第三章 80x86寻址方式与指令系统 (1)

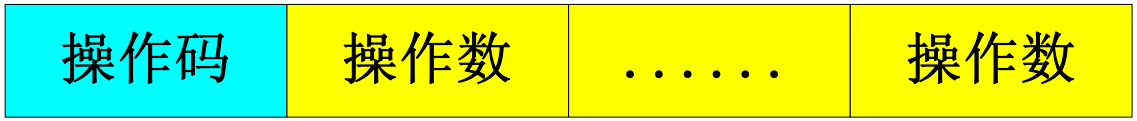
3-0 基础知识

一、指令系统概念

- 每种计算机都有一组指令集提供用户使用，这组指令集就称为计算机的指令系统。
- 机器指令：计算机能识别和执行的指令的二进制代码。如： **0000 0110 0010**
- 汇编指令：用助记符表示机器指令的操作码和操作数。

二、指令组成

- 计算机中指令由操作码字段和操作数字段两部分组成。



  - 操作码字段-----指示计算机要执行的操作
  - 操作数字段-----指出在指令执行操作过程中所需要的操作数；该字段可以是：

    1）操作数本身

    2）操作数地址或是地址的一部分；

    3）指向操作数地址的指针；

    4）或其他有关操作数的信息。

  - 大多数指令包括1～2个操作数，少数隐含第三个操作数。

三、操作数的存放

- 操作数的存放不外乎三种情况：

（1） 操作数包含在指令中

- 即指令的操作数字段包含操作数本身。这种操作数称为立即数。
- 例：**MOV AL , 08H ；** **08H**为立即数

（2）操作数包含在**CPU**的内部寄存器中

- 例：**INC CX** ；操作数存放在**CX**寄存器中

（3）操作数在内存数据区

- 操作数在内存数据区，操作数字段包含着此操作数的地址。

四、寻址

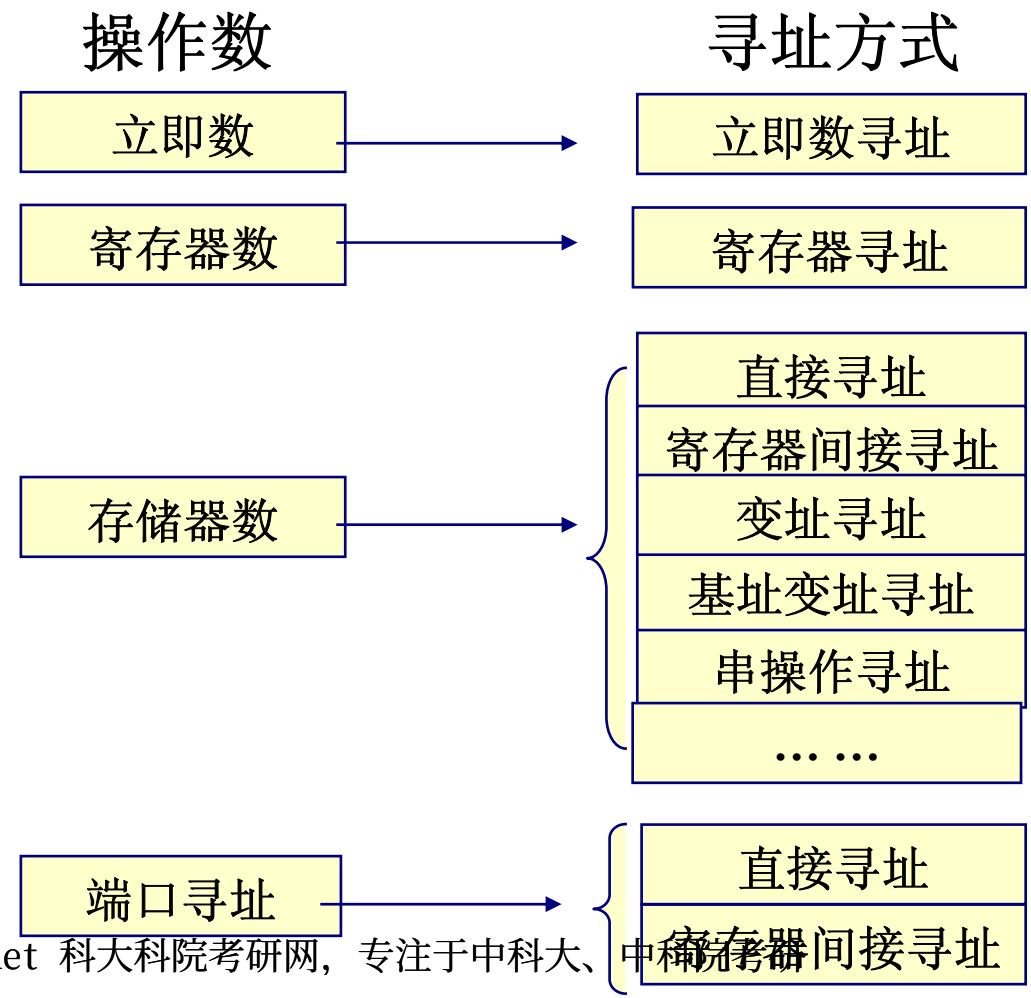
- 寻址——根据指令内容确定操作数地址的过程。

- 有效地址——根据寻址方式计算所得到的地址（**Effective Address-EA)**，也就是段内偏移地址。有效地址还需要与相应的段基地址组合才能产生物理地址（**PA**），这部分工作由**CPU**完成。

- 寻址方式——如何寻找内存操作数的方法。不同寻址方式实质上是构成段内的偏移量的方法不同。

80x86操作数的寻址方式

- 80x86计算机中操作数按存放的方法分为：
  - 立即数（指令中）
  - 寄存器数
  - 存储器数
  - I/O端口



五、80x86的数据类型与存储方式

- 80x86结构的基本数据类型
  - 字节：　　8位
  - 字：　　16位，2个字节
  - 双字：　　32位，4个字节
  - 四字：　　64位，8个字节（80486CPU引入）
  - 双四字：　　128位，16个字节（Pentium III)
- 数据在内存中存储方式
  - 多字节数据的存放原则：低位字节在低端地址，高位字节在高端地址。最低地址就是操作数的地址。
  - 对准存放。不是必须的，但是对准存放可以减少CPU读数据所需的操作步骤。

3-1 80x86的寻址方式

80x86的寻址方式可以分为10种

- 1、立即寻址
- 2、寄存器寻址
- 3、直接寻址
- 4、寄存器间接寻址
- 5、寄存器相对寻址
- 6、基址变址寻址
- 7、基址变址相对寻址
- 8、比例变址寻址
- 9、基址比例变址寻址
- 10、相对基址比例变址寻址

其中：方式3～方式10是与存储器有关的寻址方式；
方式8～方式10仅适用于80386及其后继CPU。

一、与存储器无关的二种寻址方式

1、立即寻址（Immediate addressing）

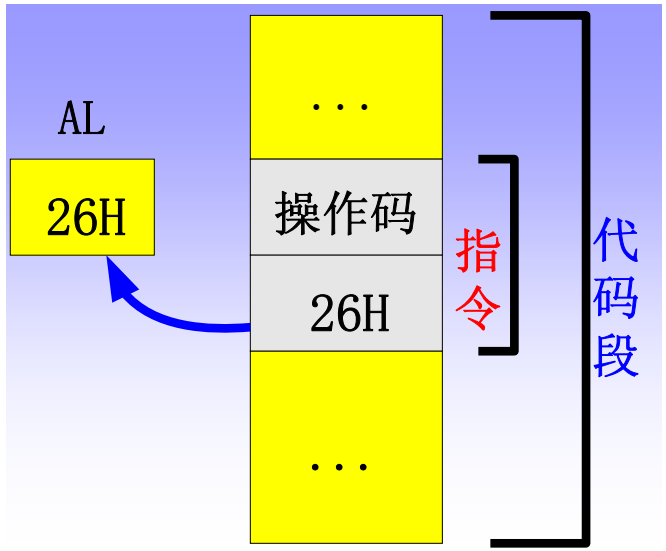
- 操作数以常量形式直接放在指令中，紧跟操作码之后。
- 机器码存放形式如下：

8位立即数

16位立即数

- 立即寻址示例1

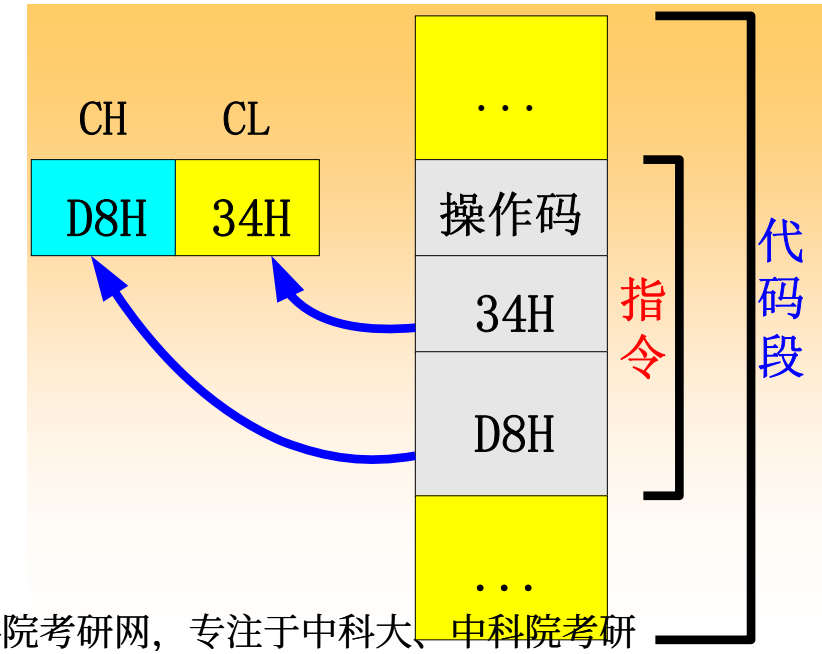
MOV AL，26H； 执行后26H →AL，即（AL）＝26H



- 立即寻址示例2

MOV CX， 0D834H；

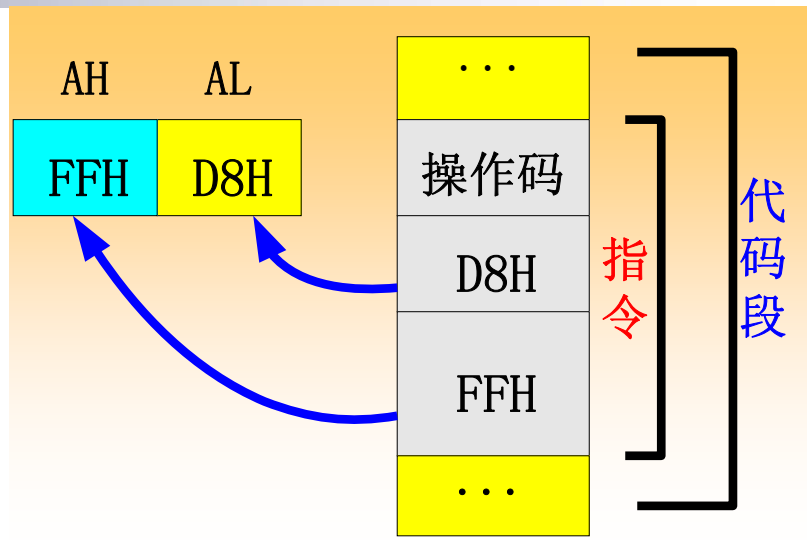
执行后（CX）＝0D834H



- 立即寻址示例3

MOV AX，-40；

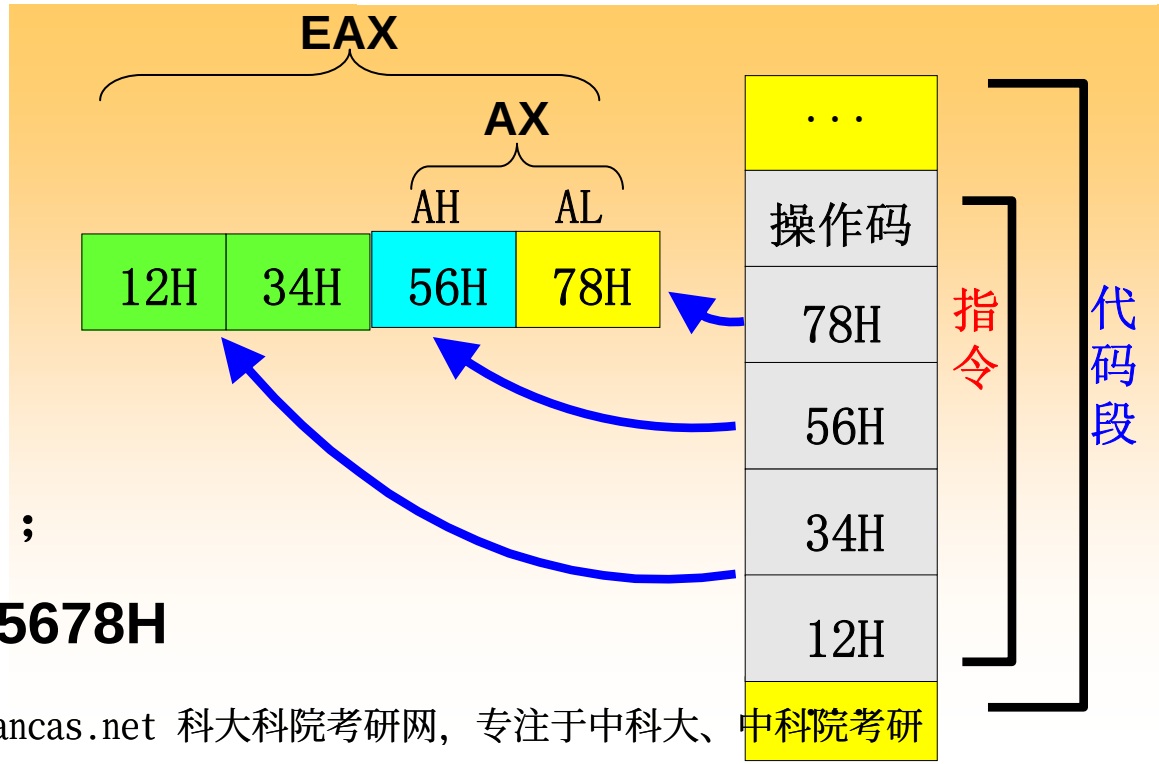
执行后（AX）＝ 0FFD8H（-40）



- 立即寻址示例4

MOV EAX, 12345678H ；

执行后（EAX）＝12345678H



- 关于立即寻址方式的几点说明：

1）立即寻址常用于给寄存器赋值；

2）立即数不仅能送到寄存器，而且也可送往存储单元；

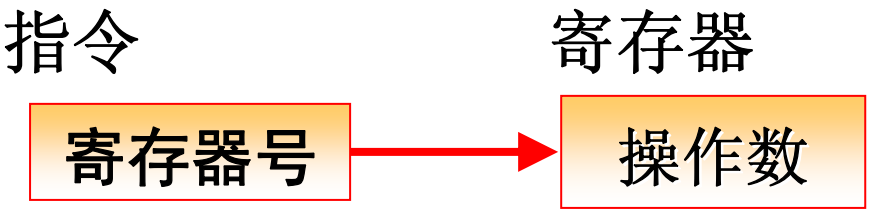
3）立即数只能是源操作数，不能作为目的操作数；

4）立即寻址的操作数就存在指令序列中，CPU不需要访问内存就可以获得操作数，指令执行速度快。

此外，以A～F开头的数字出现在指令中，前面一定要加数字0，以区别其它符号。

2、寄存器寻址（Register addressing）

- 操作数存放在某个寄存器中，指令指定寄存器号。



- 寄存器寻址示例

```
MOV AH, BL      ；(BL)→AH，即(AH)＝(BL)
MOV  DS, AX     ；(AX)→DS，即(DS)＝(AX)
MOV  SI, AX     ；(AX)→SI ，即(SI)＝(AX)
MOV  ECX, EDX   ；(EDX)→ECX，即(ECX)＝(EDX)
```

- 寄存器寻址方式也不需访问存储器即可 得到操作数，指令执行速度快。

二、与存储器有关的寻址方式

- 有效地址的计算：

  EA＝基址 +（变址×比例因子）+ 位移量

- 有效地址(EA)的4种组成成分

  1）*位移量*（displacement）

  - 存放在指令中的8位、16位或32位的数字，是一个地址。

  2）*基址*（base）

  - 存放在基址寄存器中的内容，用于指向数组的首地址。

  3）*变址*（index）

  - 存放在变址寄存器中的内容，用于访问数组的某个元素。

  4）*比例因子*（scale factor）

  - 其值可为1，2，4或8，386及其后继机型新增加的。

16/32位寻址时有效地址四种成分的组成

四种成分	16位寻址	32位寻址
位移量	0, 8, 16位	0, 8, 32位
基址寄存器	BX, BP	任何32位通用寄存器(包括ESP)
变址寄存器	SI, DI	除ESP以外的32位通用寄存器
比例因子	无	1, 2, 4, 8

默认段选择规则

访存类型	所用段及寄存器	缺省选择规则
指令	代码段 CS	用于取指
堆栈	堆栈段 SS	所有的堆栈的进栈和出栈任何用ESP或EBP作为基址寄存器的访存
目的串	附加段 ES	串处理指令的目的串
局部数据	数据段 DS	除相对于堆栈以及串处理的目的串以外的所有数据访问

访问非默认段数据的方法——段超越

- 数据存放较灵活，除了放置在默认的DS段外，还可放在其它段，对其访问时需使用段超越前缀，可用的段超越前缀有CS:、DS:、ES:、SS:、FS:和GS:。
- 段超越举例：
  - MOV AX, [10H] ；默认DS段10H处的一个字赋给AX
  - MOV AX, ES:[10H] ；ES段10H处的一个字的数据赋给AX
- 不允许使用段超越前缀的情况

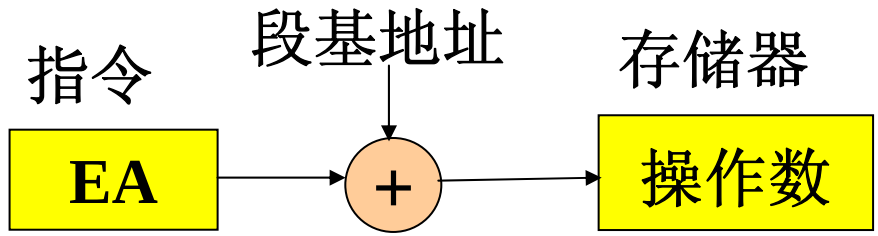
（1）串操作指令的目的串必须用ES段

（2）PUSH指令的目的和POP指令的源必须用SS段

（3）程序的指令必须存放在CS段

3、直接寻址（Direct addressing）

- 操作数在存储器中，而存放操作数的内存单元地址的偏移量（有效地址）在指令中。
- **EA＝基址＋（变址×比例因子）＋ 位移量**



- 例如：**MOV　AX, [2000H]**
  - 对比：　**MOV　AX, 2000H**　　；立即寻址
- 指令中的有效地址必须加方括号，以便与立即数相区别。
- 直接寻址方式适用于处理单个变量。
- 直接寻址方式隐含段寄存器是 **DS**，但允许段跨越。用段超越前缀说明所使用的段。

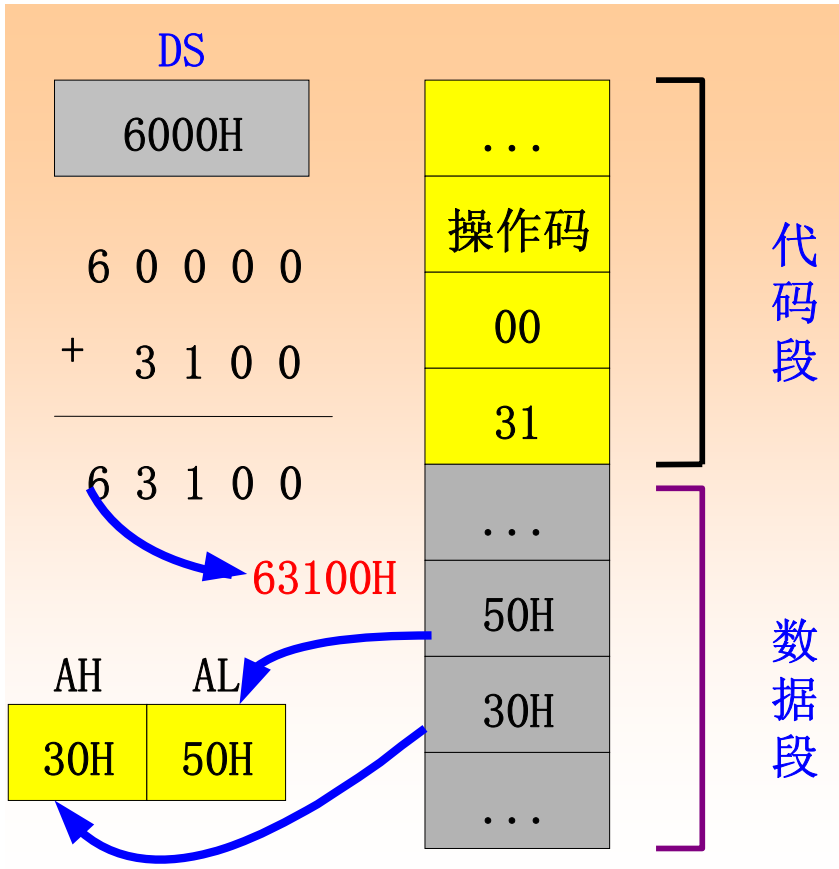
例1：MOV AX，[3100H]

假设:（DS）= 6000H，

（63100H）= 3050H 。

则指令执行后:

（AX）=3050H



若操作数不在数据段，则应指定段超越。如，假设操作数在附加段，则指令应为:

MOV AX，ES:[3100H]

“:”称为修改属性运算符号。

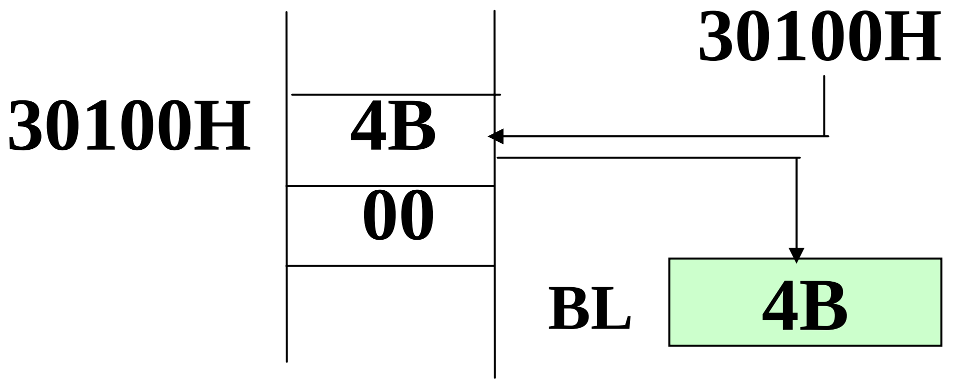
直接寻址方式

例2：假设(ES)=3000H，物理地址30100H的单元内容为004BH，以下指令执行后，BL寄存器的内容＝？

MOV BL, ES:[0100H]

物理地址

$$\begin{array}{r} 30000H \\ +\ \ 0100H \\ \hline 30100H \end{array}$$



在汇编语言指令中，可以用符号地址代替数值地址

如：　**MOV AX, BUFF**　或者**MOV AX, [BUFF]**

　　MOV AX, AREA1　或者**MOV AX, [AREA1]**

其中：**BUFF**和**AREA1**为存放数据单元的符号地址。

- 说明：
  - 在汇编语言中，类似**BUFF**和**AREA1**这样的符号不仅可以代表指令中的符号地址，也可以表示一个立即数。程序中必须事先安排说明语句（伪指令）进行说明。
  - 伪指令——不属于**CPU**的指令集，没有对应的机器代码，只是用于在汇编语言指令的编译过程中，完成变量定义、存储分配、段定义等。

例**3-9**：

```
AREA1  EQU  0867H;
... ...
MOV   AX，AREA1
```

  - **EQU是赋值伪指令**，此后**AREA1**就代表一个立即数**0867H**；**MOV AX，AREA1**执行后，寄存器**AX**的内容为**0867H**。

例3-10：AREA1 DW 0867H；

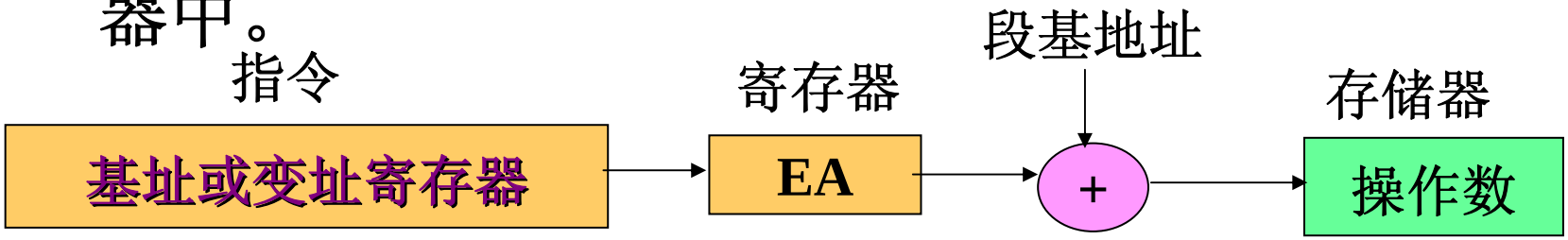
… …

MOV AX，AREA1

- DW是定义字变量伪指令语句，此后AREA1就是一个字变量的名称，变量在程序是可以修改的运算对象，是内存中的一个存储区域，AREA1就是该存储区域的符号地址，该地址存储了一个字0867H。
- MOV AX，AREA1执行后，CPU从逻辑地址为DS:AREA1的单元中取出一个字装入AX寄存器，AX的内容也是0867H。
- 注意：例3-9与例3-10的区别。

4、寄存器间接寻址方式（Reg. indirect addr.）

- 操作数在存储器中，操作数地址的偏移量在寄存器中。



- EA＝基址＋（变址×比例因子）＋ 位移量

1) 若指令中指定BX、SI或DI寄存器作为基址寄存器进行间接寻址，操作数一般在现行数据段区域中，应该使用(DS)作为段地址。即操作数物理地址（PA）

PA＝16 ×（DS）+（BX）

或 ＝16 ×（DS）+（SI）

或 ＝16 ×（DS）+（DI）

例：MOV AX，[DI]

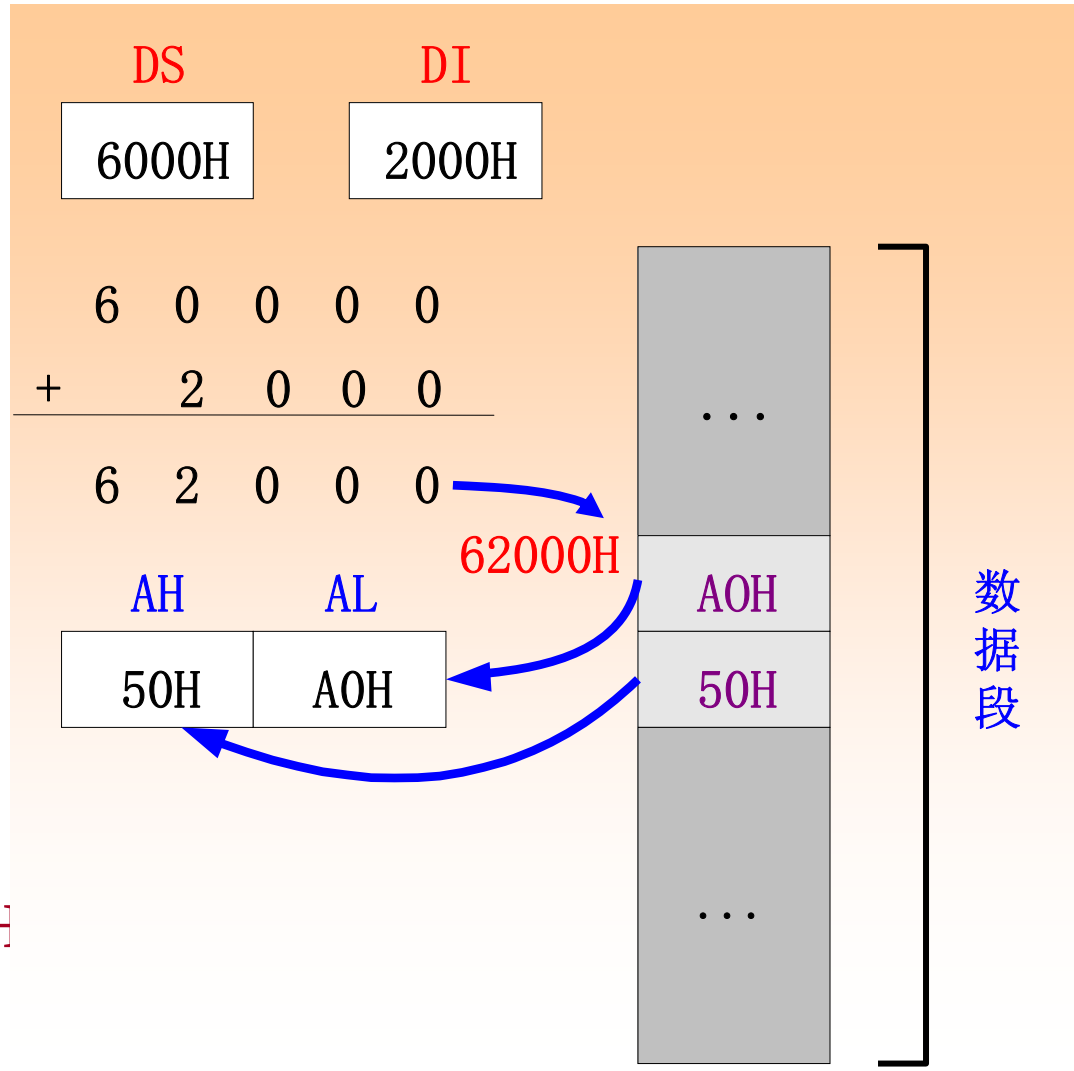
假设：(DS)=6000H

(DI)=2000H

则：PA=62000H

假设：(62000H)=50A0H

指令执行后，(AX)=50A0H



注意：寄存器名称必须加方括号，以便与寄存器寻址相区别

寄存器间接寻址方式

MOV AX，[DI]

2）若选择BP寄存器作为间接寻址

操作数在堆栈段区域中，用SS寄存器的内容作为段地址。

操作数物理地址：

PA=16×（SS）+（BP）

例：　MOV　[BP]，AX

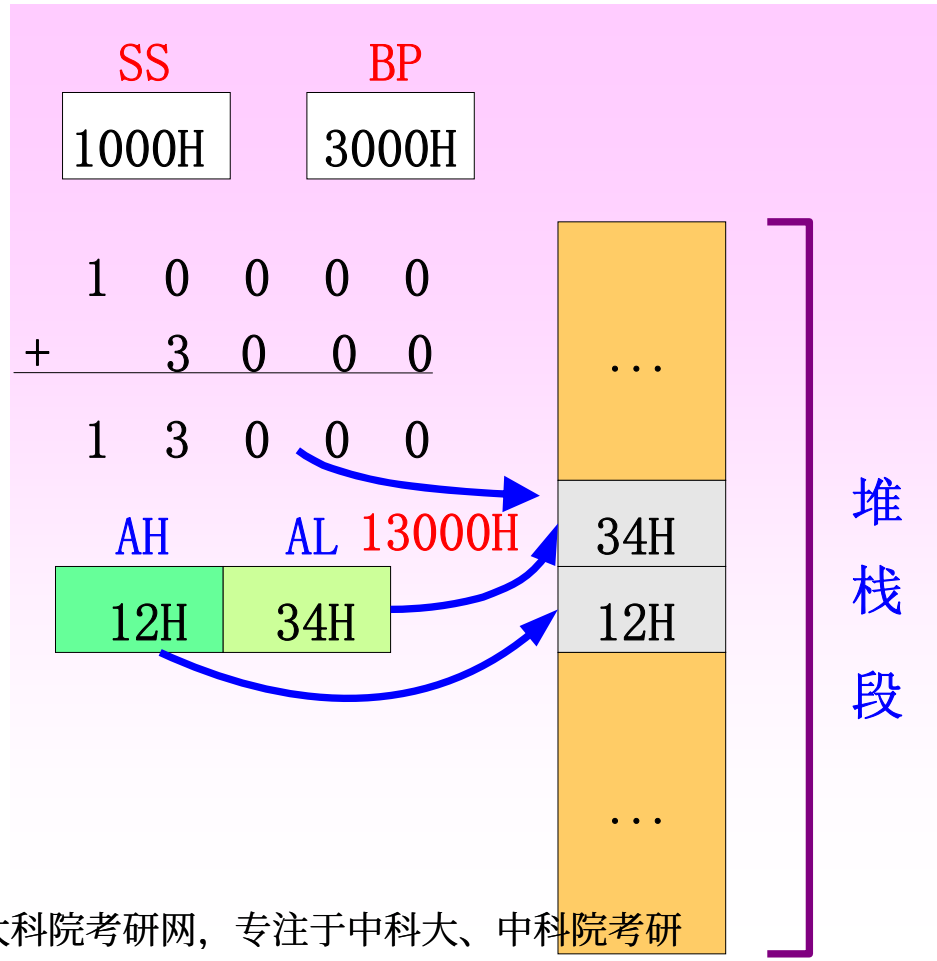
执行前：　(SS)=1000H，

(BP)=3000H，

(AX)=1234H

执行后：PA=13000H

(13000H)=1234H



3）用 SI、DI、BX 、BP作为间接寻址允许段跨越

指令中可以指定段跨越前缀来取得其它段中的数据。

例：　MOV　ES:[DI], AX；操作数在附加段ES中，
物理地址：PA = 16 ×（ES）+（DI）

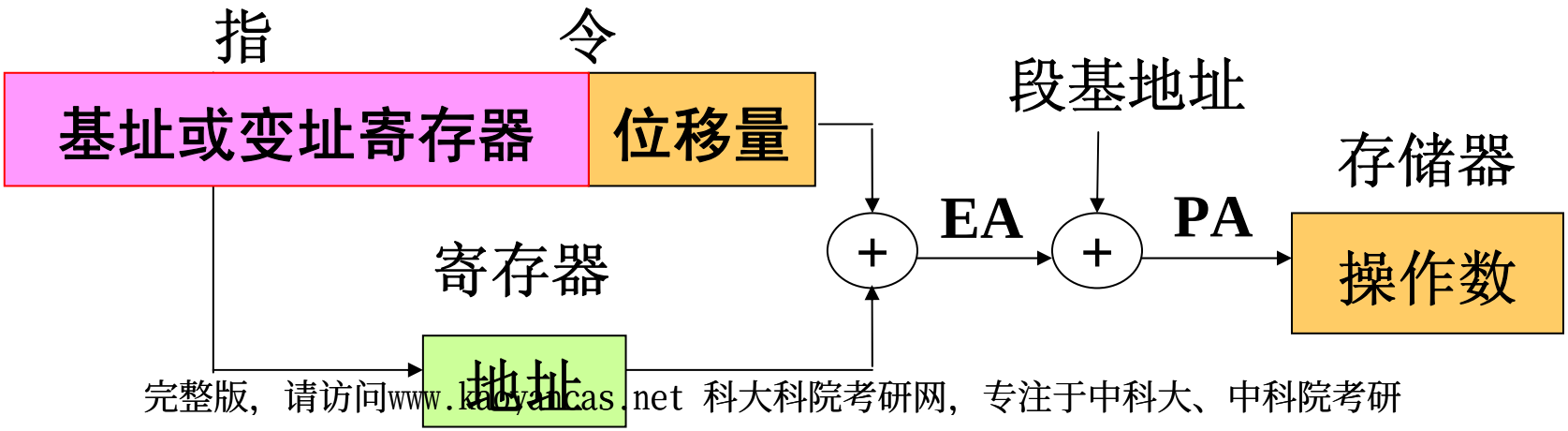
又例：MOV DX, DS:[BP];

指令中若指定BP间接寻址，默认操作数在堆栈段。但是这条指令用段超越前缀从默认段以外的DS段中取得数据，物理地址PA=16 ×（DS）+（BP）

这种寻址方法可以用于表格处理。

5、寄存器相对寻址方式（Reg. relative addr.）或变址寻址 （Index Addressing）

- 与寄存器间接寻址类似，只不过操作数的有效地址是基址（或变址）寄存器的内容再加上指令中指定的位移量。
- EA＝基址 +（变址×比例因子）+ 位移量
- 指令中指定的基址寄存器为BX、SI或DI时，默认的段寄存器是DS；若指定的寄存器是BP时，默认的段寄存器为SS。
- 同样，该方式可以使用段超越。



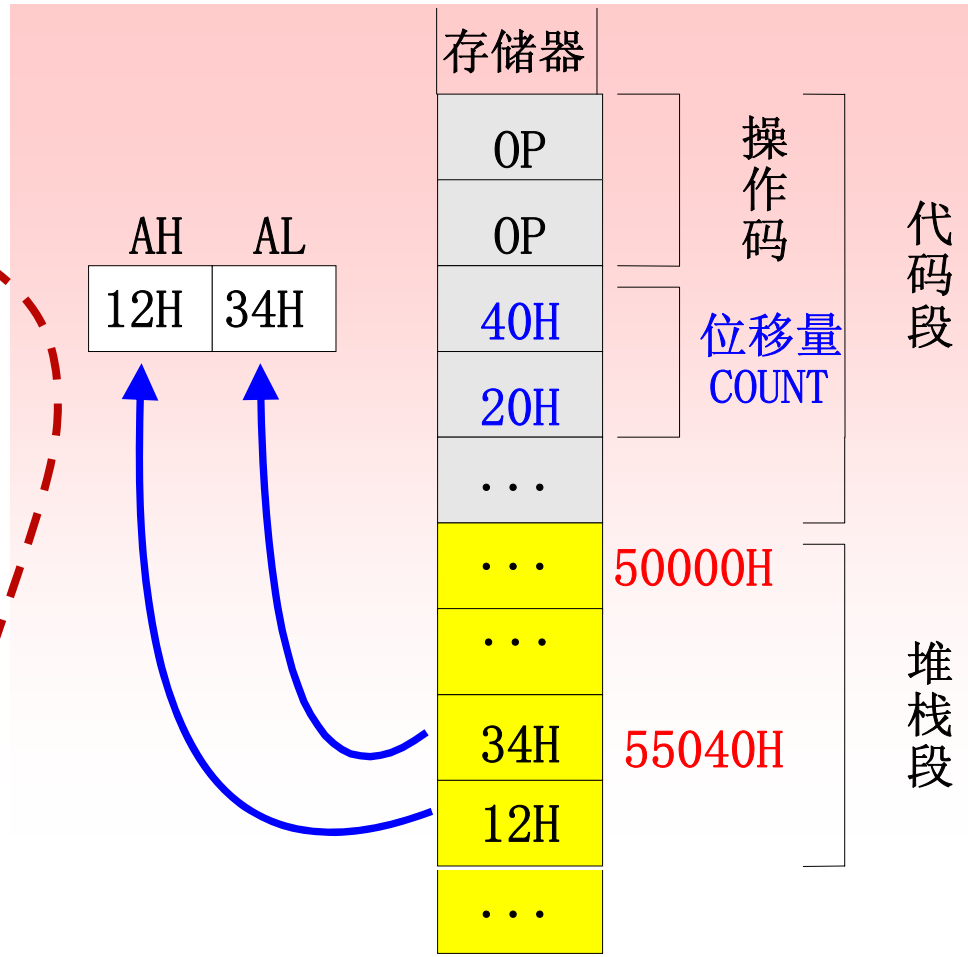
例：以下三条指令完全等价

MOV AX, COUNT [BP]
或MOV AX, [COUNT+BP]
或MOV AX, COUNT+[BP]

COUNT为符号地址，本例中为16位的偏移量。

假设指令执行前：
(SS)=5000H， (BP)=3000H，
COUNT=2040H，
(AX)=4321H

则指令执行后：
EA=5040H， PA=55040H
(55040H)=1234H
(AX)=1234H



- 用途：这种寻址方式适用于表格处理。
  - 表格首地址COUNT，修改基址或变址寄存器对表格中的表项进行访问。
- 例：某8位数据表的首地址为COUNT，欲将表中第10个数据读取到AL中。
  - 第10个数据的有效地址：EA= COUNT + 9

    MOV SI , 09H

    MOV AL , [SI+COUNT]
  - 要读取表格另外的数据，改变SI的内容即可。

6、基址加变址寻址方式（Based indexed addressing）

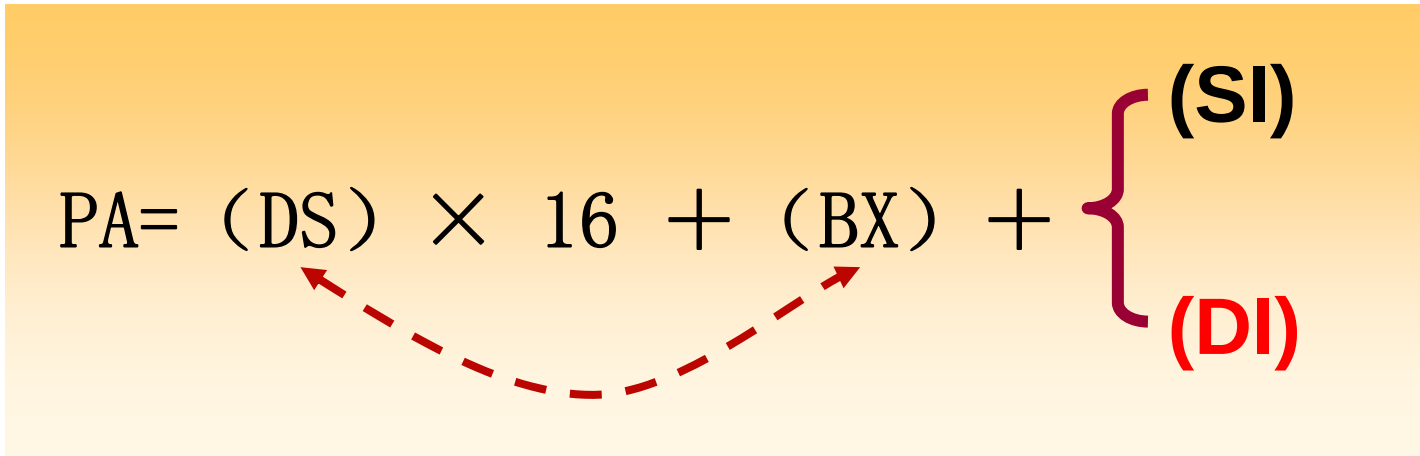
- 操作数的有效地址是一基址寄存器和一变址寄存器的内容之和，基址寄存器和变址寄存器均由指令指定。
- **EA＝基址 +（变址×比例因子）+ 位移量**
- 适用于16位和32位寻址。例如：

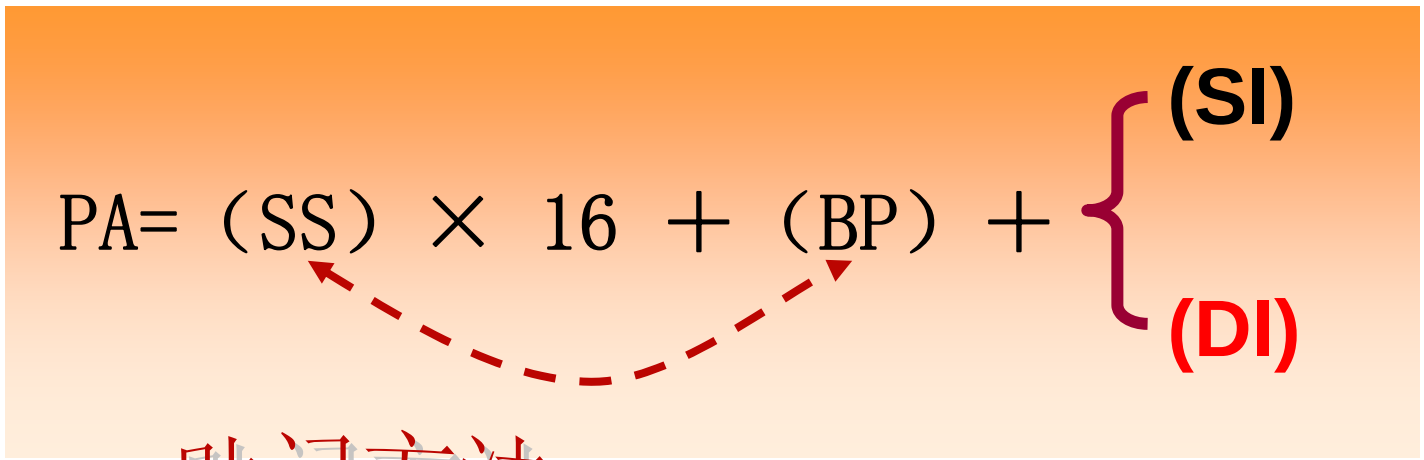
MOV AX, [BX+SI]　　MOV EAX, [EDX+EBP]

$$EA = (BX) + \begin{cases} (SI) \\ (DI) \end{cases}$$

$$EA = (BP) + \begin{cases} (SI) \\ (DI) \end{cases}$$

- 除有段跨越前缀之外，形成物理地址有二种方式：





助记方法：DX vs SP

例：以下两条指令等价

MOV AX, [BX][SI]

或 **MOV AX, [BX+SI]**

假设执行指令前：

(DS)=3200H,

(BX)=0456H,

(SI) =1094H

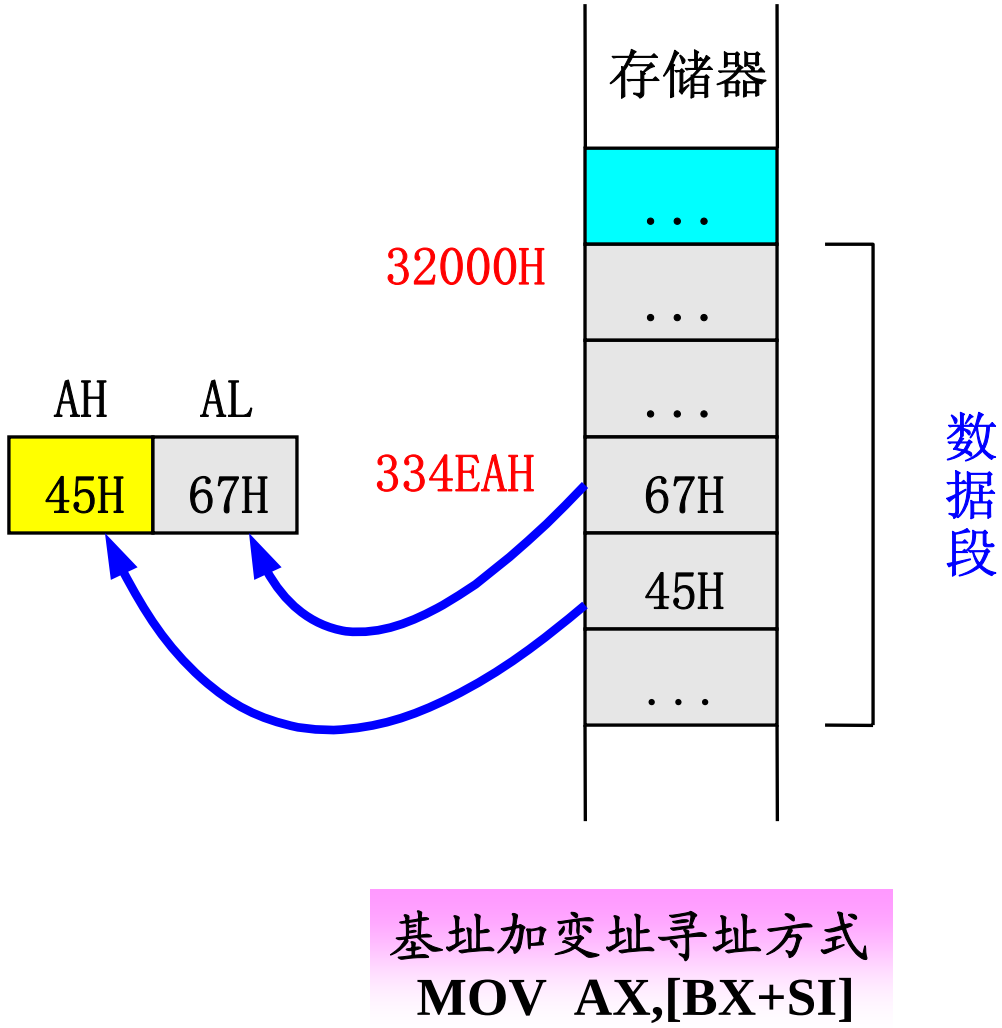
(334EAH)=4567H

执行指令后：

EA=14EAH

PA=334EAH

(AX)=4567H



基址加变址寻址方式

MOV AX,[BX+SI]

- 用途：
  - 这种寻址方式适用于二维数组操作和二重循环。
    - 访问二维数组：表格首地址放在基址寄存器中，用变址寄存器来访问数组中的元素。
    - 二重循环：基址寄存器用于外循环计数，变址寄存器用于内循环计数。
  - 二个寄存器都能修改，所以比直接变址方式更加灵活。

7、相对基址加变址寻址方式（Relative based indexed addressing）

- 操作数的有效地址是一个基址寄存器的内容、一个变址寄存器的内容再加上偏移量之和。即：

  EA＝基址 +（变址×比例因子）+ 位移量

  例如：**MOV AX, [BX+SI+50]**

$$EA = (BX) + \begin{cases} (SI) \\ (DI) \end{cases} + \begin{cases} 8\text{位偏移量} \\ 16\text{位偏移量} \end{cases}$$

$$EA = (BP) + \begin{cases} (SI) \\ (DI) \end{cases} + \begin{cases} 8\text{位偏移量} \\ 16\text{位偏移量} \end{cases}$$

- 除有段跨越前缀之外，形成物理地址有二种方式：

$$PA=(DS)\times 16+(BX)+\begin{cases}(SI)\\(DI)\end{cases}+\begin{cases}\text{8位偏移量}\\\text{16位偏移量}\end{cases}$$

$$PA=(SS)\times 16+(BP)+\begin{cases}(SI)\\(DI)\end{cases}+\begin{cases}\text{8位偏移量}\\\text{16位偏移量}\end{cases}$$

例：以下三条指令等价

MOV AX, MASK[BX][DI]

MOV AX, MASK [BX+DI]

MOV AX,[MASX+BX+DI]

假设执行指令前：

(DS)=3000H，(BX)=1346H

(DI)=0500H，MASK=1234H

(32A7AH)=4050H

执行指令后：

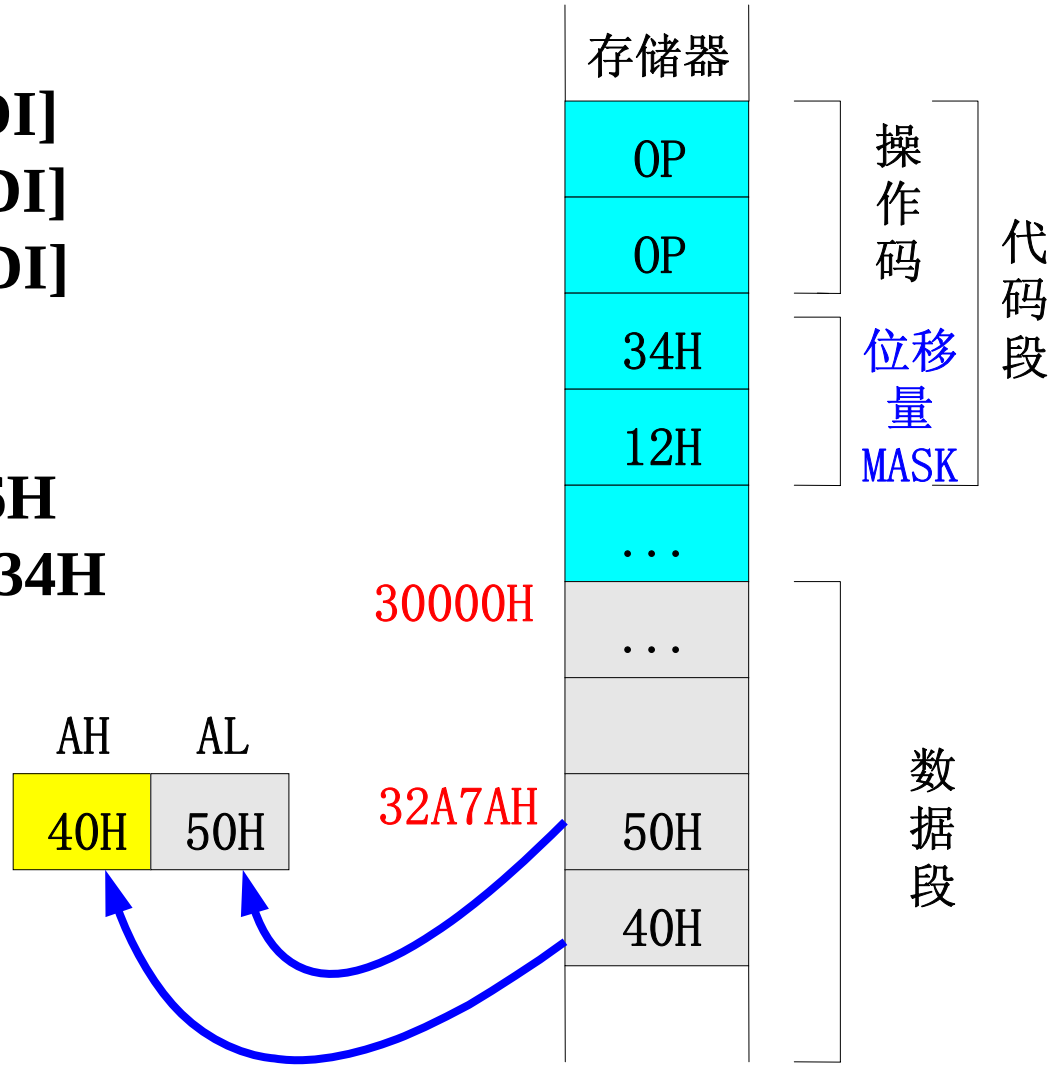
EA = MASK + BX + DI

= 2A7AH

PA = 16 × DS + EA

= 32A7AH

(AX)=4050H



相对基址加变址

MOV AX,MASK[BX+DI]

- 用途：
  - 主要用于二维数组操作，位移量为数组起始地址，适用于**16**位和**32**位寻址。
  - 另外这种寻址方式也为堆栈处理提供方便，主程序常常利用堆栈与子程序传递参数。
    - **(BP)→** 栈顶（一般 **BP**可指向栈顶）
    - 从栈顶到数组的首地址可以用位移量（**MASK**）表示。
    - 变址寄存器（**SI**）或（**DI**）——指向数组中某个元素。

8、比例变址寻址

- 80386以后CPU增加的寻址方式，只适用于32位寻址。
- 操作数的有效地址是变址寄存器的内容乘以指令中指定的比例因子及位移量之和，即

  EA＝基址 + （变址×比例因子） + 位移量

  例：MOV EAX, [ESI*4+50]

- 比例因子可以是1、2、4或8，隐含比例因子是1，位移量为8位或32位立即数。
- 适用于一维数组操作，当数组元素大小为2/4/8字节时，它更方便、有效。

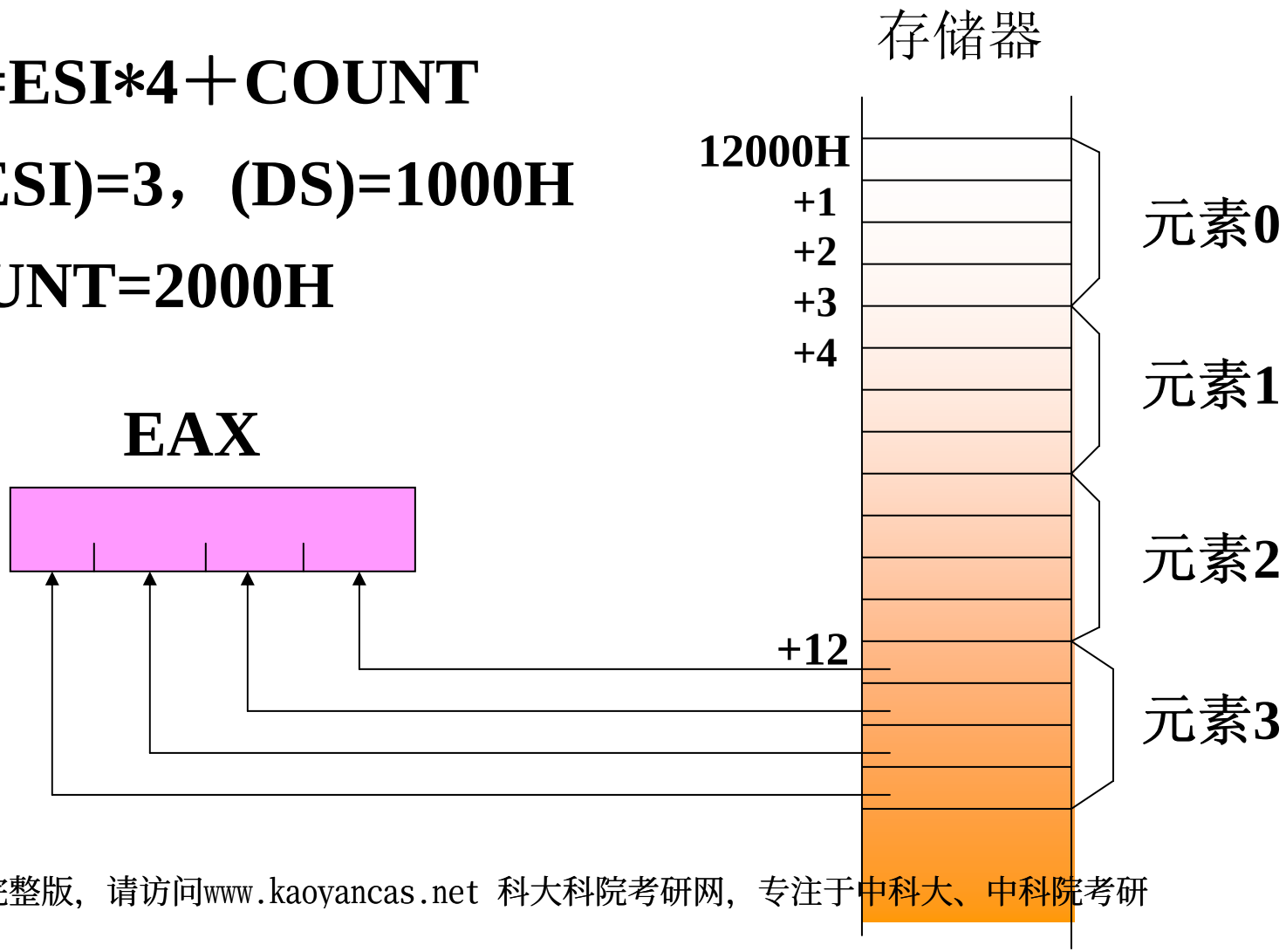
比例变址寻址示例

MOV EAX, COUNT[ESI*4]

EA=ESI*4＋COUNT

设(ESI)=3，(DS)=1000H

COUNT=2000H



9、基址比例变址寻址

- 80386以后CPU增加的寻址方式，只适用于32位寻址。
- 操作数有效地址是变址寄存器的内容乘以指令中指定的比例因子加基址寄存器内容之和，即：

  EA＝基址 +（变址×比例因子）+ 位移量

  如： MOV EAX, [EDX*8+EAX]

  MOV ECX, [EBX+ESI*8]

- 适用于数组元素大小为2/4/8字节时的二维数组操作。

10、相对基址比例变址寻址

- 80386以后CPU增加的寻址方式，只适用于32位寻址。
- 操作数的有效地址是变址寄存器的内容乘以指令中指定的比例因子加基址寄存器的内容、再加上位移量之和，即：

  EA＝基址＋（变址×比例因子）＋ 位移量

  例如：　　MOV EAX, TABLE[EBX+EDI*4]

- 适用于数组元素大小为2/4/8字节时二维数组操作，位移量为数组起始地址

三、其它寻址方式

1）隐含寻址

- 指令不指明操作数，但是有隐含规定。如指令DAA，指定对AL中的数据进行十进制调整，结果仍然保留在AL中。

2）I/O端口寻址（只有两种方式）

1)直接端口寻址。端口地址在指令中给出，是一个8位立即数。能访问的端口号为00～0FFH；

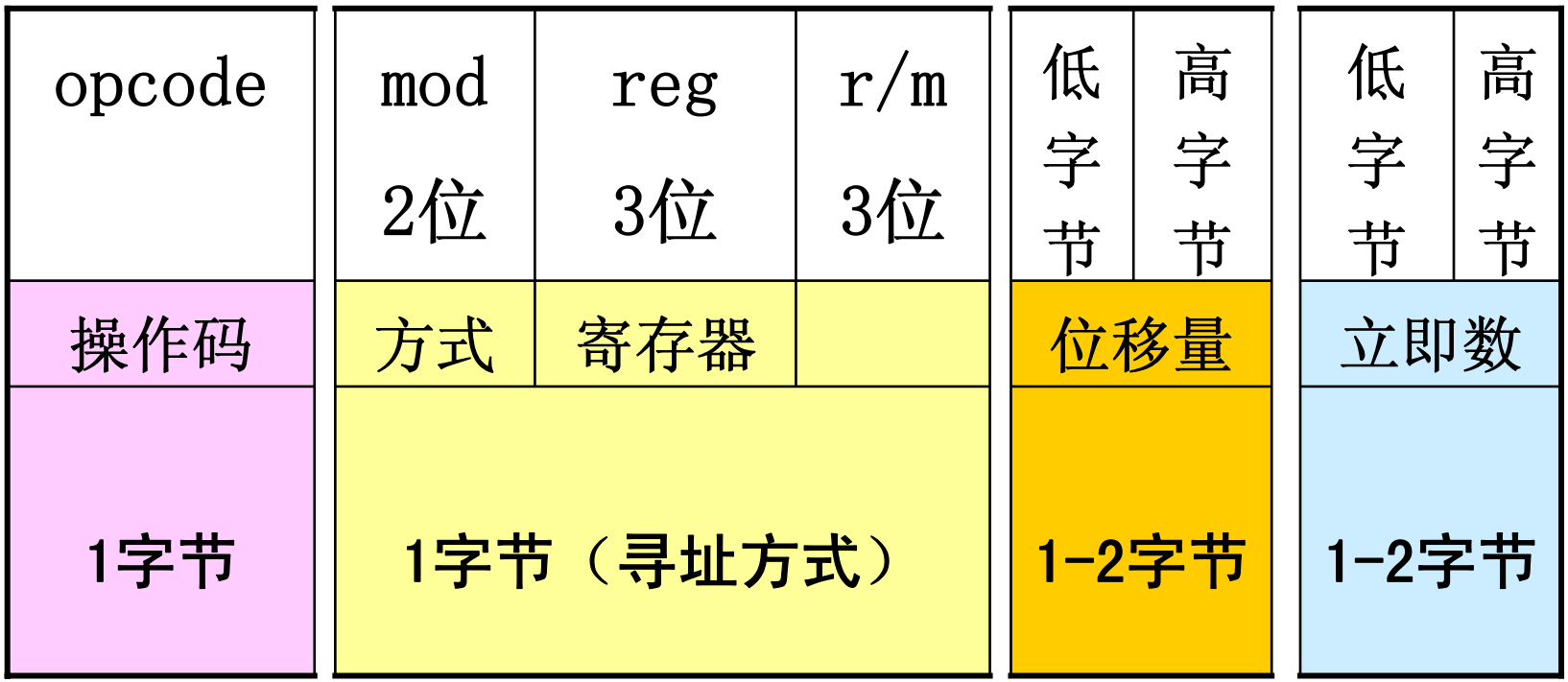
2)间接端口寻址。IO端口地址由DX寄存器提供，能访问的端口地址范围为0000～0FFFFH。

3-2 8086指令的机器码表示方法（#）

一、8086机器语言指令的特点

- 结构：8086指令由操作码域和操作数（地址码）域组成。
- 指令长度：8086的指令长度是可变的，一条指令一般由1－6个字节组成（加上前缀字节，最长可为7字节）。
- 指令机器码的设计思路：对每种基本指令类型设计了一个编码格式，其中规定了各相关含义，对照格式的定义填上不同的寻址方式、数据类型等就得到指令机器码。

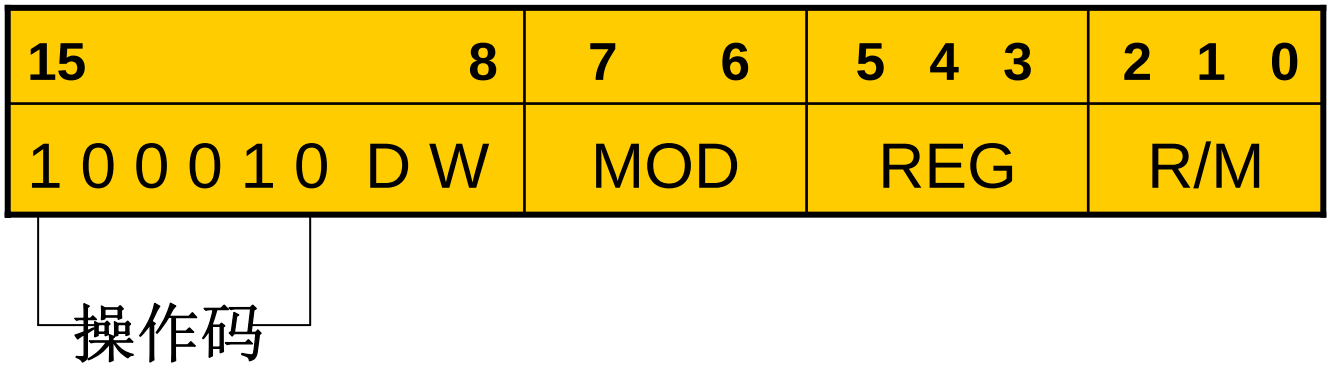
8086 CPU指令格式



二　机器语言指令代码的编制

1、编码格式说明（以数据传送指令为例）：

MOV指令格式：MOV　reg/m，reg/m



注：80x86指令系统中凡是双操作数的指令，两个操作数不能全为存储器，必须有一个是寄存器。

- 操作数1：R(register)
  - REG — R(reg)，参见教材P67表3-1。
  - W — 字长。W=1为字，W=0为字节
  - D —传送方向。D=1，→R；D=0，R→

- 操作数2：R或M(memory)
  - MOD和R/W编码确定寻址方式，由寻址方式明确R和有无位移量，参见教材P67表3-2。

15 8	7 6	5 4 3	2 1 0
1 0 0 0 1 0 D W	MOD	REG	R/M

教材P67表3-1：REG字段对应的寄存器编码

表 3-1　8086 寄存器编码表

REG	W＝1(字)	W＝0(字节)
000	AX	AL
011	BX	BL
001	CX	CL
010	DX	DL
100	SP	AH
111	DI	BH
101	BP	CH
110	SI	DH

REG	段寄存器
01	CS
11	DS
00	ES
10	SS

对于使用段寄存器的指令，**REG**字段只有两位。上右侧是表是**8086CPU**对应的段寄存器编码。

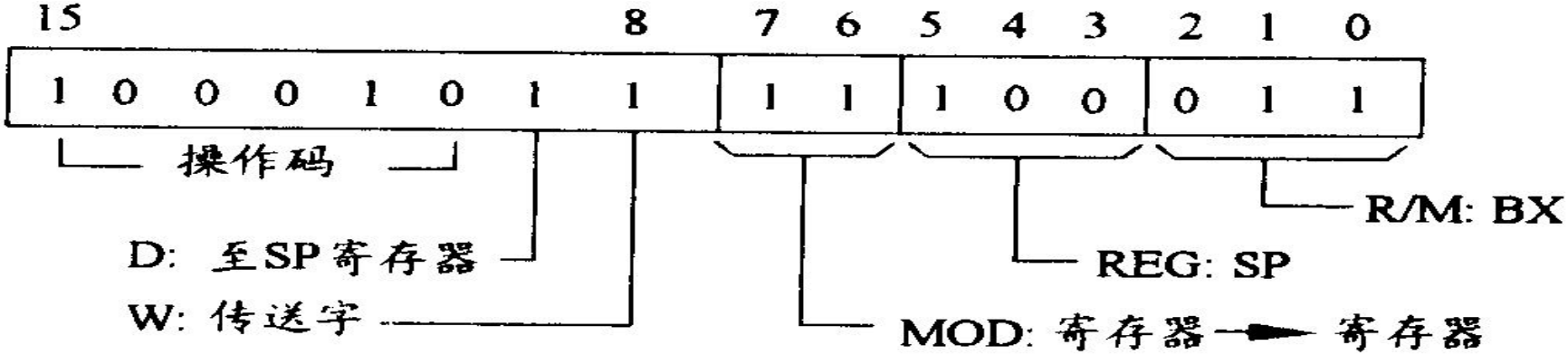
教材P67表3-2。寻址方式以及寄存器和有无位移量

表 3-2　MOD 和 R/M 的编码

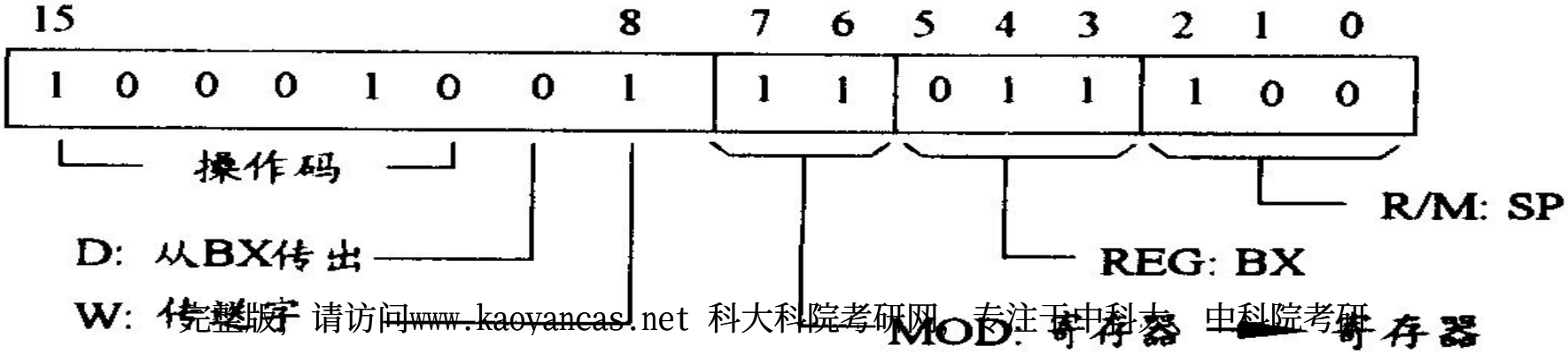
MOD / R/M	00	01	10	11	
				W=0	W=1
000	[BX]+[SI]	[BX]+[SI]+D8	[BX]+[SI]+D16	AL	AX
001	[BX]+[DI]	[BX]+[DI]+D8	[BX]+[DI]+D16	CL	CX
010	[BP]+[SI]	[BP]+[SI]+D8	[BP]+[SI]+D16	DL	DX
011	[BP]+[DI]	[BP]+[DI]+D8	[BP]+[DI]+D16	BL	BX
100	[SI]	[SI]+D8	[SI]+D16	AH	SP
101	[DI]	[DI]+D8	[DI]+D16	CH	BP
110	D16(直接地址)	[BP]+D8	[BP]+D16	DH	SI
111	[BX]	[BX]+D8	[BX]+D16	BH	DI

A）寄存器间传送指令编码示例

例3-18 求指令 MOV SP，BX 的机器码。

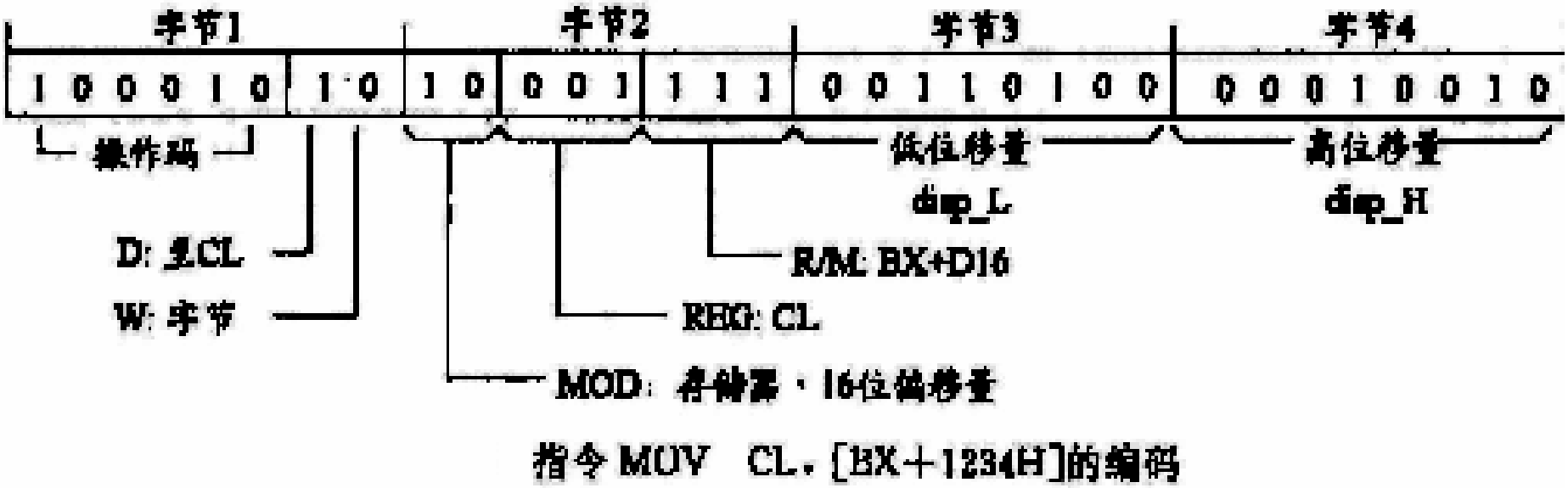


另一种形式



B）寄存器与存储器间传送指令编码示例

例3-19：求指令MOV CL，[BX+1234H]的机器码



指令 MOV CL，[BX+1234H]的编码

3-3 8086指令系统

——IA-32体系结构中的通用指令

- 指令系统：指令系统是一台计算机(μP)能识别和执行的全部指令的集合。
- 每种μP都有独立的指令系统。8086共有6大类87条指令，这些指令系统均向上兼容。

指　令	基本条数	共　计
1）数据传送	14	87
2）算术运算	20	
3）逻辑运算	13	
4）串操作	5	
5）控制转移	28	
6）处理器控制	7	

一、数据传送指令

- 数据传送指令又可以分成4种：

  1）通用数据传送

  2）累加器专用传送（输入/输出数据传送）

  3）目的地址传送

  4）标志寄存器转送

  4种指令总共有14条，见下表… …

数据传送指令（14条）

通用数据传送指令（5条）
MOV、PUSH、POP、XCHG、XLAT
输入输出指令（2条）
IN、OUT
地址目标传送指令（3条）
LEA、LDS、LES
标志传送指令（4条）
LAHF、SAHF、PUSHF、POPF

（一）通用数据传送

1. MOV传送指令

格式：MOV　DST(目的)，SRC(源)

操作：　DST ← SRC

例：　MOV　DS, AX　　;DS ←AX

　　MOV　[DX], AX　　;[DX]←AX

　　MOV　AX, [BX+0060H]

注意：

1、源和目的操作数不能都是存储器

2、目的操作数不能是立即数，也不能是CS寄存器

3、不允许两个段寄存器之间传送数据

4、立即数不能直接送段寄存器

5、不影响标志位

MOV指令允许传送数据的途径（P71）

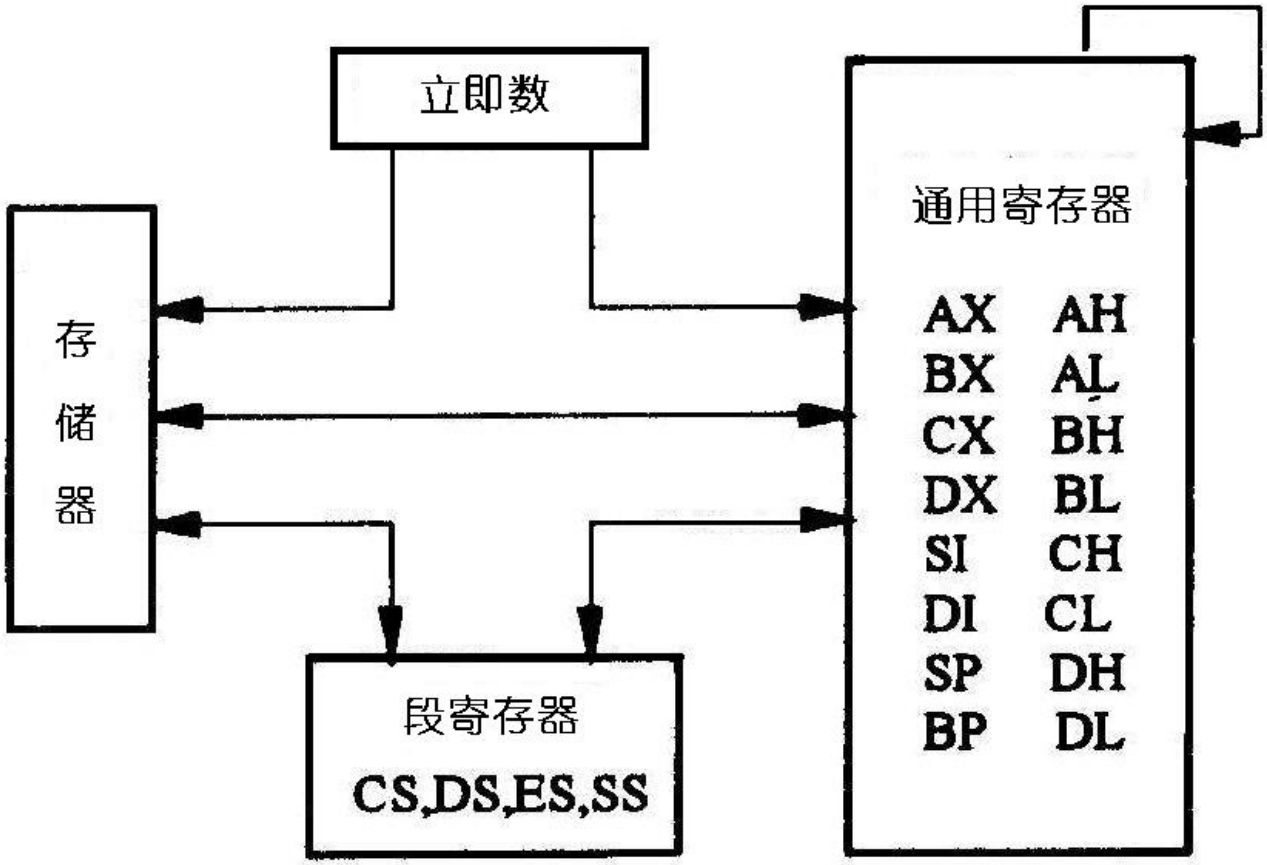


图 3-12 MOV 指令允许传送数据的途径

例：以下各条指令是否正确？

1）MOV ES，1234H

X 立即数不能直接送入段寄存器！

2）MOV 1AH，CL

X 1AH是立即数，不能作为DST！

3）MOV CS，BX

X CS不能作为目的操作数！

4）MOV W1，［BX］

X 两个内存操作数之间不能传送！

5）MOV ES，DS

X 两个段寄存器之间不能传送！

6）MOV AX, BL

X 两个操作数的位数（类型）不相同！

2. 堆栈操作指令

- 格式：PUSH　SRC(源)；
- 操作过程：

  a、SP－1，指示堆栈存放数据位置，存源SRC高8位；

  b、SP再减1，存源SRC的低8位，完成压栈操作。

- 格式：POP　　DST(目的)；
- 操作过程：

  a、将SS:SP指示的栈顶处两个字节弹出到DST操作数中；

  b、SP＋2，指示当前栈顶位置，完成出栈操作。

- 注意：
  - 1、堆栈操作只能以字为单位进行。
  - 2、不能 **POP CS**（即**CS**不能作为**POP**的目的操作数）
  - 3、**SRC**不能是立即数

- 例：设**(SS)=2000H**，**(SP)=00C0H**，执行下述指令后，**SP**的值＝？ 物理地址**PA**＝？

1）PUSH AX ； SP＝SP－2＝00BEH，PA＝200BEH

2）PUSH BX ； SP＝00BE－2＝00BCH，PA＝200BCH

3）POP CX ； SP＝00BC＋2＝00BEH，PA=200BEH

3. 交换指令

- 格式: XCHG　DST, SRC
- 注意：1、DST 与 SRC不能同时为内存单元；
  2、操作数不能为立即数，也不能为段寄存器；
  3、不影响标志位。
- 例：读下列程序段，写出AX，BX的内容。

```
MOV     AX,1234H    ；(AX)＝1234H
MOV     BX,5678H    ；(BX)＝5678H
PUSH    AX          ；
PUCH    BX          ；
POP     AX          ；  (AX)＝？
POP     BX          ；  (BX)＝？
XCHG    AX，BX      ；  (AX)＝？      (BX)＝？
```

4. 表转换指令（换码）

- 格式：XLAT
- 操作：AL←[AL+BX]
- 步骤：

  1) 准备：表首地址 → BX，序号n（表中第n项）→AL

  2) 执行：XLAT；（BX+AL）→ AL

- 用途：

  实现不规则代码转换，如：

  字符扫描码 → ASCII码

  0~9数字 → LED显示器的七段点亮码

  数字 → 控制码

表转换指令应用举例：

利用查表转换方式，写出求5的平方的指令片断：

```
… …
MOV  BX , 2000H  ；  BX指向平方
                 ；  表的首地址
MOV  AL , 5      ；  求5的平方
XLAT             ；  执行换码指令，
                 ；  AL中是结果
```

地址	平方值
2000H	0
2001H	1
2002H	4
2003H	9
	…
	…
2009H	81

（二）输入输出(I/O)指令

1）直接寻址

- 格式:

IN AL，n	OUT n，AL	；传送一个字节
IN AX，N	OUT N，AX	；传送一个字

  - 寻址空间为：0 ~ 255

2）间接寻址

- 格式:

IN AL，DX	OUT DX，AL	；传送一个字节
IN AX，DX	OUT DX，AX	；传送一个字

  - 寻址空间为：0000H ~ FFFFH

（三）地址目标传送指令

1. 取有效地址

- 格式：**LEA　reg16, SRC**
- 功能：取**SRC**的有效地址（偏移量）送到目的寄存器（**reg16**）
- 说明：**LEA**的源操作数必须是存储单元；目的操作数是一个除了段寄存器以外的**16**位寄存器。
- 例：
  - **LEA　AX，[1000H]　；　AX＝1000H**
  - **LEA　SI，　DCD　　；　将DCD的偏移地址送到SI**
  - **LEA　BX，[BP+SI]　；　BX＝(BP)＋(SI)**

2. 传送（双字）指针到DS

- 格式： LDS DST，SRC
- 功能：从SRC指定的存储单元，取出4个字节（通常是2个字节的偏移量和2个字节的段址），前2个字节（低地址）送到DST指定的目的寄存器；后2个字节（高地址）送到DS寄存器。
- 说明：LDS的源操作数必须是存储单元；目的操作数必须是一个除了段寄存器以外的16位寄存器，常用SI寄存器。
- 例：

  DS＝1200H，(12450H)＝F346H，(12452H)＝0A90H

  执行LDS SI，[450H]后，SI＝F346H，DS＝0A90H

3. 传送（双字）指针到ES

- 格式： LES DST，SRC
- 功能：与LDS指令类似。不同之处是后2个字节（段址）送到ES寄存器；目的操作数常用DI寄存器。
- 例：

  DS＝0100H，BX＝0020H，(01020H)＝0300H，(01022H)＝0500H

  执行：LES DI，[BX]后，DI＝0300H，ES＝0500H

（四）标志传送指令

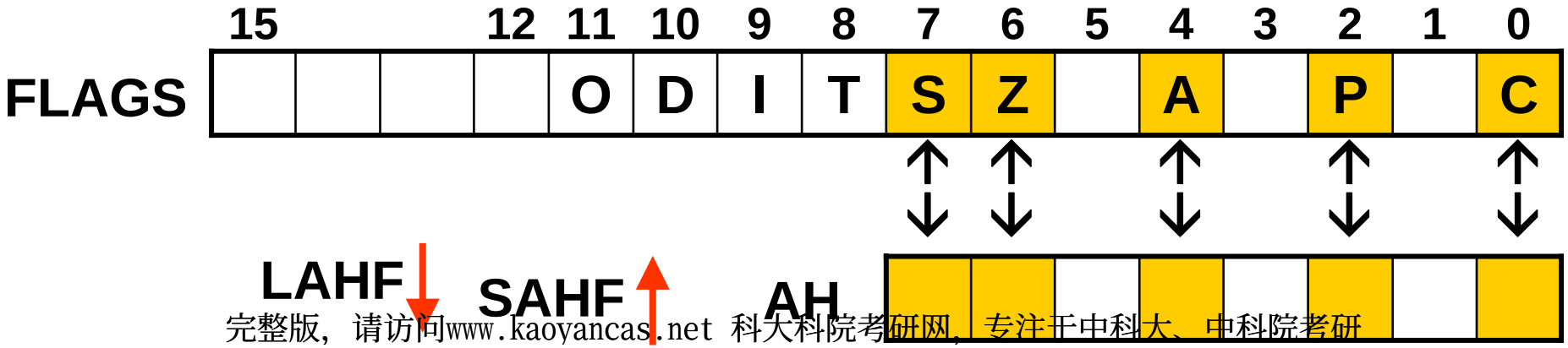
1. 读标志（标志寄存器送到AH）指令

- 格式：LAHF

2. 置标志（AH送到标志寄存器）指令

- 格式：SAHF

- LAHF与SAHF为互逆操作。如下图所示。



3.标志寄存器入栈指令

- 格式：**PUSHF**
- 功能：把标志推入堆栈。
- 步骤：1）**SP ← SP-2**，　2）**PSW(FLAGS)**入栈

4.标志寄存器出栈指令

- 格式：**POPF**
- 功能：从堆栈顶端中弹出标志字到FLAGS中
- 步骤：1）PSW（FLAG）←（[SP+1]、[SP]）
  2）SP ← SP＋2
- 说明：
  1）PUSHF和POPF要成对使用，以保护恢复PSW
  2）修改TF方法，先压栈PSW，再修改堆栈单元中的D8，再出栈。

习题一：P121～

- 1　2）、4）、6）、8）、10）
- 2　1）～6）
- 3　2）、4）、6）、8）、10）
- 5
- 6　1）～14）
- 7
- 8