第四章指令系统和汇编程序设计

8086 CPU的七种寻址方式

8086 CPU寻址方式

•8086中，CS、DS、ES和SS段寄存器在程序运行过程中分别指向当前的代码段、数据段、附加段和堆栈段。而操作数可能存放在代码段中，也可能存放在数据段、附加段、堆栈段中，还可能存放在8086CPU内部的寄存器中。**存放操作数的内存单元相对于其所在段的段起始地址偏移量称为偏移地址或有效地址EA（Effective Address）。获得操作数所在地址的方法称为寻址方式。**在8086系统中，一般将寻址方式分为两类：一类是寻找操作数的地址；另一类是寻找要执行的下一条指令的地址，即程序寻址。
•MOV DST, SRC
助记符 目的操作数 源操作数

一、立即寻址方式（Immediate addressing）

•就是直接赋值

TIPS：五个数字的是PA，提到PA就是物理地址。00000~FFFFF，提到PA就是memory。

在立即寻址（Immediate Addressing）方式下，操作数直接包含在指令中，它是一个8位或16位的常数。这类指令翻译成机器代码时，立即数作为指令的一部分紧跟在操作码之后，存放在代码段中。如果是16位数，则高8位存放在代码段的高地址单元中，低8位放在低地址单元中。**立即寻址方式常用于给寄存器赋初值，并且只能用于源操作数字段，不能用于目的操作数字段。**
•例如：MOV AX, 3064H
•指令执行后，（AX）= 3064H

二、直接寻址方式（Direct addressing）

判断：只要有[]。memory，就是直接寻址。

•在这种寻址方式中，操作数存放在存储单元中，而这个存储单元的有效地址就在指令的操作码之后，操作数的物理地址可通过（(DS)×16）再加上这个有效地址形成，如下图所示。

•在汇编语言指令中，可以用符号地址（变量名或标号）代替数值地址。直接寻址方式默认操作数在数据段中，如果操作数定义在其它段中，则应在指令中指定段跨越前缀。直接寻址方式适合于处理单个变量。

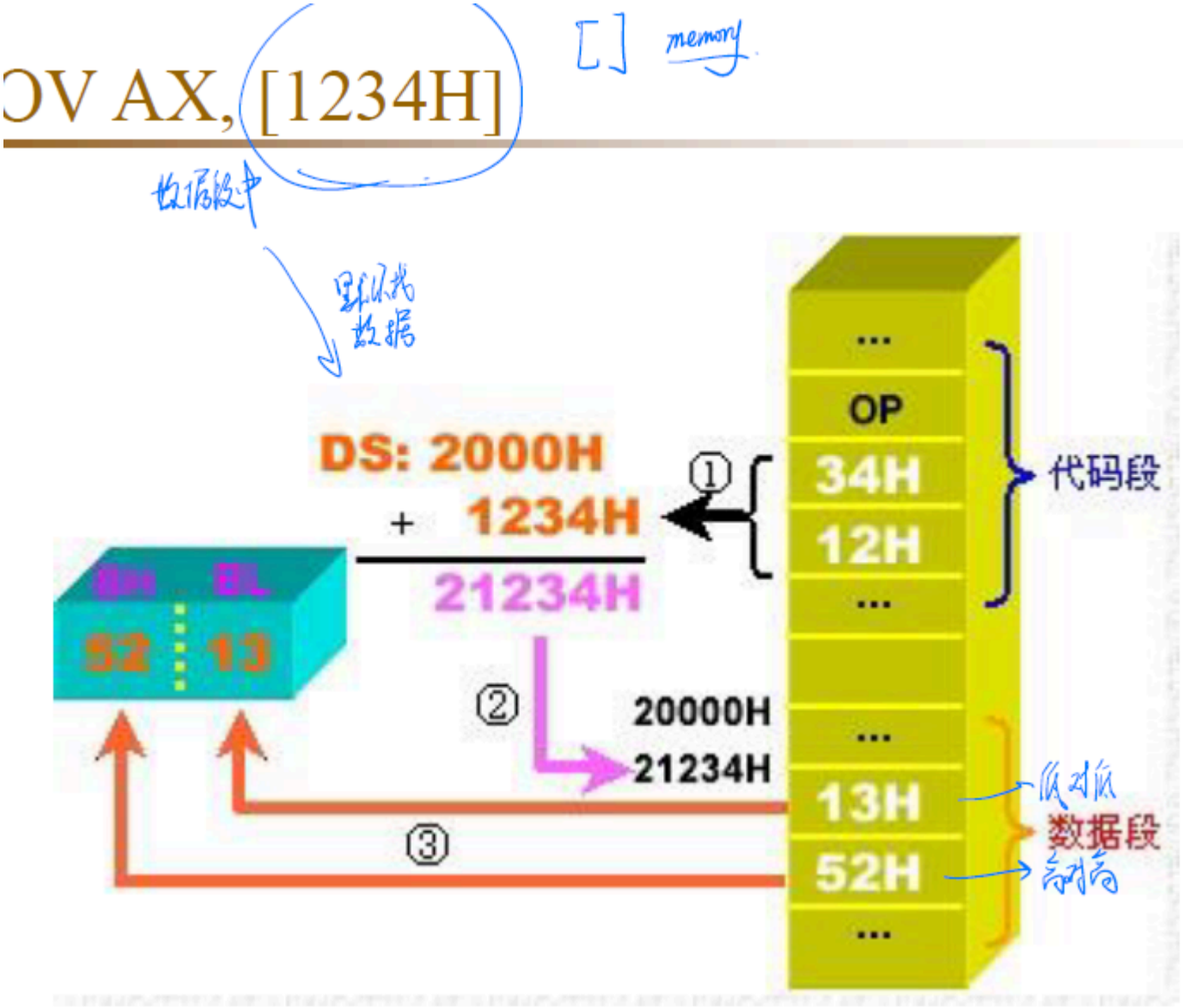
•由于数据段的段寄存器默认为DS，如果要指定访问其它段内的数据，在指令中用段前缀的方式显式地书写出来。

•例如：带有段前缀的直接寻址方式：MOV ES:[1000H], AX

这里的ES就是一个段超越，一般存数据都是DS段地址

•直接寻址方式常用于处理内存单元的数据，其操作数是内存变量的值，该寻址方式可在64K字节的段内进行寻址。
•注意：立即寻址方式和直接寻址方式的书写格式的不同，直接寻址的地址要写在括号“[ ]”内。在程序中直接地址通常用内存变量名来表示，如：MOV BX, VARW，其中，VARW是内存字变量。

MOV AX, [1234H]



compare：

立即寻址和直接寻址

•试比较下列指令中源操作数的寻址方式(VARW是内存字变量（DATA）)： MOV AX, 1234H MOV AX, [1234H]
前者是立即寻址， 后者是直接寻址
MOV AX, VARW MOV AX, [VARW]
两者是等效的， 均为直接寻址

（因为都是内存的寻址，注意：地址段，栈段，数据段，代码段都在内存中）

例：编写一段显示字符串STRING的程序

```
DATA      SEGMENT
STRING    DB    ‘ HAPPY NEW YEAR! ’, 0DH , 0AH , ‘ $ ’
COUNT     DW    17
DATA      ENDS
```

（1）直接寻址

本质是一个标号，用来寻址

```
mov   dl, string
mov   ah, 2
int    21h                    ; 显示字符 ‘H’

mov   dl, string+1
mov   ah, 2
int    21h                    ; 显示字符 ‘A’
 :
```

三、寄存器寻址方式（Register addressing）

指令所要的操作数已存储在某寄存器中，或把目标操作数存入寄存器。把在指令中指出所使用寄存器(即：寄存器的助忆符)的寻址方式称为寄存器寻址方式。

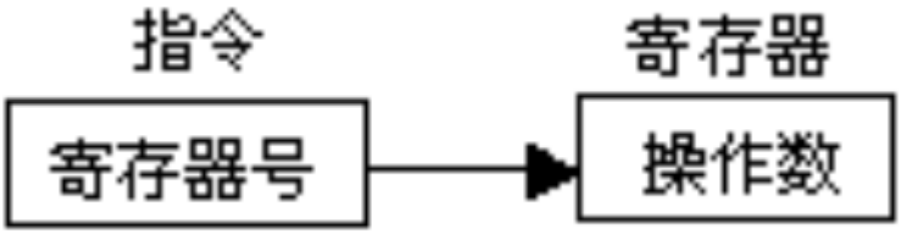
•指令中可以引用的寄存器及其符号名称如下：
8位寄存器有：AH、AL、BH、BL、CH、CL、DH和DL等
16位寄存器有：AX、BX、CX、DX、SI、DI、SP、BP和
段寄存器等；
32位寄存器有：EAX、EBX、ECX、EDX、ESI、EDI、
ESP和EBP等。
•寄存器寻址方式是一种简单快捷的寻址方式，源和目的操作数都可以是寄存器。

eg：MOVE AX,DATAS

MOVE DS,AX

注意：DS不能直接赋值

•它使用寄存器来存放要处理的操作数，寄存器号由指令指定，如下图所示。

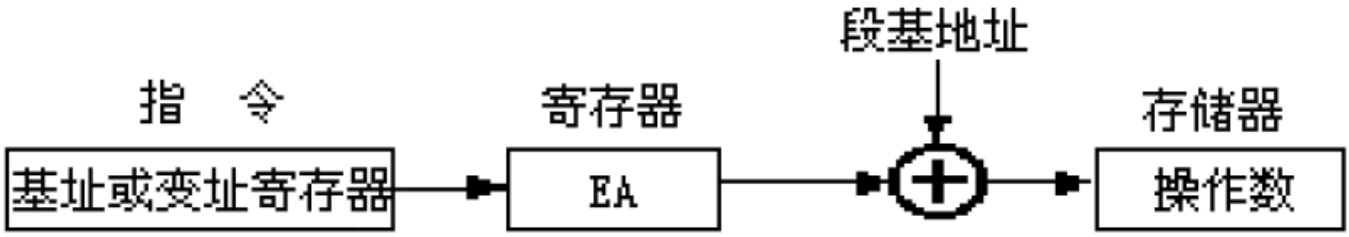


•16位操作数寄存器可以是**AX、BX、CX、DX、SI、DI、SP、BP、CS、DS、ES和SS**；对于8位操作数，寄存器可以是**AL、AH、BL、BH、CL、CH、DL、DH**。由于操作数就在寄存器中，指令执行时不需要访问存储器，因此这是一种**快速的寻址方式。**

四、寄存器 间接 寻址方式（Register indirect addressing）

（SP只能访问SS，但是BP除了SS还可以段超越）

•这种寻址方式通过 **基址 寄存器BX、BP或 变址 寄存器SI、DI来保存操作数的有效地址。如果指令中使用的寄存器是SI、DI和BX，则操作数在数据段中，（(DS)×16）再加上存器中的有效地址形成20位物理地址；如果指令中使用的寄存器是BP（这里用BP代替SP，是为了防止栈内东西被改，起到保护的作用）**，则操作数在堆栈段中，（(SS)×16）再加上BP中的有效地址形成20位物理地址。



$$PA=\begin{Bmatrix}(DS)\\(SS)\\\vdots\end{Bmatrix}\times 16+\begin{Bmatrix}(SI)\\(DI)\\(BX)\\(BP)\end{Bmatrix}$$

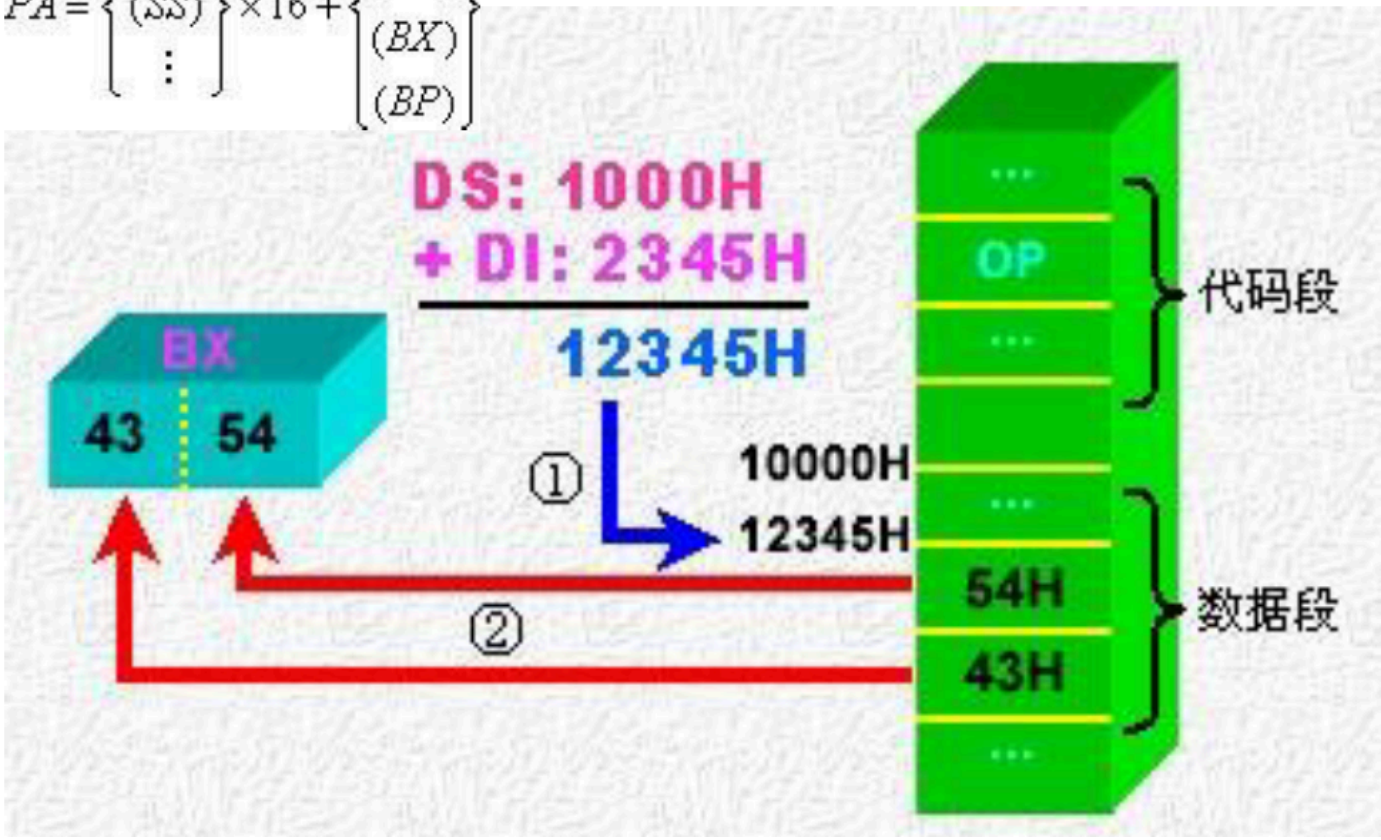
例:假设有指令：MOV BX,[DI]，在执行时，(DS)=1000H，(DI)=2345H，存储单元12345H的内容是4354H。问执行指令后，BX的值是什么?

解：根据寄存器间接寻址方式的规则，在执行本例指令时，寄存器DI的值不是操作数，而是操作数的地址。该操作数的物理地址应由DS和DI的值形成，即：

PA=(DS)*16+DI=1000H*16+2345H=12345H。

所以，该指令的执行效果是：把从物理地址为12345H开始的一个字的值传送给BX。
其执行过程如图所示。

$$PA=\begin{Bmatrix}(DS)\\(SS)\\\vdots\end{Bmatrix}\times 16+\begin{Bmatrix}(SI)\\(DI)\\(BX)\\(BP)\end{Bmatrix}$$



MOV AX, [BX]

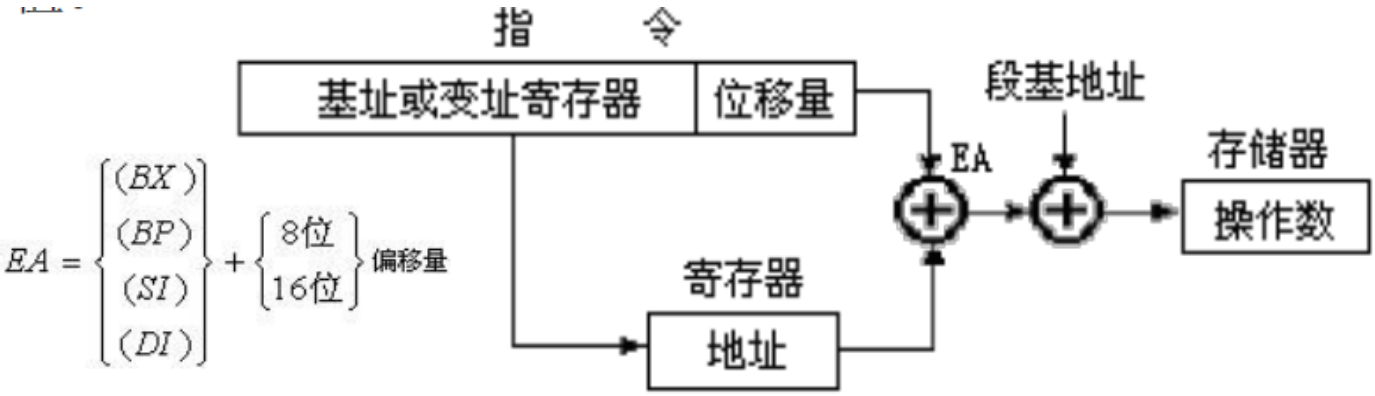
•如果（DS）= 2000H，（BX）= 1000H， 则物理地址 = 20000H + 1000H = 21000H 最后的执行结果为（AX）= 50A0H。

五、寄存器 相对 寻址方式（Register relative addressing）

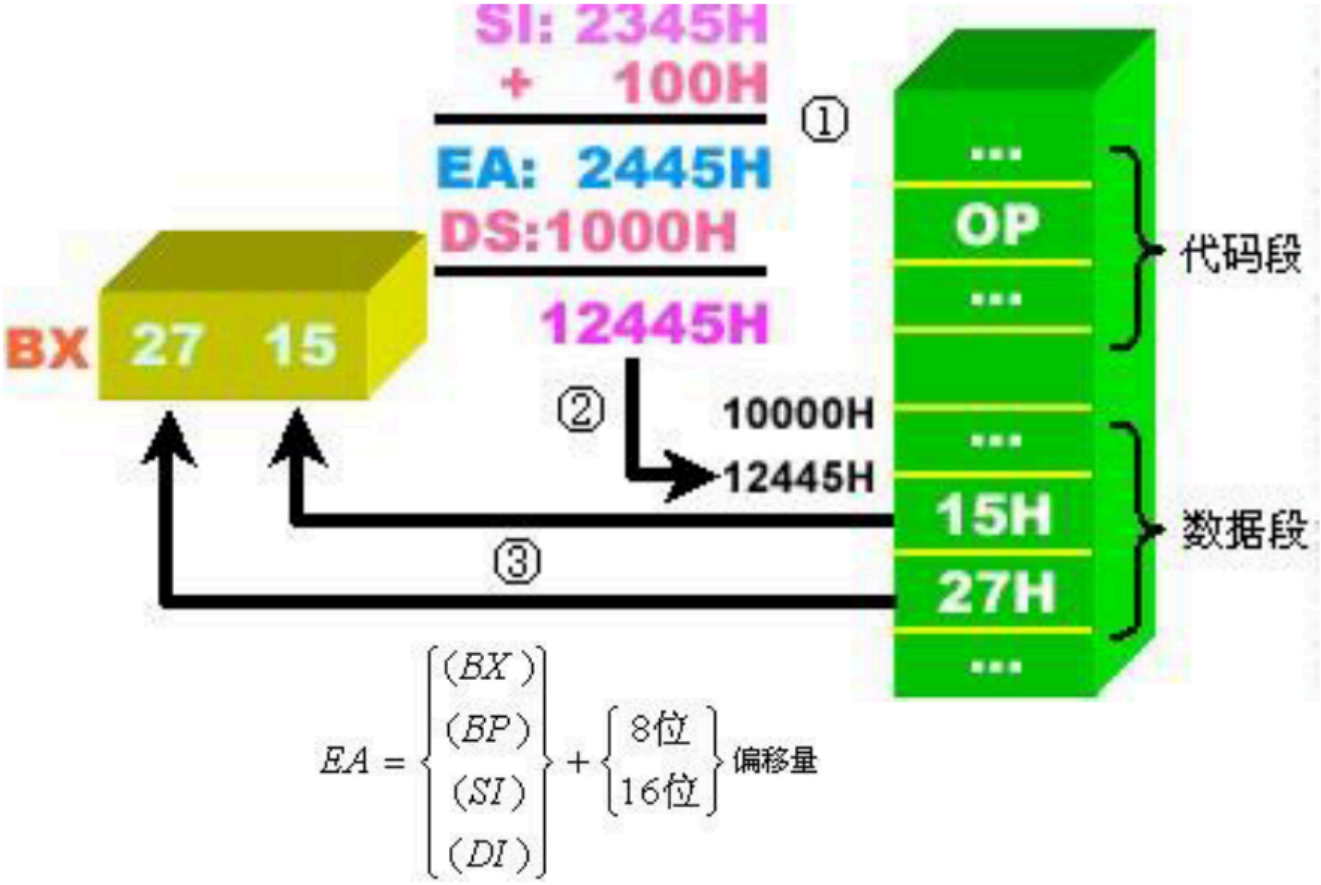
•通过基址寄存器BX、BP或变址寄存器SI、DI与一个位移量相加形成有效地址，计算物理地址的缺省段是**SI、DI和BX为DS，BP为SS。**寄存器相对寻址方式也可以使用段跨越前缀。 例如：MOV AX, ES:[DI+10h]

DI：用DI作为偏移，10h为首地址。

•寄存器相对寻址方式可用于表格处理。表格的首地址可设置为位移量，修改基址或变址寄存器的内容取得表格中的值。



- 例子：假设指令：MOV BX, [SI+100H]，在执行它时，(DS)=1000H，(SI)=2345H，内存单元12445H的内容为2715H，问该指令执行后，BX的值是什么?
  解：根据寄存器相对寻址方式的规则，在执行本例指令时，源操作数的有效地址EA为：
  EA=(SI)+100H=2345H+100H=2445H
  该操作数的物理地址应由DS和EA的值形成，即：
  PA=(DS)*16*+*EA*=*1000H*16+2445H=12445H。
  所以，该指令的执行效果是：把从物理地址为12445H开始的一个字的值传送给BX。
  其执行过程如图所示。

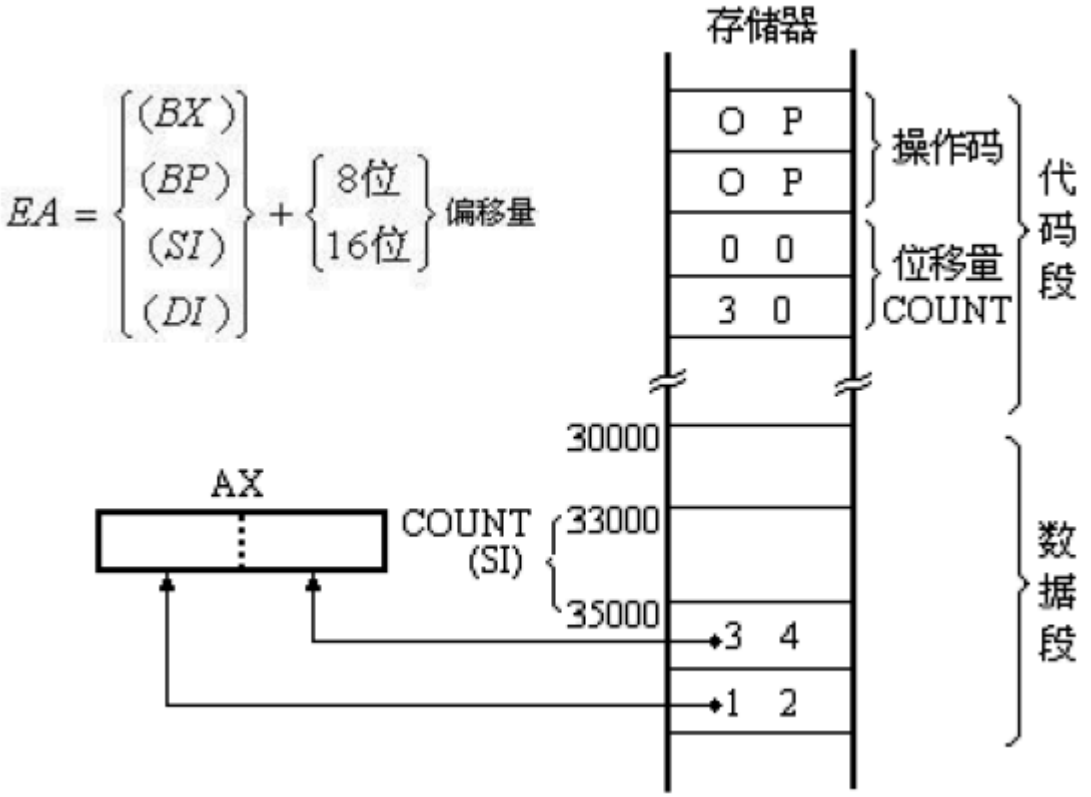


$$EA = \begin{Bmatrix} (BX) \\ (BP) \\ (SI) \\ (DI) \end{Bmatrix} + \begin{Bmatrix} 8位 \\ 16位 \end{Bmatrix} 偏移量$$

- MOV AX, COUNT[SI]

  （SI就相当于首地址）

- 如果 （DS）= 3000H，（SI）= 2000H，COUNT = 3000H 则物理地址 = 30000H + 2000H + 3000H = 35000H 指令执行的执行结果是（AX）= 1234H



六、基址变址寻址方式（Based indexed addressing)

8086 CPU指令系统

指令系统

• 微处理器通过执行程序来完成指定的任务，而程序是由一系列有序指令组成，微处理器是在这些指令的控制下工作的。微处理器可以识别的每一条指令称为机器指令，每一种处理器都有自己可以识别的一整套指令，称为指令集或指令系统。不同系列的微处理器，有不同的指令系统，它是根据CPU硬件特点研制出来的，处理器执行指令时，根据不同的指令采取不同的动作，完成不同的功能，既可以改变自己内部的工作状态，也能控制其它外围电路的工作状态。

指令的基本内容

• 计算机的指令有**微指令、机器指令和宏指令**之分。微指令是微程序级的命令，属于硬件；宏指令是由若干机器指令组成，属于软件；机器指令介于二者之间，因而是硬件和软件的界面。
•指令由操作码和操作数构成，8086指令的一般格式如下：
操作码 [操作数]，[操作数]
操作码用助记符来表示（一般是英文单词缩写）。根据指令的不同，操作数可以是一个，即单操作数，也可以是两个，即双操作数（源操作数和目标操作数），有的指令还可以没有操作数或隐含操作数（例如STI,CLI,以及自加1等等）。例如指令MOV AX，DX 中的MOV是助记符，AX，DX为操作数（双操作数），这条指令的功能是将DX中的内容送到AX中。

8086系统的操作数

（1）**立即数操作数**
• 所谓立即数是指具有固定数值的操作数，即**常数**。它可以是字节或字（8位或16位）。存放时，该操作数跟随指令操作码一起存放在指令区，故又称为指令区操作数。

（2）**寄存器操作数**register

• 操作数事**先存放在某寄存器**中（CPU的通用寄存器、专用寄存器或段寄存器），只要知道**寄存器的名称**（编号）就可以寻找到操作数。寄存器操作数既可作为源操作数，又可作为目标操作数。

（3）**存储器操作数**memory

• 操作数事先存**放在存储器中**存放数据的**某个单元**，只要知道**存储器的地址**即可寻到操作数。当然，操作数也可以存放在堆栈中（堆栈是存储器的一个特殊区域），只要知道堆栈指针，就可以用栈操作指令寻找操作数。

•通用数据传送指令：MOV

• 堆栈操作指令：PUSH ，POP

• 地址传送操作指令：LEA

• 标志寄存器传送指令：LAHF，SAHF，PUSHF，POPF

• 加法指令：ADD，ADC，INC

• 减法指令：SUB，SBB，DEC，NEG，CMP

• 乘法指令：MUL

• 除法指令：DIV

移位指令

控制转移指令JMP