第3章　8051指令系统

（课时：6学时）

教学目的

- 指令和指令系统的概念与分类。
- 掌握指令的格式和寻址方式。
- 掌握数据传送、算术运算、逻辑运算、控制转移和位操作指令的功能和应用。

学习重点和难点

- 指令的寻址方式。
- 控制转移指令和位操作指令。

◆**教学提示：** 指令是CPU用于控制功能部件完成某一指定动作的指示和命令。一台计算机全部指令的集合称为指令系统。指令系统体现了计算机的性能，也是计算机重要的组成部分，应用计算机进行程序设计的基础。单片机应用系统的运行，是依靠合理的硬件接口、用户程序和监控程序的完美结合实现的，所以掌握单片机需要学习多样的汇编程序设计方法实现运算和控制功能。

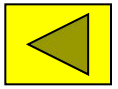
◆**教学要求：** 本章让学生了解单片机指令系统的特点和功能、操作的对象和结果、汇编语言程序结构的设计。重点掌握指令的基本形态、格式、寻址方式及汇编语言编程的基本方法，熟悉常用的子程序，能够正确运用汇编指令编制单片机应用系统的用户程序和监控程序。

第3章　8051指令系统

- 3.1　汇编语言
- 3.2　数据传送指令
- 3.3　算术运算指令
- 3.4　逻辑操作指令
- 3.5　控制程序转移类指令
- 3.6　位操作类指令
- 本章小结
- 习题

3.1　汇编语言

- 3.1.1　指令系统分类
- 3.1.2　指令格式
- 3.1.3　寻址方式
- 3.1.4　寻址空间及符号注释



3.1.1　指令系统分类

基本概念

- **指令是CPU根据人的意图来执行某种操作的命令。指令常以其英文名称或者缩写形式作为助记符。**
- **一台计算机所能执行的全部指令的集合称为这个CPU的指令系统。**
- 采用助记符表示的指令称为汇编语言。
- 使用这种指令编写的程序称为汇编语言程序。

指令系统分类（一）

- 按照指令的功能划分，8051指令可分为以下五类：
- 数据传送类指令(29条)
- 算术运算类指令(24条)
- 逻辑运算类指令(24条)
- 控制转移类指令(17条)
- 位操作类指令(17条)

指令系统分类（二）

- 按照指令占用的存储空间可分为：
- 单字节指令(49条)
- 双字节指令(45条)
- 三字节指令(17条)

- 按照指令的执行时间可分为：
- 单周期指令(64条)
- 双周期指令(45条)
- 四周期指令(2条，乘、除法指令)

3.1.2　指令格式

- 8051汇编语言指令由操作码段和操作数字段两部分组成。

- 汇编语言指令格式

  [标号:] 操作码助记符　[目的操作数] [,源操作数] [;注释]

- 机器语言指令格式
- 双字节指令格式

  操作码代码　　操作数或者地址
- 三字节指令格式

  操作码代码　　操作数或地址,操作数或地址

- MCS-51单片机指令格式
- １）指令由操作码助记符和操作数两部组成。
- ２）指令格式如下：
- [标号：]操作码助记符 [目的操作数] [，源操作数][；注释]
- 符号“[ ]”其包含的内容因指令的不同可有可无。
- ３）标号：根据编程需要给指令设定的符号地址，可有可无；通常在子程序入口或转移指令的目标地址处才赋予标号。标号由1～8个字符组成，第一个字符必须是英文字母，不能是数字或其他符号，标号后必须用冒号。

指令格式及其符号说明

4）操作码助记符：指令的核心部分，用于指示机器执行何种操作，如加、减、乘、除、传送等。

5）操作数：是指令操作的对象，可以是一个具体的数据，也可以是参加运算的数据所在的地址。操作数一般有以下几种形式：

- 没有操作数，操作数隐含在操作码中，如RET指令；
- 只有一个操作数，如INC A指令；
- 有两个操作数，如MOV A，30H指令，操作数之间以逗号相隔；
- 有3个操作数，如CJNE A，#00H，10H指令。

6）注释：对指令的解释说明，用以提高程序的可读性，注释前必须加分号，注释换行时行前也要加分号。

指令格式及其符号说明

- 指令的字节

1. 单字节指令(49条)

在MCS-5I指令系统中，单字节指令可分为两类：无操作数的单字节指令和含有操作数寄存器编号的单字节指令。

1) 无操作数单字节指令

这类指令只有操作码字段，操作数隐含在操作码中。例如：INC DPTR

指令格式及其符号说明

指令码为

位	D7	D6	D5	D4	D3	D2	D1	D0	十六进制码
操作码	1	0	1	0	0	0	1	1	A3H

2) 含有操作数寄存器号的单字节指令

单字节的指令码由操作码字段和指示操作数所在寄存器号的字段组成。

例如：MOV A，Rn

指令码为（**r r r**为寄存器Rn的编号）

位	D7	D6	D5	D4	D3	D2	D1	D0	十六进制码
操作码+操作数	1	1	1	0	1	**r**	**r**	**r**	E8H～EFH

指令格式及其符号说明

2. 双字节指令(45条)

双字节指令的操作码字节在前，其后的操作数字节可以是立即数，也可以是操作数所在的片内RAM地址。

例如：MOV A，#23H

指令码为

位	D7	D6	D5	D4	D3	D2	D1	D0	十六进制码
操作码	0	1	1	1	0	1	0	0	74H 23H
操作数(立即数)	0	0	1	0	0	0	1	1	

指令格式及其符号说明

这条8位数传送指令的含义是：把指令码第2字节立即数23H取出来存放到累加器(A)中。

该指令的操作码占1B，23H为源操作数，也是1B，累加器(A)是目的操作数寄存器，隐含在操作码字节中。

指令格式及其符号说明

3. 三字节指令(17条)

这类指令的指令码的第1字节为操作码；第2和第3字节为操作数或操作数地址，有如下4类。

1) 16位数据

例如：MOV DPTR，#26ABH

指令码为

位	D7	D6	D5	D4	D3	D2	D1	D0	十六进制码
操作码	1	0	0	1	0	0	0	0	90H 26H ABH
操作数(立即数高)	0	0	1	0	0	1	1	0	
操作数(立即数低)	1	0	1	0	1	0	1	1	

指令格式及其符号说明

2) 8位地址和8位数据

例如：MOV 74H，#0FFH

指令码为

位	D7	D6	D5	D4	D3	D2	D1	D0	十六进制码
操作码	0	1	1	1	0	1	0	1	75H 74H FFH
操作数(地址)	0	1	1	1	0	1	0	0	
操作数(立即数)	1	1	1	1	1	1	1	1	

指令格式及其符号说明

3) 8位数据和8位地址

例如：CJNE A，#00，60H

指令码为

位	D7	D6	D5	D4	D3	D2	D1	D0	十六进制码
操作码	1	0	1	1	0	1	0	0	B4H 00H 60H
操作数(立即数)	0	0	0	0	0	0	0	0	
操作数(地址)	0	1	1	0	0	0	0	0	

3.1 指令格式及其符号说明

4) 16位地址

例如：LCALL 2020H

指令码为

位	D7	D6	D5	D4	D3	D2	D1	D0	十六进制码
操作码	0	0	0	1	0	0	1	0	12H 20H 20H
操作数(地址高)	0	0	1	0	0	0	0	0	
操作数(地址低)	0	0	1	0	0	0	0	0	

程序设计中，应尽可能选用字节少的指令。这样，指令所占存储单元少，执行速度也快。

3.1　指令格式及其符号说明

- MCS-51单片机的助记符语言

　　为了便于人们识别、读/写、记忆和交流用英文单词或缩写字母来表征指令功能，这些指令的助记符形式称为汇编语言指令，常用于汇编语言源程序的程序设计。

　　MCS-51单片机制造厂家对每一条指令都给出了助记符。不同的指令，具有不同的功能和不同的操作对象。如图表3-1 MCS-51助记符意义

助记符	意　义	助记符	意　义
MOV	送数	MUL	乘法
MOVC	ROM送累加器(A)	DIV	除法
MOVX	外部送数	DA	十进制调整
PUSH	压入堆栈	AJMP	绝对转移
POP	堆栈弹出	LJMP	长转移
XCH	数据交换	SJMP	短转移
XCHD	交换低4位	JMP	相对转移
ANL	与运算	JZ	判累加器A为0转移
ORL	或运算	JNZ	判累加器A非0转移
XRL	异或运算	JC	判CY为0转移
SETB	置位	JNC	判CY非0转移
CLR	清0	JB	直接位为1转移
CPL	取反	JNB	直接位为0转移
RL	循环左移	JBC	直接位为1转移，并清该位
RLC	带进位循环左移	CJNE	比较不相等转移
RR	循环右移	DJNZ	减1不为0转移
RRC	带进位循环右移	ACALL	绝对调用子程序
SWAP	高低半字节交换	LCALL	长调用子程序
ADD	加法	RET	子程序返回
ADDC	带进位加法	RETI	中断子程序返回
SUBB	带进位减法	NOP	空操作
INC	加1	DEC	减1

- 常用符号说明

指令的书写必须遵守一定的规则，见表3-2指令描述约定。表3-2 指令描述约定

符　号	含　义
Rn	表示当前选定寄存器组的工作寄存器R0～R7，*n*=0～7
Ri	表示作为间接寻址的地址指针R0～R1，*i*=0，1
#data	表示8位立即数，即00H～FFH
#data16	表示16位立即数，即0000H～FFFFH
Addr16	16位地址，可表示用于64KB范围内寻址，用于LCALL和LJMP指令中
Addr11	11位地址，可表示2KB范围内寻址，用于ACALL和AJMP指令中
direct	8位直接地址，可以是片内RAM区的某一单元或某一专用功能寄存器的地址
rel	带符号的8位地址偏移量(−128～+127)，用于SJMP和条件转移指令中
bit	位寻址区的直接寻址位，表示片内RAM中可寻址位和SFR中的可寻址位
(X)	X地址单元中的内容，或X作为间接寻址寄存器时所指单元的内容
((X))	由X寻址的单元的内容
←	将箭头后面的内容传送到箭头前面去
$	当前指令所在地址
DPTR	数据指针
/	加在位地址之前，表示该位状态取反
@	间接寻址寄存器或基址寄存器的前缀

3.1.3　寻址方式

基本概念

- 寻址就是寻找指令中操作数或操作数所在地址。

- 寻址方式就是找到存放操作数的地址，并把操作数提取出来的方法，即寻找操作数或者是操作数地址的方法。

- 8051单片机寻址方式共有7种：寄存器寻址、直接寻址、立即数寻址、寄存器间接寻址、变址寻址、相对寻址和位寻址。

1. 寄存器寻址

- 寄存器寻址就是操作数存放在寄存器中，指令中指定的寄存器的内容就是操作数。

- 在寄存器寻址方式中以符号名称来表示寄存器。

- 寄存器寻址方式的寻址范围包括：
- 通用工作寄存器——有4组共32个通用工作寄存器。
- 部分专用寄存器——累加器A、B寄存器、DPTR寄存器。

寻 址 方 式

- 寄存器寻址

  寄存器寻址：由指令指出某一个寄存器的内容作为操作数。可以采用寄存器寻址的寄存器有

(1) 工作寄存器R0～R7；组别的选择由程序状态字(PSW)中的RS0、RS1决定；

(2) 累加器(A)；

(3) 寄存器(B)；

(4) 数据指针(DPTR)。

寻 址 方 式

【例】已知：(R0)=0AAH，执行指令：

MOV　A，R0；(A)←(R0)

指令码为：E8H

结果：(A)=0AAH

该指令的功能是将R0中的内容30H传送到累加器(A)。操作数采用寄存器寻址方式。寻址如图3.3所示。

3.2 寻 址 方 式

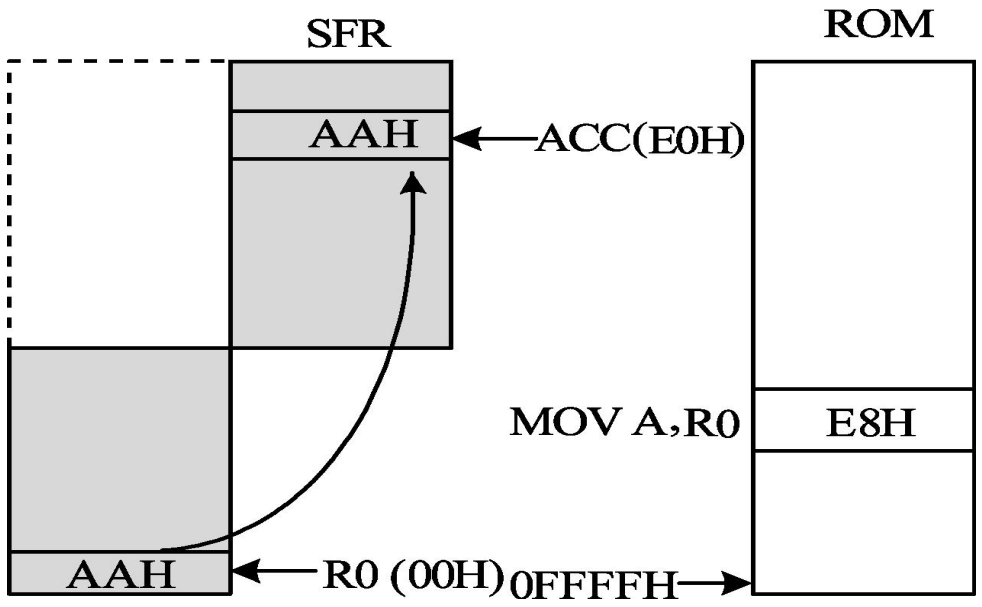


图3.3 指令MOV A，R0的执行示意图

2. 直接寻址

- 直接寻址方式就是在指令中直接给出操作数所在存储单元的地址。此时，指令中操作数部分是操作数所在地址。

- 直接寻址方式的寻址范围是内部RAM，具体包括：
- 片内RAM的128个单元——在指令中以直接地址给出。
- 特殊功能寄存器(SFR)——既可使用它们的地址，也可使用它们的名字。

3.2 寻 址 方 式

- 直接寻址

  直接寻址：指令中含有操作数的地址，该地址指出了参与运算或传送的数据所在的字节单元或位。直接寻址方式可访问的存储空间包括特殊功能寄存器和片内RAM的低128B。

(1) 特殊功能寄存器：只能用直接寻址方式访问，并且特殊功能寄存器常以符号的形式表示。

(2) 片内RAM的低128B：52及以上子系列单片机的高128B不能用直接寻址方式访问，只能用后面提到的寄存器间接寻址方式，因为高128B的编码与特殊功能寄存器的地址重叠。

3.2 寻 址 方 式

【例】已知：(30H)=0AAH，执行指令：

MOV　A，30H ；(A)←(30H)

指令码为：E5H 30H

MOV　PSW，#20H　；(PSW)← 20H

结果：(A)=0AAH　(PSW)=20H

第1条指令的功能是将片内RAM的30H单元内容“0AAH”传送到累加器(A)。第2条指令的功能是将立即数20H传送给特殊功能寄存器PSW。操作数采用直接寻址方式。第1条指令寻址如图3.2所示。

3.2 寻 址 方 式

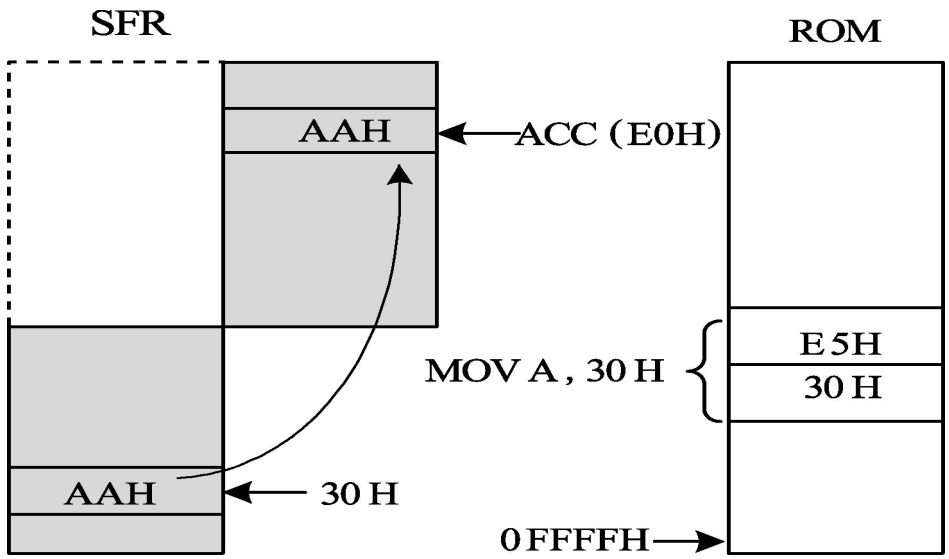


图3.2 指令 MOV A，30H执行示意图

3. 立即数寻址

- 立即数寻址就是操作数在指令中直接给出，或者说指令操作码后面立即给出一字节或两字节操作数。

- 指令中给出的操作数是立即数，立即数前加“#”号标志，以区别直接寻址中的直接地址。

- 16位立即数传送指令“MOV DPTR，#data16”，其中#data16是一个16位立即数。

3.2 寻 址 方 式

- 立即寻址

立即寻址：指令中跟在操作码后面的字节就是直接参加运算的操作数，称为立即数，用符号“#”表示，以区别直接地址。立即数通常使用8位二进制数#data，但指令系统中有一条立即数为#data16的指令。

【例】执行指令：

MOV A，#30H；(A)←30H

指令码为：74H 30H

MOV DPTR，#1638H；(DPH)←16H，(DPL)←38H

指令码为：90H 16H 38H

结果：(A)=30H

(DPTR)=1638H

第1条指令表示将立即数30H送入累加器(A)中。第2条指令表示把16位立即数送入数据指针寄存器(DPTR)中，其中高8位送DPH，低8位送DPL。

3.2 寻 址 方 式

立即寻址示意图如图3.1所示。

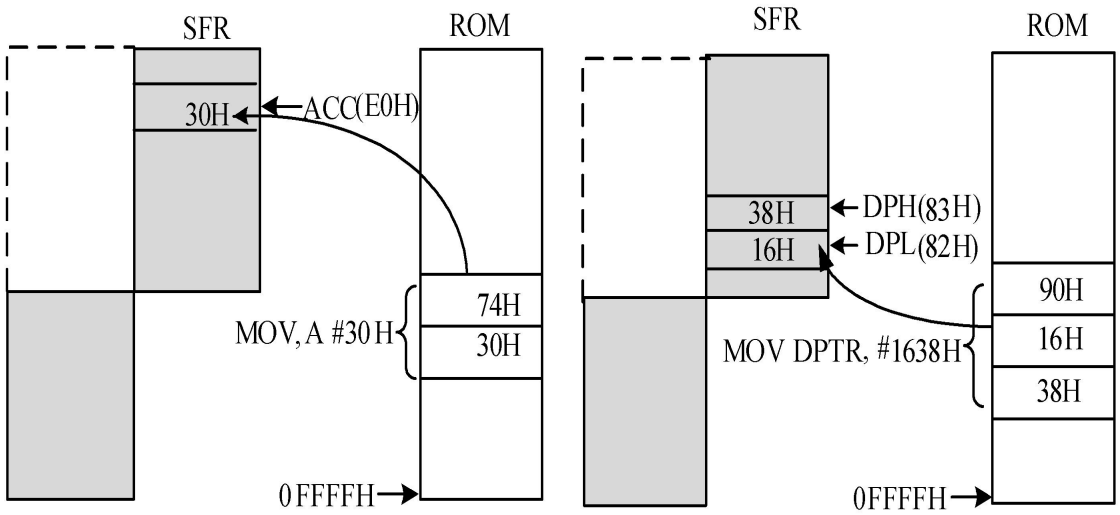


图3.1 立即寻址示意图

4. 寄存器间接寻址

- 寄存器间接寻址是把指定寄存器的内容作为操作数地址，该地址所指定的单元内容作为操作数。

- 为了区分寄存器寻址和寄存器间接寻址，在寄存器间接寻址中，所用到的寄存器的前面要加间接寻址符“@”。

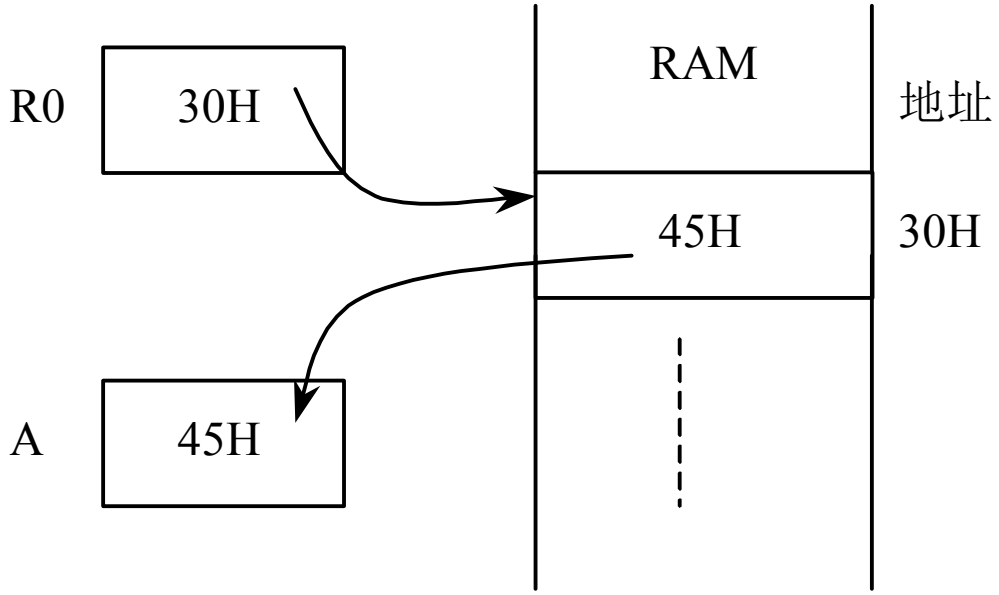
- 寄存器R0、R1和数据指针DPTR可以作为间接寻址寄存器。

4. 寄存器间接寻址

例如：寄存器R0内容为30H，片内RAM 30H单元的内容为45H。

解：指令“MOV　A，R0”的功能是将R0的内容30H传送给累加器A，指令执行结果是累加器A中的内容为30H。

指令“MOV　A，@R0”的功能是将R0的内容30H作为操作数的地址，根据这一地址找到内部RAM 30H单元，将其内容45H传送至累加器A，指令执行结果是累加器A中内容为45H。



寄存器间接寻址示意图

3.2 寻 址 方 式

- 寄存器间接寻址

寄存器间接寻址：指令中寄存器中存放的不是操作数而是操作数的地址，操作数是通过寄存器间接得到的。而寄存器寻址中寄存器存放的就是操作数。在间接寻址的寄存器前加前缀标志“@”相区别。同时寄存器间接寻址时访问片内RAM和片外RAM有一些区别：

(1)片内RAM：共有128B，采用形式为@R0、@R1或SP，访问时用MOV操作符。

3.2 寻 址 方 式

(2)片外RAM：最大容量64KB，寻址时由P2口提供高8位地址，R0、R1提供低8位地址共同寻址64KB范围；也可用16位的DPTR作寄存器间接寻址64KB存储空间，访问时用MOVX操作符。

【例】已知：(R0)=0AAH，(0AAH)=5BH，执行指令：　MOV　A，@R0 ；A←((R0))

指令码为：E6H

结果：　(A)=5BH

该例中用寄存器间接寻址将片内RAM中由R0的内容为地址所指示的单元的内容传送到累加器(A)。该指令的操作数采用寄存器间接寻址方式，如图3.4所示。

3.2 寻 址 方 式

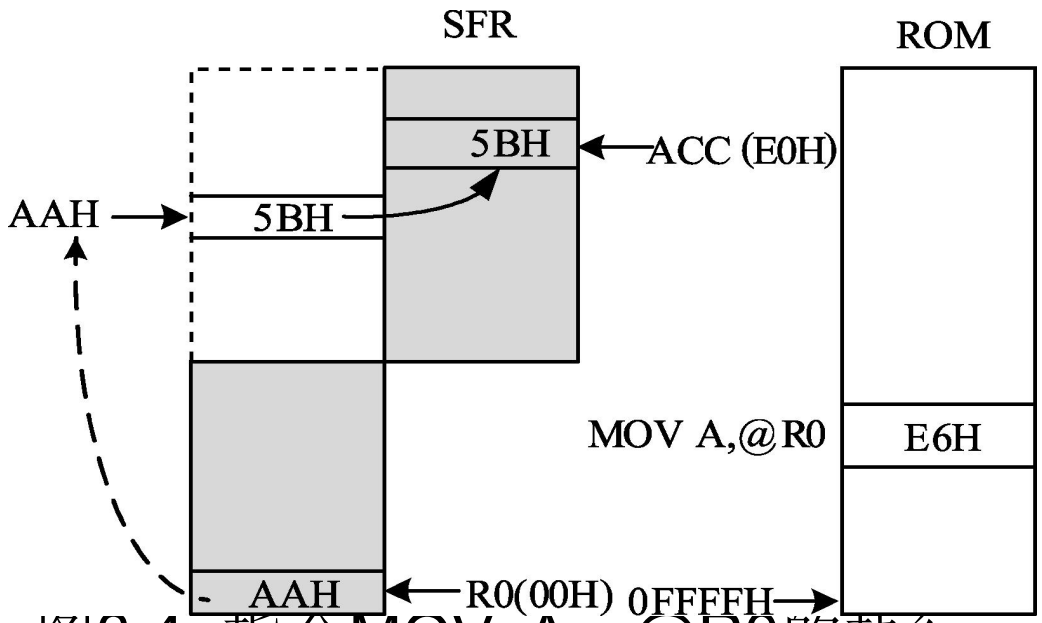


图3.4 指令MOV A，@R0的执行示意图

采用“MOVX”类操作的片外RAM的数据传送指令如：

MOVX A，@R0

MOVX A，@DPTR

5. 变址寻址(基址寄存器+变址寄存器间接寻址)

- 变址寻址是以DPTR或者PC作为基址寄存器，其内容为基地址，以累加器A作为变址寄存器，其内容为变址，并将两个寄存器内容也就是基址和变址相加，形成16位操作数地址。然后在程序存储器中找到该地址所对应的单元，其内容即为操作数。

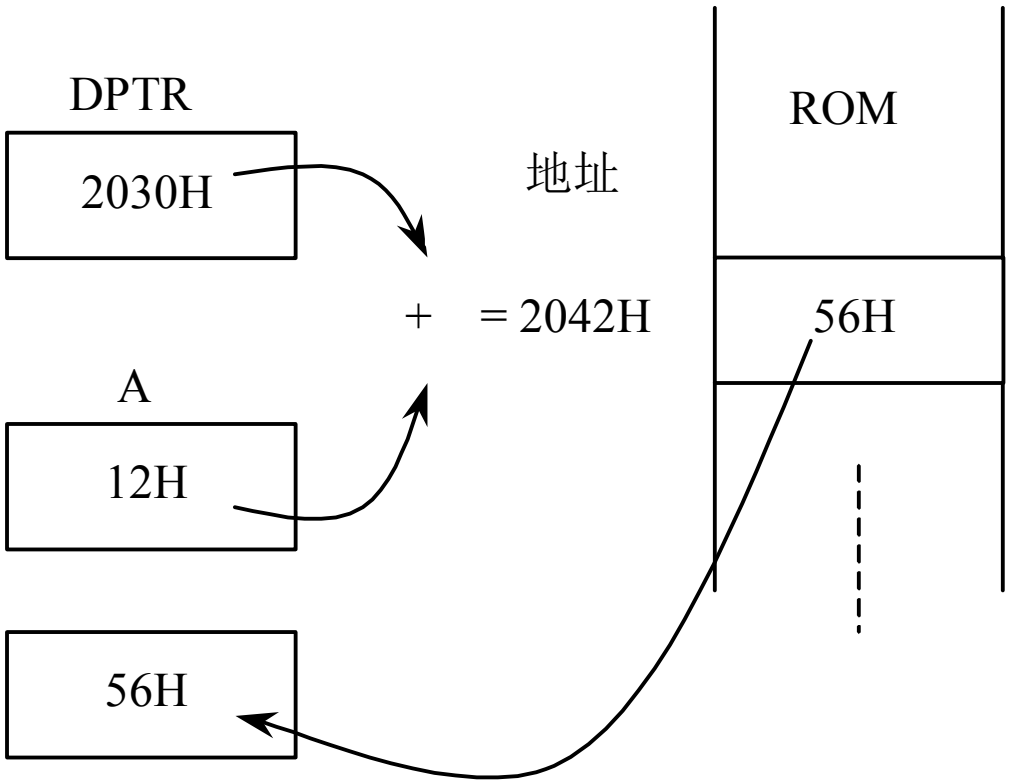
5. 变址寻址(基址寄存器+变址寄存器间接寻址)

例如：指令“MOVC A，@A+DPTR”是变址寻址。

解：假设DPTR的内容为2030H，累加器A的内容为12H。

该指令的功能是将2030H和12H相加，得到2042H作为操作数地址，在程序存储器中找到2042H单元，将其内容送至累加器A。

变址寻址示意图



3.2 寻 址 方 式

- 变址寻址

  变址寻址：操作数地址＝基址寄存器的内容＋变址寄存器的内容。

- 基址寄存器：程序计数器(PC)或数据指针(DPTR)；
- 变址寄存器：A。

  形成的地址是16位地址，这种寻址方式只能用来访问ROM的查表操作，所以变址寻址操作只有读操作而无写操作。变址寻址指令操作符有MOVC查表指令。

3.2 寻 址 方 式

【例】已知：(A)＝08H，(DPH)=20H，(DPL)=00H，即(DPTR)=2000H，(2008H)＝88H，执行指令：

MOVC　A，@A+DPTR

首先将DPTR的内容2000H与累加器(A)的内容08H相加，得到地址2008H。然后将该地址的内容88H取出传送到累加器(A)，这时，累加器(A)的内容为88H，原来累加器(A)的内容08H被冲掉。如图3.5所示。

3.2 寻 址 方 式

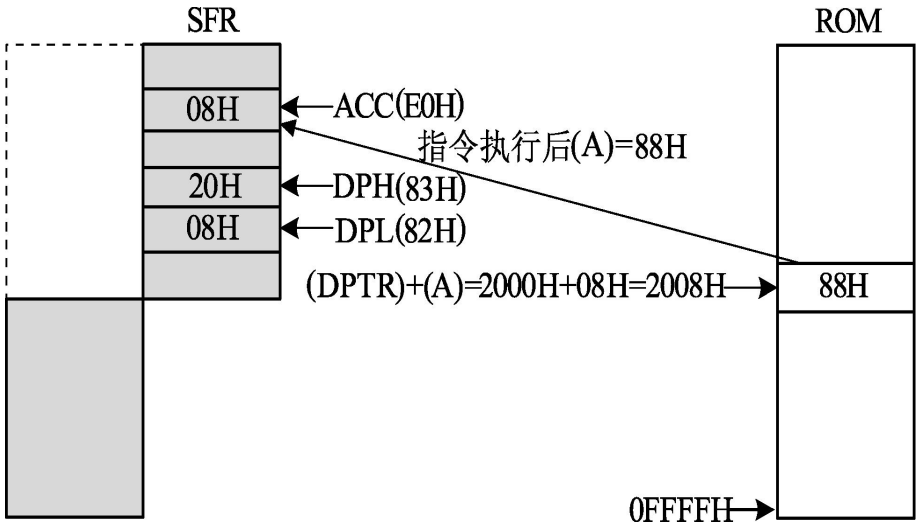


图3.5 指令MOVC A，@A+DPTR 的执行示意图

另外两条变址寻址指令为：

MOVC A，@A+PC ；JMP @A+DPTR

3.2 寻 址 方 式

前一条指令的功能是将累加器(A)的内容与程序计数器(PC)的内容相加形成操作数地址，把该地址中的数据传送到累加器(A)中。

【例】已知：(A)=30H，(1031H)=20H

执行地址1000H处的指令：

1000H：MOVC A，@A+PC

指令码：83H

这条指令占用一个单元，下一条指令的地址为1001H，即执行完本指令后(PC)=1001H，(PC)=1001H再加上累加器(A)中的30H，指令执行结果将ROM 1031H的内容20H传送给累加器(A)，不改变程序计数器(PC)的内容。示意图如图3.6所示。

3.2 寻 址 方 式

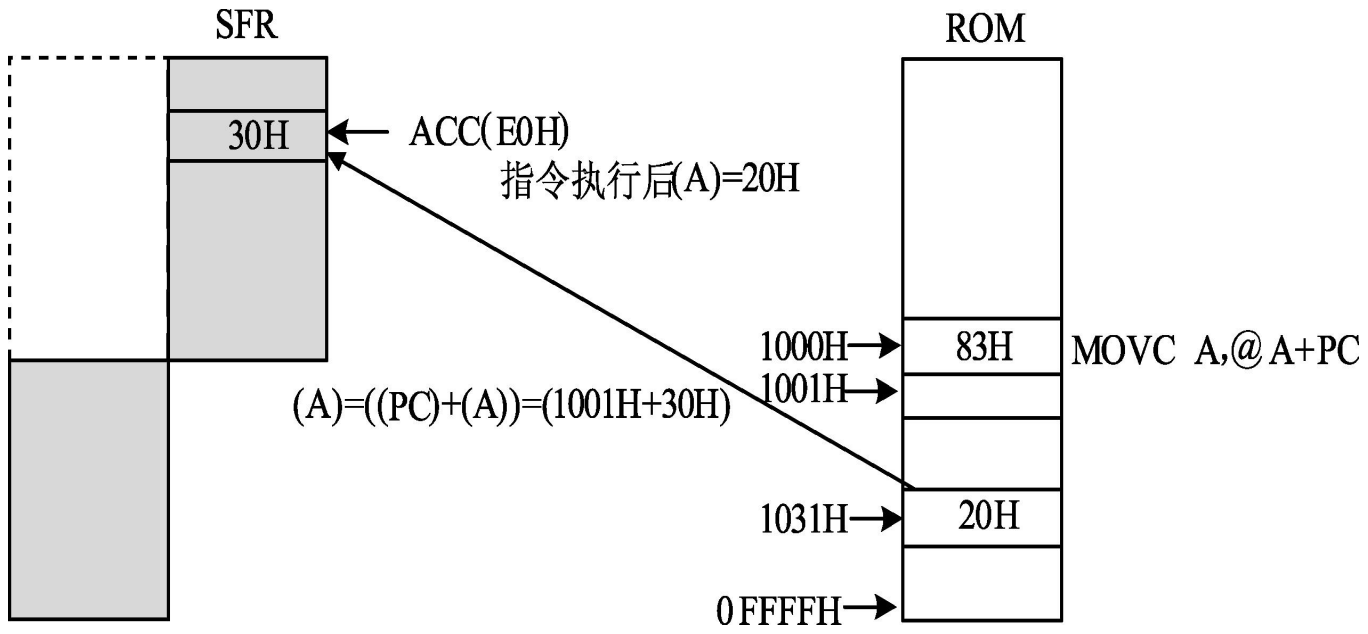


图3.6 指令MOVC A，@A+PC的执行示意图

3.2 寻 址 方 式

后一条指令的功能是将累加器(A)的内容与数据指针寄存器(DPTR)的内容相加形成指令跳转地址，从而使程序转移到该地址运行，改变程序计数器(PC)的内容，为无条件跳转指令。

【例】已知：(A)=08H，(DPTR)=2000H

执行指令JMP @A+DPTR后，(PC)=2008H，程序从ROM的2008H地址开始执行。示意图如图3.7所示。

3.2 寻 址 方 式

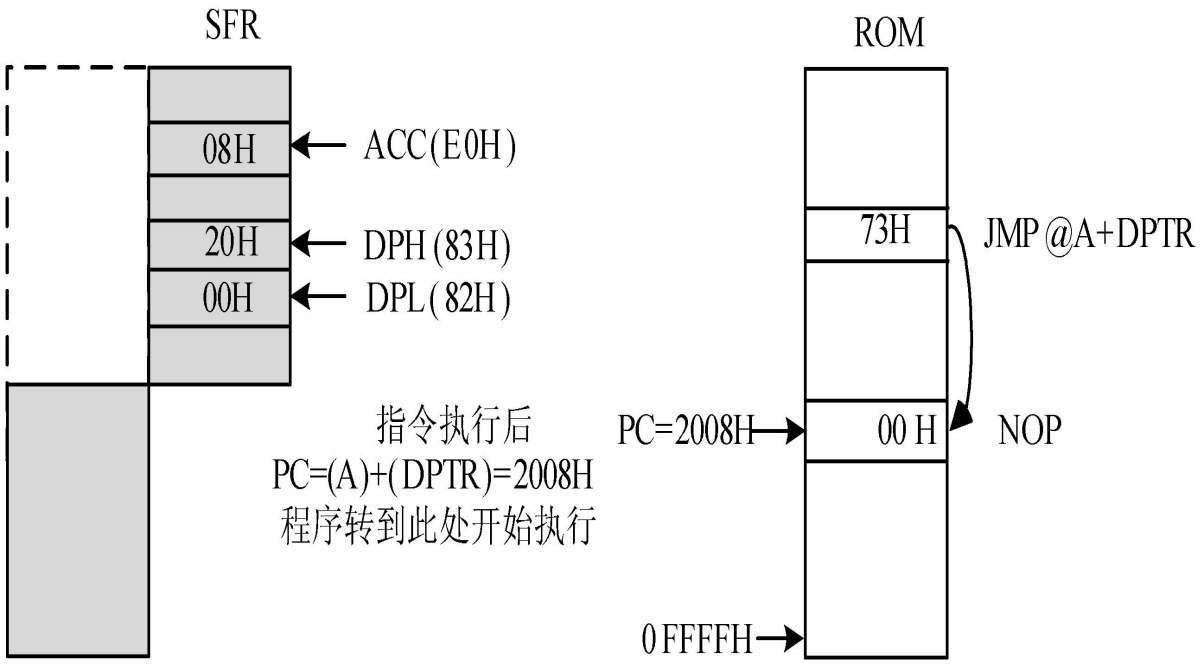


图3.7 指令JMP @A+DPTR的执行示意图

6. 相对寻址

- 相对转移指令执行时，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。这里所说的PC的当前值是执行完相对转移指令后的PC值。

- 一般将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。于是有：

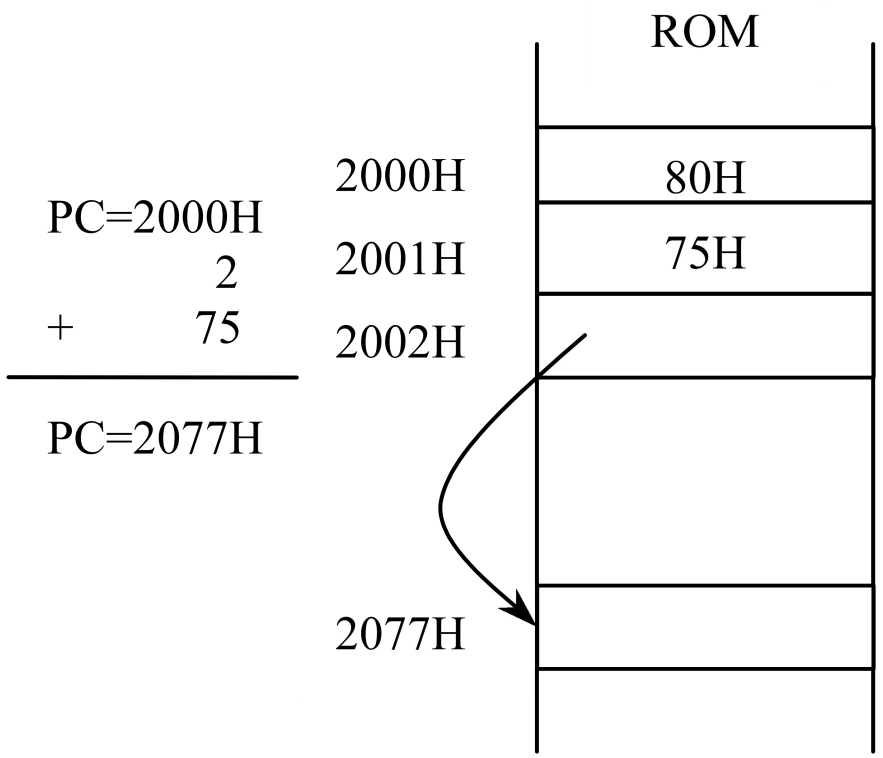
  目的地址=源地址+相对转移指令本身字节数+rel

- 相对寻址只出现在相对转移指令中，以修正PC的方式来控制程序的转移目的。

例如：在程序存储器中2000H单元有一条双字节的相对转移指令“SJMP 75H”。

解：程序存储器2000H、2001H单元的内容80H、75H是“SJMP 75H”的机器语言代码。

相对转移指令操作示意图



3.2 寻 址 方 式

- 相对寻址

相对寻址：转移的目的地址可用如下公式表示：

目的地址=转移指令下条指令地址+转移指令的字节数+rel

此种寻址方式主要用于实现程序的分支转移。其中，rel是一个带符号的8位二进制数，取值范围是-128～+127，以补码形式置于操作码之后。执行跳转指令时，先取出该指令，PC指向下一条指令地址。再把 rel 的值加到PC上以形成转移的目标地址。

3.2 寻 址 方 式

【例】已知：(PC)=2000H 执行指令：

地址：	ORG	2000H	指令码
2000H	SJMP	08H	80H 08H
2002H	NOP		00H
… …	… …		
200AH	NOP		00H

结果：程序转移到200A处开始继续执行。因为“SJMP 08H”指令码本身占2B，CPU执行完该指令之后PC值已等于下一条指令的地址即2002H，此时的PC值加上偏移量08H后赋给PC，(PC)=200AH，程序转到200AH处开始执行。操作示意图如图3.8所示。

3.2 寻 址 方 式

操作示意图如图3.8所示：

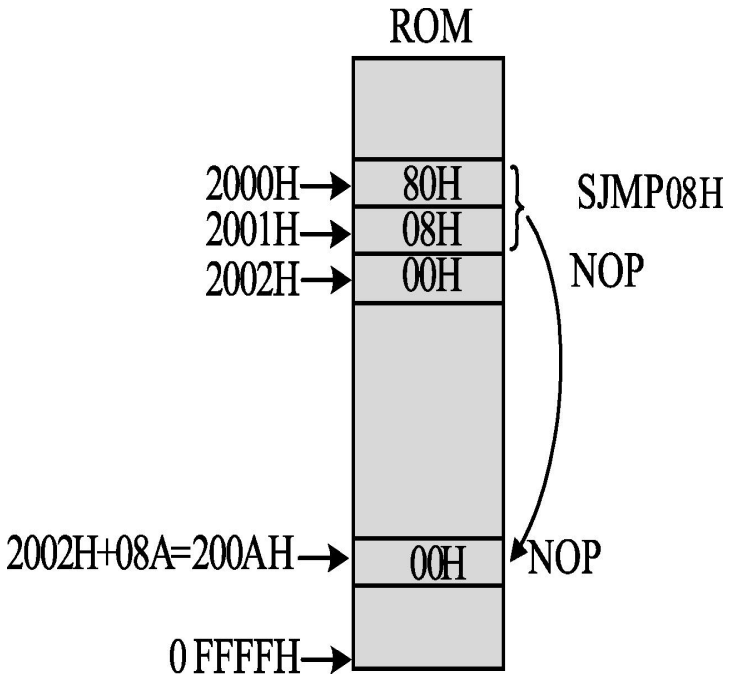


图3.8 指令SJMP 08H 的执行示意图

7. 位寻址

- 位寻址指令中给出的是位地址，即片内RAM某一单元中的一位。位地址在指令中用bit表示。

- 单片机片内RAM有两个区域可以进行位寻址。具体如下：
- 内部RAM中的位寻址区——该区共有16个单元，单元地址是20H～2FH，一共有128位，位地址为00～7FH。
- 特殊功能寄存器的可操作位——有11个单元地址能被8整除的寄存器，它们都可以进行位寻址，实际可寻址位为83个。

3.2 寻 址 方 式

- 位寻址

位寻址：指令中对数据位直接进行操作。位寻址与直接寻址不同，位寻址只给出位地址，而不是字节地址。可位寻址区为：

(1) 片内RAM中的位寻址区20H～2FH，共16个单元，128个位，位地址是00H～7FH。

(2) 特殊功能寄存器(SFR)的可寻址位。习惯上可寻址位常用符号位地址表示，如TI、RI。

【例】 执行指令：

CLR ACC．0

MOV 30H，C

第1条指令的功能都是将累加器(ACC)的位0清“0”。第2条指令的功能是把位累加器(注：在指令中用“C”表示)的内容传送到片内RAM位地址为30H的单元。

可寻址位在指令中的表示方式

① 直接使用位地址。

例：PSW寄存器的第5位可表示为D5H。

20H单元的第7位可表示为07H。

② 用位名称表示。

例：PSW寄存器的第5位可表示为F0。

③ 单元地址加位号表示。

例：PSW寄存器的第5位可表示为D0H.5。

20H单元的第7位可表示为20H.7。

④ 可以用寄存器名称加位号表示。

例：PSW寄存器的第5位可表示为PSW.5。

3.1.4　寻址空间及符号注释

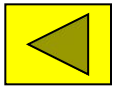
寻址方式	寻址空间
直接寻址	片内RAM低128字节和特殊功能寄存器(只能采用直接寻址)
寄存器寻址	工作寄存器R0～R7、A、B、DPTR
寄存器间接寻址	片内RAM低128字节(@R0、@R1、SP用于PUSH/POP指令时) 片外RAM(@R0、@R1、@DPTR)
变址寻址	程序存储器
相对寻址	程序存储器(控制转移用)
位寻址	片内RAM20～2FH单元的128个可寻址位和特殊功能寄存器中的83个可寻址位

寻址方式中常用符号注释

- Rn（n=0～7），当前选中的工作寄存器组R0～R7。它在片内数据存储器中的地址由PSW中的RS1和RS0确定，可以是00H～07H(第0组)、08H～0FH(第1组)、10H～17H(第2组)或18H～1FH(第3组)。
- Ri（i=0，1），当前选中的工作寄存器组中可以用于寄存器间接寻址的的两个工作寄存器R0、R1。它在片内数据存储器中的地址由RS1、RS0确定，分别有01H，02H；08H，09H；10H，11H和18H，19H。
- #data，8位立即数，即包含在指令中的8位操作数。
- #data16，16位立即数，即包含在指令中的16位操作数。
- 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，补码形式的8位地址偏移量，以下条指令第一字节地址为基值。
- 地址偏移量在-128～+127范围内。
- bit，片内RAM或SFR的直接寻址位地址。
- @，间接寻址方式中，表示间接寻址的符号。
- /，位操作指令中，表示对该位先取反再参与操作，但不影响该位原值。
- (X)，某一个寄存器或者存储单元X中的内容。
- ((X))，由X间接寻址的单元的内容，即X指向的地址单元中的内容。
- ←，指令中数据的传送方向，将箭头右边的内容送入箭头左边的单元。

3.2　数据传送指令

- 3.2.1　内部RAM数据传送指令
- 3.2.2　外部RAM传送指令
- 3.2.3　查表指令
- 3.2.4　交换指令
- 3.2.5　堆栈操作指令
- 3.2.6　数据传送类指令应用实例



- CPU在进行算术运算和逻辑运算时总需要有操作数据，所以数据传送是一种最基本最主要的操作。在MCS-51系列单片机中的数据传送指令相当丰富。数据传送指令按数据传送的区域可分为3组：
- 一组：内部数据传送；
- 二组：与片外RAM或I/O接口之间的数据传送；
- 三组：ROM到累加器(A)的传送。

3.2.1　内部RAM数据传送指令

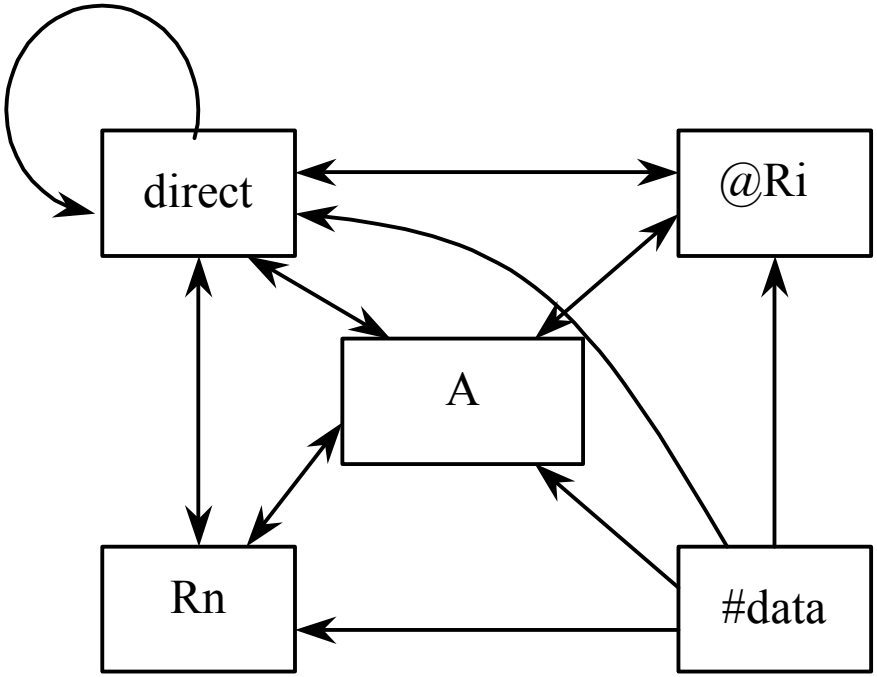
- 内部**RAM**数据传送指令共有**15**条，用于**8051**单片机片内数据存储器和寄存器之间的数据传送。
- 采用的寻址方式有：

  立即数寻址

  直接寻址

  寄存器寻址

  寄存器间接寻址
- 数据传输形式如右图所示。



3.2.1　内部RAM数据传送指令

1. 以累加器A为目的操作数的指令

汇编指令格式	机器指令格式	操　作
MOV　A,Rn	E8H～EFH	A ← (Rn)
MOV　A,direct	E5H　direct	A ← (direct)
MOV　A,@Ri	E6H～E7H	A ← ((Ri))
MOV　A,#data	74H　data	A ← #data

注意：上述操作不影响源字节和任何别的寄存器内容，只影响PSW的P标志位。

3.2.1　内部RAM数据传送指令

2. 以寄存器Rn为目的操作数的指令

汇编指令格式	机器指令格式		操　作
MOV　Rn,A	F8H～FFH		Rn ← (A)
MOV　Rn,direct	A8H～AFH	direct	Rn ← (direct)
MOV　Rn,#data	78H～7FH	data	Rn ← #data

注意：8051指令系统中没有“MOV Rn，@Ri”和“ MOV Rn，Rn”传送指令，也没有“MOV @Ri，@Ri”指令。

3.2.1　内部RAM数据传送指令

3. 以直接地址为目的操作数的指令

汇编指令格式	机器指令格式	操作
MOV direct,A	**F5H direct**	**direct ← (A)**
MOV direct,Rn	**88H～8FH direct**	**direct ← (Rn)**
MOV direct2,direct1	**85H direct1 direct2**	**direct2 ← (direct1)**
MOV direct,@Ri	**86H～87H direct**	**direct ← ((Ri))**
MOV direct,#data	**75H direct data**	**direct ← # data**

注意：“**MOV direct2，direct1**”指令在译成机器码时，源地址在前，目的地址在后。如“**MOV 50H，90H**”的机器码为“**85 90 50**”。

3.2.1　内部RAM数据传送指令

4. 间接地址为目的操作数的指令

汇编指令格式	机器指令格式	操　作
MOV　@Ri,A	F6H～F7H	(Ri) ← (A)
MOV　@Ri,direct	A6H～A7H　direct	(Ri) ← (direct)
MOV　@Ri,#data	76H～77H　data	(Ri) ← data

注意：(Ri)表示以Ri中的内容为地址所指定的RAM单元。

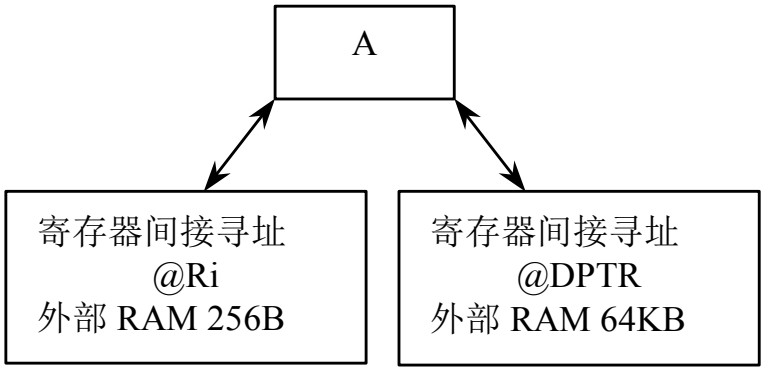
5. 十六位数据传送指令

汇编指令格式	机器指令格式	操　作
MOV　DPTR,#data16	90H dataH dataL	DPH ← dataH, DPL ← dataL

注意：这是唯一的16位立即数传送指令 。

3.2.2 外部RAM传送指令

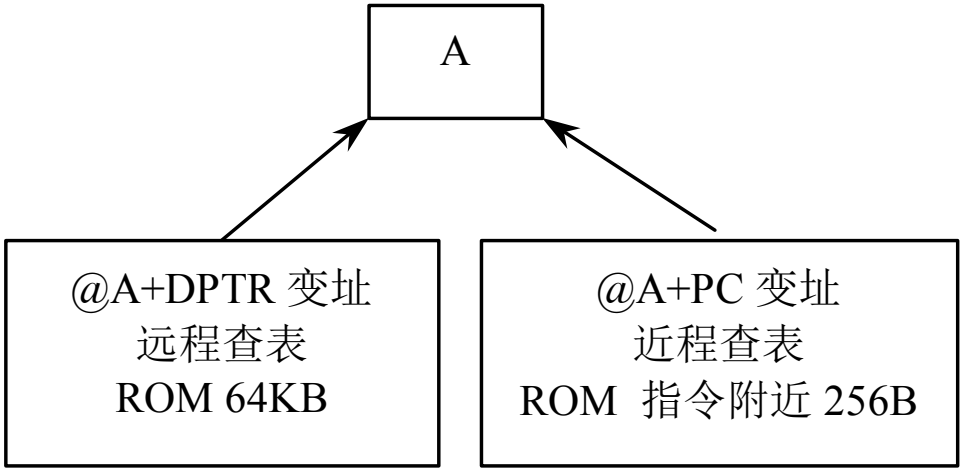
汇编指令格式	机器指令格式	操 作
MOVX @DPTR,A	**F0H**	**外((DPTR)) ← (A)**
MOVX A,@DPTR	**E0H**	**A ← 外((DPTR))**
MOVX @Ri,A	**F2H～F3H**	**外((Ri)) ← (A)**
MOVX A,@Ri	**E2H～E3H**	**A ← 外((Ri))**



注意：8051没有专门的输入/ 输出指令，在访问外部的设备时，可以采用这种方式与外部设备的端口打交道。

3. 2. 3　查表指令

汇编指令格式	机器指令格式	操　作
MOVC A,@A+DPTR	**93H**	**PC (PC)+1,**
		A←((A)+(DPTR))
MOVC A,@A+PC	**83H**	**PC ← (PC)+1,**
		A ← ((A)+(PC))



注意：前一条指令只能查找指令所在位置以后256B范围内的代码或常数，后一条指令查表范围可达整个程序存储器的64KB空间。

3. 2. 4　交换指令

1. 字节交换指令

汇编指令格式	机器指令格式	操　作
XCH　A,Rn	**C8～CFH**	**(A) ↔ (Rn)**
XCH　A,direct	**C5H**	**(A) ↔ (direct)**
XCH　A,@Ri	**C6H～C7H**	**(A) ↔ ((Ri))**

注意：该操作只影响标志位P。

2. 半字节交换指令

汇编指令格式	机器指令格式	操　作
XCHD　A,@Ri	**D6H～D7H**	**(A)0～3 ↔ ((Ri))0～3**

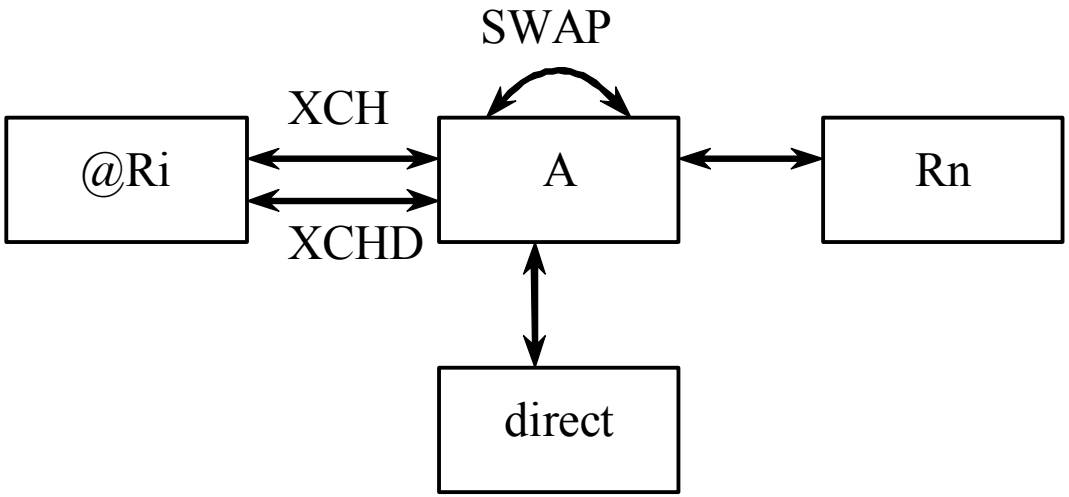
注意：该操作只影响标志位P。

3.2.4 交换指令

3. 累加器半字节交换指令

汇编指令格式	机器码格式	操 作
SWAP A	**C4H**	**(A)0～3 ↔ (A)4～7**

注意：该操作不影响标志位P。



3.2.5 堆栈操作指令

1. 入栈指令

汇编指令格式	机器指令格式	操 作
PUSH direct	**C0H direct**	**SP ← (SP)+1,** **(SP) ← (direct)**

注意：堆栈指针**SP**先加**1**，并指向栈顶的上一个空单元，然后再将直接地址**(direct)**寻址的单元内容压入当前**SP**所指示的堆栈单元中。该操作不影响标志位**P**。

2. 出栈指令

汇编指令格式	机器指令格式	操 作
POP direct	**D0H direct**	**direct ← ((SP)),** **SP ← (SP)-1**

注意：先将堆栈指针**(SP)**所指示的内部**RAM**(栈顶)单元中内容送入由直接地址寻址的单元中，然后再将栈指针**(SP)**减**1**并回送**SP**。该操作不影响标志位**P**。

- 【例】 已知：(A)=30H，(B)=70H执行指令：
- MOV SP，#60H；(SP)=60H 设堆栈指针
- PUSH ACC；(SP)←(SP)+1，(SP)=61H，((SP))←(A)
- PUSH B ；(SP)←(SP)+1，(SP)=62H， ((SP))←(B)
- 结果： (61H)=30H，(62H)=70H,(SP)=62H.
- POP B ；(B)←((SP))，(SP)=(SP)–1，(SP)=61H
- POP ACC；(ACC)←((SP))，(SP)=(SP)–1，(SP)=60H
- 结果： (B)=70H，(ACC)=30H，（SP）=60H
- 由于MCS-51单片机堆栈操作指令中的操作数只能使用直接寻址方式，不能使用寄存器寻址方式，所以将累加器(A)压入堆栈时，累加器(ACC)不能简写A。堆栈操作时指令PUSH和POP要成对出现，且先后顺序要相反排列，先进后出，后进先出。

3.2.6 数据传送类指令应用实例

【例5】 使用不同的指令将累加器**A**的内容送至内部**RAM**的**26H**单元。

解：在访问内部**RAM**时，可以有多种寻址方式供选择，在实际应用中要注意根据实际情况选择合适的寻址方式来进行数据传送。可以通过下面指令采用不同寻址方式实现。

```
①  MOV     26H，A        ；目的操作数采用直接寻址，
                          源操作数采用寄存器寻址
②  MOV     R0，#26H
   MOV     @R0，A        ；目的操作数采用寄存器间接寻址，
                          源操作数采用寄存器寻址
③  MOV     26H，ACC      ；采用直接寻址
④  MOV     26H，0E0H     ；采用直接寻址
⑤  PUSH  ACC             ；利用栈操作，直接寻址
   POP   26H
```

3.2.6 数据传送类指令应用实例

【例6】 已知**(A)=23H**，**(R0)=30H**，内部**RAM(30H)=45H**，**(45H)=56H**，试分析分别执行下列指令后累加器**A**的内容，并写出源操作数的寻址方式。

解：这四条指令代表了数据传送指令中常用的四种寻址方式，在使用时要特别注意不同寻址方式的区别，搞清楚谁是最终的操作数。要注意题中第一条指令与第二条指令、第三条与第四条指令的区别。

指令	结果	寻址方式
MOV A,R0	**(A)=30H**	寄存器寻址
MOV A,@R0	**(A)=45H**	寄存器间接寻址
MOV A,45H	**(A)=56H**	直接寻址
MOV A,#45H	**(A)=45H**	立即数寻址

3.2.6 数据传送类指令应用实例

【例7】将外部RAM 2000H单元的内容传送至外部RAM 3000H单元。

解：8051单片机指令系统中没有外部RAM两个单元直接传送数据的指令，只有外部RAM和累加器的传送指令，要想实现题目中要求的功能，必须通过累加器A进行。

具体程序如下：

```
MOV     DPTR,#2000H     ;送源数据地址
MOVX    A,@DPTR         ;源数据送累加器
MOV     DPTR,#3000H     ;送数据目的地址
MOVX    @DPTR,A         ;累加器内容送目的单元
```

3.2.6　数据传送类指令应用实例

【例8】(A)=20H，(B)=30H，分析执行下面指令以后的结果。

```
PUSH   ACC
PUSH   B
POP    ACC
POP    B
```

解：根据堆栈的“先入后出、后入先出”操作原则进行分析，执行上面指令以后结果为：(A)=30H，(B)=20H。

程序执行结果和初始状态比较，两寄存器内容进行了互换，正是由“先入后出”的存储原则造成的。

在子程序调用时，堆栈经常用来保护现场，利用PUSH保护现场，利用POP恢复现场。恢复现场时，一定要注意POP指令的顺序要和PUSH指令对称，后压入的数据先弹出，先压入的数据后弹出，使得现场正确恢复到原来状态。

3.2.6 数据传送类指令应用实例

【例9】在程序存储器中有一平方表，从2000H单元开始存放，如图所示，试通过查表指令查找出6的平方。

2000H	0
2001H	1
2002H	4
2003H	9
2004H	16
2005H	25
2006H	36
2007H	49
2008H	64
2009H	81

解：采用DPTR作为基址寄存器的查表程序比较简单，查表范围大，也容易理解。只要预先使用一条16位数据传送指令，把表的首地址2000H送入DPTR，然后进行查表就可以了。

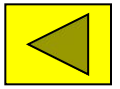
相应的程序如下：

```
MOV  A,#6                ;设定备查的表项
MOV  DPTR,#2000H         ;设置DPTR为表始址
MOVC  A,@A+DPTR          ;将A的平方值查表后送A
```

如果需要查找其他数的平方，只需要将累加器A的内容(变址)改一下即可。

3.3　算术运算指令

- 3.3.1　加法、减法指令
- 3.3.2　乘法、除法指令
- 3.3.3　加1、减1指令
- 3.3.4　十进制调整指令
- 3.3.5　算术运算类指令应用实例



3.3.1 加法、减法指令

1. 加法类指令

汇编指令格式	机器指令格式	操 作
ADD A,Rn	**28H～2FH**	**A ← (A)+(Rn)**
ADD A,direct	**25H direct**	**A ← (A)+(direct)**
ADD A,@Ri	**26H～27H**	**A ← (A)+((Ri))**
ADD A,#data	**24H data**	**A ← (A)+#data**

注意：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。如果第6位向第7位有进位而第7位没有向前进位，或者如果第7位向前有进位而第6位没有向第7位进位，OV=1，否则OV=0。该操作也影响标志位P。

3. 3. 1　加法、减法指令

2. 带进位加法指令

汇编指令格式	机器指令格式	操　作
ADDC　A,Rn	**38H～3FH**	**A ← (A) +(Rn) +CY**
ADDC　A,direct	**35H direct**	**A ← (A)+(direct)+CY**
ADDC　A,@Ri	**36H～37H**	**A ← (A)+((Ri))+CY**
ADDC　A,#data	**34H data**	**A ← (A)+#data+CY**

注意：本指令的执行将影响标志位**AC、CY、OV、P**，与**ADD**指令相同 。

3.3.1 加法、减法指令

3. 带借位减法指令

汇编指令格式	机器指令格式	操 作
SUBB A,Rn	**98H～9FH**	**A ← (A)-CY-(Rn)**
SUBB A,direct	**95H direct**	**A ← (A)-CY-(direct)**
SUBB A,@Ri	**96H～97H**	**A ← (A)-CY-((Ri))**
SUBB A,#data	**94H data**	**A ← (A)-CY-#data**

注意：在执行不带借位的运算时，可在“**SUBB**”指令前用“**CLR C**”指令将**CY**清**0**。如果第**7**位有借位，则**CY**置**1**，否则清**0**。若第**3**位有借位，则**AC**置**1**；否则清**0**。两个带符号数相减，还要考查**OV**标志，若**OV**为**1**，表示差数溢出，即破坏了正确结果的符号位。该操作也影响标志位**P**。

3.3.2 乘法、除法指令

1. 乘法指令

汇编指令格式	机器指令格式	操 作
MUL AB	**A4H**	**BA ← (A)×(B)**

注意：若乘积大于**0FFH**，则**OV**置**1**，否则清**0**(此时**B**的内容为**0**)。**CY**总是被清**0**。该操作也影响标志位**P**。

2. 除法指令

汇编指令格式	机器指令格式	操 作
DIV AB	**84H**	**A←(A)÷(B)**的商， **B←(A)÷(B)**的余数

注意：若除数**(B)=00H**，则结果无法确定，则**OV**置**1**。 **CY**总是被清**0**。该操作也影响标志位**P**。

3.3.3 加1、减1指令

1. 加1指令

汇编指令格式	机器指令格式	操 作
INC A	**04H**	**A ← (A)+1**
INC Rn	**08H～0FH**	**Rn ← (Rn)+1**
INC direct	**05H direct**	**direct ← (direct)+1**
INC @Ri	**06H～07H**	**(Ri) ← ((Ri))+l**
INC DPTR	**A3H**	**DPTR ← (DPTR)+1**

注意：该操作不影响**PSW**标志位。

3. 3. 3　加1、减1指令

2. 减1指令

汇编指令格式	机器指令格式	操　作
DEC　A	**14H**	**A ← (A)-1**
DEC　Rn	**18H～1FH**	**Rn ← (Rn)-l**
DEC　direct	**15H direct**	**direct ← (direct)-1**
DEC　@Ri	**16H～17H**	**(Ri) ← ((Ri))-1**

注意：该操作不影响**PSW**标志位。

3.3.4　十进制调整指令

汇编指令格式	机器码格式	操　作
DA　A	**D4H**	**调整累加器A内容为BCD码**

注意：这条指令一般跟在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（不能用于十进制减法的调整 ）。调整方法如下：

- **若$(A_{0\text{-}3})>9$或AC=1，则$(A_{0\text{-}3})+6\rightarrow A_{0\text{-}3}$**
- **若$(A_{4\text{-}7})>9$或CY=1，则$(A_{4\text{-}7})+6\rightarrow A_{4\text{-}7}$**

该操作影响标志位P。

3.3.5 算术运算类指令应用实例

【例13】 有两个**BCD**码表示的**4**位十进制数，分别存放在内部数据存储器的**50H～51H**单元和**60H～61H**单元，试编写程序求这两个数之和，并将结果存放在**40H～42H**单元。

解：求两个**BCD**数之和的运算程序如下：

```
MOV   A,50H          ;取第一个数低2位BCD码
ADD   A,60H          ;加第二个数低2位BCD码
DA   A               ;十进制调整
MOV   40H,A          ;保存结果的低2位
MOV   A,51H          ;取高位BCD码
ADDC   A,61H         ;高位相加
DA   A               ;十进制调整
MOV   41H,A          ;保存结果的高2位
MOV    A,#00H
ADDC   A,#00H        ;计算进位
MOV    42H,A         ;保存进位
```

3.3.5　算术运算类指令应用实例

【例14】 试编程计算**5678H-1234H**的值，结果保存在**R6**、**R5**中。

解：减数和被减数都是**16**位二进制数，计算时要先进行低**8**位的减法，然后再进行高**8**位的减法，在进行低**8**位减法时，不需要考虑借位，所以要在减法指令之前将借位标志清**0**。程序如下：

```
MOV   A,#78H          ;被减数低8位送累加器
CLR     C                   ;清进位标志位CY
SUBB  A,#34H          ;减去减数
MOV   R5,A                  ;保存低8位
MOV   A,#56H          ;被减数高8位送累加器
SUBB  A,#12H          ;减去减数
MOV   R6,A                  ;保存高8位
```

3.3.5 算术运算类指令应用实例

【例15】试分析执行下列指令以后，寄存器和内部**RAM**的状态。

```
MOV   R0,#30H          ;数30H送入R0
MOV   30H,#40H         ;数40H送入片内RAM 30H单元
MOV   31H,#50H         ;数50H送入片内RAM 31H单元
INC   @R0              ;将片内RAM 30H单元中的内容加1
INC   R0               ;将R0中的内容加1
INC   @R0              ;将片内RAM 31H单元中的内容加1
```

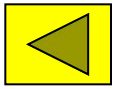
解：分析结果：**(R0)=31H**

(30H)=41H

(31H)=51H

3.4　逻辑操作指令

- 3.4.1　逻辑“与”、“或”、“异或”指令
- 3.4.2　清零、取反指令
- 3.4.3　循环移位指令
- 3.4.4　逻辑运算类指令应用实例



3.4.1 逻辑“与”、“或”、“异或”指令

1. 逻辑“与”指令

汇编指令格式	机器指令格式	操 作
ANL A,Rn	58H～5FH	A ← (A)∧(Rn)
ANL A,direct	55H direct	A ← (A)∧(direct)
ANL A,@Ri	56H～57H	A ← (A)∧((Ri))
ANL A,#data	54H data	A ← (A)∧#data
ANL direct,A	52H direct	direct ← (direct)∧(A)
ANL direct,#data	53H direct data	direct ← (direct)∧#data

注意：后2条指令若直接地址正好是I/O端口P0～P3，则为端口的“读—改—写”操作。前4条指令的操作影响标志位P。

3.4.1 逻辑“与”、“或”、“异或”指令

2. 逻辑“或”指令

汇编指令格式	机器指令格式	操 作
ORL A,Rn	48H～4FH	A ← (A)∨(Rn)
ORL A,direct	45H direct	A ← (A)∨(direct)
ORL A,@Ri	46H～47H	A ← (A)∨((Ri))
ORL A,#data	44H data	A ← (A)∨#data
ORL direct,A	42H direct	direct ← (direct)∨(A)
ORL direct,#data	43H direct data	direct ← (direct)∨#data

注意：后2条指令若直接地址正好是I/O端口P0～P3，则为端口的“读—改—写”操作。前4条指令的操作影响标志位P。

3.4.4 逻辑“与”、“或”、“异或”指令

3. 逻辑“异或”指令

汇编指令格式	机器指令格式	操 作
XRL A,Rn	68H～6FH	A ← (A)⊕ (Rn)
XRL A,direct	65H direct	A ← (direct)⊕ (A)
XRL A,@Ri	66H～67H	A ← (A)⊕ ((Ri))
XRL A,#data	64H data	A ← (A)⊕ #data
XRL direct,A	62H direct	direct ← (direct)⊕ (A)
XRL direct,#data	63H direct data	direct ← (direct)⊕ #data

注意：后2条指令若直接地址正好是I/O端口P0～P3，则为端口的“读—改—写”操作。前4条指令的操作影响标志位P。

3.4.2 清零、取反指令

1. 累加器A清0指令

汇编指令格式	机器指令格式	操 作
CLR A	**E4H**	**A ← 0**

注意：该操作影响标志位P。

2. 累加器A取反指令

汇编指令格式	机器指令格式	操 作
CPL A	**F4H**	**A ← ( )**

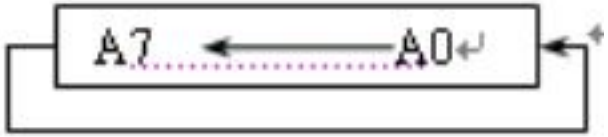
对累加器A的内容逐位取反，不影响标志位。

注意：该操作不影响标志位P。

3.4.3　循环移位指令

1. 累加器A循环左移指令

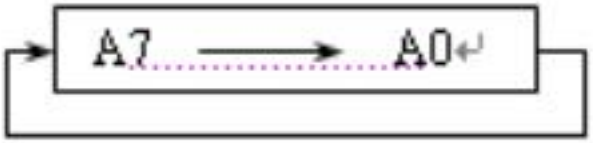
汇编指令格式	机器指令格式	操　作
RL A	**23H**	



注意：该操作不影响**PSW**标志位。

2. 累加器A循环右移指令

汇编指令格式	机器指令格式	操　作
RR A	**03H**	

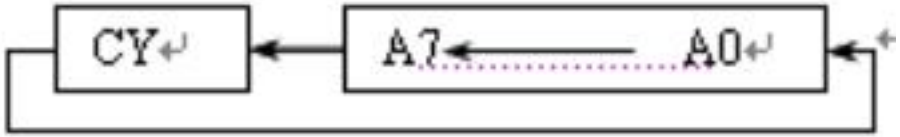


注意：该操作不影响**PSW**标志位。

3.4.3　循环移位指令

3. 累加器**A**带进位循环左移指令

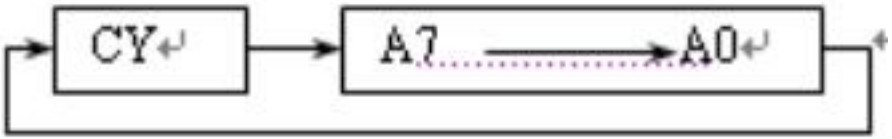
汇编指令格式	机器指令格式	操　作
RLC　A	**33H**	



注意：该操作影响标志位**P**和**CY**。

4. 累加器**A**带进位循环右移指令

汇编指令格式	机器指令格式	操　作
RRC　A	**23H**	



注意：该操作影响标志位**P**和**CY** 。

3.4.4　逻辑运算类指令应用实例

【例16】 将P1口的P1.2、P1.3、P1.7清零，其余位不变。

解：相应的指令为：

```
ANL  P1,#01110011B
```

【例17】 利用逻辑运算指令将P1口的P1.1、P1.3、P1.5置1，其余位保持不变。

解：相应的指令为：

```
ORL  P1,#00101010B
```

【例18】 利用逻辑运算指令，将内部RAM中40H单元的1、3、5、7位取反，其他位保持不变。

解：相应指令为：

```
XRL  40H,#0AAH                ;0AAH=10101010B
```

3.4.4　逻辑运算类指令应用实例

【例19】利用逻辑运算指令将当前工作寄存器设定为第3组工作寄存器。

解：相应指令为：

ORL　PSW,#00011000B

【例20】无符号8位二进制数(A)=00111101B=3DH，CY=0。试分析执行“RLC　A”指令后累加器A的内容。

解：执行指令“RLC　A”的结果为

(A)=01111010B=7AH

CY=0

7AH正是**3DH**的**2**倍，该指令执行的是乘**2**操作。

3.4.4　逻辑运算类指令应用实例

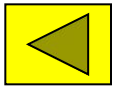
【例21】 拆字程序：在内部**RAM 40H**单元保存有以压缩**BCD**码表示的**2**位十进制数，编程将它们拆开，分别保存在内部**RAM**的**41H**、**42H**单元。

解：程序如下：

```
MOV   A,40H          ;压缩BCD码送累加器
ANL   A,#0FH         ;高4位清0,保留低4位
MOV   41H,A          ;保存低4位BCD码
MOV   A,40H          ;取数据
ANL   A,#0F0H        ;低4位清0,保留高4位
SWAP   A             ;高低位交换
MOV   42H,A          ;保存高4位BCD码
```

3.5　控制程序转移类指令

- 3.5.1　无条件转移指令
- 3.5.2　条件转移指令
- 3.5.3　调用、返回指令
- 3.5.4　空操作指令
- 3.5.5　控制转移类指令应用实例



3.5.1　无条件转移指令

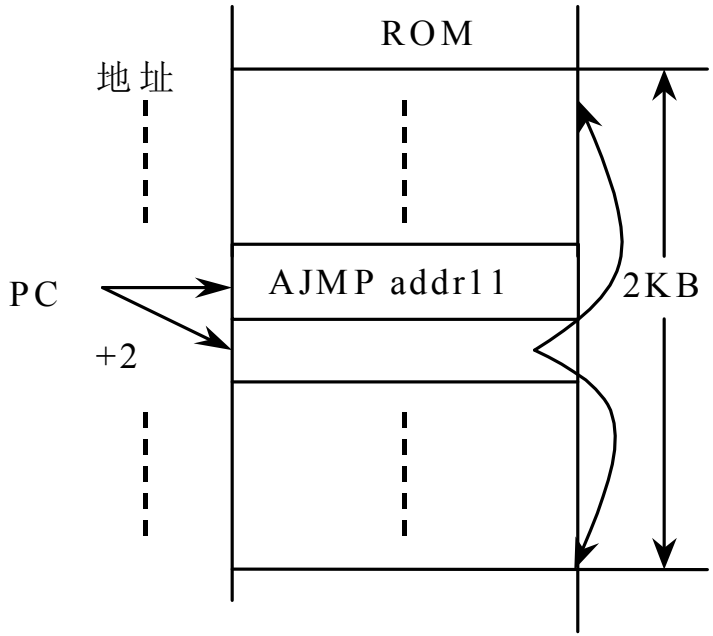
- 无条件转移指令是指当程序执行到该指令时，程序无条件转移到指令所提供的地址处执行。

- 无条件转移类指令有四类:
- 绝对转移——AJMP
- 长转移——LJMP
- 相对转移(短转移)——SJMP
- 间接转移(散转指令)——JMP

3.5.1 无条件转移指令

1. 绝对转移指令

汇编指令格式	机器指令格式	操 作
AJMP addr11	**a10a9a800001 a7～a0**	**PC←(PC)+2,** **PC10～0←addrll,** **(PC15～11)不变**



注意：本条指令提供11位地址，可在该指令后面一个存储单元所在的2KB区域内无条件转移。

- 绝对转移指令是一条双字节双周期指令，11位地址addr11（a10—a0）在指令中的分布是：
a10 a9 a8 0 0 0 01a7 a6 a5 a4 a3 a2 a1 a0，其中，00001B是操作码。在程序设计中，11位地址也可以用符号表示，但在上机执行前必须按照上述指令格式加以代真。
绝对转移指令执行时分为两步：
第一步是取指令操作，程序计数器PC中内容被加1两次；
第二步是把PC加2后的高5位地址PC15—PC11和指令代码中低11位构成目标转移地址：PC15—PC11 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 a0
- 其中，a10—a0的地址范围是全“0”——全“1”。因此，绝对转移指令可以在2KB范围内向前或向后跳转。

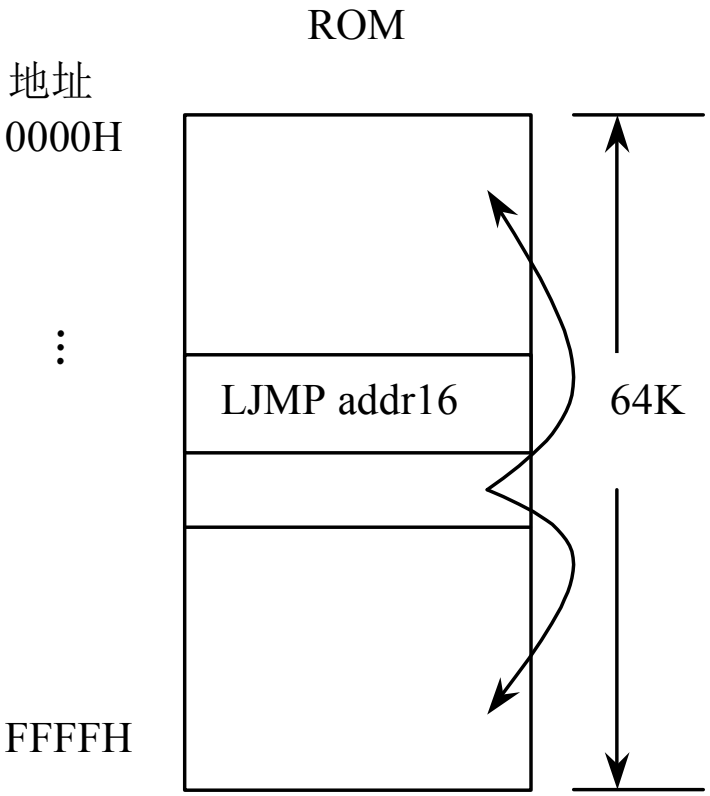
- 如果把单片机64KB寻址区分成32页（每页2KB），则PC15—PC11（00000B—11111B）称为页面地址（即：0页—31页），a10—a0称为页内地址，但应注意：AJMP指令的目标转移地址不是和AJMP指令地址在同一个2KB区域，而是应和AJMP指令取出后的PC地址（即：PC+2）在同一个2KB区域。例如：若AJMP指令地址为2FFEH，则PC+2=3000H，故目标转移地址必在3000H—37FFH这2KB区域中。

- 例如：MGH2001：AJMP addr11，
  其中，MGH2001为AJMP addr11指令的标号地址，由该指令在程序存储器中的位置确定，addr11为11位地址，试分析该指令执行后的情况以及指令码的确定方法。
  解：设MGH2001=3100H，addr11=10110100101B，则根据上述指令码格式可得绝对转移指令的格式码为：1 0 1| 0 0 0 0 1|1 0 1 0 0 1 0 1|（a10 a9 a8|操作码|a7—a0|）
  即：A1A5H。该指令执行后：
  PC15——PC11 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 a0
  PC= 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 1 B =35A5H
  即：程序转移到35A5H处执行。

3.5.1 无条件转移指令

2. 长转移指令

汇编指令格式	机器指令格式	操 作
LJMP addr16	**02H addr16**	**PC ← addrl6**



注意：本条指令提供16位目的地址，所以程序可转向64KB程序存储器地址空间的任何单元。

3.5.1 无条件转移指令

3. 相对转移(短转移)指令

汇编指令格式	机器指令格式	操 作
SJMP rel	**80H rel**	**PC ← (PC)+2,** **PC ← (PC)+rel**

注意：本条指令的操作数是相对地址，**rel**是一个带符号的偏移量(补码)，其范围为**-128～+127**共**256**字节。负数表示反向转移，正数表示正向转移。如果指令中偏移量**rel=FEH**，因为**FEH**是**-2**的补码，所以转移目的地址**=PC+2-2=PC**，结果转向自己，导致无限循环。这条指令称为原地踏步指令，即程序执行到这条指令时，不再向下执行，而在该指令处原地踏步。

3.5.1　无条件转移指令

4. 间接转移指令(散转指令)

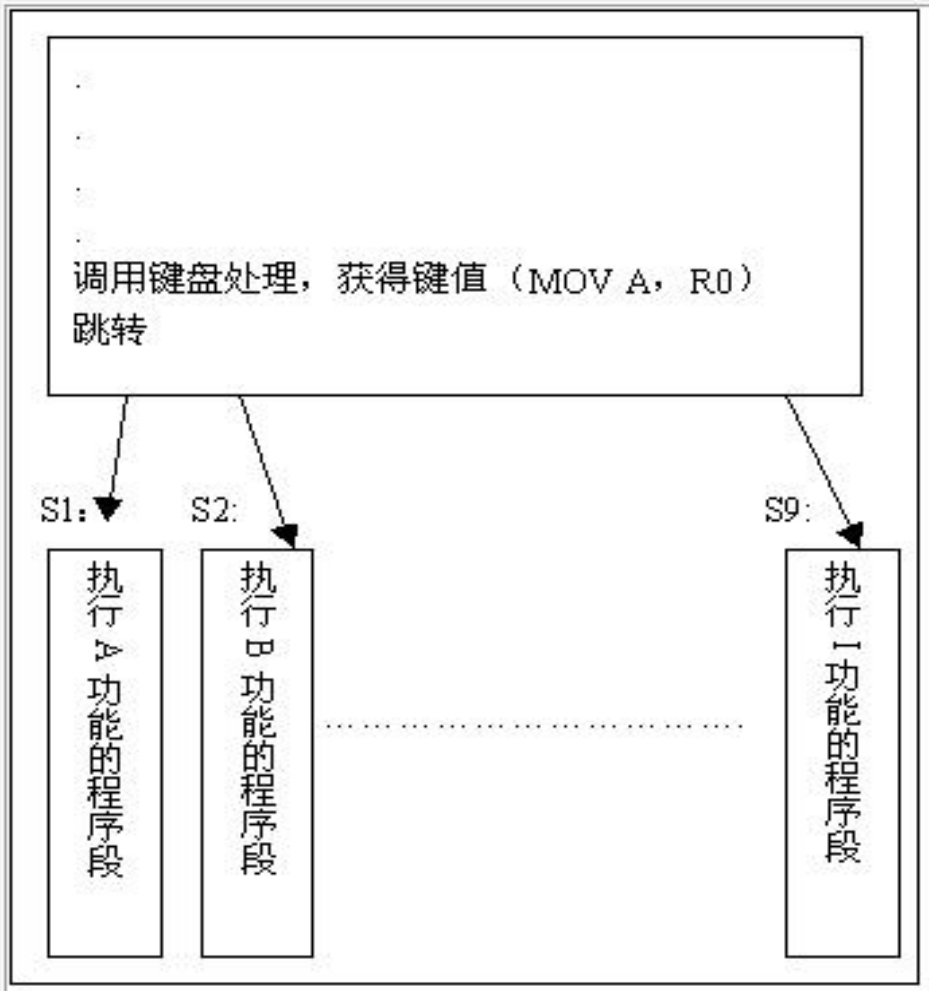
汇编指令格式	机器指令格式	操　作
JMP　@A+DPTR	**73H**	**PC ← (A)+(DPTR)**

注意：该指令采用的是变址寻址方式，指令执行过程对**DPTR**、**A**和标志位均无影响。这条指令可以根据累加器**A**的不同值实现多个方向的转移，可代替众多的判断跳转指令，具有散转功能，所以又称散转指令。

这条指令的用途也是跳转，转到什么地方去呢？这可不能由标号简单地决定了。让我们看看一个实际的例子。

- **MOV DPTR，#TAB**
- **MOV A，R0**
- **MOV B，#2**
- **MUL A，B**
- **JMP @A+DPTR**

- **TAB: AJMP S1**
- **AJMP S3**
- **……**



A	B	C
D	E	F
G	H	I

一种常用键盘的示意图

图 1

ROM		地址
.		.
.		.
.		.
AJMP	S4	TAB+6
		TAB+5
AJMP	S3	TAB+4
		TAB+3
AJMP	S2	TAB+2
		TAB+1
AJMP	S1	TAB

在单片机开发中，经常要用到键盘，见上面的9个按键的键盘。我们的要求是：当按下功能键A.....G时去完成不同的功能。这用程序设计的语言来表达的话，就是：按下不同的键去执行不同的程序段，以完成不同的功能。怎么样来实现呢？

前面的程序读入的是按键的值，如按下'A'键后获得的键值是0，按下'B'键后获得的值是'1'等等，然后根据不同的值进行跳转，如键值为0就转到S1执行，为1就转到S2执行。。。。如何来实现这一功能呢？

先从程序的下面看起，是若干个AJMP语句，这若干个AJMP语句最后在存储器中是这样存放的（见图3），也就是每个AJMP语句都占用了两个存储器的空间，并且是连续存放的。而AJMP S1存放的地址是TAB，到底TAB等于多少，我们不需要知道，把它留给汇编程序来算好了。

下面我们来看这段程序的执行过程：第一句MOV DPTR，#TAB执行完了之后，DPTR中的值就是TAB，第二句是MOV A，R0，我们假设R0是由按键处理程序获得的键值，比如按下A键，R0中的值是0，按下B键，R0中的值是1，以此类推，现在我们假设按下的是B键，则执行完第二条指令后，A中的值就是1。并且按我们的分析，按下B后应当执行S2这段程序，让我们来看一看是否是这样呢？第三条、第四条指令是将A中的值乘2，即执行完第4条指令后A中的值是2。下面就执行JMP @A+DPTR了，现在DPTR中的值是TAB，而A+DPTR后就是TAB+2，因此，执行此句程序后，将会跳到TAB+2这个地址继续执行。看一看在TAB+2这个地址里面放的是什么？就是AJMP S2这条指令。因此，马上又执行AJMP S2指令，程序将跳到S2处往下执行，这与我们的要求相符合。

这样我们用JMP @A+DPTR就实现了按下一键跳到相应的程序段去执行的这样一个要求。再问大家一个问题，为什么取得键值后要乘2？如果例程下面的所有指令换成LJMP，即：

LJMP S1,LJMP S2……这段程序还能正确地执行吗？如果不能，应该怎么改？

3.5.2　条件转移指令

- 条件转移指令是指根据给出的条件进行判断，
  若条件满足，则程序转向由偏移量确定的目的地址处去执行。
  若条件不满足，程序将不会转移，而是按原顺序执行。

- 8051有丰富的条件转移指令。

3.5.2 条件转移指令

1. 累加器判零转移指令

汇编指令格式	机器指令格式	操 作
JZ rel	**60H rel**	**PC ← (PC)+2**
		若A=0,则程序转移PC←(PC)+rel
		若A≠0,则程序往下顺序执行
JNZ rel	**50H rel**	**PC ← (PC)+2**
		若A≠0,则程序转移PC←(PC)+rel
		若A=0,则程序往下顺序执行

注意：相对偏移量为一个带符号的8位数，偏移范围为**-128～+127**，共**256**个字节。本指令不改变累加器**A**的内容，也不影响任何标志位。

3.5.2 条件转移指令

2. 比较转移指令

汇编指令格式	机器指令格式	操 作
CJNE A,direct,rel	**B5H direct rel**	累加器内容和直接寻址单元比较
CJNE A,#data,rel	**B4H data rel**	累加器和立即数比较
CJNE Rn,#data,rel	**B6H～B7H data rel**	寄存器内容和立即数比较
CJNE @Ri,#data,rel	**B8H～BFH data rel**	间接寻址单元内容和立即 数比较

注意：若目的操作数=源操作数，程序顺序执行，**CY=0**；

若目的操作数>源操作数，程序转移，**PC←(PC)+rel**，并且**CY=0**；

若目的操作数<源操作数，程序转移，**PC←(PC)+rel**，并且**CY=1**。本指令执行后不影响任何操作数。

3.5.2 条件转移指令

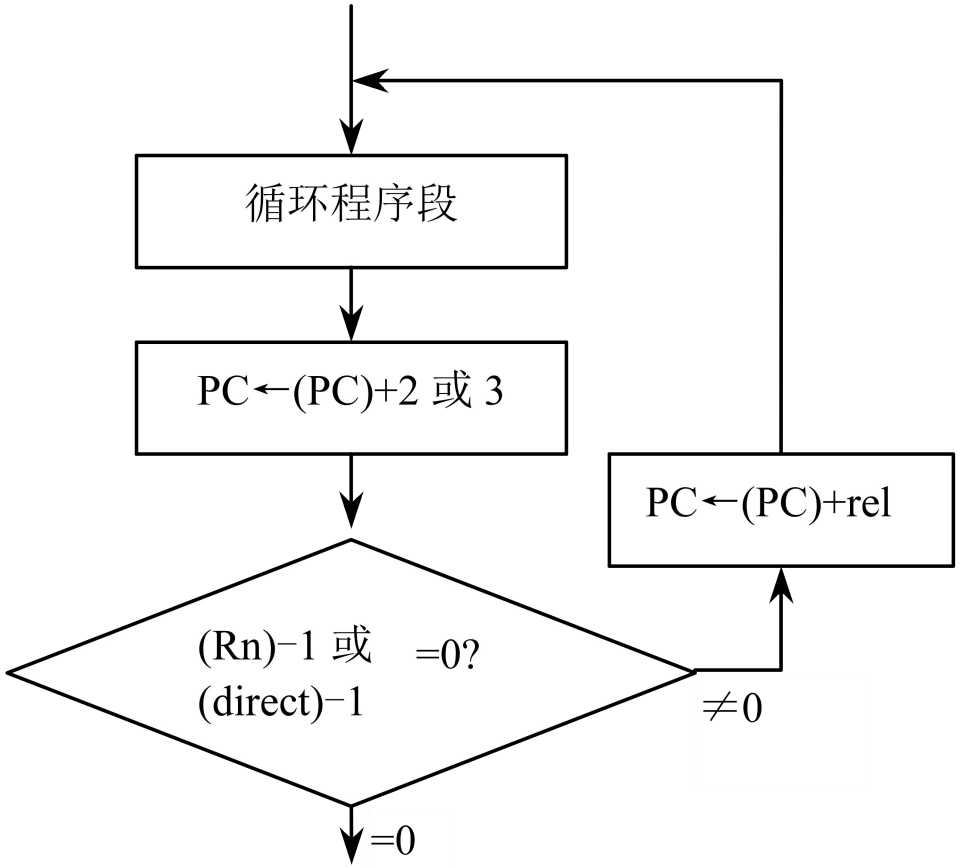
3. 减1非0转移指令(循环转移指令)

汇编指令格式	机器指令格式	操 作
DJNZ Rn,rel	**D8H～DFH rel**	**PC ← (PC)+2,** **Rn ← (Rn)-1;** **若(Rn)≠0,则程序转 PC←(PC)+rel;** **若(Rn)=0,则程序往下顺序执行**
DJNZ direct,rel	**D5H direct rel**	**PC ←(PC)+3,** **direct ← (direct)-1** **若(direct)≠0,则程序转移** **PC←(PC)+rel;** **若(direct)=0,则程序往下顺序执行**

3.5.2 条件转移指令

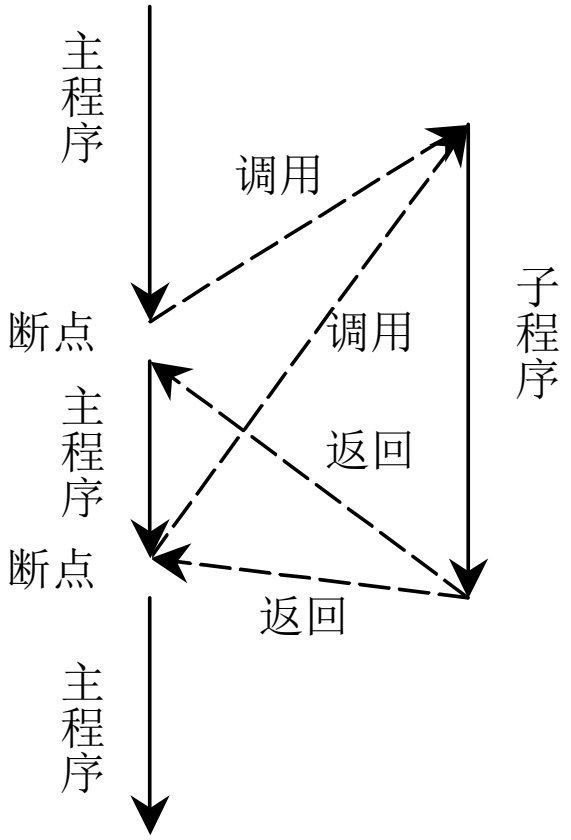
3. 减1非0转移指令(循环转移指令)

注意：在一般的应用中，经常把rel设为负值，使得程序负向跳转。通过改变指令中Rn或者direct单元的内容，就可以控制程序负向跳转的次数，也就控制了程序循环的次数，所以该指令又称为循环转移指令。



3.5.3　调用、返回指令

- 通常把具有一定功能的公用程序段作为子程序，在主程序中采用调用指令调用子程序，子程序的最后一条指令为返回主程序指令(RET)。
- 8051指令系统中有两条调用指令，分别是绝对调用和长调用指令。
- 主程序调用子程序及从子程序返回主程序的过程如右图所示。



3.5.3　调用、返回指令

1. 绝对调用指令

汇编指令格式	机器指令格式	操　作
ACALL addr11	**a10a9a810001 a7～a0**	**PC ← (PC)+2** **SP ← (SP)+1, (SP) ← (PC0～PC7)** **SP ← (SP)+1 ,(SP) ← (PC8～ PC 15)** **PC0～PC10 ← addr0～10,PC11～PC15不变**

注意：本指令提供11位子程序目的地址，调用地址的形成方法和绝对转移指令类似；被调用的子程序首地址必须在ACALL指令后一字节所在的2KB范围内的程序存储器中。

3.5.3 调用、返回指令

2. 长调用指令

汇编指令格式	机器指令格式	操 作
LCALL addr16	**12H addr16**	**PC ← (PC)+3** **SP ← (SP)+1 ,(SP) ← (PC0～PC7)** **SP ← (SP)+1 ,(SP) ← (PC8～PC15)** **PC ← addr16**

注意：本指令提供16位子程序目的地址，被调用的子程序首地址可设置在64KB程序存储器地址空间的任何位置。

3.5.3 调用、返回指令

3. 返回指令

汇编指令格式 机器指令格式 操 作

RET 22H PC8～15 ← ((SP)), 弹出断点高8位

SP ← (SP)-1

PC0～7 ← ((SP)), 弹出断点低8位

SP ← (SP)-1

注意：本指令的作用是从子程序返回。当程序执行到本指令时，表示结束子程序的执行，返回调用指令(ACALL或LCALL)的下一条指令处(断点)继续往下执行。因此，它的主要操作是将栈顶的断点地址送PC，于是，子程序返回主程序继续执行。

3.5.3　调用、返回指令

3. 返回指令

汇编指令格式	机器指令格式	操　作
RETI	**32H**	**PC8～15 ← ((SP)), 弹出断点高8位**
		SP ← ((SP)-1
		PC0～7 ← ((SP)), 弹出断点低8位
		SP ← (SP)-1

注意：本指令是中断返回指令，除具有**RET**指令的功能外，还具有开放低优先级中断、恢复中断逻辑等功能。在编程时不能将两种返回指令混用，中断返回一定要安排在中断服务程序的最后。

3. 5. 4　空操作指令

汇编指令格式	机器指令格式	操　作
NOP	**00H**	**PC ← (PC)+1**

注意：这是一条单字节指令，除**PC**加**1**指向下一条指令以外，它不执行其他任何操作，不影响其他寄存器和标志位。**NOP**指令常用来产生一个机器周期的延迟，用来编写软件延时程序。

3.5.5　控制转移类指令应用实例

【例22】 在累加器A中保存有命令键键值，编写程序使程序根据键值不同而转向不同的子程序入口。

解：本题可以采用散转指令，程序如下：

```
    KEY:  CLR  C                      ;清进位
          RLC   A                      ;键值乘2
          MOV  DPTR,# KEYTAB;DPTR指向命令键跳转表首址
          JMP  @A+DPTR                 ;散转到命令键入口
KEYTAB:  AJMP   KEYPR0                 ;转向0号键处理程序
          AJMP   KEYPR1                ;转向1号键处理程序
          AJMP   KEYPR2                ;转向2号键处理程序
          ……
```

从程序中看出，当(A)=00H时，散转到KEYPR 0；当(A)=01H，散转到KEYPR I……。由于AJMP是双字节指令，转移表中相邻的AJMP指令地址相差2个字节，所以散转前应先将A中键值乘2。

3.5.5　控制转移类指令应用实例

【例23】设(SP)=30H，符号地址PROG1指向程序存储器的5678H单元，当前PC值为0123H。从0123H处执行指令“LCALL PROG1”，分析执行后PC、SP的值和相关存储器的内容。

解：执行过程为：

(PC)+3=0123H+3=0126H。

将PC内容压入堆栈：向(SP)+1=31H中压入26H，向(SP)+1=32H中压入01H，(SP)=33H。

将PROG1=5678H送入PC，即(PC)=5678H。程序转向以5678H为首地址的子程序执行。

最终执行结果是：(PC)=5678H、(SP)=33H、(31H)=26H、(32H)=01H。

3.5.5　控制转移类指令应用实例

【例24】编程判断内部RAM 30H单元中的数据是奇数还是偶数，如果是偶数，程序转向PROG0处，如果是奇数程序转向PROG1处(0按照偶数对待)。

解：程序如下：

```
MOV   A,30H              ;数据送累加器
ANL   A,#01H             ;高7位清0,保留最低位
JZ   PROG0               ;如果全为0说明是偶数,转向
                          PROG0
SJMP   PROG1   ;数据为奇数,转向PROG1
```

【例25】利用DJNZ指令和NOP指令编写一循环程序，实现延时1ms(晶振频率为12MHz)。

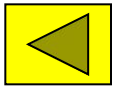
解：程序如下：

```
DELAY:  MOV  A,#0AH                    ;1μs
 LOOP:  MOV  R2,#30H                   ;1μs
        DJNZ R2,$                  ;2×48μs
        DJNZ R1,LOOP     ;1μs× (1+2×48+1)×10
        NOP                           ;1μs
        NOP                           ;1μs
        NOP                           ;1μs
        NOP                           ;1μs
        NOP                           ;1μs
        RET                           ;2μs
```

总的延时时间为：1+(1+2×48+1)×10+7=998μs，若再加上调用本子程序的调用指令所用的时间2μs共1000μs，即1ms。

3.6　位操作类指令

- 3.6.1　位数据传送指令
- 3.6.2　位逻辑运算指令
- 3.6.3　位清0、置1指令
- 3.6.4　位条件转移类指令
- 3.6.5　位操作类指令应用实例



3.6　位操作类指令

- **8051**硬件结构中有个位处理机又称布尔处理机，它具有一套完整的处理位变量的指令集，包括位变量传送、逻辑运算、控制程序转移指令等。
- 在进行位寻址时，**PSW**中的进位标志**CY**作为位处理机的累加器，称为位累加器。
- 位寻址空间包括以下两部分：
- 片内**RAM**中位寻址区——即字节地址**20H**～**2FH**单元中连续的**128**个位，位地址为**00H**～**7FH**。
- 部分特殊功能寄存器中的可寻址位——凡**SFR**中字节地址能被**8**整除的特殊功能寄存器都可以进行位寻址。位地址为**80**～**F7H**，一共**83**位。

3.6.1　位数据传送指令

汇编指令格式	机器指令格式	操　作
MOV　C,bit	**A2H　bit**	**C ← (bit)**
MOV　bit,C	**92H　bit**	**bit ← (C)**

注意：本指令一个操作数为位地址**(bit)**，另一个必定为位累加器**C(**即进位标志位**CY)**。此指令不影响其他寄存器或标志位。

在位操作指令中，位地址**bit**表示方法除前面已讲过的**4**种之外，如果事先用伪指令定义，还可以采用伪指令定义过的字符名称来表示一个可寻址位。

3.6.2 位逻辑运算指令

1. 位逻辑“与”指令

汇编指令格式	机器指令格式	操 作
ANL C,bit	82H bit	C ← (C)∧(bit)
ANL C,/bit	B0H bit	C ← (C)∧(bit)

注意：斜杠“/”表示对该位取反后再参与运算，但不改变原来的数值。8051单片机中没有位逻辑“异或”指令。

2. 位逻辑“或”指令

汇编指令格式	机器指令格式	操 作
ORL C,bit	72H bit	C ← (C)∨(bit)
ORL C,/bit	A0H bit	C ← (C)∨(bit)

注意：斜杠“/”表示对该位取反后再参与运算，但不改变原来的数值。8051单片机中没有位逻辑“异或”指令。

3.6.3 位清0、置1指令

1. 位清0指令

汇编指令格式	机器指令格式	操 作
CLR C	C3H	C ← 0
CLR bit	C2H bit	bit ← 0

注意：本指令执行结果不影响其他标志位。当直接位地址为端口P0～P3中的某一位时，具有“读—改—写”功能。

2. 位置1指令

汇编指令格式	机器指令格式	操 作
SETB C	D3H	C ← 1
SETB bit	D2H bit	bit ← 1

注意：本指令执行结果不影响其他标志位。当直接位地址为端口P0～P3中的某一位时，具有“读—改—写”功能。

3.6.3　位清0、置1指令

3. 位取反指令

汇编指令格式	机器指令格式	操　作
CPL　C	**B3H**	**C ←（　）**
CPL　bit	**B2H　bit**	**bit ←（　）**

注意：本指令执行结果不影响其他标志位。当直接位地址为端口**P0～P3**中的某一位时，具有“读—改—写”功能。

3.6.4 位条件转移类指令

1. 判位累加器C转移指令

汇编指令格式	机器指令格式	操 作
JC rel	**40H rel**	**PC ← (PC)+2**
		若(C)=1,则程序转移PC ← (PC)+rel
		若(C)=0,则程序往下顺序执行
JNC rel	**50H rel**	**PC ← (PC)+2**
		若(C)=0,则程序转移PC ← (PC)+rel
		若(C)=1,则程序往下顺序执行

注意：本指令执行结果不影响**PSW**标志位。

3.6.4 位条件转移类指令

2. 判位变量转移指令

汇编指令格式	机器指令格式	操 作
JB bit,rel	**30H bit rel**	**PC ← (PC)+3**
		若(bit)=1,则程序转移 ← (PC)+rel
		若(bit)=0,则程序往下顺序执行
JNB bit,rel	**20H bit rel**	**PC ← (PC)+3;**
		若(bit)=0,则程序转移
		PC ← (PC)+rel
		若(bit)=1,则程序往下顺序执行

注意：本指令执行结果不影响**PSW**标志位。

3.6.4 位条件转移类指令

3. 判位变量清0转移指令

汇编指令格式	机器指令格式	操 作
JBC bit,rel	**10H bit rel**	**PC ← (PC)+3** **若(bit)=1,则程序转移PC←(PC)+rel,** **且bit←0** **若(bit)=0,则程序往下顺序执行**

注意：本指令执行结果不影响**PSW**标志位。

3.6.5 位操作类指令应用实例

【例26】将P1.4的状态取反后传送给P1.6。

解：相应的指令为：

```
MOV   C,P1.4
CPL   C
MOV   P1.6,C
```

【例27】编程判断内部RAM 30H单元中存放的有符号数是正数还是负数，如果是正数，程序转移到PROP处；如果是负数，程序转移到PRON处；如果是0，程序转移到ZERO处。

解：程序如下：

```
MOV   A,30H              ;取数据
JZ  ZERO                 ;如果为0,转移至ZERO处
JB        ACC.7,PRON     ;ACC.7=1,说明是负数,转
                           移至PRON
SJMP  PROP                 ;否则,是正数,转移至PROP
```

3.6.5 位操作类指令应用实例

【例28】 比较内部**RAM**中**40H**、**41H**两个单元中的数据大小，将大的数送至**42H**单元。

解：程序如下：

```
        MOV   A,40H              ;取第一个数
        CJNE   A,41H,NEQU        ;比较,不相等转移至NEQU处
        MOV   42H,40H            ;相等, (40H)→42H
        SJMP   TOOFF             ;完成,转移至结尾退出
NEQU: JC   LESS                  ;若CY=1,说明(40H)＜(41H)转移
        MOV   42H,40H;(40H)＞(41H),(40H)→40H
        SJMP   TOOFF             ;完成,转移至结尾退出
LESS: MOV  42H,41H               ;(40H)＜(41H),(41H)→40H
TOOFF: RET
```

3.6.5　位操作类指令应用实例

【例29】 利用逻辑运算指令实现逻辑关系：

Y=(A∧B)∨(C∧D)，A、B、C、D均为位变量。

解：将A、B、C、D分别接至P1.0～P1.3，P1.4输出即为Y信号，相应程序如下：

```
MOV   C,P1.0
ANL   C,P1.1          ;A∧B
MOV   00H,C           ;暂存在00H位
MOV   C,P1.2
ANL   C,P1.3          ;C∧D
ORL   C,00H           ;(A∧B)∨(C∧D)
MOV   P1.4,C          ;逻辑关系输出
```

本章小结

- 指令系统是计算机可执行命令的集合，是程序设计的基础。本章主要介绍8051单片机的指令系统。熟悉和掌握指令系统对于单片机的汇编语言程序设计是十分重要的。
- 8051单片机具有功能强大的指令系统，根据功能可分为：数据传送类指令、算术运算类指令、逻辑运算和移位操作指令、控制转移类指令和位操作指令。
- 寻址方式是寻找操作数或操作数地址的方式。要正确理解指令的功能一定要分析指令中操作数是如何获取的，也就是要清楚寻址方式。8051单片机支持多种寻址方式，分别是：寄存器寻址、立即数寻址、直接寻址、寄存器间接寻址、变址寻址、相对寻址、位寻址。要注意区分不同寻址方式的区别，特别是要区分寄存器寻址和寄存器间接寻址、直接寻址和立即数寻址。

本章小结

- 每一种寻址方式都有相应的寻址空间。寄存器寻址可以访问工作寄存器R0～R7、A、B、DPTR，直接寻址可以访问内部RAM低128B和特殊功能寄存器(SFR)，寄存器间接寻址可以访问片内RAM低128B和片外RAM 64KB，变址寻址可以访问程序存储器。要注意特殊功能寄存器(SFR)只能采用直接寻址，片外RAM只能采用寄存器间接寻址。
- 变址寻址一般用于查表指令中，用来查找存放在程序存储器中的常数表格。根据基址寄存器的不同，又可以分为近程查表和远程查表，近程查表用PC作为基址寄存器，远程查表采用DPTR作为基址寄存器。

本章小结

- 数据传送类指令是指令系统中应用最普遍的指令，这类指令是把源地址单元的内容传送到目的地址单元中去，而源地址单元内容不变。数据传送指令分为内部数据传送指令、累加器和外部**RAM**传送指令、查表指令、堆栈操作指令等。外部**RAM**数据传送指令只能通过累加器**A**进行，没有两个外部**RAM**单元之间直接传送数据的指令。堆栈操作指令可以将某一直接寻址单元内容入栈，也可以把栈顶单元弹出到某一直接寻址单元，入栈和出栈要遵循“后入先出”的存储原则。数据传送类指令中还包含了一种交换指令，能将源地址单元和目的地址单元内容互换。

本章小结

- 算术运算指令可以完成加、减、乘、除和加1、减1等运算。加、减、乘、除指令要影响PSW中的标志位CY、AC、OV。乘除运算只能通过累加器A和B寄存器进行。如果是进行BCD码运算，在加法指令后面还要紧跟一条十进制调整指令“DA　A”，它可以根据运算结果自动进行十进制调整，使结果满足BCD码运算规则。

本章小结

- 逻辑运算和移位操作指令可以实现包括清0、置1、取反、逻辑与、逻辑或、逻辑异或等逻辑运算和循环移位操作。逻辑运算是将对应的存储单元按位进行逻辑操作，将结果保存在累加器A中或者是某一个直接寻址存储单元中。如果保存结果的直接寻址单元是端口P0～P3，则为“读—改—写”指令，即：将端口的内容读入CPU进行逻辑运算，然后再回写到端口。

本章小结

- 控制转移类指令是用来控制程序流程的，使用控制转移指令可以实现分支、循环等复杂程序结构，使程序变得巧妙、实用、高效。控制转移指令的特点是修改PC的内容，8051单片机也正是通过修改PC的内容来控制程序流程的。8051的控制转移指令分为无条件转移指令、条件转移指令、子程序调用和返回指令。在使用转移指令和调用指令时要注意转移范围和调用范围。绝对转移和绝对调用的范围是指令下一个存储单元所在的2KB空间。长转移和长调用的范围是64KB空间。采用相对寻址的转移指令转移范围是256B。

本章小结

- 位操作指令又称为布尔操作指令，这类指令是对某一个可寻址位进行清0、取反等操作，或者是根据位的状态进行控制转移。位操作指令采用的是位寻址方式，位寻址的寻址空间分为两部分：一是内部RAM中的位寻址区，即内部RAM的20H～2FH单元，一共128位，位地址是00H～7FH；二是字节地址能被8整除的特殊功能寄存器的可寻址位，共83位。

习　　题

1. 什么是指令系统？8051单片机的指令按照功能分为几种？分别有多少条指令?
2. 什么是寻址方式？8051单片机有哪几种寻址方式?
3. 简述8051单片机汇编语言指令格式和机器语言指令格式。
4. 指出下列指令中源操作数的寻址方式。

   MOV A,#55H
   MOV A,2AH
   MOV C,20H
   MOV A,@R0
   MOV @R0,A
   MOV A,R0
   MOVX A,@DPTR
   MOV DPTR,#0123H
   MOVC A,@A+PC
5. 简述8051的各种寻址方式的寻址空间。
6. 8051若访问特殊功能寄存器，可使用哪种寻址方式?
7. 8051若访问片外RAM单元，可使用哪种寻址方式?
8. 8051若访问片内RAM单元，可使用哪些寻址方式?
9. 8051如果访问程序存储器，应该使用哪种寻址方式？

习　　题

10. 已知(A)=7AH，(R0)=34H，(34H)=A5H，请写出分别执行下面各条指令后累加器A的内容。
    (1) MOV　A，R0
    (2) MOV　A，@R0
    (3) MOV　A，34H
    (4) MOV　A，#34H
11. 已知(A)=70H，(R0)=30H，(30H)=A0H，(PSW)=80H，请写出执行下列各条指令后累加器A和相关存储单元的内容。
    (1) XCH　A，R0
    (2) XCH　A，30H
    (3) XCH　A，@R0
    (4) SWAP　A
    (5) ADD　A，R0
    (6) ADD　A，30H
    (7) ADD　A，#30H
    (8) ADDC　A，30H
    (9) SUBB　A，30H

习　　题

12. 设(R0)=32H，(A)=48H，片内RAM中(32H)=60H，(40H)=61H。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,#40H
```

13. 已知(A)=83H，(R0)=47H，(47H)=34H。请写出执行完下列程序段后A的内容。

```
ANL  A,47H
ORL  47H,A
XRL A,@R0
SWAP  A
```

14. 说明下段程序执行过程中，SP的内容及堆栈中内容的改变过程。

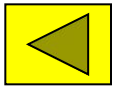
```
MOV  SP,#30H
MOV  30,#55H
MOV  40,#66H
PUSH  30H
PUSH  40H
POP  30H
POP  40H
```

习　　题

15. 编程实现两个16位二进制数的减法。设被减数放在40H、41H单元中，减数放在50H、51H单元，差仍存于被减数地址单元中，减数、被减数都是低地址单元存放低8位。
16. 编写一BCD码拼字程序，将存放在40H、41H单元的两个一位十进制数的BCD码合并构成一个字节的压缩BCD码，并将结果保存在42H单元中(高地址单元的BCD码放在高4位)。
17. 把片外数据存储器4020H单元中的数据读到累加器中，应用哪几条指令?
18. 试编写程序将外部RAM 5000H单元内容传送至外部RAM 6030H单元。
19. 度编写一多字节加法程序，将分别存放在40H、41H和50H、51H的两个16位数相加(高地址单元存放高8位)，结果存放于40H、41H、42H单元中。
20. 试编写一段程序，将累加器A的高4位由P1口的高4位输出，P1口低4位保持不变。
21. 试编写一段程序，将P1口的高5位置位，低3位不变。
22. 试编写一段程序，将R2中的数乘4(用移位指令)。
23. 试编写程序，当累加器A的内容(无符号数)小于10时，程序转NEXT处，否则顺序执行。

习　　题

24. 试编程比较内部RAM 40H、41H单元的无符号数的大小，将较小的数存放在42H单元中。
25. 使用位操作指令实现下列逻辑关系。
    (1)　P1.0=(10H) ∨Pl.1)∧(ACC.0∨CY)
    (2)　P1.3=(ACC.2∧P1.0)⊕ (ACC.1∨P1.1)
26. 试编程将内部RAM 30H～3FH单元中的内容全部清0(利用循环转移指令)。
27. 在程序存储器中有一个常数表，从2040H单元开始分别存放表格的第0项、第1项……第n项，试编程查出表格第m项的内容。



Q & A?

Thanks!