第3章 指令系统
3.1 指令格式

1. 指令（Instruction）：把要求计算机执行的各种操作以命令的形式写下来，通常一条指令对应着一种基本操作。
2. 指令格式

操作码　操作数，操作数

干什么　参与的对象

操作数：数，寄存器，变量，内存地址，端口地址

3.指令长度

1-6字节　　1 字节　只有操作码　如　NOP

3.2 寻址方式

寻址方式： 寻找操作数的方法

寻址方式分四种：

立即寻址　　寄存器寻址

存储器寻址　　端口寻址

1.立即寻址

操作数在指令中，紧跟在操作码之后，操作数也称为立即数。

MOV　AL　, 34H

执行后　AL=34H

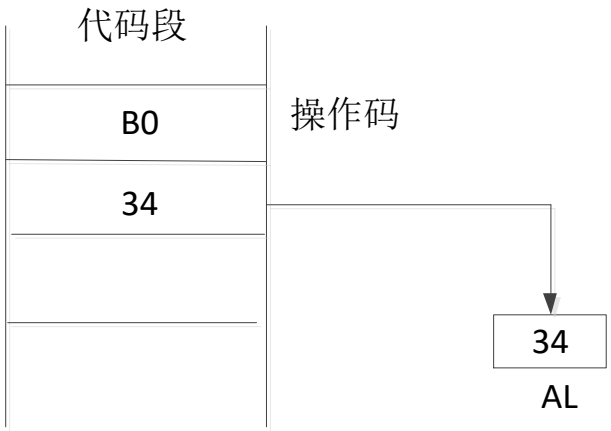
MOV　BL　,10

执行后　BL=0AH

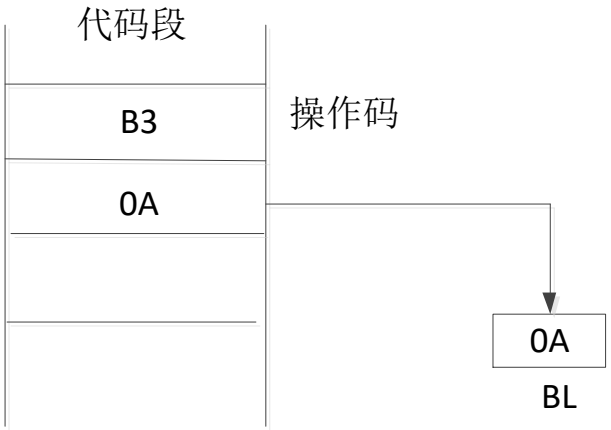
MOV　AX　, 1234H

执行后　AX=1234H

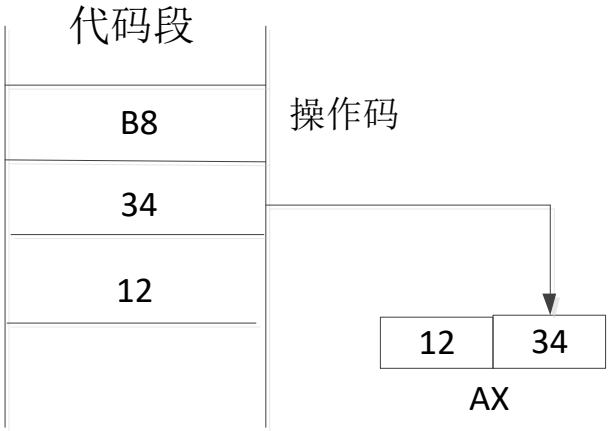
MOV AL ,34H



MOV BL ,10



MOV AX ,1234H



2. 寄存器寻址

寄存器的内容是操作数

1）MOV　BL , AL

执行前 AL=23H ,BL=10H

执行后 AL=23H ,BL=23H

2）MOV　AH, AL

3）MOV　BX, AL

4）MOV　DL, AX

5）MOV　BX, BP

1）MOV　AH, AL　√

2）MOV　BX,　AL　×

3）MOV　DL,　AX　×

4）MOV　BX,　BP　√

错误原因：类型不匹配

在C语言中

uchar　a　;　　uint　b;

b=a;　√　　a=b;　√

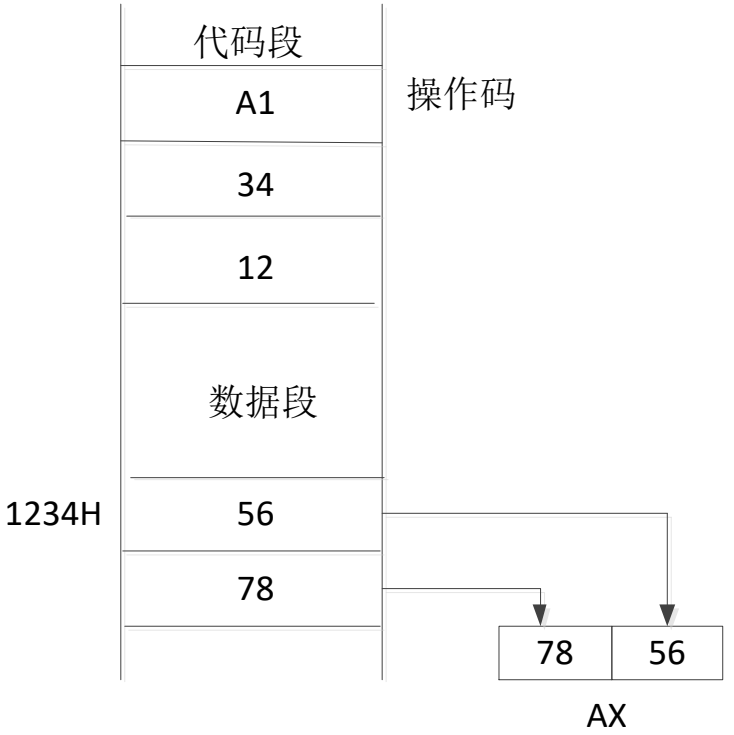
3. 存储器寻址

1）直接寻址

操作码的后面是操作数的（偏移）地址

偏移地址就是有效地址

MOV AX，[1234H]

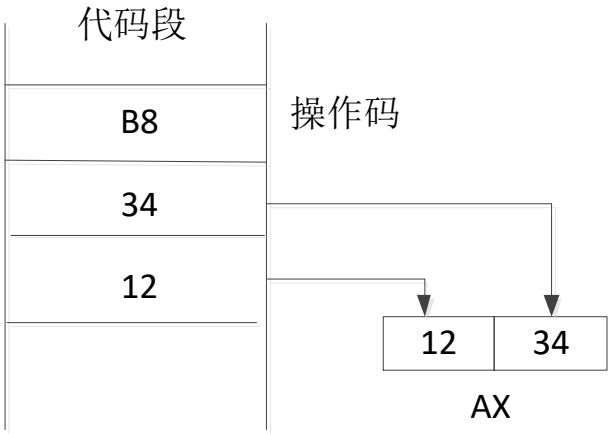


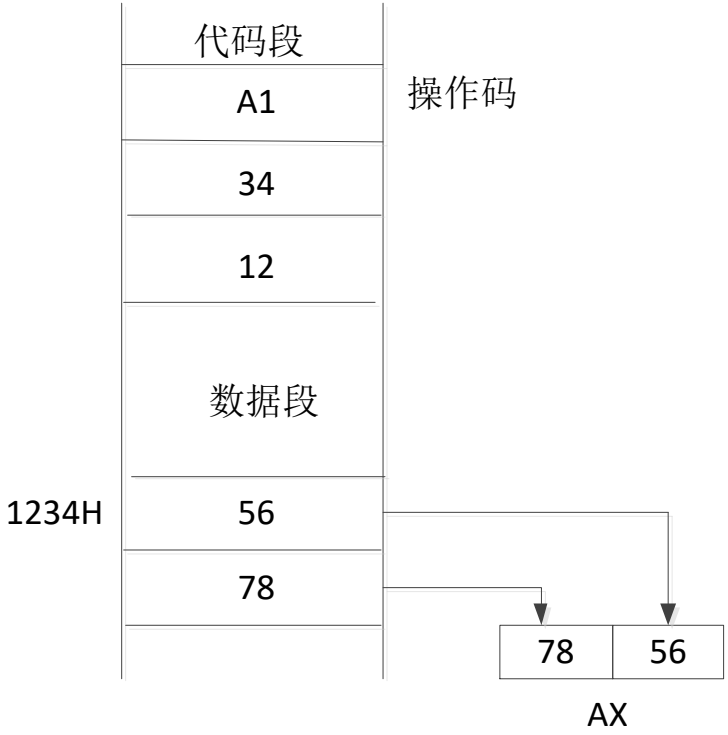
直接寻址默认在数据段

设 DS=1000H

这操作数的物理地址为11234H

- 区别
- MOV　AX，1234H
- MOV　AX，[1234H]





- MOV　AX，1234H

例1 完整的执行指令包括取指、译码、执行三个过过程，设计算机主频为5MHZ，此语句三个字节，存放IP=1000H，问8086/8088完整执行此语句访问内存需要多少时间？

答：8086取指令2个总线周期　译码和执不需要访问总线

t=2*4*T=8/5M

8088取指令3个总线周期　译码和执行不需要访问总线

t=3*4*T=12/5M

MOV　AX，[1234H]

例2完整的执行指令包括取指、译码、执行三个过过程，设计算机主频为5MHZ，此语句三个字节，IP=1000H，问8086/8088完整执行此语句，访问内存需要多少时间？

答：8086取指令2个总线周期　译码忽略不计　执行1个总线周期　　t=(2+1)*4*T=12/5M

8088取指令3个总线周期　译码忽略不计　执行2个总线周期　　t=(3+2)*4*T=20/5M

变量（符号地址），直接寻址的常用的方式

在数据段定义

ORG 0000H

BUF DB 13H，14H，15H；字节变量

在代码段使用

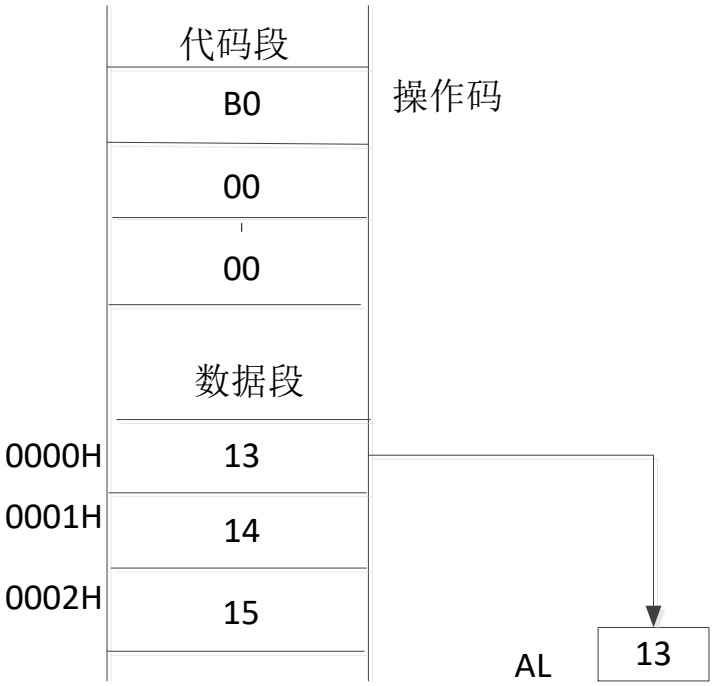
MOV AL ,[BUF] == MOV AL，[0000H]

执行后 AL=13H

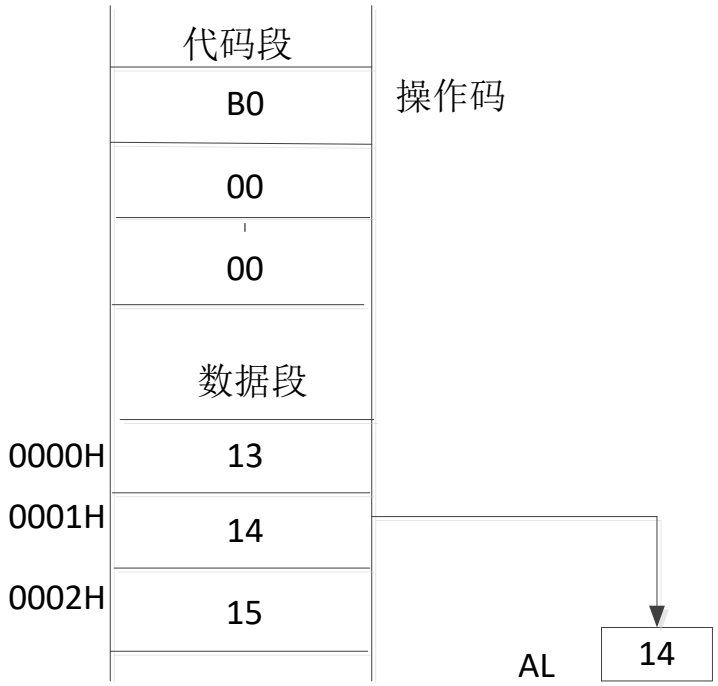
MOV AL，[BUF+1]== MOV AL，[0001H]

执行后 AL=14H

MOV AL ,[BUF]



MOV AL ,[BUF+1]



在数据段定义

ORG 0000H

BUF1 DW 1314H，1516H；字变量

在代码段使用

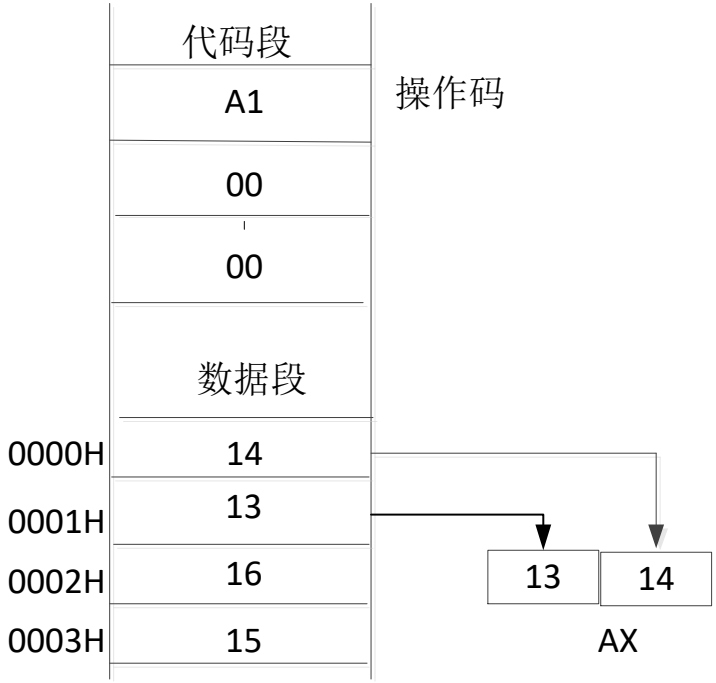
MOV AX，BUF1 == MOV AX，[0000H]

执行后 AX=1314H

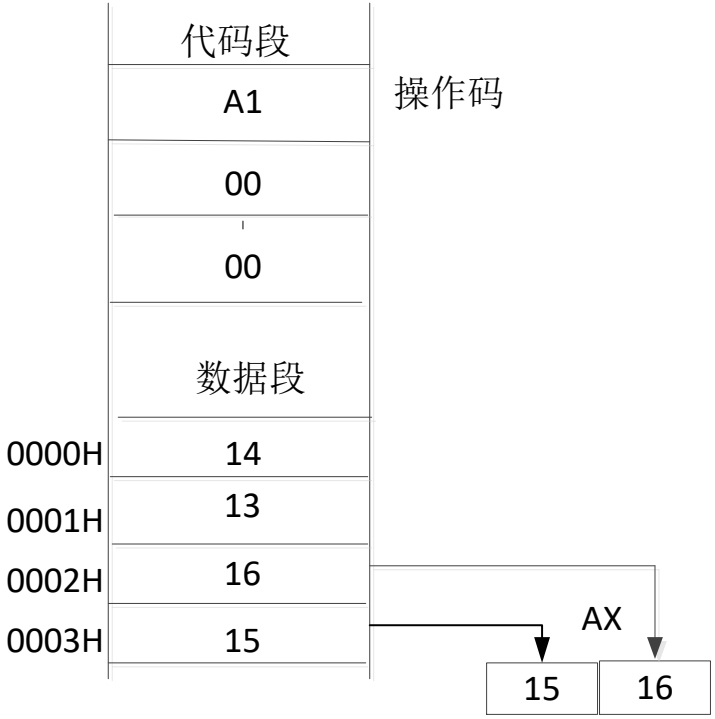
MOV AX，BUF1+2== MOV AX，[0002H]

执行后 AX=1516H

MOV AX，[BUF1]



MOV AX，[BUF1+2]



一般在汇编语言中变量（符号地址）

MOV AL , [BUF]

省掉括号写成

MOV AL， BUF

MOV A X,[BUF1]

写成

MOV AX， BUF1

直接寻址方式的物理地址

MOV　AX，[1234H]

默认情况在数据段

物理地址=　DS*16+直接地址

MOV　AX，ES：[1234H]

段超越前缀改变段属性

物理地址=　ES*16+直接地址

2）寄存器间接寻址

寄存器的内容是操作数的地址

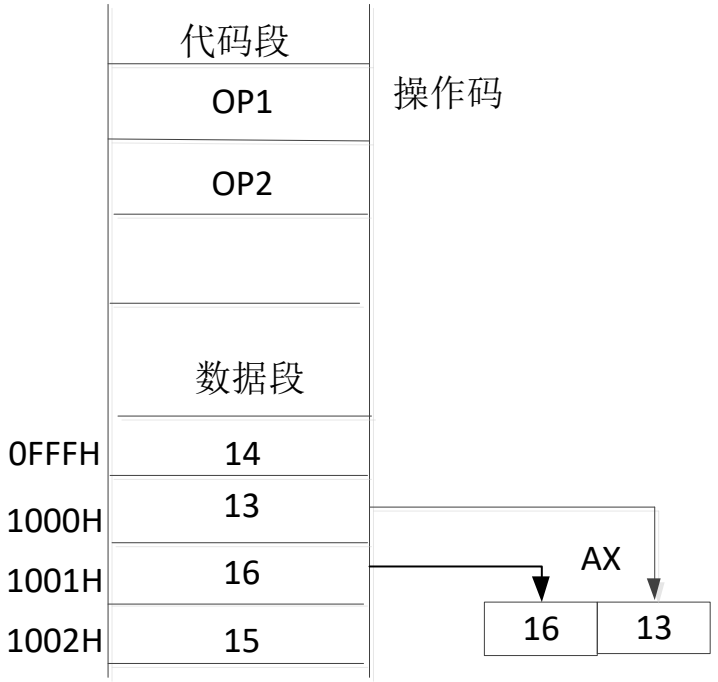
MOV　AX，[BX]

设 BX=1000H

设DS=2000H

操作数的实际地址

21000H



可用于寄存器间接寻址的寄存器：

BX SI DI BP

如：

MOV AL，[BX]

MOV AX，[BP]

MOV BX，[SI]

MOV DX，[DI]

寄存器间接寻址方式的物理地址

默认情况在数据段

BX SI DI

物理地址=DS*16+寄存器（BX或SI或DI）的值

默认在堆栈段

BP

物理地址= SS*16+寄存器BP的值

段超越前缀改变段默认段属性

MOV　AX，　ES：[SI]

物理地址=　ES*16+寄存器SI的值

寄存器间接寻址的作用

寄存器里存放数据的首地址，实际是指针。

将数组的首地址给BX

如果是字节变量，BX加1指向下一个字节元素

```
    ORG  1000H
 BUF   DB  12H,34H,56H,78H
MOV  BX,1000H
MOV  AL ,[BX]
ADD BX，1      ；    MOV   BX,1001H
MOV   AL ,[BX]
```

如果是字变量，BX加2指向下一个字元素

```
    ORG  0200H
BUF1 DW  1234H,5678H
 MOV BX,0200H
 MOV AX,[BX]
 ADD  BX，2    ；MOV  BX,0202H
 MOV   AX,[BX]
```

3）相对寄存器寻址
（寄存器相对寻址）

相对量加上寄存器的内容是操作数的地址

MOV　AL，[BX+100H]

如果BX=1000H，

则　操作数的偏移地址=1100H

假如　DS=1000H

则CPU访问内存的实际地址=11100H

也可以写成：

MOV　AL，100H[BX]

寄存器相对间接寻址的作用

要处理的数据的首地址可以作为相对量，相对首地址的偏移地址存放在寄存器中。

将数组的首地址作为偏移量

如果是字节变量，BX加1指向下一个字节元素

```
    ORG  0100H
BUF   DB  12H,34H,56H,78H
      ……
MOV  BX,0H
MOV  AL ,[BX+100H]
MOV   BX,1H
MOV   AL ,[BX+100H]
```

或者：

```
        ORG    0100
     BUF   DB  12H,34H,56H,78H
          ........
     LEA    BX,BUF
     ;取变量的偏移地址，相当于把
0100H取出来赋值给BX
     MOV  AL ,[BX+00H]  ;取12H
     MOV   AL ,[BX+01H] ;取34H
     ;此处的相对量是离BUF变量的起始
位置的偏移
```

```
ORG  0100H
 BUF   DW  1234H,5678H
    ......
MOV  BX,  0H
MOV  AX ,   [BX+0100H]
MOV   BX,   2
MOV   AX ,  [BX+0100H]
```

4）基址变址寻址

操作数的内容（偏移）地址由基址BX、BP寄存器内容和变址SI、DI寄存器的内容之和组成

基址	变址
BX	SI
BP	DI

MOV AX,[BX+SI] √

MOV AX ,[BX+DI] √

MOV AX,[BP+SI] √

MOV AX,[BP+DI] √

MOV AX,[BX+BP] ×

MOV AX,[SI+DI] ×

默认约定

物理地址=16*DS +BX+ DI或SI

物理地址=16*SS+BP+DI或SI

段超越前缀改变段属性

```
MOV   AX,SS:[BX+SI]
```

基址变址寻址的作用

基址寄存器存放数据的首地址，变址寄存器存放偏移地址。

将数组的首地址给BX

如果是字节变量，SI加1指向下一个字节元素

```
ORG  0200H
BUF   DB  12H,34H,56H,78H
MOV  BX,0200H
MOV   SI,0
MOV  AL ,[BX+SI]
MOV   SI ,1
MOV   AL ,[BX+SI]
```

5）相对基址变址寻址

操作数的内容（偏移）地址由基址BX、BP寄存器内容和变址SI、DI寄存器的内容再加上偏移量之和组成

基址	变址
BX	SI
BP	DI

各种寻址方式

```
MOV   AL ,03H
MOV  AX, BX
MOV   AX, 1234H
MOV   AX,[1234H]
MOV   AL ,BUF （字节变量）
MOV    AL ,[BX]
MOV    AL,10H[BX]
MOV    AL ,[BX+SI]
MOV    AL,10H[BX+SI]
```

各种寻址方式除立即寻址方式和寄存器寻址外，其他的寻址方式都是给出的操作的偏移地址，计算机根据各种寻址方式的给出偏移（有效）地址以及其段属性，就能计算出对应操作数所在的物理地址，进而可以访问内存获取该操作数。

各种寻址方式

MOV　　AX，　　1234H

操作码　目的操作数，　源操作

　　除了立即寻址方式，其他都可以作为目的操作数的寻址方式

```
MOV   BX     , AX
MOV  [1234H], AX
MOV   BUF    , AX
MOV   [BX]   , AL
MOV  10H[BX] , AL
MOV   [BX+SI] ,  AL
MOV  10H[BX+SI] , AL
```

3.3 数据传送指令

指令按功能分为六类：

传送指令 √

算术运算指令 √

逻辑运算指令 √

串操作指令 ×

程序控制指令 √

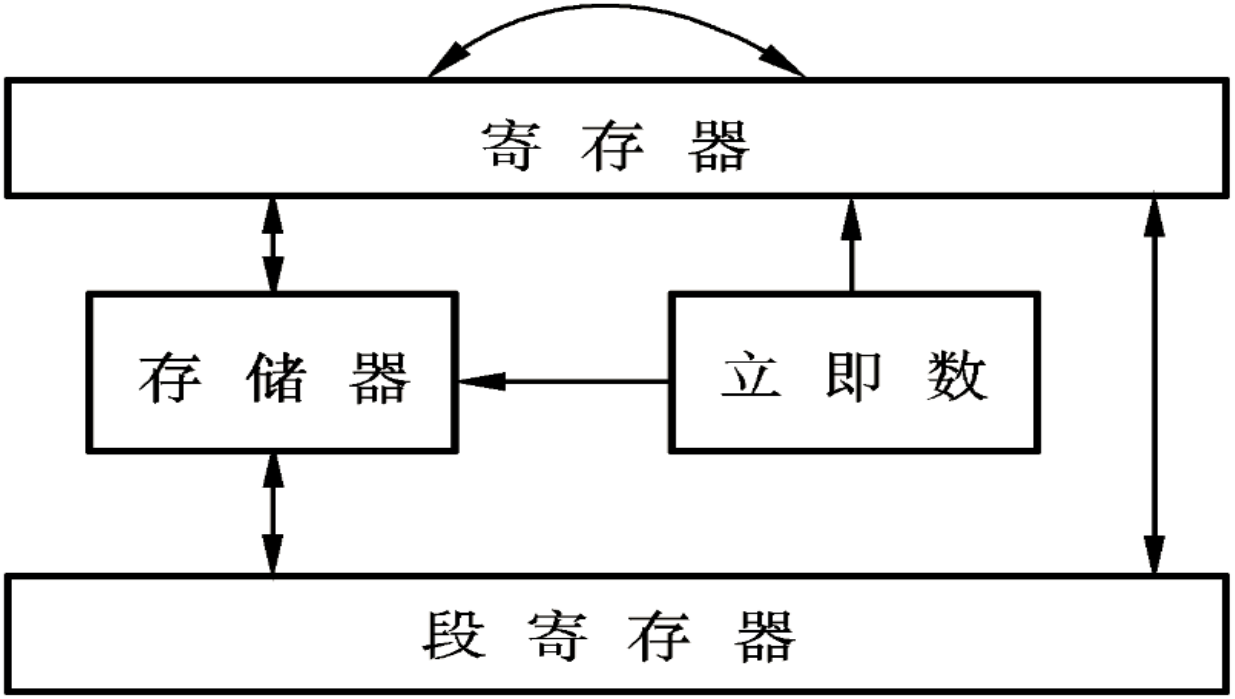
CPU控制指令 √

MOV ， XCHG ， PUSH POP, LEA

XLAT OUT IN （以后讲）

1.MOV

格式　MOV　DST, SRC　;(B/W)（字节/字）



- 1）寄存器之间

```
MOV   BL ,BH
MOV   DH ,AL
MOV   CX, AX
MOV   DL,  BX
MOV   CS, AX
MOV   BP ,BX
```

- 1.寄存器之间

  MOV　BL ,BH

  MOV　DH ,AL

  MOV　CX, AX

  MOV　DL,　BX　×

  MOV　CS, AX　×

  MOV　BP ,BX

1）类型不匹配错误

2）代码段寄存器不允许用语句赋值，一般由系统赋值

- 2）寄存器和存储器之间

```
MOV  [1000H], AL
MOV  CX  ,[2000H]
```

3）立即数和寄存器、存储器之间

```
MOV   AL，30
MOV  BL  ，'A'
MOV   [1000H], 20H
MOV   [1000H], 200H
MOV   [BX]，30H
MOV   [BX]，300H
```

MOV [1000H], 20H ×

错因，不明确把20H存放字节单元还是字单元

MOV [1000H], 200H √

只能是字单元存放

MOV [BX]，30H ×

错因，不明确把30H存放字节单元还是字单元

MOV [BX]，300H √

只能是字单元存放

4）段寄存器赋值(CS DS SS ES)

MOV DS，1000H ×;不能立即数赋值

由下面替代

MOV AX, 1000H ；可以是BX,CX,DX

MOV DS, AX

MOV CS, AX ×

；不能在程序中给CS用语句赋值，（并不是不能改变CS的值）

5） 不允许内存单元之间直接用MOV传送

MOV [BX]，[1000] ×

MOV [BX]，BUF （字节） ×

可改为

MOV AX，[1000H] ;字传送

MOV [BX]，AX

或

MOV AL，[1000H] ;字节传送

MOV [BX]，AL

2. 堆栈操作指令 PUSH POP （W）

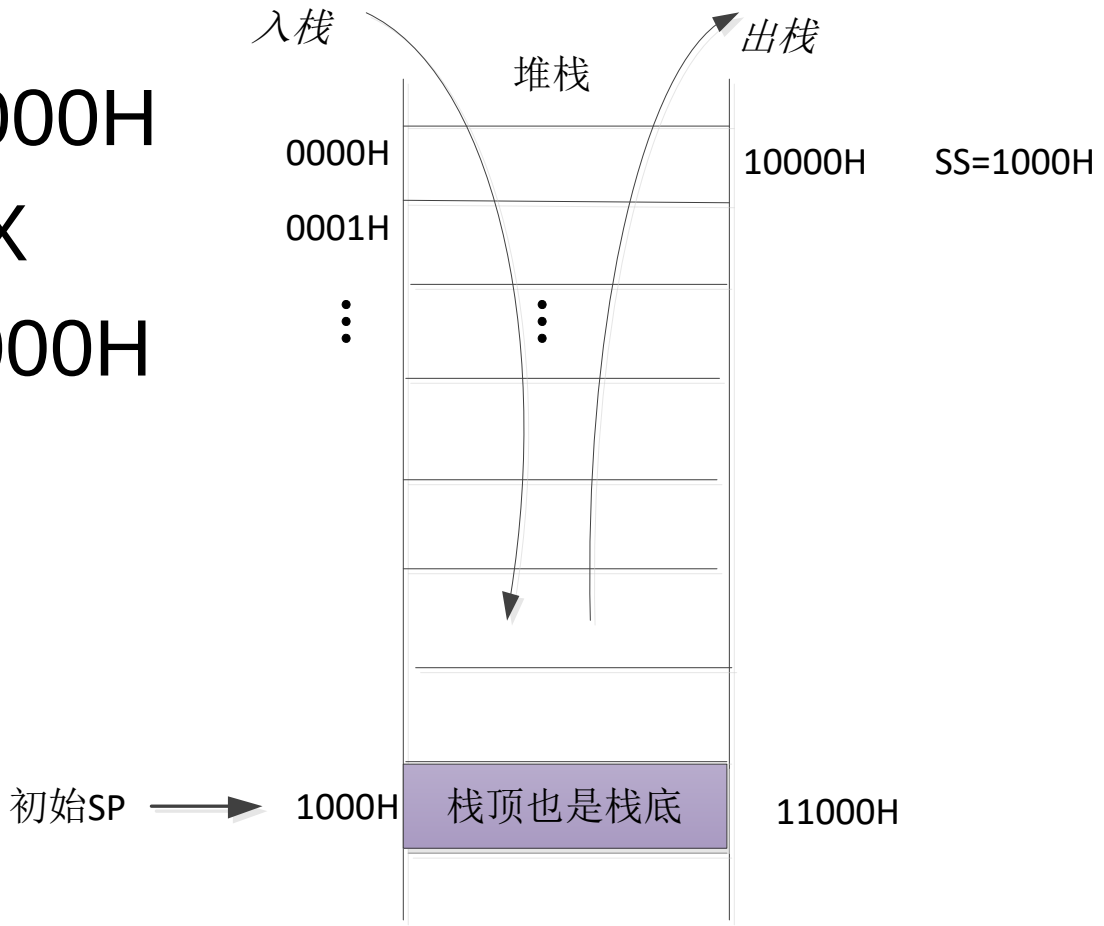
堆栈：是一个采用“先进后出”原则的主存区域，位于堆栈段中，使用SS段寄存器记录其段地址。

堆栈大小（高度）：由堆栈指针寄存器SP设定

SP始终指向 栈顶

堆栈

```
MOV   AX,1000H
MOV  SS, AX
MOV   SP,1000H
```



1） PUSH （W)

MOV AX, 1234H

PUSH AX

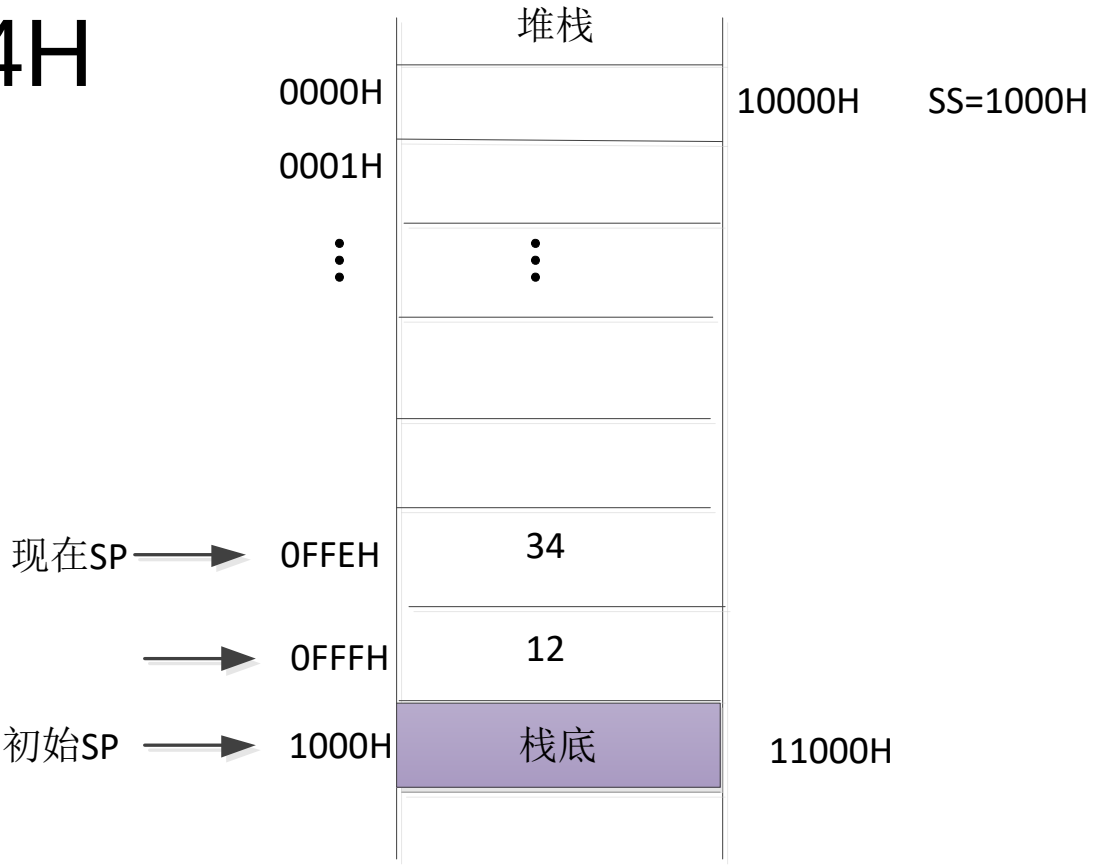
步骤

1）SP-1→SP

2）AH →[SP]

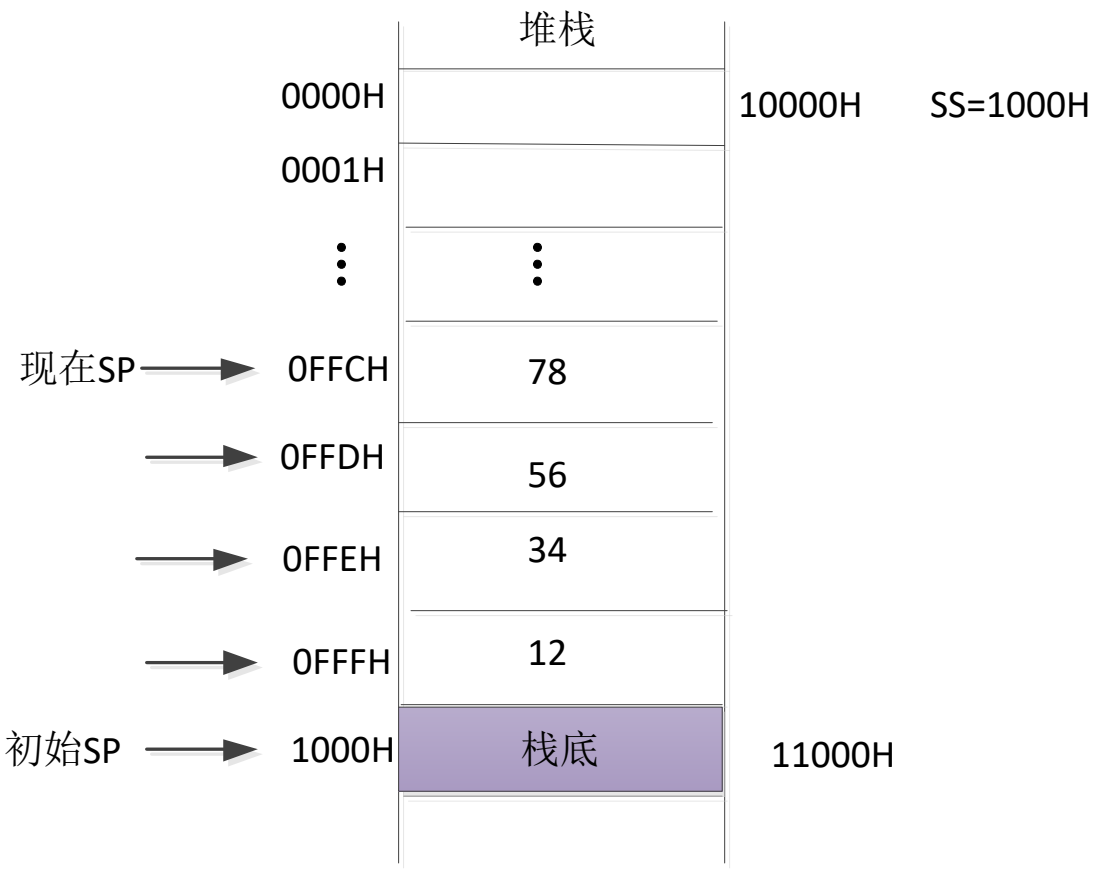
3）SP-1→SP

4）AL → [SP]



一次PUSH压栈操作 SP-2

```
MOV   AX, 1000H
 MOV  SS, AX
 MOV   SP,1000H
 MOV  AX,1234H
 MOV  BX ,5678H
PUSH  AX
PUSH  BX
```



- 2)POP (W)
- MOV AX, 1000H

MOV SS, AX

MOV SP,1000H

MOV AX,1234H

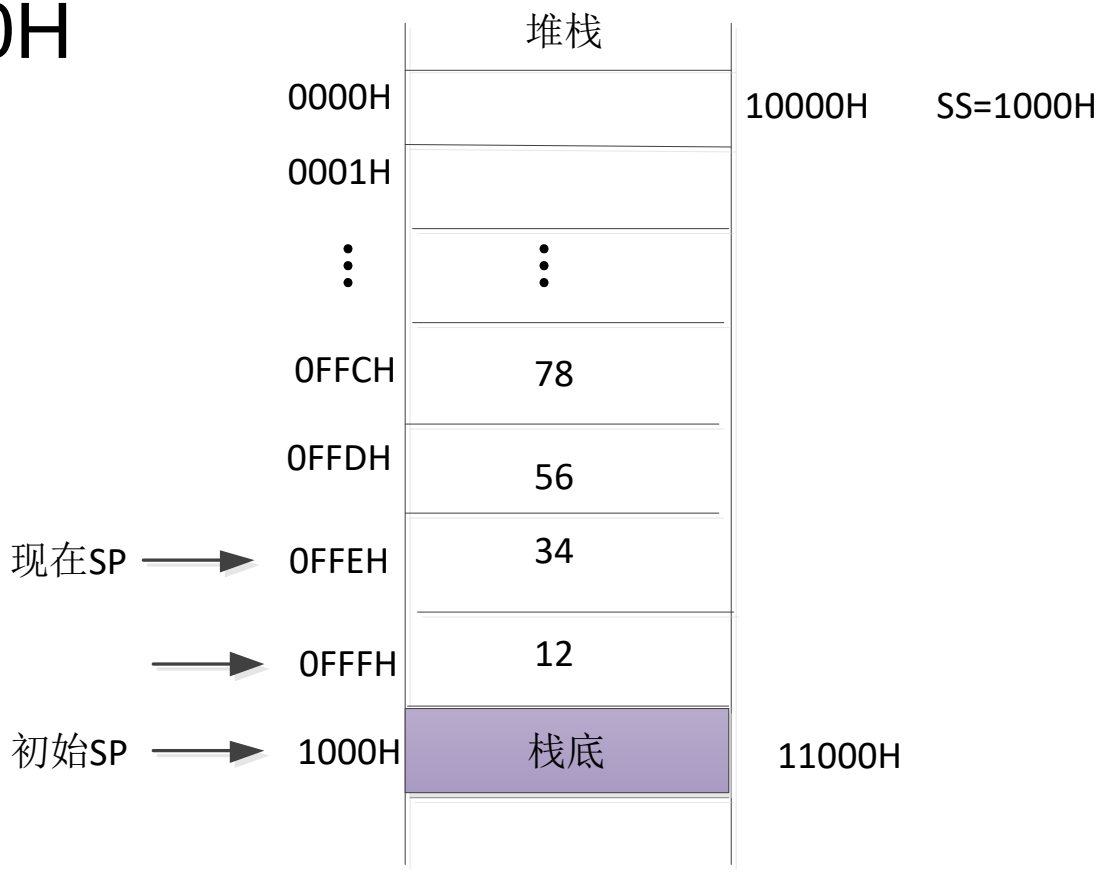
MOV BX ,5678H

PUSH AX

PUSH BX

POP BX

（没有AX出栈）



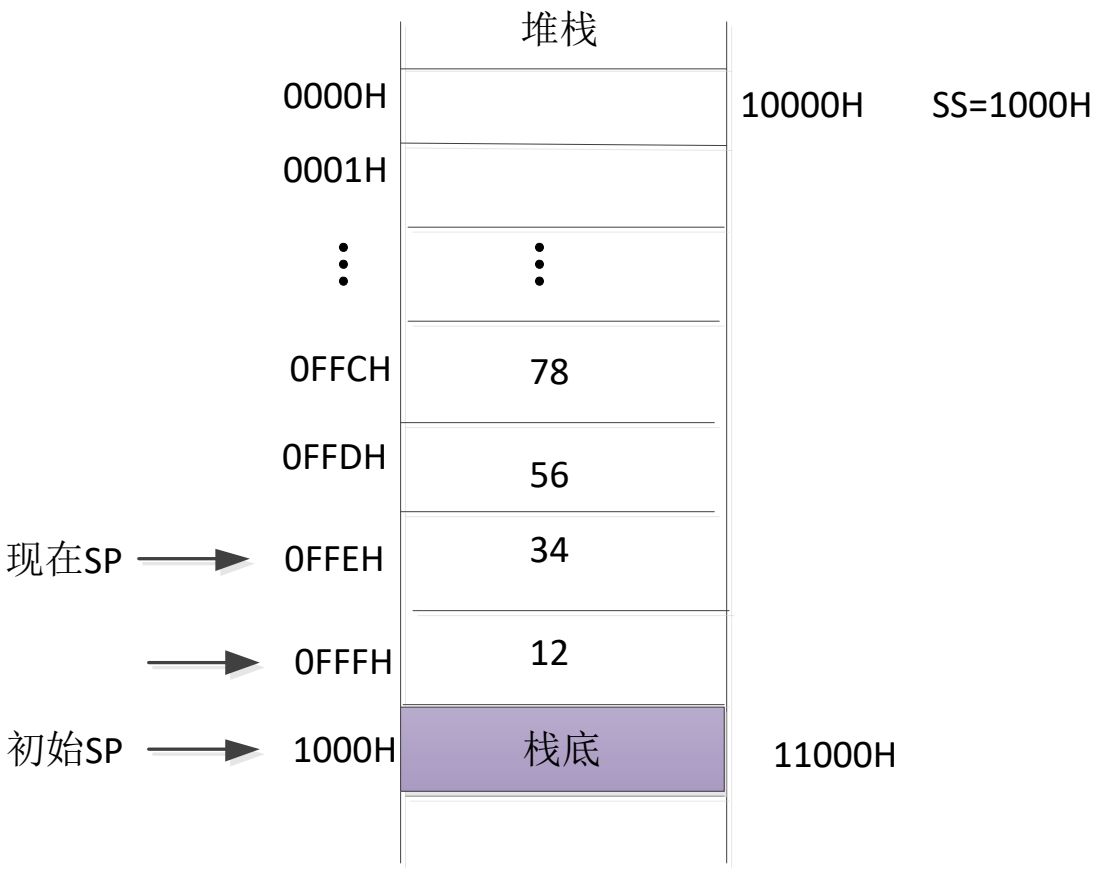
步骤

1）[SP] →AL

2）SP+1→SP

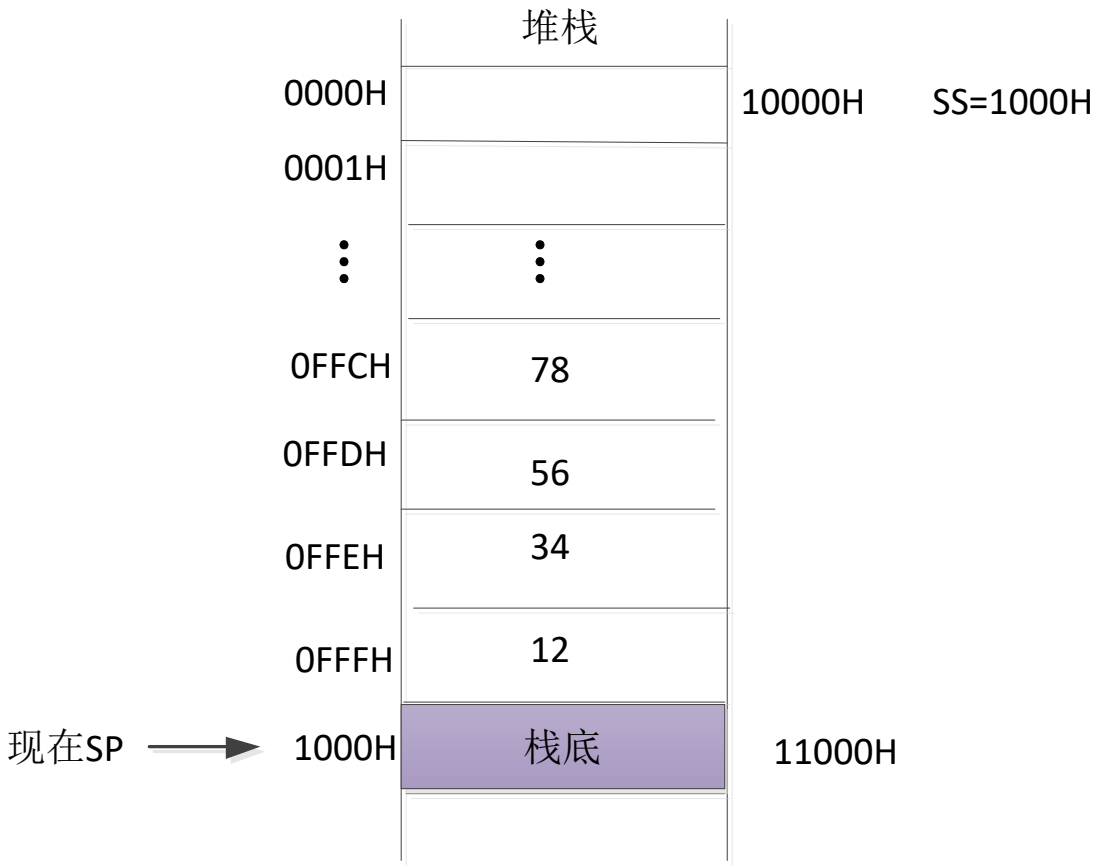
3）[SP] → AH

4）SP+1→SP



一次POP出栈操作 SP+2

```
MOV   AX, 1000H
MOV  SS, AX
MOV   SP,1000H
MOV  AX,1234H
MOV  BX ,5678H
PUSH  AX
PUSH  BX
POP    BX
POP   AX
```



```
MOV  AX,1234H
MOV  BX ,5678H   ；AX，BX存放全局值
；；；破坏AX,BX数据的操作
MOV   AX，0
MOV   BX，0
；；；；；；；；；
MOV   BUF，AX   ；AX，BX的值在此处使用
MOV   BUF1，BX
```

;;解决方法 通过别的寄存器暂存数据

```
 MOV  AX，1234H
 MOV  BX，5678H  ；AX，BX存放全局值
MOV  DX，AX    ；暂存数据到别的寄存器
MOV  CX，BX
 MOV   AX，0  ；某种破坏操作
 MOV   BX，0
MOV   AX，DX  ；恢复原值
MOV   BX，CX
MOV   BUF，AX  ；AX，BX的值在此处使用
MOV  BUF1，BX
```

例1

```
MOV  AX，1234H
MOV  BX，5678H
PUSH  AX
PUSH  BX
MOV   AX，0  ；某种破坏操作
MOV   BX，0
POP   AX
POP   BX
```

执行完以上指令 AX=？ BX=？

例2 MOV AX，1000H

```
MOV  SS，AX
MOV SP， 0100H
MOV AX， 1234H
MOV BX，5678H
PUSH  AX
PUSH BX
MOV   AX，0  ；某种破坏操作
MOV   BX，0
POP   BX
POP   AX
```

执行完以上指令AX=？ BX=？ 栈顶物理地址？

例3 MOV AX，1000H

MOV SS，AX

MOV SP， 0100H

MOV AX，1234H

MOV BX，5678H

PUSH AX

PUSH BX

MOV AX，0 ；某种破坏操作

MOV BX，0

POP BX

执行完以上指令AX=？ BX=？ 栈顶物理地址？

例4

```
MOV  AX，1000H
MOV   SS，AX
MOV  SP，  0100H
MOV  AX，1234H
MOV  BX ，5678H
PUSH   AX
PUSH  BX
POP    BX
```

执行完以上指令AX=？　BX=？　栈顶物理地址？

3.4 算术运算

加减法分进（借）位和不进（借）位

ADD　ADC

SUB　SBB

其他加减法

INC　DEC　（加减1）

　　CMP　（做减法比较）

DAA　DAS　（BCD码运算）

乘除法分符号数和无符号数

MUL　IMUL

DIV　　IDIV

1. ADD ； 加法指令

格式： ADD DST, SRC ； B/W

操作 (DST)←（DST)+(SRC)

例1

MOV AL,56H

MOV BL,24H

ADD AL , BL

执行后

AL=7AH ,BL=24H

CF=0 OF=0 ZF=0 SF=0 AF=0

例2

```
MOV   AL,0FFH
MOV   BL,01H
ADD   AL , BL
```

```
      FFH
    +01H          CF=1 ；OF=0；  ZF=1
  1  00H          SF=0；PF=1；  AF=1
```

- 看成符号数 结果正确
- 看成无符号数 结果不正确

2. ADC ； 带进位加法指令

一般使用在多字节相加

格式： ADC DST, SRC ； B/W

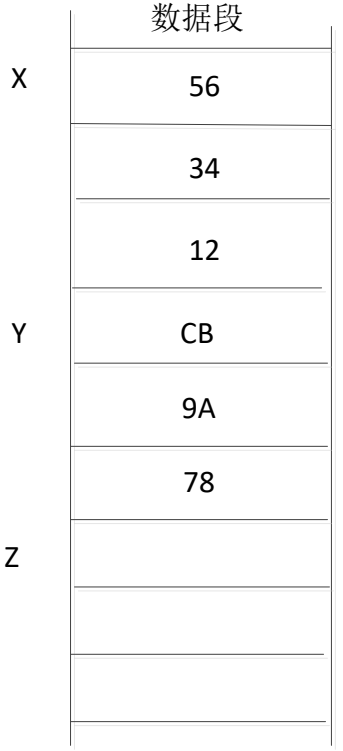
操作 (DST)←（DST)+(SRC)+CF

例1 将变量X和Y中的3字节数据相加结果存入Z开始的三个字节单元中

X 123456H

Y 789ACBH

Z 8ACF21H



```
MOV   AL , X     ;  取X变量的最低字节数据
ADD   AL, Y      ;  X和Y最低字节相加
MOV   Z  ,  AL   ;  结果存入Z变量最低单元
MOV   AL , X+1   ;  取X+1单元字节数据
ADC   AL  ,  Y+1 ;  (X+1)+(Y+1)+CF
MOV   Z+1,  AL   ;  结果存入Z+1单元
MOV   AL , X+2   ;
ADC   AL ,  Y+2  ;  (X+2)+(Y+2)+CF
MOV   Z+2,   AL
```

解决ADD指令例2中无符号溢出的问题

```
MOV   AL,0FFH  ；看成无符号数
MOV   BL,01H   ；看成无符号数
ADD   AL , BL
    FFH
+   01H          CF=1；
1   00H
```

解决ADD指令例2中无符号溢出的问题

方法1

```
MOV   AX,0FFH  ；将值付给一个16位寄存器
MOV   BX,01H   ；ADD   AX , BX
   00FFH
+  0001H
  0100H
```

方法2：

```
MOV   AL,0FFH  ；看成无符号数
MOV   BL,01H   ；看成无符号数
ADD   AL , BL    ；此处CF=1
   FFH
+  01H          CF=1 ；
1  00H
MOV  AH,0  ；
ADC   AH,0  ；CF=1
```

最后 AX=0100H

解决符号数加法有溢出的问题

例3

```
MOV   AL,0FFH  ；看成符号数    -1
MOV   BL,80H   ；看成符号数  -128
ADD   AL , BL    ；此处OF=1
   FFH
+  80H
1  7FH       OF=1；CF=1
```

符号数进行位数拓展

采用指令 CBW　　将 字节→ 字数据

符号数进行位数拓展

采用指令 CBW　　将 字节→ 字数据

如果是正数（字节）　高8位　补00H

负数（字节）　高8位　补FFH

CBW

1）必须将待拓展字节数据存入AL

2）CBW

3）拓展完的16位数在AX里

```
MOV   AL,0FFH  ；看成符号数   -1
CBW
MOV    DX  ,AX   ；暂存结果FFFFH到DX
MOV   BL,80H    ；看成符号数  -128
MOV   AL ,BL     ；BL数据存入AL中准备拓展
CBW              ；BL拓展完为FF80H
MOV   BX,AX      ；将数据还原到BX中
MOV   AX, DX     ；将暂存数据还原到AX中
ADD   AX,BX      ；最终结果在AX中
```

3. SUB ； 减法指令

格式： SUB DST, SRC ； B/W

操作 (DST)←（DST)-(SRC)

例1

```
MOV   AX,9754H          9754H
MOV   BX,2378H        - 2378H
SUB   AX , BX           74DCH
```

执行后

AX=73DCH ,BX=2378H

CF=0 OF=1 ZF=0 SF=0 AF=1

4. SBB ； 带借位减法指令

格式： SBB DST, SRC ； B/W

操作 (DST)←（DST)-(SRC)-CF

例1

```
MOV   AX ,3654H
MOV   BX ,4775H
SUB   AX , BX
MOV   CX ,1234H
SBB   AX,CX
```

例1

MOV AX,3654H

MOV BX,4775H

SUB AX , BX

CF=1 OF=0 ZF=0 SF=1 AF=1

MOV CX ,1234H

SBB AX,CX

AX= DCAAH

BX=4775H

CX=1234H

CF=0 OF=0 ZF=0 SF=1 AF=0

```
  3654H
- 4775H
  EEDFH

  EEDFH
- 1234H
- 0001H
  DCAAH
```

5. INC ； 加1

格式： INC SRC ； B/W

操作 (SRC)←(SRC)+1

例1

```
MOV   AX,0FFFFH
INC   AX
```

AX=0000H

6. DEC ； 减1

格式：DEC SRC ； B/W

操作 (SRC)←(SRC)-1

例1

```
MOV  AX,0FFFH
DEC  AX
```

AX=0FFEH

例2

```
MOV   AX,0
DEC   AX
```

AX=FFFFH

7. MUL ； 无符号乘法指令

格式： MUL SRC ； B/W

字节操作： AX←(SRC)*AL ;

8位×8位指令要求注意

1)被乘数必须存放在AL

2)乘数不能是立即数

例1 用汇编语言表示 100*2

MOV AL,100

MUL 2 ； X

正确

```
MOV   AL,100  ;被乘数先存放在AL中
MOV   BL ,2     ;可以使用其他8位寄存器
MUL   BL
```

结果在 AX=00C8H

字操作　DX,AX←(SRC)*AX

字乘法指令要求注意

1)被乘数字数据必须存放在AX

2)乘数不能是立即数

例2 用汇编语言表示 100*267

```
MOV   AX,100
MOV   BX,267
MUL   BX
```

结果 DX= 0　　AX=684CH

例3 用汇编语言表示

```
MOV   AL,-1     ;FFH
MOV   BL,1
MUL   BL
```

结果 AX= 00FFH

看成符号数：

```
MOV   AL,-1     ;FFH
MOV   BL,1
IMUL   BL
```

结果 AX= FFFFH

8. IMUL ； 符号乘法指令

格式： IMUL SRC ； B/W

字节操作 AX←(SRC)*AL ;

8位×8位指令要求注意

1）被乘数必须存放在AL

2）乘数不能是立即数

例1 用汇编语言表示 -100*2

```
MOV   AL,-100  ;被乘数先存放在AL中
MOV   BL ,2    ;可以使用其他8位寄存器
IMUL   BL
```

结果在 AX=FF38H （-200的补码）

字操作　DX,AX←(SRC)*AX

字乘法指令要求注意

1）被乘数字数据必须存放在AX

2）乘数不能是立即数

例2 用汇编语言表示 100*267

```
MOV   AX,100
MOV   BX,267
IMUL   BX
```

结果 DX= 0　　AX=684CH

例3

```
MOV   AX,100
MOV   BX,267
MUL   BX
```

结果 DX= 0　　AX=684CH

例4 用汇编语言编程实现字节变量（无符号数）的算术运算表达X*Y+Z，并将结果存入到字变量RESULT中

```
    数据段定义
      X       DB   1
      Y       DB   2
      Z       DB   80H
  RESULT   DW
```

无符号数

```
MOV   AL, X
MUL   Y          ;X*Y→AX
MOV   BX,AX

MOV  AL , Z
CBW
ADD AX,BX

MOV  RESLUT ,AX
```

无符号数

```
MOV   AL, X
MUL   Y          ;X*Y→AX

MOV   BL , Z
MOV   BH, 0
ADD AX,BX

MOV  RESLUT ,AX
```

无符号数

```
MOV   AL, X
MUL   Y          ;X*Y→AX
ADD  AL , Z
ADC  AH,0     ;X*Y+Z→AX
MOV  RESLUT ,AX
```

有符号数

```
MOV   AL, X
IMUL   Y          ;X*Y→AX
ADD  AL , Z
ADC  AH,0     ;X*Y+Z→AX
MOV  RESLUT ,AX
```

错误写法

有符号数正确编程

```
MOV   AL, X
IMUL   Y          ;X*Y→AX
MOV BX, AX   ;暂存结果
MOV  AL , Z
CBW               ;拓展Z为16位
ADD  BX ,AX  ;X*Y+Z
MOV  RESULT ,BX
```

9. DIV ； 无符号除法指令

格式： DIV SRC ； B/W

字节操作 ：

AL←(AX)/(SRC) 商;

AH←(AX)/(SRC) 余数;

除法指令要求注意

1、被除数字节数据必须存放在AX

2、除数不能是立即数

例1 用汇编语言表示 100/2

```
MOV   AX,100
MOV   BL,2
DIV   BL
AH=0   AL=32H(50)
```

例2 用汇编语言表示 600/2

```
MOV   AX,600
MOV   BL,2
DIV   BL
 AL=?
```

结果是 OVERFLOW

10. IDIV ； 符号除法指令

格式： IDIV SRC ； B/W

字节操作 ：

AL←(AX)/(SRC) 商;

AH←(AX)/(SRC) 余数;

除法指令要求注意

1、被除数字节数据必须存放在AX

2、除数不能是立即数

例1 用汇编语言表示 -100/2

```
MOV   AX,-100
MOV   BL,2
IDIV   BL
AH= 0      AL=  CEH(-50)
MOV   AX,-100
MOV   BL,2
DIV   BL
AH= 0     AL= 4EH
```

例2 用汇编语言编程实现字节变量（无符号数）的算术运算表达(X*Y-Z)/W，并将商和余数分别存入到变量RESULT开始的两个单元中

```
    数据段定义
      X       DB   1
      Y       DB   2
      Z       DB   80H
      W       DB    2
  RESULT   DB   ?,?
```

无符号数

```
MOV   AL, X
MUL   Y          ;X*Y→AX
SUB  AL , Z
SBB  AH,0        ;X*Y-Z→AX
DIV   W          ; (X*Y-Z)/W
MOV  RESULT ,  AL       ;存商
MOV  RESULT+1, AH    ;存余数
```

当变量都为符号数编程

```
MOV   AL, X
IMUL   Y          ;X*Y→AX
MOV BX, AX   ;暂存结果
MOV  AL , Z
CBW               ;拓展Z为16位
SUB  BX ,AX  ;X*Y-Z
MOV  AX,BX  ;或  XCHG    AX, BX
IDIV   W          ;(X*Y-Z)/W
MOV   RESULT  ,AL
MOV   RESULT+1,AH
```

补 DAA　只能在AL调整

AX=6498H　BX=2677H 要求将这两个数按十进制相加，结果存在AX

```
ADD   AL,BL
DAA
MOV  CL,AL ;暂存在CL
MOV  AL, AH
ADC  AL ,BH
DAA
MOV   AH,AL
MOV   AL,CL
```

3.5 逻辑运算及移位指令

逻辑运算：AND OR XOR TEST NOT

移位：

SHL	SHR
SAL	SAR
ROL	ROR
RCL	RCR

一、逻辑运算

1.AND

格式： AND DST, SRC ； B/W

功能： (DST)←(SRC) & (DST) ;

例1 MOV AL,66H

AND AL,0FH

执行后 AL=06H

起到屏蔽某些位的作用

2.OR

格式： OR DST，SRC ； B/W

功能： (DST)←(SRC) ^ (DST) ;

例1 MOV AL,66H

OR AL,0FH

执行后 AL=6FH

使某些位置1的作用

3.XOR

格式： XOR DST，SRC ； B/W

功能： (DST)←(SRC)⊕(DST) ;

例1 MOV AL,66H

XOR AL, 0FH

执行后 AL=69H

使某些位取反

例2　MOV　AL,66H

　　XOR　AL, AL

执行后　AL=00H

自身异或等于清零

写出能使寄存器AX清零的所有功能语句

W　DW 0000H　　MOV　AX,W

MOV　BL ,0

MUL　BL

1） MOV AX,0

2）XOR AX, AX

3）AND AX,0

4）SUB AX,AX

LEA AX, [0000H]

5) MOV BL ,0

MUL BL ; AX ←BL*AL

4.NOT

格式： NOT DST ； B/W

功能： (DST)←$\overline{D}\overline{S}\overline{T}$

例1 MOV AL,66H

NOT AL

执行后 AL=99H

使数据整体取反

5.TEST ;测试

格式： TEST DST，SRC ； B/W

功能：(SRC) & (DST)；只做与不保留结果

例1 测试AL的D_7 的是否为0，不为0跳转

```
TEST  AL，80H
JNZ  NEXT
```

NEXT:

总结逻辑运算指令对标志位的影响

AND OR XOR TEST 使得 CF=0; OF=0

- 写出能使寄存器AX清零，且CF=0,OF=0的所有功能语句

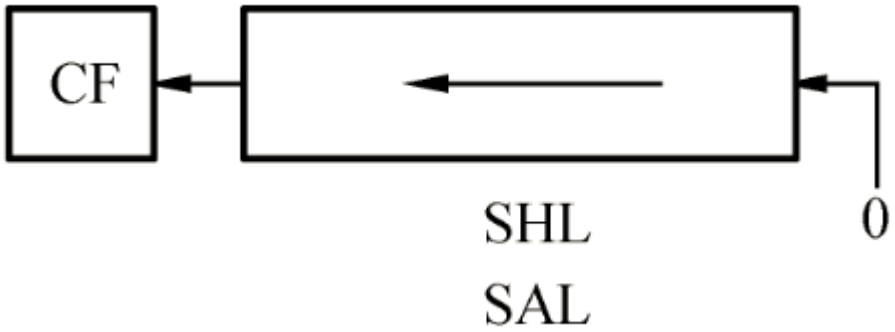
二、移位指令

1. SHL（Shift logical left）　　逻辑左移

　SAL（Shift arithmetic left）　算术左移

格式：　SHL DST，SRC ; B/W

　　SAL DST. SRC :B/W



例1 将DH的内容左移3位

　MOV　DH，01H

SHL　DH ,03H; ×

移位一次可以直接写成

SHL　DH ,01H ；√

MOV　CL, 03H ；移位次数大于1，存放在CL

SHL　DH ,CL

结果为 DH =08H

左移一次满足×2的关系：

1.对于无符数，必须每次移位完CF=0, 也就是移出的是0。

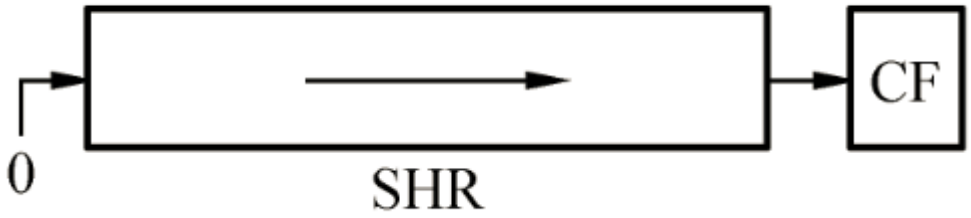
2. 对于符号数，必须每次移位完最高位和CF要同为1或0。

```
MOV   AL，0FFH；看成符号数为-1
SHL  AL，1          ；AL=FEH  为-2
```

- 将内存中的字节变量X的高低四位（0-9，没有A-F的数据）分别变成ASCII，并存放在在字节变量Y开始两个单元中（高四位存高地址）。
- 将内存中的字节变量X的个、十、百分别存入到字节变量GEI,SHI，BAI中。
- 将内存中的字变量X的个、十、百、千以及万分别存入到字节变量GEI,SHI，BAI，QIAN,WAN中。

2. SHR（Shift logical right） 逻辑右移

格式： SHR DST，SRC ; B/W



无符号数 ÷2

例1 MOV AL，F6H ； 246

SHR AL，1 ； AL=7BH ， 123

例2 MOV AL，F7H ； 247

SHR AL，1 ； AL=7BH ， 123

3.SAR（Shift arithmetic right） 算术右移

格式： SAR DST，SRC ;B/W

保持最高位不变

符号数 ÷2

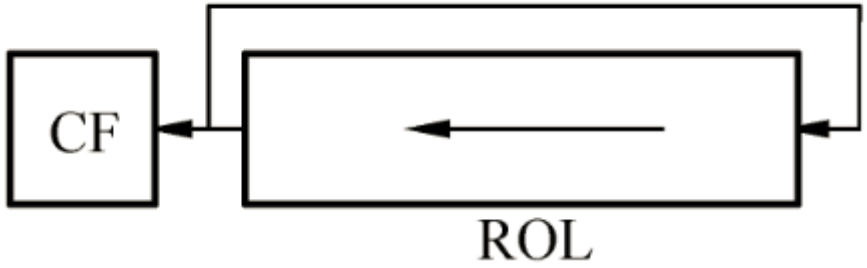
例1 MOV AL，0F6H；-10

SAR AL，1 ；AL=FBH，-5

三、循环移位指令

1.ROL（Rotate left）

格式： ROL DST，SRC ; B/W



例1 MOV AL ，0C3H

ROL AL ，1

执行后 CF=1 AL=87H

2.ROR（Rotate right）

格式： ROR DST，SRC ; B/W

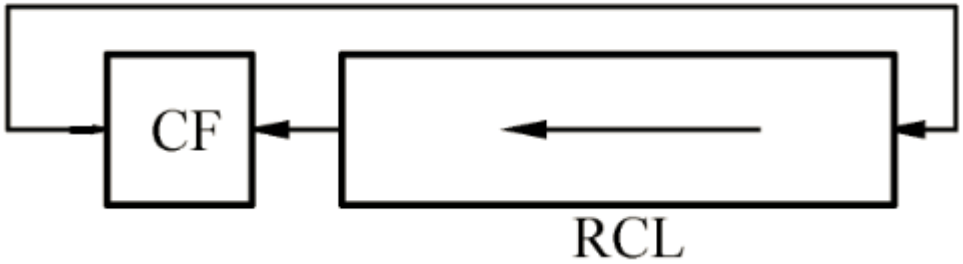
例1 MOV AL ，0C3H

ROR AL ，1

执行后 CF=1 AL=E1H

3.RCL（Rotate left through carry）

格式： RCL DST，SRC ; B/W



例1 XOR AL， AL ；CF=0

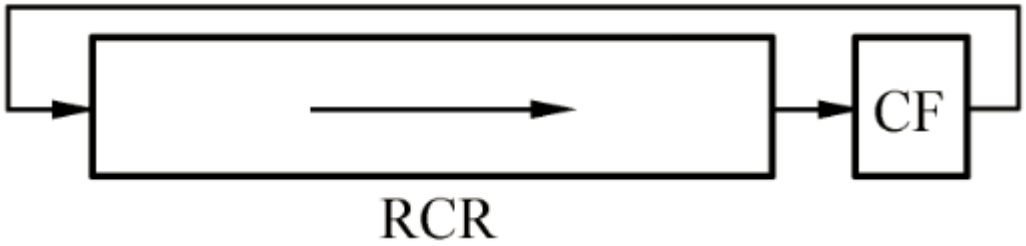
MOV AL ，0C3H

RCL AL ，1

执行后 CF=1 AL=86H

4.RCR（Rotate right through carry）

格式： RCR DST，SRC ; B/W



例1 XOR AL， AL ； CF=0

MOV AL ，0C3H

RCR AL ，1

执行后 CF=1 AL=61H

3.6 程序控制指令

控制程序执行顺序

- **无条件转移指令**
- **条件转移指令**
- **循环控制指令**
- **过程（子程序）调用和返回**

（下一章子程序讲）

3.6.1无条件转移指令

1.段内直接转移指令

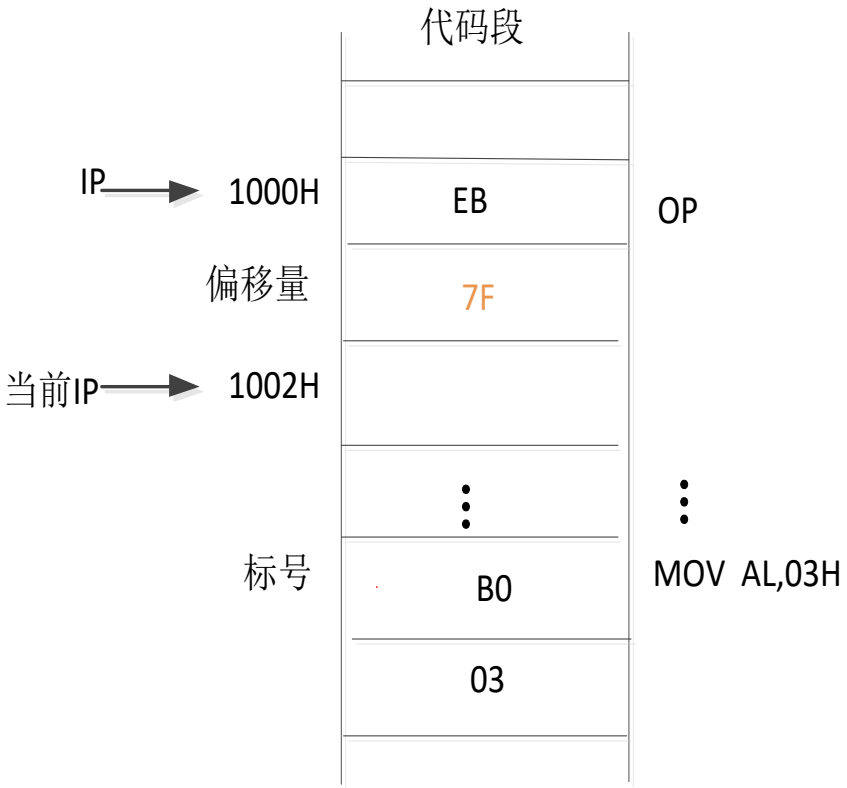
1）JMP SHORT 标号 ；段内短跳转

跳转范围：-128--- 127

形如：

```
        JMP  SHORT  NEXT
          。 。 。
NEXT：MOV AL，03H ；跳转的目标语句
```

目标IP =当前IP+偏移量



向前（正向）跳转：

转移地址（目标）IP

=当前IP+00 8位偏移量

向后（负向）跳转：

转移地址（目标）IP

=当前IP+FF 8位偏移量

4－16　假设指令 JMP SHORT NEXT 存在代码段的 2100H,2101H 单元中,它的相对量为(1)38H;(2)0D8H。请写出每种相对量的转移地址是什么？写出计算过程。

2）JMP NEAR 标号 ；段内直接近转移

跳转范围：-32768--- 32767

形如：

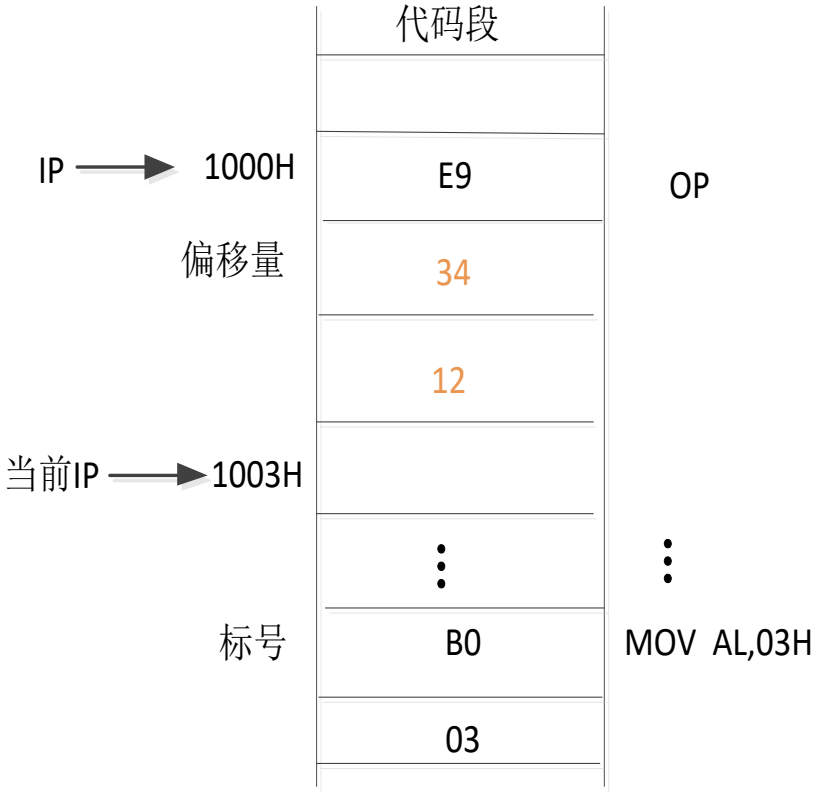
```
    JMP   NEAR   NEXT1
          。  。  。
NEXT1：MOV  AL，03H ；跳转的目标语
          。  。  。
```

段内直接转移 JMP 标号

目标IP =当前IP+偏移量

向前（正向）跳转：
转移地址（目标）IP
=当前IP+16位偏移量

向后（负向）跳转：
转移地址（目标）IP
=当前IP+16位偏移量



2.段内间接转移指令

JMP （WORD PTR） OPR ；段间间接转移

例1. JMP (WORD PTR) BX

BX=3000H，执行指令后，IP=3000H

例2. JMP WORD PTR [SI+400H]

SI=2000H,[2400H]=44H,[2401H]=12H

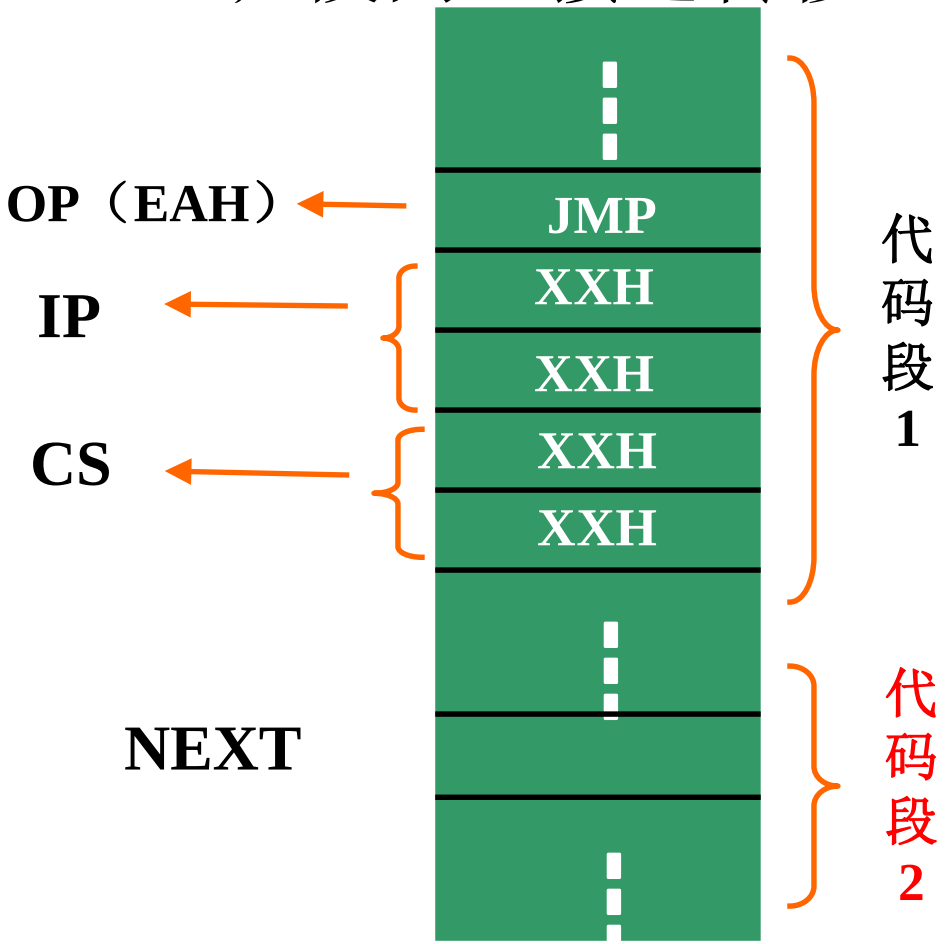
执行后，IP=1244H

段内转移指令，只改变IP，不改变CS

3.段间直接远转移指令

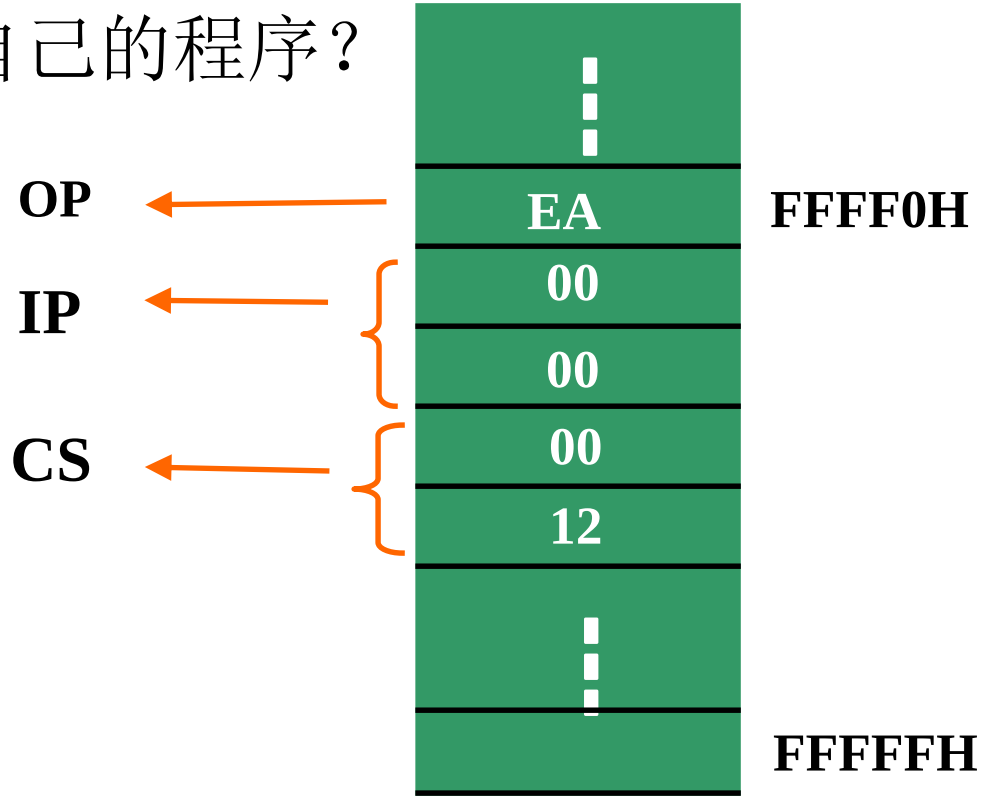
JMP FAR PTR OPR ；段间直接远转移

```
C1    SEGMENT
      ◦ ◦ ◦
       JMP   NEXT
  ◦ ◦ ◦ ◦ ◦
   C1    ENDS
C2    SEGMENT
      ◦ ◦ ◦
NEXT：
  ◦ ◦ ◦ ◦ ◦
   C2    ENDS
```



例1 段间直接远转移指令应用

CPU复位时从FFFF0H开始执行指令，如何使CPU跳转到内存12000H地址（代码段首地址）去执行软件设计者自己的程序？



4.段内间接转移指令

JMP DWORD PTR OPR ；段间间接转移

```
C1    SEGMENT
    。 。 。
      JMP   DWORD PRT [SI]
  。 。 。 。 。
   C1    ENDS
  C2    SEGMENT
    。 。 。
目标标号：
  。 。 。 。 。
   C2    ENDS
```

设SI=1600H：

[1600H]=34H

[1601H]=12H

[1602H]=00H

[1603H]=10H

跳转目标：

IP=1234H

CS =1000H

3.6.2 条件转移指令

以某个引起标志位变化的操作为前提，再根据标志位做跳转。

1.单个标志位

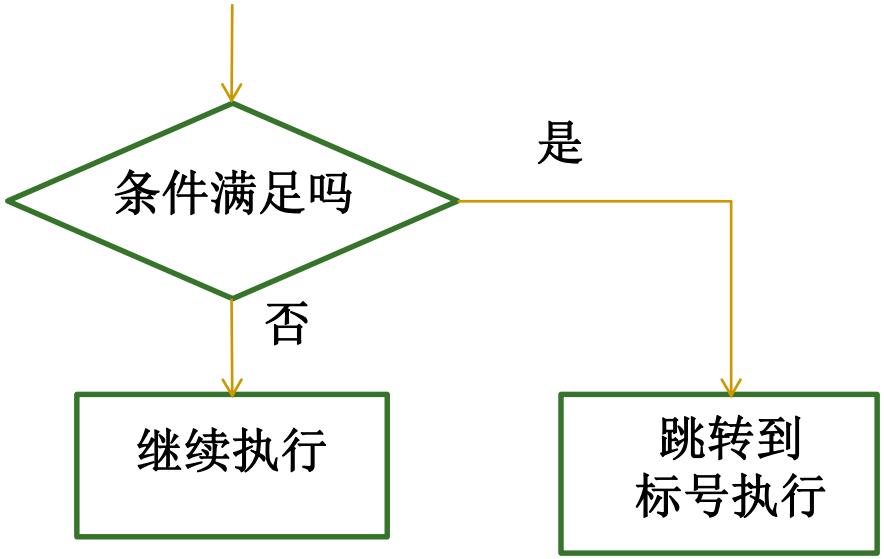
JC/JNC CF=1/CF=0

JZ/JNZ ZF=1/ZF=0

JO/JNO OF=1/OF=0

JS/JNS SF=1/SF=0

JP/JPE PF=1/PF=0



条件转移指令流程图

- 例 比较两个16位的字数据，当二者相等，则跳转到动作2（BX=1），否则顺序执行动作1（BX=0）

```
            CMP   AX ，DX
            JZ    ACTION2
ACTION1:  MOV   BX , 0

ACTION2:  MOV   BX , 1
    OUTT:
```

2.无符号数比较转移（A，B，E）

一般操作为减法（CMP）

JA（JNBE）	CF=0	X>Y
JAE（JNB）	CF=0或ZF=1	X>Y或 X=Y
JB（JNAE）	CF=1	X<Y
JBE（JNA）	CF=1 或ZF=1	X<Y或 X=Y

- 例 当一个16位无符号 X，
  - X<100 ，AH=1
  - X=100， AH=2　编写相关程序段
  - X>100 ， AH=3

```
MOV   BX，  X
CMP   BX，  100
JA    A_100
JZ    E_100
MOV  AH ,1

E_100: MOV  AH ,2

A_100: MOV  AH ,3
```

3.符号数比较转移（L，G，E）

一般操作减法（CMP）

JG（JNLE）　SF⊕OF=0 且ZF=0；X>Y

JGE（JNL）　SF⊕OF=0 或 ZF=1；X≥Y

JL（JNAE）　SF⊕OF=1 且ZF=0；X<Y

JLE（JNA）　SF⊕OF=1 或 ZF=1；X≤Y

符号数X，Y　　做 X-Y

1.　X>Y

1）X>Y>0　　SF=0（正数）　OF=0（同号减）

2）0>X>Y　　SF=0（正数）　OF=0（同号减）

3）X>0 , Y<0　　SF=0（正数）　OF=0（异号减）

　　SF=1（负数）　OF=1（异号减）

2.　X<Y

1）X<Y<0　　SF=1（负数）　OF=0（同号减）

2　0<X<Y　　SF=1（负数）　OF=0（同号减）

3）X<0 , Y>0　　SF=1（负数）　OF=0（异号减）

　　SF=0（正数）　OF=1（异号减）

- 例 当一个16位符号数 X，

  X<100 ，AH=1

  X=100， AH=2　编写相关程序段

  X>100 ， AH=3

```
    MOV   BX，  X
    CMP   BX，  100
    JG    G_100
    JZ    E_100
    MOV  AH ,1
     JMP   OUTT
E_100: MOV  AH ,2
     JMP   OUTT
G_100: MOV  AH ,3
 OUTT；
```

3.6.3 循环控制指令

1. LOOP 标号

先CX←CX-1，若CX 不等于0 ， 继续跳转标号处循环。

```
MOV   CX , N          MOV  DX , N
LOP: …          LOP:  ….
      …                ….
                       DEC   DX
     LOOP  LOP         JNZ   LOP
```

例1.

```
    MOV  CX，10
    XOR  AX，AX
LOP: ADD  AX ,CX
    LOOP  LOP
```

执行完上面程序段

AX=?

CX=?

例2 对数组BUF中的4字节无符号数据求和并存入SUM字变量中。

```
BUF   DB  10，20，30，40
SUM   DW  0
        。。。
      MOV  AX，0
      MOV  CX，4
      MOV  BX，OFFSET BUF
LOP1：ADD  AL，[BX]
      ADC  AH，0
      INC  BX
      LOOP  LOP1
      MOV  SUM，AX
```

例3 对数组BUF中的4字节无符号数据求和并存入SUM字变量中。

采用相对寄存器寻址

```
BUF   DB  10，20，30，40
SUM   DW  0
      。 。 。
        MOV  AX，0
        MOV  CX，4
         MOV  BX，0
LOP1：ADD   AL，BUF[BX]
         ADC   AH，0
         INC   BX
         LOOP   LOP1
         MOV   SUM，AX
```

采用基址变址寻址

```
      。 。 。
        MOV  AX，0
        MOV  CX，4
         MOV  SI，0
        MOV  BX，OFFSET  BUF
LOP1：ADD   AL，[BX+SI]
         ADC   AH，0
         INC   SI
        LOOP   LOP1
         MOV   SUM，AX
```

例4 对数组BUF中的4字节符号数求和并存入SUM字变量中。

采用寄存器间接寻址

```
BUF   DB   0FEH，80H，0FFH, 81H
SUM   DW   0
        。 。 。
    ;;;  MOV  AX，0
         MOV  CX，4
         LEA   BX, BUF
LOP1：  MOV  AL, [BX]
         CBW
         ADD SUM ,AX
         LOOP   LOP1
```

采用基址变址寻址

```
          。 。 。
    ;;;  MOV  AX，0
         MOV  CX，4
         MOV  SI，0
         MOV  BX，OFFSET  BUF
LOP1：  MOV   AL，[BX+SI]
         CBW
         ADD SUM,AX
         INC   SI
         LOOP   LOP1
```

例5 对数组BUF中的4无符号字数据求和，并存入SUM字变量中(每个字数据不超过10000)。

采用相对寄存器寻址

```
BUF   DW  10，20，30，40
SUM   DW  0
      。 。 。
      MOV  CX，4
      MOV  BX，0
LOP1：ADD  SUM，BUF[BX]
      INC  BX
      INC  BX
      LOOP  LOP1
```

采用基址变址寻址

```
      。 。 。
      MOV  AX，0
      MOV  CX，4
      MOV  SI，0
      MOV  BX，OFFSET  BUF
LOP1：ADD  AX，[BX+SI]
      INC  SI
      INC  SI
      LOOP  LOP1
      MOV  SUM，AX
```

3.7 CPU控制指令

CLC ； 使得 CF=0

STC ； 使得 CF=1

CLI ； 使得 IF=0

STI ； 使得 IF=1

HLT ； 程序暂停执行

NOP ； 空操作