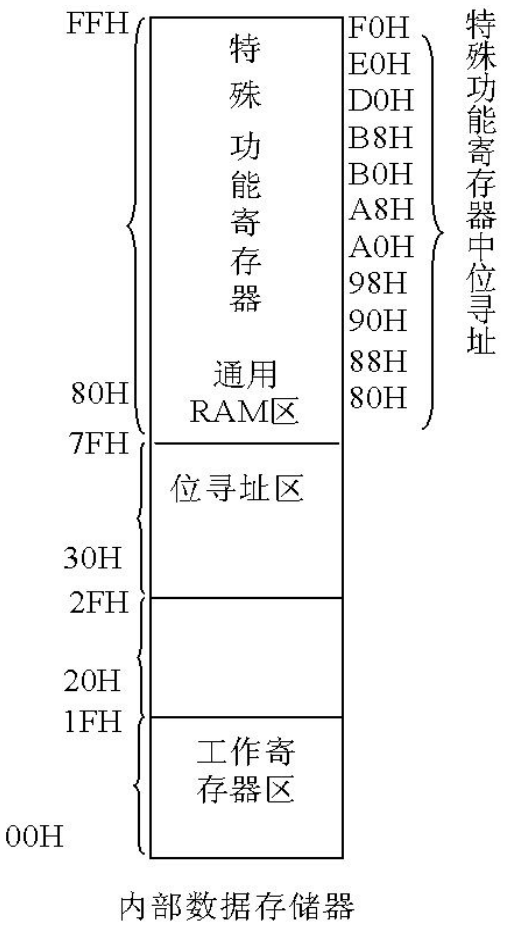
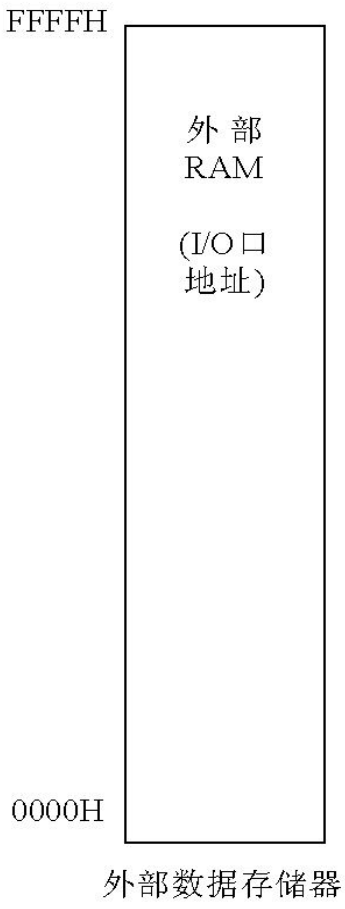
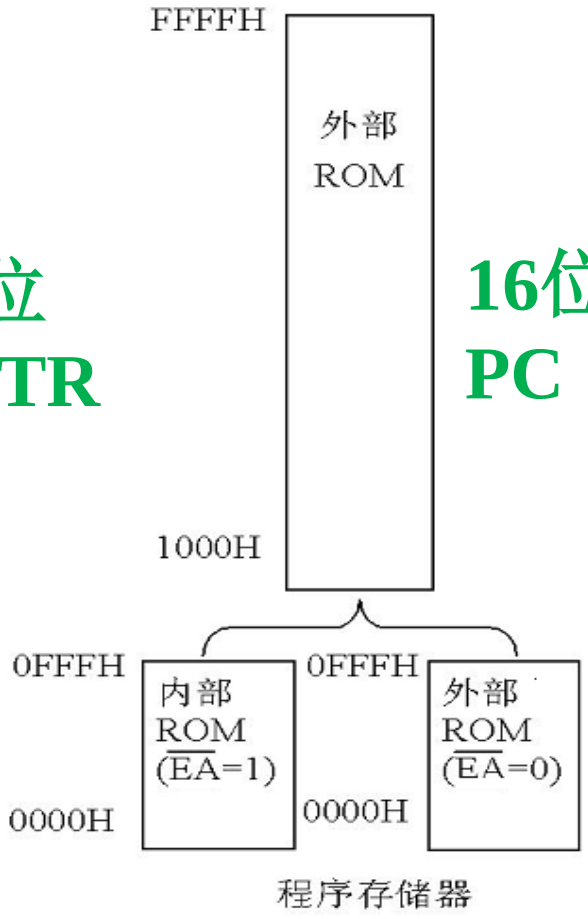
复习1

第二章复习





16位
DPTR



16位
PC

MOV

MOVX

MOVC

复习2

内部低128单元(00-7FH)

数据缓冲区/堆栈区：

30H～7FH

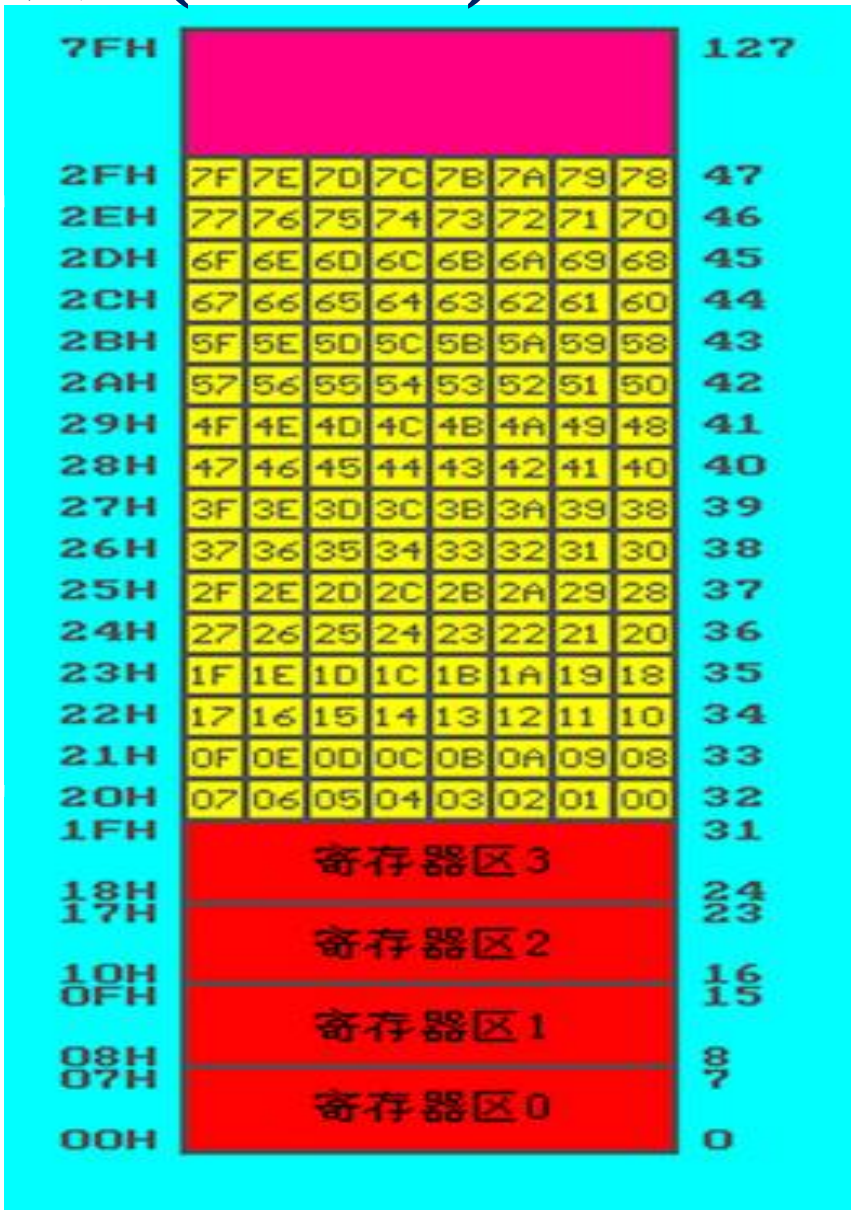
位寻址区：

字节地址：20H～2FH

位地址为：00H～7FH

工作寄存器区：

字节地址：00H～1FH



复习3　　单片机复位后的初始状态

表 2-4　复位后各专用寄存器的状态

寄存器	内　容	寄存器	内　容
PC	0000H	T2CON	00H
ACC	00H	TH0	00H
B	00H	TL0	00H
PSW	00H	TH1	00H
SP	07H	TL1	00H
DPTR	0000H	TH2	00H
P0～P3	0FFH	TL2	00H
IP(8051)	××00000B	RLDH	00H
(8052)	××000000B	RLDL	00H
IE(8051)	0××00000B	SCON	00H
(8052)	0×000000B	SBUF	不定
TMOD	00H	PCON(HMOS)	0××××××B
TCON	00H	(CHMOS)	0××0000B

第3章　MCS-51单片机指令系统

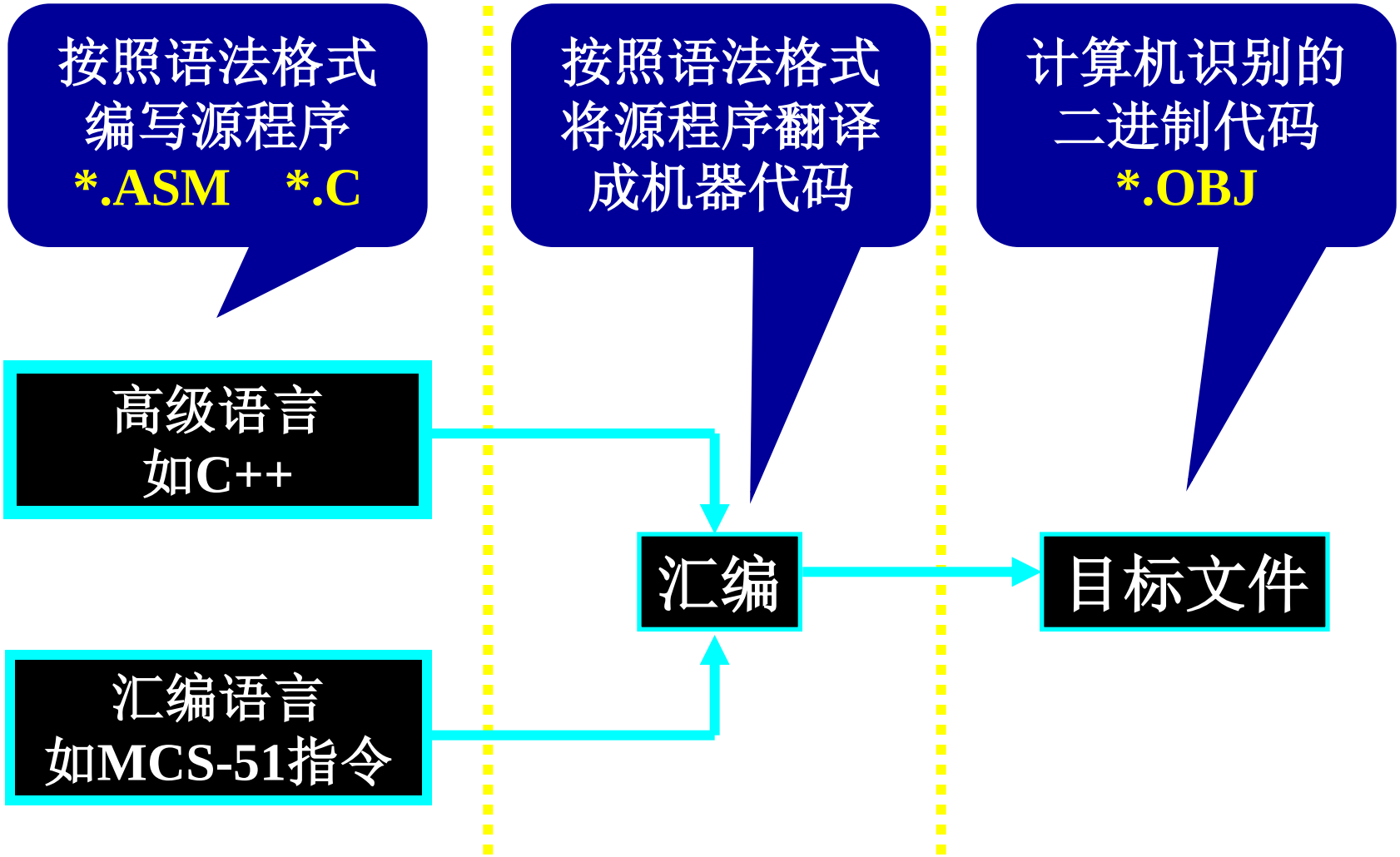
3.1　MCS-51指令系统概述

3.2　寻址方式

3.3　分类指令

3.1 MCS-51指令系统概述

一、编译过程：



二、111条 MCS-51指令的分类方法

1、按照每条指令执行时间划分：

单机器周期（12个振荡周期）有64条
双机器周期（24个振荡周期）有45条
四机器周期（48个振荡周期）有2条

2、按照每条指令翻译成机器码的字节数划分：

单字节指令，有49条
双字节指令，有45条
三字节指令，有17条

3、按照指令功能划分：

数据传递与交换、算术运算、逻辑运算、程序转移、布尔处理操作、CPU控制等6类。
布尔处理操作类指令又称位操作指令。

三、指令格式

一般分为两部分：操作码和操作数

操作码：规定指令进行什么操作

操作数：表示指令操作的对象，可以是数或是数据所在的地址，但最终对象都是数据。

格式为：[标号：] 操作码助记符 [目的操作数，] [源操作数] [；注释]

MAIN: MOV A , #20H ；####

按指令的字节数(机器语言)不同，分为一字节指令，二字节指令，三字节指令

1. 一字节指令

8位编码，既包含操作码，又包含操作数

如果指令无操作数（如指令：NOP）或操作数都在Rn，A，B， DPTR，C中，则该指令一定是一字节指令。

a. 如 INC DPTR

操作码 A3H	10100011

INC A

操作码 04H	00000100

b. 若是有工作寄存器，则指令码中的rrr三位的不同编码来指定该寄存器

MOV A，Rn 指令码为11101rrr (E8H～EFH)

2. 二字节指令

16位编码，既包含操作码，又包含操作数，如果指令操作数只有一个8位直接地址或立即数，则该指令一定是二字节指令。

ADD A，#22H

操作码 24H	00100100
操作数 22H	00100010

ADD A，22H

操作码 25H	00100101
操作数 22H	00100010

3. 三字节指令

24位编码，既包含操作码，又包含操作数

如果指令操作数有两个8位直接地址或是立即数，或是16位直接地址，则该指令一定是三字节指令。

MOV 5EH，4FH

操作码	85H
目的操作数	5EH
源操作数	4FH

MOV 5EH，#4FH

操作码	75H
目的操作数	5EH
源操作数	4FH

MOV DPTR，#5E4FH

操作码	90H
操作数高八位	5EH
操作数低八位	4FH

四、 指令描述符号介绍

Rn —— 当前选中的寄存器组中的8个工作寄存器R0～R7（n=0～7）。

Ri —— 当前选中的寄存器组中的可作为间接寻址寄存器使用的2个工作寄存器R0、R1 （i=0，1）。

Direct — 8位的内部数据存储器单元中的地址。可以是内部RAM单元地址或专用寄存器地址。 75H、 0A5H 、P1

#data——包含在指令中的8位立即数。 ＃75H、 ＃80H、 ＃0A5H（0~255）

#data16—包含在指令中的16位立即数。＃2480H、 ＃0D256H（0~65535）

addr16— 16位目的地址。用于长转移指令中，能转移到64KB程序存储器的任何地方。

addr11— 11位目的地址。目的地址应与下一条指令第一个字节在同一个2KB程序存储器的地址空间之内。

Rel —— 8位带符号的偏移字节，简称偏移量。偏移量相对于下一条指令的第一个字节计算，在-128--+127范围内取值。

DPTR——数据指针，可用作16位地址寄存器。

Bit —— 内部RAM或专用寄存器中的直接寻址位。

A —— 累加器。

ACC —— 直接寻址方式的累加器。

B —— 专用寄存器，用于乘法和除法指令中。

CY —— 进位标志或进位位，或布尔处理机中的累加器。

@ —— 间址寄存器或基址寄存器的前缀， 如@Ri，@DPTR。

/ —— 位操作数的前缀，表示对该位操作数取反， 如/bit。

× —— 片内RAM的直接地址或寄存器。

(×) —— 直接寻址方式中，表示直接地址X中的内容。注释用

((×)) — 在间接寻址方式中，表示由间址寄存器X指出的地址单元中的内容。注释用

← ——箭头左边的内容被箭头右边的内容所代替。注释用

3.2 寻址方式

寻址方式——找到存放数据的地址，提取操作数.

以我校通知搬家公司搬运桌子为例，两者区别是 单片机数据是拷贝型的，原数据还在。

1、 寄存器寻址
2、 直接寻址
3、 寄存器间接寻址
4、 立即寻址
5、 变址寻址
6、 位寻址
7、 相对寻址

返回本章首页

1.寄存器寻址方式：搬家公司→人事处→桌子组

操作数给出形式为寄存器，

操作对象（数据）存放在寄存器当中

例如：　　**MOV A ，R0**

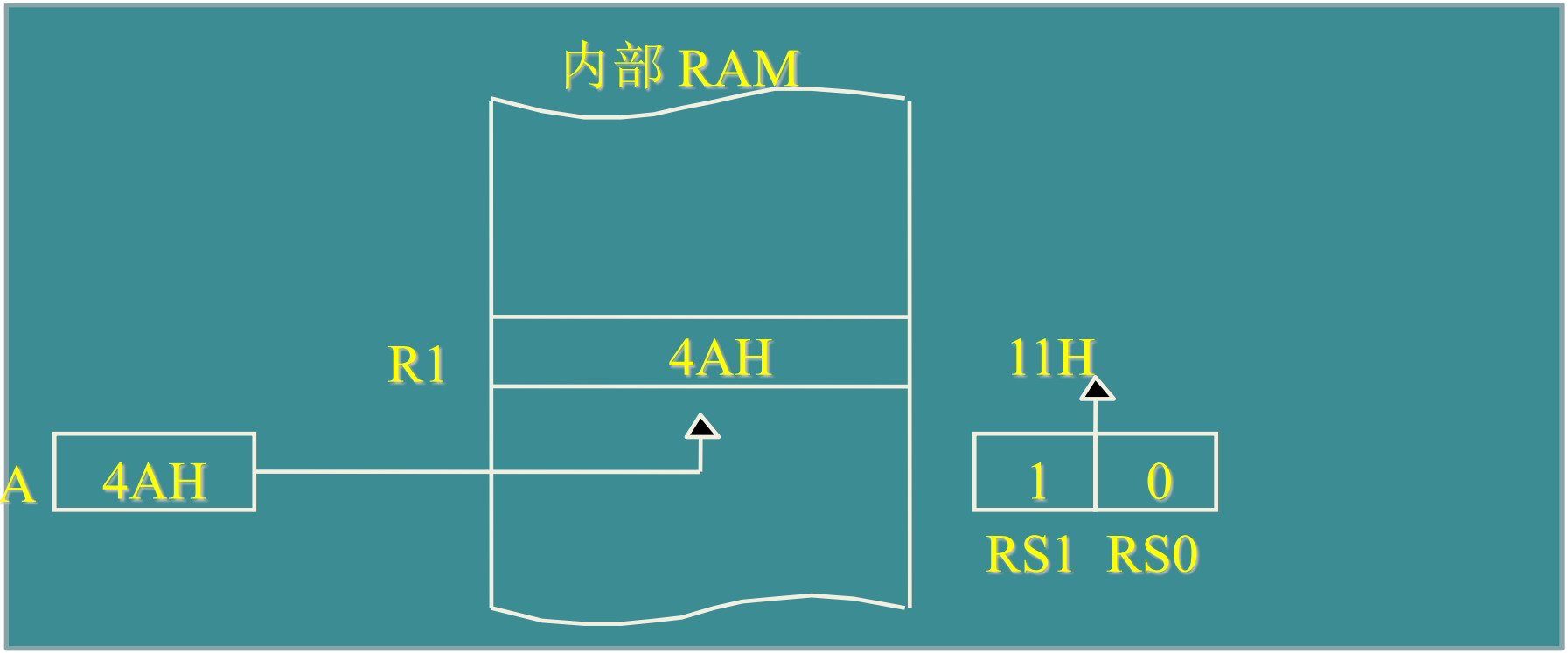
MOV R0 ， A

寄存器：　　**a.工作寄存器：Rn**

b.专用寄存器：A、B、DPTR

这里源操作数和目标操作数均采用寄存器寻址。

如果程序状态寄存器PSW的RS1RS0=10（选中第二组工作寄存器，对应地址为10H～17H），设累加器A的内容为4AH，则执行MOV R1，A 指令后，内部RAM 11H单元的值就变为4AH，如寄存器寻址示意图所示。



寄存器寻址示意图

2.直接寻址方式：搬家公司→5#楼211室→桌子组

操作数给出形式为单元地址，

操作对象（数据）存放在单元地址当中

例如：　　**MOV　A　，5BH**

MOV　5BH，　A

直接地址：**a.** 内部**RAM**低**128**字节区

b.专用寄存器区（直接寻址是访问**SFR**区的唯一方法）

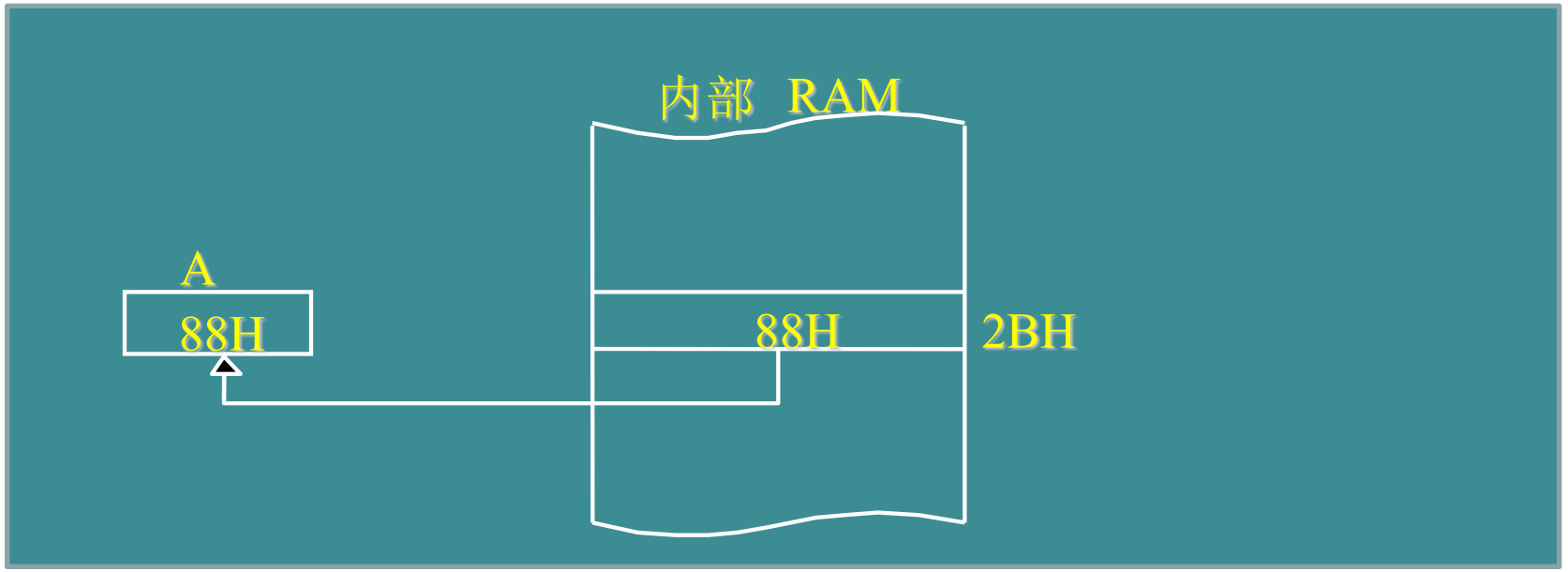
既可用单元地址形式给出，也可用寄存器符号给出

MOV A ，P1　（二字节指令）

MOV A，90H （二字节指令）

例如指令：**MOV A，2BH** 执行的操作是将内部RAM 中地址为2BH的单元内容传送到累加器A中，其操作数2BH就是存放数据的单元地址，因此该指令是直接寻址。

设内部RAM 2BH单元的内容是88H，那么指令**MOV A，2BH**的执行过程如立即数寻址示意图所示。



直接寻址示意图

3.寄存器间接寻址方式： @

搬家公司→值班室拿条子→ 5#楼211室→桌子组

寄存器寻址： 寄存器当中存放的是操作对象

寄存器间接寻址：寄存器当中存放的是操作对象（数据）的地址，而不是操作对象本身。

例如：

```
MOV  30H,  #20H
MOV  R0 ,  #30H
MOV  A   ,  R0     ;  R0 → A        A=30H
MOV  A   ,  @R0    ;  (R0) → A      A=20H
```

寄存器间接寻址范围：

a. 访问内部RAM（SFR 不能用于寄存器间接寻址）：

只能使用R0和R1来作为间接寄存器，用@Ri表示（i＝0、1）

MOV A，@Ri

b. 访问外部低256字节 ：MOVX A，@Ri

c. 访问外部RAM所有64K字节： MOVX A，@DPTR

d.访问内部RAM SP, SP隐含了

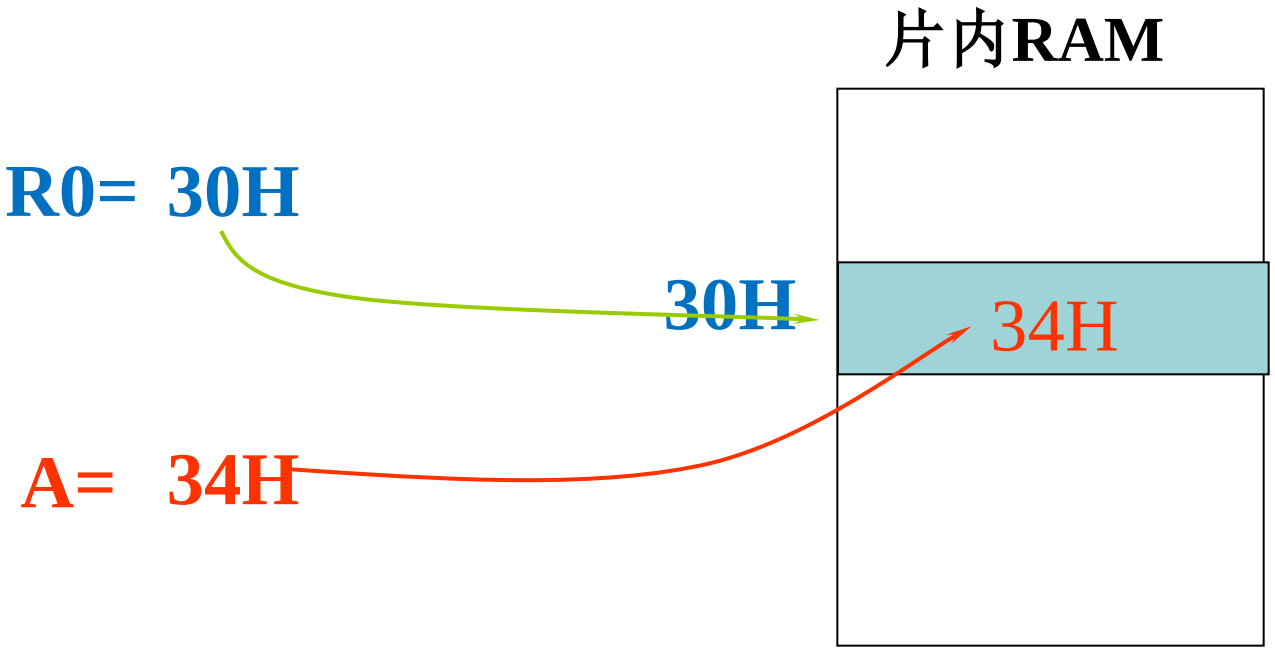
PUSH B 相当于 MOV @SP，B

POP B 相当于 MOV B，@SP

注意：SFR不能用间接寻址方式，只能用直接寻址方式

例如：MOV @R0，A ；内部RAM(R0)←A

其指令操作过程示意图如下图所示。



寄存器间接寻址示意图

4 .立即寻址方式： #　　搬家公司→桌子组

操作数就是操作对象（数据）本身，

此时把指令中操作数称为立即数，用#data表示

例如：

MOV A ， # 3AH　　MOV DPTR,#3F00H　　MOV 30H, #30H

唯一一条16位立即寻址指令：　　MOV DPTR ， #data16

通常与MOVX A，@DPTR 类指令搭配使用，#data16中表示的是16位地址

MOV DPTR ， #3425H

MOVX A，@DPTR

5.变址寻址方式：　搬家公司 → （主楼5层+1层 ）→桌子组

变址寻址是指将基址寄存器与变址寄存器的内容相加，结果作为操作数的地址。**DPTR**或**PC**是基址寄存器，累加器**A**是变址寄存器。该类寻址方式主要用于查表操作即访问程序存储器（**ROM**）中的数据表格，专门针对程序存储器的寻址方式。

只有三条：　**MOVC A，@A+DPTR ; A←（A+DPTR）**

MOVC A，@A+PC ;　A←（A+PC）

JMP @A+DPTR ；程序跳转到以 A+DPTR 为地址的单元

例如，指令 **MOVC A，@A+DPTR** 执行的操作是将累加器**A**和基址寄存器**DPTR**的内容相加，相加结果作为操作数存放的地址，再将操作数取出来送到累加器**A**中。

6、 位寻址　　搬家公司→某个桌子

1）寻址范围：

a. 内部RAM中的位寻址区：16个字节单元 ： 20H～2FH
128位 ： 00H～7FH

b.专用寄存器的可寻址位：
可供位寻址的专用寄存器11个，实际寻址位83个，即有些位不能寻址

2）表示方法：

直接使用位地址，例如PSW第5位 ： 0D5H
位名称 ： F0
单元地址加位 ： 0D0H.5
专用寄存器符号名加位 ： PSW.5

位寻址只能对有位地址的单元作位寻址操作。位寻址其实是一种直接寻址方式，不过其地址是位地址。在进行位操作时，用进位位C作为位累加器。位地址在指令中用bit表示。

7 .相对寻址方式：

相对寻址是指程序计数器PC的当前内容与指令中的操作数相加，其结果作为跳转指令的转移地址（也称目的地址）。相对寻址用于修改PC值，主要用于实现程序的分支转移（跳转）。相当于C语言中的goto语句。

例如：SJMP rel ；PC＋rel→PC

注意两点：

a. 当前PC值：执行完该条相对寻址指令后的PC值
   所以有：目的地址＝指令地址＋指令长度＋rel

b. rel为8位补码：－128～+127，负数表示向后转移，正数表示向前转移

以指令地址为基点：

向地址增大方向，最大可转移（127＋指令长度）个单元地址，

向地址减小方向，最大可转移（128－指令长度）个单元地址。

寻址方式小结：

寄存器寻址：　工作寄存器R0—R7，A，B，DPTR

直接寻址：　片内RAM低128字节和专用寄存器区,这是SFR区的唯一寻址方式。

寄存器间接寻址：　片内RAM （@R0, @R1,SP)

片外RAM （@R0, @R1,@DPTR)

变址寻址：　程序存储器(用于访问程序存储器中的数据表格)

相对寻址：　程序存储器256字节范围(解决在ROM内部程序的转移)

作业：

1、80C51单片机有哪几种寻址方式？各寻址方式所对应的寄存器或存储器空间如何？

2、试根据指令编码表写出下列指令的机器码。

（1）MOV A，#88H

（2）MOV R3，50H

（3）MOV P1.1，55H

3.3　分类指令

按照指令的功能，可以把MSC-51的111条指令分成五类：

数据传送类指令（29条）

算术运算类指令（24条）

逻辑操作类指令（24条）

控制转移类指令（17条）

位操作类指令　（17条）

一、 数据传送类指令

共29条：

- 内部RAM间的数据传送指令（16条）
- 访问外部RAM的数据传送指令（4条）
- 访问ROM程序存储器的数据传送指令（2条）
- 交换类指令（5条）
- 堆栈指令（2条）

通用格式为： MOV　<目的操作数>，<源操作数>

传送指令的约定：从右向左传送数据，是将源操作数送到目的操作数。指令执行后，源操作数不变，目的操作数被源操作数取代。

数据传送类指令用到的助记符有**MOV、MOVX、MOVC、XCH、XCHD、SWAP、PUSH、POP**，共**8**种。

<2>十六位数据传送指令（1条）　　P51

MOV　DPTR，#data16　；(DPH)←dataH8，(DPL)←dataL8

如：**MOV　DPTR，#2368H　⇔　MOV　DPH，#23H**

MOV　DPL，#68H

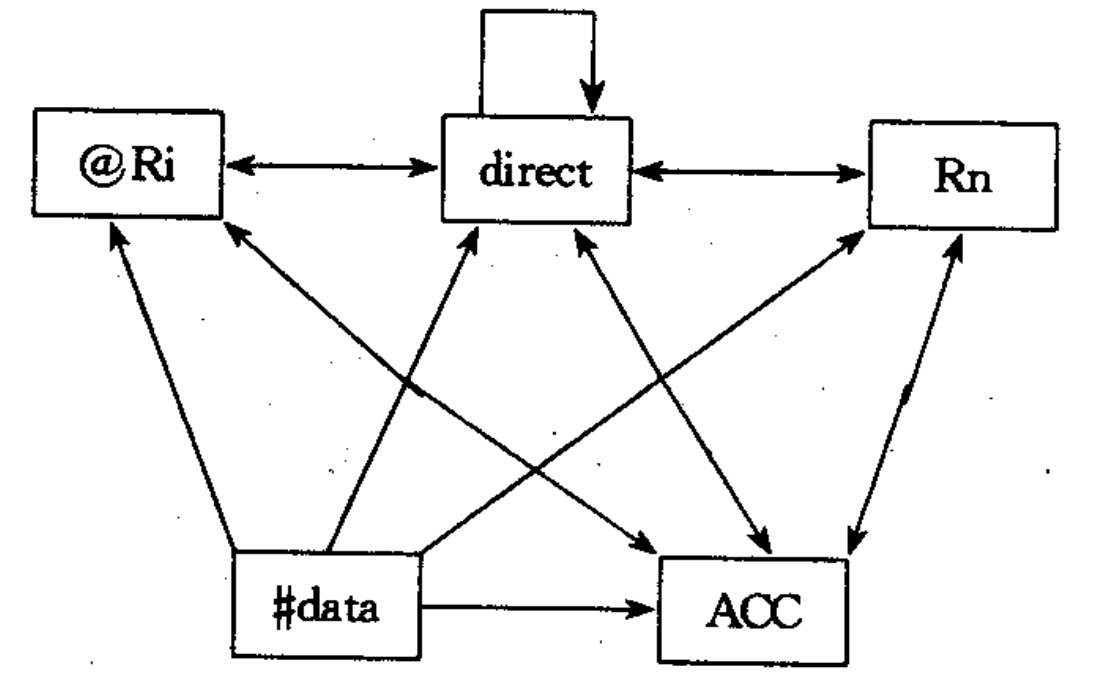
MOV　DPTR，#35326

举例：设内部RAM40H单元的内容为30H，30H单元的内容为20H，20H单元的内容为10H，端口P1的内容为AAH，试问结果如何？

```
MOV   R0, #40H                          改为MOV     R0, 40H
MOV   A ,@R0      ;A=30H                ; A=20H
MOV   R1,A        ; R1=30H              ; R1=20H
MOV   B,@R1       ; B=20H               ; B=10H
MOV   @R1,P1      ; (30H)=AAH           ; (20H)=AAH
MOV   P2,P1       ; P2=AAH              ; P2=AAH
MOV   40H,#10H    ; (40H)=10H           ; (40H)=10H
```

问：R0=?　R1=?　B=?　(40H)=?　(30H)=?　P2=?　(20H)=?

MOV指令在片内RAM的允许操作图



不允许的操作有：

	@Ri←→@Ri	Rn←→Rn	@Ri←→Rn
如：	MOV Rn，Rn	x	
	MOV @Ri, @Ri	x	
	MOV Rn, @Ri	x	
	MOV #data, A	x	等等指令是非法指令。

2、外部数据存储器读写指令（4条）　　P52

用间接寻址方式

MOVX　A，@Ri　　；A←((Ri))，且使/RD=0

MOVX　A，@ DPTR　；A←((DPTR))，且使/RD=0

MOVX　@Ri，A　　；((Ri)) ← (A)，且使/WR=0

MOVX　@ DPTR，A　；((DPTR)) ← (A)，且使/WR=0

1) 外部数据的传送只能使用A

2) @ DPTR两条指令以DPTR为片外RAM16位地址指针，寻址范围为64KB空间；P2输出高8位地址，P0口输出低8位地址和数据,分时复用

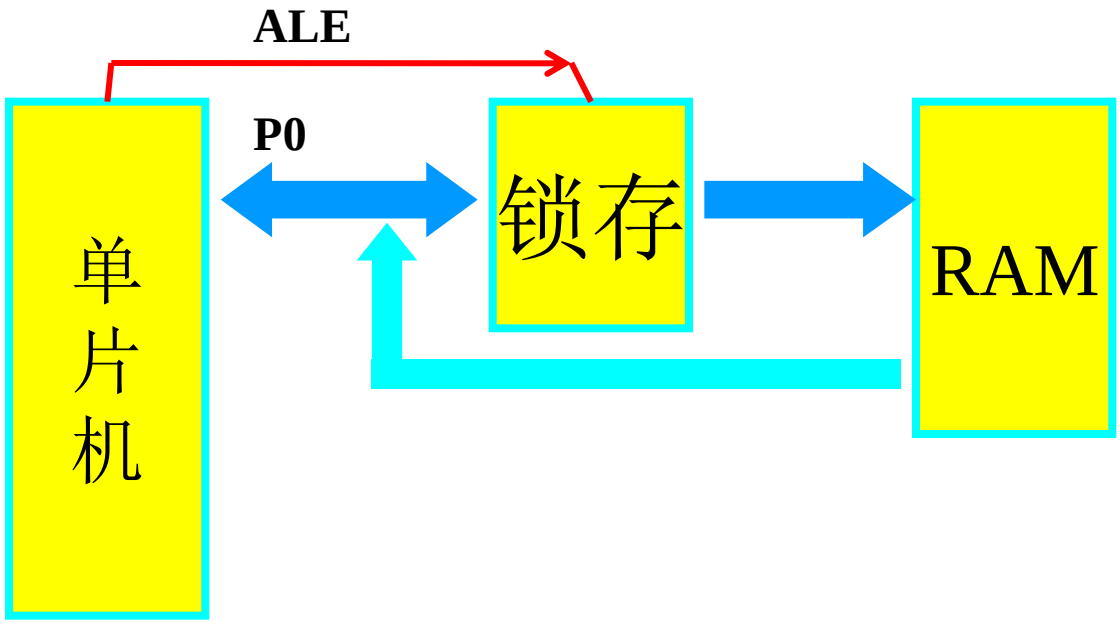
3) @Ri两条指令以R0或R1作低8位地址指针，由P0口送出，寻址范围为256B空间（P2口仍可作通用I/O口）。

例：将外部RAM中F5H单元中的内容送到0645H单元中

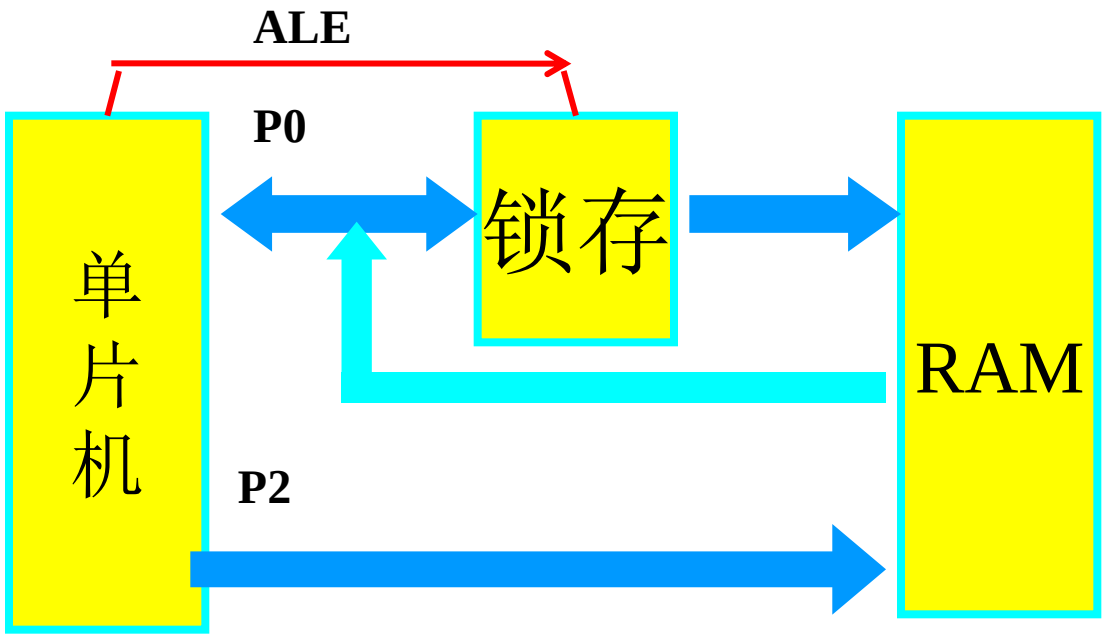
```
MOV     DPTR,#0645H
MOV     R0,#0F5H
MOVX    A , @R0
MOVX    @DPTR , A
```

例：将外部RAM中0F5H单元中的内容送到A单元中。

```
MOV     R0,#0F5H
MOVX    A , @R0
```



8bit寻址256字节



16bit寻址64K字节

3、程序存储器读指令（查表指令）（2条）

MOVC　A，@A+DPTR　　；　A←((A)+（DPTR))

一般DPTR放表的首地址，A放所查数据在表中的偏移；查表范围为64KB空间，称为远程查表。

MOVC　A，　@A+PC　　；　A←((A) + (PC))

PC的值为下条指令的地址，　A放所查数据相对PC值的偏移；查表范围为最大为256B空间，称为近程查表。

注意：内外程序存储器在逻辑上连续统一，传送单向，只读。

这类指令用于访问程序存储器中的数据表格。

例1：近程查表 P53

```
地址   标号   语句
2000   HBA:  INC    A              ;  A=(00H--0FH)
2001         MOVC  A，@A+PC       ;  查表
2002         RET
       TAB:  DB     30H            ;  0的ASC码
             DB     31H            ;  1的ASC码
             。 。 。
             DB     39H            ;  9的ASC码
             DB     41H            ;  A的ASC码
             。 。 。
             DB     46H            ;  F的ASC码
```

例如A初值01H，查表得31H； A初值0FH，查表得46H

4、数据交换指令（5条）

（1）字节交换指令

XCH　A，Rn　　；　(A) ←→ (Rn)

XCH　A，direct　　；　(A) ←→ (direct)

XCH　A，@Ri　　；　(A) ←→ ((Ri))

（2）半字节交换指令

低半字节交换指令

XCHD　A，@Ri　　；　(A) 0 ~3 ←→ ((Ri)) 0~3

半字节交换操作

如：设(A) =36H，(R1) =65H，(65H) =42H

XCHD　A，@R1　；(A) =32H，(65H) =46H

（3）累加器A高、低半字节交换指令

SWAP　A　　；(A) 0~3←→ (A) 4~7

如：设（A）=36H

SWAP　A　　；（A）=63H

5、堆栈操作指令（2条）

PUSH direct ;先(SP)+1→SP，后 (direct) → ((SP))

POP direct ;先((SP)) → direct，后(SP)－1→SP

如：PUSH 0E0H ；实际是(SP)+1→SP,(A) → ((SP))

POP 05H ；实际是((SP)) →R5 , (SP)－1→SP

注意：其操作数为直接地址，不能用寄存器名。

POP ACC 正确

POP A 错误

例题

设SP=30H DPH=01H DPL=23H PSW=25H A=10H

```
PUSH   DPL    ；SP=31H  （31H）=DPL=23H
PUSH   DPH    ；SP=32H  （32H）=DPH=01H
PUSH   PSW    ；SP=33H  （33H）=PSW=25H
PUSH   ACC    ；SP=34H  （34H）=A   =10H
```

也可以写成：SP不会变动

```
MOV  31H，DPL
MOV  32H，DPH
MOV  33H，PSW
MOV  34H，A
```

三、算术运算指令

共24条指令，分成七个小类

不带进位加法指令（4条）

带进位加法指令　（4条）

带借位减法指令　（4条）

加1指令　（5条）

减1指令　（4条）

乘除法指令　（2条）

进制数调整指令　（1条）

主要是对8位无符号数进行算术运算。算术运算结果将影响PSW的有关位。

1、不带进位加法指令（4条）

ADD A，Rn ；(A) ←(A) + (Rn)

ADD A，direct ； (A) ←(A) + (direct)

ADD A，@Ri ； (A) ←(A) + ((Ri))

ADD A，#data ； (A) ←(A) +#data

注意：如果位3有进位，则半进位标志位AC置1，否则清零。

如果位7有进位，则进位标志位CY置1，否则清零。

如果两个数（看作有符号数时）相加溢出，则OV置1，否则清零。

如果两个同符号数相加的结果变号即溢出，则OV置1，否则清零。

两个同符号数相加的结果变号，则OV置1，否则清零。

例:（A）=0C3H，（R0）=0AAH

执行“ADD A，R0”的和为6DH，标志位

CY=1，OV=1，AC=0, P=1。

OV=C7⊕C6　对第6、第7位的进位位C7、C6异或。

例:（A）=0BAH，（R0）=88H

执行“ADD A，R0”的和为42H，标志位

CY=1，OV=1，AC=1，P=0。

OV=C7⊕C6　对第6、第7位的进位位C7、C6异或。

2、带进位加法指令（4条）

ADDC　A，Rn　　；　(A)←(A)+ (Rn) +CY

ADDC　A，direct　　；　(A)←(A) + (direct) +CY

ADDC　A，@Ri　　；　(A)←(A) + ((Ri)) +CY

ADDC　A，#data　　；　(A)←(A) +#data +CY

C为来自PSW状态寄存器中的进位位C。

把源操作数指出的内容和进位标志CY都同累加器A的内容相加，结果放于A中。多用于多字节数的加法。

3、带借位减法指令（4条）

SUBB　A，Rn　　；　(A)←(A) - CY - (Rn)

SUBB　A，direct　；　(A)←(A) - CY - (direct)

SUBB　A，@Ri　　；　(A)←(A) - CY - ((Ri))

SUBB　A，#data　；　(A)←(A) - CY - #data

注意：如果位3有借位，则半进位标志位AC置1，否则清零。

如果位7有借位，则进位标志位CY置1，否则清零。

如果两个数（看作有符号数时）相减溢出，则OV置1，否则清零。如果两个异号数相减的结果与被减数符号不同即溢出，OV置1，否则清零。

无不带借位减法指令，需要时，先执行一条CLR　C指令既可。

例:(A)=0D1H，(R0)=54H　(CY)=1

执行“SUBB　A，R0”的结果为 7CH，标志位CY= 0，OV= 1，

AC= 1，P=　1。

4、加1指令（5条）

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

说明：INC　A 影响P外，不影响PSW其它位(即标志位CY、AC、OV)。

例：
```
MOV A,#0FEH          ;P=1
INC  A               ;P=0
```

5、减1指令（4条）

```
DEC   A        ; (A) ←(A)-1
DEC   Rn       ; ( Rn ) ← ( Rn ) -1
DEC   direct   ; (direct ) ← (direct ) -1
DEC   @Ri      ; (( Ri )) ← (( Ri )) -1
DEC   DPTR     ; 无此指令
```

说明：DEC A 影响P外，不影响PSW其它位(即 标志位CY、AC、OV)。

例：
```
MOV A,#0FFH        ;P=0
DEC A              ;P=1
```

6、乘除法指令（2条，单字节四周期指令）

（1）乘法指令

MUL　AB　　；　(A) x (B)→B15~8，A7~0

为无符号乘法；若结果的B≠0 (乘积大于255) ，则OV=1，

若B=0，则OV=0；CY总是清零，AC不影响，P影响。

（2）除法指令

DIV　AB　　；　(A)/ (B)的商→A，余数→B

为无符号除法；若除数B=0 （除法无意义），则OV=1，

若B≠ 0，则OV=0；CY总是清零，AC不影响，P影响。

7、十进制数调整指令（1条）

DA　A　；调整累加器内容为BCD码(二进制编码的十进制)

说明：

（1）此指令跟在ADD或ADDC指令之后，将A中的和调整为BCD码，并且ADD或ADDC的两个操作数是BCD码，即两个BCD码相加必须经过 DA A指令后，才能得到正确的BCD码的结果。

（2）调整方法：

若(A0~3) >9或AC=1，则(A0~3) +6→ (A0~3) ;

若(A4~7)>9或CY=1，则(A4~7) +6→ (A4~7);

若(A4~7) ≥9且(A0~3) >9，则(A)＋66H → (A);

二—十进制编码BCD码

BCD码(Binary Coded Decimal)

二进制代码表示的十进制数。

十进制	BCD 码
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001
12	0001 0010
13	0001 0011
14	0001 0100
15	0001 0101

8421 BCD码

例：求十进制数876的BCD码

$[876]_{BCD}$ =1000 0111 0110

876= $36C_H$ =0011 0110 1100_B

BCD码运算

十进制调整：计算机实际按二进制法则计算，加入十进制调整操作，可计算BCD码。

例：计算BCD码 56+67=?

```
  0 1 0 1 0 1 1 0        [56]BCD
+ 0 1 1 0 0 1 1 1      + [67]BCD
  1 0 1 1 1 1 0 1        1 23
```

产生非BCD码和半进位

```
+ 0 1 1 0 0 1 1 0   +66H调整
1 0 0 1 0 0 0 1 1   带进位结果：
```

四、逻辑运算指令

包括与、或、异或、清除、求反、移位等操作。这类指令一般不影响标志位CY、AC和OV。

共24条指令，分成五个小类。

1、逻辑“与”指令（6条）

2、逻辑“或”指令（6条）

3、逻辑“异或”指令（6条）

4、累加器A清0与取反指令（2条）

5、移位指令（4条）

逻辑运算是按位进行的；

当需要只改变字节数据的某几位，而其余位不变时，不能使用直接传送方法，只能通过逻辑运算完成。

1、逻辑“与”，“或”，“异或”指令（18条）

ANL（ORL，XRL） A，Rn

ANL（ORL，XRL） A，direct

ANL（ORL，XRL） A，@Ri

ANL（ORL，XRL） A，#data

ANL（ORL，XRL） direct，A

ANL（ORL，XRL） direct，#data

说明：

（1）目的操作数只能是A或者direct；按位进行操作

（2）前4条指令仅影响标志位P；后两条不影响标志位；

（3）“与”运算常用于使某些位清0；“或”或运算常用于使某些位置1；“异或”运算常用于使某些位取反或不变。用1异或使对应位取反，用0异或使对应位不变，

例：试分析下列程序执行结果

```
MOV A,＃0FFH    ；(A)＝0FFH
ANL P1，＃00H   ；SFR中P1口清零
ORL P1，＃55H   ；P1口内容为55H
XRL P1，A       ；P1口内容为0AAH
```

4、累加器A清0与取反指令（2条）

（1）累加器A清 0 指令

CLR　A　　；　（A）←0

；　说明：只影响标志位P。

（2）累加器A取反指令（按位取反）

CPL　A　　；　(A)←(/A)，相当于0FFH - A→A

；　说明：不影响标志位。

如：　（A）=56H

CPL　A　；结果为0A9H

5、移位指令（4条）

累加器A循环左移

RL　A　　；

累加器A循环右移

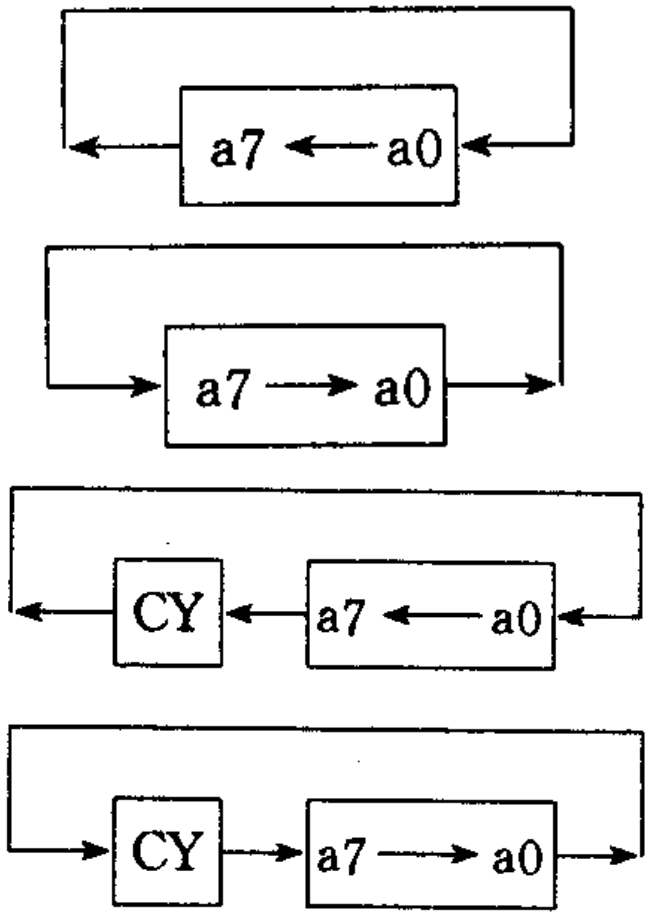
RR　A　　；

累加器A带进位位循环左移

RLC　A　　；

累加器A带进位位循环右移

RRC　A　　；

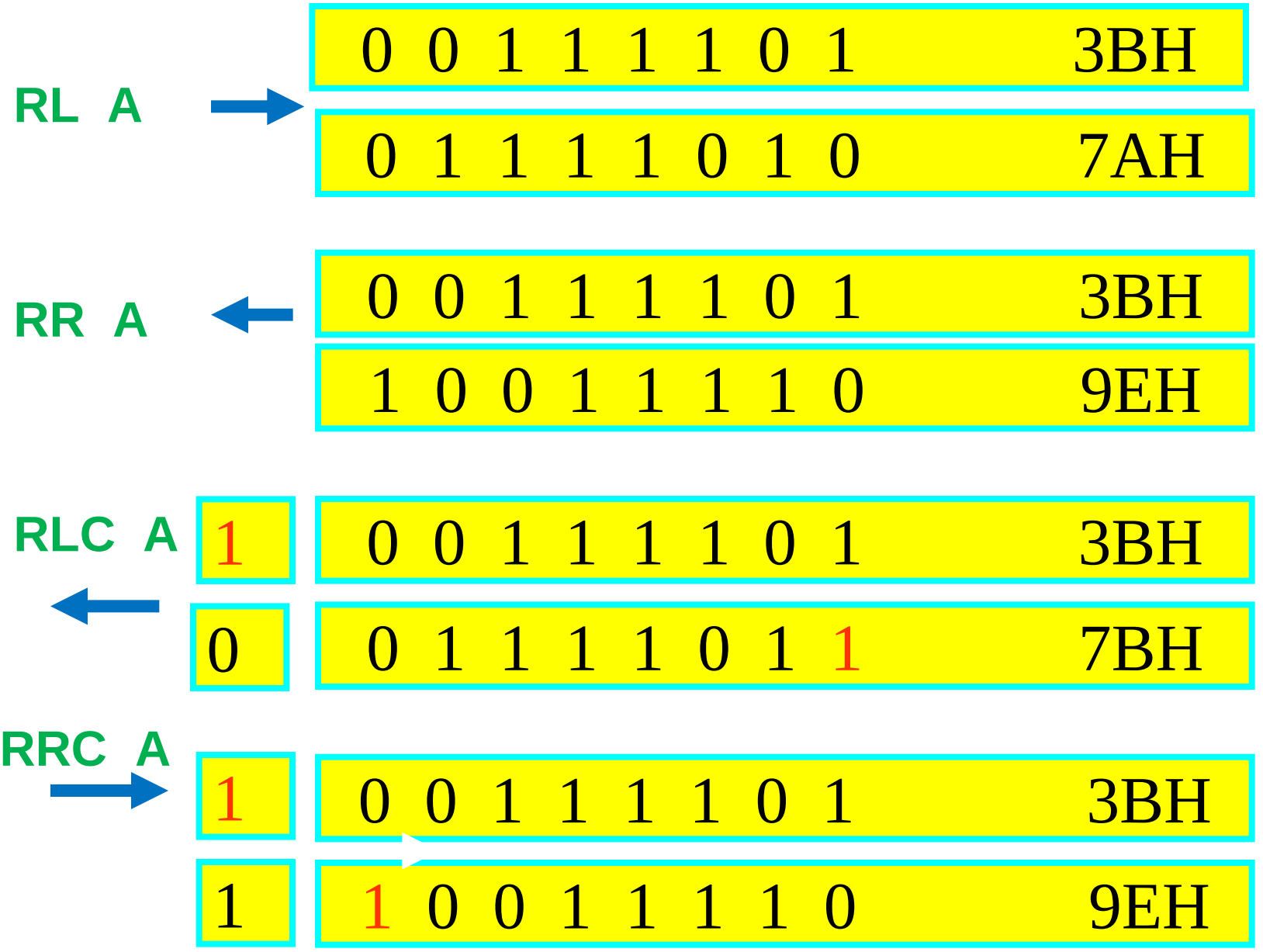


说明：

（1）各条指令每次只移动一位；

（2）左移一位相当于乘以2；右移一位相当于除以2；

（3）带进位位移动的影响标志位CY和P。


RL A
0 0 1 1 1 1 0 1 3BH
0 1 1 1 1 0 1 0 7AH
RR A
0 0 1 1 1 1 0 1 3BH
1 0 0 1 1 1 1 0 9EH
RLC A
1
0 0 1 1 1 1 0 1 3BH
0
0 1 1 1 1 0 1 1 7BH
RRC A
1
0 0 1 1 1 1 0 1 3BH
1
1 0 0 1 1 1 1 0 9EH

五、控制转移类指令

包括无条件转移、条件转移、子程序调用和返回指令等，共17条。只有比较转移指令影响标志。

1、无条件转移指令（4条）

2、条件转移指令（8条）

3、子程序调用和返回指令（4条）

4、空操作指令（1条）

1、无条件转移指令（4条）

（1）长转移指令

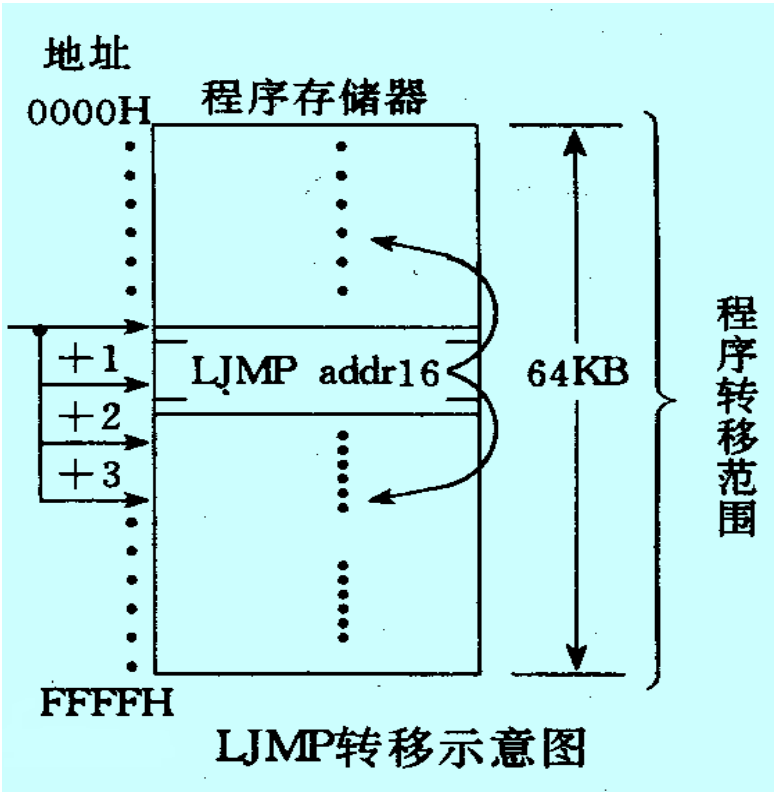
LJMP　addr16　；　(PC) ← addr16

说明：转移范围：64KB全部程序空间任何单元。

如：

LJMP　NEXT

(PC)



LJMP转移示意图

（2）绝对转移指令

A10A9A80 0 0 0 1
A7A6A5A4A3A2A1A0

AJMP addr11；先(PC) ←(PC)+2

；再(PC10~0) ← addr11, (PC15~11)不变

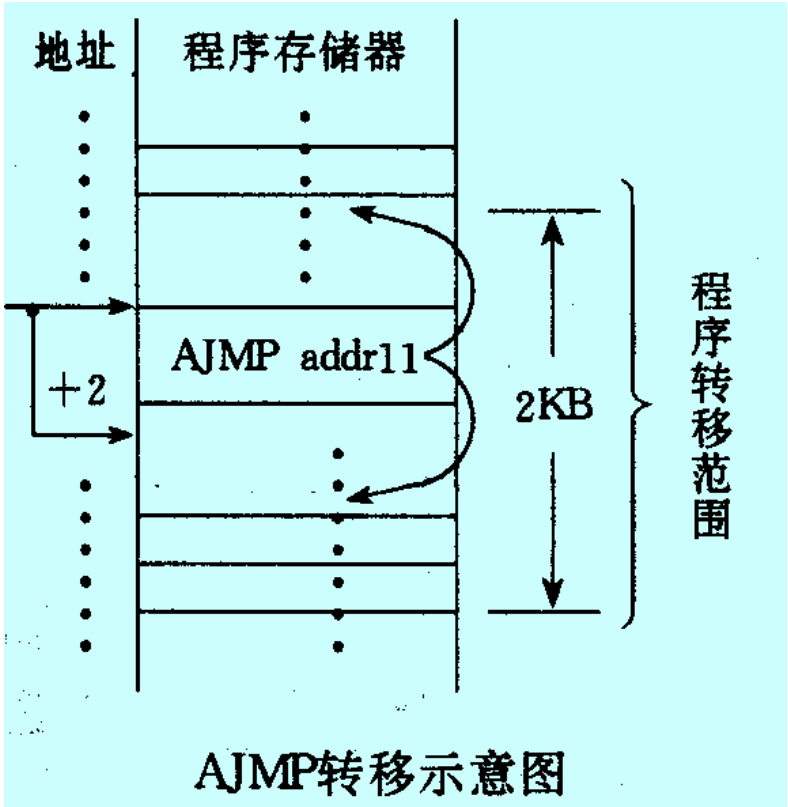
说明：两字节编码；转移范围：含有下一条指令首地址的同一个2KB范围，即 高5位地址相同；

如：

2800H: **AJMP SUB**

若SUB＝543H

则PC ＝?2D43H



AJMP转移示意图

2070H: AJMP 16AH **P65**

PC = 2072H = 0010 0000 0111 0010

目标地址＝16AH =0000 0001 0110 1010

新地址 ＝ 216AH＝0010 0001 0110 1010

A10	A9	A8	0	0	0	0	1
A7	A6	A5	A4	A3	A2	A1	A0

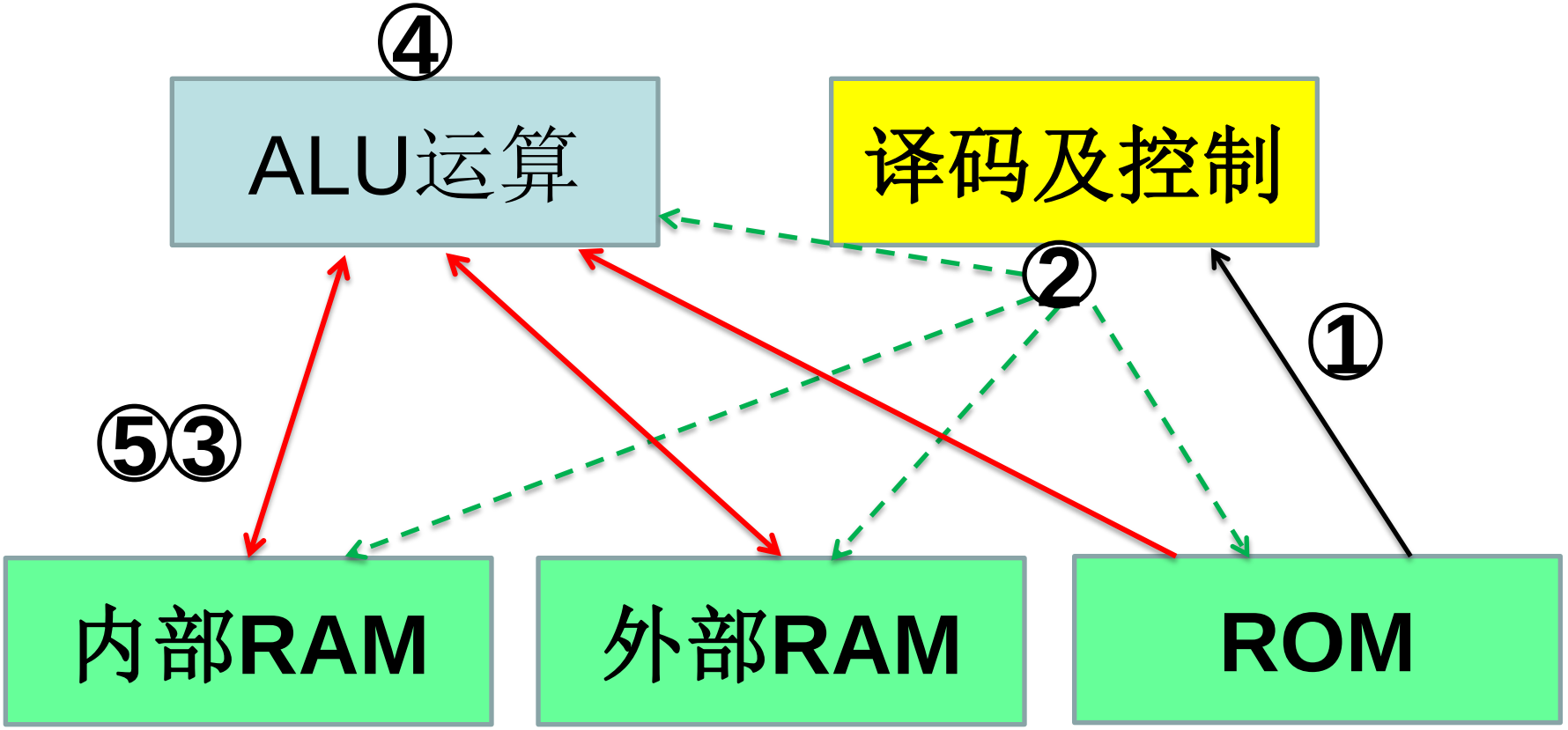
0	0	1	0	0	0	0	1
0	1	1	0	1	0	1	0

作业

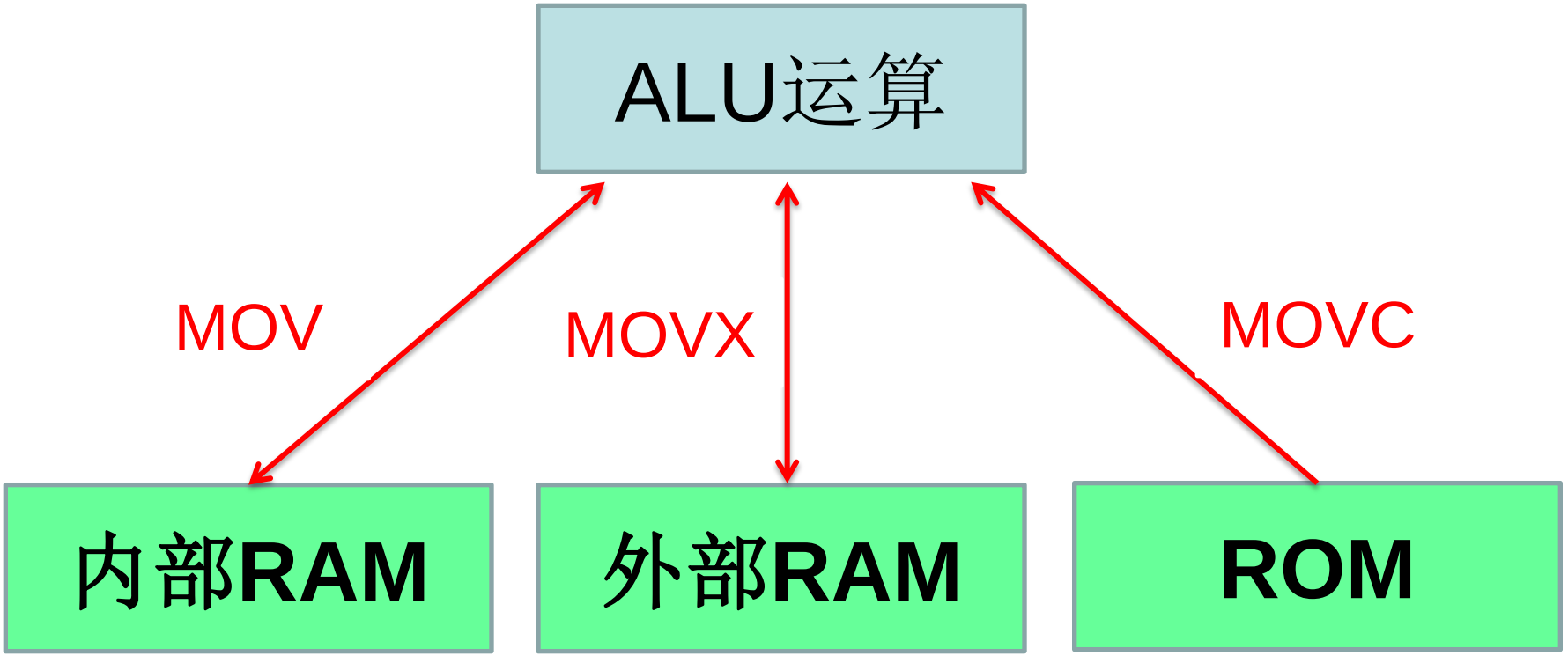
1、书74页——填空题的“6”

2、书75页——其它类型的“3”

单片机指令执行顺序



单片机取指指令



（3）短转移指令

SJMP　rel　　；先(PC)+2→(PC),后（PC）+rel→(PC)

说明：两字节编码,rel是补码形式存在，转移范围：-128~+127；对应rel值为：00H~7FH（0~+127）、80H~FFH（-128~-1）；

如：

HERE:　SJMP　HERE；无限循环执行本指令，REL= ?FE H

SJMP　$　　；与上条指令相同

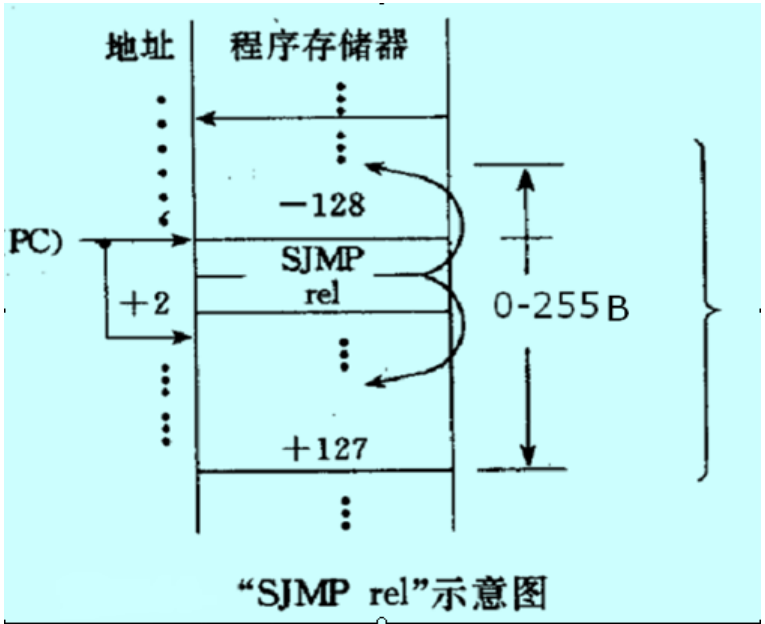
例：

1100H　SJMP　25H

PC=?

1100H　SJMP　E7H

PC=?

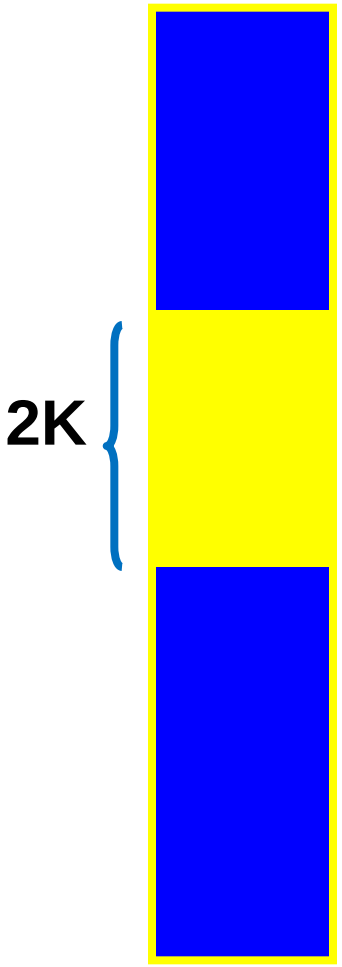


“SJMP rel”示意图

三种跳转对比

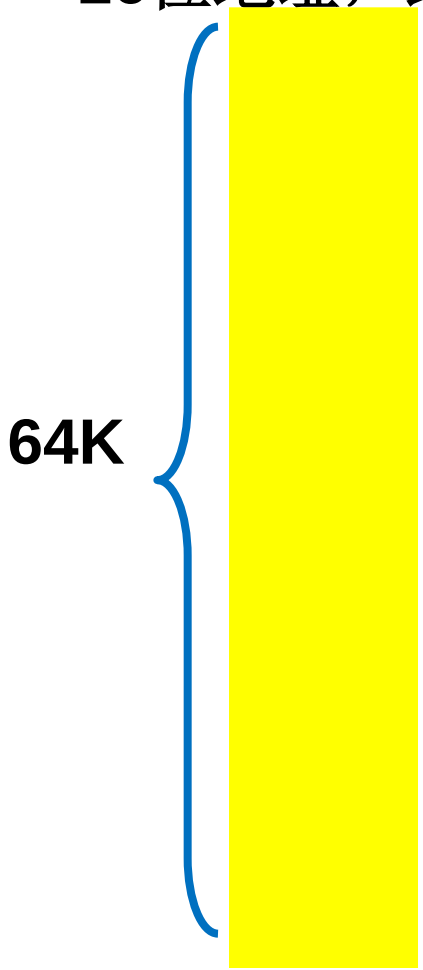
AJMP addr11

11位地址，跳转2K



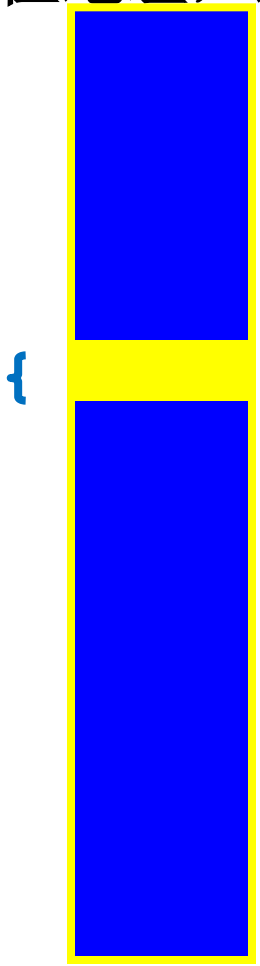
LJMP addr16

16位地址，跳转64K



SJMP rel

8位地址，跳转256



（4）间接转移指令

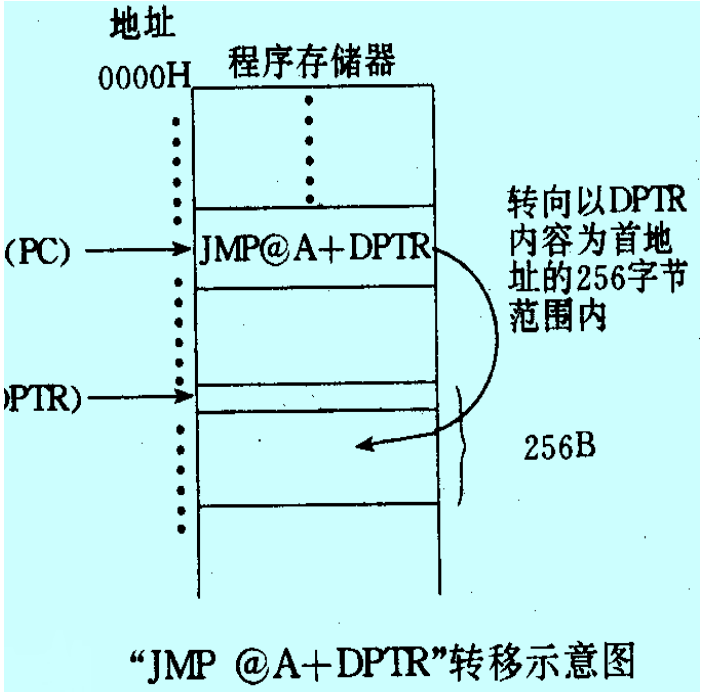
JMP　@A+DPTR　　；(A)+(DPTR)→(PC)

说明：具有多分枝转移功能，即散转功能，又叫散转指令；

转移范围：是以DPTR为首地址的256B。 DPTR作为基址寄存器，A作为变址寄存器。转移范围64KB.

例　根据累加器A中的命令键键值，设计命令键操作程序入口跳转表。

```
        CLR    C
        RLC    A
        MOV    DPTR，#JPTAB
        JMP    @A+DPTR
JPTAB:  AJMP   CCS0
        AJMP   CCS1
        AJMP   CCS2
        AJMP   CCS3
```



“JMP @A+DPTR”转移示意图

1、本程序为典型的散转程序，非常适合类似根据键盘值跳转。

2、A必须取偶数的原因是：AJMP为双字节指令；如果全部改为LJMP三字节指令，则A必须为0、3、6、9、12时，程序才能正确散转。

2、条件转移指令（8条）　　均为相对寻址方式。

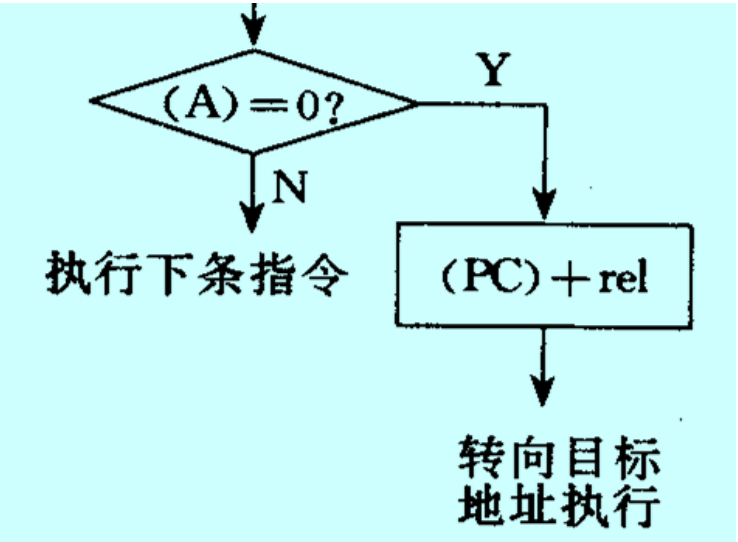
（1）累加器A为零（非零）转移指令（2条）

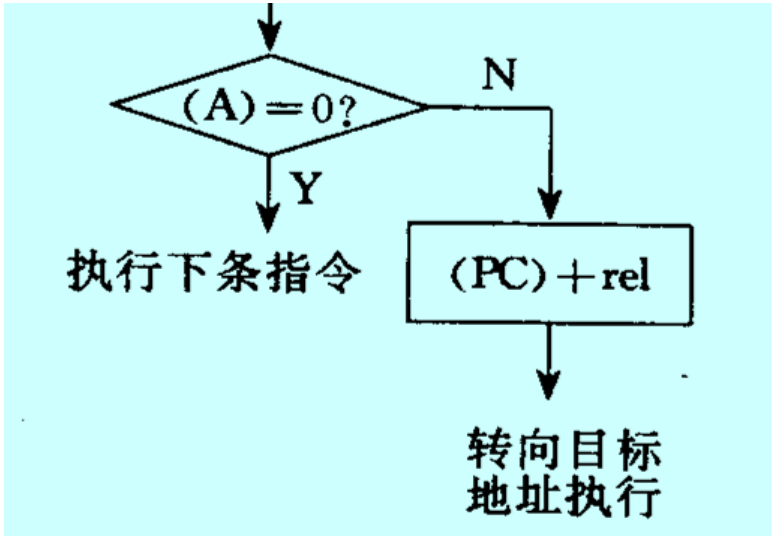
JZ　　rel　　；当A=0时，（PC）+2+rel→（PC）转移；

；当A≠0时，顺序执行（PC）+2→（PC）。

JNZ　　rel　　；当A≠0时，（PC）+2+rel→（PC）转移；

；当A=0时，顺序执行（PC）+2→（PC）。





例将外部RAM的一个数据块（首址为DATAl）传送到内部RAM（首址为DATA2），遇到传送的数据为零时停止。

```
START：MOV   R0，＃DATA2      ；置内部RAM数据指针
       MOV   DPTR，＃DATAl    ；置外部RAM数据指针
LOOPl：MOVX  A，@DPTR         ；外部RAM单元内容送A
       JZ    LOOP2            ；判是否为零，A为零则转移
       MOV   @R0，A           ；数据不为零，送内部RAM
       INC   R0               ；修改地址指针
       INC   DPTR
       SJMP  LOOP1            ；继续传送
LOOP2：RET                    ；结束传送，返回主程序
```

（2）比较转移指令（4条）

CJNE　A，direct，rel
CJNE　A，#data，rel
CJNE　Rn，#data，rel
CJNE　@Ri，#data，rel

均为三字节指令。一般形式为：

CJNE　X1（目的操作数），X2（源操作数），rel

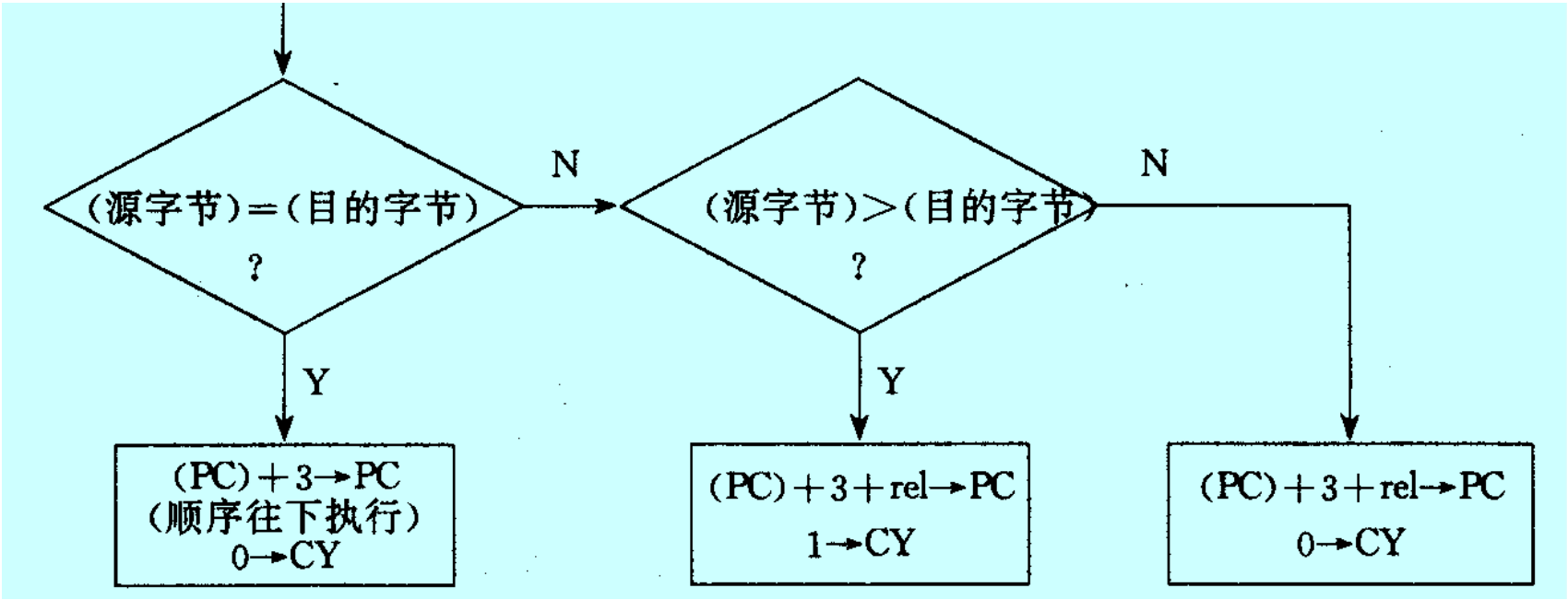
语句说明：**CJNE　X1，　X2，rel**

；若X1 > X2，则(PC)+3+rel→(PC)，且0→CY；

；若X1 < X2，则(PC) +3 +rel→(PC)，且1→CY；

；若X1 = X2，则顺序执行(PC)+3→(PC)，且0→CY。

实质是进行一次减法操作。（目的操作数－源操作数），相减结果若为负，则借位标志CY置1。不影响两个操作数。


(源字节)=(目的字节)
?
N
(源字节)>(目的字节)
?
N
Y
Y
(PC)+3→PC
(顺序往下执行)
0→CY
(PC)+3+rel→PC
1→CY
(PC)+3+rel→PC
0→CY

语句使用举例

```
        CJNE  A,direct,NOEQU  ;不相等跳到NOEQU
        ……                    ;相等继续执行
        LJMP  EXIT            ;执行完汇集
NOEQU:  JC    SMALL           ;判断大小
        ……                    ;A>direct时执行
        LJMP  EXIT            ;执行完汇集
SMALL:  ……                    ;A<direct时执行
        ……
EXIT:   ……                    ;汇集点
```

（3）减1非零条件转移指令（2条）

DJNZ　Rn，rel　；（Rn）-1→Rn；

；若(Rn)≠0, 则(PC)+2+rel →PC ；

；若 = 0, 则结束循环，顺序执行(PC)+2 →PC

DJNZ　direct，rel　；（ direct ）-1→ direct ；

；若(direct)≠0，则(PC)+3+rel →PC ；

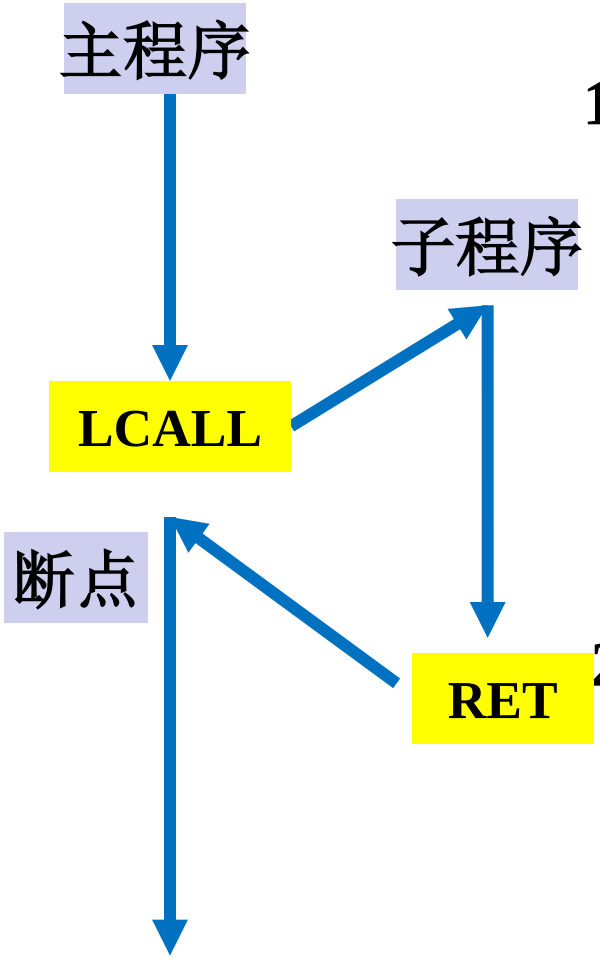
；若 = 0，则结束循环，顺序执行(PC)+3 →PC

说明：Rn、direct相当于控制循环的计数器.

例：将外部RAM地址1000H~10FFH的256个单元清零。

```
        MOV     R2, #0FFH
        MOV     DPTR,#1000H
        CLR     A
LOOP:   MOVX    @DPTR,A
        INC     DPTR
        DJNZ    R2,  LOOP
        MOVX    @DPTR,A
        RET
```

3、子程序调用和返回指令（4条）



1、主程序转向子程序，必须执行LCALL ACALL语句，此时CPU把子程序的入口地址给PC，程序转入子程序执行，同时CPU把主程序的断点地址自动入栈保护，以便子程序执行完返回主程序。

2、子程序返回主程序，必须执行RET 此时CPU把保护在栈顶的断点地址数据弹出给PC，程序返回主程序的断点。

3、PC是16bit，分高8bit/低8bit地址，而RAM是8bit，故2次入/出栈,注意顺序

（1）绝对调用指令

ACALL addr11 ； (PC)+2→(PC)

； (SP)+1→ (SP), (PC0~7)→((SP))

； (SP)+1→ (SP), (PC8~15)→((SP))

； addr11→(PC0~10), （ PC11~15不变）

AJMP addr11

转移范围：含有下一条指令首地址 高5位的同一个2KB范围。

A10	A9	A8	1	0	0	0	1
A7	A6	A5	A4	A3	A2	A1	A0

（2）长调用指令

LCALL　addr16　；　(PC)+3→(PC)

；　(SP) +1→(SP), (PC 0~7)→ ((SP))

；　(SP) +1→(SP), (PC 8~15)→ ((SP))

；　addr16→ (PC)　LJMP addr16

说明：　转移范围：整个程序存储空间，64KB范围。

如：LCALL　SUBPRO

例　设（SP）=5FH，符号地址“SUBRTN”指向5678H，执行指令

0123H　LCALL　SUBRTN

0126H　……..

5678H　SUBRTN：……

结果：　(60H)＝　26H　(61H)＝　01H　(PC)＝5678H

分析：26H→(60H)，[（SP）+1→(SP)，26H →（SP）]

01H→(61H)，[（SP）+1→(SP)，01H →（SP）]

5678H →(PC)，转去执行“SUBRTN”子程序

（3）返回指令（2条）

子程序返回指令

RET ；((SP)) →PC 8~15 , (SP) -1→SP

；((SP)) →PC 0~7 , (SP) -1→SP

例(SP) =63H,(62H)=07H,(63H)=30H 执行RET后，

(SP)= 61H (PC)= 3007H

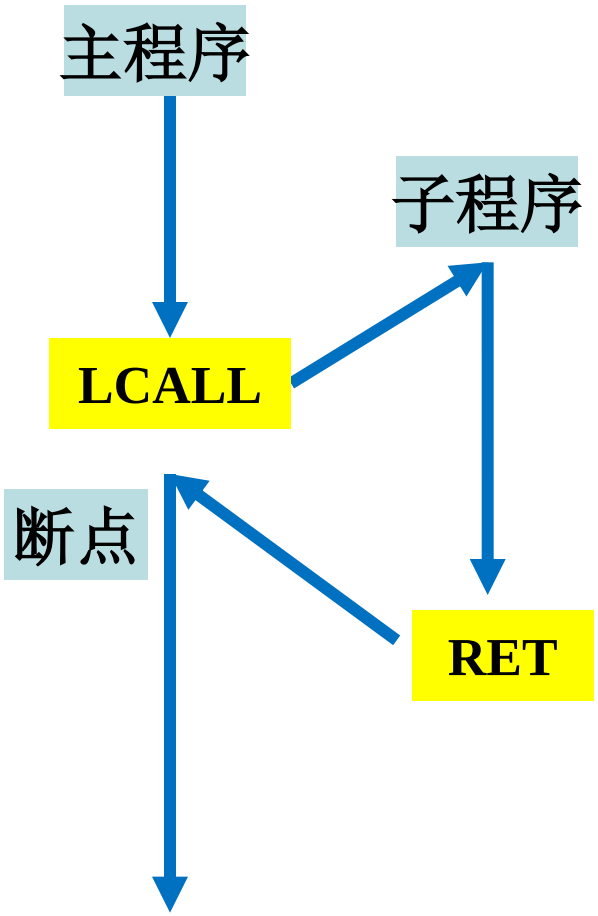
中断服务程序返回指令

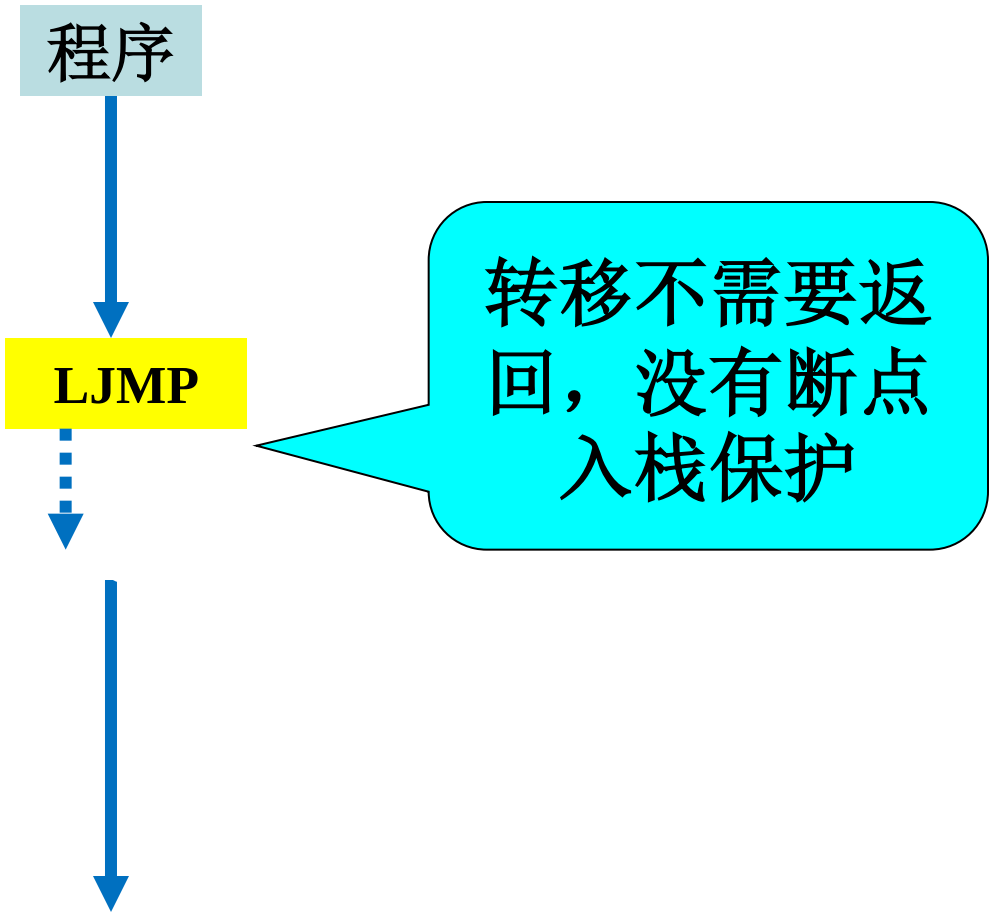
RETI ；((SP)) →PC 8~15 , (SP) -1→SP

；((SP)) →PC 0~7 , (SP) -1→SP

；开放中断逻辑

子程序调用和转移的对比





4、空操作指令　　1条

NOP　　；（PC）+1→（PC）

仍然取指令，并译码，但不做任何操作，只是为了耗费一个机器周期，作用：测试，延时

例：要求从P1口的P1.1输出10个方波，每个方波周期为10个机器周期：

```
        MOV  R1,#20
LOOP:   CPL  P1.1          ;  1T
        NOP                ;  1T
        NOP                ;  1T
        DJNZ R1,   LOOP    ;  2T
```

六、 位操作类指令

包括位变量传送、逻辑运算、控制转移等指令，共17条，分成4个小类。只有部分指令影响CY标志。

1、位数据传送指令（2条）

2、位修正指令（6条）

3、位逻辑运算指令（4条）

4、位条件转移类指令（5条）

进位位CY作为位累加器C，利用C完成位的传送和逻辑运算。位寻址区包括20H~2FH和专用寄存器区的11个专用寄存器

位地址的表示方法:

（1）直接用位地址　　如：D4H

（2）用特殊功能寄存器名加位数　如：PSW.4

（3）用SFR直接地址加位数　如：D0H.4

（4）用位名称　　如：RS1

（5）用伪指令bit定义的有名字的位地址

如：SUB.REG　bit　RS1,　　FLAGRUN　bit　02H

1．位数据传送指令　（2条）

MOV　C，bit；C←（bit）

MOV　bit，C；bit←C

只允许直接寻址位与位累加器C之间进行传送。

2．位状态修改指令　(6条)

CLR　C　；　C←0

CLR　bit　；　（bit）←0

CPL　C　；　C←C

CPL　bit　；　（bit）←（bit）

SETB　C　；　C←1

SETB　bit　；　（bit）←1

对进位标志位CY或直接寻址位进行清除、取反和置位操作。

如：CLR　TR0

SETB　06H

CPL　EA

3．位逻辑运算指令 （4条）

ANL C，bit；C←C∧（bit）

ANL C，/bit；C←C∧（bit）

ORL C，bit；C←C∨（bit）

ORL C，/bit；C←C∨（bit）

“/”表示对该位取反后再参与运算，但bit位的原值并不改变。

如：**ANL C，P1.0**

ORL C，/P1.2

4、位条件转移类指令（5条）

（1）判断C值转移指令

JC　rel　；（C）=1，则（PC）+2+rel→（PC）

；=0，则（PC）+2→（PC）顺序向下执行

JNC　rel；（C）=0，则（PC）+2+rel→（PC）

；=1，则（PC）+2→（PC）顺序向下执行

（2）判断位值转移指令

JB　bit，rel

；若（bit）=1，则（PC） +3+rel → PC

；=0，则（PC）+3→ （ PC ）顺序向下执行

JNB　bit，rel

；若（bit）=0，则（PC） +3+rel → PC

；=1，则（PC）+3→ （ PC ）顺序向下执行

（3）判断位值并清0转移指令

JBC　bit，rel

；若(bit)=1, 则(PC) +3 +rel→ PC, 0→ bit

；=0，则（PC）+3→ （ PC ）则顺序向下执行

帮助记忆指令助记符

MOV	;MOVE	内部数据RAM区读
MOVX	;MOVE EXTERNAL	外部数据RAM区读写
MOVC	;MOVE CODE	程序代码ROM区读
PUSH	;PUSH	内部数据入栈保护
POP	;POP	内部数据出栈保护
XCH	;EXCHANGE	内部数据交换
XCHD	;EXCHANGE DOWN?	内部数据低4位交换
SWAP	;SWOP	内部数据高低位交换
ADD	;ADD	内部数据与A累加
ADDC	;ADD WITH CARRY	内部数据与A及进位C累加
INC	;INCREASE	内部数据加1
DA	;DECIMAL ADJUST	内部数据10进制调整
SUBB	;SUBTRACT	内部数据与A及借位C相减
DEC	;DECREASE	内部数据减1

MUL	;MULTIPLY	内部数据A和B相乘
DIV	;DIVIDE	内部数据A和B相除
ANL	;AND LOGIC	逻辑与
ORL	;OR LOGIC	逻辑或
XRL	;EXCEPTION OR LOGIC	逻辑异或
CLR	;CLEAR	清零
CPL	;	求反
RR	;RECURRENCE RIGHT	循环右移
RL	;RECURRENCE LEFT	循环左移
AJMP	;ABSOLUTENESS JUMP	绝对跳转
LJMP	;LONG JUMP	长跳转
SJMP	;SHORT JUMP	短跳转
JZ	;JUMP IF ZERO	为零跳

JNZ	;JUMP IF NO ZERO	不为零跳
CJNE	;COMPARE JUMP IF NO EQUAL	比较不相等时跳转
DJNZ	;DECREASE JUMP IF NO ZERO	减1不为零时跳转
ACALL	;ABSOLUTENESS CALL	绝对调用
LCALL	;LONG CALL	长调用
RET	;RETURN	子程序返回
RETI	;RETURN INTERRUPT	中断子程序返回
NOP	;NO OPERATION	空操作
JC	;JUMP IF CARRY=1	跳转如果有进位
JNC	;JUMP IF CARRY=0	跳转如果无进位
JB	;JUMP IF BIT=1	跳转如果位为1
JNB	;JUMP IF BIT=0	跳转如果位为0
JBC	;JUMP IF BIT=1 THEN CLAER	跳转如果位为1,再清零

作业

1、书75页——其它类型的“2”

2、书75页——其它类型的“5”