第3章 寻址方式与指令系统

3.1 指令系统概述

3.2 寻址方式

3.3 8086CPU指令系统

3.4 80x86新增指令简介

3.1 指令系统概述

一、基本概念

指令：计算机能够识别和执行的操作命令，每条指令都明确规定了计算机必须完成的一套操作以及对哪一组操作数进行操作

程序：由完成一个完整任务的一系列有序指令组成的。

指令系统：CPU所能执行的各种指令的集合，它定义了计算机硬件所能完成的基本操作。

3.1 指令格式

二、8086汇编指令格式

［标号：］　操作码　操作数　［；注释］

说明要执行的是什么操作

操作对象，可以有0个、1个或2个

目的　　源

操作码　操作数，操作数

操作码与操作数之间要用空格分隔;若为双操作数，之间要用逗号“，”分隔

标号：一条指令的位移地址，汇编源程序中，只有在需要转向一条指令处时，才需为该指令设置标号

注释：是对有关指令及程序功能的标注性说明，以增加程序的可读性

三、操作数类型

1、立即数（常数）

指具有固定数值的操作数（即常数），它具体可以是一个字节、字或双字

取值范围如下表：

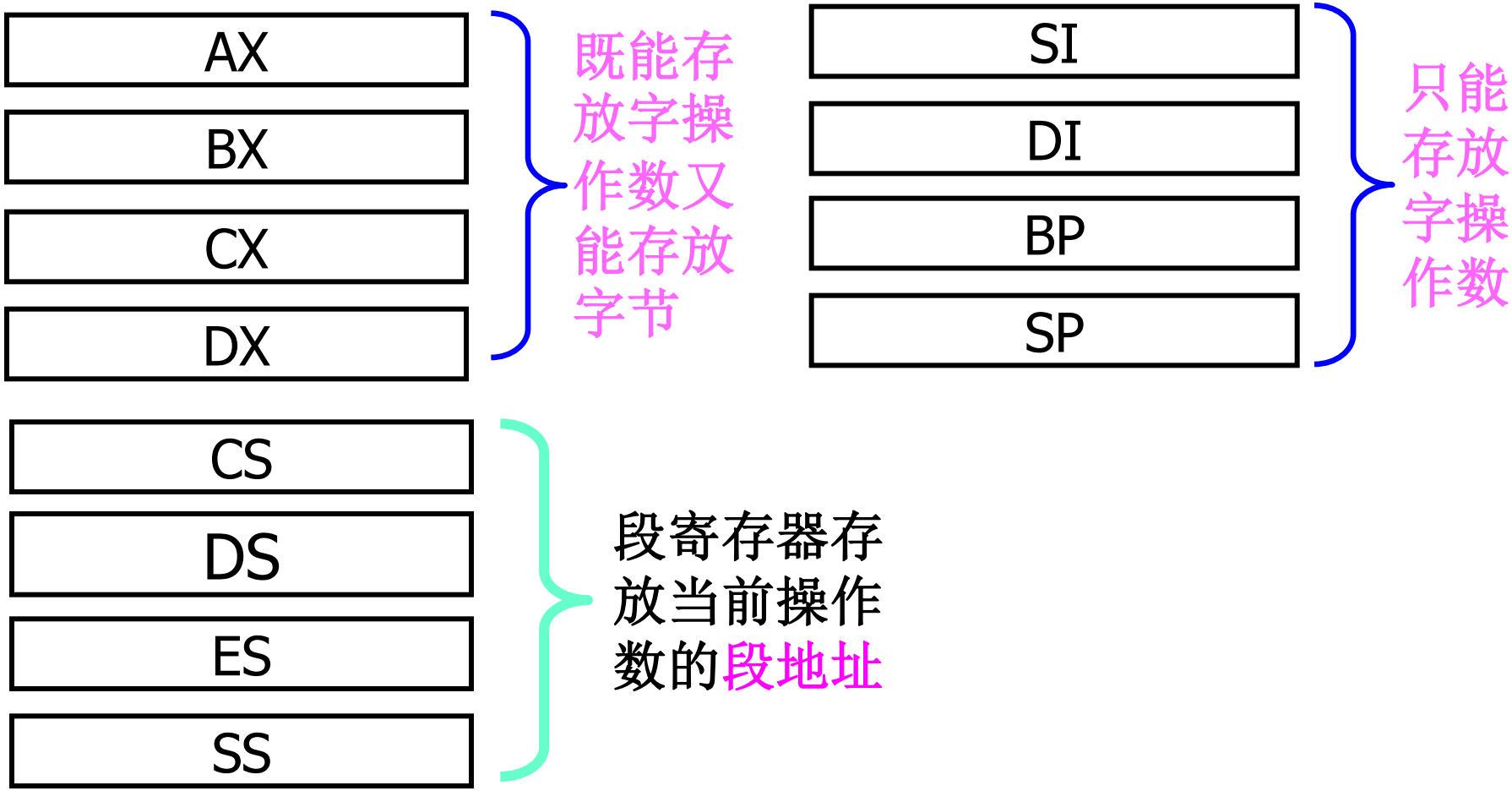
	8位	16位
无符号数	00H-FFH(0-255)	0000H-FFFFH(0-65535)
带符号数	80H-7FH(-128~127)	**8000H-7FFFH(-32768~32767)**

注意：立即数只能用作源操作数

3.1 指令格式

2、寄存器操作数

放在8个通用寄存器或4个段寄存器中的操作数



3.1 指令格式

3、存储器操作数

存储器操作数：存放在存储单元中的操作数

存储器操作数	类型	存储单元个数
	字节	1
	字	2
	双字	4

注意：大部分指令不允许两个操作数同时为存储器操作数

说明：

- 若指令中没有指明所涉及的段寄存器，CPU就采用默认的段寄存器来确定操作数所在的段。
- 存储器操作数的偏移地址可以通过不同的寻址方式由指令给出。

3.2 寻址方式

寻址方式——操作数的存放形式与如何寻找操作数。

依据寻址方式，可方便地访问到各类操作数

在8086指令系统中，说明操作数所在地址的寻址方式可分为3大种（7小种）：

- 立即数寻址方式
- 寄存器寻址方式
- 存储器寻址方式
  - 寄存器间接寻址
  - 寄存器相对寻址
  - 基址变址寻址
  - 直接寻址
  - 基址变址相对寻址

3.2 寻址方式

3.2.1、立即寻址

操作数包含在指令中，紧跟着操作码并与操作码一起存放在代码段中，与代码一起被取入CPU的指令队列，指令执行时不需要再访问存储器。

立即寻址中的操作数可以是计算机有效数制下的数值，也可以是带单引号的字符。

例如： MOV AX，3102H ；AX ← 3102H,执行后，

MOV AL，‘A’ ；AL=41H

分析以下指令的错误原因：

MOV 05H，AL　　MOV BL，324D

MOV CH，2050H　　MOV DL，‘25’

注意：1. 立即数只能作为源操作数使用，常用于给Reg赋初值

2. 源操作数和目的操作数的类型要互相匹配。

3.2 寻址方式

3.2.2、 寄存器寻址

操作数放在某个寄存器中,注意 源操作数与目的操作数字长要相同(匹配)，例如：

MOV AX, BX　　MOV [3F00H], AX

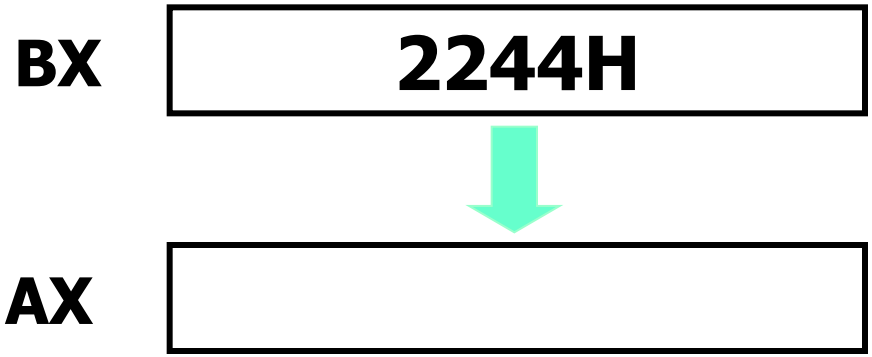
MOV CL, AL

错误例： **MOV AX, BL　　；字长不同**

对于指令：**MOV AX，BX　；AX ← BX**

指令执行前：BX=2244H　AX=2003H

指令执行后：BX=2244H　AX=2244H



说明：寄存器寻址的操作数在Reg中，无需访问总线，执行速度快。若选用AX，执行指令时间更短

3.2 寻址方式

3.2.3、存储器寻址方式

程序运行时用到的数据绝大部分存放在内存中，如何寻找内存中的数据就是存储器寻址方式。

- **操作数在存储器中**
- **操作数的有效地址EA由指令提供**
- **段地址在默认的或段超越前缀指定的段寄存器中**

存储器寻址方式有5种：

- **直接寻址**
- **寄存器间接寻址**
- **寄存器相对寻址**
- **基址变址寻址**
- **基址变址相对寻址**

3.2 寻址方式

1、直接寻址

直接寻址：在指令中直接给出操作数所在单元的偏移地址，具体有两种形式

(1) MOV　AX，[2050H]；AX←[DS×16＋2050H]

偏移地址通常以变量的形式出现，在指令中就直接写变量的名字

(2) MOV　SI , BUF　；　BUF为一变量

默认的段寄存器为DS，但也可以显式地指定其他段寄存器。

称为“段超越”前缀。例如：

MOV　DX，ES:[2050H]；使用段超越；DX←[ES×16＋2050H]

段超越指令格式

3.2 寻址方式

指令操作例：MOV AX，[2000H]

如果DS=3000H, 则操作数的物理地址为：

30000H+2000H = 32000H

区别MOV AX，2000H

(32000H) = 02H，(32001H) = 31H

指令执行后：AX = 3102H

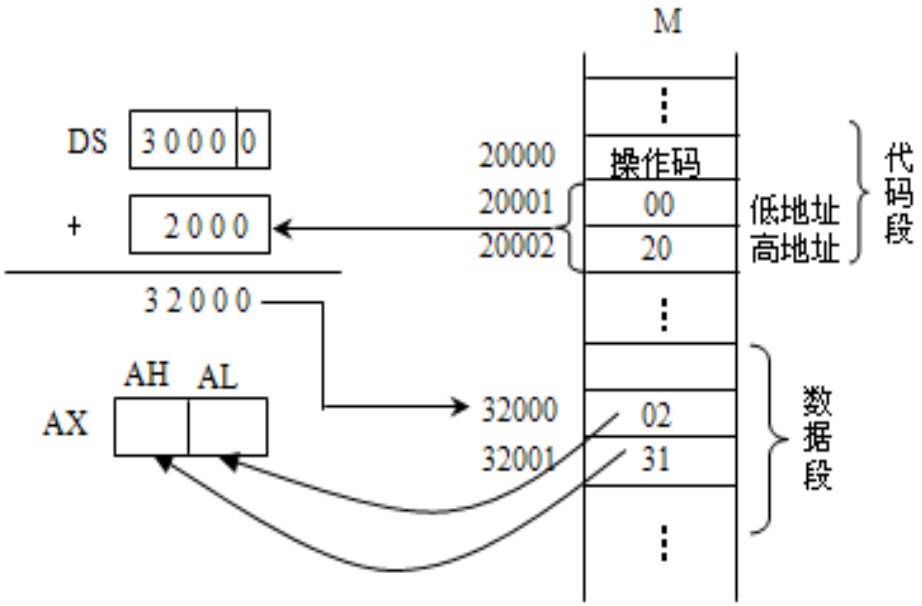


图 3.2 MOV AX,[2000H] 指令的寻址及执行过程

3.2 寻址方式

2、寄存器间接寻址

操作数的偏移地址放在寄存器中，只有SI、DI、BX和BP可作间址寄存器

$$EA = \left\{ \begin{array}{c} BX \\ BP \\ SI \\ DI \end{array} \right\}$$

```
MOV   AL, [BX]; AL←[DS×16+BX]
MOV   AX, [SI]; AX←[DS×16+SI]
MOV   EAX, [DI]; EAX←[DS×16+DI]
MOV   AX, [BP]; AX←[SS×16+BP]
MOV   AX, ES:[BX]; AX←[ES×16+BX]
MOV   EAX, DS:[BP]; EAX←[DS×16+BP]
```

字节操作

字 操 作

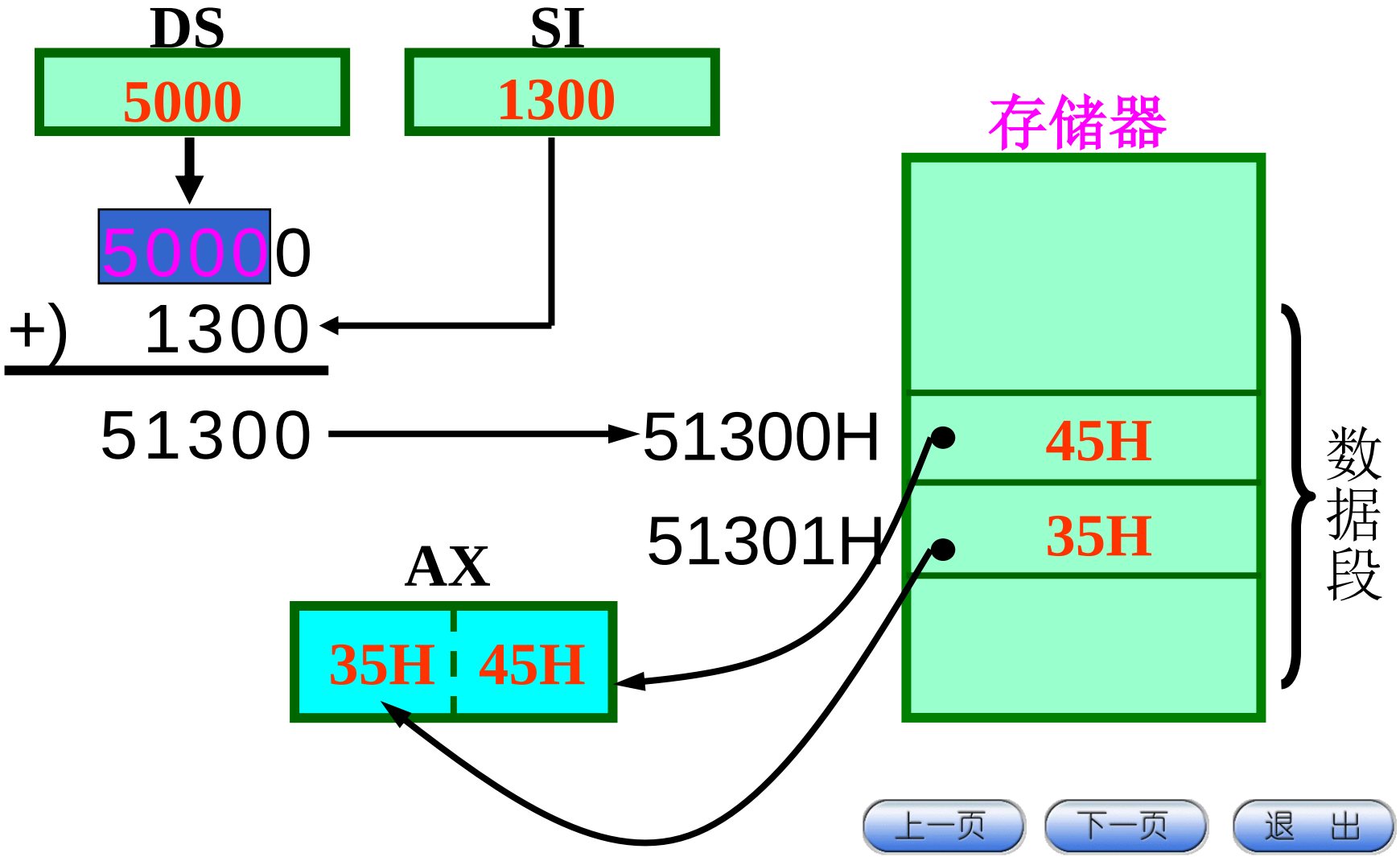
双字操作

段寄存器使用SS

段超越

3.2 寻址方式

指令操作例：若 DS=5000H，SI=1300H，(51300H)=45H，(51301H)=35H,则执行MOV AX，[SI]指令后，AX=3545H



3.2 寻址方式

3、寄存器相对寻址

EA=间址寄存器的内容加上一个8/16位的位移量

$$EA = \left\{ \begin{array}{c} BX \\ BP \\ SI \\ DI \end{array} \right\} + \left\{ \begin{array}{c} 8位 \\ 16位 \end{array} \right\} 位移量$$

寄存器相对寻址方式操作数的书写形式可以采用多种，以下是与MOV AX,BUF[BX]指令等价的另外几种书写形式：

```
MOV  AX,[BX+ BUF]
MOV  AX,[BUF + BX]
MOV  AX,[ BX] BUF
```

3.2 寻址方式

例：设DS=3000H，BX=1000H，BUF=4000H，物理地址（35000H）字单元中的内容为1990H，则在执行指令：

MOV AX, BUF [BX] 后，AX=1990H

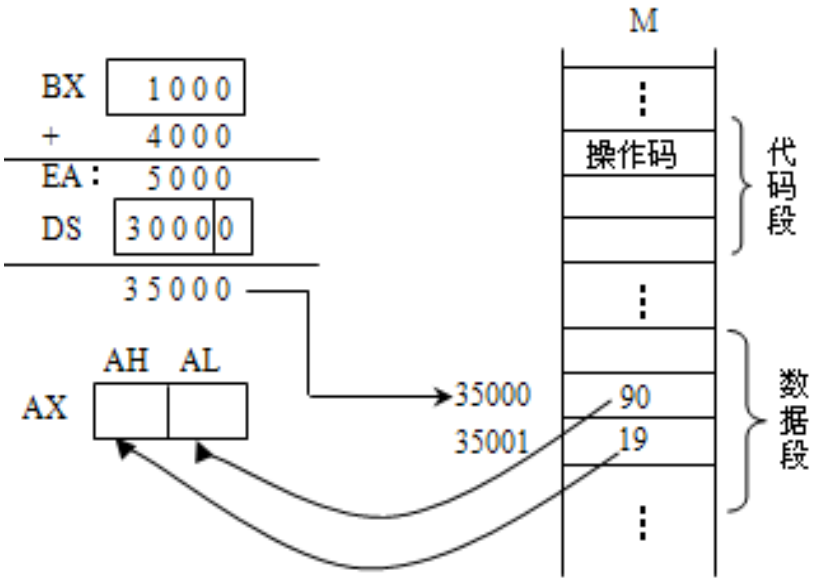


图 3.4 MOV AX, COUNT[BX] 指令的寻址及执行结果

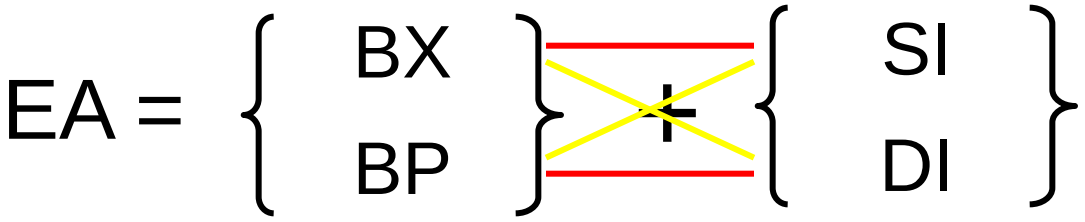
3.2 寻址方式

4、基址变址寻址

由一个基址寄存器的内容和一个变址寄存器的内容相加而形成操作数的偏移地址。

基址寄存器由(BX或BP)给出

变址寄存器由(SI或DI)给出



正确示例：

MOV AX, [BX] [SI]

MOV AX, [BX+SI]

MOV AX, DS: [BP] [DI]

错误示例：

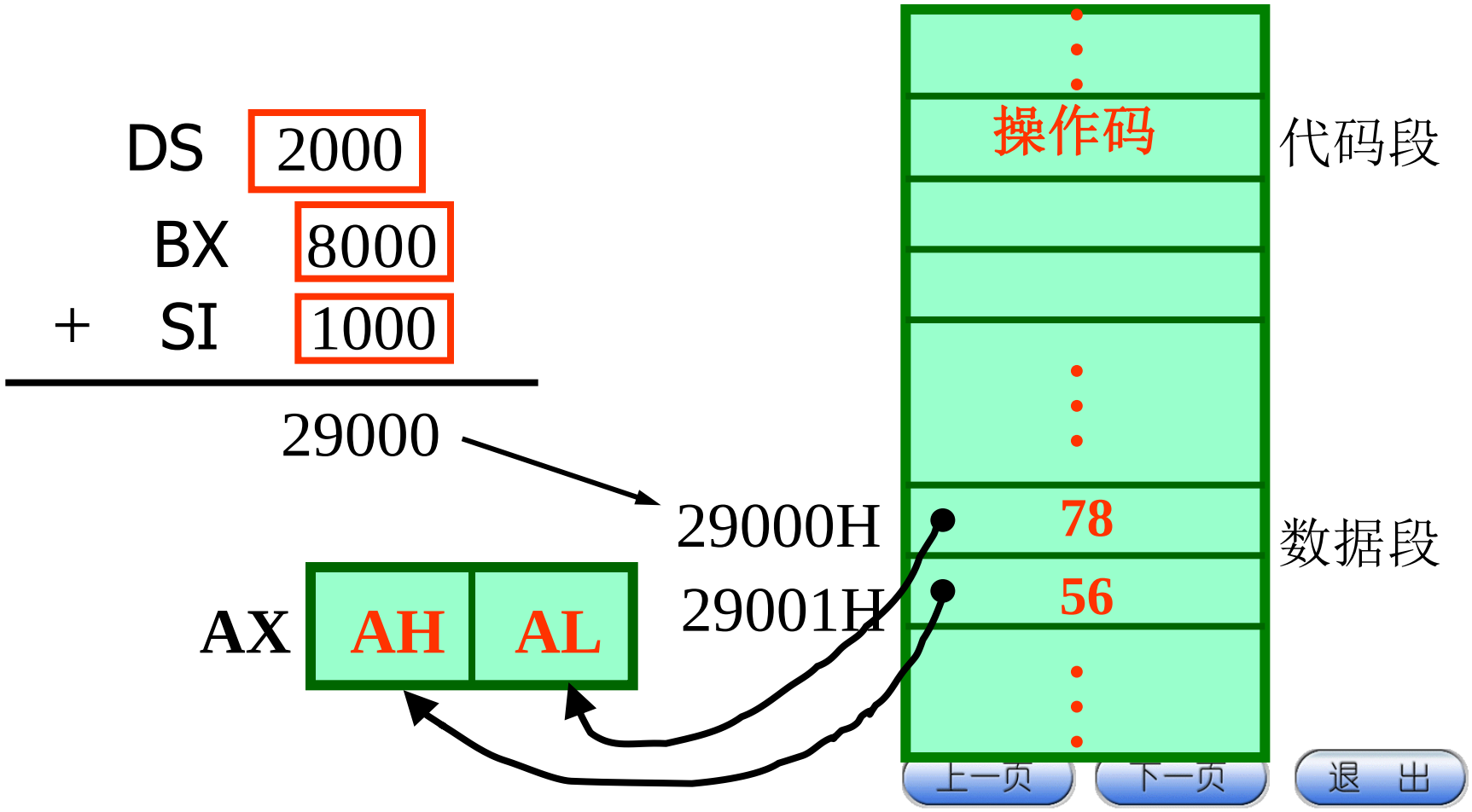
× **MOV AX, [BX] [BP]**

× **MOV AX, [DI] [SI]**

3.2 寻址方式

指令操作例：

设DS=2000H，BX=8000H，SI=1000H，物理地址（29000H）单元内容为5678H，则执行指令MOV AX，[BX][SI]后，AX=5678H



3.2 寻址方式

5、基址变址相对寻址

在基址＋变址寻址的基础上再增加一个8位或16位位移量，三者之和为位移地址

$$EA = \left\{ \begin{array}{l} BX \\ BP \end{array} \right\} + \left\{ \begin{array}{l} SI \\ DI \end{array} \right\} + \left\{ \begin{array}{l} 8位 \\ 16位 \end{array} \right\} \text{位移量}$$

与基址变址寻址方式相同，基址变址相对寻址方式的操作数也可采用多种等价形式，例：

MOV DX ,disp [BX][SI]

MOV DX ,disp [BX + SI]

MOV DX,[BX + SI + disp]

MOV DX,[BX + SI]disp

3.2 寻址方式

设DS=3000H，BX=2000H，SI=1000H，位移量disp=0250H，物理地址=DS×16＋BX＋SI＋disp=30000H＋2000H＋1000H＋0250H=33250H的字单元内容为3132H，则执行指令：

MOV DX,disp [BX][SI] 后，DX中的内容为3132H

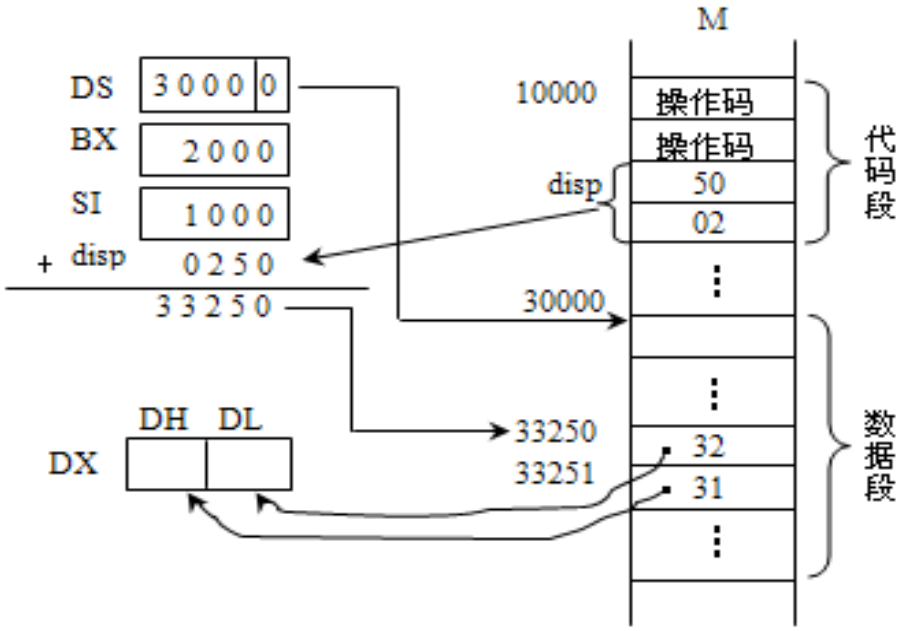


图 3.5 MOV DX,disp[BX][SI] 指令的寻址及执行过程示意图

小结

指令由（操作码）和 （操作数）构成

如何取得操作数——称为（寻址）

说明：

① 在直接寻址中，地址可以用符号代替，如

“MOV AX，BUF”，相当于“MOV AX，[xxxxH]”

“BUF”称为变量或符号地址，后面介绍。

②偏移量可以是变量或常量，也在后面介绍。

③如果一条指令有两个操作数，源操作数和目的操作数的寻址方式可能相同，也可能不同，如指令

“MOV AX，[BX]”，源操作数为寄存器间接寻址，目的操作数为寄存器寻址。

3.3 8086 CPU指令系统

汇编语言源程序基本结构

```
DATA SEGMENT
STR DB  'PROGRAM EXAMPLE$'
DATA ENDS
CODE SEGMENT
     ASSUME DS:DATA,CS:CODE
START:  MOV AX,DATA
        MOV DS,AX
        MOV AH,9
        LEA DX,STR
        INT 21H
        MOV AH,4CH
        INT 21H
CODE ENDS
     END START
```

```
DATA SEGMENT
        ¦
        ¦
        ¦
DATA ENDS
CODE SEGMENT
     ASSUME DS:DATA,CS:CODE
START:  MOV AX,DATA
        MOV DS,AX
              ¦
              ¦
              ¦
        MOV AH,4CH
        INT 21H
CODE ENDS
     END START
```

3.3 8086 CPU指令系统

4、指令系统分类

指令系统是CPU指令的集合，Intel 8086指令系统共有117条基本指令，按功能分为六大类：

（1）数据传送类；
（2）算术运算类；
（3）逻辑运算和移位（位操作类）；
（4）控制转移类；
（5）串操作；
（6）处理器控制

3.3 8086 CPU指令系统

3.3.1、数据传送类指令

（1）通用数据传送指令MOV
（2）数据交换指令XCHG
（3）地址传送指令LEA/LDS/LES
（4）堆栈操作指令PUSH / POP
（5）标志寄存器传送指令LAHF/SAHF、 PUSHF/POPF
（6）换码指令XLAT

- 数据传送是计算机中最基本、最重要的一种操作,数据传送指令负责把数据、地址或立即数传送到寄存器或存储单元中。是最常用的一类指令。
- 除标志寄存器传送指令外，均不影响标志位
- 重点掌握

  MOV XCHG XLAT PUSH POP LEA

3.3 8086 CPU指令系统

1、通用数据传送指令－MOV

指令格式：MOV dest,src

功能：将源操作数src的内容传送给目的操作数dest，源操作数内容不变

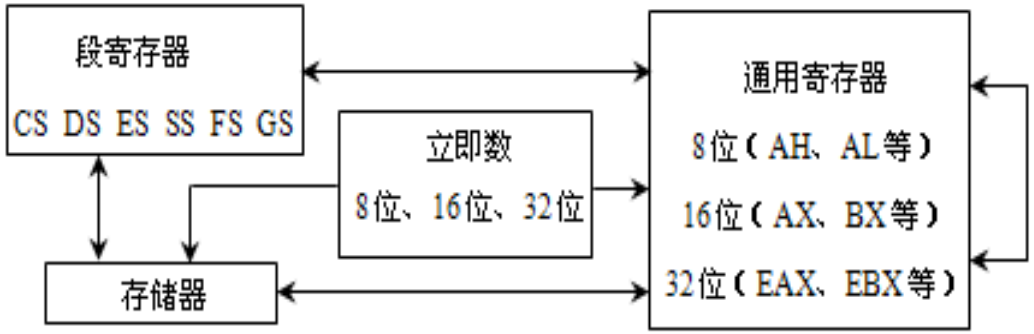


图 3.13 数据传送方向示意图

MOV指令不影响标志位

3.3 8086 CPU指令系统

MOV指令的补充说明：

图中描述的是MOV指令能实现的大多数的数据传送方式，但也存在一些特殊情况。

特殊规定：

（1）当是双操作数时，两个操作数的位数需一致；

（2）两个操作数不能同时为段寄存器和存储器操作数。

（3）CS不能作为目的操作数使用，但可作为源操作数。

（4）立即数不能作为目的操作数，也不能直接传送给段寄存器。

（5）指令指针寄存器IP不能作为操作数使用。

2、数据交换指令—XCHG

格式：XCHG reg,mem/reg

功能：实现源操作数和目的操作数之间的内容相互交换

例：XCHG AL,BL ； XCHG AX,DX ； XCHG CX,[3450H]

以下3条指令实现寄存器与存储器之间的数据交换。

```
MOV  AX，5678H          ； AX=5678H
MOV  BX，0FFFFH         ； BX=0FFFFH
XCHG  AX，BX            ; AX=0FFFFH ， BX=5678H
```

XCHG指令不影响标志位

注意：交换只能在通用寄存器之间，通用寄存器与存储器之间

3、堆栈操作指令—PUSH/POP

格式：

```
PUSH r16/m16/seg
; SP←SP－2
; SS:[SP]←r16/m16/seg
```

```
POP r16/m16/seg
; r16/m16/seg←SS:[SP]
; SP←SP＋2
```

3.3 8086 CPU指令系统

示例：若SP=00F8H，SS=4000H，AX=5102H，执行指令PUSH AX后，SP=00F6H，（400F6H）=5102H。

图示为指令执行前后堆栈变化示意图

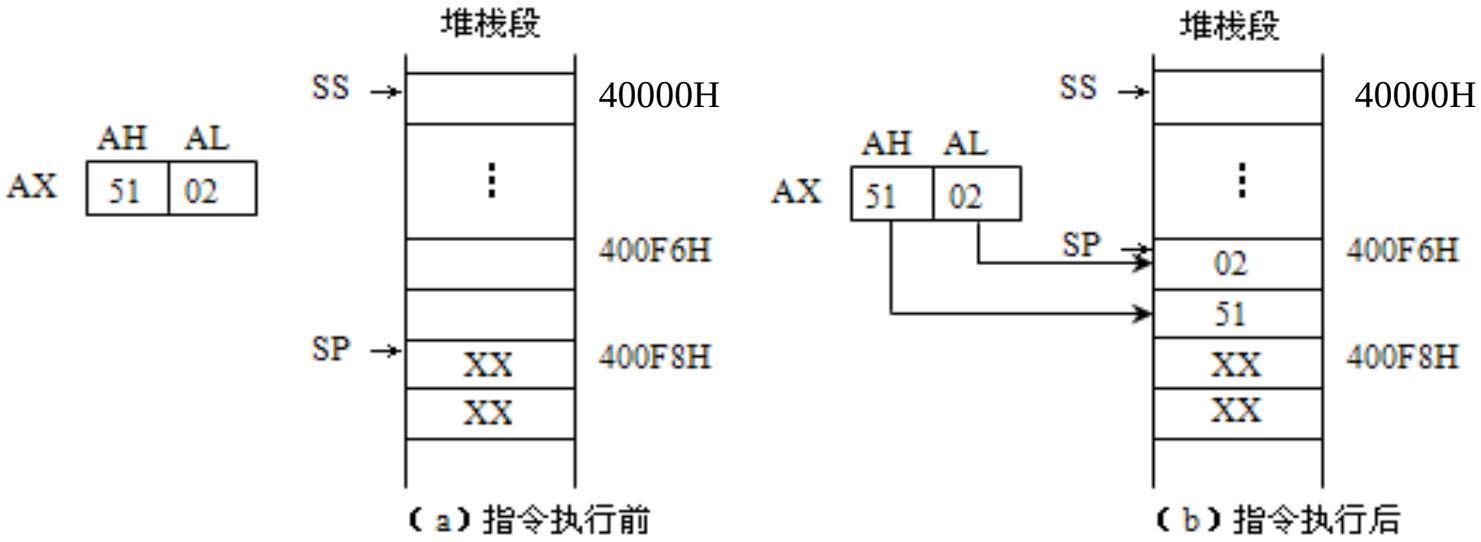
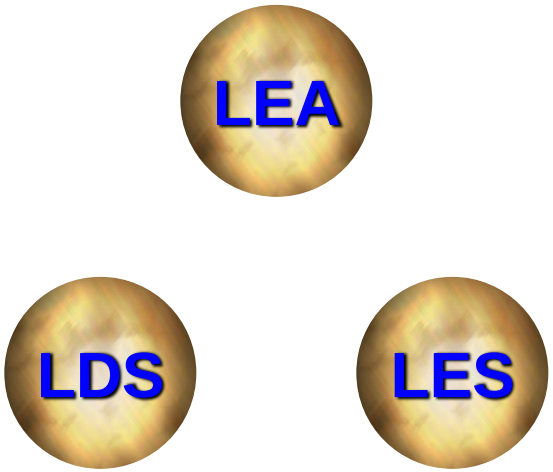


图 3.7 PUSH AX 指令执行前后的堆栈变化

4、地址传送指令

特点：传送存储器操作数的地址（偏移地址、段地址）；
对标志寄存器的各状态标志位均无影响；
源操作数必须是存储器操作数，目的操作数可以是任何一个16 位的通用寄存器



3.3 8086 CPU指令系统

（1）有效地址传送指令LEA

格式　　LEA r16,mem　　；r16←mem的有效地址EA

功能：将源操作数的偏移地址传送给指定的16位寄存器

例：设BX=0400H，SI=003CH

```
LEA BP，[2050h]　　；BP=2050H
LEA BP，[SI]　　；BP=003CH
LEA BX，[BX+SI+0F62H]　　；BX=139EH
```

说明：①“r16”通常为寄存器BX、BP、SI、DI。

②源操作数mem为存储器操作数，常用变量名，而不是上面例子中举的形式，如：LEA BX，BUFFER。

3.3 8086 CPU指令系统

示例：设BX=1000H，DS=6000H， [61032H]=33H，[61033H]=44H，试比较以下两条指令的执行结果。

```
LEA  BX,50[BX]
MOV  BX,50[BX]
```

分析：执行第一条指令后，BX=1032H；
执行第二条指令后，BX=4433H

- 获得内存单元的有效地址（偏移地址）
- 可以实现计算功能

注意：LEA指令和MOV指令间的区别

5、标志寄存器传送指令

标志寄存器传送指令用来传送标志寄存器FLAGS的内容，方便进行对各个标志位的直接操作。

有2对4条指令

低8位传送：LAHF和SAHF

16位传送：PUSHF和POPF

6、换码指令XLAT

格式：XLAT或XLAT 表首址

功能：将AL与BX寄存器的内容之和作为偏移地址，将偏移地址所对应的存储单元内容送入AL寄存器。

换码指令执行前：

- 在内存建立一个字节数据表，内含要转换成的目标代码
- BX中存放数据表首地址，AL存放相对表格首地址的位移量

换码指令执行后：

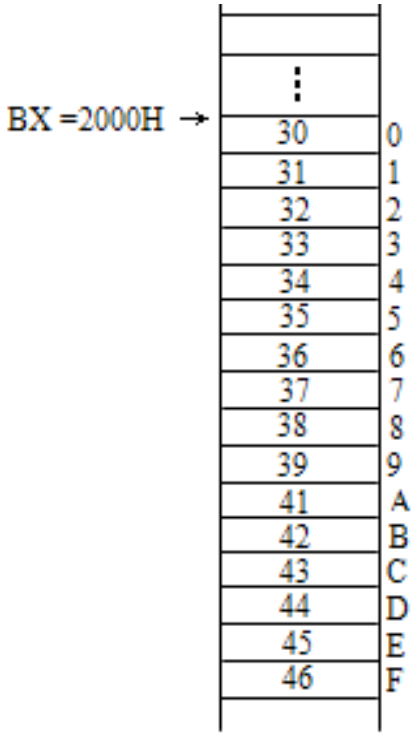
将AL寄存器的内容转换为目标代码

示例：编程实现将十六进制数0～FH间的某一十六进制数据转换为其字符对应的ASCII码值，假设存放ASCII码值的数据表首地址为2000H。

以下是要取出“B”字符所对应的ASCII码值的程序段：

```
MOV  BX,2000H
MOV  AL,0BH
XLAT
```

完成代码转换后，AL=42H



所定义的 0～FH 间的 16 进制数对应的 ASCII 码值

3.3.2、算术运算类指令

算术运算是计算机经常进行的一种操作。算术运算指令实现二进制（和十进制）数据的四则运算

注意：

算术运算类指令对标志位的影响

掌握：ADD/ADC/INC、SUB/SBB/DEC/NEG/CMP

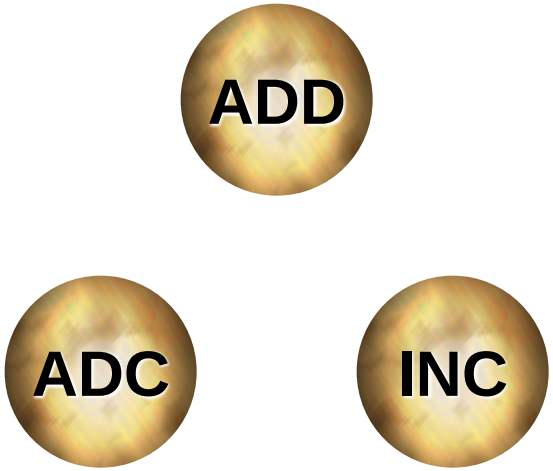
熟悉：MUL/IMUL、DIV/IDIV

理解：CBW/CWD

- 加法指令
- 减法指令
- 乘法指令
- 除法指令
- 符号扩展指令

3.3 8086 CPU指令系统

1、加法指令



1）加法指令ADD

格式:

```
ADD reg,imm/reg/mem
                ; reg←reg＋imm/reg/mem

ADD mem,imm/reg
                ; mem←mem＋imm/reg
```

功能：将源操作数的内容与目的操作数的内容相加，相加后的结果保存在目的操作数中，并根据结果置相关标志位 。

注意：源操作数可以是通用寄存器、存储器操作数或立即数，目的操作数只能为与源操作数类型相匹配的通用寄存器或存储器操作数，两者不能同时为存储器操作数

3.3 8086 CPU指令系统

例：加法运算

```
ADD  AL ，  30          ；累加器与立即数相加
ADD  BX ，  [3000H]     ；通用寄存器与存储器操作数内容相加
ADD  DI ，  CX          ；通用寄存器之间相加
ADD  BETA[SI] ，  DX    ；存储器操作数与寄存器相加
```

这些指令对状态标志位 都有影响。

例：设ADD AL, BL指令执行前，AL=66H，BL=20H，执行指令之后，AL=86H，BL仍为20H。

标志位影响情况为：

CF=0，ZF=0，SF=1，AF=0，OF=1，PF=0

2）带进位加法指令ADC

格式:

```
ADC reg,imm/reg/mem
          ; reg←reg＋imm/reg/mem＋CF

ADC mem,imm/reg
          ; mem←mem＋imm/reg＋CF
```

功能：ADC指令将源操作数内容、目的操作数内容及进位标志位CF的值三项相加，相加结果送给目的操作数

ADC指令按状态标志的定义相应设置

用法：ADC指令主要与ADD配合，实现多精度数加法运算

3）增1指令INC

格式:

```
INC reg/mem
; reg/mem←reg/mem＋1
```

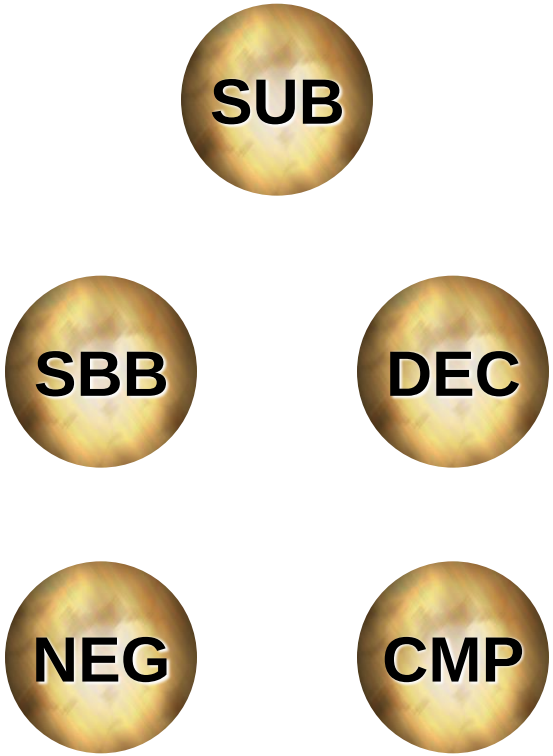
```
INC BX
INC BYTE PTR [BX]
```

功能：对指令中指定操作数的内容加1后，又回送给该操作数

INC指令不影响进位标志CF，按定义设置其他状态标志

用法：主要用于在循环程序中修改地址指针和循环次数等

2、减法指令



1）减法指令SUB

格式:

```
SUB reg,imm/reg/mem
                ; reg←reg -imm/reg/mem

SUB mem,imm/reg
                ; mem←mem -imm/reg
```

功能：将目的操作数内容与源操作数内容相减之后的结果（差）存入目的操作数中，对操作数的使用及标志位的影响同ADD指令

2）带借位减法指令SBB

格式：

```
SBB reg,imm/reg/mem
          ; reg←reg－imm/reg/mem－CF

SBB mem,imm/reg
          ; mem←mem－imm/reg－CF
```

功能：用目的操作数内容减去源操作数内容，还要减去标志位CF的值，并将相减的结果回送给目的操作数。与ADC指令类似，在实际编程中，SBB指令主要用于两个多字节/字二进制数的相减运算。

用法：SBB指令主要与SUB配合，实现多精度减法运算

3、减1指令DEC

格式:

```
DEC reg/mem
; reg/mem←reg/mem -1
```

```
DEC CX
DEC WORD PTR [SI]
```

功能：完成对指令中指定操作数的内容减1后，又回送给该操作数。与INC指令一样，DEC指令通常也用于在循环过程中对地址指针和循环次数的修改

- INC指令和DEC指令都是单操作数指令
- 主要用于对计数器和地址指针的调整

4）求补指令NEG

格式：

```
NEG reg/mem
; reg/mem←0 - reg/mem
```

功能：对指令中指定的操作数内容取补后，再将结果回送给该操作数。

对一个操作数的求补实际上就相当于用0减去该操作数的内容，故NEG指令也属于减法指令。该指令的间接求法是将操作数的内容按位求反末位加1后，再回送给该操作数

0-XXXX XXXXB =(1111 1111B + 1)-XXXX XXXXB

求补运算也可以表达成：将操作数按位取反后加1

NEG指令对标志的影响与用零作减法的SUB指令一样

5）比较指令CMP

格式:

```
CMP reg,imm/reg/mem
                    ; reg -imm/reg/mem
CMP mem,imm/reg
                    ; mem -imm/reg
```

功能：将目的操作数与源操作数内容相减，相减结果不回送目的操作数，仅根据结果影响标志位，对标志位的影响与SUB相同

设指令CMP AL,CL执行之前， AL=68H，CL=9AH，该指令执行之后，AL=68H，CL=9AH，CF=1，ZF=0，SF=1，AF=1，OF=1，PF=0

3.3 8086 CPU指令系统

3、乘法指令

```
MUL r8/m8
; 无符号字节乘法
; AX←AL×r8/m8
MUL r16/m16
; 无符号字乘法
; DX.AX←AX×r16/m16
```

```
IMUL r8/m8
; 有符号字节乘法
; AX←AL×r8/m8
IMUL r16/m16
; 有符号字乘法
; DX.AX←AX×r16/m16
```

乘法指令分无符号和有符号乘法指令

乘法指令的源操作数显式给出，隐含使用的另一个操作数为**AX**和**DX**

字节数据相乘：**AL**与**r8/m8**相乘，得到的**16**位结果存入**AX**

字数据相乘：**AX**与**r16/m16**相乘，得到的**32**位结果，其高字存入**DX**，低字存入**AX**

3.3 8086 CPU指令系统

乘法指令对标志位的影响：利用OF和CF判断乘积的高一半是否具有有效数值

- ✓ MUL指令——若乘积的高一半（AH或DX）为0，则OF=CF=0；否则OF=CF=1
- ✓ IMUL指令——若乘积的高一半是低一半的符号扩展，则OF=CF=0；否则OF=CF=1

乘法指令对其他状态标志没有定义

- 对标志位没有定义：指令执行后这些标志位是任意的、不可预测（就是不知道是0还是1）
- 对标志位没有影响：指令执行不改变标志位状态

4、除法指令

DIV r8/m8 ；无符号字节除法：
AL←AX÷r8/m8的商，AH←AX÷r8/m8的余数
DIV r16/m16 ；无符号字除法：
；AX←DX.AX÷r16/m16的商，DX←DX.AX÷r16/m16的余数

IDIV r8/m8 ；有符号字节除法：
AL←AX÷r8/m8的商，AH←AX÷r8/m8的余数
IDIV r16/m16 ；有符号字除法：
；AX←DX.AX÷r16/m16的商，DX←DX.AX÷r16/m16的余数

5、符号扩展指令

不影响标志位

```
CBW   ；AL的符号扩展至AH
；若AL的最高位是0，则AH＝00
；AL的最高位为1，则AH＝FFH，AL不变
```

```
CWD   ；AX的符号扩展至DX
；若AX的最高位是0，则DX＝00
；AX的最高位为1，则DX＝FFFFH，AX不变
```

3.3.3逻辑运算与移位类（位操作）

位操作类指令以二进制位为基本单位进行数据的操作；包含逻辑运算类指令和移位指令。是一类常用的指令，应该特别掌握

注意这些指令对标志位的影响

要求：全面而准确地理解每条指令的功能和应用

一、逻辑运算指令

AND/OR/XOR/NOT/TEST

二、移位指令

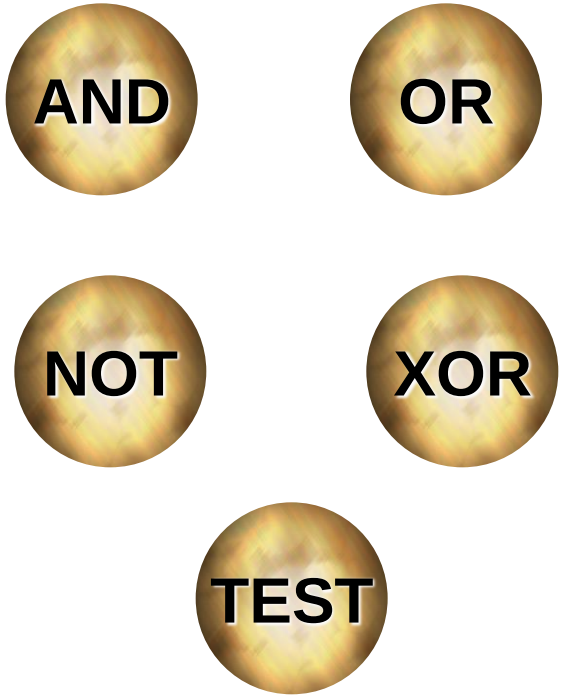
SHL/SHR/SAL/SAR

三、循环移位指令

ROL/ROR/RCL/RCR

3.3 8086 CPU指令系统

1、逻辑运算指令



3.3 8086 CPU指令系统

1）逻辑与指令AND

AND reg,imm/reg/mem ； reg←reg∧imm/reg/mem

AND mem,imm/reg ； mem←mem∧imm/reg

功能：对两个操作数执行按位的逻辑与运算，结果送到目的操作数

说明：（1）按位的逻辑与运算；

（2）两个操作数不能同时为存储器操作数；寄存器只能是通用寄存器；

（3）注意对标志位的影响；

思考：

（1）某一个操作数自己和自己相逻辑与，结果是？

（2）AND指令主要用在什么场合？ 屏蔽某些位

2）逻辑或指令OR

OR reg,imm/reg/mem ； reg←reg∨imm/reg/mem

OR mem,imm/reg ； mem←mem∨imm/reg

功能：对两个操作数执行按位逻辑或运算，结果送到目的操作数

说明:(1)按位逻辑或运算；

（2）OR指令对操作数的限制和对标志位的影响；

思考：

（1）某一个操作数自己和自己相逻辑或，结果是？

（2）OR指令主要用在什么场合？ 置某些位

3）逻辑异或指令XOR

```
XOR reg,imm/reg/mem ; reg←reg⊕imm/reg/mem
XOR mem,imm/reg ; mem←mem⊕imm/reg
```

功能：对两个操作数执行按位逻辑异或运算，结果送到目的操作数

说明：

（1）只有相“异或”的两位不相同，结果才是1；

（2） XOR指令对操作数的限制和对标志位的影响同AND指令

思考：

（1）某一个操作数自己和自己相异或，结果是？ 寄存器清 0

（2）XOR指令主要用在什么场合？ 求反某些位

4）逻辑非指令NOT

NOT reg/mem ； reg/mem←~reg/mem

功能：对一个操作数执行按位逻辑非运算

说明：

（1）按位取反，原来是“0”的位变为“1”，原来是“1”的位变为“0”；

（2）NOT指令是一个单操作数指令；

（3）NOT指令不影响标志位；

5）测试指令TEST

```
TEST reg,imm/reg/mem ; reg←reg∧imm/reg/mem
TEST mem,imm/reg ; mem←mem∧imm/reg
```

功能：对两个操作数执行逻辑与运算，结果不回送到目的操作数，但要影响标志位，标志位影响同AND指令。

说明：

（1）本条指令通常是用于检测一些条件是否满足，但又不希望改变原有的操作数的情况下。

（2）本条指令通常在其后紧跟一条条件转移指令。

2、移位指令

1）非循环移位指令

SHL　　SHR

SAL　　SAR

- ✓ 其中两条左移指令对应同一条机器指令，所以，从机器指令的角度来说，移位指令只有三条。
- ✓ 四条指令分成逻辑移位和算术移位，分别具有左移或右移操作，将操作数移动1位或多位。

3.3 8086 CPU指令系统

格式：

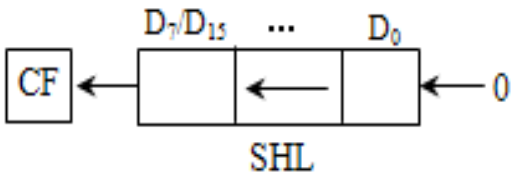
SHL reg/mem,1/CL
；逻辑左移，最高位进入CF，最低位补0

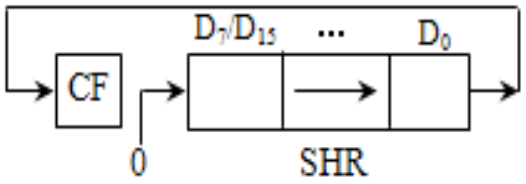
SHR reg/mem,1/CL
；逻辑右移，最低位进入CF，最高位补0

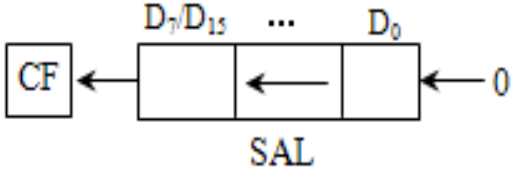
SAL与SHL相同

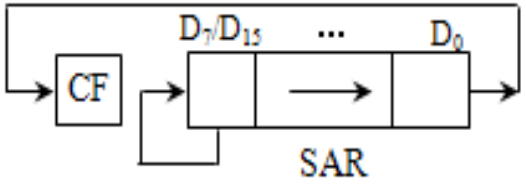
SAL reg/mem,1/CL
；算术左移，最高位进入CF，最低位补0

SAR reg/mem,1/CL
；算术右移，最低位进入CF，但符号位保持不变



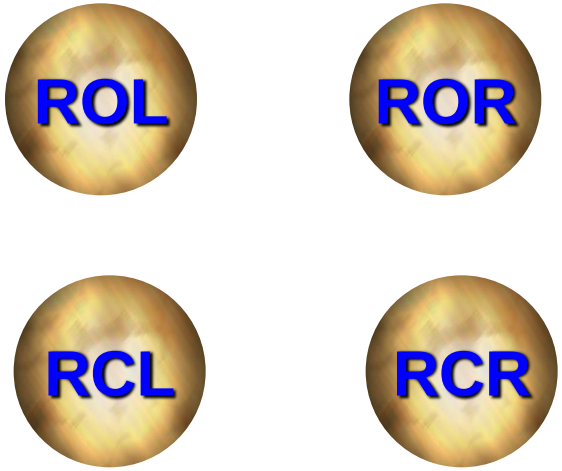






退 出

2）循环移位指令

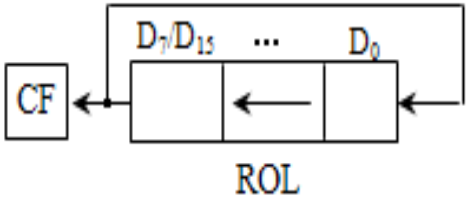


将操作数从一端移出的位返回到另一端形成循环，分成不带进位和带进位，分别具有左移或右移操作

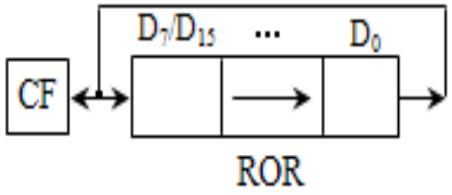
3.3 8086 CPU指令系统

循环移位指令格式

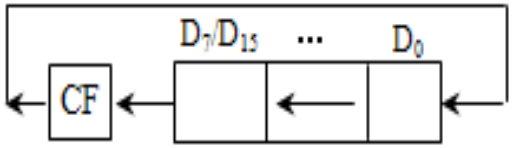
```
ROL reg/mem,1/CL   ；不带进位循环左移
ROR reg/mem,1/CL   ；不带进位循环右移
RCL reg/mem,1/CL   ；带进位循环左移
RCR reg/mem,1/CL   ；带进位循环右移
```



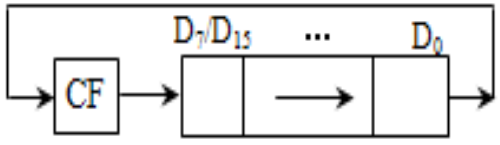
ROL



ROR



RCL



RCR

3.3 8086 CPU指令系统

3.3.4控制转移指令

- 无条件、条件转移指令
- 循环指令
- 子程序调用及返回指令
- 中断指令
- 系统功能调用

重点：Jmp、Jcc、Loop、 CALL/RET 、Int指令的理解
难点：Jmp、Jcc、Loop、Int指令的应用

控制转移类指令通过改变IP（和CS）值，实现程序执行顺序的改变，用于实现分支、循环、子程序等程序结构，是仅次于传送指令的常用指令，需很好掌握

3.3 8086 CPU指令系统

1、无条件转移指令

格式：JMP label

功能：程序转向label标号指定的地址处执行

说明：

JMP指令分成4种类型：

(1) 段内直接转移

(2) 段内间接转移

(3) 段间直接转移

(4) 段间间接转移

3.3 8086 CPU指令系统

举例

JMP label 例如：

jmp output ; IP←label的偏移地址

段内直接转移

.....

output:mov result,al

JMP r16/m16 例如：

jmp ax

jmp word ptr [2000h]

段内间接转移

JMP far ptr label 例如：

; IP←label的偏移地址
; CS←label的段地址

jmp far ptr otherseg

段间直接转移

JMP far ptr mem 例如：

段间间接转移

JMP far ptr [bx]

2、条件转移指令

格式：

```
Jcc label
```

功能：若指定的条件cc如果成立，程序转移到由标号label指定的目标地址去执行；若条件不成立，顺序执行下一条指令

说明：（1） Jcc？

（2）操作数label是采用段内短转移

（3） Jcc指令不影响标志，但要利用标志

Jcc指令的分类

根据利用的标志位不同，16条指令分成3种情况：

（1） 判断单个标志位状态

（2） 比较无符号数高低

（3） 比较有符号数大小

3.3 8086 CPU指令系统

表3-4 条件转移类指令

类型	指令书写格式	转移测试条件	功能描述
单个标志位	JC opr	CF=1	有进(借)位转移
	JNC opr	CF=0	无进(借)位转移
	JP/JPE opr	PF=1	奇偶性为1(偶)状态转移
	JNP/JPO opr	PF=0	奇偶性为0(奇)状态转移
	JZ/JE opr	ZF=1	结果为0/相等转移
	JNZ/JNE opr	ZF=0	结果不为0/不相等转移
	JS opr	SF=1	符号位为1转移
	JNS opr	SF=0	符号位为0转移
	JO opr	OF=1	溢出转移
	JNO opr	OF=0	无溢出转移
无符号数	JA/JNBE opr	CF=0且ZF=0	高于/不低于也不等于转移
	JNA/JBE opr	CF=1或ZF=1	不高于/低于或等于转移
	JB/JNAE opr	CF=1且ZF=0	低于/不高于也不等于转移
	JNB/JAE opr	CF=0或ZF=1	不低于/高于或等于转移
有符号数	JG/JNLE opr	SF=OF且ZF = 0	大于/不小于也不等于转移
	JNG/JLE opr	SF≠OF或ZF=1	不大于/小于或等于转移
	JL/JNGE opr	SF≠OF且ZF=0	小于/不大于也不等于转移
	JNL/JGE opr	SF=OF或ZF=1	不小于/大于或等于转移

3、循环指令

```
LOOP label      ；CX←CX－1，
    ；CX≠0，循环到标号label

LOOPZ label     ；CX←CX－1，
    ；CX≠0且ZF＝1，循环到标号label

LOOPNZ label    ；CX←CX－1，
    ；CX≠0且ZF＝0，循环到标号label

JCXZ label      ；CX＝0，转移到标号label
```

循环指令默认利用CX计数器，方便实现计数循环的程序结构

3.3 8086 CPU指令系统

示例：LOOP指令应用例子。编写一程序段，实现求1+2+…+100之和，并把结果存入AX中。

```
        XOR  AX,AX
        MOV  CX,100
AGAIN:  ADC  AX,CX
        LOOP AGAIN
```

方法1：利用循环计数器CX进行累加

```
        XOR  AX,AX
        MOV  CX,100
        MOV  BX, 1
AGAIN:  ADC  AX, BX
        INC  BX
        LOOP AGAIN
```

方法2：不用循环计数器CX进行累加

4、子程序指令

子程序是完成特定功能的一段程序

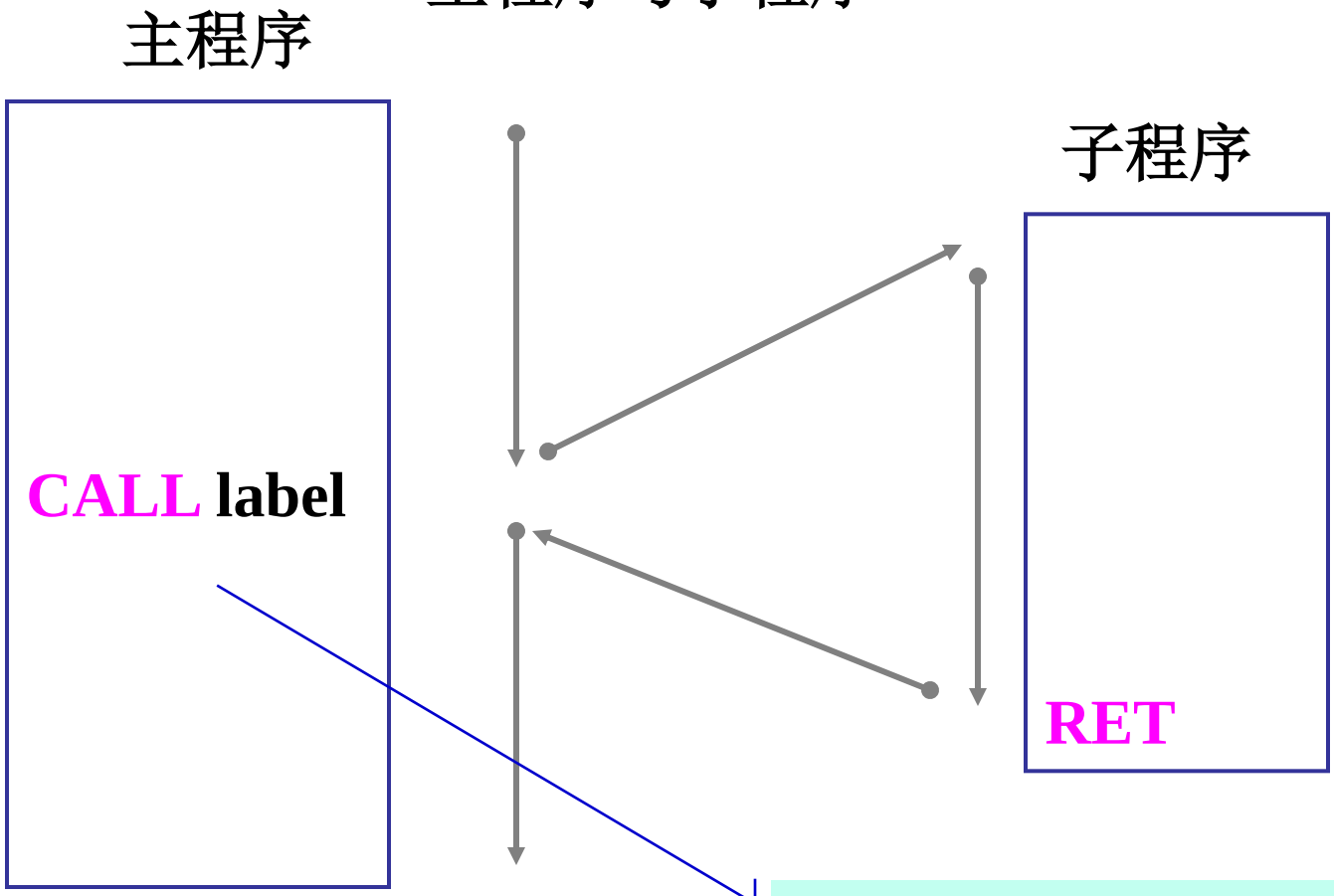
当主程序（调用程序）需要执行这个功能时，采用CALL调用指令转移到该子程序的起始处执行

当运行完子程序功能后，采用RET返回指令回到主程序继续执行

CALL指令和RET指令都不影响状态标志位

3.3 8086 CPU指令系统

主程序与子程序



5、中断指令

中断是又一种改变程序执行顺序的方法。

1、中断指令INT

格式： INT n

功能：产生一个中断类型号为n的内部中断，其n值是一个0～FFH范围内的整数

INT n指令的具体执行步骤和操作 在8.2.3中介绍

2、溢出中断指令INTO

格式：INTO

功能：产生一个溢出中断

3、中断返回指令 IRET

功能：退出中断过程，返回到中断时的断点处继续执行。

所有的中断程序，不管是由软件引起还是由硬件引起，其最后一条指令必须是IRET

3.3.5串操作指令

串操作指令是8086指令系统中比较独特的一类指令，采用比较特殊的数据串寻址方式，是可以直接实现存储区之间操作的指令。在操作内存连续区域的数据时，特别好用。

串操作指令的操作对象（操作数）是内存中连续存放的数据串——即在连续的内存区域中以字（W）为单位的字串，或是以字节（B）为单位的字节串

为什么要用：数据传送类指令每次只能传送一个数据，若要传送大批数据就需要重复编程（通过循环来实现），这样就浪费了大量的时间和空间。为此8086提供了一组处理内存中连续存放数据的串指令，这就是串操作指令。

3.3.6处理器控制类指令

1、标志位操作

这组指令共有7 条，均为无操作数指令，它们的操作数隐含在标志寄存器的某些标志位上，即能直接对指定的标志位进行操作，但不影响别的标志位

表 3-6 CLC 等 7 条标志位操作指令

指令	功能
清除进位指令 CLC	该指令使进位标志位 CF=0
置 1 进位位指令 STC	该指令使 CF=1
取反进位位指令 CMC	该指令使 CF 的值取反
清除方向标志位指令 CLD	该指令使方向标志 DF=0
置 1 方向标志位指令 STD	该指令使 DF=1
清除中断标志位指令 CLI	该指令使中断标志位 IF=0
置 1 中断标志位指令 STI	该指令使 IF=1

2、空操作指令NOP

- ✓ NOP指令不执行任何操作，但却要消耗3个时钟周期的时间
- ✓ NOP常用于程序调试

  在需要预留指令空间时用NOP填充

  代码空间多余时也可以用NOP填充

  还可以用NOP实现软件延时，相当于“XCHG AX, AX”指令

作业要求

- https://cslabcg.whu.edu.cn/admin/index.jsp
- 默认密码为学号
- 在设定时间内提交作业
- 如果需要延期，请私信学号和姓名，并说明迟交原因

- **批阅后补交的作业只计提交次数，内容不再批阅**