在前,源地址写在后。

在 STC15 指令系统中,按指令包含的二进制代码的字节数,有 1B、2B 和 3B 等不同长度的指令,这表示了该指令在内存中所占的单元数;按指令执行所需要的系统时钟个数,有单周期指令、双周期指令、4 周期指令等,这表示了指令执行所需的时间。在学习指令时,应该注意指令的这两个属性。

汇编语言的一条语句的格式如下。

[标号:]　操作码助记符 [目的操作数][,源操作数] [;注释]

其中,方括号[]中的内容不是每一条语句必需的。

标号,表示本语句(指令)所在的地址,它可以作为转移指令的目标或子程序的入口(子程序名)。标号是由一串以字母开头的字母数字串。注释,是编程者对语句的说明,它是可选的。语句中的其他成分,本章和第 4 章中将会详细讲解。

3.1.2 指令的分类

STC51 单片机共有 111 条指令,其中,单字节指令 49 条,双字节指令 45 条,三字节指令 17 条。在 111 条指令中,有 64 条是单机器周期指令,45 条是双机器周期指令,只有乘法和除法指令需 4 个机器周期。STC51 单片机的指令系统按功能可划分为以下 5 类。

(1) 数据传送类指令 28 条;
(2) 算术运算类指令 24 条;
(3) 逻辑操作类指令 25 条;
(4) 位操作类指令 17 条;
(5) 控制转移类指令 17 条。

3.1.3 常用符号说明

以下符号是在介绍指令时所用到的。

Rn:当前寄存器组的 8 个通用寄存器 R0～R7(其中,n=0～7)。

Ri：可用作间接寻址的寄存器，只能是 R0、R1 两个寄存器(其中，i=0，1)。

Direct：内部 RAM 的 8 位地址。既可以是内部 RAM 的低 128 个单元地址，也可以是专用寄存器的单元地址或符号。因此在指令中 direct 表示直接寻址方式。

#data：8 位立即数。

#data16：16 位立即数。

addr16：16 位目的地址，只限于在 LCALL 和 LJMP 指令中使用。

addr11：11 位目的地址，只限于在 ACALL 和 AJMP 指令中使用。

rel：相对转移指令中的偏移量，为 8 位带符号补码数。

DPTR：数据指针。

bit：内部 RAM(包括专用寄存器)中的直接寻址位。

A：累加器。

B：B 寄存器。

C：进位标志位，它是布尔处理机的累加器，也称为累加位。

@：间址寄存器的前缀标志。

3.2 寻址方式

寻址方式就是指令中表示操作数或操作数地址的方式。一般来说，寻址方式越多，指令就越灵活，指令系统功能就越强。

STC15 系列单片机中，存放数据的存储空间有多类：内部基本的数据 RAM(包括一般的数据 RAM 和特殊功能寄存器 SFR)，STC15 扩展的片内数据 RAM，外部扩展的数据 RAM 和程序存储器空间。对于不同存储器中的数据操作，采用不同的寻址方式。STC15 共有 7 种寻址方式，下面分别介绍这几种寻址方式。

3.2.1 立即寻址

立即寻址是操作数作为指令的一部分而直接写在指令中，在指令中直接以常数形式表示要操作的数据，这种寻址方式为立即寻址。常数的操作数称为立即数。在 51 单片机的指令系统中，立即数用前面加“#”的 8 位二进制数(#data)或 16 位二进制数(#data16)表示。

【例 3.1】 以下指令的源操作数均为立即寻址。

```
MOV     A,#80H          ; 累加器 A←80H,十六进制表示的立即数
MOV     DPTR,#3200H     ; DPTR←3200H,十六进制表示的立即数
MOV     A,#00111100B    ; 二进制形式表示的立即数
MOV     A,#10           ; 十进制表示的立即数
```

3.2.2 直接寻址

指令中操作数直接以单元地址的形式给出，称为直接寻址。

直接寻址方式只能使用 8 位二进制地址，这种寻址方式的寻址空间为：内部基本的数

据 RAM，包括 SFR，但不包括附加的 RAM，即片内 80H～FFH 的附加的数据 RAM 区。特殊功能寄存器一般用寄存器名给出，也可以以单元地址形式给出。值得注意的是，直接寻址是访问特殊功能寄存器的唯一方法。

【例 3.2】 直接寻址示例。

```
MOV     A,34H           ; A←(34H),直接寻址单元(34H)内容送 A 累加器
MOV     P1,#36H         ; P1←36H,P1 是直接寻址
MOV     23H,45H         ; 将直接地址单元(45H)内容送到单元(23H)中
```

3.2.3 寄存器寻址

寄存器寻址就是操作数在寄存器中，因此指定了寄存器就能得到操作数。寻址指令中以符号名称来表示寄存器。寄存器寻址方式的寻址范围如下。

(1) 4 个寄存器组共 32 个通用寄存器。但在指令中只能使用当前寄存器组，因此在使用前常需通过对 PSW 中 RS1、RS0 位的状态设置，来进行当前寄存器组的选择。

(2) 部分专用寄存器。例如，累加器 A、B 寄存器以及数据指针 DPTR 等。

【例 3.3】 寄存器寻址示例。

```
MOV     A,R3            ; A←R3
MOV     B,R0            ; B←R0
PUSH    ACC             ; A 的内容推入堆栈
```

3.2.4 寄存器间接寻址

寄存器寻址方式寄存器中存放的是操作数，而寄存器间接寻址方式中，寄存器中存放的则为操作数的地址。CPU 执行这类指令时，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，并完成相应的操作。

寄存器间接寻址也以寄存器符号的形式表示。为了区别寄存器寻址和寄存器间接寻址，在寄存器间接寻址方式中，在寄存器名称前面加前缀标志“@”。

寄存器间接寻址方式的寻址范围如下。

(1) 内部基本数据 RAM 低 128B 及高 128B 的附加数据 RAM(寄存器间接寻址是访问高 128B 附加数据 RAM 的唯一寻址方式)，对这些单元的间接寻址，应使用 R0 或 R1 作间址寄存器，汇编表示形式为@Ri(i=0 或 1)。

(2) 外部扩展的数据 RAM，或 STC15 单片机片内扩展的数据 RAM。

对外部扩展的 RAM 间接寻址可以使用 DPTR(汇编表示形式为@DPTR)或 Ri 作间址寄存器，若使用 Ri 间接寻址，则访问的片外数据存储器单元低 8 位地址由 Ri 提供，高 8 位地址由 P2 口锁存器当前值给出。

【例 3.4】 设 R0=56H，DPTR=0325H。

```
MOV     A,@R0           ; 将地址为 56H 的内部 RAM 的内容送至累加器 A
MOVX    @R0,A           ; 将 A 的内容送至低 8 位地址为 56H 的片外 RAM
MOVX    @DPTR,A         ; 将地址为 0325H 的片外 RAM 的内容送至累加器 A
```

注意这种寻址方式不能用于访问特殊功能寄存器。

3.2.5 变址寻址

变址寻址是以 DPTR 或 PC 作为基址寄存器，以累加器 A 作为变址寄存器，将两寄存器的内容相加形成 16 位地址形成操作数的实际地址。变址寻址方式只能对程序存储器进行寻址，或者说它是专门针对程序存储器的寻址方式。变址寻址方式的表示方式是@A＋DPTR 和@A＋PC。

【例 3.5】 假定指令执行前(A)＝54H，(DPTR)＝3F21H，以下指令序列：

```
MOVC    A,@A + DPTR        ; 以 DPTR 内容 + A 得到操作数地址
MOVC    A,@A + PC          ; 以本指令的下一条指令的 PC 值 + A 得到操作数地址
JMP     @A + DPTR
```

第一条指令的功能是把 DPTR 和 A 的内容相加，再把所得到的程序存储器地址单元的内容送 A，变址寻址形成的操作数地址为 3F21H＋54H＝3F75H，而 3F75H 单元的内容为 7FH，故该指令执行的结果是 A 的内容为 7FH。

尽管变址寻址方式较为复杂，但变址寻址的指令却是 1B 指令。

3.2.6 位寻址

STC15 有位处理功能，可以对数据位进行操作，因此就有相应的位寻址方式。位寻址的寻址范围如下。

(1) 内部一般的数据 RAM 中的位寻址区。单元地址为 20H～2FH，共有 16 个单元 128 位，位地址是 00H～7FH。对这 128 个位的寻址使用直接位地址表示。例如：MOV C，2BH 指令功能是把位寻址区的 2BH 位状态送位累加位 C。

(2) 特殊功能寄存器的可寻址位，即那些地址可被 8 整除的 SFR 的各位。

位寻址在指令中有如下 4 种表示方法。

(1) 直接使用位地址。例如，PSW 寄存器位 5 地址为 D5H。

(2) 位名称的表示方法。专用寄存器中的一些寻址位是有符号名称的，例如，PSW 寄存器位 5 是 F0 标志位，则可使用 F0 表示该位。

(3) 专用寄存器符号加位的表示方法。例如，PSW 寄存器的位 5，表示为 PSW.5。

(4) 单元地址加位的表示方法。例如，D0H 单元(即 PSW 寄存器)位 5，表示为 D0H.5。

【例 3.6】 位寻址示例。

```
ANL     C,30H              ; 进位位 C 与 30H 位相与，结果保存在 C 中
MOV     35H,C              ; 进位位 C 送 35H 位
SETB    20H                ; 20H 位置 1
```

3.2.7 相对寻址

相对寻址方式是为了程序的相对转移而设计的，为相对转移指令所采用。在相对寻址的转移指令中，给出了地址偏移量，把 PC 的当前值加上偏移量就构成了程序转移的目的地

址,从而实现程序的转移。但这里的 PC 当前值是指执行完该转移指令后的 PC 值,即转移指令所在的 PC 值加上其指令的字节数。转移的目的地址可参见如下表达式:

目的地址=转移指令地址+转移指令字节数+偏移量

【例 3.7】 该指令的字节数为 2,设该指令地址为 2000H,则指令

```
SJMP 0AH          ; 执行后,转移到地址为 2000H + 02H + 0AH = 200CH 处执行程序
```

注意:偏移量是有正负号之分的,偏移量的取值范围是当前 PC 值的−128～+127。

需要说明的是,各种寻址方式都有其适用范围,关于这一点,会在介绍指令时做具体说明。

3.3 数据传送与交换指令

数据传送指令共有 29 条,数据传送指令一般的操作是把源操作数传送到目的操作数,指令执行完成后,源操作数不变,目的操作数等于源操作数。数据传送指令不影响标志 C、AC 和 OV(除非以 PSW 为目的的指令),但可能会对奇偶标志 P 有影响。

3.3.1 内部数据传送指令

单片机芯片内部是数据传送最频繁的部分,有关的传送类指令也最多,包括寄存器、累加器、RAM 单元以及专用寄存器之间的数据传送。

1. 以累加器 A 为目的操作数的指令

这 4 条指令的作用是把源操作数指向的内容送到累加器 A,有直接寻址、立即数寻址、寄存器寻址和寄存器间接寻址方式。

```
MOV    A,data          ; (data)→(A),直接单元地址中的内容送到累加器 A
MOV    A,#data         ; #data→(A),立即数送到累加器 A 中
MOV    A,Rn            ; (Rn)→(A),Rn 中的内容送到累加器 A 中
MOV    A,@Ri           ; ((Ri))→(A),Ri 内容指向的地址单元中的内容送到累加器 A
```

【例 3.8】 R1=45H,(45H)=0AH,执行以下指令。

```
MOV    A,#08H          ; 累加器 A←08H
MOV    A,@R1           ; 将 R1 中的内容作为地址,将相应的存储单元的内容送累加器 A,该指令
                         执行后,A = 0AH
MOV    A,45H           ; 将地址为 45H 单元的内容送累加器 A,该指令执行后,A = 0AH
MOV    A,R1            ; 累加器 A←R1,该指令执行后,A = 45H
```

2. 以寄存器 R*n* 为目的操作数的指令

这三条指令的功能是把源操作数指定的内容送到所选定的工作寄存器 R*n* 中,有直接寻址、立即数寻址和寄存器寻址方式。

```
MOV    Rn,data         ; (Rn)←(data) 直接寻址单元中的内容送到寄存器 Rn 中
MOV    Rn,#data        ; (Rn)←#data 立即数直接送到寄存器 Rn 中
MOV    Rn,A            ; (Rn)←A 累加器 A 中的内容送到寄存器 Rn 中
```

【例 3.9】 A＝08H,(45H)＝80H,执行以下指令。

```
MOV    R1,45H          ; (Rn)←(45H),执行后,R1 = 80H
MOV    R1,#45H         ; (Rn)←45H,执行后,R1 = 45H
MOV    R1,A            ; (Rn)←A,执行后,R1 = 08H
```

3. 16 位数据传送指令

这条指令的功能是把 16 位常数送入数据指针寄存器。

```
MOV    DPTR,#data16    ; (DPH)←#dataH,(DPL)←#dataL
```

16 位常数的高 8 位送到 DPH,低 8 位送到 DPL。

【例 3.10】 执行指令 MOV DPTR,#2000H 后,DPTR＝2000H。

4. 以直接地址为目的操作数的指令

指令的功能是把源操作数指定的内容送到由直接地址 data 所选定的片内 RAM 中,有直接寻址、立即寻址、寄存器寻址和寄存器间接寻址 4 种方式。

```
MOV    data,data       ; (data)←(data) 直接地址单元中的内容送到直接地址单元
MOV    data,#data      ; (data)←#data 立即数送到直接地址单元
MOV    data,A          ; (data)←A 累加器 A 中的内容送到直接地址单元
MOV    data,Rn         ; (data)←(Rn) 寄存器 Rn 中的内容送到直接地址单元
MOV    data,@Ri        ; (data)←((Ri)) 寄存器 Ri 中的内容指定的地址单元中数据送到直接
                         地址单元
```

【例 3.11】

```
MOV    80H,#0AH        ; (80H)←0AH
MOV    80H,A           ; (80H)←A
MOV    80H,R1          ; (80H)←(R1)
MOV    80H,@R0         ; (data)←(R0)
```

5. 以间接地址为目的操作数的指令

这组指令的功能是把源操作数指定的内容送到以 R*i* 中的内容为地址的片内 RAM 中,有直接寻址、立即寻址和寄存器寻址三种寻址方式。

```
MOV    @Ri,data        ; (Ri)←(data) 直接地址单元中的内容送到以 Ri 中的内容为地址的 RAM
                       ; 单元
MOV    @Ri,#data       ; (Ri)←#data 立即数送到以 Ri 中的内容为地址的 RAM 单元
MOV    @Ri,A           ; (Ri)←(A) 累加器 A 中的内容送到以 Ri 中的内容为地址的 RAM 单元
```

【例 3.12】

```
MOV    @R1,20H         ; (R1)←(20H)
MOV    @R0,#20H        ; (R0)←#20H
MOV    @R1,A           ; (R1)←(A)
```

3.3.2 外部数据存储器的传送指令

外部数据存储器只能和累加器 A 进行数据传送,而不能与内部 RAM 或 SFR 直接进行数据传送。累加器 A 与片外数据存储器 RAM 传送指令共有 4 条。

这 4 条指令的作用是累加器 A 与片外数据存储器间的数据传送。使用寄存器寻址方式：

```
MOVX  @DPTR,A     ; ((DPTR))←(A)累加器中的内容送到 DPTR 数据指针指向的片外数据存储器地
                  ; 址单元中
MOVX  A,@DPTR     ; (A)←((DPTR))数据指针指向的片外数据存储器地址中的内容送到累加器 A 中
MOVX  A,@Ri       ; (A)←((Ri)),寄存器 Ri 指向片外 RAM 地址低 8 位,片外数据存储器高 8 位地址
                  ; 由 P2 口锁存器当前值提供,将此单元的内容送到累加器 A 中
MOVX  @Ri,A       ; ((Ri))←(A)累加器中的内容送到寄存器 Ri 提供低 8 位地址,P2 口锁存器提供
                  ; 高 8 位地址的片外数据存储器中
```

【例 3.13】 将单元地址为 45H 的内部 RAM 的内容送到外部 RAM 的 0FFFH 单元中。

```
MOV   A,45H
MOV   DPTR,#0FFFH
MOVX  @DPTR,A
```

3.3.3 读程序存储器单元内容的指令

这组指令的功能是对存放于程序存储器中的数据进行读取操作，使用变址寻址方式：

```
MOVC  A,@A+DPTR ; (A)←((A))+(DPTR),将 DPTR 当前内容加上 A 累加器内容,所得和作为程序存
                ; 储器单元的地址,将此程序存储器地址单元内容送给 A
MOVC  A,@A+PC   ; 将 PC 当前值(即本指令的地址+1)加上 A 累加器内容,所得和作为程序存储器
                ; 单元的地址,将此程序存储器地址单元内容送给 A
```

【例 3.14】 有如下程序段：

```
MOV   A,#04H
MOV   DPTR,#L1    ; DPTR 内容为标号 L1 所对应的地址
MOVC  A,@A+DPTR
RET
L1:   DB  01H,04H,09H,10H,19H
```

从 L1 开始存放有一个表格，程序中 MOVC 指令将地址为(L1+4)单元的内容送到 A，因此该程序段的执行结果为：A=19H。

3.3.4 堆栈操作指令

堆栈操作指令的作用是把直接寻址单元的内容传送到堆栈指针 SP 所指的单元中，以及把 SP 所指单元的内容送到直接寻址单元中。

1. 入栈操作指令

PUSH　data；SP←SP+1，(SP)←(data)，执行的操作是：堆栈指针首先加 1，再直接将寻址单元中的数据送到堆栈指针 SP 所指的单元中。

2. 出栈操作指令

POP　data；(SP)←(data)(SP)−1→(SP)，执行的操作是：堆栈指针 SP 所指的单元数据送到直接寻址单元中，再将堆栈指针 SP 进行减 1 操作。

3. 堆栈操作指令说明

(1) 单片机开机复位后,(SP)默认为 07H,但一般都需要重新赋值,设置新的 SP 首址。入栈的第一个数据必须存放于 SP+1 所指存储单元,故实际的堆栈底为 SP+1 所指的存储单元。

(2) 堆栈操作是字节数据操作,每次压入或弹出一个字节。

(3) 堆栈的生长方向是:入栈时栈顶向地址增加的方向生长,即 SP 先加 1,再压入;弹出时按地址减少的方向进行,即先弹出,SP 再减 1。

【例 3.15】

```
MOV   A,#08H
MOV   SP,#20H
PUSH  ACC          ; 此时 SP = 21H,(21H) = 08H
POP   22H          ; 将当前堆栈顶内容弹出至 22H 单元,执行后(22H) = 08H,SP = 20H
```

3.3.5 数据交换指令

这 5 条指令的功能是把累加器 A 中的内容与源操作数所指的数据相互交换。

```
XCH   A,Rn         ; (A)←→(Rn),累加器与工作寄存器 Rn 中的内容互换
XCH   A,@Ri        ; (A)←→((Ri)),累加器与工作寄存器 Ri 所指的存储单元中的内容互换
XCH   A,data       ; (A)←→(data),累加器与直接地址单元中的内容互换
XCHD  A,@Ri        ; (A3-0)←→((Ri)3-0),累加器低 4 位与工作寄存器 Ri 所指的存储单元内容低
                   ; 4 位互换
SWAP  A            ; (A3-0)←→(A7-4),累加器中的内容高低 4 位互换
```

【例 3.16】 A=43H,R1=80H,(80H)=07H,分别执行如下指令:

```
XCH   A,R1         ; A 与 R1 中的内容互换,A = 80H,R1 = 43H
XCHD  A,@R1        ; A 与 R1 的低半字节互换,A = 47H,(80H) = 03H
SWAP  A            ; A 中的内容高低半字节互换,A = 34H
```

3.4 算术运算指令

算术运算指令共有 24 条,算术运算主要是执行加、减、乘、除四则运算。另外,STC15 指令系统中有相当一部分是进行加 1、减 1 操作,BCD 码的运算和调整,都归类为运算指令。虽然 STC15 单片机的算术逻辑单元 ALU 仅能对 8 位无符号整数进行运算,但利用进位标志 C,则可进行多字节无符号整数的运算。同时利用溢出标志,还可以对带符号数进行补码运算。需要指出的是,除加 1、减 1 指令外,这类指令大多数都会对 PSW(程序状态字)有影响,使用中应特别注意。

3.4.1 加减法指令

1. 加法指令

这 4 条指令的源操作数的寻址方式分别为立即数寻址、直接寻址、寄存器寻址及间接寻

址方式，目的操作数为累加器 A，运算结果存在 A 中。

```
ADD   A,#data     ;(A)←(A)+#data,累加器 A 中的内容与立即数#data 相加,结果存在 A 中
ADD   A,data      ;(A)←(A)+(data),累加器 A 中的内容与直接地址单元中的内容相加,结果存
                  ;在 A 中
ADD   A,Rn        ;(A)←(A)+(Rn),累加器 A 中的内容与工作寄存器 Rn 中的内容相加,结果存在
                  ;A 中
ADD   A,@Ri       ;(A)←(A)+((Ri)),累加器 A 中的内容与工作寄存器 Ri 所指向地址单元中的
                  ;内容相加,结果存在 A 中
```

使用这些指令时应注意以下几个问题。

(1) 参加运算的两个操作数必须是 8 位二进制数，操作结果也是一个 8 位二进制数，且对 PSW 中所有标志位产生影响。

(2) 用户既可以根据编程需要把参加运算的两个操作数看作无符号数(0～255)，也可以把它们看作带符号数(－128～127)。

例如，若把二进制数 11010011B 看作无符号数，则该数的十进制值为 211；若把它看作一个带符号补码数，则它的十进制值为－45。

(3) 无论用户把这两个操作数看作无符号数还是带符号数，CPU 总是按照二进制相加的规则进行相加，并产生 PSW 中标志位。各标志位产生的规则如下。

进位位 CY：相加(或相减)的最高位向前有进位(或借位)，则 CY＝1。

溢出标志 OV：相加(或相减)的最高位及次高位向前同时有进位(或借位)，则 OV＝0，反之则 OV＝1。

半进位 AC：相加(或相减)的 D3 位向前有进位(或借位)，则 AC＝1。

奇偶标志 P：总是以 A 累加器中的 1 的个数为判断对象，若 A 中 1 的个数为奇数，则 P＝1；否则 P＝0。

在采用加法指令来对两个带符号数的加减法运算时，对结果的取舍，应先检测 PSW 中 OV 标志位状态。若 OV＝0，则相加减结果无溢出，A 中的结果正确；若 OV＝1，则相加减结果有溢出，A 中结果不正确。

2. 带进位加法指令

带进位的加法运算，共有三个数参加运算，即累加器 A，不同寻址方式的加数，以及进位标志位 CY 的状态。运算结果送累加器 A。其余与加法指令 ADD 相同。

```
ADDC  A,data      ;(A)←(A)+(data)+(CY),累加器 A 中的内容与直接地址单元的内容及进位位
                  ;相加,结果存在 A 中
ADDC  A,#data     ;(A)←(A)+#data+(CY),累加器 A 中的内容与立即数及进位位相加,结果存
                  ;在 A 中
ADDC  A,Rn        ;(A)←(A)+Rn+(CY),累加器 A 中的内容与工作寄存器 Rn 中的内容及进位位
                  ;相加,结果存在 A 中
ADDC  A,@Ri       ;(A)←(A)+((Ri))+(C),累加器 A 中的内容与工作寄存器 Ri 指向地址单元中
                  ;的内容及进位位相加,结果存在 A 中
```

带进位的加法运算指令常用于多字节数的加法运算。

【例 3.17】 双字节无符号数相加，被加数放在内部 RAM 20H、21H 单元(低位在前)，加数放在内部 RAM 2AH、2BH 单元(低位在前)，和存放在 30H、31H 单元。编写程序如下。

```
MOV   A,20H        ; 被加数低字节送 A
ADD   A,2AH        ; 低位字节相加
MOV   30H,A        ; 结果送 30H 单元
MOV   A,21H        ; 取高字节被加数
ADDC  A,2BH        ; 加高位字节和低位相加产生的进位
MOV   31H,A        ; 结果送 31H 单元
```

3. 带借位减法指令

这组指令的功能是从累加器 A 中减去不同寻址方式的操作数以及进位标志 CY 状态，其差再回送累加器 A。

```
SUBB  A,data       ; (A)←(A)-(data)-(C),累加器 A 中的内容与直接地址单元中的内容、连同借
                   ; 位位相减,结果存在 A 中
SUBB  A,#data      ; (A)←(A)-#data-(C),累加器 A 中的内容与立即数、连同借位位相减,结果
                   ; 存在 A 中
SUBB  A,Rn         ; (A)←(A)-(Rn)-(C),累加器 A 中的内容与工作寄存器中的内容、连同借位位
                   ; 相减,结果存在 A 中
SUBB  A,@Ri        ; (A)←(A)-((Ri))-(C),累加器 A 中的内容与工作寄存器 Ri 指向的地址单元
                   ; 中的内容、连同借位位相减,结果存在 A 中
```

在进行减法运算中，CY＝1 表示有借位，CY＝0 则无借位。OV＝1 表明带符号数相减时，结果产生溢出，结果应抛弃。

例如：(A)＝C9H，(R2)＝54H，(CY)＝1。执行 SUBB A，R2 指令：

$$\begin{array}{r} 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1 \\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0 \\ -\qquad\qquad\qquad 1 \\ \hline 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0 \end{array}$$

运算结果为(A)＝74H，(CY)＝0，(OV)＝1，若 C9 和 54H 是两个无符号数，则结果 74H 是正确的；反之，若为两个带符号数，则由于有溢出而表明结果是错误的，因为负数减正数其差不可能是正数。

减法运算只有带借位的减法运算，而没有不带借位的减法运算，如果要进行不带借位的减法运算，只需用 CLR C 指令先把进位标志位清零即可。

【例 3.18】 利用 SUBB 指令进行两字节的减法运算。被减数放在内部 RAM 20H、21H 单元(低位在前)，减数放在内部 RAM 2AH、2BH 单元(低位在前)，差存放在 30H、31H 单元。编写程序如下。

```
MOV   A,20H        ; 被减数低字节送 A
CLR   C            ; CY 清零
SUBB  A,2AH        ; 低位字节相减
MOV   30H,A        ; 结果送 30H 单元
MOV   A,21H        ; 被减数高字节送 A
SUBB  A,2BH        ; 高位字节相减
MOV   31H,A        ; 结果送 31H 单元
```

4. 加 1 指令

如下 5 条指令的功能均为原操作数的内容加 1，结果送回原寄存器。这组指令可对累

加器、寄存器、内部 RAM 单元以及数据指针进行加 1 操作。加 1 指令不会对 CY、AC、VO 标志有影响，如果原寄存器的内容为 FFH，执行加 1 后，结果就会是 00H。

```
INC   A          ; (A) + 1←(A)
INC   data       ; (data) + 1←(data)
INC   @Ri        ; ((Ri)) + 1←((Ri))
INC   Rn         ; (Rn) + 1←(Rn)
INC   DPTR       ; (DPTR) + 1←(DPTR)
```

在 INC data 这条指令中，如果直接地址是 I/O，其功能是先读入 I/O 锁存器的内容，然后在 CPU 进行加 1 操作，再输出到 I/O 上，这就是“读—修改—写”操作。

5. 减 1 指令

这组指令可以进行累加器、寄存器以及内部 RAM 单元的减 1 操作。操作不影响程序状态字 PSW 的 CY、AC、VO 标志的状态。若原寄存器的内容为 00H，减 1 后即为 FFH。这组指令共有直接、寄存器、寄存器间址等寻址方式，当直接地址是 I/O 口锁存器时，“读—修改—写”操作与加 1 指令类似。

```
DEC   A          ; (A) - 1←(A)，累加器 A 中的内容减 1，结果送回累加器 A 中
DEC   data       ; (data) - 1←(data)，直接地址单元中的内容减 1，结果送回直接地址单元中
DEC   @Ri        ; ((Ri)) - 1←((Ri))，寄存器 Ri 指向的地址单元中的内容减 1，结果送回原地址
                 ; 单元中
DEC   Rn         ; (Rn) - 1←(Rn)，寄存器 Rn 中的内容减 1，结果送回寄存器 Rn 中
```

6. 十进制调整指令

在进行 BCD 码运算时，这条指令总是跟在 ADD 或 ADDC 指令之后，其功能是将执行加法运算后存于累加器 A 中的结果进行调整和修正，以得到正确的 BCD 码结果。

```
DA  A
```

这条指令是在进行 BCD 码运算时，跟在 ADD 和 ADDC 指令之后，将相加后存入在累加器中的结果进行修正。

修正的条件和方法如下。

若低 4 位大于 9 或(AC)＝1，则低 4 位加 6；

若高 4 位大于 9 或(CY)＝1，则高 4 位加 6。

若以上两条同时发生，或高 4 位虽等于 9 但低 4 位修正后有进位，则应加 66H 修正。

修正后相加的结果就是正确的 BCD 码了。修正是由硬件中的修正电路自动进行的，用户不必考虑何时该加“6”，使用时只需在 ADD 和 ADDC 后面紧跟一条 DA　A 指令即可。

【例 3.19】 利用十进制调整指令作十进制加法运算。设被加数放在内部 RAM 20H 单元，假设为 BCD 码 35H，即十进制数 35，加数放在内部 RAM 2AH 单元，假设为 BCD 码 49H，即十进制数 49，和存放在 30H 单元。

```
MOV   A,20H      ; 取被加数
ADD   A,2AH      ; 求和，和为 35H + 49H = 7EH
DA    A          ; 十进制调整，调整后 A = 84H，为正确的 BCD 码的和
MOV   30H,A      ; 存结果
```

3.4.2 乘法和除法指令

1. 乘法指令

这个指令的作用是把累加器 A 和寄存器 B 中的 8 位无符号数相乘，所得到的是 16 位乘积，这个结果低 8 位存在累加器 A 中，而高 8 位存在寄存器 B 中。如果 OV＝1，说明乘积大于 FFH，否则 OV＝0，但进位标志位 CY 总是等于 0。

```
MUL   AB          ; (A,B)←(A)×(B),结果 A 为高 8 位,B 为低 8 位
```

【例 3.20】 利用乘法指令编写两个数 20H×30H 的程序，将乘积的高 8 位存入 31H 单元，低 8 位存入 30H 单元。

```
MOV   A,＃20H
MOV   B,＃30H
MUL   AB
MOV   30H,A
MOV   31H,B
```

2. 除法指令

这个指令的作用是把累加器 A 的 8 位无符号整数除以寄存器 B 中的 8 位无符号整数，所得到的商存在累加器 A 中，而余数存在寄存器 B 中。除法运算总是使进位标志位 CY 等于 0。如果 OV＝1，表明寄存器 B 中的内容为 00H，那么执行结果为不确定值，表示除法有溢出，否则 OV＝0。

```
DIV   AB          ; (A)←(A) ÷(B)的商,(B)←(A)÷(B)的余数
```

【例 3.21】 (A)＝87H，(B)＝0CH，执行指令 DIV AB，则结果如下所示。

```
(A) = 0BH  (B) = 03H  OV = 0  CY = 0
```

3.5 逻辑运算及移位指令

3.5.1 逻辑运算指令

逻辑运算常用于对数据位进行加工。逻辑运算是按位进行的，逻辑运算指令包括与、或、取反、异或等运算。与、或、异或的运算法则如下。

与：两个二进制位相与，其中有 0 则结果为 0，否则结果为 1。

或：两个二进制位相或，其中有 1 则结果为 1，否则结果为 0。

异或：两个二进制位相异或，它们相同则结果为 0，不同则结果为 1。

1. 求反指令

这条指令将累加器中的内容按位取反。

```
CPL   A           ; 累加器中的内容按位取反
```

2. 清零指令

这条指令将累加器中的内容清零。

```
CLR   A              ; 0←(A),累加器中的内容清零
```

3. 逻辑与操作指令

这组指令的作用是将两个单元中的内容执行逻辑与操作。

```
ANL   A,data         ; 累加器 A 中的内容和直接地址单元中的内容执行与逻辑操作.结果存在 A 中
ANL   data,#data     ; 直接地址单元中的内容和立即数执行与逻辑操作.结果存在直接地址单元中
ANL   A,#data        ; 累加器 A 的内容和立即数执行与逻辑操作.结果存在累加器 A 中
ANL   A,Rn           ; 累加器 A 的内容和寄存器 Rn 中的内容执行与逻辑操作.结果存在累加器 A 中
ANL   data,A         ; 直接地址单元中的内容和累加器 A 的内容执行与逻辑操作.结果存在直接地址
                     ; 单元中
ANL   A,@Ri          ; 累加器 A 的内容和工作寄存器 Ri 指向的地址单元中的内容执行与逻辑操作.
                     ; 结果存在累加器 A 中
```

4. 逻辑或操作指令

这组指令的作用是将两个单元中的内容执行逻辑或操作。

```
ORL   A,data         ; 累加器 A 中的内容和直接地址单元中的内容执行逻辑或操作.结果存在寄存器
                     ; A 中
ORL   data,#data     ; 直接地址单元中的内容和立即数执行逻辑或操作.结果存在直接地址单元中
ORL   A,#data        ; 累加器 A 的内容和立即数执行逻辑或操作.结果存在累加器 A 中
ORL   A,Rn           ; 累加器 A 的内容和寄存器 Rn 中的内容执行逻辑或操作.结果存在累加器 A 中
ORL   data,A         ; 直接地址单元中的内容和累加器 A 的内容执行逻辑或操作.结果存在直接地址
                     ; 单元中
ORL   A,@Ri          ; 累加器 A 的内容和工作寄存器 Ri 指向的地址单元中的内容执行逻辑或操作。
                     ; 结果存在累加器 A 中
```

5. 逻辑异或操作指令

这组指令的作用是将两个单元中的内容执行逻辑异或操作。

```
XRL   A,data         ; 累加器 A 中的内容和直接地址单元中的内容执行逻辑异或操作.结果存在寄存
                     ; 器 A 中
XRL   data,#data     ; 直接地址单元中的内容和立即数执行逻辑异或操作.结果存在直接地址单元中
XRL   A,#data        ; 累加器 A 的内容和立即数执行逻辑异或操作.结果存在累加器 A 中
XRL   A,Rn           ; 累加器 A 的内容和寄存器 Rn 中的内容执行逻辑异或操作.结果存在累加器
                     ; A 中
XRL   data,A         ; 直接地址单元中的内容和累加器 A 的内容执行逻辑异或操作.结果存在直接地
                     ; 址单元中
XRL   A,@Ri          ; 累加器 A 的内容和工作寄存器 Ri 指向的地址单元中的内容执行逻辑异或操
                     ; 作.结果存在累加器 A 中
```

【例 3.22】 设 A 中的内容为 23H,分别执行下列程序。

```
ANL   A,#0FH         ; A = 03H
ORL   A,#0FH         ; A = 2FH
XRL   A,#0FH         ; A = 2CH
```

3.5.2 移位指令

这 4 条指令的作用是将累加器中的内容循环左移或右移一位，后两条指令是连同进位位 CY 一起移位。循环移位指令操作示意图如图 3.1 所示。

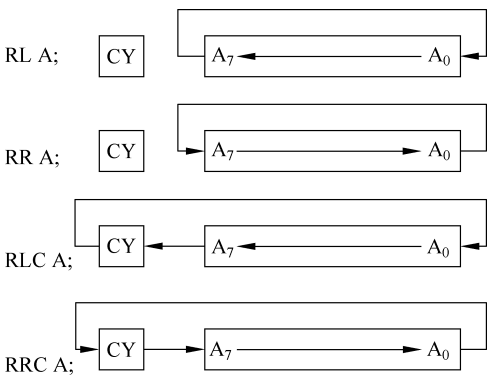

图 3.1 循环移位指令操作示意

【例 3.23】 设 A 中的内容为 04H，编程如下。

```
RL    A          ; A = 08H
RL    A          ; A = 10H
RR    A          ; A = 08H
```

从例 3.23 可以看出，可以利用左移和右移指令实现乘以 2 或除以 2 的运算。

3.6 控制转移指令

控制转移指令用于控制程序的流向，所控制的范围即为程序存储器区间，STC15 系列单片机的控制转移指令主要有以下几种类型。

(1) 无条件转移：无须判断，执行该指令就转移到目的地址。

(2) 条件转移：需判断是否满足条件，若条件满足，则转移到目的地址，否则顺序执行。

(3) 子程序调用。

(4) 子程序返回。

这些指令的执行一般都不会对标志位有影响。

3.6.1 无条件转移指令

这组指令执行完后，程序就会无条件转移到指令所指向的地址。

1. 长转移指令

指令格式：

```
LJMP  addr16    ; (PC)←addr16
```

该指令提供 16 位目标地址，将指令中第二字节和第三字节地址码分别装入 PC 的高 8 位和低 8 位中，无条件转移的目标地址范围是整个存储器的 64KB 空间。

2. 绝对转移指令

指令格式：

```
AJMP  addr11    ; (PC)←(PC) + 2,(PC10 - 0)←addr11
```

本指令是双字节指令，该指令提供目标地址的低 11 位，转移目标的高 5 位地址保持当前的 PC 值(即本指令的下一条指令地址)的高 5 位不变，所以，绝对转移的目标地址范围是从下一条指令开始的 2KB 空间。

AJMP 指令的功能是构造程序转移的目的地址，实现程序转移。其构造方法是：以指令提供的 11 位地址去替换 PC 的低 11 位内容，形成新的 PC 值，此即转移的目的地址。但要注意，被替换的 PC 值是本条指令地址加 1 以后的 PC 值，即指向下一条指令的 PC 值。

例如，程序中 2000 地址单元有绝对转移指令：

```
2000H   AJMP  275H
```

11 位绝对转移地址为 010 0111 0101B(275H)，其指令代码如图 3.2 所示。

图 3.2　指令 AJMP 257H 的指令代码

程序计数器 PC 加 2 后的内容为 0010 0010 0000 0010B(2002H)，以 11 位绝对转移地址替换 PC 的低 11 位内容，最后形成的目的地址为 0010 0010 0111 0101B(2275H)。

3. 相对短转移指令

指令格式：

```
SJMP  rel    ; (PC)←(PC) + 2 + rel
```

该指令控制程序无条件转向指定地址。该指令的 rel 是一个 8 位带符号的相对偏移量，范围为－128～＋127。负数表示向后转移，正数表示向前转移。

【例 3.24】 2354H 地址上有 SJMP 指令：

```
2354H   SJMP  24H
```

源地址为 2354H，rel＝24H 是正数，因此程序向前转移。目的地址＝2354H＋02H＋24H＝237AH。即执行完本指令后，程序转到 237AH 地址去执行。

若该指令改为：

```
2354H  SJMP  0F8H
```

F8H 是负数－08H 的补码，因此程序向后转移，目的地址＝2354H＋02H－8H＝234EH。即执行完本指令后，程序向后转到 234EH 地址去执行。

4. 间接转移指令

指令格式：

```
JMP   @A + DPTR     ; (PC)←(A) + (DPTR)
```

该指令把累加器 A 中的无符号数与作为基址寄存器 DPTR 中的 16 位数据相加，所得的值送入 PC 作为转移的目的地址。该指令执行后既不影响累加器 A 和数据指针 DPTR 中的内容，也不影响任何标志位。该指令的转移地址不是编程时确定的，而是在程序运行时动态决定的。因此，可以在 DPTR 中装入多条转移程序的首地址，而由累加器 A 中的内容来动态选择该时刻应转向哪一条分支，即可实现程序的多分支转移。在第 4 章将对此做详细说明。

3.6.2 条件转移指令

条件转移就是程序转移是有条件的。执行条件转移指令时，如指令中规定的条件为真，则进行程序转移，否则程序顺序执行。

1. 以累加器是否为 0 作条件的转移指令

```
JZ    rel           ; (A) = 0,则(PC)←(PC) + 2 + rel
JNZ   rel           ; (A)≠0,则(PC)←(PC) + 2 + rel
```

本指令不是以 PSW 标志位为条件判断的，而是以 A 累加器的值是否为零判断的。两条都是相对转移指令，以 rel 为偏移量，rel 是一个 8 位带符号的相对偏移量，范围为－128～＋127。负数表示向后转移，正数表示向前转移。

2. 比较转移指令

比较转移指令通过比较两个操作数的大小，将两个操作数相减，差会丢弃。如果它们的值不相等则转移，相等则继续顺序执行下面的指令。

```
CJNE  A,data,rel          ; (A)≠(data),则(PC)←(PC) + 3 + rel
CJNE  A,#data,rel         ; (A)≠#data,则(PC)←(PC) + 3 + rel
CJNE  Rn,#data,rel        ; (A)≠#data,则(PC)←(PC) + 3 + rel
CJNE  @Ri,#data,rel       ; (A)≠#data,则(PC)←(PC) + 3 + rel
```

指令执行后要影响进位标志 CY，若操作数 1 小于操作数 2，则 CY＝1，若操作数 1 大于操作数 2，则 CY＝0。rel 为偏移量。

3. 减 1 条件转移指令

```
DJNZ  Rn,rel              ; (Rn) - 1←(Rn),(Rn)≠0,则(PC) ← (PC) + 2 + rel
DJNZ  data,rel            ; (data) - 1←(data),(data)≠0,则(PC)←(PC) + 2 + rel
```

这两条指令执行时，先将 Rn 或者 data 单元的内容减 1，然后判断结果是否为 0，若不为 0，则转移，若为 0 则继续顺序执行。这两条指令主要用于控制程序循环。

3.6.3 子程序调用和返回指令

为了便于程序编写，节约存储空间，可将一些需要反复执行的程序段编写成子程序，当

需要用它们时，就用一个调用命令使程序按调用的地址去执行，这就需要子程序的调用指令和返回指令。

1. 子程序调用指令

(1) 长调用指令。指令格式如下：

```
LCALL  addr16
```

该指令为三字节指令，调用地址在指令中直接给出，可在 64KB 空间调用子程序。指令操作为：

① (PC)←(PC) + 3
② (SP)←(SP) + 1,(SP)←($PC_{7\sim0}$)
③ (SP)←(SP) + 1,(SP)←($PC_{15\sim8}$)
④ (PC)←addr16

指令执行时，先修改堆栈指针，再将本指令的下一条指令地址存入 SP 指向的堆栈单元，然后将子程序首址(指令中的目标地址 addr16)赋给 PC，实现子程序的调用。

(2) 绝对调用指令。指令格式如下：

```
ACALL  addr11
```

该指令提供 11 位目标地址，除了将指令中的低 11 位地址送给 PC，PC 高 5 位为当前值不变以外，其他同 LCALL 指令。指令执行的操作为：

① (PC)←(PC) + 2
② (SP)←(SP) + 1,(SP)←($PC_{7\sim0}$)
③ (SP)←(SP) + 1,(SP)←($PC_{15\sim8}$)
④ (PC10～0)←addr11

2. 返回指令

返回指令共有两条：

```
RET   ; (SP)←(PC15~8),(SP)←(SP) - 1
      ; (SP)←(PC7~0),(SP)←(SP) - 1
RETI  ; 中断返回指令
```

这两条指令执行时，首先将当前堆栈顶保存的返回地址赋给 PC，再将堆栈指针 SP 减 2，从而实现返回功能。RETI 除具有 RET 功能外，还具有恢复中断逻辑的功能，所以，RETI 指令和 RET 指令决不能互换使用。

3.6.4 空操作指令

空操作指令的指令格式为：

```
NOP   ; (PC) + 1←(PC)
```

这条指令除了使 PC 加 1，消耗一个机器周期外，没有执行任何操作。该指令常用于程序的等待或时间的延时。

【例 3.25】 软件延时 1s 子程序，设系统主时钟频率为 6MHz，采用 12T 系统(即系统

时钟为主时钟的 12 分频，也就是说每个机器周期为 2μs）。

```
DELY:   MOV  22H,#05H          ; 2Tm,即所需时间为两个机器周期。下同
L3:     MOV  23H,#064H         ; 2Tm
L2:     MOV  24H,#0C7H         ; 2Tm
L1:     NOP                    ; 1Tm
        NOP                    ; 1Tm
        NOP                    ; 1Tm
        DJNZ 24H,L1            ; 2Tm
        DJNZ 23H,L2            ; 2Tm
        DJNZ 22H,L3            ; 2Tm
        RET
```

请自行验算，执行完这个三重循环，需要多少个机器周期？

3.7 位操作指令

STC51 系列单片机的位操作指令共有 17 条，分为位传送、位置位和位清零、位运算以及位控制转移指令等 4 类。

3.7.1 位传送指令

位传送指令就是可寻址位与累加位 CY 之间的传送，指令有两条。

```
MOV    C,bit          ; (C)←bit
MOV    bit,C          ; bit←(C)
```

指令中的 C 就是 CY。两个可寻址位之间没有直接的传送指令，可使用这两条指令以 CY 作中介来实现这种传送。

【例 3.26】 将 20H 位的内容传送到 50H 位。

```
MOV    10H,C          ; 暂存 CY 的内容
MOV    C,20H          ; 20H 位送 CY
MOV    50H,C          ; CY 送 50H 位
MOV    C,10H          ; 恢复 CY 内容
```

3.7.2 位置位复位指令

这组指令对 CY 及可寻址位进行置位或复位操作，共有 4 条指令。

```
CLR    C              ; CY←0
CLR    bit            ; bit←0
SETB   C              ; CY←1
SETB   bit            ; bit←1
```

3.7.3 位运算指令

位运算都是逻辑运算，有与、或、非三种指令，共 6 条。

```
ANL    C,bit        ; (C)←(C)∧(bit),与运算
ANL    C,/bit       ; (C)←(C)∧(bit 取反),与运算
ORL    C,bit        ; (C)←(C)∨(bit),或运算
ORL    C,/bit       ; (C)←(C)∧(bit 取反),或运算
CPL    C            ; (C)←(C 取反),取反的运算
CPL    bit          ; bit←bit 取反,取反的运算
```

上述指令中，/bit 是对位 bit 取反，再去作相关运算，注意 bit 位本身内容不变。位运算指令中没有位的异或运算，如需要时可由多条上述位操作指令实现。

【例 3.27】 若 E、F、D 代表位地址，进行 E、F 内容的异或操作，结果送 D，可用以下程序段实现。

```
MOV    C,F
ANL    C,/E         ; (C)←Ē∧F
MOV    D,C
MOV    C,E
ANL    C,/F         ; (C)←E∧ /F
ORL    C,D
MOV    D,C          ; 结果送 D 位
```

通过逻辑运算，可以对各种组合逻辑电路进行模拟，即用软件方法来获得组合逻辑电路的逻辑功能。

3.7.4 位控制转移指令

位控制转移指令是以位的状态作为实现程序转移的判断条件。这类指令都是相对寻址，即指令中提供一个转移的偏移量 rel。

1. 以 C 状态为条件的转移指令

```
JC     rel          ; (CY) = 1 转移,(PC)←PC + 2 + rel,否则程序顺序往下执行,(PC) + 2←PC
JNC    rel          ; (CY) = 0 转移,(PC)←PC + 2 + rel,否则程序顺序往下执行,(PC) + 2←PC
```

2. 以位状态为条件的转移指令

```
JB     bit,rel      ; 位 bit 状态为 1 转移
JNB    bit,rel      ; 位 bit 状态为 0 转移
JBC    bit,rel      ; 位 bit 状态为 1 转移,并使该位清零
```

这三条指令都是三字节指令，如果条件满足，(PC)＋3＋rel←PC，否则程序顺序往下执行，(PC)＋3←PC。

小结

本章着重介绍 51 系列单片机的指令系统。从指令的基本格式、操作数的寻址方式、功能以及应用等方面详细介绍了传送类、算术运算类、逻辑运算与移位类、控制转移类及位操作等 5 类指令。这些指令是学习程序设计的基础，读者应正确理解和掌握它们，在学习的过程中可通过多写实例掌握指令的格式、功能及寻址方式。

习题

1. 简述 STC15 指令的格式。

2. STC15 单片机有哪几种指令寻址方式？这几种寻址方式分别适用于什么地址空间？

3. 指出下列指令中源操作数和目的操作数的寻址方式。

```
ADD     A,＃34H
MOV     B,40H
PUSH    ACC
MOV     @R0,PSW
MOVX    @DPTR,A
MOVC    A,@A + DPTR
```

4. 判断下列各条指令的书写格式是否有错，如有错说明原因。

```
LJMP    ＃1000H
CLR     A
MUL     R0,R1
MOV     DPTR,1050H
MOV     A,@R7
ADD     40H,42H
MOV     R1,C
JMP     @R0 + DPTR
MOV     A,＃3000H
MOVC    @A + DPTR,A
MOVX    A,@A + DPTR
```

5. 试写出完成以下每种操作的指令序列。

(1) 将 R0 的内容传送到 R1；

(2) 内部 RAM 单元 60H 的内容传送到寄存器 R2；

(3) 外部 RAM 单元 1000H 的内容传送到内部 RAM 单元 60H；

(4) 外部 RAM 单元 1000H 的内容传送到寄存器 R2；

(5) 外部 RAM 单元 1000H 的内容传送到外部 RAM 单元 2000H。

6. 假定(SP)＝40H，(39H)＝30H，(40H)＝60H。执行下列指令：

```
POP DPH
POP DPL
```

后，DPTR 的内容为_____，SP 的内容是_____。

7. 若(R1)＝30H，(A)＝40H，(30H)＝60H，(40H)＝08H。试分析执行下列程序段后上述各单元内容的变化。

```
MOV    A,@R1
MOV    @R1,40H
MOV    40H,A
MOV    R1,#7FH 10
```

8. 下列程序段执行后，(R0)＝_____，(7EH)＝_____，(7FH)＝_____。

```
MOV    R0,#7FH
MOV    7EH,#0
MOV    7FH,#40H
DEC    @R0
DEC    R0
DEC    @R0
```

9. 下列程序段执行后，(A)＝_____，(B)＝_____。

```
MOV    A,#0FBH
MOV    B,#12H
DIV    AB
```

10. 执行下列程序段中第一条指令后，(P1.7)＝_____，(P1.3)＝_____，(P1.2)＝_____；

执行第二条指令后，(P1.5)＝_____，(P1.4)＝_____，(P1.3)＝_____。

```
ANL    P1,#73H
ORL    P1,#38H
```

11. 请使用位操作指令，实现下列逻辑操作。

```
P1.5 = ACC.2 ∧ P2.7 ∨ ACC.1 ∧ P2.0
```

12. 试用位操作指令实现下列逻辑操作。要求不得改变未涉及的位的内容。

(1) 使 ACC.0 置位；

(2) 清除累加器高 4 位；

(3) 清除 ACC.3，ACC.4，ACC.5，ACC.6。

13. 试编写程序，完成两个 16 位数的减法：7F4DH－2B4EH，结果存入内部 RAM 的 30H 和 31H 单元，31H 单元存差的高 8 位，30H 单元存差的低 8 位。

14. 试编写程序，将内部 RAM 的 20H、21H 单元的两个无符号数相乘，结果存放在 R2、R3 中，R2 中存放高 8 位，R3 中存放低 8 位。

15. 用三种方法实现将 A 累加器中的无符号数乘以 8，乘积存放在 B 和 A 寄存器中。