单片机原理、接口及应用—嵌入式系统技术基础

第2章 51单片机的指令系统

内 容 提 要

★寻址方式

★数据传送与交换指令

★算术运算、逻辑运算指令

★控制转移指令

★位操作指令

计算机通过执行程序完成人们指定的任务，程序由一条一条指令构成，能为CPU识别并执行的指令的集合就是该CPU的指令系统。

MCS-51单片机汇编语言指令格式：

操作符　目的操作数，源操作数

操作符指明该指令完成什么操作；
操作数是指明该指令的操作对象。
目的操作数是存放结果的。
指令中操作数提供的方式称为寻址方式。

指令中的常用符号

Rn: n=(0～7），表示当前工作寄存器R0～R7中的一个

Ri: i=(0、1），代表R0和R1寄存器中的一个，用作间接寻址寄存器

dir : 8 位直接字节地址（片内 RAM 和 SFR ）

#data: 8位立即数，即8位常数。可以为2进制(B)、10进制、 16进制(H)、字符（ ‘ ’ ）

#data16: 表示16位立即数，即16位常数，取值范围为#0000H～#0FFFFH

addr16 : 表示16位地址

addr11 : 表示11位地址

rel : 相对偏移量（为一字节补码）用于相对转移指令中

bit :位地址，在位地址空间中。

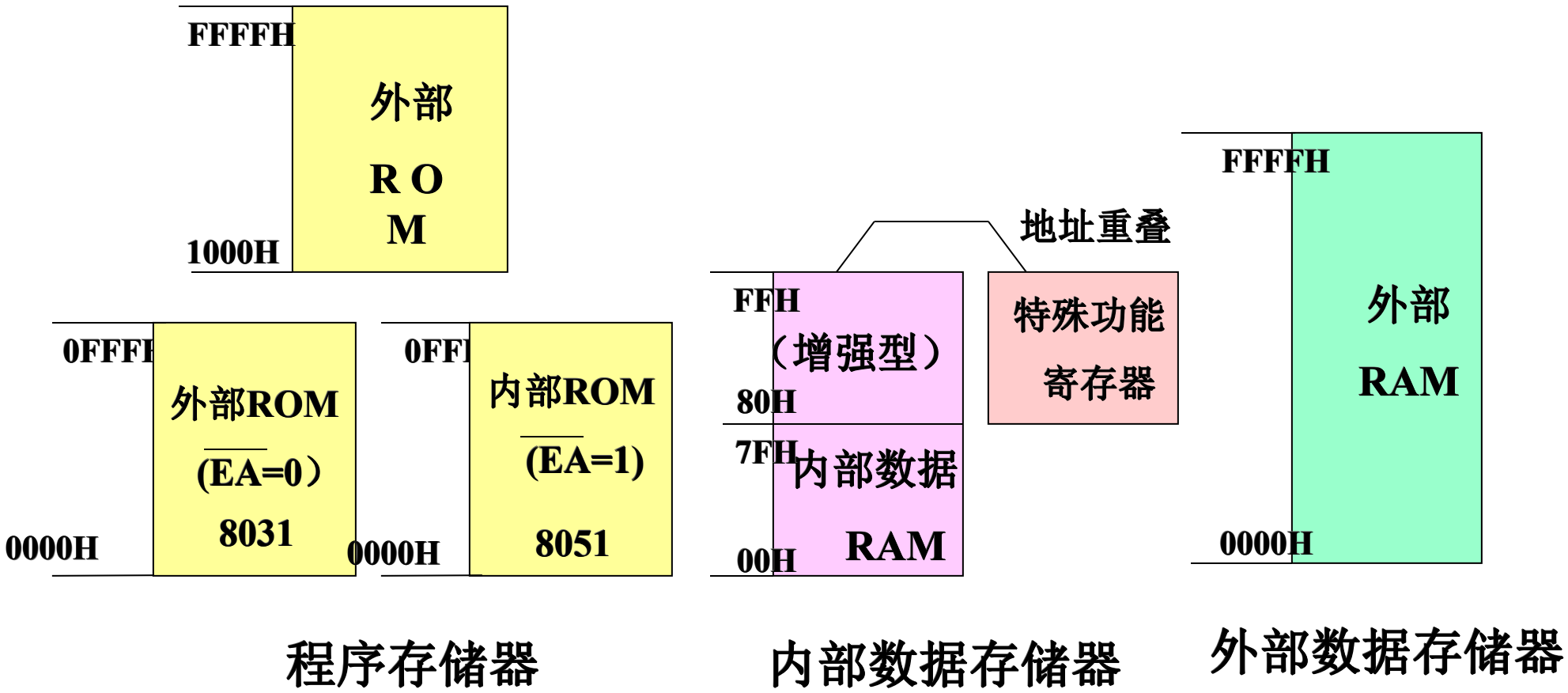
$: 表示当前指令的地址。

2.1 寻址方式

1、立即寻址

指令中直接给出操作数的寻址方式。在51系列单片机的指令系统中，立即数用一个前面加“#“号的8位数(#data，如#30H)或16位数(#data16，如#2052H)表示。立即寻址中的数，称为立即数。

例如指令：MOV A，#30H



程序存储器　　内部数据存储器　　外部数据存储器

2、直接寻址

操作数的地址直接出现在指令中。

寻址对象：

①内部数据存贮器：使用它的地址。

②特殊功能寄存器：既可使用它的地址，也可以 直接使用寄存器名。

例：MOV A，40H ；A=56H

机器码 E540

41H	78H
40H	56H

MOV 40H，41H ；机器码为854140

内部RAM （41H）→（40H）

（40H）=（41H）=78H

41H	78H
40H	78H

MOV P0，#45H ； 45H→P0，

P0为直接寻址的 SFR，其地址为 80H，

机器码为758045

3、寄存器寻址

操作数存放在寄存器中。

寻址对象：A，B，DPTR，R0～R7 。

B 仅在乘除法指令中为寄存器寻址，在其他指令中为直接寻址。

A 可以寄存器寻址又可以直接寻址，直接寻址时写作ACC

例如：MOV A，R0 ；R0→A，A、R0均为寄存器寻址，机器码E8

MUL AB ；A*B→BA，A、B为寄存器寻址，机器码A4

MOV B，R0 ；R0→B，R0为寄存器寻址，B为直接寻址

机器码 88F0，其中 F0为B的字节地址（见表1-2）

PUSH ACC ；A的内容压入堆栈

机器码C0E0

4、寄存器间址

操作数存放在以寄存器内容为地址的单元中。

例如：

```
MOV  R0,#20H
MOV  @R0，A      ；A→(20H) 地址的内部RAM
MOVX  A，@R1     ；外部RAM（地址为P2 R1 ）
                    的 内容→A
MOVX  @DPTR，A   ；A→以DPTR内容为地址的
                    外部RAM
```

5、变址寻址

以DPTR或PC寄存器内容为基地址，和A的内容为相加形成操作数的地址。其中累加器A内容是可变的。

例如： MOVC A, @A+DPTR

6、相对寻址

相对寻址是将程序计数器PC的当前值与指令第二字节给出的偏移量相加，从而形成转移的目标地址。

例如：JZ 61H

7、位寻址

对片内RAM中20H～2FH中的128个位地址及SFR中的可位寻址的位地址寻址。

例如： MOV C，20H；20H位的内容送CY标志位，C称为位累加器。

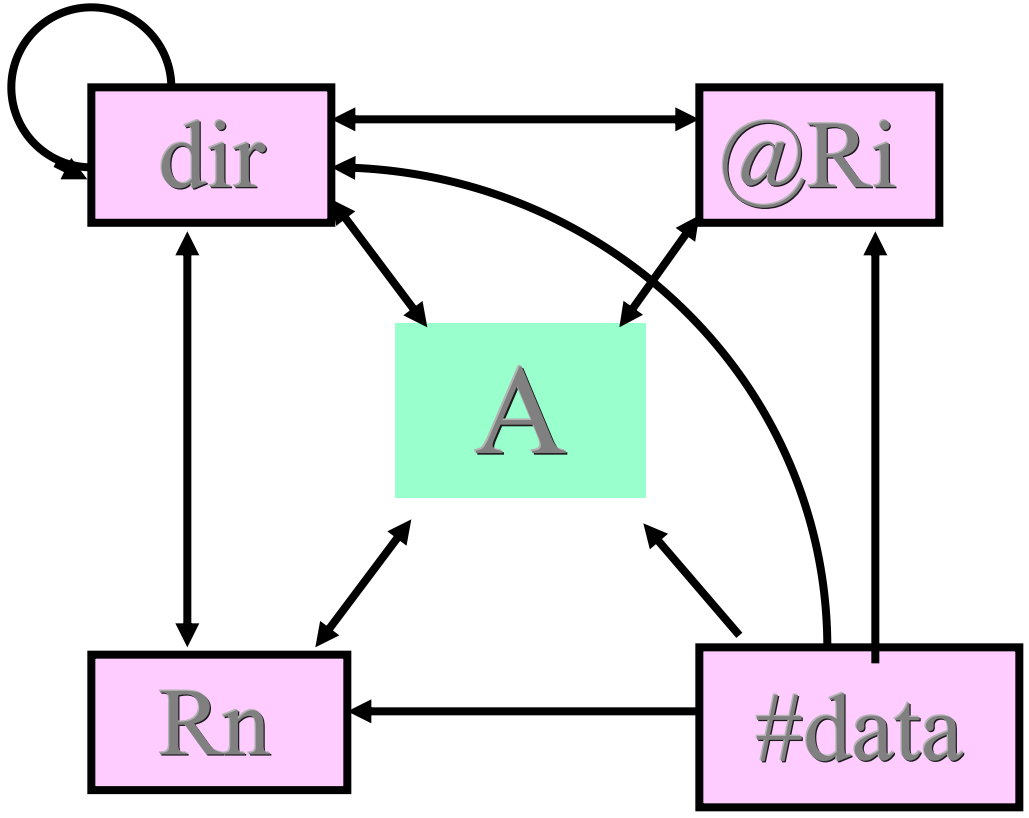
MOV A，20H；字节寻址，将内部RAM中20H单元中的内容送给累加器A。

以上两条指令均为寻址，究竟是位寻址还是字节寻址，根据两操作数类型一致的原则，由另一个操作数决定。

2.2 数据传送与交换指令

2.2.1 传送类指令

1、内部RAM、SFR之间的传送MOV指令



MOV A , { Rn, #data, dir, @Ri }

MOV dir , { A, Rn, @Ri, #data, dir }

MOV Rn , { A, dir, #data }

MOV @Ri, { A, dir, #data }

例 R1=20H，（20H）=55H，

指令MOV A，@R1执行后，A=55H。

例 （40H）=30H，指令 MOV R7，40H 执行后，R7=30H。

例 MOV R7，＃40H 执行后，R7=40H。

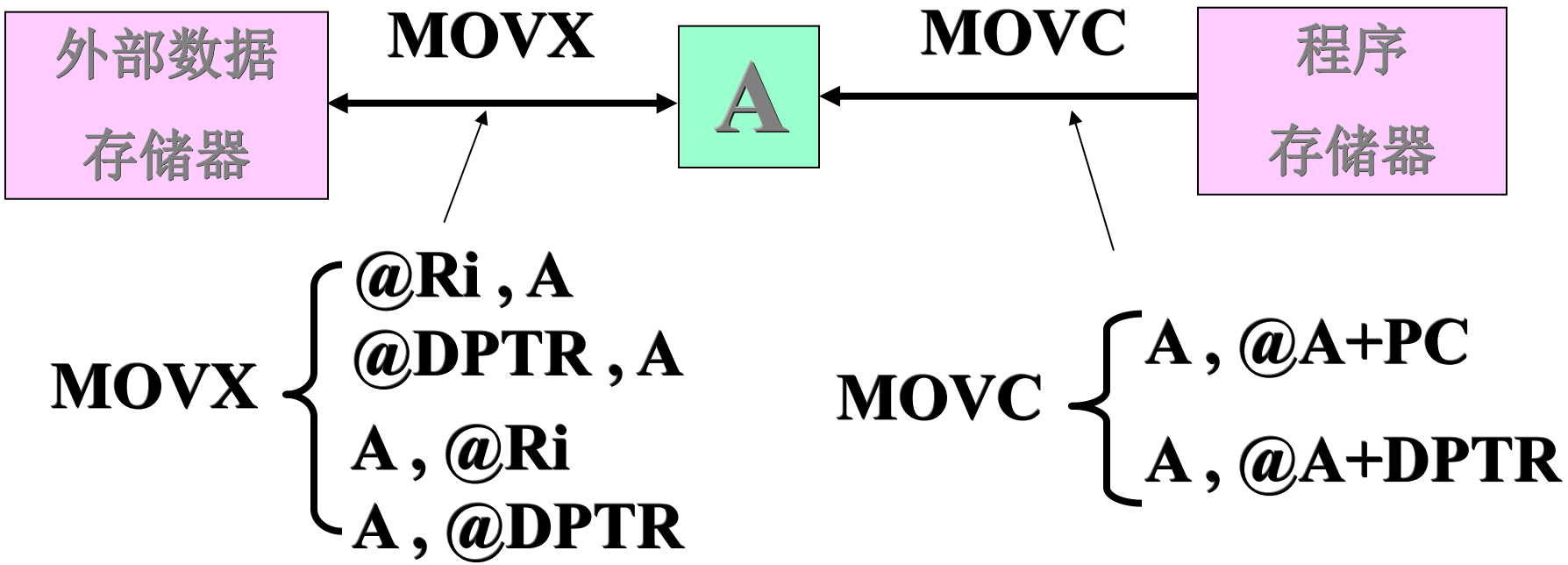
- 例 判断下列指令的正误：

MOV 29H,R7	√	MOV 25H,P1	√
MOV 56H,#70H	√	MOV 34H,28H	√
MOV R3,R7	×	MOV @R3,R7	×
MOV R3,#D2H	√	MOV #34H,28H	×
MOV A,#280H	×	MOV P3,P1	√

- 编程将R3的内容送R1。

```
MOV A，R3
MOV R1，A
```

2、外部存储器和A累加器之间的传送



MOVX指令举例

例: 实现片外数据存储器数据传送 (2000H)→(2100H)。

```
MOV   DPTR，#2000H        ；DPTR= 2000H
MOVX  A，@DPTR            ；A← X
MOV   DPTR，#2100H        ；DPTR= 2100H
MOVX  @DPTR，A            ；(2100H)← X
```

	片外RAM	
	地址	内容
DPTR→	2000H	X
	…	
	2100H	

	片外 RAM	
	地址	内容
	2000H	X
	…	
DPTR→	2100H	X

片外数据存储器不能直接寻址。下列为非法指令：

MOVX A，2000H ×

MOVX 2100H，2000H ×

使用P2口和8位寄存器Ri间址：

```
MOV  P2，#20H     ；高位地址
MOV  R0，#00      ；低位地址
MOVX  A，@R0      ；读片外RAM
MOV  P2，#21H     ；改变高位地址
MOVX  @R0，A      ；写片外RAM
```

查表指令MOVC

注：①只能从程序存储器读取数据到A累加器。

②只能使用变址间接寻址方式

多用于查常数表程序，直接求取常数表中的函数值

1）DPTR为基址寄存器

MOVC A，@A+DPTR ； A←(A+DPTR)

查表范围为 64KB 程序存储器任意空间。

2）PC为基址寄存器

MOVC A，@A+PC；A← (A+PC)

常数表只能在查表指令后 256B 范围内。

例 查表法求 $Y=X^2$。设 $X(0\leq X\leq 15)$在片内 RAM 20H单元，要求将查表求 Y，存入片内 RAM 21H单元

1）用DPTR作基址寄存器

```
        ORG  0100H
SQU:    MOV  DPTR，#TAB        ；确定表首地址
        MOV  A，20H            ；取 X
        MOVC A，@A+DPTR ；查表求 Y=X²
        MOV  21H，A            ；保存Y
        RET                    ；子程序结束
          …                    ；其它程序段
        ORG  0200H             ；常数表格首地址
TAB:    DB  00，01，04，09，…，225 ；平方表
```

2）用PC作基址寄存器

```
指令地址              源程序
                      ORG 0100H    ；程序起始地址
0100H          SQU: MOV A，20H；取X
0102H               ADD A，#3     ；修正偏移量
0104H               MOVC A，@A+PC ；查表求Y=X²
0105H               MOV 21H，A；存结果
0107H               RET                    ；子程序结束
0108H          TAB: DB  00，01，04         ；平方表
010BH                DB  09，…，225
```

思考题 当0≤X≤255时，如何用查表法编程求$Y=X^2$

3.堆栈操作指令

入栈指令：PUSH dir ；SP←SP+1，(SP) ←(dir)

出栈指令：POP dir ；(dir)←(SP)，SP←SP-1

例：设 A=02，B=56H，执行下列指令序列后，SP＝？A＝？，B＝？

```
SBR：MOV  SP，#30H        ；设栈底
     PUSH  A              ；保护现场
     PUSH  B

     MOV   A，#0
     MOV   B，#01
           …
     POP    B
     POP   A              ；恢复现场
     RET
```

堆栈操作示意：

片内 RAM	
34H	
33H	
32H	
31H	
30H	

2.2.2 交换指令

实现片内RAM区的数据双向传送

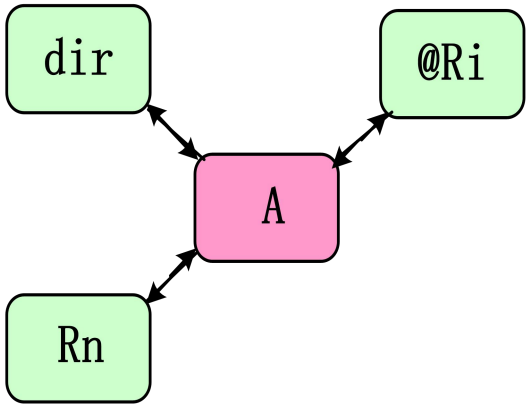


图2.8 XCH交换指令示意图

1. 字节交换指令

XCH A，Rn ；A←→Rn

XCH A，@Ri；A←→(Ri)

XCH A，DIR ；A←→DIR

例 设A= 29H，（2AH）=38H 执行指令

XCH A，2AH 后，

A= ？ ，(2AH)= ？

38H 29H

习题 将片内RAM 60H单元与 61H单元的数据交换

XCH 60H，61H ←对吗？

2. 半字节交换指令

XCHD A，@Ri；$A_{0\sim3} \rightleftharpoons (Ri)_{0\sim3}$

SWAP A ；$A_{4\sim7} \rightleftharpoons A_{0\sim3}$

A	×	◎
A	◎	×

片内 RAM	
地址	内容
2BH	35H
2AH	38H
…	
20H	85H

例3-4-3：将片内 RAM 2AH和2BH单元中的 ASCII码转换成压缩式 BCD码存入 20H单元

数字 0～9的ASCII码30H～39H

压缩式BCD码
千位 百位
十位 个位

压缩的BCD码和非压缩的BCD码见图

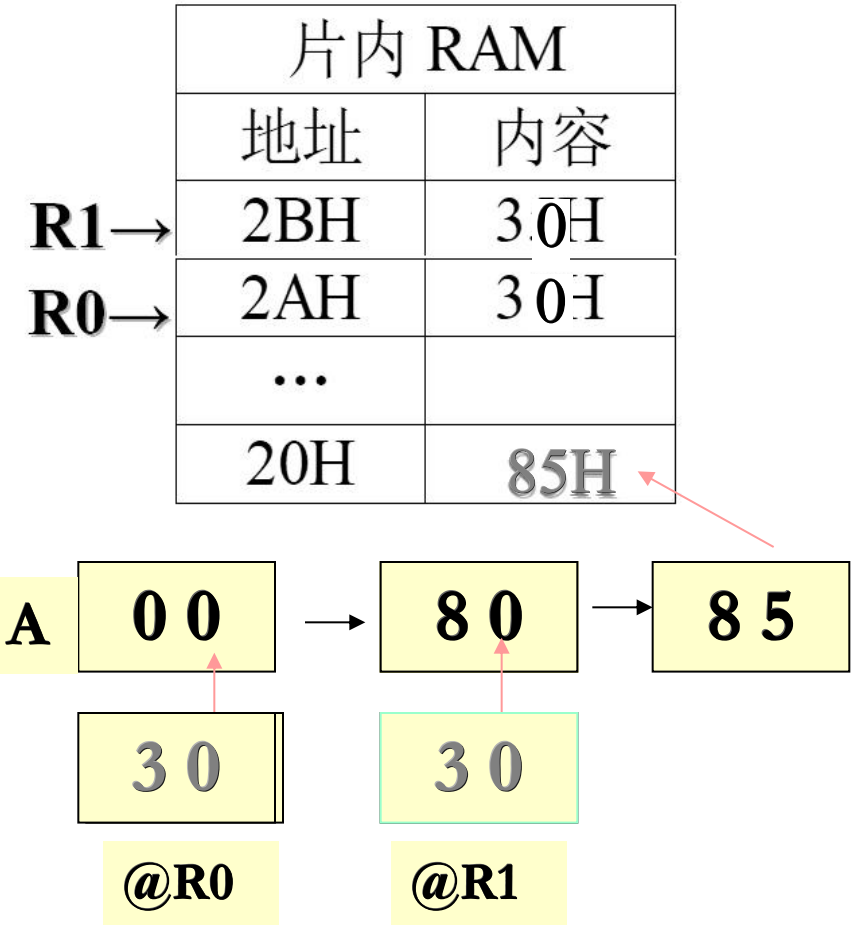
如 1823压缩的BCD码为1823H

非压缩的BCD码为01080203H

非压缩BCD
0000 千位
0000 百位
0000 十位
0000 个位

例 将片内RAM 2AH和 2BH单元中的ASCII码转换成压缩式BCD码存入20H单元

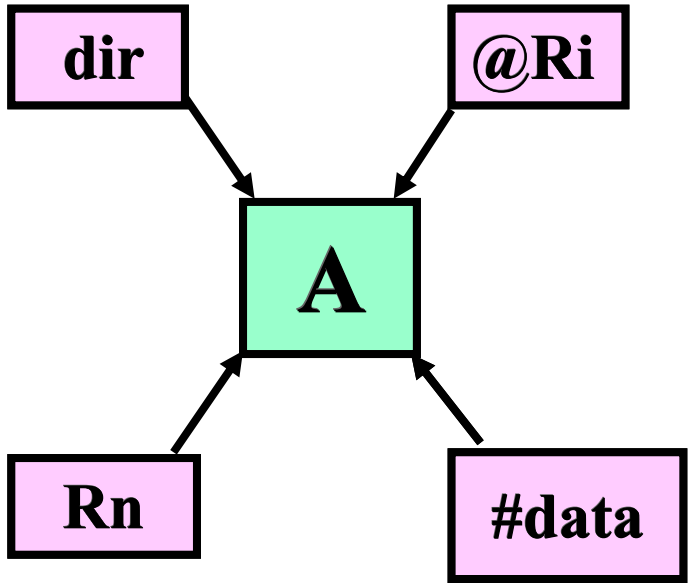
```
MOV A,  #00H  ；A=00
MOV R0,  #2AH
MOV R1,  #2BH
XCHD A,  @R0  ；低4位交换A=08
SWAP A        ；A=80H
XCHD A,  @R1  ；低4位交换
XCH   A,  20H  ；(20H)=85H
```



习题：交换片内RAM 40H单元和 41H单元的低半字节

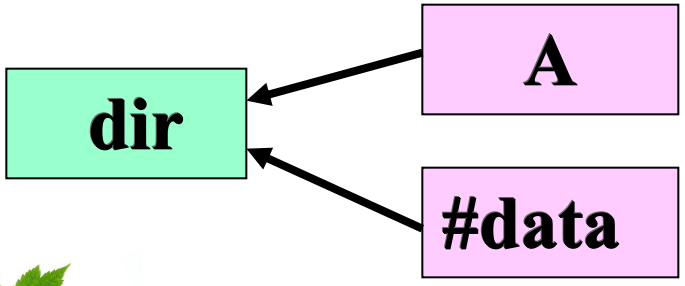
2.3算术运算和逻辑运算指令

1.以A为目的操作数的算术、逻辑运算指令（24条）



加	ADD	A ,	Rn,
进位加	ADDC		@Ri
借位减	SUBB		#data
与	ANL		dir
或	ORL		
异或	XRL		

以dir为目的操作数逻辑运算指令（6条）



与	ANL	dir ,	A
或	ORL		
异或	XRL		#data

算术类指令的操作意义非常明确，不一一赘述，注意减指令只有带借位减，因此在多字节减法中，最低字节作减法时，注意先清CY。

逻辑运算是按位进行的，两数运算的运算法则是：

与：有“0”则“0”；

或：有“1”则“1”

异或：同为“0”， 异为“1”；

与“0”异或值不变：与“1”异或值变反。

逻辑指令常用于对数据位进行加工。

例：A=0FH， 执行ORL A,#86H 后 A＝8FH

例：A=0FH， 执行XRL A,#86H 后 A＝89H

```
  00001111
∨ 10000110
----------
  10001111

  00001111
⊕ 10000110
----------
  10001001
```

2.加1、减1指令

加 1 指令:

INC { **A** / **Rn** / **@Ri** / **dir** / **DPTR**

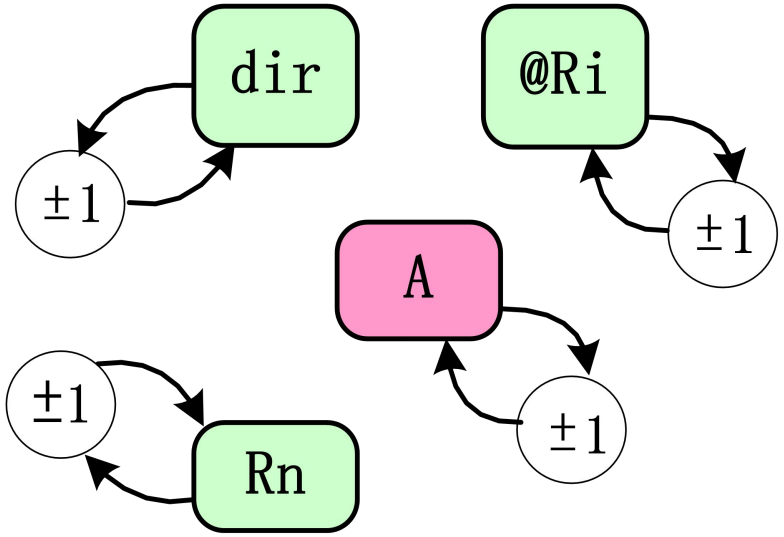


图2.13 INC、DEC指令

减 1 指令:

DEC { **A** / **Rn** / **@Ri** / **dir**

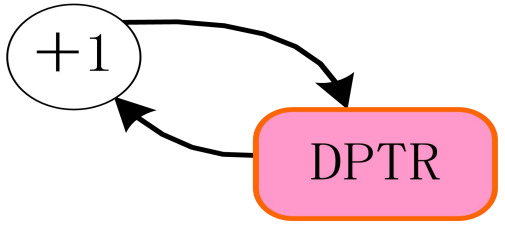


图2.14 INC DPTR

提问：没有DEC DPTR指令，怎么解决DPTR的减 1?

3. 十进制调整指令

计算机完成二进制加法其和也为二进制，如果是十进制相加（即BCD码相加）想得到十进制的结果，就必须进行十进制调整（即BCD 码调整）。

调整指令： DA A ；将A中二进制相加和调整成BCD码

调整方法：和低4位大于9或有半进位则低4位加6；

和的高4位大于9或有进位，则高4位加6。

指令根据相加和及标志自行进行判断，因此该指令应紧跟在加指令之后，至少在加指令和该指令之间不能有影响标志的指令。

DA A指令只对一个字节和调整，如为多字节相加必须进行多次调整。此指令不能对减法结果进行调整。

例 完成56+17的编程。

```
MOV A，#56H ；A存放BCD码56H
MOV B，#17H ；B存放BCD码17H
ADD A，  B      ；A=6dH
DA A            ；A=73H
SJMP $
```

```
   5 6 H
+  1 7 H
---------
   6 d H
+    6
---------
   7 3 H
```

指令对标志位的影响有如下规律：

1） 凡是对A 操作指令（包括传送指令）都将A中1个的奇偶反映到PSW的P标志位上。即A中奇数个“1”，P=1；偶数个“1”，P=0。

2） 传送指令、加 1、减 1 指令、逻辑运算指令不影响Cy、OV、AC 标志位。

3） 加减运算指令影响标志位，乘除指令使Cy=0，当乘积大于255，或除数为0时，OV置1。

4） 对进位位Cy（指令中用C表示）进行操作的指令和大环移指令，显然会影响Cy。

具体指令对标志位的影响可参阅附录A。

标志位的状态是控制转移指令的条件，因此指令对标志位的影响应该记住。

例：A= 9AH，R2= E3H，PSW= 0，执行指令

ADDC A，R2 后求：

A= 7DH，Cy= 1，OV= 1，AC= 0，P= 0

PSW= ？

10000100 = 84H

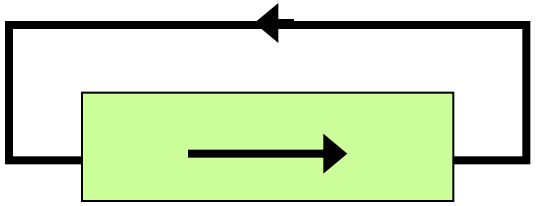
```
  1001 1010
  1110 0011
+          0
-----------
1 0111 1101
```

CY	AC	F0	RS1	RS0	OV	---	P
1	0	0	0	0	1	0	0

4.移位指令（仅对 A）

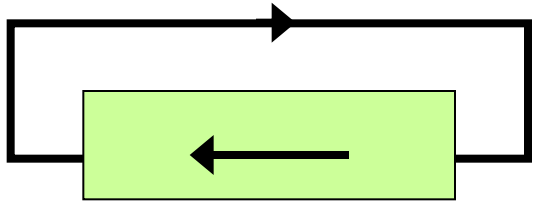
设 A 10010110 CY 1

RR A



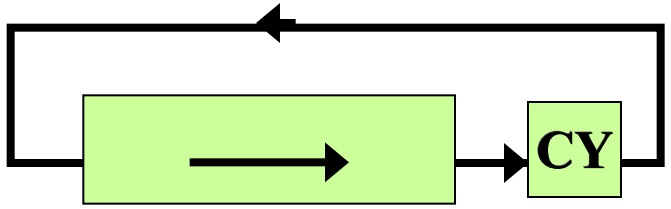
A 01001011

RL A



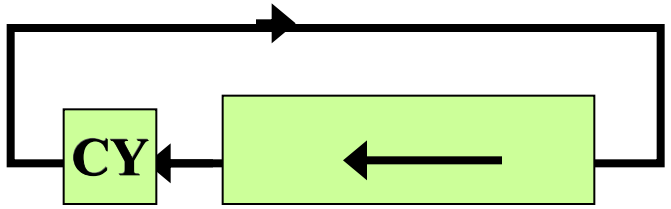
A 00101101

RRC A



A 11001011 CY 0

RLC A



A 00101101 CY 1

2.4 控制转移指令

这一类指令的功能是改变指令的执行顺序，转到指令指示的新的PC地址执行。

MCS-51单片机的控制转移指令有以下类型：

无条件转移：无需判断，执行该指令就转移到目的地址。

条件转移：需判断标志位是否满足条件，满足条件转移到目的地址，否则顺序执行。

绝对转移：转移的目的地址用绝对地址指示，通常为无条件转移。

相对转移：转移的目的地址用相对于当前PC的偏差（偏移量）指示，通常为条件转移。

长转移或长调用：目的地址距当前PC 64KB地址范围内。

短转移或短调用：目的地址距当前PC 2KB地址范围。

2.4.1 调用程序和返回类指令

1.长调用

LCALL addrl16 ； addr16→PC0～15

说明：

（1）该指令功能是

①保护断点，即当前PC（本指令的下一条指令的首地址）压入堆栈。

②子程序的入口地址 addr16 送PC，转子程序执行。

（2）本指令为64KB地址范围内的调子程序指令，子程序可在64KB地址空间的任一处。

（3）本指令的机器码为三字节 12 addr16。

2.短调用

ACALL addr11 ;addr11→PC0～10

说明：

（1）该指令的功能是

①保护断点，即当前PC压入堆栈。

② addrl11→PC0～10,而PC11～15保持原值不变。

（2）本指令为2KB地址范围的调子程序指令，子程序入口距当前PC不得超过2KB地址范围。

（3）本指令的机器码为二字节，设addr11的各位是a10a9a8…a2a1a0，则ACALL指令机器码a10a9a810001a7a6a5a4a3a2a1a0，其中10001是ACALL指令的操作码。

例 子程序调用指令ACALL在 程序存储器中的首地址为0100H，子程序入口地址为0205H。试确定能否使用ACALL指令实现调用？如果能使用，确定该指令的机器码。

解: 因为ACALL指令首地址在0100H，而ACALL是 2字节指令，所以下一条指令的首地址在0102H。0102H和0250H在同一2KB地址范围内，故可用ACALL调用。调用入口地址为0250H，ACALL指令的机器码形式为：

0101000101010000B=5150H。

3.子程序返回指令

RET ；从调用子程序返回。

功能：从栈顶弹出断点到PC。

RETI ； 从中断服务程序返回。

功能：从栈顶弹出断点到PC，并恢复中断优先级状态触发器。

2.4.2 转移指令

1. 无条件转移指令

（1）短转移

AJMP addr11 ； addr11→PC0～10

说明：

①转移范围：本指令为2KB地址范围内的转移指令。对转移目的地址的要求与ACALL指令对子程序入口地址的要求相同。

②机器码形式：本指令为2字节指令。设addr11的各位是a10a9a8…a2a1a0,则指令的机器码为a10a9a800001a7a6a5a4a3a2a1a0。

（2）长转移

LJMP addr16 ；addr16→PC0～15

说明：

①本指令为64KB程序存储空间的全范围转移指令。转移地址可为16位地址中的任意值。

②本指令为3字节指令02 addr16。

（3）间接转移

JMP @A+DPTR ；A+DPTR→PC

例 A=02H，DPTR=2000H，指令JMP @A+DPTR执行后，PC=2002H。也就是说，程序转移到2002H地址单元去执行。

例　现有一段程序如下：

```
        MOV   DPTR，#TABLE
        JMP   @A+DPTR
TABLE：AJMP   PROC0
        AJMP   PROC1
        AJMP   PROC2
        AJMP   PROC2
```

根据JMP @A+DPTR指令的操作可知，

当A=00H时，程序转入到地址 PROC0 处执行；

当A=02H时，转到PROC1处执行……

可见这是一段多路转移程序，进入的路数由A确定。因为AJMP指令是2字节指令，所以 A 必须为偶数。

以上均为绝对转移指令，下面介绍相对转移指令。

（4）无条件相对转移

SJMP rel ；PC+rel→PC，

即As＋2＋rel→PC，机器码为80 rel

说明：

As为源地址（本指令的首地址），该指令为2字节指令，执行本指令时

当前PC＝As+2，rel 为转移的偏移量，转移可以向前转（目的地址小于源地址），也可以向后转（目的地址大于源地址），因此偏移量rel 是 1 字节有符号数，用补码表示（－128～＋127），所以指令转移范围在离源地址As的－126～＋129字节之间。

2. 条件转移指令

（1）累加器为零（非零）转移

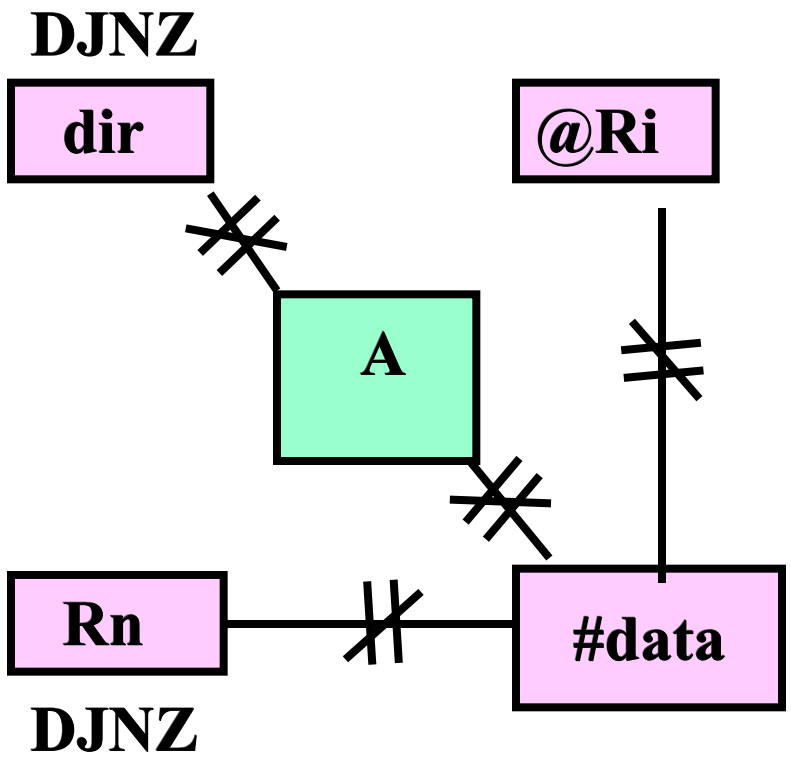
```
JZ    rel
    ；A=0 则转移（As+2+rel→PC）
JNZ   rel
    ；A≠0 程序顺序执行，机器码为60rel
```

（2）减 1 不等于零转移

DJNZ Rn , rel ； Rn-1

DJNZ dir , .rel

本指令有自动减 1 功能。



（3）比较转移指令

CJNE A,dir , rel

CJNE A , #data , rel

CJNE Rn , #data , rel

CJNE @Ri , #data , rel

说明：

① CJNE指令都是3字节指令，作减操作，不回送结果，影响CY标志。

② 若第一操作数大于或等于第二 操作数，则标志CY=0。若第一操作数小于第二操作数，则CY=1。

这几条指令除实现两操作数相等与否的判断外，利用对CY的判断，还可完成两数大小的比较。

试说明以下一段程序运行后A中的结果。

```
        MOV   23H，#0AH
        CLR    A
LOOP：  ADD   A，23H
        DJNZ   23H，LOOP
        SJMP   $
```

根据程序可知

A=10+9+8+7+6+5+4+3+2+1=55=37H

例 编写程序，要求读P1 端口上的信息，若不为55H，则程序等待，直到P1端口为55H时，程序才往下顺序执行。

程序：

```
MOV  A，#55H  ；A＝55H
CJNE  A，P1，$ ；P1≠55H，则程序循环执行本指令
…
```

在实际编程中，转移的目的地址不管是addr11、addr16、还是rel ,均是一符号地址表示的（如SJMP ABC，AJMP LOOP…），转移的类型是通过指令的操作符来决定的。

3.相对偏移量rel的求法

在相对转移中，用偏移量 rel 和转移指令所处的地址值来计算转移的目的地址，rel 是 1 字节补码. 在填机器码时，需计算rel,下面介绍计算rel 的方法。

设 本条转移指令的首地址为As——源地址，

指令字节数为Bn——2字节或3字节，

要转移的目标地址为Ad——目的地址，

当前PC= As+ Bn 因为在执行本条指令时，PC已经指向了下一条指令,见下图：

Ad ——→0100　　MN:　　…….

Bn=3　　…….

As ——→0125　BF 05 rel　　CJNE R7,#06,MN

当前PC——→0128

当前PC= As +Bn=0125+3=0128

于是　rel =Ad-当前PC =Ad－(As+Bn)= Ad - As - Bn

在上例中 rel=Ad－当前PC =0100H－0128H=－28H

－28求补得D8H

于是　　rel=(Ad-As-Bn)补

这就是在已知源地址，目的地址和指令的长度时，计算rel 大小的公式。

例 MCS-51单片机指令系统中，没有停机指令，通常用短转移指令SJMP $ ($为本条指令的首地址)来实现动态停机的操作，试写出这条指令中机器码。

解： 查附录A，SJMP rel 的指令码为80rel 据题意 本条指令的首地址 As=$，转移的目的地址是本条指令地址，即Ad=$ 该指令为两字节，即Bn=2，

rel= (Ad-As-Bn)补= ($－$－2)补=（－2）补=FEH

所以SJMP $指令的机器码是80FEH。

例 计算下面程序中CJNE指令的偏移量。

```
LOOP：MOV  A，P1
      CJNE  A，#55H，LOOP
```

解：由于MOV A,P1是2字节指令，故CJNE指令的首地址是LOOP+2。又因为CJNE是3字节指令，于是有：Ad=LOOP ,As=LOOP+2, Bn=3

rel=[LOOP-(LOOP+2)-3]补=[-5]补=FBH

所以CJNE A,#55H,LOOP的指令码为B455FBH。

2.4.3 空操作指令

NOP 机器码 00

该指令经取指，译码后不进行任何操作（空操作）而转到下一条指令，常用于生产一个机器周期的延时，或上机修改程序时作填充指令，以方便增减指令。

例 将A累加器的低四为取反四次、高四位不变。每变换一次，从P1输出。

方法一 加 1 计数：

```
      MOV  R0，#0          ；计数初值送0
LL：  XRL  A，#0FH         ；高4位不变，低四位取反
      INC  R0              ；次数加1
      MOV  P1，A           ；从P1输出
      CJNE R0，#04，LL     ；不满四次循环
      RET
```

方法二 减1计数:

```
     MOV  R0,#04H          ;计数初值送4
LL: XRL  A，#0FH
      MOV  P1，A
      DJNZ R0，LL          ；次数减1不等于0循环
      RET
```

例 在内部RAM的40H地址单元中，有1字节符号数，编写求其绝对值后放回原单元的程序。

程序如下：
```
MOV  A，40H
ANL  A，#80H
JNZ  NEG        ；为负数转移
SJMP  $         ；为正数，绝对值=原数，不改变原单元内容
NEG：MOV A，40H ；为负数求补，得其绝对值
CPL  A
INC  A
MOV  40H，A
SJMP  $
```

有符号数在计算机中以补码形式存放，例如－5，存放在内部RAM中为FBH，求补后得5， 即|－5|=5。

2.5 位操作指令

MCS-51单片机的特色之一就是具有丰富的位处理功能，以进位标志CY为位累加器C，使得开关量控制系统的设计变得十分方便。

在程序中位地址的表达有多种方式：

1）用直接位地址表示，如D4H。

2）用“.”操作符号表示，如PSW.4，或D0H.4

3）用位名称表示，如RS1。

4）用用户自定义名表示。如ABC BIT D4H，其中ABC定义为D4H位的位名，BIT为位定义伪指令。以上各例均表示PSW.4的RS1位。

位操作类指令的对象是C和直接位地址，由于C是位累加器，所以位的逻辑运算指令目的操作数只能是C，这就是位操作指令的特点。下面将位操作的17条指令介绍如下。

1．位清零

CLR C ；0→CY

CLR bit ；0→bit

2．位置 1

SETB C ；1→CY

SETB bit ； 1→bit

3．位取反

CPL C ； CY →CY

CPL bit ； bit → bit

4.位与

ANL C，bit ；CY∧（bit）→CY

ANL C，/bit ；CY∧（bit）→CY

5.位或

ORL C，bit ；CY∨（bit）→CY

ORL C，/bit ；CY∨（bit）→CY

6. 位传送

MOV C，bit ；（bit）→CY

MOV bit,C ;CY→bit

7. 位转移

位转移根据位的值决定转移，均为相对转移指令，设As为下面各指令的首地址。

JC rel ;CY=1，则转移（As+2+rel→PC），否则程序顺序执行

JNC rel ;CY=0,则转移（As+2+rel→PC），否则程序顺序执行

JB bit，rel ;(bit)=1,则转移（As+3+rel→PC），否则程序顺序执行

JNB bit,rel ;(bit)=0,则转移（As+3+rel→PC），否则程序顺序执行

JBC bit,rel ;(bit)=1,则转移（As+3+rel→PC），且该位清零；否则程序顺序执行

例 用位操作指令实现 X＝X0⊕ X1，设X0为P1.0， X1为P1.1， X为ACC.0。 ■

解(1)：因位操作指令中无异或指令，依据X＝X0⊕ X1＝X0X1＋X0X1 ■用与、或指令完成，编程如下： ■

```
 X   BIT   ACC.0
 X0  BIT   P1.0
 X1  BIT   P1.1      ;位定义
MOV  C，X0
ANL  C，  /X1       ;C=X0∧X1
 MOV  20H，C        ;暂存于20H 单元
MOV  C，X1
ANL  C，/X0         ;C=X0∧X1
ORL  C，20H         ；C＝X0X1＋X0X1
MOV  X，C
SJMP  $
```

解（2）：根据异或规则，一个数与“0”异或，该数值不变；与“1”异或，该数值变反，编程如下：

```
        MOV  C，X0
        JNB   X1，NCEX    ; X1=0，X=C=X0
        CPL  C
NCEX：MOV  X，C    ;X1=1，X=C=X0
        SJMP   $
```

小 结

（1）51系列单片机指令系统的特点是不同的存储空间寻址方式不同，适用的指令不同，必须进行区分。

（2）指令是程序设计的基础，应重点掌握传送指令、算术运算指令、逻辑运算指令、控制转移指令和位操作指令，掌握指令的功能，操作的对象和结果，对标志位的影响，应要求熟记。