

# 第3章

## 指令系统 ( I )

## 3.1 指令格式

- 指令：计算机提供给用户的硬件命令；
- 指令系统（集）：CPU所有可执行的指令的集合；
- 机器指令：指能被CPU识别并执行的指令的二进制代码，也称机器码；
- 汇编语言指令或符号指令：帮助记忆！

【例】 ADD CL, BH

在计算机内部的表示：

00000010 11001111

# 指令的构成

操作码

地址码

➤ **操作码**：规定了计算机要执行哪种操作。如传送、运算、移位、跳转等操作，它是指令中不可缺少的组成部分。

用助记符表示：每种指令的操作码：用一个唯一的助记符表示（英文缩写），

对应着一个二进制编码的机器指令。

➤ **地址码**：或称操作数，指令执行的参与者，即各种操作的对象。有些指令不需要操作数，通常的指令都有1个或2个操作数，也有个别指令有3个甚至4个操作数。

操作数的存放：

可以是一个具体的数值（立即数）；可以是存放数据的寄存器、存储器；

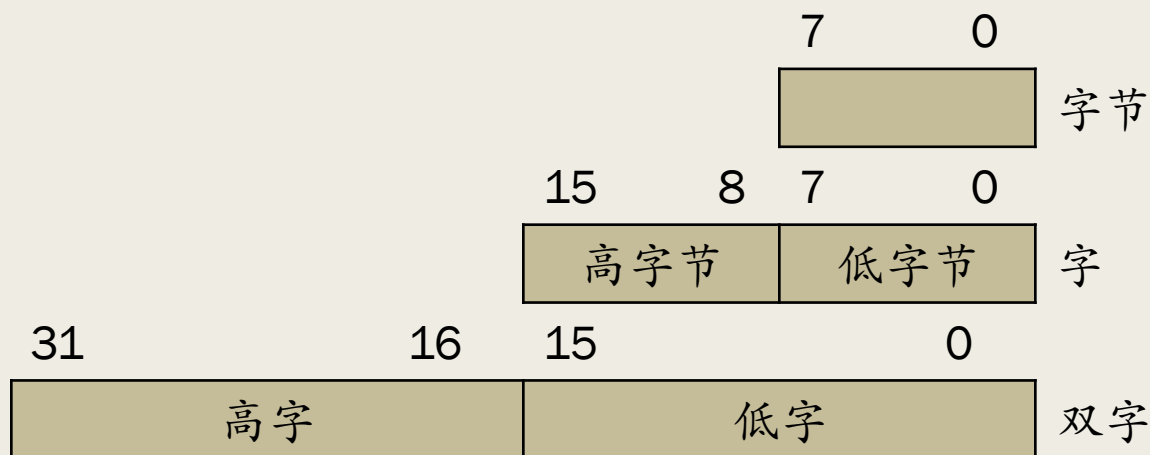
可以用多种方式指明寻址单元在存储器中的地址。

# 8086指令的基本格式

► 一条8086指令的长度在1-6字节，格式如表：

Opcode	Mod 2位	Reg 3位	r/m 3位	低字节	高字节	低字节	高字节
操作码	方式	寄存器	寄存器/ 存储器	位移量		立即数	
1字节	1字节（寻址方式）			1~2字节		1~2字节	

# 8086的基本数据类型



- ❖ 基本数据在内存中作为操作数被引用的顺序为：对于字而言，低字节存放在低地址单元，高字节存放在高地址单元，低地址即为该数据的地址。

## 存储器

**
CDH
ABH
78H
56H
**

2004H

2005H

2006H

2007H

在地址  
2004H中的  
双字为  
5678ABCDH

# 寻址的概念及操作数类别

对于一条指令，要解决2个问题

- ①要指出进行什么操作 ← 指令操作码来表明；
  - ②要指出操作数的来源 ← 操作数的寻址方式；
- 寻址方式——是指CPU在执行指令时寻找操作数或操作数地址的方式，根据寻址方式可方便地访问各类操作数；

计算机中大多数指令采用1个或2个操作数。根据操作数存放的位置，通常有3中表示形式：

- ①立即数
- ②寄存器操作数
- ③存储器操作数

## 3.2 8086寻址方式

8086/8088有24种寻址方式，按其处理数据的类别分为两大类：

### ❖ 与数据有关的寻址方式

①立即数寻址

②寄存器寻址

存储器寻址 ③直接寻址

④寄存器间接寻址

⑤寄存器相对寻址

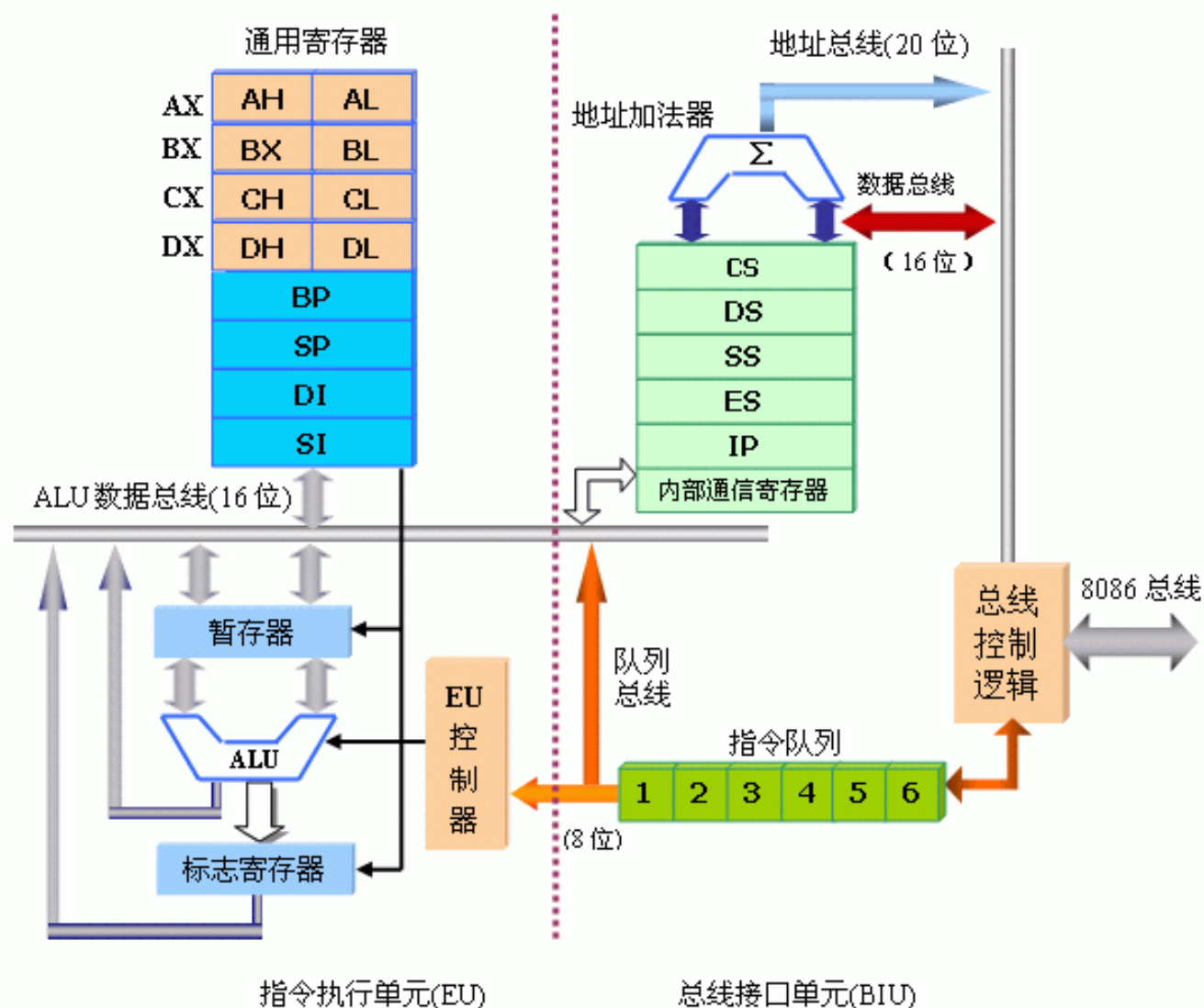
⑥基址加变址寻址

⑦相对基址变址寻址

### ❖ 与I/O端口有关的寻址方式

①直接端口寻址；②间接端口寻址

# 8086 CPU的功能结构



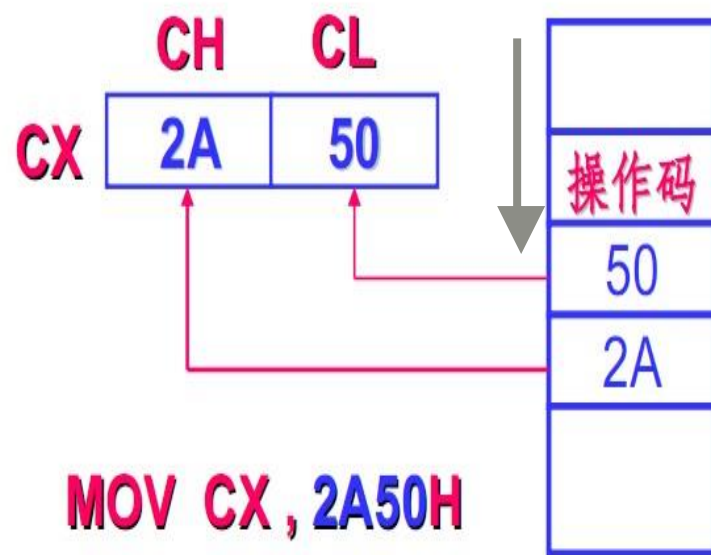
指令不论哪种寻址方式，都存放在代码段，BIU从代码段取出指令到指令队列，EU译码后，根据操作数的来源决定是否再次访问存储器。



# 1. 立即数寻址 (Immediate Addressing)

- ❑ 操作数就包含在指令中，作为指令的一部分，跟在操作码后；
- ❑ 立即数和操作码一起被取入CPU指令队列，在指令执行时不需要访问存储器，不执行总线周期，显著特点是执行速度快；
- ❑ 立即数可以是8位，也可以是16位。如是16位，按“高高低低”原则存放。
- ❑ 主要用于赋初值。

【例】 MOV CX, 2A50H



高字节 高地址  
低字节 低地址

## 注意：

- 1) 立即数可以送到寄存器、一个存储单元(8位)、两个连续的存储单元(16位) 中去；
- 2) 立即数只能是整数，不能是小数、变量或其他类型的数据；
- 3) 立即数只能作源操作数，不能作目的操作数；
- 4) 以A~F打头的数字出现在指令中时，前面必须加数字0。以免与其它符号相混淆(如：0AF22H)。

## 2 寄存器寻址(Register Addressing)

MOV AL, BL

ADD AX, BX

1. 含义：其操作数存放在CPU的内部寄存器中，在指令中直接给出寄存器名。

2. 特点：

16位操作数：通用寄存器：AX、BX、CX、DX、SI、DI、SP和BP。

段寄存器：CS、DS、ES、SS。

8位操作数：寄存器AH、AL、BH、BL、CH、CL、DH和DL。

3. 作用：寄存器之间传递数据。

4. 注意：寄存器可存放源操作数，也可存放目的操作数，还可以两者都用于寄存器寻址方式；源操作数的长度必须与目的操作数一致；

【例】 判断并改正                      MOV DX, AL

×，源操作数和目标操作数类型不一致；

MOV     DX, AX

5. 优点：

□ 此方式属于CPU的内部操作，不需访问总线周期，因而执行速度快。

□ 寄存器号比内存地址短

在编程中，如有可能，尽量使用这种寻址方式的指令。

## 存储器寻址

### 8086提供了5种针对存储器的寻址方式

由于寄存器数量有限，所以程序中的大多数操作数需要从内存中获得；内存的寻址方式有多种，最终都将得到存放操作数的PA；

操作数一般位于数据段、堆栈段或附加段的存储器中，指令中给出的是存储器单元的地址或产生存储器单元地址的信息。

注意：采用存储器寻址的指令中只能有一个存储器操作数，或者是源操作数，或者是目的操作数；指令书写时将存储器操作数地址放在方括号[ ]之中；

### 3. 直接寻址(Direct Addressing)

MOV AX, [0002H]

1. 含义：存储单元的有效地址EA（即：操作数的偏移地址）直接由指令给出。

2. 特点：

机器码中，有效地址存放在代码段中指令的操作码之后，而该地址单元中的数据总是存放在存储器中。须先求出操作数的PA，再从存储器中取得操作数。

$$\text{操作数的PA} = 16 \times \text{DS} + \text{EA}$$

3. 作用：实现对存储单元的读/写操作。

## 4. 注意:

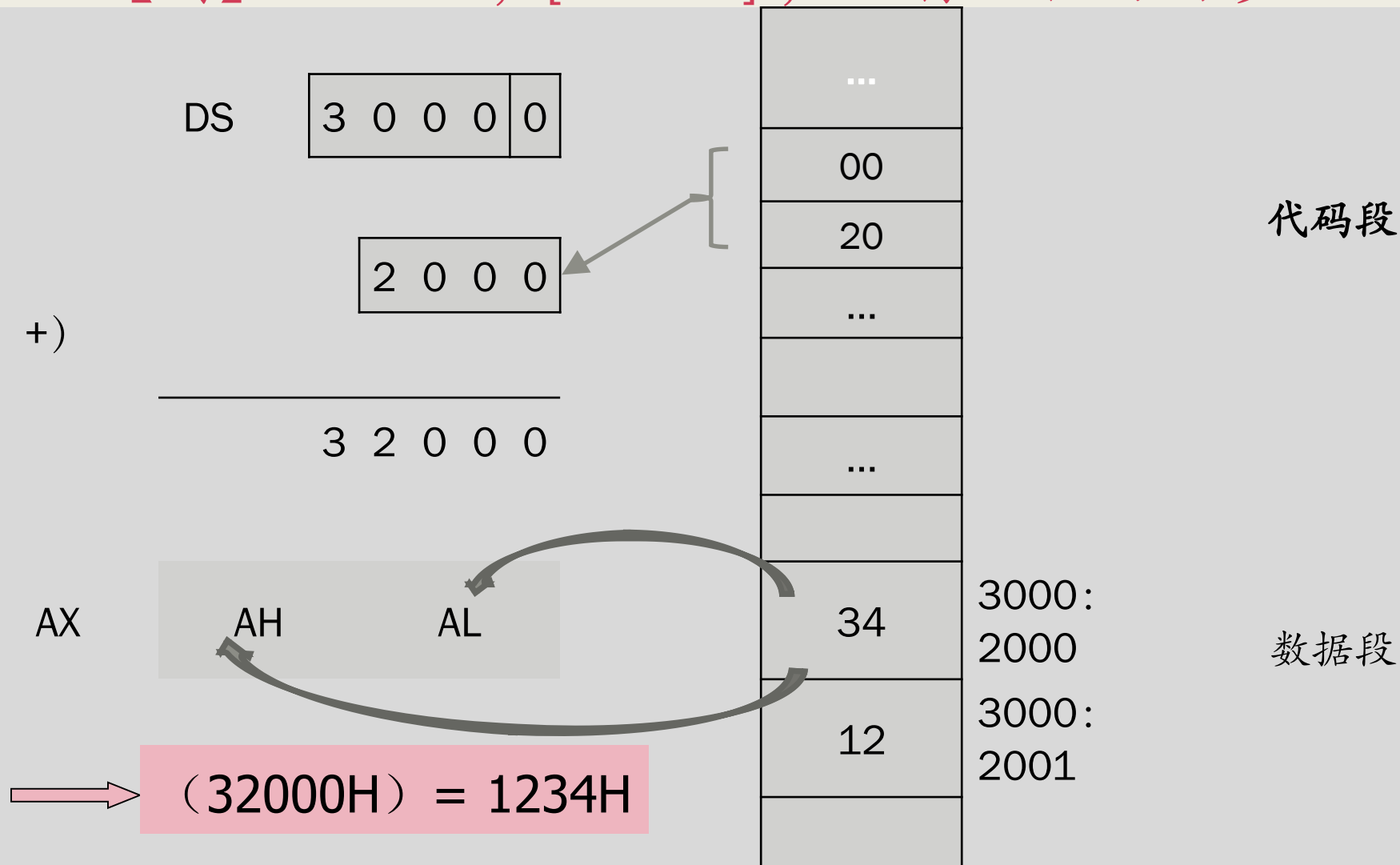
- ❑ DS: 默认的段寄存器;
- ❑ 指令中有效地址的表示: [立即数], 以便与立即数相区别;
- ❑ **关于段超越前缀**: 有效地址前用“:”(称为修改属性运算符)修改运算属性。
- ❑ 可对代码段(CS)、堆栈段(SS)或附加段(ES)寄存器直接寻址。

**【例】** MOV AX, ES: [3000H];

将附加段ES段中偏移地址为3000H和3001H的两单元内容送AX中

$$\text{物理地址} = 16 \times \text{ES} + 3000\text{H}$$

【例】 MOV AX, [2000H] ; 将DS段的偏移



则： AX = 1234H，指令执行过程如图所示。



## 4. 寄存器间接寻址(Register Indirect Addressing)

MOV AX, [BX]

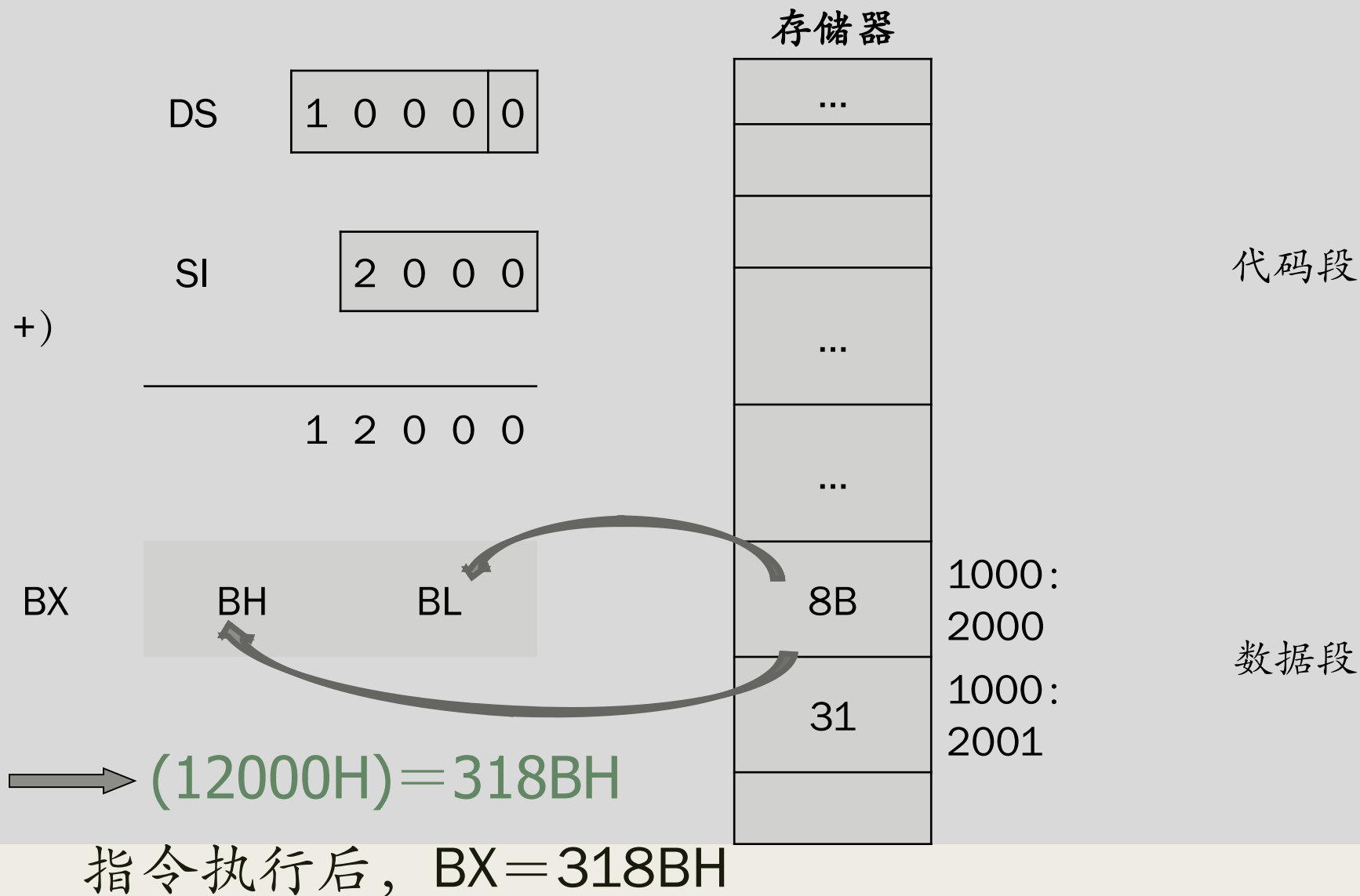
1. 含义：寄存器中值不是操作数本身，而是操作数的有效地址。

2. 特点：可使用4个寄存器：基址寄存器BX、基址指针寄存器BP，变址寄存器SI、DI，即有效地址其实就是其中一个寄存器中的内容  
操作数的  $PA = 16 \times DS + SI / DI / BX$

或  $= 16 \times SS + BP$

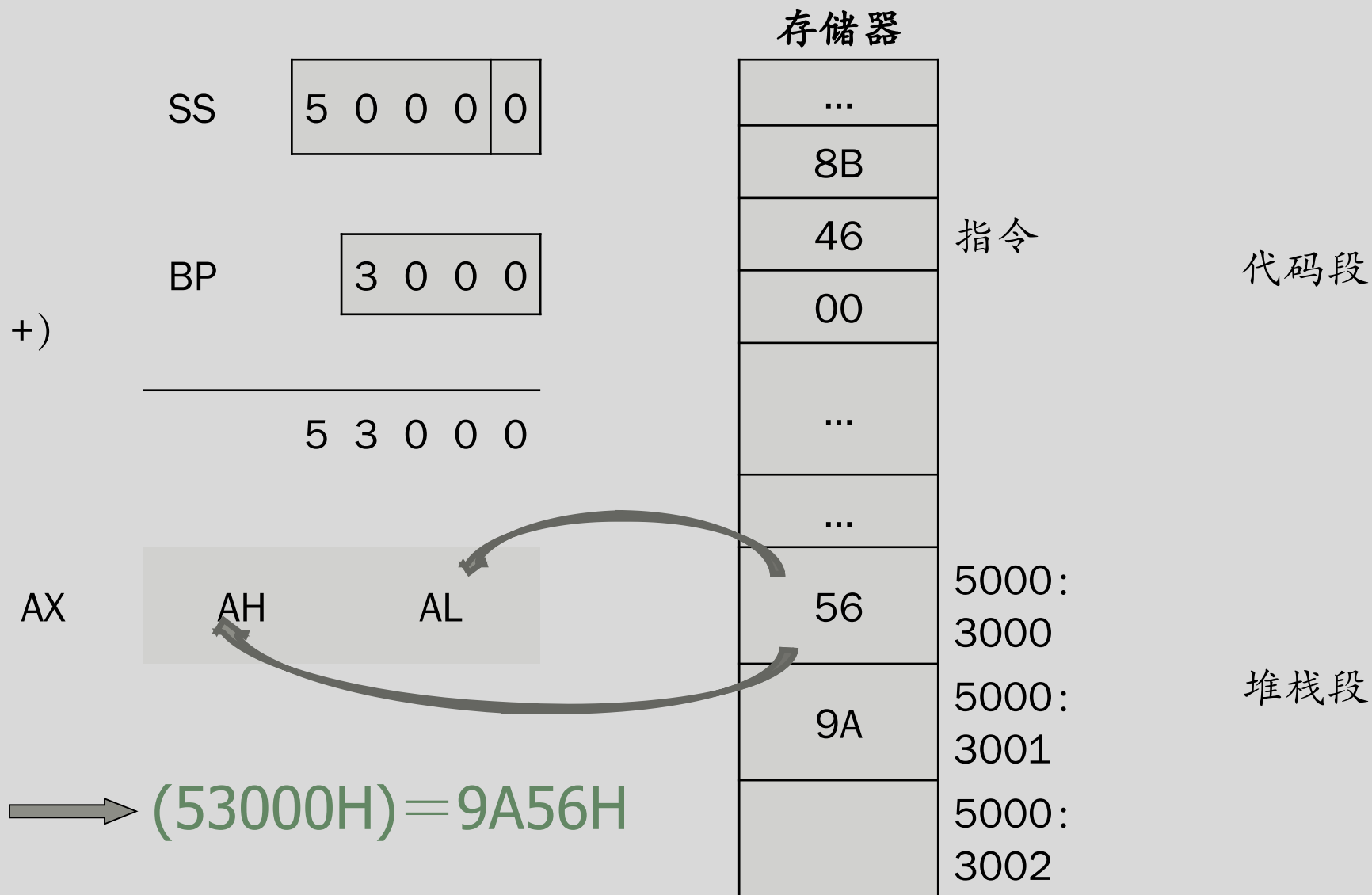
3. 作用：有效地址可以存放在寄存器中。

## 【例】 MOV BX, [SI]



【例】 MOV AX, [BP] ;

将以SS为段



#### 4. 注意:

寄存器名称外必须加方括号，以区别寄存器寻址方式；

**段超越：**前缀用来从默认段以外的段中取得数据；

【例】  $\text{MOV BX, ES:[SI]}$   $\longrightarrow \text{PA} = \text{ES} \times 16 + \text{SI}$

**关于默认段：**

指定寄存器BX、SI或DI，默认操作数存放在数据段DS中；

(DS:BX, SI, DI)

指定寄存器BP，默认操作数存放在堆栈段SS中；

(SS:BP)

## 5. 寄存器相对寻址方式(Register Relative Addressing)/变址寻址(Index Addressing)

MOV AX, [BX + 10H]

1. 含义：操作数的有效地址是一个寄存器内容与指令中给定的8位或16位位移量之和。即

[DS左移4位]+[SI/DI/BX]

PA= +8位偏移量/16位偏移量  
[SS左移4位] + [BP]

2. 特点：

使用：BX、BP、SI、DI。

操作数的物理地址 =  $16 \times DS + BX/SI/DI + C$   
或 =  $16 \times SS + BP$

### 3. 注意:

- ❑ 寄存器名称外必须加**方括号**，位移量可以在括号内，也可以在括号外；
- ❑ **段超越前缀**来从默认段以外的段中取得数据；

**【例】** MOV DH, ES:ARRAY[SI]

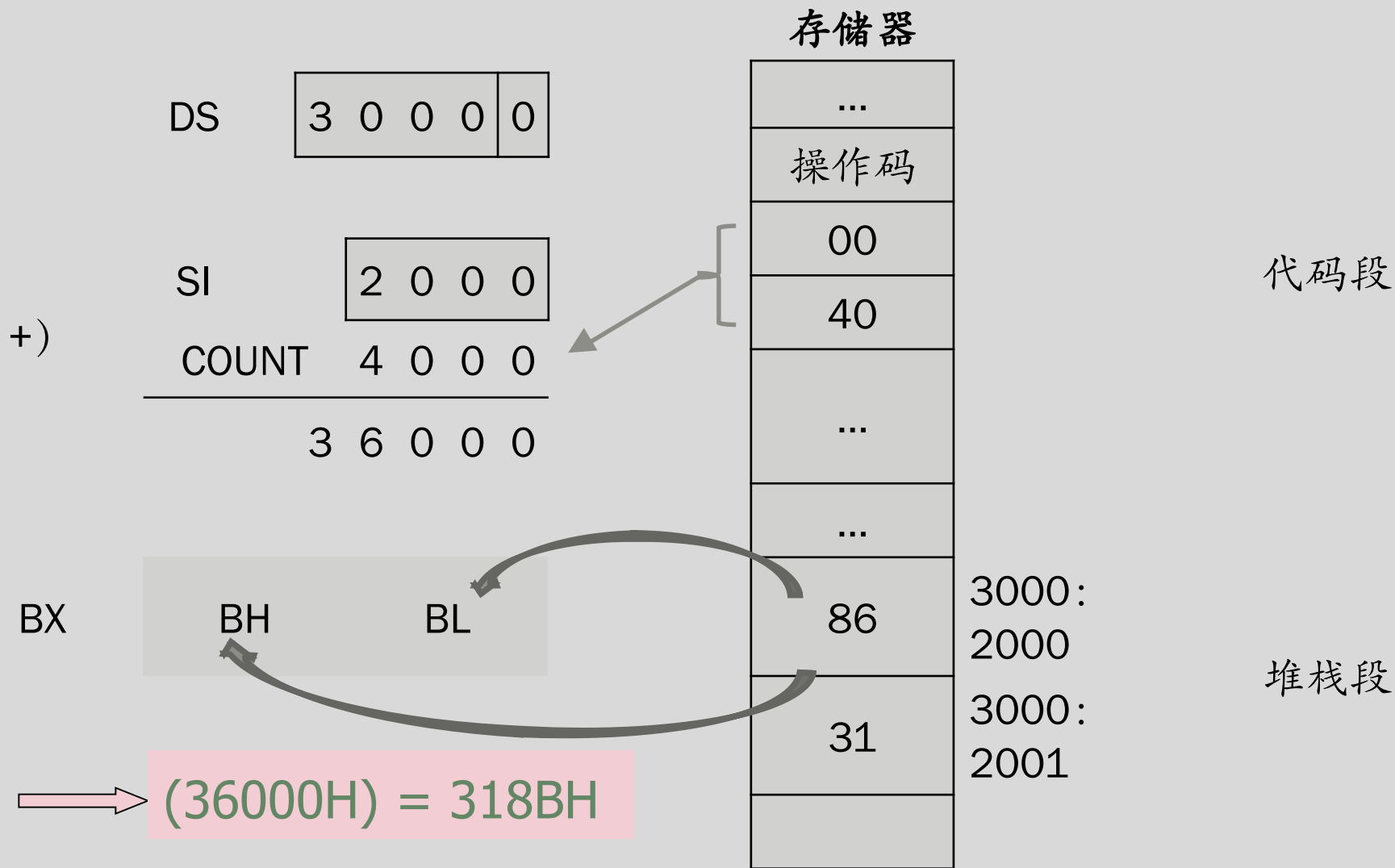
- ❑ 关于**默认段**:

指定寄存器**BX**、**SI**或**DI**，默认操作数存放在数据段**DS**中；

(**DS:BX**, **SI**, **DI**)

指定寄存器**BP**，默认操作数存放在堆栈段**SS**中；  
(**SS:BP**)

【例】 MOV BX, COUNT[SI];



指令执行后: BX = 318BH

## 6. 基址加变址寻址(Based Indexed Addressing)

MOV AX, [BX + SI]

### 1. 含义:

操作数的有效地址是一个基址寄存器(BX/BP)的内容与一个变址寄存器(SI/DI)的内容之和。

### 2. 特点:

使用的寄存器：基址寄存器BX、BP，变址寄存器SI、DI。

$$\begin{aligned}\text{操作数的物理地址} &= 16 \times \text{DS} + \text{BX} + \text{SI} / \text{DI} \\ &= 16 \times \text{SS} + \text{BP} + \text{SI} / \text{DI}\end{aligned}$$

3. 注意：寄存器**SI和DI、BP和BX不能同时**出现在[ ]中。



## 【例】MOV AX, [BX][SI] 或写成

DS      3 0 0 0 0

BX      1 2 0 0

SI      0 5 0 0

+) \_\_\_\_\_

3 1 7 0 0

存储器

...
...
00
20
...
...
CD
AB
...

代码段

3000:  
1700

3000:  
1701

堆栈段

AX

AH

AL

指令执行后: AX = 00001701

## 7. 相对基址加变址寻址

MOV AX, [BX+DI+30H]

### 1. 含义:

操作数的有效地址是1个基址寄存器(BX/BP)的内容与1个变址寄存器(SI/DI)的内容, 在加上指令中给定的8位或16位的偏移量。

$EA = \text{BX/BP的值} + \text{SI/DI的值} + 8/16\text{位位移量}$

### 2. 特点:

使用BX寄存器, 默认的段寄存器是DS; 使用BP寄存器, 默认的段寄存器是SS, 允许段超越。

## 寻址方式总结

带方括号的地址表达式必须遵循下列规则：

- ❑ 立即数可以出现在方括号内，表示直接地址，如[2000H]
- ❑ 只有BX、BP、SI、DI这四个寄存器可以出现在[ ]内，它们可以单独出现，也可以相加，或与常数相加，  
但：BX和BP寄存器、SI和DI寄存器不允许出现在同一个[ ]内。

❑ 方括号表示相加，下面几种写法等价：

6[BX][SI]

[BX+6][SI]

[BX+SI+6]

❑ 不同寄存器对应不同的隐含段基址：

SS:BP;

DS:BX, SI, DI;

物理地址 =  $16 \times \text{相应段基址} + \text{EA}$

$\text{EA} = \text{BX} / \text{BP} + \text{SI} / \text{DI} + \text{DI} / \text{SP}$

(注：可以是单一寄存器、两个寄存器组合、和 DI/SP 组合；DI/SP 也可以为 0)

▲可用段超越前缀修改段基址。

## 段寄存器使用的基本约定

访问存储器的方式	默认	可超越	偏移地址
取指令	CS	无	IP
堆栈操作	SS	无	SP
一般数据访问	DS	CS ES SS	有效地址EA
BP基址的寻址方式	SS	CS ES DS	有效地址EA
串操作的源操作数	DS	CS ES SS	SI
串操作的目的操作数	ES	无	DI

程序只能在CS段；堆栈操作数只能在SS段；目的串操作数只能在ES；  
其他操作虽然也有默认段，但允许段超越。

【例】判断并改正：①POP [AX]  
②MOV AX, DX  
③MOV AX, [SI + DI]

①×，AX不能用于间接寻址，间接寻址只能用BX, BP, SI, DI四个寄存器之一；

POP[BX]

②√

③×，使用存储器寻址时，2个基址寄存器或2个变址寄存器不能同时使用，源操作数寻址方式错，2个寄存器都是变址寄存器；

MOV AX, [SI + BX]

# 课堂练习：课后习题1

	操作码		目的操作数		源操作数
(1)	MOV	DI	寄存器寻址	100	立即寻址
(2)	MOV	CX	寄存器寻址	100[SI]	变址寻址
(3)	MOV	[SI]	寄存器间接寻址	AX	寄存器寻址
(4)	ADD	AX	寄存器寻址	[BX+DI]	基址加变址寻址
(5)	AND	AX	寄存器寻址	BX	寄存器寻址
(6)	MOV	DX	寄存器寻址	[1000]	直接寻址
(7)	MOV	BX	寄存器寻址	[BP+DI+100]	相对基址加变址寻址
(8)	PUSHF		寄存器间接寻址		寄存器寻址
(9)	SUB	[1050]	直接寻址	CX	寄存器寻址
(10)	AND	DH	寄存器寻址	[BP+4]	变址寻址

## 4. I/O端口寻址

- 8086的I/O端口采用独立编址，有64个字节端口或32个半字节端口，采用专门的输入输出指令
- 操作数在外设端口中，I/O寻址方式有直接寻址和间接寻址两种

### 4.1 直接端口寻址

是在指令中直接给出要访问的端口地址，一般采用2位16进制数表示，也可以用符号表示，可访问的端口范围0 ~ 0FFH，即最大256个

**【例】** IN AL, 30H;

从地址为30H的I/O端口中的内容取出送寄存器AL



## 4.2 间接端口寻址

若I/O端口地址超过8位，必须用间接端口寻址。  
外设端口的16位地址必须存放在DX中，即DX值表示I/O端口地址，可访问的端口数为0 ~ 65535，  
即最大为65536；

I/O端口地址在8位以内时，也可用间接寻址。

**【例】** `MOV DX, 285H` ；

将端口地址285H送到DX寄存器

`IN AL, DX` ；

把地址为285H的端口中的内容取出送AL

`OUT DX, AL` ；

将AL中的内容输出到DX指定的端口

# 8086指令系统概述

❖ 8086指令系统包含133条基本指令，这些指令与寻址方式结合，再加上不同的数据形式——有的为字处理，有的为字节处理。

这些指令按功能可分为6类：

- 1、数据传送类
- 2、算术运算类
- 3、逻辑运算与移位类
- 4、串操作类
- 5、控制转移类
- 6、标志操作和处理器控制类

## 指令遵循的规则

- ❖ 两个操作数的类型要一致
- ❖ 两个操作数不能都在存储器
- ❖ 目的操作数不能是立即数和CS
- ❖ 单操作数指令中的操作数不能是立即数

## 3.3 数据传送类指令

### 概述

- 数据传送是计算机中最基本、最重要的一种操作
- 可实现CPU寄存器之间、寄存器与存储器之间、累加器与I/O端口之间、立即数到寄存器或存储器的字节或字的传送

分为四种：

- 通用数据传送指令
- 累加器专用数据传送指令
- 地址-目的数据传送指令
- 标志数据传送指令

通用数据传送指令		地址目标传送指令	
<b>MOV</b>	字节或字的传送	<b>LEA</b>	装入有效地址
<b>PUSH</b>	如栈指令	<b>LDS</b>	装入数据段寄存器
<b>POP</b>	出栈指令	<b>LES</b>	装入附加段寄存器
<b>XCHG</b>	交换字或字节	标志传送指令	
<b>XLAT</b>	表转换	<b>LAHF</b>	标志寄存器低字节装入AH
输入输出指令		<b>SAHF</b>	AH内容装入标志寄存器低字节
<b>IN</b>	输入	<b>PUSHF</b>	标志寄存器入栈指令
<b>OUT</b>	输出	<b>POPF</b>	出栈，并送入标志寄存器

## 特点:

- 传送指令把源操作数送到目标操作数中，源操作数不变；
- 除标志寄存器传送指令**SAHF**、**POPF**外，所有数据传送指令均不影响标志位；
- 在传送时，目标操作数和源操作数数据类型**必须一致**，**两操作数不能同时是存储操作数**；
- 代码段**CS**和立即数不能为目标操作数；
- 目标操作数和源操作数不能同时为段寄存器；
- 立即数不能直接传送到段寄存器；

## 3.3.1 通用数据传送指令

### MOV传送指令(Move)

□ 指令格式:

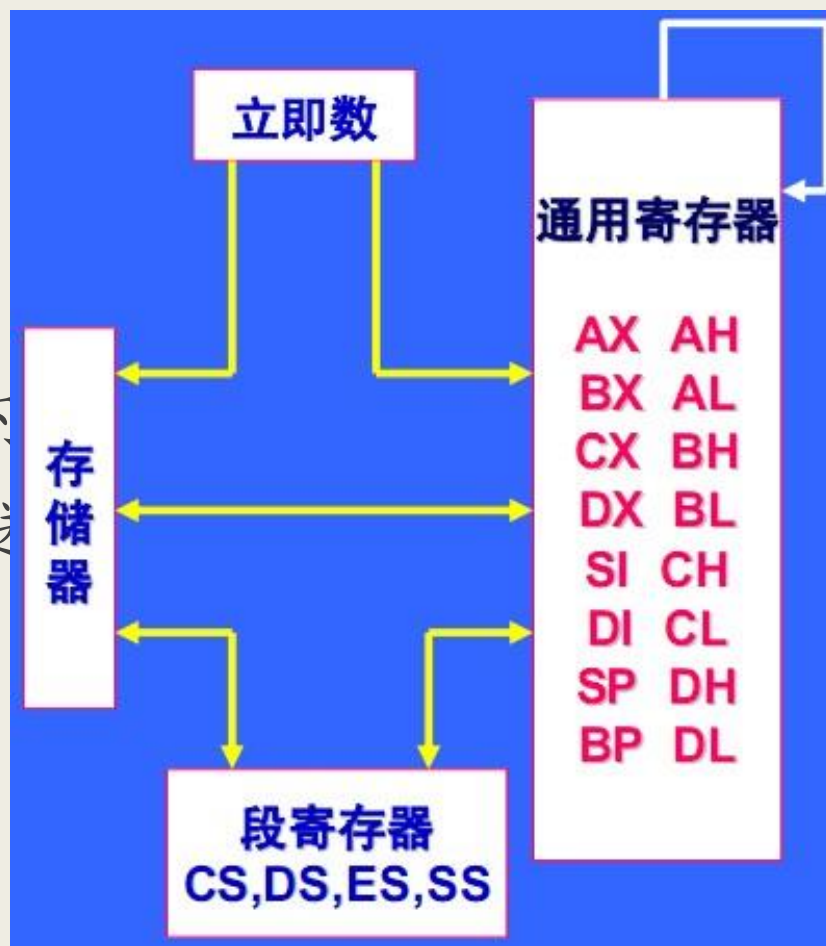
**MOV** 目的(dst), 源(src)

□ 指令功能: 实现CPU的内部寄存器与存储器间的数据传送

□ 标志位: 不影响



**注意: MOV也并非任意传送**



## MOV指令举例

指令格式	使用说明
MOV AL, BL	；通用寄存器之间传送字节数据
MOV DS, AX	；通用寄存器与段寄存器之间传送数
MOV AX, 0FF3B	；立即数传送到通用寄存器
MOV AL, [1000H]	；通用寄存器与存储器之间传送数据
MOV [BP+DI], ES	；段寄存器与存储器之间传送数据
MOV AX, DATA_SEG MOV DS, AX	；段地址必须通过寄存器如AX寄存器送到DS寄存器。



## 指令格式

MOV AL, 'E'

MOV BX, OFFSET TABLE

MOV AX, Y[BP][SI]

## 使用说明

；把立即数（字符E的ASCII码）送到AL寄存器。

；把TABLE的偏移地址（而非内容）送到BX寄存器。其中OFFSET为属性操作符，表示把后面符号地址的值（不是内容）作为操作数。

；把地址为  $16 * (SS) + (BP) + (SI) + \text{位移量}Y$  的存储单元的内容送给AX寄存器。

(1)OPR1 与OPR2位数匹配;

MOV AL , BX

×      MOV AL , BL    ✓

(2)立即数只能作源操作数，不能作目的操作数。

MOV 5H , AL

×      MOV AL , 5H    ✓

(3) 不能直接在两个存储单元之间进行数据交换，必须用内部寄存器作为过渡传送数据。

MOV [3000H], [2000H]

×      MOV AL, [2000H]

MOV [3000H], AL    ✓

MOV [BX] , [SI]

×      MOV BX, SI      ✓

(4) 不能用立即数直接对段寄存器赋值，必须用内部寄存器或存储单元作为过渡。

也不允许在两段寄存器之间直接传送数据，但可以由段寄存器向内部寄存器或存储单元传送数据。

MOV DS , 2000H

× MOV DX , 2000H

MOV DS , DX ✓

(5) CS和IP不能作为目的操作数。

(6) MOV指令不影响标志位。

## MOV传送指令注意点:

- ❑ 指令中至少要有一项明确说明传送的是字节还是字;
- ❑ IP寄存器不能用作源操作数或目的操作数;
- ❑ 立即数和CS寄存器不能用作目的操作数;
- ❑ 除了源操作数为立即数的情况外, 两个操作数中必有一个是寄存器, 但不能都是段寄存器; 即MOV指令不能在两个存储单元之间直接传送数据, 也不能在两个段寄存器之间直接传送数据。

## 课堂练习：课后习题2

- `MOV AX, 2000H` → 源操作数是立即寻址；  
`MOV AX, [2000H]` → 源操作数是直接寻址；
- `MOV AX, 2000H` → 功能是把立即数  
2000H送入AX中；  
`MOV AX, [2000H]` → 功能是把内存2000H  
单元与2001H单元的内容取出送入AX中；
- 两者的机器代码不同，执行速度不同，前者执行时间快，后者执行时间慢。

## 3.3.2 交换指令

**XCHG reg, mem/reg**

数据交换方向:



**注意点:**

- ❑ 交换可以在寄存器之间、寄存器与存储器之间进行;
- ❑ 段寄存器中的内容不能交换;
- ❑ 不能直接交换两个存储单元中的内容;
- ❑ 不能使用CS、IP作为操作数; 指令不影响标志位;
- ❑ 寄存器和累加器AX的内容互换时, 指令的机器代码为1字节, 执行速度较快;

【例】 设 $AX = 2000H$ ， $DS = 3000H$ ， $BX = 1800H$ ， $(31A00H) = 1995H$ ，则执行指令 **XCHG AX, [BX+200H]** 后，结果如何？

解：把内存中的一个字与AX 中的内容进行交换。

源操作数物理地址  $= 3000 \times 10H + 1800H + 200H$   
 $= 31A00H$ ，

指令执行后： **$AX = 1995H$ ， $(31A00H) = 2000H$**

【例】 判断并改正

XCHG     DX, 0FFFFH

×

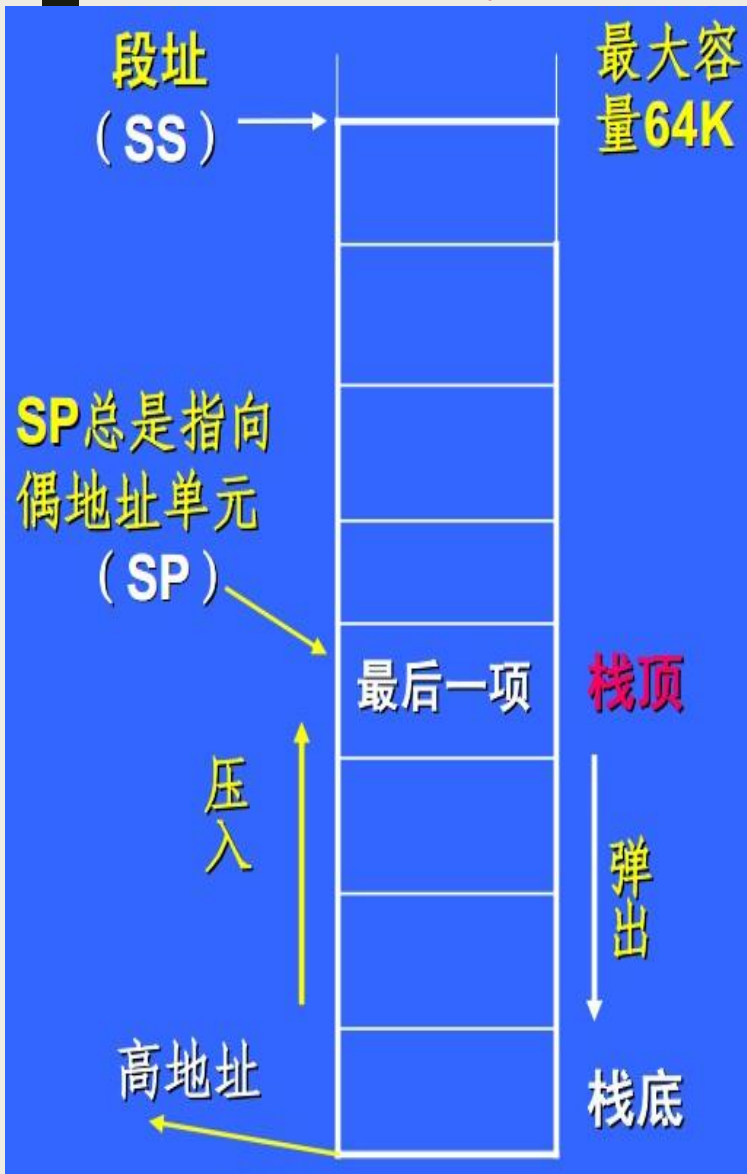
XCHG指令不允许立即数做它的操作数；

MOV     CX, 0FFFFH

XCHG    DX, CX



### 3.3.3 堆栈指



- ❖ 堆栈是以“后进先出”方式工作的一个存储区(内存区)，堆栈操作必须在堆栈段中进行，其段地址由堆栈段寄存器SS确定。
- ❖ 堆栈只有一个出入口，一端固定，另一端浮动，固定一端叫**栈底**，浮动一端叫**栈顶**（低地址端）。所以只有一个堆栈指针寄存器SP，SP的内容在任何时候都指向当前的栈顶，SP的内容是栈顶的偏移地址。
- ❖ 进栈和出栈指令都必须根据当前SP的内容来确定进栈或出栈的单元，必须及时修改SP的值，使SP的内容指向当前的栈顶。

堆栈只有两种基本操作：进栈和出栈，对应两条指令PUSH和POP

注意点：

- ❑ 可以是16位通用寄存器、段寄存器、存储器中的数据字；
- ❑ 立即数不能作为PUSH的源操作数
- ❑ CS不能作为POP的目的操作数；
- ❑ 堆栈操作的单位是字

# 进栈指令PUSH (Push word onto the stack)

格式: PUSH OPRD

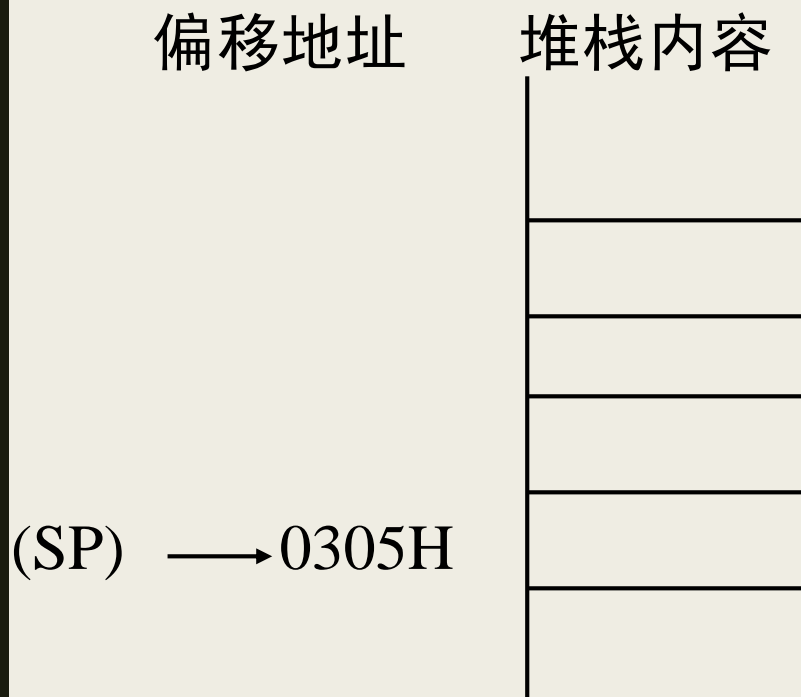
执行操作:	(SP)	←	(SP) - 1
	[(SP)]	←	(OPRD的高字节)
	(SP)	←	(SP) - 1
	[(SP)]	←	(OPRD的低字节)

指令执行后,  $(SP) = (SP) - 2$

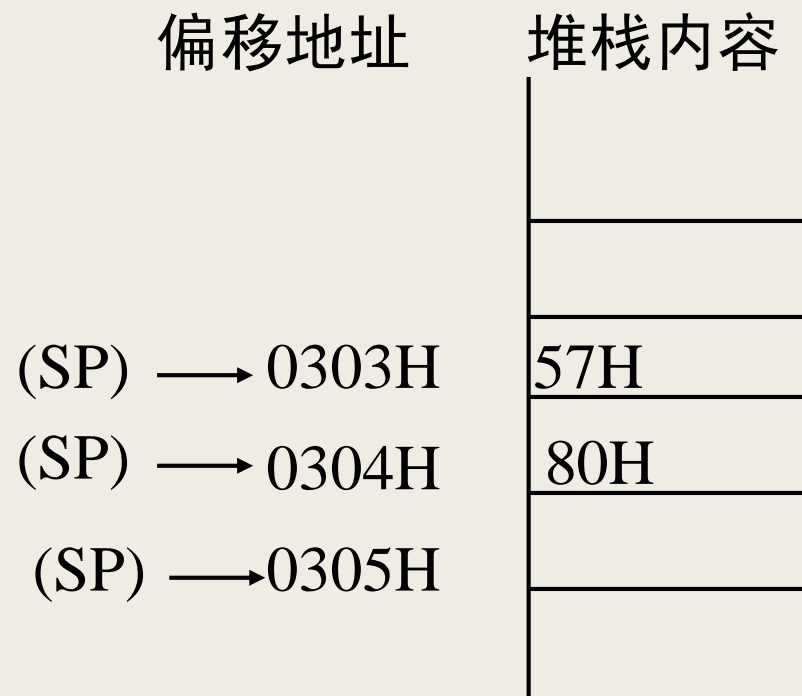
说明: OPRD可以使用除立即数以外的任何一种寻址方式

## 【例】PUSH AX

已知指令前 $(SP) = 0305H$ ,  
 $(AX) = 8057H$ 。指令的执行过程如下：



(1) 入栈前的情况  
 $(SP) = 0305H$




(2) PUSH AX后的情况  
 $(SP) = 0303H$

# 出栈指令POP(Pop word from the stack)

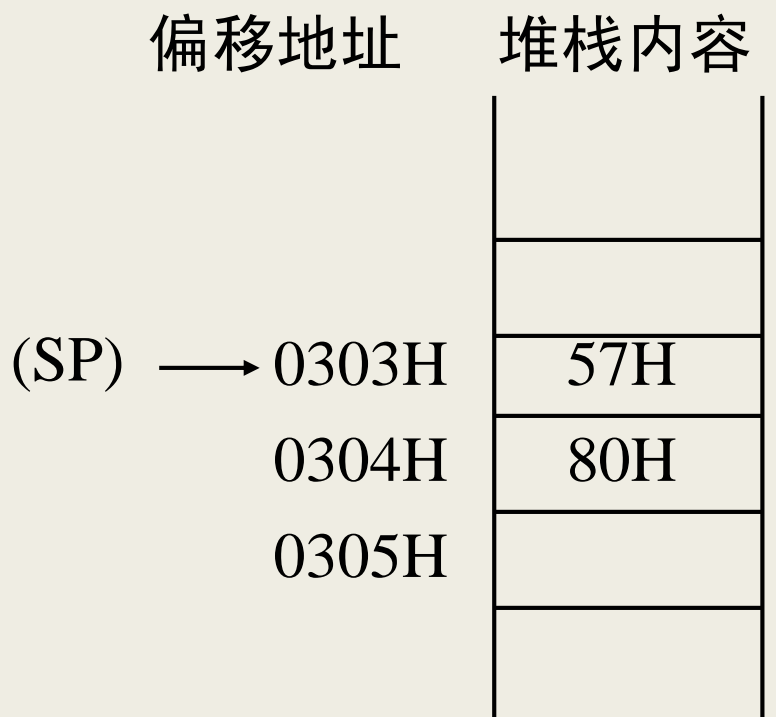
格式: POP OPRD

执行操作: (OPRD的低字节)  $\leftarrow$  [(SP)]  
(SP)  $\leftarrow$  (SP) + 1  
(OPRD的高字节)  $\leftarrow$  [(SP)]  
(SP)  $\leftarrow$  (SP) + 1

 **注意:** POP指令不允许使用立即数和CS寄存器

## 【例】POP BX

已知执行指令前， $(SP) = 0303H$ ， $(BX) = 1234H$ 。指令的执行过程如下：



(1) POP BX前的情况  
 $(SP) = 0303H$



(2) POP BX前的情况  
指令执行后， $(SP) = 0305H$ ， $(BX) = 8057H$

## 【例】判断并改正

PUSH            BL

×

堆栈指令操作数只能以字为单位进行，BL是一个字节；

PUSH    BX

## 3.3.4 地址传送指令

① 有效地址传送指令：LEA

② 指针传送指令：LDS和LES

三条指令的功能是将存储单元的地址送入指定的寄存器中，用来传送操作数的段地址和偏移地址。



# 1) LEA取有效地址指令(Load Effective Address)

- ❑ 指令格式：LEA 目的，源
- ❑ 指令功能：将指定存储器操作数的16位偏移地址送入指定的8个通用的16位寄存器之一中；
- ❑ 操作数要求：源操作数必须是存储单元，目的操作数必须是一个除段寄存器之外的16位寄存器。
- ❑ 使用时要注意与MOV指令的区别，MOV指令传送的一般是元操作数中的内容而不是地址。

## 【例】判断并改正 LEA SI, BX

×

LEA指令取源操作数的EA，源操作数必须是存储器操作数而不是寄存器BX，目标操作数必须是通用寄存器之一；

LEA      SI, [BX]

【例】 假设SI=1000H, DS=5000H,  
(51000H) = 1234H

LEA BX, [SI] ;

执行完该指令后, BX=1000H (送偏移地址)

MOV BX, [SI] ;

执行完该指令后, BX=1234H (送内容)

## 2) LDS将双字指针送到寄存器和DS指令(Load Pointer using DS)

- ❑ 指令格式：LDS 目的，源
- ❑ 指令功能：从源操作数指定的存储单元中，取出一个变量的4字节(32位)地址指针，送进一对目的寄存器。其中前两个字节(表示变量的偏移地址)送到指令中指定的目的寄存器中，后两个字节(表示变量的段地址)送入DS寄存器。
- ❑ 操作数要求：源操作数必是存储单元，该单元开始的连续4个字节存放一个变量的地址指针。  
目的操作数必须是16位寄存器，常用SI寄存器，但不能用段寄存器。

【例】 设  $DS = 1200H$ ,  $(12450H) = F346H$ ,  
 $(12452H) = 0A90H$ , (32位连续地址)

执行指令  $LDS\ SI, [0450H]$  后,  
 $SI = F346H$ ,  $DS = 0A90H$

注:

源操作数的  $PA = DS \times 10H + 450H$   
 $= 12450H$

### 3) LES将双字指针送到寄存器和ES指令(Load Pointer using ES)

- 指令格式：LES 目的，源
- 指令功能：与LDS指令的操作基本相同，**不同的是：**要将源操作数所指向的存储单元里存放的地址指针中的段地址部分送到ES寄存器中，而不是DS寄存器，**目的操作数常用DI寄存器。**

【例】 设  $DS = 0100H$ ,  $BX = 0020H$ ,  
 $(01020H) = 0300H$ ,  $(01022H) = 0500H$ 。

(32位连续地址)

执行指令  $LES\ DI, [BX]$  后,

$DI = 0300H$ ,  $ES = 0500H$

注: 源操作数的物理地址  $= DS \times 10H + BX$   
 $= 0100H \times 10H + 0020H$   
 $= 01020H$

## 课堂练习：课后习题4

已知：DS = 4000H, BX = 0800H, [40800H]  
= 05A0H, [40802H] = 2000H;

LDS SI, [BX] → SI = 05A0H, DS = 2000H

LES DI, [BX] → DI = 05A0H, ES = 2000H