



武汉纺织大学
WUHAN TEXTILE UNIVERSITY

单片机原理及应用

课程性质：必修（考试） 学时：48 学分：3
考核方式：闭卷考试



第3章 指令系统

- § 3. 1 汇编语言
- § 3. 2 寻址方式
- § 3. 3 89C51指令系统
- § 3. 4 思考题与习题



§ 3. 1 汇编语言

- § 3. 1. 1 指令和程序设计语言
- § 3. 1. 2 指令格式



§ 3.1.1 指令和程序设计语言

- **指令**：是CPU根据人的意图来执行某种操作的命令。
- **程序设计语言**：是实现人机交换信息的基本工具，分为机器语言、汇编语言和高级语言。
- **机器语言**：用二进制编码表示每条指令，是计算机能直接识别和执行的语言。
- **汇编语言**：是用助记符、符号和数字等来表示指令的程序设计语言。它与机器语言指令是一一对应的。



- 89C51汇编语言指令格式

[标号:][操作码 [目的操作数][, 源操作数][; 注释]

- 汇编语言指令对应的二进制代码格式

单字节指令

双字节指令

三字节指令



单字节指令

1、指令码中隐含着对某一种寄存器的操作

如：指令“**INC DPTR**”的指令代码格式为：

A3H= 1 0 1 0 0 0 1 1

2、由指令中的rrr三位的不同编码指定某一寄存器

如：指令“**MOV A, Rn**”的指令代码格式

为：1 1 1 0 1 r r r



双字节指令

- 用一个字节表示操作码，另一个字节表示操作数或操作数所在的地址。
- 格式为：

操作码

立即数或地址



三字节指令

- 一个字节操作码，两个字节操作数。
- 格式为：

操作码

立即数或地址

立即数或地址



§ 3.2

寻址方式

什么是寻址方式：如何找到存放操作数的地址，把操作数提取出来的方法。

- 1、7种寻址方式：
- 2、寻址空间及符号注释：



1、7种寻址方式

- 1) 寄存器寻址
- 2) 直接寻址
- 3) 立即数寻址
- 4) 寄存器间接寻址
- 5) 变址寻址
- 6) 相对寻址
- 7) 位寻址



1) 寄存器寻址

- **寄存器寻址：** 由指令指出寄存器组R0~R7中的某一个或其他寄存器(A,B,DPTR等)的内容作为操作数。

例如：MOV A, R0; (R0)→A

MOV P1, A; (A)→P1口

ADD A, R0; (A)+(R0)→A



2) 直接寻址

- **直接寻址方式：** 在指令中直接给出操作数所在存储单元的地址。指令中操作数部分是操作数所在地址。
- 直接寻址方式可访问片内**RAM**的 **128**个单元以及所有的**SFR**。对于**SFR**，既可以使用它们的地址，也可以使用它们的名字。
- **例如：** **MOV A, 3AH; (3AH) → A**
MOV A, P1; (P1口) → A
或： **MOV A, 90H; 90H是P1口的地址**



3) 立即数寻址

- 立即数寻址：指令操作码后面紧跟的是一字节或两字节操作数，用“#”号表示，以区别直接地址。

- 例如:

MOV A, 3AH; (3AH) → A

MOV A, #3AH; 3AH→A

MOV DPTR, #2000H ; 2000H→DPTR

; (DPH) =20H

; (DPL) =00H

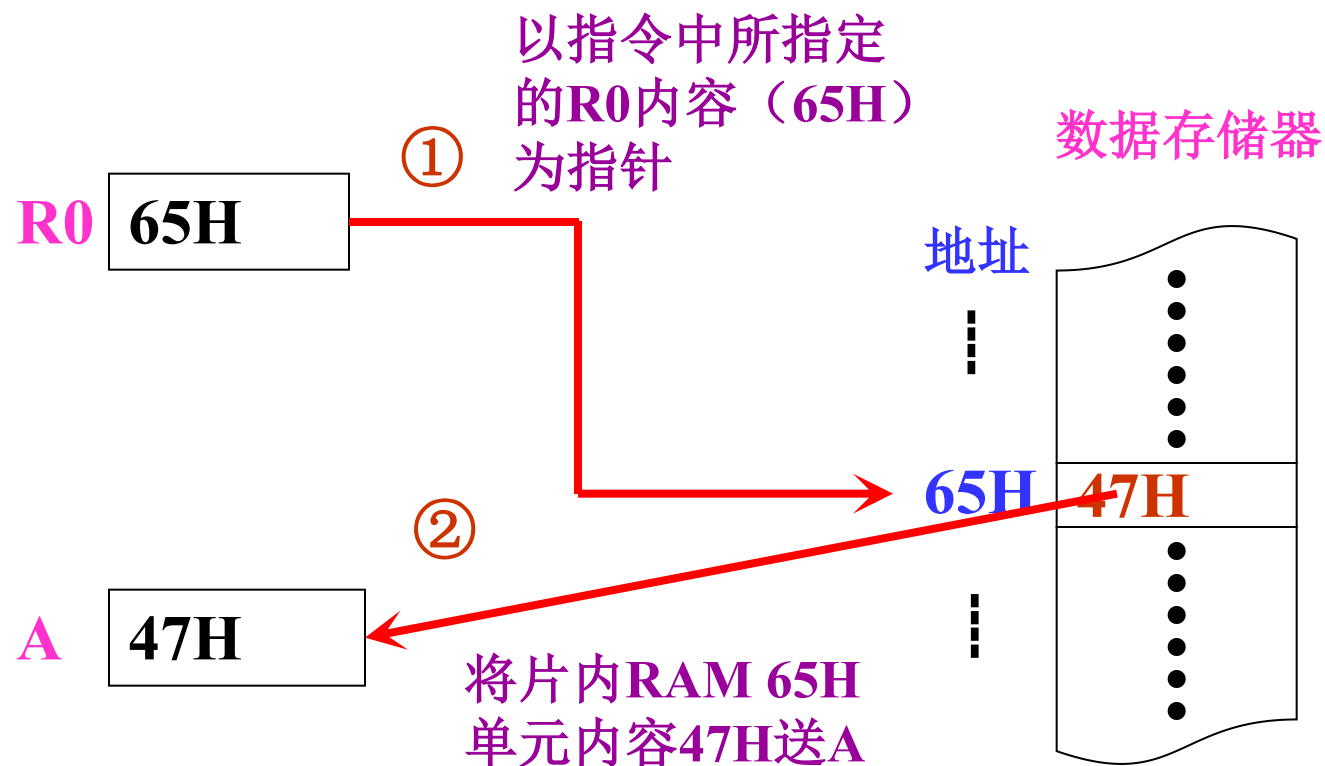


4) 寄存器间接寻址

- **寄存器间接寻址：**操作数的地址事先存放在某个寄存器中，寄存器间接寻址是把指定寄存器的内容作为地址，由该地址所指定的单元内容作为操作数。
- 89C51规定**R0**或**R1**为间接寻址寄存器，它可寻址内部地址RAM低位的**128B**单元内容。还可采用**DPTR**作为间接寻址寄存器，寻址外部数据存储器的**64KB**空间。
- 例如



- **例如：**将片内RAM 65H单元内容47H送A，可执行指令“MOV A, @R0”。其中R0内容为65H。如图3-3所示：



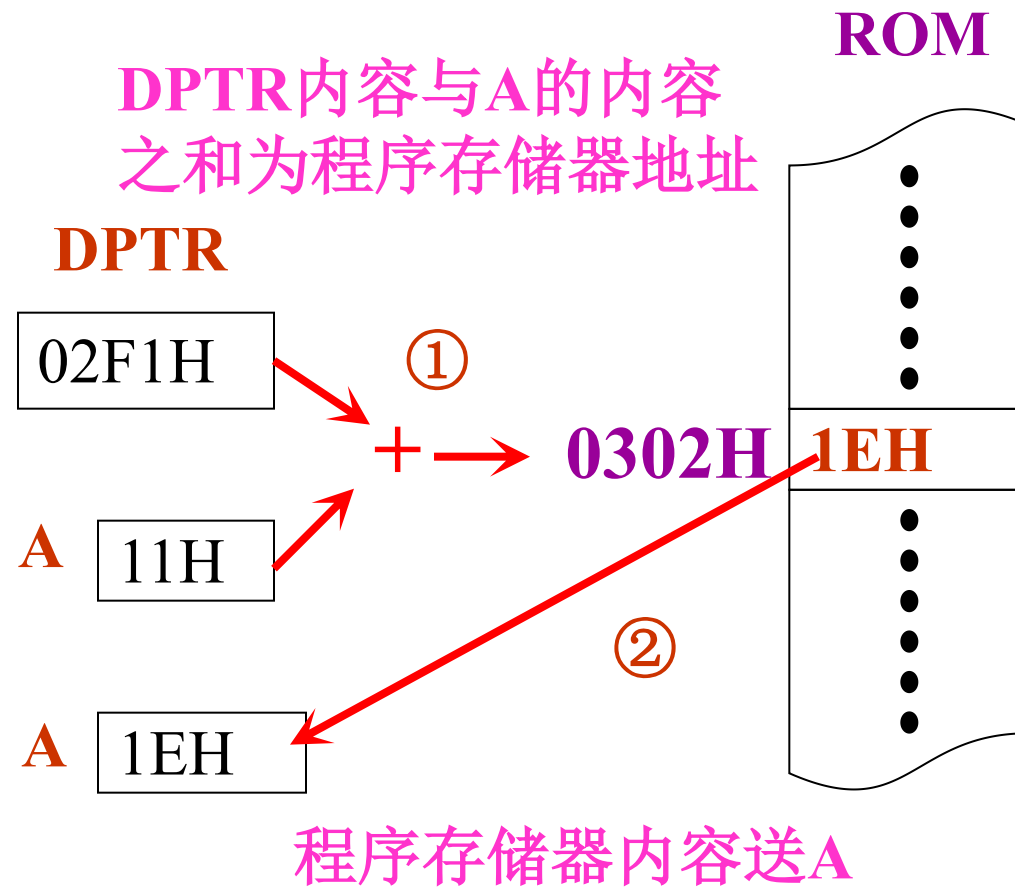


5) 变址寻址(基址寄存器+变址寄存器间接寻址)

- **变址寻址：**以某个寄存器的内容为基地址，在这个基地址的基础上加上地址偏移量形成真正的操作数地址。
- **89C51中采用DPTR或PC为基址寄存器，A的内容为地址偏移量。**
- **变址寻址只能访问程序存储器，访问范围为64KB。**
- **例如**



- 例如: **MOVC A, @A+DPTR; ((A)+(DPTR))→A**
- 如图3-4所示



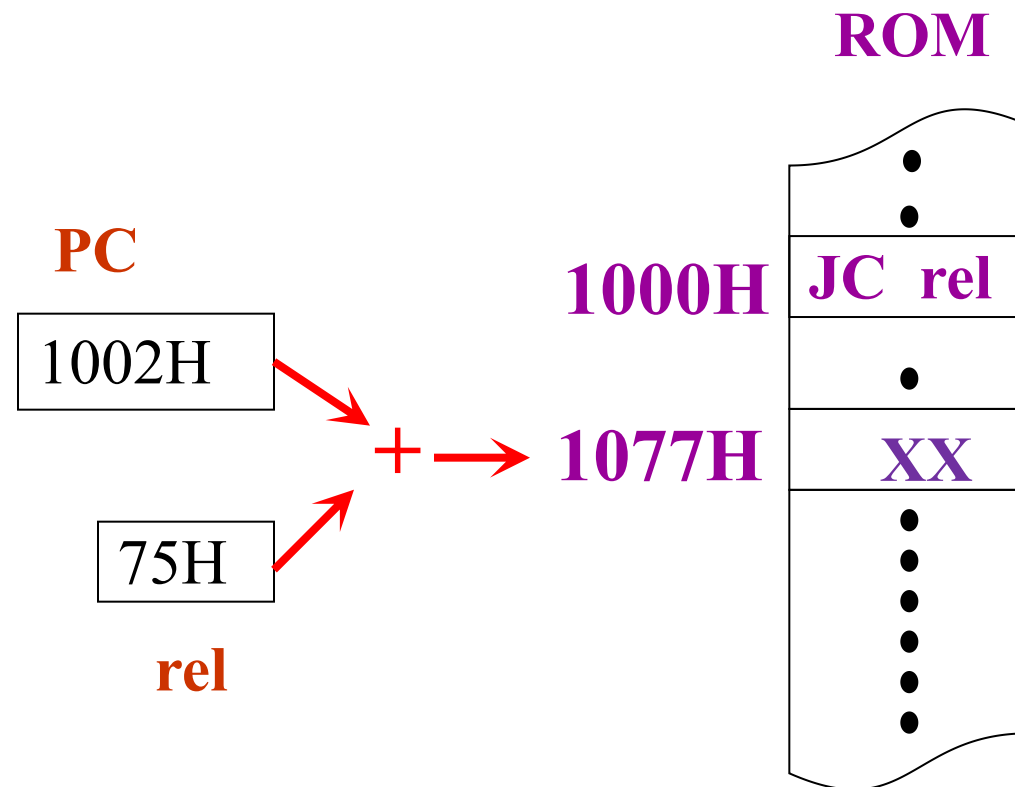


6) 相对寻址

- **相对寻址**：是以当前的PC值加上指令中规定的偏移量rel而形成实际的转移地址。
- **相对寻址只出现在相对转移指令中。**
- 当前的PC值是指执行完相对指令后的PC值；相对转移指令操作码所在地址称为源地址；转移后的地址称为目的地址。
- **目的地址=源地址+相对转移指令字节数+rel**
- **rel**为补码表示的有符号数。



- 例如：JC rel 其中rel=75H, CY=1





7) 位寻址

- **位寻址：**采用位寻址方式的指令的操作数是8位二进制数中的某一位，指令中给出的是位地址。位地址在指令中用bit表示。
- 例如：CLR bit;
- **位地址的两种表示方法：**直接使用位地址，如D3H；直接用寄存器名字加位数，如PSW.3。
- **位寻址区域：**片内RAM的20H-2FH的16个单元中的128位；字节地址能被8整除的SFR。



2、寻址空间及符号注释

- 1) 寻址空间：见表3-2。
- 2) 符号注释：



表3-2 操作数寻址方式和有关空间

寻址方式	寻址空间
立即数寻址	程序存储器ROM
直接寻址	片内RAM低128B、特殊功能寄存器
寄存器寻址	工作寄存器R0-R7、A、B、C、DPTR
寄存器间接寻址	片内RAM低128B、片外RAM
变址寻址	程序存储器（ @A+PC, @A+DPTR ）
相对寻址	程序存储器256B范围（PC+偏移量）
位寻址	片内RAM的20H-2FH字节地址、部分SFR



符号注释

- **R_n(n=0~7)**: 当前选中的工作寄存器组R0~R7。
- **R_i(I=0,1)**: 作为地址指针的两个工作寄存器R0, R1。
- **#data**: 8位立即数。
- **#data16**: 16位立即数。
- **direct**: 8位片内RAM单元（包括SFR）的直接地址。
- **addr11**: 11位目的地址，用于ACALL和AJMP指令中。
- **addr16**: 16位目的地址。用于LCALL和LJMP指令中。
- **rel**: 补码表示的8位地址偏移量。范围：-128~+127D。
- **bit**: 片内RAM或SFR的直接寻址位地址。
- **@**: 间接寄存器的符号。
- **/**: 位操作指令中对该位先取反再参与操作，不影响原值。
- **(×)**: ×中的内容。
- **((×))**: ×指出的地址单元中的内容。
- **→**: 指令操作流程方向。



§ 3.3 89C51指令系统

概述：

89C51指令系统由111条指令组成。其中，单字节指令49条，双字节指令45条，三字节指令17条。

从指令执行时间看，单周期指令64条，双周期45条，只有乘、除指令为4个周期。

89C51 指令系统可分为五大类



- [1] 数据传送指令：28条
- [2] 算术运算指令：24条
- [3] 逻辑运算及移位指令：25条
- [4] 控制转移指令：17条
- [5] 位操作指令（布尔操作）：17条



§ 3.3.1 数据传送指令

1. 以累加器A为目的的操作数的指令(4条, 即4种寻址方式)
2. 以寄存器Rn为目的的操作数的指令 (3条)
3. 以直接地址为目的的操作数的指令 (5条)
4. 以间接地址为目的的操作数的指令 (3条)
5. 十六位数据传送指令(1条)
6. 查表指令 (2条)
7. 累加器A与片外RAM传送指令(4条)
8. 栈操作指令 (二条)
9. 交换指令(5条)



1. 以累加器A为目的的操作数的指令(4条, 即4种寻址方式)

汇编指令格式	机器码格式	操作	注释
MOV A, Rn;	1110 1rrr	(Rn)→A	将工作寄存器Rn（即R0~R7）内容传送到累加器A中
MOV A, direct;	1110 0101 direct	(direct) →A	将直接寻址所得的片内RAM单元内容或特殊功能寄存器中的内容传送到累加器A中
MOV A,@Ri;	1110 011i	((Ri)) → A	将间接寻址（Ri为R0或R1）所得的片内RAM单元内容或特殊功能积存器中的内容传送到累加器A中
MOV A,#data	0111 0100 data	#data → A	将立即数传送到累加器A中



例：起始30H单元内容为11H，R1中为30H

MOV A, #30H ; (A) =30H

MOV A, 30H ; (A) =11H

MOV A, @R1 ; (A) =11H

MOV A, R1 ; (A) =30H



2. 以寄存器Rn为目的操作数的指令（3条）

汇编指令格式	机器码格式	操作	注释
MOV Rn, A;	1111 1rrr	(A)→ Rn	将累加器A中内容传送到工作寄存器Rn（即R0-R7）中
MOV Rn, direct;	1010 1rrr direct	(direct) → Rn	将直接寻址所得的片内RAM单元内容或特殊功能寄存器中的内容传送到工作寄存器Rn（即R0-R7）中
MOV Rn, #data;	0111 1rrr data	#data → Rn	将立即数传送到工作寄存器Rn（即R0-R7）中

这组指令的功能是把源操作数所指定的内容送到当前工作寄存器组R0-R7中的某个寄存器。源操作数有寄存器寻址、直接寻址和立即数寻址三种方式。



如：(A) =78H, (R5) =47H, (70H) =F2H, 执行指令：

- MOV R5, A ; (A) → R5, (R5) =78H
- MOV R5, 70H ; (70H) → R5, (R5) =F2H
- MOV R5, #A3H; A3H→R5, (R5) =A3H

注意：在89C51指令系统中没有“MOV Rn, Rn”
传送指令。



3. 以直接地址为目的操作数的指令（5条）

汇编指令格式	机器码格式	操作	注释
MOV direct ,A;	1111 0101 direct	(A) → direct	将A中内容传送到直接地址direct所指出的片内存储单元中。
MOV direct ,Rn ;	1000 1rrr direct	(Rn) → direct	将工作寄存器Rn（即R0-R7）中内容传送到直接地址direct所指出的片内存储单元中。
MOV direct1 , direct2 ;	1000 0101 direct2 direct1	(direct2) →direct 1	将直接地址源direct所指出的片内存储单元中内容传送到直接地址目的direct所指出的片内存储单元中
MOV direct ,@Ri ;	1000 011i direct	((Ri)) → direct	将间接寻址（Ri为R0或R1）所得的片内RAM单元内容传送到直接地址direct所指出的片内存储单元中
MOV direct ,#data;	0111 0101 direct data	#data → direct	将立即数传送到直接地址direct所指出的片内存储单元中

这组指令的功能是把源操作数所指定的内容送入由直接地址direct所指出的片内存储单元中。源操作数有寄存器寻址，直接寻址，寄存器间接寻址和立即寻址等方式。



4. 以间接地址为目的操作数的指令（3条）

汇编指令格式	机器码格式	操作	注释
MOV @Ri,A;	1111 011i	(A)→(Ri)	将累加器A中内容传送到间接寻址（Ri为R0或R1）所得的片内RAM单元中。
MOV @Ri, direct;	0101 011i direct	(direct) → (Ri)	将直接寻址得的片内RAM单元内容或特殊功能寄存器中内容传送到间接寻址（Ri为R0或R1）所得的片内RAM单元中。
MOV @Ri, #data;	0111 011i data	#data → (Ri)	将立即数data传送到间接寻址（Ri为R0或R1）所得的片内RAM单元中。

(Ri) 表示Ri中的内容为指定的RAM单元。



MOV指令在片内存储器的操作功能如图3-6示。

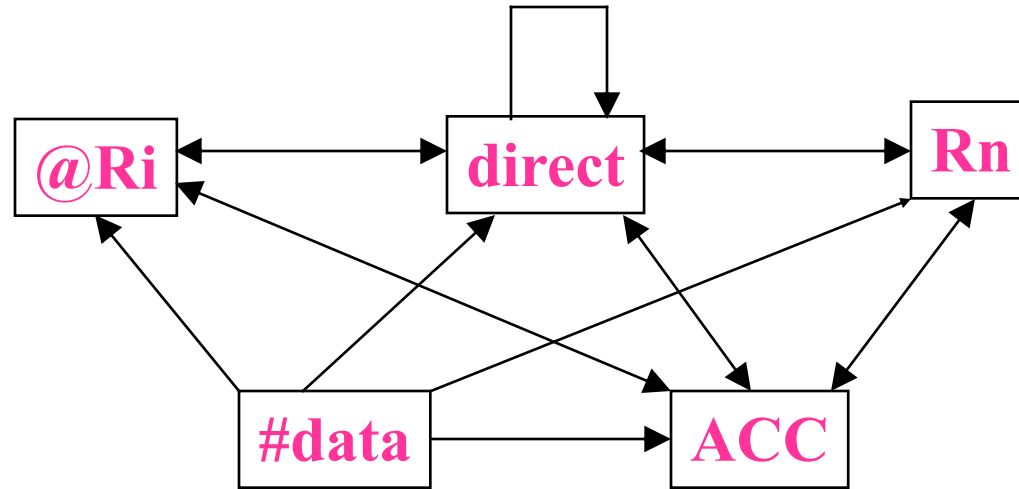


图3-6 传送指令在片内存储器的操作功能



5. 十六位数据传送指令(1条)

汇编指令格式	机器码格式	操作	注释
MOV DPTR, #data16;	1001 0000 data(h) data(l)	#data16→DPTR	将16位立即数传送到DPTR中

- 这条指令的功能是把16位常数送入DPTR。DPTR由DPH和DPL组成。

例: MOV DPTR #1000H;

	; (DPTR) =1000H,
	; (DPH) =10H,
	; (DPL) =00H



6. 查表指令（2条）

汇编指令格式	机器码格式	操作	注释
MOVC A, @A+DPTR;	1001 0011	$((A)+(DPTR)) \rightarrow A$	将程序存储器内容传送到A中(远程查表)
MOVC A, @A+PC;	1000 0011	先 $(PC)+1 \rightarrow PC$ $((A)+(PC)) \rightarrow A$	将程序存储器内容传送到A中(近程查表)

上述两条指令的操作过程如图3-7所示。

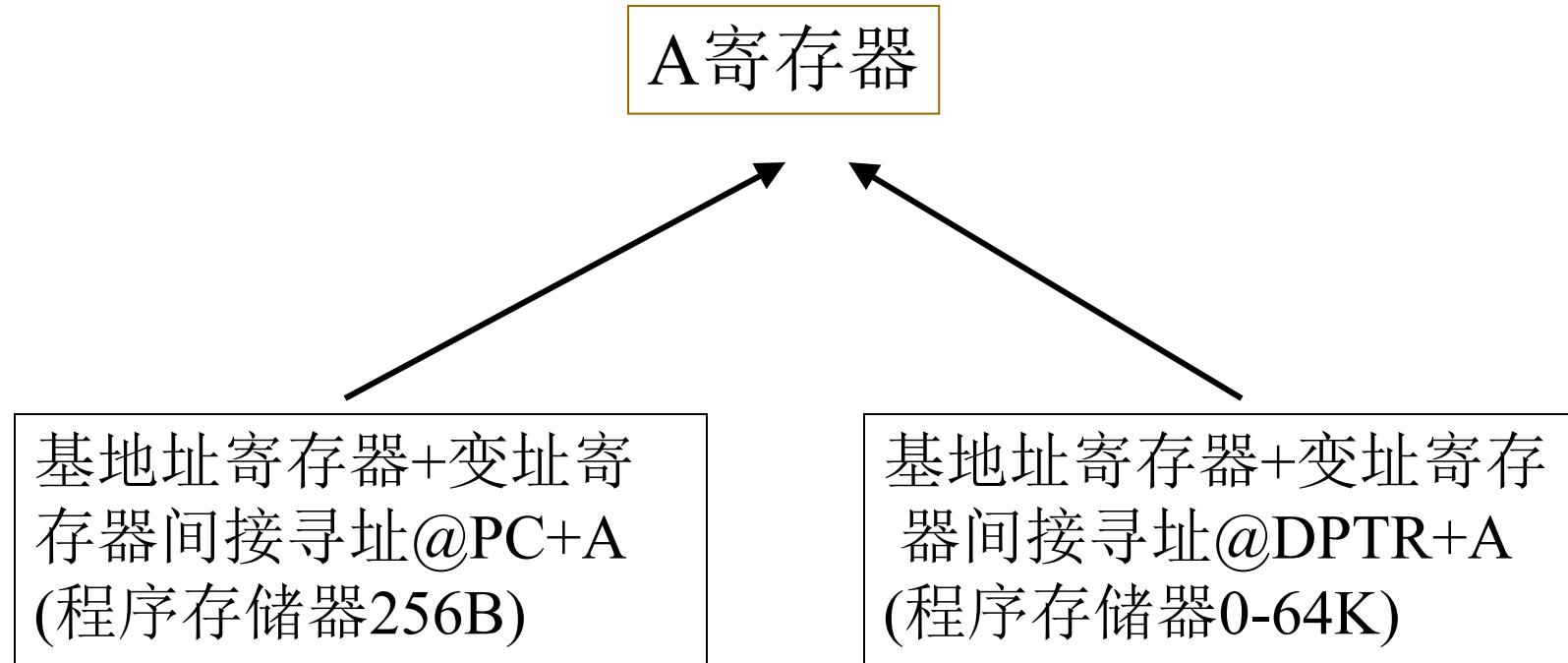


图3-7 程序存储器传送



例：若：初始 ROM 内容：

2000H	00H
2001H	01H
2002H	09H
2003H	04H
2004H	06H
2005H	08H
2006H	FFH
2007H	66H
2008H	45H
2009H	ABH
200AH	11H

执行下面程序段后结果如何？

```
MOV  DPTR, #2000H ; (DPTR) =2000H
MOV  A     , #0AH  ; (A) =0AH
MOVC  A, @A+DPTR  ; (A)=(200AH)=11H
```



例:若初始ROM内容为:

1010H	01H
1011H	02H
1012H	03H
1013H	04H

执行下面程序后的结果如何?

1000H 74 0D MOV A, #0DH ; (A) = 0DH
1002H 83 MOVC A, @A+PC; (PC) = 1002+1=1003H,
(A) = ((A) + (PC)) = (0DH+1003H)
= (1010H)
(A) = 01H
1003H F8 MOV R0, A ; (R0) = (A) = 01H

所以, 最后结果为: (A)=01 (R0)=01
(PC)=1004



7. 累加器A与片外RAM传送指令(4条)

在89C51指令系统中，CPU对片外RAM的访问只能用寄存器间接寻址的方式，且仅有四条指令：

汇编指令格式	机器码格式	操作
MOVX A, @Ri	1110 001i	$((Ri)) \rightarrow A$
MOVX A, @DPTR	1110 0000	$((DPTR)) \rightarrow A$
MOVX @Ri, A	1111 001i	$(A) \rightarrow (Ri)$
MOVX @DPTR, A	1111 0000	$(A) \rightarrow (DPTR)$



7. 累加器A与片外RAM传送指令(4条)

- 第2，4两条指令以DPTR为片外数据存储器16位地址指针，寻址范围达64KB。其功能是DPTR所指定的片外数据存储器与累加器A之间传送数据。
- 第1，3两条指令是用R0或R1作低8位地址指针，由P0口送出，寻址范围是256B。这两条指令完成以R0或R1为地址指针的片外数据存储器与累加器A之间的传送数据。
- 上述四条指令的操作如图3-8所示：

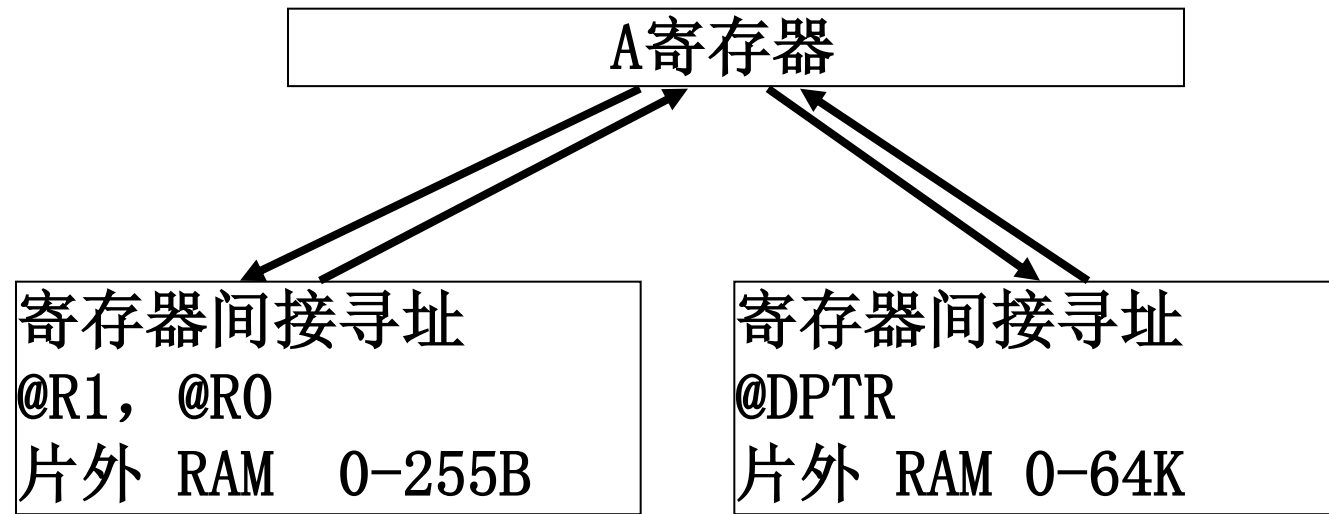


图 3-8 外部数据存储器传送操作



例：若片内RAM (30H)=01H，片外RAM (30H)=02H，执行下面程序段后的结果如何。

MOV R1, #30H	;	(R1)=30H
MOVBX A, @R1	;	(A)=02H
MOV R0, A	;	(R0)=(A)=02H
MOV A, @R1	;	(A)=((R1))=01H
MOV R2, A	;	(R2)=(A)=01H



8. 栈操作指令（2条）

汇编指令格式	机器码格式	操作	注释
PUSH direct	1100 0000 direct	先 $(SP)+1 \rightarrow SP$ 后 $(direct) \rightarrow (SP)$	将 direct 内容压入堆栈
POP direct	1101 0000 direct	先 $((SP)) \rightarrow direct$ 后 $(SP)-1 \rightarrow SP$	将堆栈内容弹出到 direct 单元中



例:

MOV SP, #18H	; (SP) =18H
MOV A, #30H	; (A) =30H
MOV DPTR, #1000H	; (DPTR) =1000H
PUSH A	; (SP) =19H (19H) =30H
PUSH DPH	; (SP) =1AH (1AH) =10H
PUSH DPL	; (SP) =1BH (1BH) =00H
POP DPL	; (DPL)=00H (SP)=1AH
POP DPH	; (DPH)=10H (SP)=19H
POP A	; (A)=30H (SP)=18H



9. 交换指令(4条)

(1) 字节变换指令

汇编指令格式	机器码格式	操作	注释
XCH A, Rn	1100 1rrr	(A)← → (Rn)	A的内容与Rn的内容互换
XCH A,direct	1100 0101 direct	(A)← → (direct)	A的内容与direct的内容互换
XCH A,@Ri	1100 011i	(A)← → ((Ri))	A的内容与((Ri))的内容互换

例:初始时: (A)=34H, (30H)=11H

XCH A, 30H ; (A)=11H, (30H)=34H

MOV R1, #30H ; (R1)=30H

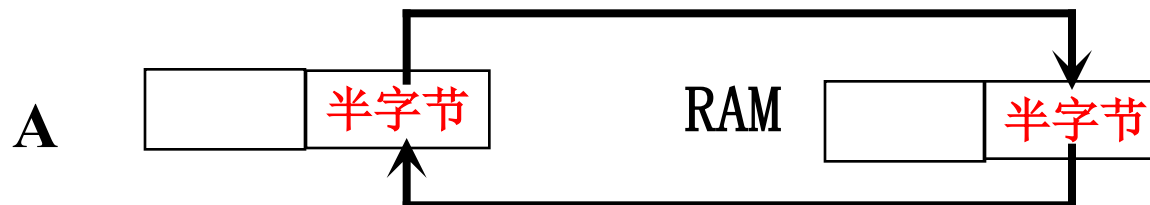
XCH A, @R1 ; (A)=34H, (30H)=11H



(2) 半字节交换指令

汇编指令格式	机器码格式	操作
XCHD A,@Ri	1101 011i	(A)0-3 $\leftarrow \rightarrow$ ((Ri))0-3
SWAP A	1100 0100	(A)0-3 $\leftarrow \rightarrow$ (A)4-7

第一条指令为低半字节交换指令。该指令将累加器A的低4位与R0或R1所指出的片内RAM单元的低4位数据相互交换，各自的高4位不变，其操作表示为：



第二条指令为A的低四位与高四位交换指令。该指令将累加器A的低4位与A的高4位数据相互交换。

如：(R1)=30H, (30H)=11H, (A)=34H

则：XCHD A, @R1 ; (A)=31H , (30H)=14H

SWAP A ; (A)=13H



§ 3.3.2 算术运算指令

1. 不带进位加法指令 (4条)
2. 带进位加法指令 (4条)
3. 带借位减法指令 (4条)
4. 乘法指令 (1条)
5. 除法指令 (1条)
6. 加1指令 (5条)
7. 减1指令 (4条)
8. 十进制调整指令 (1条)



1. 不带进位加法指令(4条)

汇编指令格式	机器码格式	操作	注释
ADD A, Rn;	0010 1rrr	$(A)+(Rn) \rightarrow A$	将工作寄存器内容和累加器A中的数相加，“和”存放于累加器A中
ADD A, direct;	0010 0101 direct	$(A)+(direct) \rightarrow A$	将内部RAM单元内容和累加器A中的数相加，“和”存放于累加器A中
ADD A, @Ri;	0010 011i	$(A)+((Ri)) \rightarrow A$	将间接寻址(Ri为R0或R1)所得的片内RAM单元中内容和累加器A中的数相加，“和”存放于累加器A中
ADD A, #data;	0010 0100 data	$(A)+\#data \rightarrow A$	将立即数的8位无符号二进制数和累加器A中的数相加，“和”存放于累加器A中



说明：上述指令的执行将影响标志位AC，Cy，OV，P。当“和”的第3位或第7位有进位时，分别将AC，CY标志位置1，否则为0。溢出标志位OV=C7 + C6（异或），该标志位只有带符号数运算时才有用。

例：若： (A) =78H ， (R0) =64H
执行ADD A, R0 后，结果及PSW=?

(A) :	78H=	0111	1000	B
+ (R0) :	64H=	0110	0100	B
<hr/>				
(A) :	DCH=	1101	1100	B

标志位:	CY=0,	AC=0,	OV=0,	P=1,
即:	PSW=01H			
结果:	(A) =DCH	(R0) =64H		



2. 带进位加法指令（4条）

汇编指令格式	机器码格式	操作	注释
ADDC A, Rn;	0011 1rrr	$(A) + C_Y + (Rn) \rightarrow A$	将工作寄存器内容与 C_Y 及累加器A中的数相加，“和”存放于累加器A中
ADDC A, direct;	0011 0101 direct	$(A) + C_Y + (\text{direct}) \rightarrow A$	将内部RAM单元内容与 C_Y 及累加器A中的数相加，“和”存放于累加器A中
ADDC A, @Ri;	0011 011i	$(A) + C_Y + ((Ri)) \rightarrow A$	将间接寻址(Ri 为R0或R1)所得的片内RAM单元中内容与 C_Y 及累加器A中的数相加，“和”存放于累加器A中
ADDC A, #data;	0011 0100 data	$(A) + C_Y + \#data \rightarrow A$	将立即数的8位无符号二进制数与 C_Y 及累加器A中的数相加，“和”存放于累加器A中



说明：本组指令的功能是同时把源操作数所指出的内容和进位标志位CY都加到累加器A中，结果存放到A中，其余的功能和上面的ADD指令相同。本组指令常用于多字节加法。

例3-2：设 (A) = 0C3H, (R0) = 0AAH, (CY) = 1。
执行指令 “ADDC A, R0” 后的结果及标志位如何？
解：

	(A) :	C3H=	1100	0011
+	(CY) :	1=	0000	0001
<hr/>				
			1100	0100
+	(R0) :	AAH=	1010	1010
<hr/>				
	(A) :	6EH=	0110	1110

标志位：CY=1, OV=1, AC=0,
结果：(A) = 6EH, (R0) = 0AAH。



例：编程，将(30H)，(31H)单元中的数与
(40H)，(41H)单元中的数相加，结果存于
(30H)，(31H)单元中。

解：

```
MOV    A, 30H
ADD    A, 40H
MOV    30H, A
MOV    A, 31H
ADDC   A, 41H
MOV    31H, A
```



3. 带借位减法指令（4条）

汇编指令格式	机器码格式	操作	注释
SUBB A, Rn;	1001 1rrr	$(A) - C_Y - (Rn) \rightarrow A$	将工作寄存器内容与 C_Y 及累加器A中的数相减，“差”存放于累加器A中
SUBB A, direct;	1001 0101 direct	$(A) - C_Y - (\text{direct}) \rightarrow A$	将内部RAM单元内容与 C_Y 及累加器A中的数相减，“差”存放于累加器A中
SUBB A, @Ri;	1001 011i	$(A) - C_Y - ((Ri)) \rightarrow A$	将间接寻址(Ri 为R0或R1)所得的片内RAM单元中内容与 C_Y 及累加器A中的数相减，“差”存放于累加器A中
SUBB A, #data;	1001 0100 data	$(A) - C_Y - \#data \rightarrow A$	将立即数的8位无符号二进制数与 C_Y 及累加器A中的数相减，“差”存放于累加器A中



说明： 这组指令的功能是从累加器A中减去源操作数所指出的数及进位位CY的值, 差保留在累加器A中。

由于89C51指令系统中没有不带借位的减法指令，如需要的话，可以在“SUBB”指令前用“CLR C”指令将Cy清0，这一点必须注意。



例3-3：设 $(A) = 0C9H$, $(R2) = 54H$, $Cy = 1$ 。

执行指令 “SUBB A, R2” 的结果如何？

解：

$$\begin{array}{r} (A) = 0C9H = 11001001 \text{ B} \\ -) \quad Cy = 1 = 00000001 \text{ B} \\ \hline 11001000 \text{ B} \\ -) \quad (R2) = 54H = 01010100 \text{ B} \\ \hline (A) = 74H = \textcolor{red}{0}1110100 \text{ B} \end{array}$$

结果为： $(A) = 74H$

标志位为： $Cy = 0$ $AC = 0$ $OV = 1$ $P = 0$



4. 乘法指令（1条）

汇编指令格式	机器码格式	操作
MUL AB ;	1010 0100	$(A) \times (B) \rightarrow B_{15-8}A_{7-0}$

该指令将累加器A和寄存器B中两个无符号数相乘，所得16位积的低字节存放在A中，高字节存放在B中。

指令若乘积大于0FFH，则OV置1，否则OV清0。Cy位总是被清0。

例3-4：（A）=4EH，（B）=5DH，执行指令“**MUL AB**”后结果如何？

解： 结果为：（B）=1CH，（A）=56H，
表示积：（BA）=1C56H，OV=1。



5. 除法指令（1条）

汇编指令格式	机器码格式	操作
DIV AB ;	1000 0100	(A)/(B)的商 → A, (A)/(B)的余数→ B

A中内容除以B中内容，整数商存于A中，余数存于B中。

该指令执行后，CY和OV均被清0。若原（B）=00H，则结果无法确定，用OV=1表示，CY仍为0。

例3-5：（A）=BFH，（B）=32H。执行指令“DIV AB”后：

结果为（A）=03H，（B）=29H；
标志位 CY=0，OV=0。



6. 加1指令（5条）

汇编指令格式	机器码格式	操作	注释
INC A ;	0000 0100	$(A)+1 \rightarrow A$	A中内容加1。
INC Rn;	0000 1rrr	$(Rn)+1 \rightarrow Rn$	Rn中内容加1。
INC direct;	0000 0101 direct	$(direct)+1 \rightarrow direct$	direct单元中内容加1。
INC @Ri;	0000 011i	$((Ri))+1 \rightarrow (Ri)$	Ri间接寻址所得的片内RAM 单元中内容加1。
INC DPTR;	1010 0011	$(DPTR)+1 \rightarrow DPTR$	DPTR中内容加1

该组指令的操作不影响PSW。若原单元内容为FFH，加1后溢出为00H，也不影响PSW标志。

例：比较指令“INC A”和“ADD A, #01H”的结果。

解：“INC A”指令只将A的内容加1，标志位没有变化。

而“ADD A, #01H”指令不仅将A的内容加1，还影响标志位变化。



例： 若： $(R1) = 30H$, $(30H) = 11H$

求执行下面指令后的结果。

INC @R1; $(30H) = 12H$

INC R1 ; $(R1) = 31H$

解： 结果为： $(30H) = 12H$, $(R1) = 31H$ 。

不影响PSW。



7. 减1指令(4条)

汇编指令格式	机器码格式	操作	注释
DEC A ;	0001 0100	$(A)-1 \rightarrow A$	A中内容减1。
DEC Rn;	0001 1rrr	$(Rn)-1 \rightarrow Rn$	Rn中内容减1。
DEC direct;	0001 0101 direct	$(direct)-1 \rightarrow direct$	direct单元中内容减1。
DEC @Ri;	0001 011i	$((Ri))-1 \rightarrow (Ri)$	Ri间接寻址所得的片内RAM 单元中内容减1。

该组指令的操作不影响PSW。



8. 十进制调整指令(1条)

汇编指令格式	机器码格式	操作
DA A;	1101 0100	若 $(A_{0-3}) > 9$ 或 $AC=1$, 则 $(A_{0-3}) + 6 \rightarrow A_{0-3}$ 同时, 若 $(A_{4-7}) > 9$ 或 $CY=1$, 则 $(A_{4-7}) + 6 \rightarrow A_{4-7}$

- 该指令调整累加器内容为**BCD码**。
- 这条指令跟在ADD或ADDC指令**后**, 将相加后存放在累加器中的结果进行**十进制调整**, 完成十进制加法运算功能。
- 必须注意, 本指令不能简单地把累加器A的16进制数变换成BCD码, 也不能用于十进制减法的调整。



例3-6：设累加器A内容为：01010110B（即为56的BCD码），寄存器R3的内容为01100111B（即67的BCD码），CY内容为1。求执行下列指令后的结果。

ADDC A, R3;

DA A;

解：先执行ADDC A, R3;

(A) :	0101 0110	BCD: 56
(R3) :	0110 0111	BCD: 67
(+) (CY) :	0000 0001	BCD: 01
<hr/>		
和 :	1011 1110	

即 (A) =1011 1110 且影响标志位 CY=0, AC=0;

再执行DA A; 因为A中的高四位值为11，大于9，低四位值为14，也大于9，所以内部调整自动进行加66H的操作：

	1011 1110	
调整+)	0110 0110	
<hr/>		
1	0010 0100	BCD: 124

即 (A) =0010 0100=24BCD, CY=1; AC=1。



§ 3. 3. 3

逻辑操作指令

逻辑操作指令包括：与、或、异或、清除、求反、移位等操作。该指令组全部操作数都是8位25条指令。

1. 简单操作指令（2条）
2. 移位指令（4条）
3. 逻辑“与”指令（6条）
4. 逻辑“或”指令（6条）
5. 逻辑“异或”指令（6条）

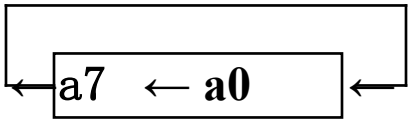
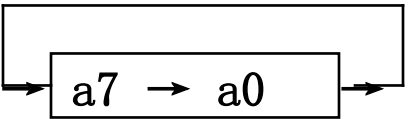
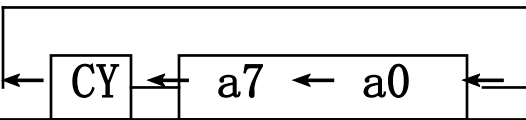
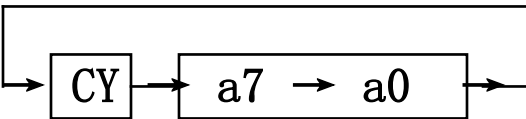


1. 简单操作指令（2条）

汇编指令格式	机器码格式	操作	注释
CLR A ;	1110 0100	$0 \rightarrow A$	累加器A清0指令，只影响标志位P。
CPL A ;	1111 0100	$(\overline{A}) \rightarrow A$	累加器A取反指令，不影响标志位P。



2. 移位指令（4条）

汇编指令格式	机器码格式	操作	注释
RL A ;	0010 0011		A中内容循环左移一位。
RR A ;	0000 0011		A中内容循环右移一位。
RLC A ;	0011 0011		A中内容连同进位位CY一起循环左移一位。
RRC A ;	0001 0011		A中内容连同进位位CY一起循环右移一位。

我们通常用RLC A指令将累加器A的内容做乘2运算。

例3-7：无符号8位二进制数 (A) = 10111101B = BDH, (CY) = 0。

将(A)乘2，执行指令“RLC A”后

结果：(A) = 0111 1010B = 7AH, (CY) = 1，而7AH正是BDH的2倍。



3. 逻辑“与”指令（6条）

汇编指令格式	机器码格式	操作
ANL A, Rn;	0101 1rrr	$(A) \wedge (Rn) \rightarrow A$
ANL A, direct;	0101 0101 direct	$(A) \wedge (\text{direct}) \rightarrow A$
ANL A, @Ri;	0101 011i	$(A) \wedge ((Ri)) \rightarrow A$
ANL A, #data;	0101 0100 data	$(A) \wedge \text{data} \rightarrow A$
ANL direct, A;	0101 0010 direct	$(\text{direct}) \wedge (A) \rightarrow \text{direct}$
ANL direct, #data;	0101 0011 direct data	$(\text{direct}) \wedge \#data \rightarrow \text{direct}$

- 这组指令中前四条指令是将累加器A的内容和操作数所指出的内容按位进行逻辑“与”，结果存放在A中。
- 后两条指令是将直接地址单元中的内容和操作数所指出的单元的内容按位进行逻辑“与”，结果存入直接地址单元中。若直接地址正好是I/O端口，则为“读——改——写”操作。



4. 逻辑“或”指令（6条）

汇编指令格式	机器码格式	操作
ORL A, Rn;	0100 1rrr	$(A) \vee (Rn) \rightarrow A$
ORL A, direct;	0100 0101 direct	$(A) \vee (\text{direct}) \rightarrow A$
ORL A, @Ri;	0100 011i	$(A) \vee ((Ri)) \rightarrow A$
ORL A, #data;	0100 0100 data	$(A) \vee \text{data} \rightarrow A$
ORL direct, A;	0100 0010 direct	$(\text{direct}) \vee (A) \rightarrow \text{direct}$
ORL direct, #data;	0100 0011 direct data	$(\text{direct}) \vee \#data \rightarrow \text{direct}$

- 这组指令的功能是将两个指定的操作数按位进行逻辑“或”，前四条指令的操作结果存放在累加器A中，后两条指令的操作结果存放在直接地址单元中。



5 . 逻辑“异或”指令（6条）

汇编指令格式	机器码格式	操作
XRL A, Rn;	0110 1rrr	$(A) \oplus (Rn) \rightarrow A$
XRL A, direct;	0110 0101 direct	$(A) \oplus (direct) \rightarrow A$
XRL A, @Ri;	0110 011i	$(A) \oplus ((Ri)) \rightarrow A$
XRL A, #data;	0110 0100 data	$(A) \oplus data \rightarrow A$
XRL direct, A;	0110 0010 direct	$(direct) \oplus (A) \rightarrow direct$
XRL direct, #data;	0110 0011 direct data	$(direct) \oplus \#data \rightarrow direct$

- 这组指令的功能是将两个指定的操作数按位进行“异或”，前四条指令的结果存放在累加器A中，后两条指令的操作结果存放在直接地址单元中
- 这类指令的操作均只影响标志位P。



§ 3.3.4 控制程序转移类指令

• 89C51单片机有丰富的转移类指令（17条），包括无条件转移、条件转移和调用指令及返回指令等。所有这些指令的目标地址都是在**64KB**程序存储器地址空间范围内。

1. 无条件转移指令（4条）
2. 空操作指令（1条）
3. 条件转移指令（8条）
4. 调用和返回指令（4条）



1.无条件转移指令（4条）

- 无条件转移指令是当程序执行到该指令时，程序无条件的转移到指令所提供的地址处执行。无条件转移指令有长转移、短转移、相对转移和间接转移4条指令。

(1) 长转移指令

指令格式	机器码	注 释
LJMP addr16;	0000 0010 $a_{15}-a_8$ a_7-a_0	addr16→PC

- 指令提供16位目标地址，将指令的第二、第三字节地址码分别装入PC的高8位和低8位中，程序无条件转向指定的目标地址去执行。由于直接提供16位目标地址，所以程序可转向64K程序存储器地址空间的任何单元。



例：若程序存储器中，指令 **LJMP**
LOOP 的首地址为**1000H**，其转向的
目标地址为**1234H**，执行下列程序：

```
ORG 1000H
1000H: LJMP LOOP
      ⋮
ORG 1234H
LOOP: MOV A, R2
      ⋮
```

此时指令**LJMP LOOP** 的指令码（机
器码）为：**02H 12H 34H**
即：

ROM	
1000H 1001H 1002H	02H
	12H
	34H
1234H	⋮
	EAH
	⋮



(2) 短转移指令(绝对转移)

AJMP addr11;	$a_{10} a_9 a_8$ 0 0001	先(PC)+2→PC
	$a_7 - a_0$	后addr11 → PC ₁₀₋₀ (PC ₁₅₋₁₁)不变

- 这条指令提供了11位地址，可在2K范围内无条件转移到由a10-a0所指出的地址单元中。
- 因为指令只提供低11位地址，高5位为原PC₁₁₋₁₅位值，因此，转移的目标地址必须在AJMP指令后面指令的第一个字节开始的同一2K字节范围内。



例: **ORG 0500H**
0500H: AJMP 0703H

 ; 0703H=0000 0**111** 0000 0011 B
 ;Addr11= **111 0000 0011** B
 ;指令地址PC=0500H
 ;即: PC =0000 0101 0000 0000B

该指令为二字节指令在执行时先PC+2, (PC=0502H=0000 0101 0000 0010B, PC的高五位为: **0000 0B**,与0703H的高五位相同, 在同一个2k范围内.) 然后将Addr11送PC的低11位: PC= **0000 0 111 0000 0011** B.

指令的机器码: $a_{10} a_9 a_8 0 \quad 0001$ =1110 0001= E1H
 $a_7 -a_0$ =0000 0011= 03H



例: **ORG 0700H**
0700H: AJMP 0832H

; 0832H=0000 1000 0011 0010 B

;Addr11= 000 0011 0010 B

;指令地址PC=0700H

;即: PC =0000 0111 0000 0000B

该指令为二字节指令在执行时先PC+2, (PC=0702H=0000 0111 0000 0010B,PC的高五位为: 0000 0B,于0832H的高五位不同,不在同一个2k范围内.) 该指令跨越2k的范围, 错误。



(3)相对转移指令(短转移)

SJMP rel;	1000 0000 rel	先(PC)+2→PC 后(PC)+rel→PC
------------------	--------------------------------	--

• 指令的操作数是相对地址，rel是一个带符号的偏移字节数（补码表示），其范围为-128~+127，负数表示反向转移，正数表示正向转移。该指令为二字节，执行时先将PC内容加2，再加相对地址，就得到了转目标地址。

例：

1000H: SJMP 55H;

其转移目的地址=?

解： ∵ rel=55H=0101 0101B，为正数。

∴ 目的 PC=1000H+2+rel

=1002H+55H

=1057H



例：

1000H SJMP F6H;

其转移目的地址=?

解： ∵ rel=F6H=1111 0110B为负数，表示向低地址方向转移。

∴ 目的 PC=1000H+2+rel=1002H+F6H 不能直接相加，

要先计算出F6H的真值。即：

F6H=1111 0110B取反加1后为： 0000 1010B=0AH

∴ 目的 PC=1002H+F6H

=1002H-0AH (+FFF6H)

=0FF8H



例：

指令码	程序	
	ORG 1000H	
1000H:E4H	CLR A	
1001H:78H 30H	MOV R0, #30H	
1003H:C3H	CLR C	
1004H:36H	LOOP: ADDC A, @R0	
1005H:F6H	MOV @R0, A	
1006H:08H	INC R0	
1007H:02H 10H 04H	LJMP LOOP	
(1007H: 01H 04H)	(AJMP LOOP)	
(1007H:80H FBH)	(SJMP LOOP)	

AJMP LOOP 指令的机器码：
Addr11=000 0000 0100
0000 0001=01H
0000 0100=04H

SJMP LOOP 指令的机器码：80H rel
rel=目的地址-(源地址+2)
=1004H-(1007H+2)=FBH



(4)间接转移指令(散转)

JMP @A+DPTR; 0111 0011 (A)+(DPTR)→PC

- 该指令的转移地址由数据指针DPTR的16位数和累加器A的8位数作**无符号数**相加形成，并直接送入。指令执行过程对DPTR、A和标志位均无影响。这条指令可代替众多的判别跳转指令，具有散转功能。



例3-11：根据累加器A中命令键键值，设计命令键操作程序入口跳转表：

CLR	C	； 清进位
RLC	A	； 键值乘2
MOV	DPTR, #JPTAB	； 指向命令键跳转表首址
JMP	@A+DPTR	； 散转入命令键入口
JPTAB:	AJMP CCS0	； 双字节指令
	AJMP CCS1	
	AJMP CCS2	

- 从程序中看出，当 (A) = 00H 时，散转到 CCS0；当 (A) = 01H 时，散转到 CCS1；……。由于 AJMP 是双字节指令，散转前的键值应乘2。



2.空操作指令（1条）

NOP ; 0000 0000 (PC)+1→PC

- 这是一条单字节指令，除PC加1外，不影响其它寄存器和标志位。“NOP”指令常用于产生一个机器周期的延迟。



3.条件转移指令（8条）

(1) 判零转移指令

汇编指令格式	机器码格式	操作
JZ rel ;	0110 0000 rel	先(PC)+2→PC 若(A)=0, 则(PC)+rel→PC; 否则: (A)≠0, 程序顺序执行
JNZ rel ;	0111 0000 rel	先(PC)+2→PC 若(A)≠0, 则(PC)+rel→PC; 否则: (A)=0, 程序顺序执行

•**JZ**和**JNZ**指令分别对累加器A的内容为**全零**和**不为零**进行检测并转移，当不满足各自的条件时，程序继续往下执行，当各自的条件满足时，则程序转向指定的目标地址。其目标地址是以下一条指令第一个字节的地址为基础加上指令的第二各字节中的相对偏移量。相对偏移量为一个带符号的8位数，偏移范围为-128—+127字节，在指令汇编和手工汇编时被确定，它是目标地址于下条指令地址之差。本指令不改变累加器A内容和影响任何标志位。



(2) 比较转移指令（4条）

CJNE（目的字节），（源字节），rel；三字节指令

它的功能是对指定的目的字节和源字节进行比较，先 $(PC) + 3 \rightarrow PC$ ：

- 若（目的字节） \neq （源字节），则转移，PC的当前值 $(PC) + rel \rightarrow PC$ 。
 - 若（目的字节） $>$ （源字节），则清0进位标志位Cy；
 - 若（目的字节） $<$ （源字节），则置位进位标志位Cy；
- 若（目的字节） $=$ （源字节），则往下执行。
- 该组指令执行后不影响任何操作数。



汇编指令格式	机器码格式	操作
CJNE A, direct, rel ;	1011 0101 direct rel	先 $(PC)+3 \rightarrow PC$ 若 $(A) \neq (direct)$, 则 $(PC)+rel \rightarrow PC$; 若 $(A) > (direct)$, $Cy=0$, 若 $(A) < (direct)$, $Cy=1$, 否则: $(A)=(direct)$, 程序顺序执行
CJNE A, #data, rel ;	1011 0100 data rel	先 $(PC)+3 \rightarrow PC$ 若 $(A) \neq data$, 则 $(PC)+rel \rightarrow PC$; 若 $(A) > data$, $Cy=0$, 若 $(A) < data$, $Cy=1$, 否则: $(A)=data$, 程序顺序执行



汇编指令格式	机器码格式	操作
CJNE Rn, #data, rel;	1011 0rrr data rel	先 $(PC)+3 \rightarrow PC$ 若 $(Rn) \neq data$, 则 $(PC)+rel \rightarrow PC$; 若 $(Rn) > data$ Cy=0, 若 $(Rn) < data$, Cy=1, 否则: $(Rn)=(data)$, 程序顺序执行
CJNE @Ri, #data, rel;	1011 011i data rel	先 $(PC)+3 \rightarrow PC$ 若 $((Ri)) \neq data$, 则 $(PC)+rel \rightarrow PC$; 若 $((Ri)) > data$, Cy=0, 若 $((Ri)) < data$, Cy=1, 否则: $((Ri))=data$, 程序顺序执行



(3) 循环转移指令（2条）

汇编指令格式	机器码格式	操作
DJNZ Rn, rel;	1101 1rrr rel	先 $(PC)+2 \rightarrow PC$, $(Rn)-1 \rightarrow Rn$ 若 $(Rn) \neq 0$, 则 $(PC)+rel \rightarrow PC$; 否则: $(Rn)=0$, 程序顺序执行
DJNZ direct, rel;	1101 0101 direct rel	先 $(PC)+3 \rightarrow PC$, $(direct)-1 \rightarrow direct$ 若 $(direct) \neq 0$, 则 $(PC)+rel \rightarrow PC$; 否则: $(direct)=0$, 程序顺序执行

- 程序每执行一次本指令，将第一操作数的字节变量减1，并判字节变量是否为零，若不为0，则转移到目标地址，继续执行循环程序段；若为0，则结束循环程序段的执行，程序往下执行。
- 其中，rel为相对于DJNZ指令的下一条指令的第一个字节相对偏移量，用一个带符号的8位数表示。所以，循环转移的目标地址应为DJNZ指令的下条指令地址和偏移量之和。



4. 调用和返回指令(4条)

在程序设计中，有时因操作要求，需反复执行某段程序，使这段程序能被公用。这样可减少程序编写和调试的工作量，于是引进了主程序和子程序的概念。指令系统中一般都有主程序调用子程序的指令和从子程序返回主程序的指令。通常把具有一定功能的公用程序段作为子程序，子程序的最后一条指令为返回主程序指令（RET）。



(1) 长调用指令

汇编指令格式	机器码格式	操作
LCALL addr16; 长调用指令提供了16位 目标地址, 在64K地 址空间内调用。	0001 0010 addr ₈₋₁₅ addr ₀₋₇	先 (PC)+3→PC (断点PC) , 后 (SP)+1→SP, (PC ₀₋₇) → (SP) (保护断点) (SP)+1→SP, (PC ₈₋₁₅) → (SP) addr ₀₋₁₅ →PC (子程序地址送PC)

(2) 短调用指令

汇编指令格式	机器码格式	操作
ACALL addr11; 短调用指令提供了11位 目标地址, 限定在2K地 址空间内调用。	a ₁₀ a ₉ a ₈ 1 0001 addr ₀₋₇	先 (PC)+2→PC (断点PC) , 后 (SP)+1→SP, (PC ₀₋₇) → (SP) (保护断点) (SP)+1→SP, (PC ₈₋₁₅) → (SP) addr ₀₋₁₀ →PC ₀₋₁₀ (子程序地址送PC) (PC ₁₁₋₁₅) 不变



(3) 返回指令

汇编指令格式	机器码格式	操作
RET ; 子程序返回	0010 0010	$((SP)) \rightarrow PC_{8-15}$, $(SP)-1 \rightarrow SP$, (弹出断点高8位) $((SP)) \rightarrow PC_{0-7}$, $(SP)-1 \rightarrow SP$, (弹出断点低8位)
RETI ; 中断服务返回	0011 0010	$((SP)) \rightarrow PC_{8-15}$, $(SP)-1 \rightarrow SP$, (弹出断点高8位) $((SP)) \rightarrow PC_{0-7}$, $(SP)-1 \rightarrow SP$, (弹出断点低8位) 开放中断逻辑



例如：主程序及子程序段如下：

```
0100H      MAIN:  ORG    0100H
0103H      MOV    SP,#60H
0106H      LCALL  SUB1
...
0200H      SUB1:  ORG    0200H
...
RET
```

分析执行 LCALL SUB1 及 RET 指令后的结果。

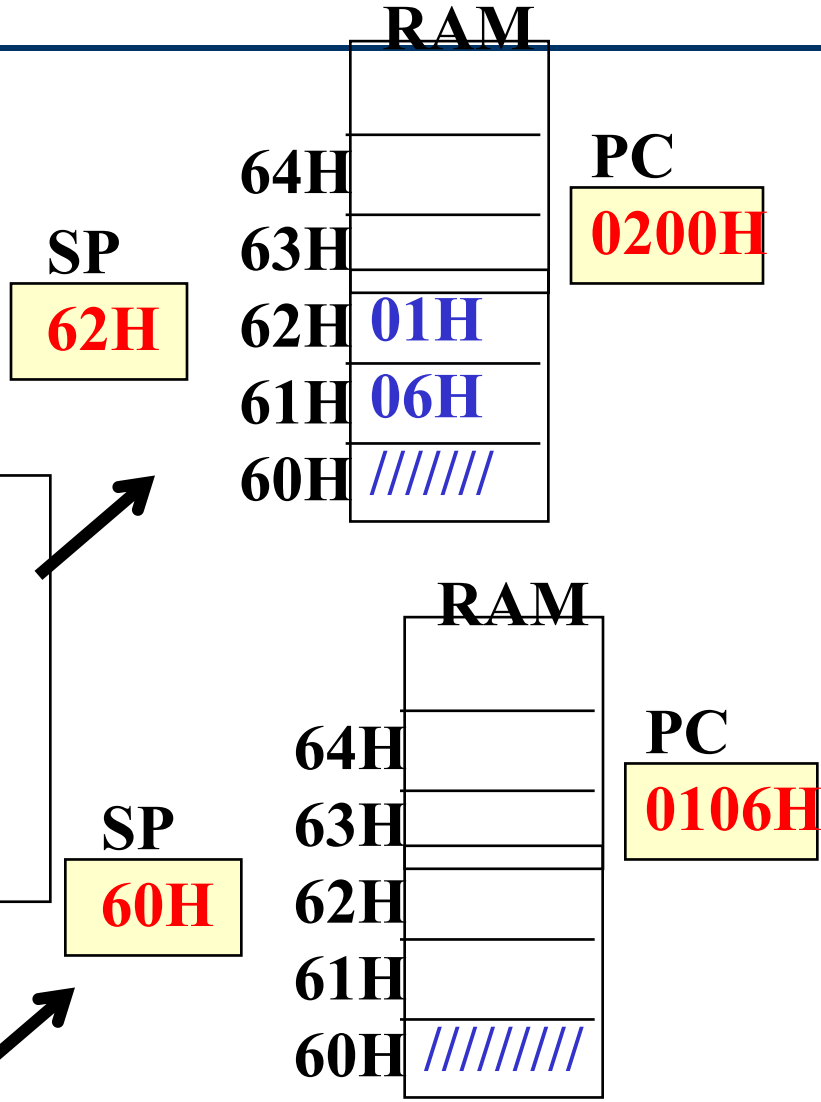
解：执行 LCALL SUB1 的过程：结果如图所示。

(PC)+3→PC : (PC)=0106H
(SP)+1→SP : (SP)=61H
(PC₀₋₇)→(SP) : (61H)=06H
(SP)+1→SP : (SP)=62H
(PC₈₋₁₅)→(SP) : (62H)=01H
addr₀₋₁₅→PC : (PC)=0200H(进入子程序)

执行 RET 的过程：如图所示。

((SP))→PC₈₋₁₅ : (PC₈₋₁₅)=01H
(SP)-1→SP : (SP)=61H
((SP))→PC₀₋₇ : (PC₀₋₇)=06H
(SP)-1→SP : (SP)=60H

所以：(PC)=0106H, 返回主程序。





例如：主程序及子程序段如下：



解：执行 **ACALL SUB1** 的过程：结果如图所示。

$(PC)+2 \rightarrow PC$: $(PC)=0104H$

$(SP)+1 \rightarrow SP$: $(SP)=61H$

$(PC_{0-7}) \rightarrow (SP)$: $(61H)=05H$

$(SP)+1 \rightarrow SP$: $(SP)=62H$

$(PC_{8-15}) \rightarrow (SP)$: $(62H)=01H$

$addr_{0-10} \rightarrow PC_{0-10}$: $(PC_{0-10}) = addr_{0-10} = 010\ 0000\ 0000\ B$

(PC_{11-15}) 不变 : $(PC_{11-15}) = 0000\ 0\ B$, 与 SUB1 的地址的高 5 位相同 (同一个 2K 范围内)。

$\therefore (PC) = 0000\ 0010\ 0000\ 0000\ B = 0200H$



§ 3.3.5 位操作(布尔处理)类指令(17条)

- 在进行位操作时，进位标志位CY——布尔累加器（位累加），简写成C。
- 位寻址区：片内RAM字节地址20H~2FH单元中连续的128位（位地址00H~7FH）和部分特殊功能寄存器SFR。

位地址的表达方式：

- 直接写位地址：如D4H；
- 点操作符号：如PSW. 4或(D0H). 4；
- 位名称方式：如RS1；
- 用户定义名方式：如用伪指令bit



位操作(布尔处理)类指令

1. 位数据传送指令（2条）
2. 位修正指令（6条）
3. 逻辑运算指令（4条）
4. 位条件转移类指令（5条）



1. 位数据传送指令（2条）

汇编指令格式	机器码格式	操作
MOV C, bit ;	1010 0010	(bit) → C
MOV bit, C ;	1001 0010	(C) → bit

例：初始时位地址（20H）=1

MOV C, 20H; CY=1

MOV P1.0, C; P1.0=1



2. 位修正指令（6条）

汇编指令格式	机器码格式	操作	注释
CLR C ;	1100 0011	$0 \rightarrow C$	C清0指令
CLR bit ;	1100 0010 位地址(bit)	$0 \rightarrow \text{bit}$	位地址清0指令
SETB C ;	1101 0011	$1 \rightarrow C$	C置1指令
SETB bit ;	1101 0010 位地址(bit)	$1 \rightarrow \text{bit}$	位地址置1指令
CPL C ;	1011 0011	$\overline{C} \rightarrow C$	C取反指令
CPL bit ;	1011 0010 位地址(bit)	$\overline{(\text{bit})} \rightarrow \text{bit}$	位地址取反置1指令



例如:

```
CLR  C      ;CY=0
CLR  ACC.0  ;ACC.0=0
CPL  ACC.0  ;ACC.0=1
SETB RS1    ;RS1=1
CLR  RS0     ;RS0=0 选择当前工作寄存器为第2组
CPL  C      ;CY=1
```



3. 逻辑运算指令（4条）

汇编指令格式	机器码格式	操作	注释
ANL C, bit ;	1000 0010 位地址 (bit)	$(C) \wedge (bit) \rightarrow C$	位逻辑“与”指令
ANL C, /bit ;	1011 0000 位地址 (bit)	$(C) \wedge (bit) \rightarrow C$	位逻辑“与”指令
ORL C, bit ;	0111 0010 位地址 (bit)	$(C) \vee (bit) \rightarrow C$	位逻辑“或”指令
ORL C, /bit ;	0101 0000 位地址 (bit)	$(C) \vee (bit) \rightarrow C$	位逻辑“或”指令

例：若位地址（20H）=1，位累加器（C）=0

执行指令：ANL C, /20H ; 后的结果：（C）=0，（20H）=1。

而执行指令：CPL 20H ;

ANL C, 20H; 后的结果：（C）=0，（20H）=0。



4、位条件转移类指令（5条）

(1) 判布尔累加器C转移指令（2条）

汇编指令格式	机器码格式	操作
JC rel ;	0100 0000 rel	先PC+2→PC, 若 (C) =1, 则 (PC) +rel→PC 若 (C) =0, 则顺序往下执行
JNC rel ;	0101 0000 rel	先PC+2→PC, 若 (C) =0, 则 (PC) +rel→PC 若 (C) =1, 则顺序往下执行

(2) 判位变量转移指令（2条）

汇编指令格式	机器码格式	操作
JB bit, rel ;	0010 0000 bit rel	先PC+3→PC, 若 (bit) =1, 则 (PC) +rel→PC 若 (bit) =0, 则顺序往下执行
JNB bit, rel;	0011 0000 bit rel	先PC+3→PC, 若 (bit) =0, 则 (PC) +rel →PC 若 (bit) =1, 则顺序往下执行



(3) 判位变量并清0转移指令（1条）

汇编指令格式	机器码格式	操作
JBC bit, rel ;	0001 0000 bit rel	先PC+3→PC, 若 (bit) =1, 则 (PC) +rel→PC, 0→bit 若 (bit) =0, 则顺序往下执行

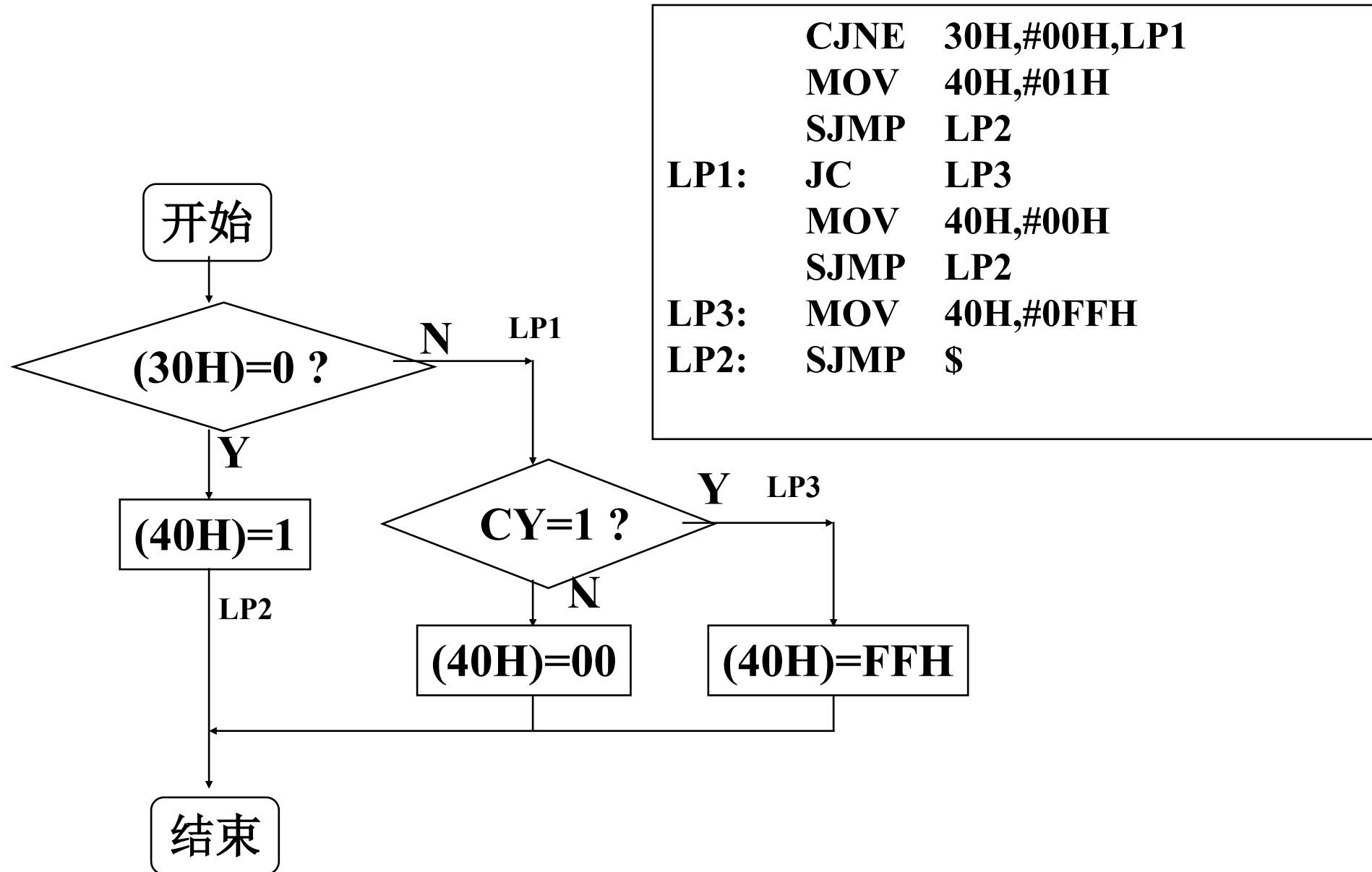


程序设计举例

1. 简单程序设计举例
2. 分支程序
3. 循环程序
4. 子程序设计举例
5. 代码转换程序设计举例
6. 运算类程序设计举例



例：程序设计，若片内RAM 30H单元内容大于0，40H单元置00；
30H单元内容<于0，40H单元置FFH； 30H单元内容等于0，40H单元置1；





例：数据块传送。将片内RAM 20H单元开始的10个字节，传送到片外RAM 1000H单元开始的单元中去。

```
解：          ORG    0800H

              START:MOV    R0, #20H

                  MOV    DPTR, #1000H

                  MOV    R7, #0AH

LOOP:  MOV     A, @R0

        MOVX   @DPTR, A

        INC    R0

        INC    DPTR

        DJNZ   R7, LOOP

        SJMP   $
```



例：统计字符串长度。内存从STRING开始有一字符串，该字符串以\$ (ASCII码为24H) 结束，试统计该字符串长度，结果存于NUM单元中

解：程序如下：

```
ORG      0800H
NUM      DATA 20H
STRING   DATA 21H

START: CLR    A
        MOV    R0, #STRING
LOOP:   CJNE   @R0, #24H, NEXT
        SJMP   COMP
NEXT:   INC     A
        INC     R0
        SJMP   LOOP
COMP:   MOV     NUM, A
        SJMP   $
```



例：用查表法编一子程序，将40H单元中的BCD码转换成ASCII码。

入口：待转换数在（40H） 出口：转换后的数（40H）

```
解：  ORG    1000H
      TAB:  DB    30H, 31H, 32H, 33H, 34H, 35H, 36H, 37H, 38H, 39H
      SUB1: MOV    A, 40H
           MOV    DPTR, #TAB
           MOVC   A, @A+DPTR
           MOV    40H, A
           RET
```

若将30H单元BCD码转换成ASCII码，利用SUB1子程序。

```
      ORG    2000H
MAIN: MOV    40H, 30H
      LCALL   SUB1
      MOV     30H, 40H
      SJMP    $
```

例：R1中存有一个BCD码，编一子程序将其转换为ASCII码，存入片外RAM1000H单元中去。（直接转换）



解：

```
                ORG    0100H
BAS1: MOV      A, R1
        ADD    A, #30H
        MOV    DPTR, #1000H
        MOVX   @DPTR, A
        RET
```




例：片外RAM2000H单元中有一BCD码，编一子程序将其转换成ASCII码。

```
                ORG    0100H  
BAS2: MOV      DPTR, #2000H  
        MOVX   A, @DPTR  
        ADD    A, #30H  
        MOVX   @DPTR, A  
        RET
```



例：片内RAM30H单元开始有10个字节的二进制数。编一子程序，求它们的和。（和<256）将和存入20H单元中。

解：方法1

```
ORG    1000H
ASUB1: MOV    R0, #30H
        CLR    A
        MOV    R7, #10
        CLR    C
LOOP:   ADDC   A, @R0
        INC    R0
        DJNZ   R7, LOOP
        MOV    20H, A
        RET
```

方法2

```
ORG    2000H
ASUB2: MOV    R0, #30H
        MOV    A, @R0
        MOV    R7, #09
        CLR    C
LOOP:   INC    R0
        ADDC   A, @R0
        DJNZ   R7, LOOP
        MOV    20H, A
        RET
```



例：编写将30H，31H单元中的两字节二进制数乘以2的程序（积小于65536）

解：

```
                ORG    1000H
MSUB: CLR      C
             MOV     A, 30H
             RLC     A
             MOV     30H, A
             MOV     A, 31H
             RLC     A
             MOV     31H, A
             RET
```



3.4 思考题与习题

1. 简述下列基本概念： 指令、指令系统、机器语言、汇编语言、高级语言。
2. 什么是计算机的指令和指令系统？
3. 简述89C51汇编指令格式。
4. 简述89C51的寻址方式和所能涉及的寻址空间。
5. 要访问特殊功能寄存器和片外数据存储器，应采用哪些寻址方式？



6. 在89C51片内RAM中，已知(30H)=38H，(38H)=40H，(40H)=48H，(48H)=90H。请分析下面各是什么指令，说明源操作数的寻址方式以及按顺序执行每条指令后的结果。

```
MOV      A, 40H
MOV      R0, A
MOV      P1, #0F0H
MOV      @R0,30H
MOV      DPTR, #3848H
MOV      40H, 38H
MOV      R0, 30H
MOV      P0, R0
MOV      18H, #30H
MOV      A, @R0
MOV      P2, P1
```



7. 对89C51片内RAM的高128字节的地址空间寻址要注意什么？

8. 指出下列指令的本质区别？

MOV A, data

MOV A,#data

MOV data1,data2

MOV 74H, #78H

9. 设R0的内容为32H，A的内容为48H，片内RAM的32H单元内容为80H，40H单元内容为08H。请指出在执行下列程序段后上述各单元内容的变化。

MOV A,@R0

MOV @R0, 40H

MOV 40H, A

MOV R0, #35H



10. 如何访问SFR，可使用哪些寻址方式？
11. 如何访问片外RAM单元，可使用哪些寻址方式？
12. 如何访问片内RAM单元，可使用哪些寻址方式？
13. 如何访问片内外程序存储器，可使用哪些寻址方式？
14. 说明十进制调整的原因和方法。
15. 说明89C51的布尔处理机功能。
16. 已知(A)=83H，(R0)=17H，(17H)=34H。请写出执行完下列程序段后A的内容。

ANL A, #17H

ORL 17H, A

XRL A, @R0

CPL A



17. 使用位操作指令实现下列逻辑操作。要求不得改变未涉及位的内容。
 - (1) 使ACC.0置1;
 - (2) 清除累加器高4位;
 - (3) 清除ACC.3, ACC.4, ACC.5, ACC.6。
18. 编程实现把内部RAM R0~R7的内容传递到20H~27H单元。
19. 试编写程序，将内部RAM的20H、21H和22H 3个连续单元的内容依次存入2FH、2EH和2DH中。
20. 编写程序，进行两个16位数的减法：6F5DH-13B4H，结果存入内部RAM的30H和31H单元，30H存储低8位。
21. 编写程序，若累加器A的内容分别满足下列条件，则程序转至LABEL存储单元。设A中存的是无符号数。
 - (1) $A \geq 10$;
 - (2) $A > 10$;
 - (3) $A \leq 10$ 。



22. 已知 $(SP)=25H$, $(PC)=2345H$, $(24H)=12H$, $(25H)=34H$, $(26H)=56H$ 。问此时执行“RET”指令以后, $(SP)=?$ $(PC)=?$
23. 若 $(SP)=25H$, $(PC)=2345H$, 标号LABEL所在的地址为3456H。问执行长调用指令“LCALL LABEL”后, 堆栈指针和堆栈的内容发生什么变化?PC的值等于什么?
24. 上题中的LCALL指令能否直接换成ACALL指令, 为什么?如果使用ACALL指令, 则可调用的地址范围是什么?
25. 试编写程序, 查找在内部RAM的20H~50H单元中是否有0AAH这一数据。若有, 则将51H单元置为01H; 若未找到, 则将51H单元清0。
26. 试编写程序, 查找在内部RAM的20H~50H单元中出现00H的次数, 并将查找的结果存入51H单元。
27. 外部数据RAM中有一个数据块, 存有若干字符、数字, 首地址为SOURCE。要求将该数据块传送到内部RAM以DIST开始的区域, 直到遇到字符“\$”时结束(“\$”也要传送, 它的ASCII码为24H)。



28. 已知R3和R4中存有一个16位的二进制数，高位在R3中，低位在R4中。请编程将其求补，并存回原处。
29. 已知30H和31H中存有一个16位的二进制数，高位在前，低位在后。请编程将它们乘以2，再存回原单元中。
30. 内存中有两个4字节以压缩的BCD码形式存放的十进制数，一个存放在30H~33H的单元中，一个存放在40H~43H的单元中。请编程求它们的和，结果放在30H~33H中。
31. 编写一个程序，把片外RAM从2000H开始存放的8个数传送到片内30H开始的单元中。
32. 要将片内RAM中0FH单元的内容传送到寄存器B，对0FH单元的寻址可以有3种方法：
- (1) R寻址；
 - (2) R间址；
 - (3) direct寻址。
- 请分别编出相应程序，比较其字节数、机器周期数和优缺点。



33. 阅读下列程序，要求：

- (1) 说明该程序的功能；
- (2) 填写所缺的机器码；
- (3) 试修改程序，使片内RAM的内容成为如图3-20所示的结果。

```
7A —      MOV   R2, #0AH
— —      MOV   R0, #50H
E4      CLR   A
F6  LOOP: MOV   @R0,A
08      INC   R0
DA —      DJNZ  R2, LOOP
      DONE:
```

⋮	
00H	50H
01H	51H
02H	52H
03H	53H
04H	54H
05H	55H
06H	56H
07H	57H
08H	58H
09H	59H
⋮	

图3-20 片内RAM



34. 设(R0)=7EH,(DPTR)=10FEH,片内RAM中7E单元的内容为0FFH, 7F单元的内容为38H。试为下列程序的每条指令注释其执行结果。

```
INC    @R0
INC    R0
INC    @R0
INC    DPTR
INC    DPTR
INC    DPTR
```

35. 下列程序段经汇编后, 从1000H开始的各有关存储单元的内容将是什么?

```
ORG    1000H
TAB1    EQU    1234H
TAB2    EQU    3000H
        DB     "START"
        DW     TAB1, TAB2, 70H
```



36. 阅读下列程序，并要求：

- (1) 说明程序的功能；
- (2) 写出涉及的寄存器及片内RAM单元(如图3-21所示)的最后结果。

图3-22片内RAM

```
MOV    R0,#40H
MOV    A, @R0
INC     R0
ADD     A,@R0
INC     R0
MOV     @R0,A
CLR     A
ADDC    A,#0
INC     R0
MOV     @R0,A
```

40H	⋮
	98H
	AFH
	⋮

图3-21 片内RAM



37. 要求同上题(如图3-22所示), 程序如下:

```
MOV  A, 61H
MOV  B, #02H
MUL  AB
ADD  A, 62H
MOV  63H, A
CLR  A
ADDC A, B
MOV  64H, A
```

61H

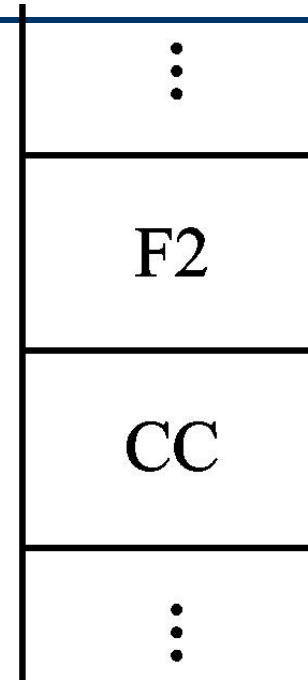


图3-22片内RAM



38. 试编写程序：采用“与”运算，判断某8位二进制数是奇数还是偶数个1。
39. 试编写程序：采用“或”运算，使任意8位二进制数的符号位必为1。
40. 请思考：采用“异或”运算，怎样可使一带符号数的符号位改变，数据位不变；怎样可使该数必然变为0。



第三章结束