第三章 MCS-51单片机指令系统

帅千钧

Email:sqj@cuc.edu.cn

办公室：主楼813

本章学习内容

- 3.1 MCS-51指令格式
- 3.2 MCS-51寻址方式
- 3.3 MCS-51指令系统祥解

相关概念

- **指令，**规定的计算机进行基本操作的命令语句。
  - 指令的组成：**操作码 操作数**。
- **机器语言，**采用二进制编码表示的指令，使机器能够直接识别和执行的语言。
- **汇编语言，**采用助记符、数字、符号来表示指令的程序语言，与机器语言是一一对应的。

MCS-51单片机指令系统概述

- 7种寻址方式，111条指令！

字节数
- 单字节指令：49条
- 双字节指令：45条
- 三字节指令：17条

运算速度
- 单周期指令：64条
- 双周期指令：45条
- 四周期指令：2 条

功能
- 数据传送类：29条
- 算术运算类：24条
- 逻辑运算类：24条
- 控制转移类：17条
- 位操作类： 17条

常用符号

Rn：工作寄存器中的寄存器Rn，R0…R7之一

Ri：工作寄存器中可作为地址寄存器使用的R0/R1

#data：8位立即数

#data16：16位立即数

direct：片内RAM或SFR的地址(8位)

@：间接寻址寄存器前缀符号

Bit：片内RAM或SFR的位地址

addr11：11位目的地址

addr16：16位目的地址

rel: 补码形式的8位地址偏移量。

偏移范围为-128～127

/：位操作指令中，该位求反后参与操作，不影响该位

X：片内RAM的直接地址或寄存器

(X)：相应地址单元中的内容

((X))：以X间接寻址的单元中的内容

→：箭头左边的内容送入箭头右边的单元内

3.1 MCS-51单片机指令格式

- 指令格式：就是指令的表示方式。
- MCS-51汇编语言指令格式为：

  [标号:]操作码　[操作数1],[操作数2]　[;注释]

  - 标号：指令语句地址的标志符号；
  - 操作码：指明指令功能，用指令助记符表示；
  - 操作数：指令操作对象(数据、地址、寄存器名及约定符号)；
  - 注释行：说明指令在程序中的作用。
  - 换行表示一条指令结束。
  - 分隔符：冒号　空格　逗号　分号

3.1 MCS-51单片机指令格式

- 操作码和操作数是指令的主体。

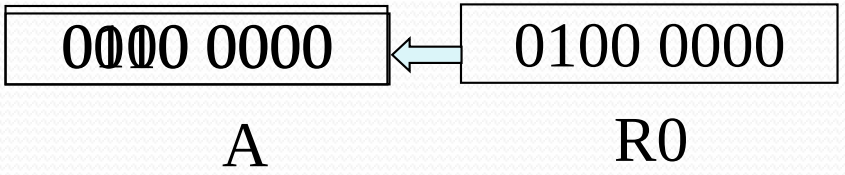
举例：

汇编语言：	机器语言：
MOV　A，R0	E8H
MOV　R6，#32H	7E 32H
MOV　40H，#64H	75 40 64H

3.2 MCS-51寻址方式

1. 寄存器寻址方式

就是**指令操作数为寄存器中的内容，即操作数在寄存器中。**

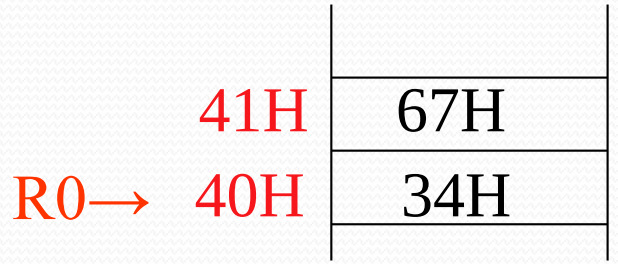


例：　MOV A，R0　　;A←R0

设指令执行前:A=20H，R0=40H,

执行指令后:A= 40H ,R0= 40H

2. 寄存器间接寻址方式



- 寄存器中的内容为操作数的地址。
- 存放地址的寄存器称为间址寄存器或称数据指针。

例： MOV A,@R0 ;A←(R0)

设指令执行前:A=20H，R0=40H，地址为40H的存储器单元内容如图所示，执行指令后:

A= 34H ,R0 = 40H ,(40H)= 34H

- 寄存器间接寻址方式的寻址范围：
  - 内部RAM低128字节，用R0或R1

    MOV A,@R0
  - 外部RAM的64KB，用DPTR

    MOVX A,@DPTR
  - 外部RAM的低256字节，用R0或R1

    MOVX A,@R0
  - 堆栈操作指令PUSH和POP(以SP作间址）

3. 直接寻址方式

指令操作数是存储单元的直接地址，数据在存储单元中。

41H	78H
40H	56H

例：　MOV A,40H

设指令执行前:A=36H，

指令执行后:A= 56H

4. 立即寻址方式

- 指令中给出实际操作数据(立即数)
- 一般用于为寄存器或存储器赋常数初值。

例：

```
8位立即数：   MOV  A,#40H              ;A←40H
16位立即数：MOV  DPTR,#2100H          ;DPTR←2100H
```

- 思考：直接寻址方式指令和立即寻址方式指令有什么不同？

5. 变址寻址方式

- **以DPTR或PC作为基址寄存器，累加器A作为变址寄存器: 数据地址 = 基地址 + 变址**
- 主要用于对ROM的查表操作。

2008H	89H
2009H	12H

例: MOVC A,@A+DPTR ;A←(A+DPTR)

MOVC A,@A+PC ;A←(A+PC)

设指令执行前：A=09H，DPTR=2000H，

存储器单元内容如图所示，执行指令后：

A= 12H ， DPTR= 2000H

6. 位寻址方式

- 对片内RAM的位寻址区和某些可位寻址的特殊功能寄存器进行位操作时的寻址方式。

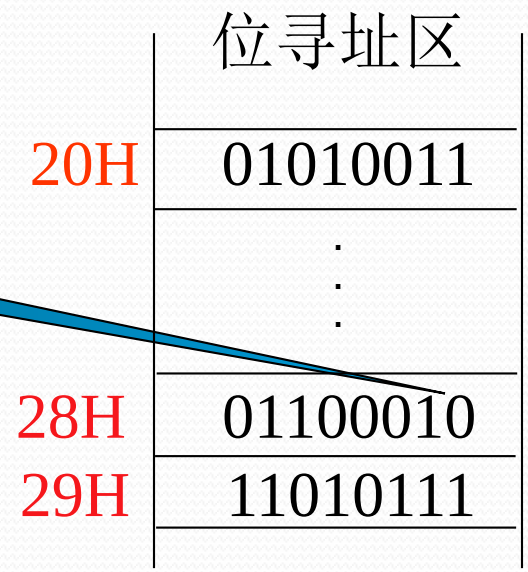
例： MOV C,40H ;CY←(位地址40H)

设指令执行前：CY=1

位地址40H存储器单元如图：

执行指令后：CY= 0

CLR C ;CY位清0



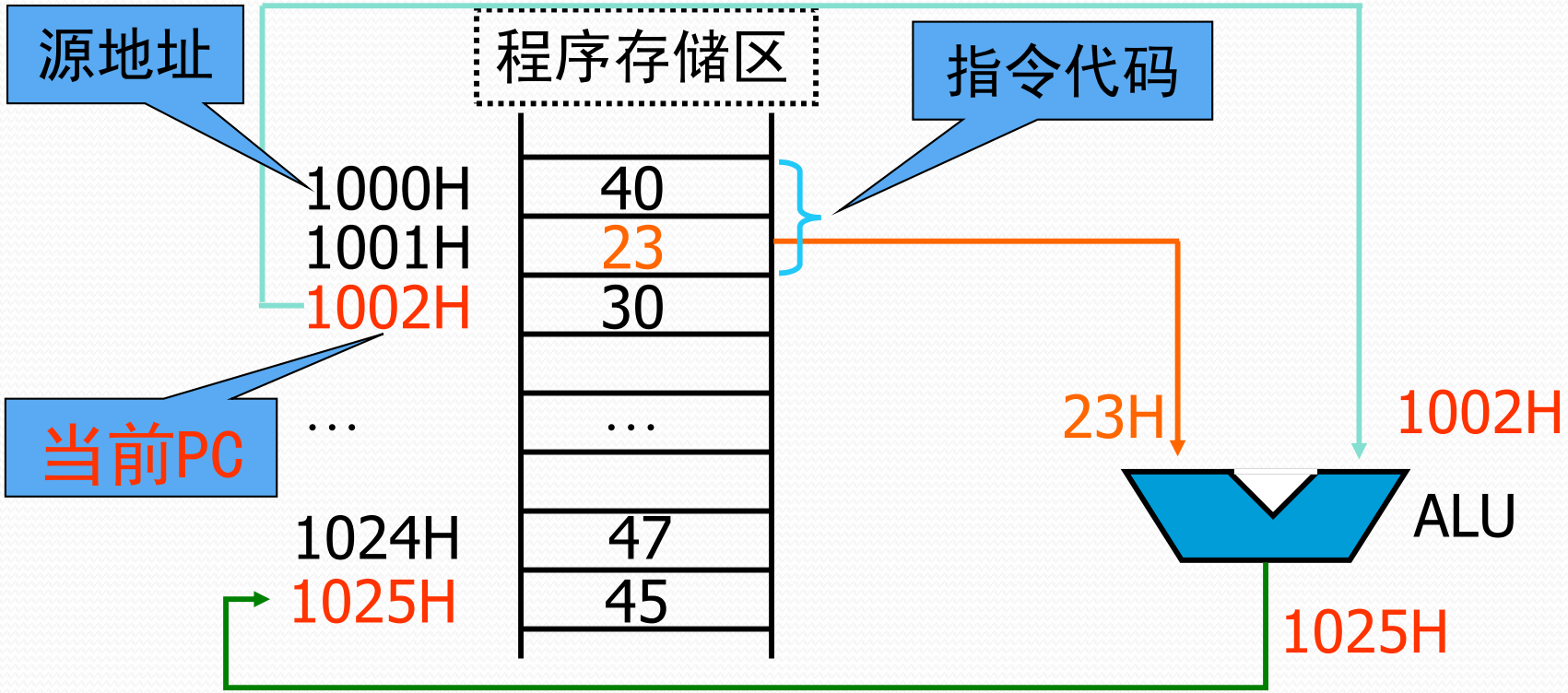
7. 相对寻址方式

- 将PC的当前内容与指令第二字节给出的数(rel)相加，结果作为跳转指令的转移地址(转移目的地址)。
- 本条转移指令所在的地址称为源地址，PC的当前内容称为基地址，指令第二字节给出的数据称为偏移量(rel)。

基地址 = 源地址 + 本条转移指令的字节数

目的地址 = 基地址 + rel

例：JC　23


源地址
程序存储区
指令代码
1000H
1001H
1002H
当前PC
…
1024H
1025H
40
23
30
…
47
45
23H
1002H
ALU
1025H

寻址方式涉及的存储器空间

寻址方式	寻址空间(操作数存放空间)
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:@R0, @R1, SP 片外RAM:@R0 , @R1, @DPTR
直接寻址	片内RAM低128字节、SFR
立即寻址	程序存储器/数据存储器
变址寻址	程序存储器:@A+PC, @A+DPTR
位寻址	片内RAM的位寻址区(20H～2FH字节地址) 某些可位寻址的SFR
相对寻址	程序存储器256字节范围内:PC+偏移量

3.3 MCS-51指令系统详解

- 学习要点
  - 了解指令的操作功能
  - 了解指令的寻址方式
  - 注意指令对程序状态字(PSW)的影响

例如：将累加器A中的内容加1

```
– ADD A，#01H      ；双字节单周期指令
                   ；影响PSW标志位
– INC A            ；单字节单周期指令
                   ；不影响PSW标志位
```

- MCS-51单片机共111条指令：

功能
- 数据传送类：29条
- 算术运算类：24条
- 逻辑运算类：24条
- 控制转移类：17条
- 位操作类：　17条

一、数据传送类指令

- 数据传送

  指令助记符：**MOV**、**MOVX**、**MOVC**
- 数据交换

  指令助记符：**XCH**、**XCHD**、**SWAP**
- 堆栈操作

  指令助记符：**PUSH**、**POP**

1. 数据传送

将数据由源操作数传给目的操作数

MOV〈目的操作数〉,〈源操作数〉

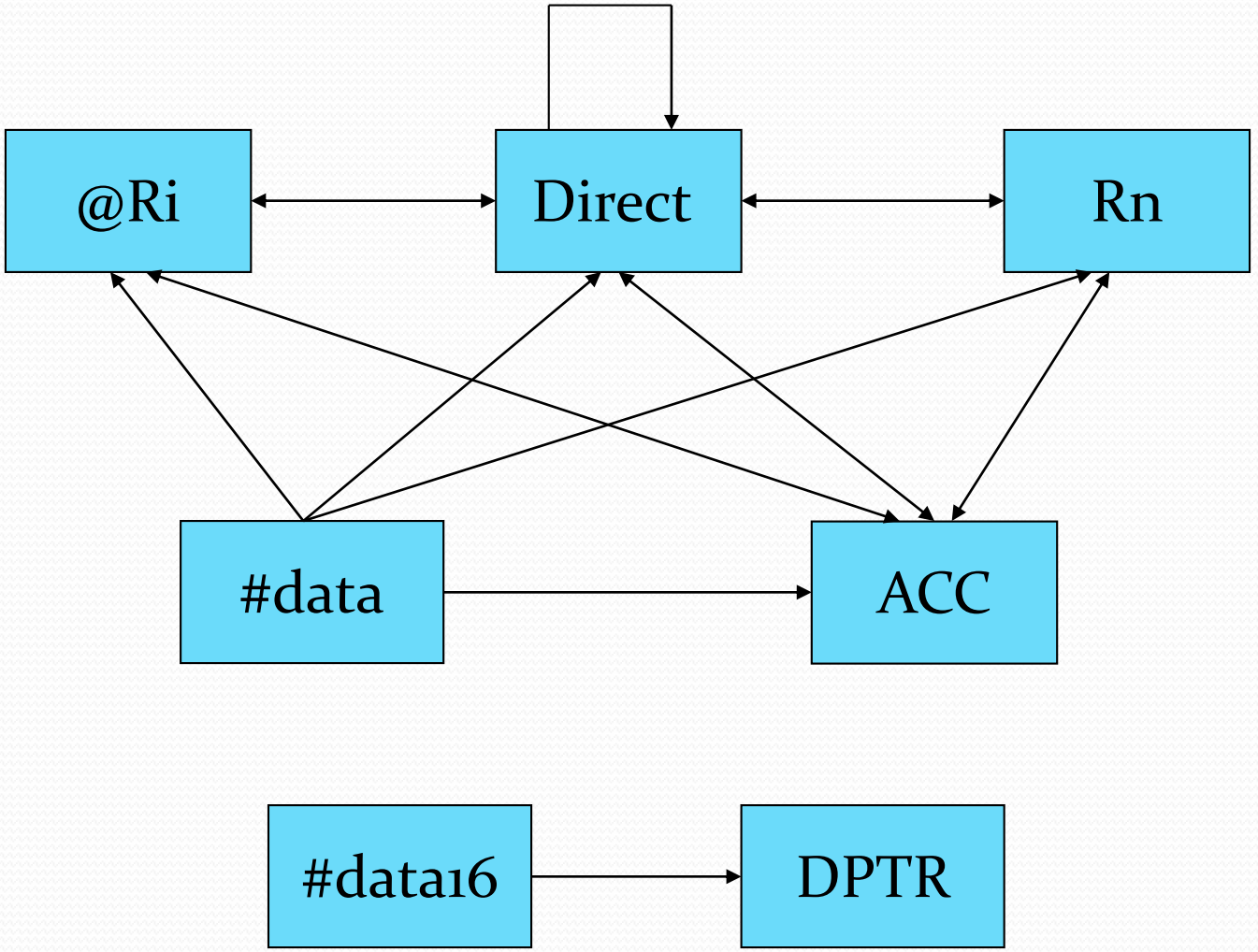
- 源操作数

  累加器A，通用寄存器Rn，直接地址(direct),间址寄存器，立即数。

- 目的操作数

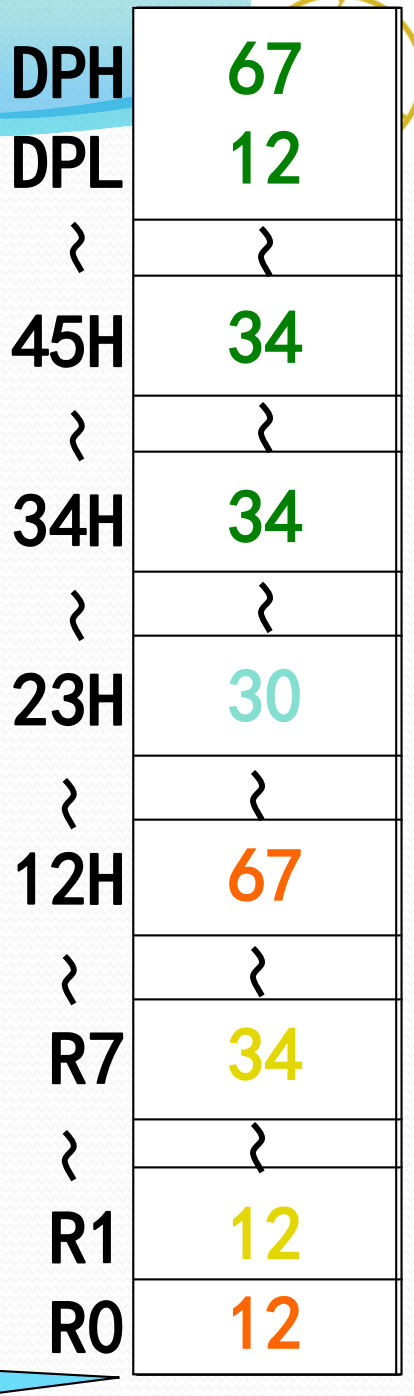
  累加器A，通用寄存器Rn，直接地址(direct),间址寄存器。

MOV：片内RAM的数据传送

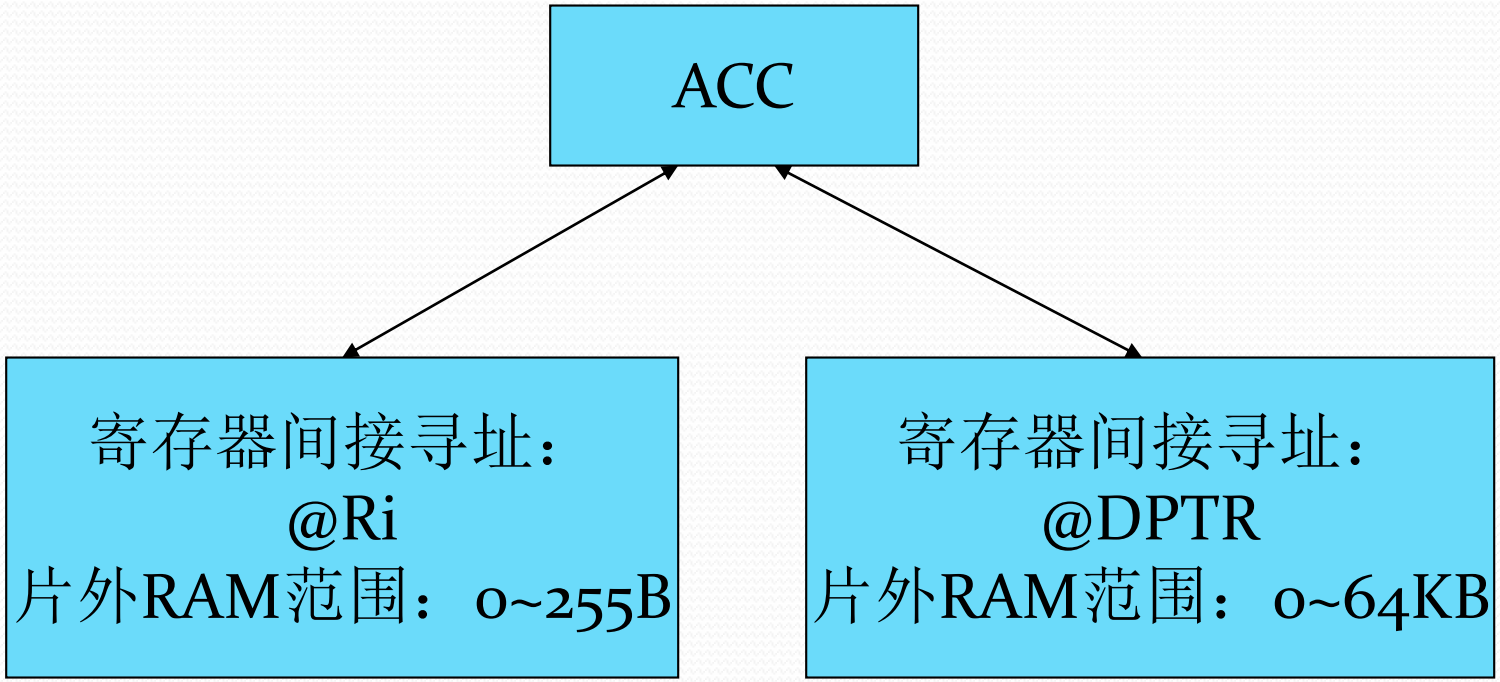


练习：给出每条指令执行后的结果

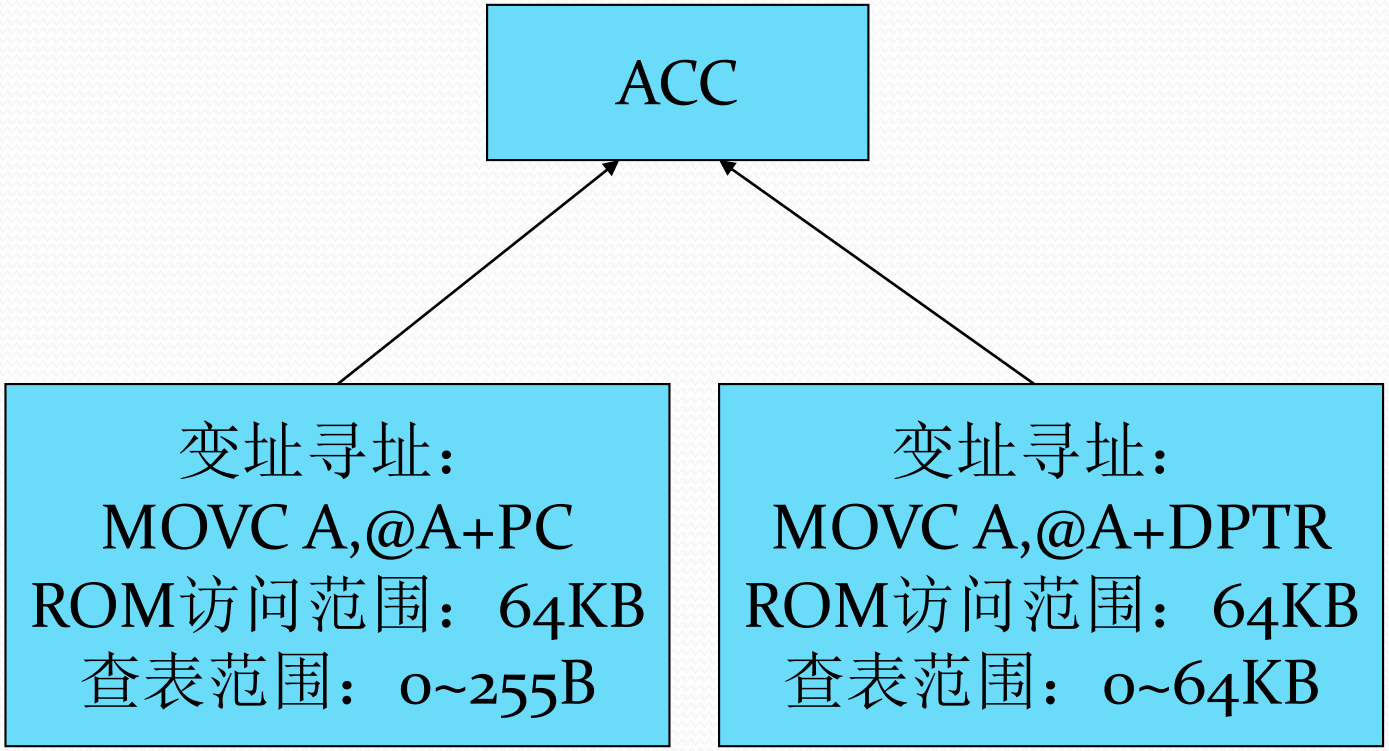
```
MOV 23H, #30H          ; (23H)=30H
MOV 12H, #34H          ; (12H)=34H
MOV R0, #23H           ; R0=23H
MOV R7, 12H            ; R7=34H
MOV R1, #12H           ; R1=12H
MOV A, @R0             ; A=30H
MOV 34H, @R1           ; (34H)=34H
MOV 45H, 34H           ; (45H)=34H
MOV DPTR, #6712H       ; DPTR=6712H
MOV 12H, DPH           ; (12H)=67H
MOV R0, DPL            ; R0=12H
MOV A, @R0             ; A=67H
```



MOVX：片外RAM的数据传送



MOVC：ROM数据传送(查表)



2. 数据交换

- 片内RAM单元与累加器A进行8位数据交换：XCH
- 累加器A的低4位与间址Ri单元的低4位进行半字节交换，高4位不变：XCHD
- 累加器高、低半字节进行交换：SWAP

➢ 数据交换类操作，一般不影响PSW的其它标志位，只对它的奇偶校验位(P)有影响。

练习：计算指令执行结果

执行前： A=40H,R0=3CH,R1=4DH

4DH	6FH
	~
40H	59H

```
XCH A,R0        ;A=3CH,R0=40H
SWAP A          ;A=C3H
XCH A,@R1       ;A=6FH,(4DH)=C3H
SWAP A          ;A=F6H
XCHD A,@R0      ;A=F9H,(40H)=56H
```

3. 堆栈操作

通过堆栈指针SP对堆栈进行读写操作。

- 进栈指令

  PUSH direct　　;SP=SP+1,(direct)→(SP)

- 出栈指令

  POP direct　　;(SP)→(direct),SP-1=SP

二、算术运算类指令

- 主要是对8位无符号数进行算术运算。
- 单片机的算术运算是单字节运算。

➢ 特点：

除了部分INC、DEC指令外，其它的运算类指令的目的操作数必定是累加器A，会影响PSW的标志位状态。

算术运算类指令分组

1. 加法指令 ADD
2. 带进位加法指令 ADDC
3. 带借位减法指令 SUBB
4. 加1指令 INC
5. 减1指令 DEC
6. 乘除指令组 MUL、DIV
7. 十进制调整指令 DA

加法指令

- ADD 两位无符号数加法

```
ADD A, #data        ; A ← (A) + data
ADD A, direct       ; A ← (A) +(direct)
ADD A, @Ri          ; A ← (A) + ((Ri))
ADD A, Rn           ; A ← (A) + (Rn)
```

- ADDC 带进位的无符号数加法

```
ADDC A,#data        ; A ← (A) + data +(CY)
ADDC A, direct      ; A ← (A) +(direct) +(CY)
ADDC A, @Ri         ; A ← (A) + ((Ri)) +(CY)
ADDC A, Rn          ; A ← (A) + (Rn) +(CY)
```

例： 1067H+30A0H

0001 0000	0110 0111	1067H
0011 0000	1010 0000	30A0H
1		
0100 0001	0000 0111	4107H

- 第一步：做67H+A0H=107H，而107H显然超过了0FFH，因此，最终保存在A中的是07H，而进位1保存在PSW中的CY位。
- 第二步：做10H + 30H + CY，结果是41H，所以最终的结果是4107H。

编程计算：1067H + 30A0H

```
MOV R0，#30H
MOV R1，#31H
MOV A,#67H
ADD A, #A0H          ; A =07H, CY=1
MOV @R0,A            ;(30)=07H
MOV A,#10H
ADDC A,#30H          ; A =41H,CY=0
MOV @R1,A            ;(31)=41H
```

减法指令

- SUBB带借位的无符号数减法

```
SUBB A, #dataH      ; A ← (A) – data – (CY)
SUBB A, direct      ; A ← (A) – (direct)– (CY)
SUBB A, @Ri         ; A ← (A) – ((Ri)) – (CY)
SUBB A, Rn          ; A ← (A) – (Rn) – (CY)
```

加1指令

INC A

INC direct

INC @Ri

INC Rn

INC DPTR

减1指令

DEC A

DEC direct

DEC @Ri

DEC Rn

例如：

内部RAM中，(40H)=50H，(41H)=60H，(50H)=30H，(51H)=70H，执行下列指令后的结果是…？

a.
```
MOV  R0，40H      ;R0=50H
MOV  A，@R0       ;A=30H
INC  R0           ;R0=51H
MOV  @R0，A       ;(51H)=30H
```

b.
```
MOV  R0，#40H     ;R0=40H
MOV  A，@R0       ;A=50H
INC  @R0          ;(40)=51H
MOV  A，@R0       ;A=51H
```

乘法指令

MUL AB ;A×B→BA

例如：

执行前：A=30H，B=0A0H

执行后：乘积为1E00H，得到B=1EH，A=00H，

标志位：CY=0，OV=1

除法指令

DIV AB ；A÷B，商→A，余数→B

- 当除数B=0时，OV=1，表示除法没有意义，不能进行操作。
- B≠0时，OV=0，除法可以正常进行。

例如：

执行前：A=40H，B=0AH

执行后：商为6，余数为4，得到A=06H，B=04H，

标志位：CY=0，OV=0

十进制调整指令

DA A

- 在进行BCD码加法运算时，跟在ADD和ADDC指令的二进制操作之后。
- 用来对BCD码加法运算结果进行自动修正，以得到正确的BCD码结果。

例如：执行下面代码后A=？

MOV A,#08H

ADD A,#03H

DA A

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

三、逻辑运算类指令

- 主要是对2个操作数按位进行逻辑操作，结果送到A或直接寻址单元。
  - 主要操作：

    与、或、异或、清零、取反、移位等。
  - 对标志位的影响：

    除了目的操作数为ACC的指令影响奇偶标志P外，一般不影响标志位。

- 逻辑运算类
  - ANL　目的操作数，源操作数

    两数相与，有一位为0，即为清0。
  - ORL　目的操作数，源操作数

    两数相或，有一位为1，即为置1。
  - XRL　目的操作数，源操作数

    两数异或，对应位相同，结果为0；对应位不同，结果为1。
- 例如：ORL P1，#0FH　　;将P1口的低4位置1

- 逻辑运算指令组
  - ANL A, #data
  - ANL A, direct
  - ANL A, @Ri
  - ANL A, Rn
  - ANL direct, A
  - ANL direct, #data

- 累加器清0、取反指令

CLR A ;0→A

CPL A ;$\overline{A}$ →A

例：**71H和56H相或：**

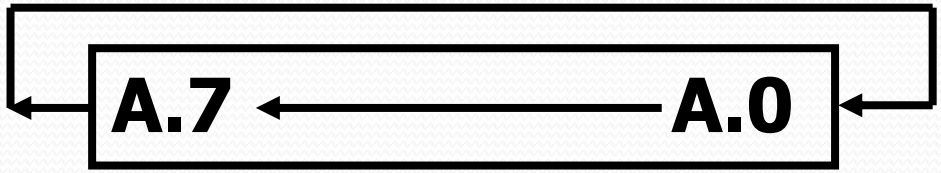
01110001 （71H）

∨ 01010110 （56H）

01110111 即77H

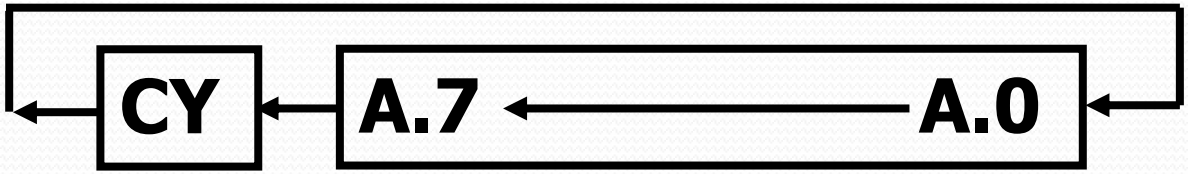
循环移位指令

RL A

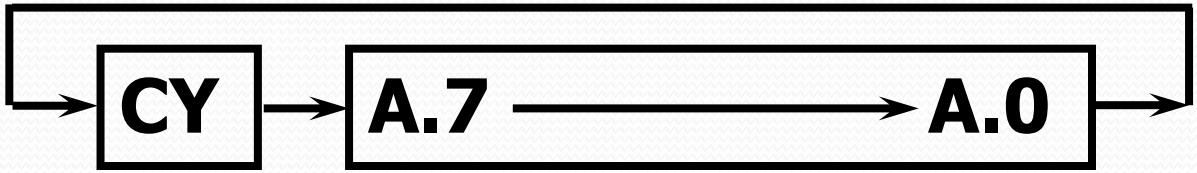


RR A

RLC A



RRC A



- 后两条指令，影响P标志和CY。

例如：

- 若A=5CH，CY=1，执行RLC A后，结果：
  A=B9H，CY=0，P=1

- 对RLC、RRC指令，CY=0，则：
  - RLC相当于乘以2
  - RRC相当于除以2

四、控制转移类指令

- 无条件转移
- 条件转移
- 子程序调用及返回
- 空操作

无条件转移

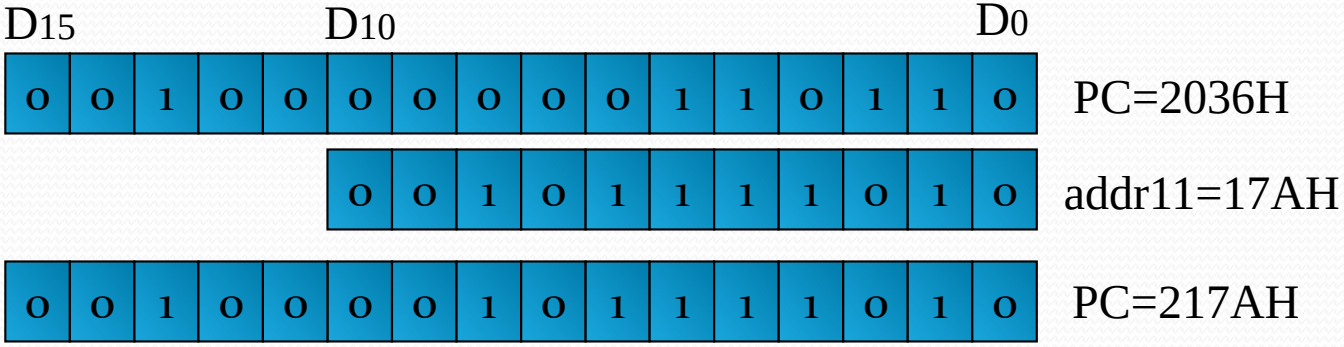
1. 绝对转移指令

AJMP　addr11　;PC+2→PC,addr11 → PC10~0

- 转移范围：PC+2后的当前地址的2K

例如：程序存储器中2034H地址单元存放了绝对转移指令：2034H　　AJMP 17AH

取出指令后：PC=2034H+2=2036H



2. 长转移指令

LJMP addr16 ； addr16→PC

转移范围为：64KROM区

3. 短转移指令(相对寻址)

SJMP rel

rel为偏移量

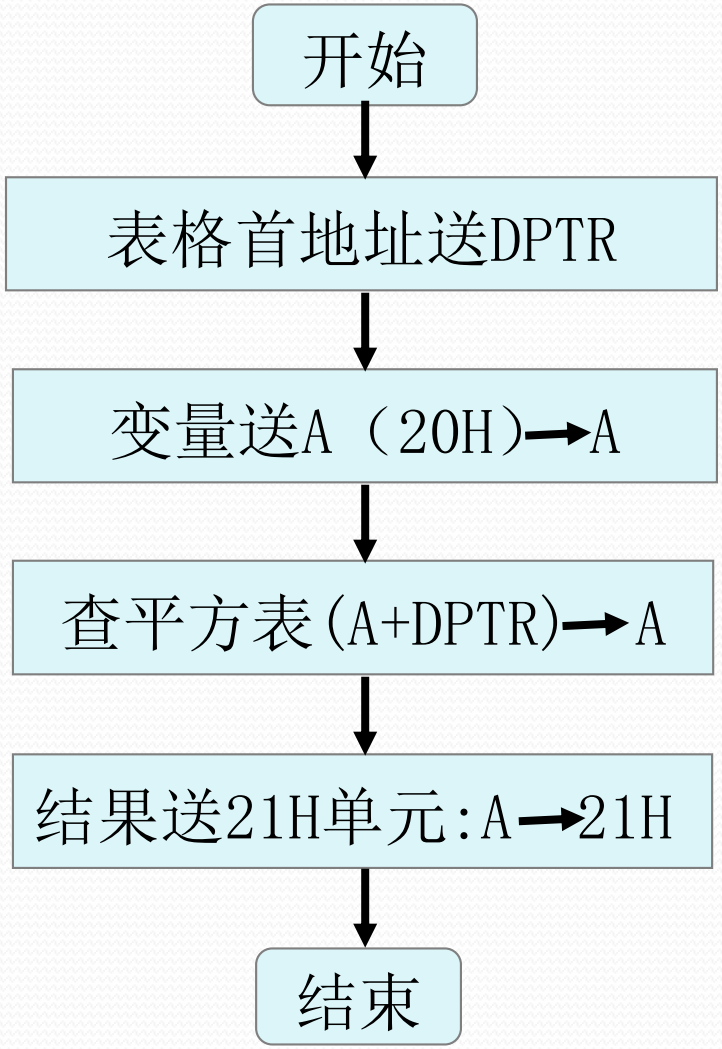
目的地址=源地址+2+rel

4. 间接转移指令(变址寻址)

JMP @A+DPTR ； A+DPTR →PC

例：试编程求内部RAM的20H单元中变量的平方值，变量取值范围为0~5，结果存在21H单元。

```
ORG  1000H
     MOV  DPTR,#Table
     MOV  A,20H
     MOVC  A,@A+DPTR
     MOV  21H,A
     SJMP  $
ORG  2000H
Table:DB  0,1,4,9,16,25
END
```



条件转移

1. 累加器判0转移

JZ　　rel　　;A=0则跳转，A≠0则不跳

JNZ　rel　　;A≠0则跳转，A=0则不跳

2. 数值比较转移(书P67)

CJNE 操作数1，操作数2，rel

注意：CY的变化

CJNE A,#data,rel

CJNE A,direct,rel

CJNE Rn,#data,rel

CJNE @Ri,#data,rel

若(A)= data，则PC ←PC+3，CY ← 0
若(A)> data，则PC ←PC+3+rel，CY← 0
若(A)< data，则PC ←PC+3+rel，CY← 1

条件转移

3. 减1条件转移(先减再判断)

   指令助记符：DJNZ

例如：

```
Delay:   MOV R2，#02H
LOOP2:   MOV R3，#0FFH
LOOP1:   DJNZ R3，LOOP1
         DJNZ R2，LOOP2
         RET
```

例：将内部RAM中起始地址为data的数据串送到外部RAM中起始地址为buffer的存储区域中，直到发现‘$’字符，传送停止。

事先不知道循环次数，先判断，后执行。

```
        MOV  R0, #data
        MOV  DPTR, #buffer
        MOV R1,#20H
LOOP1:  MOV  A, @R0
        CJNE  A,#24H,LOOP2      ;判断是否为 $ 字符
        SJMP  LOOP3             ;是，转结束
LOOP2:  MOVX  @DPTR, A          ;不是，传送数据
        INC R0
        INC DPTR
        DJNZ R1, LOOP1          ;传送下一数据
LOOP3:  SJMP LOOP3              ;停止
```

- 子程序调用
  - 绝对调用：　ACALL　　addr11
  - 长调用：　　LCALL　　addr16
- 子程序返回
  - 子程序返回：RET　　;返回PC值
  - 中断返回：　RETI
- 空操作：NOP

复习一、数据传送类指令

- 数据传送

  指令助记符：**MOV**、**MOVX**、**MOVC**
- 数据交换

  指令助记符：**XCH**、**XCHD**、**SWAP**
- 堆栈操作

  指令助记符：**PUSH**、**POP**

复习二、算术运算类指令

ADD	加法指令
ADDC	带进位加法指令
SUBB	带借位减法指令
INC	加1指令
DEC	减1指令
MUL、DIV	乘除指令组
DA	十进制调整指令

复习：三、逻辑运算类指令

- 逻辑运算类

```
ANL                ;两数相与
ORL                ;两数相或
XRL                ;两数异或
CLR  A             ;累加器清0
CPL  A             ;累加器取反
RL  A              ;累加器循环左移
RR  A              ;累加器循环右移
RLC  A             ;累加器带进位循环左移
RRC  A             ;累加器带进位循环右移
```

四、控制转移类指令

- 无条件转移
  - AJMP addr11
  - LJMP addr16
  - SJMP rel
  - JMP @A+DPTR
- 条件转移
  - JZ rel
  - JNZ rel
  - CJNE 操作数1,操作数2,rel
  - DJNZ 操作数1,rel

四、控制转移类指令

- 子程序调用及返回
  - ACALL addr11
  - LCALL addr16
  - 子程序返回：RET
  - 中断返回：　RETI
- 空操作
  - NOP

- 设（20H）=00H，(24H)=01H,执行完下列指令后，CY=（　　）。

```
MOV C,20H
```

- 执行以下程序段后，A=（　）,CY=(　)。

```
MOV   A,#7AH
MOV   R0,#30H
MOV   30H,#0A5H
SETB  C
SUBB   A,@R0
```

- 设A=83H，R0=25H，（25H）=34H，执行下列指令后，A=（ ），（25H）=( )。

```
ANL  A，#25H
ORL  25H，A
XRL  A，@R0
CPL  A
```

- 设SP=60H，执行完下面指令后PC=(　　),SP=(　)。

```
ORG 2000H
LCALL  3000H
```

- 设（SP）=60H，片内RAM （30H）=20H，（31H）=21H，执行程序

```
PUSH 30H
PUSH 31H
POP  DPL
POP  DPH
MOV  A,#00H
MOVX @DPTR,A
```

最后执行结果将0送到（　　）单元。

五、位操作类指令

- 是以一个数据位(bit)为单位进行的运算和操作。
- 常以进位位CY作为为累加器C来进行位的传送和逻辑运算。
  - 位传送

    指令助记符：MOV
  - 位清0、置1

    指令助记符：CLR、SETB
  - 逻辑运算

    指令助记符：ANL、ORL 、CPL
  - 位控制转移(控制转移类指令)

    指令助记符：JC、JNC、JB、JNB、JBC

26H	1	0	1	1	0	1	0	1
25H	1	0	0	0	0	1	1	0
24H	0	1	1	1	0	0	0	0
…								
20H								

- 位传送：MOV

MOV C，bit ；(bit)→C

MOV bit，C ；C → (bit)

例如：MOV C，20H

MOV 30H，C

- 位清0、置1：CLR、SETB

CLR C ；0→CY

CLR bit ；0→(bit)

例如：CLR 30H ；0→(位地址30H)

例题：编程将10H位的内容传送到30H位。

```
MOV   40H, C
MOV   C, 10H
MOV   30H, C
MOV   C, 40H
```

位逻辑运算：ANL、ORL 、CPL

位逻辑运算(与、或)：将C中的内容与某位内容进行逻辑与、或操作结果送入位累加器C。

ANL C，bit ； C←C∧(bit)

ANL C，/bit ； C←C∧/(bit)

例如：CY=1

ANL C，/30H ； CY=0

ORL C，2AH ； CY=1

CPL C ； CY=0

CPL 27H

26H	1	0	1	1	0	1	0	1
25H	1	0	0	0	0	1	1	0
24H	1	1	1	1	0	0	0	0
…								
20H								

位条件转移

1. JC rel ；CY=1，转移

   JNC rel ；CY=0，转移

2. JB bit，rel ；bit=1，转移

   JNB bit，rel ；bit=0，转移

   JBC bit，rel ；bit=1，bit位清零并转移

例题： 编程求解y=syn(x)

$$y = \begin{cases} +1 & , x>0 \\ 0 & , x=0 \\ -1 & , x<0 \end{cases}$$

```
SYNF:    MOV R0,#x
         CJNE R0,#00H,COMP0  ;
         MOV R1,#00H
         SJMP COMPED
COMP0:   JC COMP1
         MOV R1,#01H
         SJMP COMPED
COMP1:   MOV R1,#0FFH
COMPED:  RET
```

y=syn(x)

```
SYNF:    MOV A,#x
         JZ COMP0;
         JB ACC.7,COMP1
         MOV R1,#01H
         SJMP COMPED
COMP0:   MOV R1,#00H
         SJMP COMPED
COMP1:   MOV R1,#0FFH
COMPED:  RET
```

- 练习：试编写1234H-09ABH的程序段，将结果高8位存入51H， 低8位存入50H单元。

```
CLR  C
MOV  A，#34H
SUBB A，#0ABH
MOV  50H，A
MOV  A，#12H
SUBB A，#09H
MOV  51H，A
```

练习：把累加器A中的低4位状态，通过P1口的高4位输出， P1口的低4位状态不变。

	0	1
ANL	清0	保留
ORL	保留	置1

```
ANL  A,#0FH          ;屏蔽A的高4位
SWAP  A
ANL  P1,#0FH         ;清P1口高4位
ORL  P1,A            ;P1口高4位输出A的低4位;
```

练习：

排序编程，设(40)=06H，(41)=0AH，(42)=01H，试编程将这3个单元里的数据从小到大排序。

```
        MOV R1,#02H
COMP0:  MOV A,R1
        MOV R4,A
        MOV R0,#40H
COMP1:  MOV A,@R0
        MOV R2,A
        INC R0
        MOV A,@R0
        MOV R3，A
        CLR C
        SUBB A,R2
        JNC  CONT       ;C=0,41H大，不交换,接着比较
        MOV A,R2        ;C=0,40H大，交换;取40H单元数据
        MOV @R0,A       ;40H单元数据存入41H单元
        DEC R0          ;R0指向40H
        MOV A,R3        ;取原41H单元数据
        MOV @R0,A       ;存入40H单元数据
        INC R0          ;R0指向41H
CONT:   DJNZ R4,COMP1
        DJNZ R1,COMP0
        SJMP COMP
COMPEN: END
```

上机练习

- 熟悉Keil C、Proteus仿真软件的调试开发过程，练习所学的指令。
- 练习P74习题一、二。
- 练习P75练习题3-3、3-4、3-5、3-6，编程查看程序运行结果。
- 试编写1234H-09ABH的程序段，将结果高8位存入51H， 低8位存入50H单元。

单片机程序仿真与调试

- Keil C软件仿真
  1. 创建工程
  2. 设置工程参数,选择MCU型号
  3. 创建编辑源文件,.asm或.c
  4. 编译/汇编源程序
  5. 调试程序
  6. 输出代码文件，.hex
- Keil C与Proteus联调硬件电路

项目管理式的程序开发与调试

1. 新建项目文件，名称要符合DOS文件名要求。
2. 高级设置，设置项目文件路径，应与源程序文件路径一致。
3. 添加文件，可以是C51程序，扩展名为：.c或是汇编语言程序，扩展名为：.asm。
4. 编译/汇编，产生消息与程序关联
5. 产生代码，装入仿真器并跟踪调试
6. 输出代码文件，16进制：.hex；2进制：.bin