

第3章 MCS-51单片机指令系统

3.1 MCS-51单片机指令系统概述和寻址方式

指令系统：计算机所能执行的指令集合。

机器语言：指令的二进制编码。

汇编语言：指令的符号（助记符）表示。

MCS-51有111条指令：数据传送指令29条，算术运算指令24条，逻辑运算及移位24条，控制转移指令17条，位操作指令17条。

指令格式：

操作码	操作数或操作数地址
-----	-----------

指令的字节数：单字节指令（49条），双字节指令（46条），三字节指令（16条）

寻址方式

寻址方式：寻找操作数的方法，即确定操作数地址的方法。共有7种寻址方式。

1、寄存器寻址方式：操作数在寄存器中，以符号名称表示寄存器。例如：**MOV A, R₀**

寻址范围：①四个寄存器组，共**32**个，使用前设置**PSW**中**RS1**、**RS2**位状态以确定当前寄存器组。
②部分专用寄存器：**A**、**DPTR**等。

2、直接寻址方式：操作数直接以单元地址形式给出。例如：**MOV A, 3AH**

寻址范围：①低**128**单元。②专用寄存器（写出单元地址或寄存器符号）。

3、寄存器间接寻址方式：寄存器中存放操作数的地址。例如：**MOV A, @R₀**，其中@表示寄存器间接寻址。

寻址范围：①内部**RAM**低**128**单元，使用**R₀**、**R₁**间接寻址。②外部**RAM**，**64kB**，使用**DPTR**。③外部**RAM**低**256**单元也可以用**R₀**、**R₁**间接寻址，

如**MOVX A, @R₀**。④**PUSH**、**POP**是以**SP**作间接寻址。

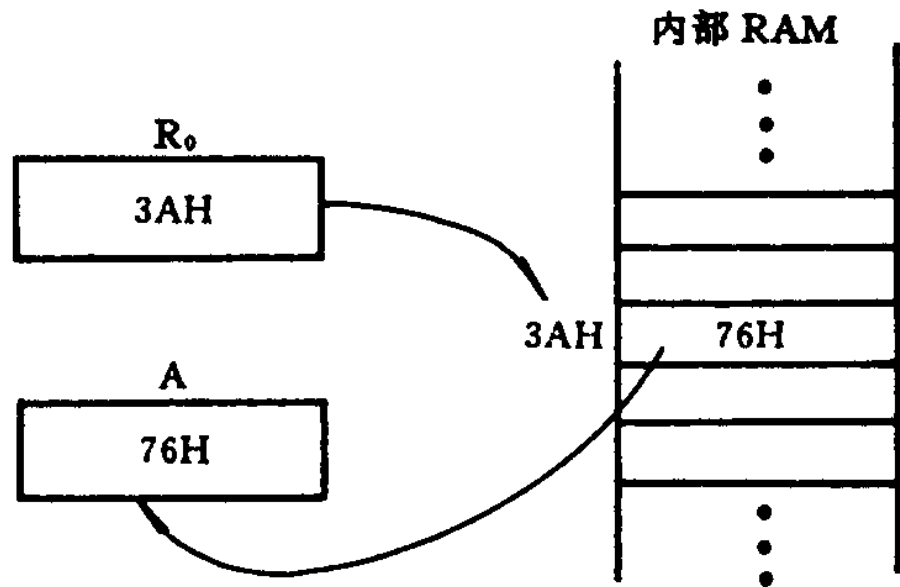


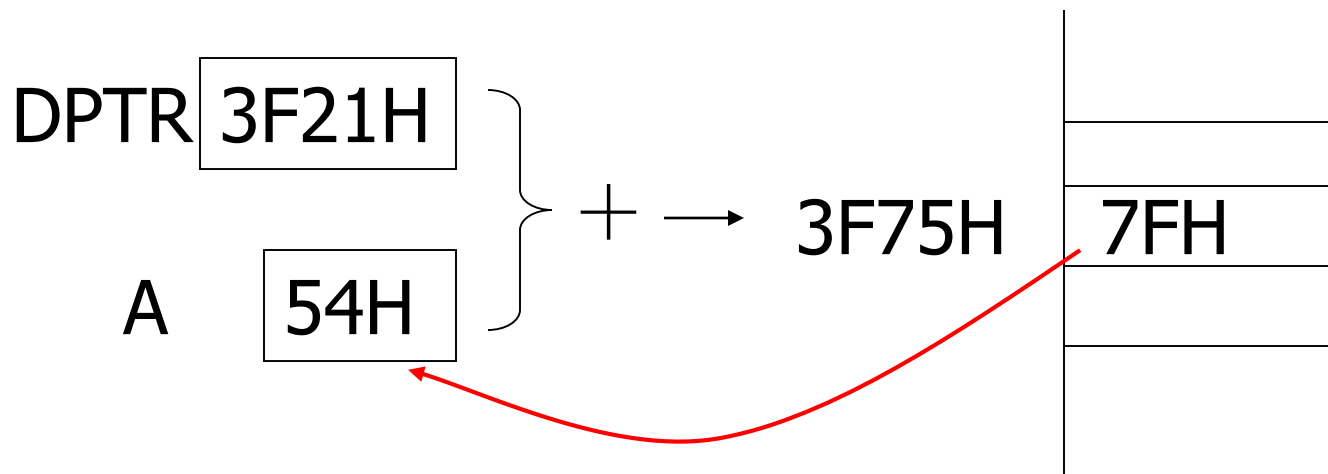
图 3.1 寄存器间接寻址示意图

4、立即寻址方式：操作数在指令中直接给出。

例如：**MOV A, #3AH**，其中**#**表示立即数，与直接地址区别。

5、变址寻址方式：以**DPTR**或**PC**作基址寄存器，以累加器**A**作变址寄存器，并以两者内容相加形成的**16位地址**作为操作数地址。**寻址范围：**只对程序存储器寻址。

例如：**MOVC A, @A+DPTR**。设执行前**(A)=54H**，**(DPTR)=3F21H**，**3F75H**中的内容为**7FH**，则操作数地址为**3F21H+54H=3F75H**，执行后**(A)=7FH**。





6、位寻址方式：指令中使用位地址，对数据位进行操作。

寻址范围：①内部**RAM**位寻址区，**20H~2FH**，共**16**个单元**128**位，位地址**00H~7FH**。见**P18表2_2**。②专用寄存器的可寻址位，**4**种表示方法：

- 直接使用位地址，**P22表2.4**。
- 位名称表示法，**P22表2.4**。
- 单元地址加位，如：**0D0H.5**
- 专用寄存器符号加位，如：**PSW.5**

7、相对寻址方式：实现程序的相对转移。

目的地址 = 转移指令地址 + 转移指令字节数 + **rel**

rel为偏移量，带符号**8**位二进制补码，**-128~+127**

指令中符号的意义

R_n ——通用寄存器， $R_0 \sim R_7$

R_i ——可用作间接寻址的寄存器， $R_0 \sim R_1$

direct ——8位直接地址，内部RAM低128单元，SFR的地址或符号

#data ——8位立即数

#data16 ——16位立即数

addr16 ——16位目的地址

addr11 ——11位目的地址

rel ——偏移量，8位带符号补码

DPTR——数据指针

bit ——直接寻址位

A——累加器

ACC ——直接寻址方式的累加器

B ——寄存器B

C ——进位标志位

@ ——间址寄存器

/——位状态取反

(×) ——寄存器或单元的内容

((×)) ——由×间接寻址的单元中的内容

← ——左边内容被右边内容所取代

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

3.2 数据传送指令

一、内部RAM数据传送指令

1. 立即数传送指令（5条）

MOV A, #data; $A \leftarrow \text{data}$

MOV direct, #data; $\text{direct} \leftarrow \text{data}$

MOV Rn, #data; $R_n \leftarrow \text{data}$

MOV @Ri, #data; $(R_i) \leftarrow \text{data}$

MOV DPTR, #data; $\text{DPTR} \leftarrow \text{data}_{16}$

(DPH \leftarrow 高8位; DPL \leftarrow 低8位)

2. 内部RAM单元之间的数据传送（5条）

MOV direct2, direct1; direct2 \leftarrow (direct1)

MOV direct, Rn; direct \leftarrow (Rn)

MOV Rn, direct; Rn \leftarrow (direct)

MOV direct, @Ri; direct \leftarrow ((Ri))

MOV @Ri, direct; (Ri) \leftarrow (direct)

3. 累加器的数据传送（6条）

MOV A, Rn; $A \leftarrow (Rn)$

MOV Rn, A; $Rn \leftarrow (A)$

MOV A, direct; $A \leftarrow (\text{direct})$

MOV direct, A; $\text{direct} \leftarrow (A)$

MOV A, @Ri; $A \leftarrow ((Ri))$

MOV @Ri, A; $(Ri) \leftarrow A$

二、外部RAM数据传送指令

1. 使用DPTR间接寻址，寻址范围64k

MOVX A, @DPTR; $A \leftarrow ((DPTR))$

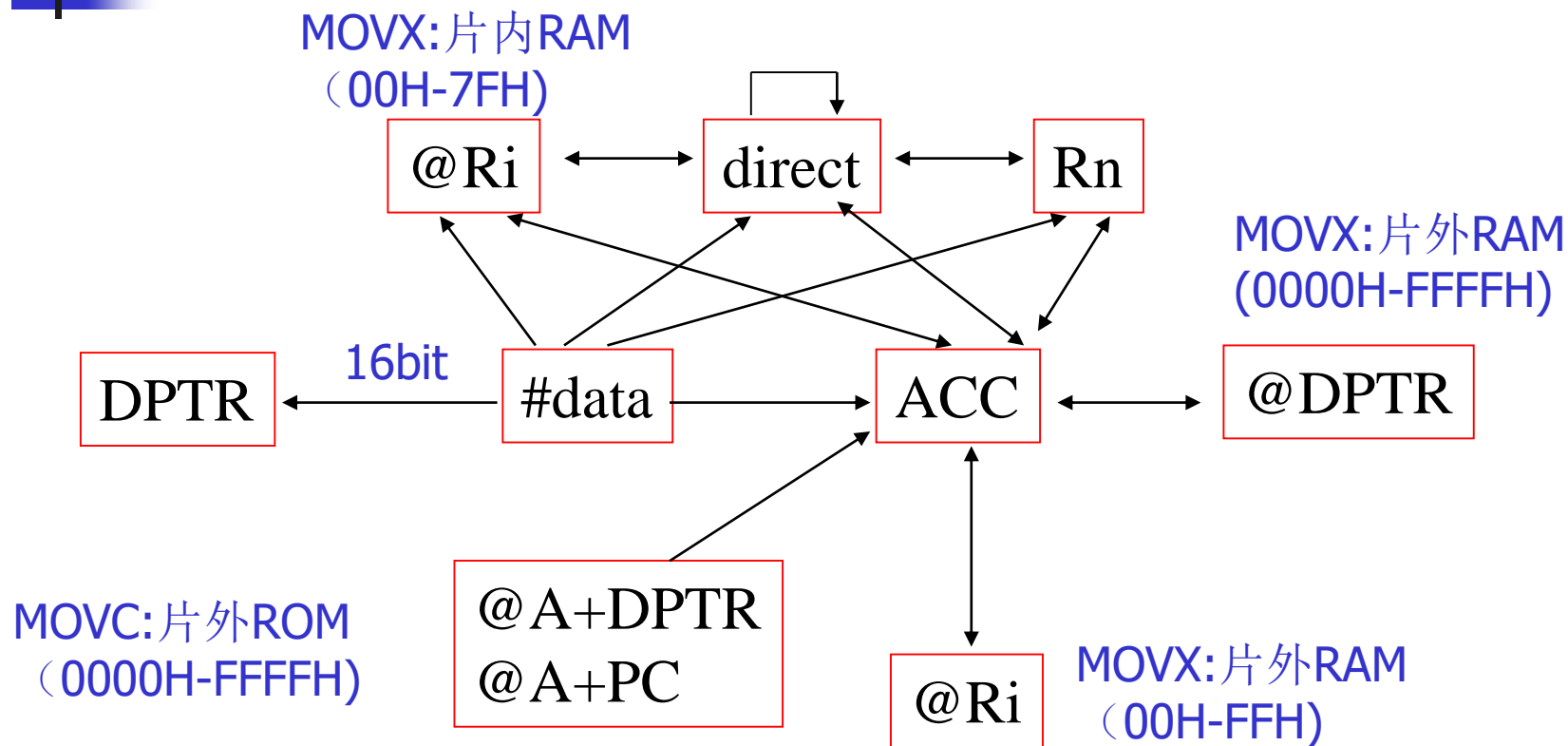
MOVX @DPTR, A; $(DPTR) \leftarrow (A)$

2. 使用Ri间接寻址，外部RAM低256单元

MOVX A, @Ri; $A \leftarrow ((Ri))$

MOVX @Ri, A; $(Ri) \leftarrow (A)$

传送指令在存储器的操作功能



传送指令网络图

三、程序存储器数据传送指令

MOVC A, @A+DPTR; $A \leftarrow ((A) + (DPTR))$

MOVC A, @A+PC; $A \leftarrow ((A) + (PC))$

其中, (A)为8位无符号数

例: 将A中十六进制数变成ASCII码, 编写查表程序。

2000	HBA: INC A
2001	MOVC A, @A+PC
2002	RET
2003	DB 30H
2004	DB 31H
⋮	⋮
2011	DB 45H
2012	DB 46H

四、数据交换指令

1. 整字节交换

XCH A, Rn; (A)↔(Rn)

XCH A, direct; (A) ↔(direct)

XCH A, @Ri; (A) ↔((Ri))

2. 半字节交换

XCHD A, @Ri; (A)_{3~0} ↔((Ri))_{3~0}

3. 累加器高低半字节交换

SWAP A; (A)_{3~0}↔(A)_{7~4}

例如：执行前(R₀)=20H,(A)=3FH,(20H)=75H

执行指令 XCHD A, @R₀以后：

(R₀=20H,(A)=35H,(20H)=7FH)



五、堆栈操作指令

进栈

PUSH direct; $SP \leftarrow (SP) + 1, (SP) \leftarrow (\text{direct})$

出栈

POP direct; $\text{direct} \leftarrow ((SP)), SP \leftarrow (SP) - 1$

堆栈操作是以SP为间址寄存器的间接寻址方式。

3.3 算术运算类指令

一、加法指令，4条

ADD A, Rn; $A \leftarrow (A) + (Rn)$

ADD A, direct; $A \leftarrow (A) + (\text{direct})$

ADD A, @Ri; $A \leftarrow (A) + ((Ri))$

ADD A, #data; $A \leftarrow (A) + \text{data}$

影响标志位AC、CY、OV

AC: 位3有进位时置1，反之清0

CY: 位7有进位时置1，反之清0

OV: 位6有进位而位7没有进位或者位7有进位而位6没有进位，则OV置1；反之清0。

例(A)=0C2H,(R₀)=0A9H, 执行ADD A, R₀后:

(A)=6BH,(AC)=0,(CY)=1,(OV)=1

$$\begin{array}{r} 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0 \\ +\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1 \\ \hline 1\leftarrow 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \end{array}$$

二、带进位加法指令，4条

ADDC A, Rn; $A \leftarrow (A) + (Rn) + (CY)$

ADDC A, direct; $A \leftarrow (A) + (\text{direct}) + (CY)$

ADDC A, @Ri; $A \leftarrow (A) + ((Ri)) + (CY)$

ADDC A, #data; $A \leftarrow (A) + \text{data} + (CY)$

影响标志位**AC**、**CY**、**OV**同上

[例 3.3] 带进位加法指令常用于多字节数的加法运算。例如三字节无符号数相加,被加数放在内部 RAM 20H~22H 单元(低位在前),加数放在内部 RAM 2AH~2CH 单元(低位在前)。可编写如下程序。

MOV	R ₀ , #20H	;被加数首地址
MOV	R ₁ , #2AH	;加数首地址
MOV	R ₇ , #03H	;字节数
CLR	C	;清 CY
LOOP:	MOV A, @R ₀	;取被加数一个字节
	ADDC A, @R ₁	;与加数的一个字节相加
	MOV @R ₀ , A	;暂存中间结果
	INC R ₀	;地址增量
	INC R ₁	
	DJNZ R ₇ , LOOP	;次数减 1,不为 0 转移
	CLR A	
	ADDC A, #00H	;处理进位
	MOV @R ₀ , A	;存进位

三、带借位减法指令，4条

SUBB A, Rn; $A \leftarrow (A) - (Rn) - (CY)$

SUBB A, direct; $A \leftarrow (A) - (\text{direct}) - (CY)$

SUBB A, @Ri; $A \leftarrow (A) - ((Ri)) - (CY)$

SUBB A, #data; $A \leftarrow (A) - \text{data} - (CY)$

影响标志位**AC**、**CY**、**OV**如下

AC: 位3有借位时置1，反之清0

CY: 位7有借位时置1，反之清0

OV: 位6有借位而位7没有借位或者位7有借位而位6没有借位，则**OV**置1；反之清0。

減法指令示例:

(A)=0C9H,(R₂)=54H,(CY)=1. 执行SUBB A, R₂后:

(A)=74H,(AC)=0,(CY)=0,(OV)=1

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

四、加1指令，5条，不影响PSW

INC A; $A \leftarrow (A)+1$

INC Rn; $Rn \leftarrow (Rn)+1$

INC direct; $direct \leftarrow (direct)+1$

INC @Ri; $(Ri) \leftarrow ((Ri))+1$

INC DPTR; $DPTR \leftarrow (DPTR)+1$

五、减1指令，4条，不影响PSW

DEC A; $A \leftarrow (A)-1$

DEC Rn; $Rn \leftarrow (Rn)-1$

DEC direct; $direct \leftarrow (direct)-1$

DEC @Ri; $(Ri) \leftarrow ((Ri))-1$

六、乘除指令

1. 乘法指令

MUL AB; 乘积低字节在A，高字节在B

影响标志位：CY=0，OV=0(乘积 $\leq 0FFH$)或
OV=1(乘积 $> 0FFH$)

2. 除法指令

DIV AB; 被除数在A，除数在B，执行后商在A，
余数在B

影响标志位：CY=0，OV=1((B)=00H)或
OV=0(其它)

七、十进制调整指令

DA A

专门用于对BCD码十进制数加法运算的结果进行修正。

为什么要进行十进制调整？因为二进制数加法指令不能完全适用于**BCD**码十进制数的加法运算。请看：

(a) $6 + 3 = 9$

$$\begin{array}{r} 0110 \\ + 0011 \\ \hline 1001 \end{array}$$

(b) $8 + 7 = 15$

$$\begin{array}{r} 1000 \\ + 0111 \\ \hline 1111 \end{array}$$

(c) $8 + 9 = 17$

$$\begin{array}{r} 1000 \\ + 1001 \\ \hline 1\leftarrow 0001 \end{array}$$

出错原因：**1.**相加结果大于**9**，说明进入无效编码区；**2.**相加结果有进位，说明跳过无效编码区。

十进制调整的方法:

- 累加器低4位大于9或辅助进位位(AC)=1, 则低4位加6修正, 即 $A \leftarrow (A) + 06H$ 。
- 累加器高4位大于9或进位标志位(CY)=1, 则高4位加6修正, 即 $A \leftarrow (A) + 60H$ 。
- 累加器高4位为9、低4位大于9, 则高4位和低4位分别加6修正, 即 $A \leftarrow (A) + 66H$ 。

例如: $(A)=56H$, $(R_5)=67H$, 执行指令:

ADD A, R₅; $(A)=0BDH$

DA A; $A \leftarrow (A) + 66H$

结果为 $(A)=23H$, $CY=1$



3.4 逻辑运算及移位类指令

一、逻辑与，6条

ANL A, Rn; $A \leftarrow (A) \wedge (Rn)$

ANL A, direct; $A \leftarrow (A) \wedge (\text{direct})$

ANL A, @Ri; $A \leftarrow (A) \wedge ((Ri))$

ANL A, #data; $A \leftarrow (A) \wedge \text{data}$

ANL direct, A; $\text{direct} \leftarrow (\text{direct}) \wedge (A)$

ANL direct, #data; $\text{direct} \leftarrow (\text{direct}) \wedge \text{data}$

二、逻辑或，6条

ORL A, Rn; $A \leftarrow (A) \vee (Rn)$

ORL A, direct; $A \leftarrow (A) \vee (\text{direct})$

ORL A, @Ri; $A \leftarrow (A) \vee ((Ri))$

ORL A, #data; $A \leftarrow (A) \vee \text{data}$

ORL direct, A; $\text{direct} \leftarrow (\text{direct}) \vee (A)$

ORL direct, #data; $\text{direct} \leftarrow (\text{direct}) \vee \text{data}$

三、逻辑异或，6条

异或运算规则：相异出1。即 $0\oplus 0=0$, $1\oplus 1=0$,
 $0\oplus 1=1$, $1\oplus 0=1$ 。

XRL A, Rn; $A \leftarrow (A) \oplus (Rn)$

XRL A, direct; $A \leftarrow (A) \oplus (\text{direct})$

XRL A, @Ri; $A \leftarrow (A) \oplus ((Ri))$

XRL A, #data; $A \leftarrow (A) \oplus \text{data}$

XRL direct, A; $\text{direct} \leftarrow (\text{direct}) \oplus (A)$

XRL direct, #data; $\text{direct} \leftarrow (\text{direct}) \oplus \text{data}$

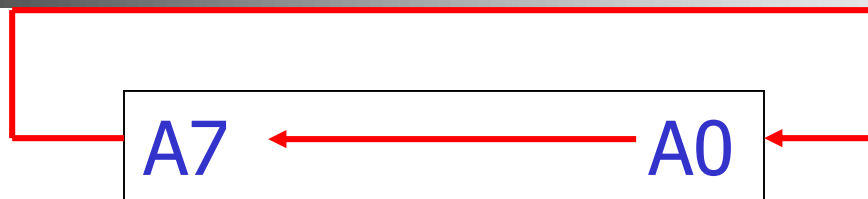
四、累加器清0和取反

CLR A; $A \leftarrow 0$

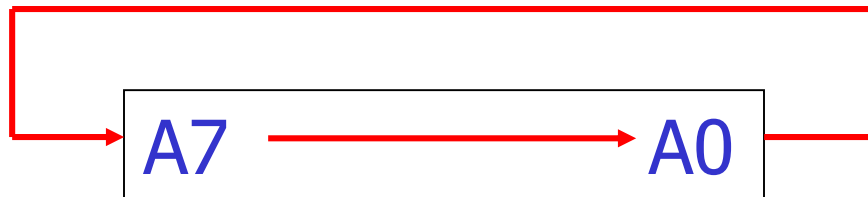
CPL A; $A \leftarrow /(A)$

五、移位指令，对A进行移位，4条

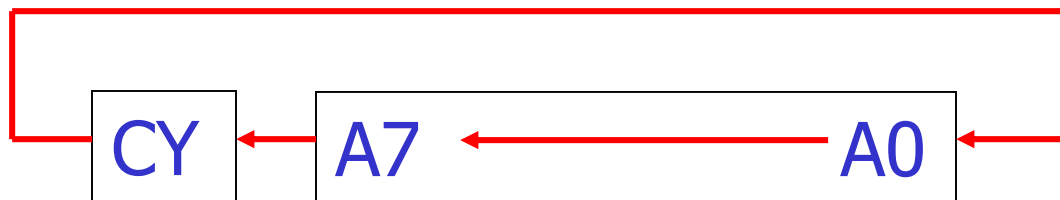
1. 循环左移 RL A



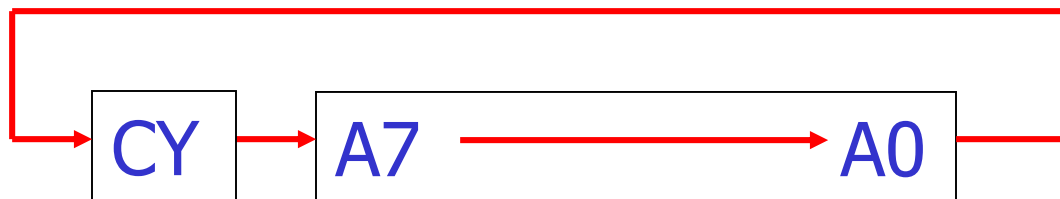
2. 循环右移 RR A



3. 带进位循环左移 RLC A



4. 带进位循环右移 RRC A



例：将A的低4位传送到P1口的低4位，但P1口的高4位不变。

MOV	R ₀ , A	; A 内容暂存 R ₀
ANL	A, #0FH	; 屏蔽 A 的高 4 位(低 4 位不变)
ANL	P ₁ , #0F0H	; 屏蔽 P ₁ 口的低 4 位(高 4 位不变)
ORL	P ₁ , A	; 实现低 4 位传递
MOV	A, R ₀	; 恢复 A 的内容

3.5 控制转移类指令

一、无条件转移指令，4条

1.长转移指令，转移范围64k

LJMP addr16; PC←addr16

2.绝对转移指令，转移范围2k

AJMP addr11; PC←(PC)+2, PC_{10~0}←addr11

例：

2070H AJMP 16AH;

则 $(PC)+2=2072H=0010 \ 0000 \ 0111 \ 0010$
 $001 \ 0110 \ 1010$

目的地址为 $0010 \ 0001 \ 0110 \ 1010=216AH$

3.短转移指令，转移范围**-128~+127**

SJMP rel; $PC \leftarrow (PC)+2+rel$

例：

① 835AH SJMP E7H;

目的地址 $=835AH+02H-19H=8343H$

② 835AH SJMP 35H;

目的地址 $=835AH+02H+35H=8391H$

- 实际编程时只需写出地址标号即可，例：

HERE: SJMP HERE; 原地踏步

HERE: SJMP \$; 原地踏步，\$表示PC当前值

4.变址寻址转移指令

JMP @A+DPTR; $PC \leftarrow (A) + (DPTR)$

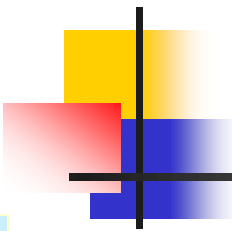
以DPTR内容为基址，以A的内容作变址。把DPTR的值固定，给A赋予不同的值，可以实现程序的多分支转移。如键盘译码程序。

二、条件转移指令

1. 累加器判零转移指令，2条

JZ rel; 若 $(A)=0$, 则 $PC \leftarrow (PC) + 2 + rel$

若 $(A) \neq 0$, 则 $PC \leftarrow (PC) + 2$



JNZ rel; 若(A)≠0, 则 $PC \leftarrow (PC) + 2 + rel$
若(A)=0, 则 $PC \leftarrow (PC) + 2$

2. 数值比较转移指令, 4条

CJNE A, #data, rel; (A)≠data转移

CJNE A, direct, rel; (A)≠(direct)转移

CJNE Rn, #data, rel; (Rn)≠data转移

CJNE @Ri, #data, rel; ((Ri))≠data转移

程序转移: $PC \leftarrow (PC) + 3 + rel$

标志位: 左操作数≥右操作数, (CY)=0

左操作数<右操作数, (CY)=1

3. 减1条件转移指令，2条

①寄存器减1条件转移

DJNZ Rn, rel;

$Rn \leftarrow (Rn) - 1$, 若 $(Rn) \neq 0$, 则 $PC \leftarrow (PC) + 2 + rel$
若 $(Rn) = 0$, 则 $PC \leftarrow (PC) + 2$

②直接寻址单元减1条件转移

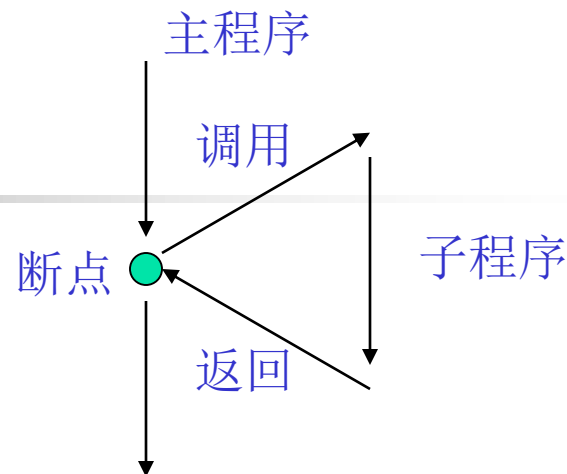
DJNZ direct, rel;

$direct \leftarrow (direct) - 1$, 若 $(direct) \neq 0$, 则 $PC \leftarrow (PC) + 3 + rel$
若 $(direct) = 0$, 则 $PC \leftarrow (PC) + 3$

例如把 2000H 开始的外部 RAM 单元中的数据送到 3000H 开始的外部 RAM 单元中,数据个数已在内部 RAM 35H 单元中。

	MOV	DPTR, # 2000H	;源数据区首址
	PUSH	DPL	;源首址暂存堆栈
	PUSH	DPH	
	MOV	DPTR, # 3000H	;目的数据区首址
	MOV	R ₂ , DPL	;目的首址暂存寄存器
	MOV	R ₃ , DPH	
LOOP:	POP	DPH	;取回源地址
	POP	DPL	
	MOVB	A, @DPTR	;取出数据
	INC	DPTR	;源地址增量
	PUSH	DPL	;源地址暂存堆栈
	PUSH	DPH	
	MOV	DPL, R ₂	;取回目的地址
	MOV	DPH, R ₃	
	MOVB	@DPTR, A	;数据送目的区
	INC	DPTR	;目的地址增量
	MOV	R ₂ , DPL	;目的地址暂存寄存器
	MOV	R ₃ , DPH	
	DJNZ	35H, LOOP	;没完,继续循环
	RET		;返回主程序

三、子程序调用与返回



1. 绝对调用指令，调用范围2k

ACALL addr11

$PC \leftarrow (PC) + 2$

$SP \leftarrow (SP) + 1, (SP) \leftarrow (PC)_{7 \sim 0}$

$SP \leftarrow (SP) + 1, (SP) \leftarrow (PC)_{15 \sim 8}$

$PC_{10 \sim 0} \leftarrow \text{addr11}$

2. 长调用指令，调用范围16k

LCALL addr16

$PC \leftarrow (PC) + 3$

$SP \leftarrow (SP) + 1, (SP) \leftarrow (PC)_{7 \sim 0}$

$SP \leftarrow (SP) + 1, (SP) \leftarrow (PC)_{15 \sim 8}$

$PC \leftarrow \text{addr16}$

3. 返回指令

RET; 子程序返回

RETI; 中断服务子程序返回

$PC_{15 \sim 8} \leftarrow (SP), SP \leftarrow (SP) - 1$

$PC_{7 \sim 0} \leftarrow (SP), SP \leftarrow (SP) - 1$



4. 空操作指令

NOP; $PC \leftarrow (PC) + 1$

3.6 位操作类指令

1. 位传送指令，2条

MOV C, bit; $CY \leftarrow (bit)$

MOV bit, C; $bit \leftarrow (CY)$

2. 位置位复位指令，4条

CLR C; $CY \leftarrow 0$

CLR bit; $bit \leftarrow 0$

SETB C; $CY \leftarrow 1$

SETB bit; $bit \leftarrow 1$

3. 位运算指令，6条

ANL C, bit; $CY \leftarrow (CY) \wedge (bit)$

ANL C, /bit; $CY \leftarrow (CY) \wedge /(bit)$

ORL C, bit; $CY \leftarrow (CY) \vee (bit)$

ORL C, /bit; $CY \leftarrow (CY) \vee /(bit)$

CPL C; $CY \leftarrow /(CY)$

CPL bit; $bit \leftarrow /(bit)$

4. 位控制转移指令

①以C为条件的转移指令

JC rel; 若 $(CY)=1$, 则 $PC \leftarrow (PC)+2+rel$

若 $(CY) \neq 1$, 则 $PC \leftarrow (PC)+2$

JNC rel; 若 $(CY)=0$, 则 $PC \leftarrow (PC)+2+rel$

若 $(CY) \neq 0$, 则 $PC \leftarrow (PC)+2$

②以位状态为条件的转移指令

JB bit, rel; 若 $(bit)=1$, 则 $PC \leftarrow (PC)+3+rel$

若 $(bit) \neq 1$, 则 $PC \leftarrow (PC)+3$

JNB bit, rel; 若 $(bit)=0$, 则 $PC \leftarrow (PC)+3+rel$

若 $(bit) \neq 0$, 则 $PC \leftarrow (PC)+3$

JBC bit, rel; 若 $(bit)=1$, 则 $PC \leftarrow (PC)+3+rel$,
并且 $bit \leftarrow 0$

若 $(bit) \neq 1$, 则 $PC \leftarrow (PC)+3$

I/O口访问的问题:

MCS_51中的4个I/O口作为专用寄存器操作。注意
事项: 读引脚前先写1。