



## 第2章 8086指令系统



## 第2章 8086指令系统

◇ 8086指令系统包括6大类、133种基本指令

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

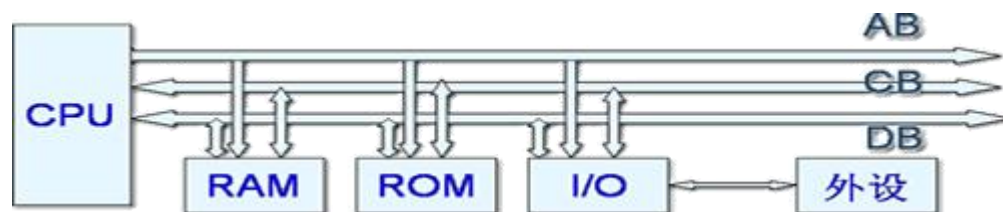
⑥处理机控制类

◇ 通过寻址方式的变化与数据形式（字节、字型）的组合，可构成上千条指令。

## 2.3 数据传送类指令

- ◇ 数据传送是计算机中最基本、最重要的操作, 传送指令也是程序中最常使用的一类指令。
- ◇ 传送指令把数据从一个位置传送到另一个位置。
- ◇ 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**



## 2.3.1 通用数据传送指令

- ◇提供方便灵活的通用传送操作
- ◇有3条指令
  - ◆**MOV (MOVE)**
  - ◆**XCHG (EXCHANGE)**
  - ◆**XLAT (TRANSLATE)**



MOV

XCHG

XLAT

# 1. 传送指令MOV (move)

- ◇ 功能：把一个字节或字操作数从源地址传送至目的地址，用于程序中的变量赋值。

MOV reg/mem,imm

举例

MOV reg/mem/seg,reg

举例

MOV reg/seg,mem

举例

MOV reg/mem,seg

举例

； 段寄存器送寄存器或主存

演示

## MOV指令:立即数传送实例

- ◇ `mov cl,4` ; `cl`←4, 字节传送
- ◇ `mov dx,0ffh` ; `dx`←00ffh, 字传送
- ◇ `mov si,200h` ; `si`←0200h, 字传送
- ◇ `mov bvar,0ah` ; 字节传送  
; 假设**bvar**是一个字节变量, 定义如下: `bvar db 0`
- ◇ `mov wvar,0bh` ; 字传送  
; 假设**wvar**是一个字变量, 定义如下: `wvar dw 0`

以字母开头的常数要有前导0

字节操作还是字操作

## MOV指令:寄存器传送实例

- ◇ `mov ah,al` ;  $ah \leftarrow al$ , 字节传送
- ◇ `mov bvar,ch` ;  $bvar \leftarrow ch$ , 字节传送
- ◇ `mov ax,bx` ;  $ax \leftarrow bx$ , 字传送
- ◇ `mov ds,ax` ;  $ds \leftarrow ax$ , 字传送
- ◇ `mov [bx],al` ;  $[bx] \leftarrow al$ , 字节传送

寄存器具有明确的字节和字类型

## MOV指令:存储器传送实例

- ◇ `mov al,[bx]` ;  $al \leftarrow ds:[bx]$
- ◇ `mov dx,[bp]` ;  $dx \leftarrow ss:[bp+0]$
- ◇ `mov dx,[bp+4]` ;  $dx \leftarrow ss:[bp+4]$
- ◇ `mov es,[si]` ;  $es \leftarrow ds:[si]$

不存在存储器向存储器的传送指令

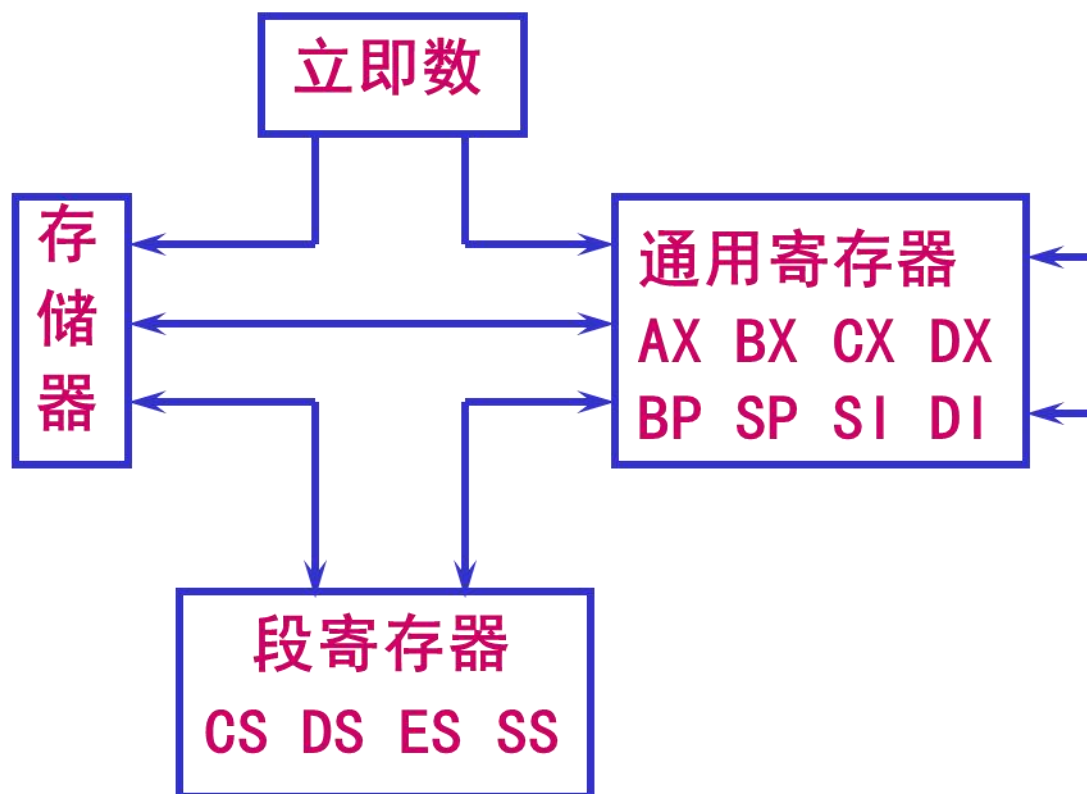


## MOV指令:段寄存器传送实例

- ◇ `mov [si],ds`
- ◇ `mov ax,ds` ;  $ax \leftarrow ds$   
`mov es,ax` ;  $es \leftarrow ax \leftarrow ds$

对段寄存器的操作不灵活

# MOV指令传送功能图解



MOV指令也并非任意传送!

# 非法指令

## 非法指令的主要现象：

- ◇ 两个操作数的类型不一致
- ◇ 无法确定是字节量还是字量操作
- ◇ 两个操作数都是存储器
- ◇ 段寄存器的操作有一些限制

## 非法指令:(1)两个操作数类型不一致

- ◇ 双操作数指令中，目的操作数和源操作数必须具有一致的数据类型，或者同为字量，或者同为字节量，否则为非法指令。

MOV AL, 050AH

非法指令，修正：mov ax,050ah

MOV SI, DL

非法指令，修正：

mov dh,0  
mov si,dx

## 非法指令:(2)无法确定是字节量还是字量操作

✧当无法通过任一个操作数确定操作类型时，需要利用汇编语言的操作符显式指明。

MOV [BX+SI], 255 ; 非法指令，修正：

; mov **byte ptr** [bx+si],255

; byte ptr 说明是字节操作

; mov **word ptr** [bx+si],255

; word ptr 说明是字操作

## 非法指令:(3)两个操作数都是存储器

8086指令系统除串操作指令外，不允许两个操作数都是存储器操作数。

**MOV buf2, buf1** ; 非法指令

修正:

; 假设buf1和buf2是两个**字**变量

; mov ax,buf1

; mov buf2,ax

; 假设buf1和buf2是两个**字节**变量

; mov al,buf1

; mov buf2,al

## 非法指令:(4)段寄存器操作的限制

8086指令系统中，能直接对段寄存器操作的指令只有MOV等个别传送指令，并且不灵活。

**MOV DS, ES**

；非法指令，修正：

； mov ax,es

； mov ds,ax

**MOV DS, 100H**

；非法指令，修正：

； mov ax,100h

； mov ds,ax

**MOV CS, [SI]**

；非法指令

；指令存在，但不能执行

## 2. 交换指令XCHG (exchange)

◇ 功能：把存储在两个地方的数据进行互换

XCHG reg,reg/mem

; reg  $\leftrightarrow$  reg/mem

- ◇ 寄存器与寄存器之间对换数据
- ◇ 寄存器与存储器之间对换数据
- ◇ 不能在存储器与存储器之间对换数据
- ◇ 不能对段寄存器执行该指令



## 交换指令XCHG实例

mov ax,1199h ; ax=1199h

xchg ah,al ; ax=9911h

; 等同于 xchg al,ah

mov wvar,5566h ; wvar是一个字变量

xchg ax,wvar ; ax=5566h, wvar=9911h

; 等同于 xchg wvar,ax

xchg al,byte ptr wvar+1

; ax=5599h, wvar=6611h

; “byte ptr wvar+1”强制为字节量，只取高字节与AL类型交换，否则数据类型不匹配

### 3. 换码指令XLAT (translate)

- ◇ 功能：将BX指定的缓冲区中由AL指定的位移处的一个字节数据取出赋给AL。

**XLAT** ; al ← ds:[bx+al]

演示

- ◇ 换码指令执行前：

在主存建立一个字节量表格，内含要转换成的目标代码，表格首地址存放于BX，AL存放目标数据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

### 3. 换码指令XLAT应用举例---LED数码管驱动

Table DB C0H,F9H,A4H,B0H,99H ;0,1,2,3,4

DB 92H,82H,F8H,80H,90H ;5,6,7,8,9

**Mov bx,offset Table**

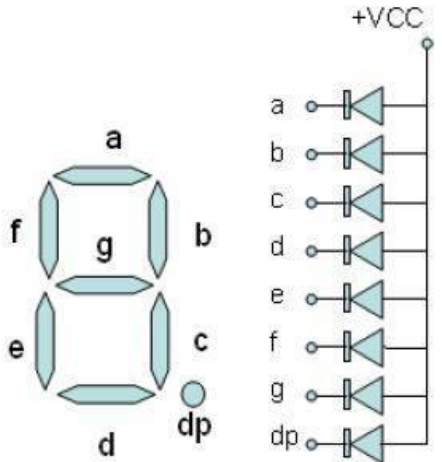
**Mov al,5**

**XLat**

;xlat指令执行后,

al=92h

显示字型	dp,g,f,e,d,c,b,a	字符码
0	1 1 0 0 0 0 0 0	C0H
1	1 1 1 1 1 0 0 1	F9H
2	1 0 1 0 0 1 0 0	A4H
3	1 0 1 1 0 0 0 0	B0H
4	1 0 0 1 1 0 0 1	99H
5	1 0 0 1 0 0 1 0	92H
6	1 0 0 0 0 0 1 0	82H
7	1 1 1 1 1 0 0 0	F8H
8	1 0 0 0 0 0 0 0	80H



## 2.3.2 堆栈操作指令

- ◇ 堆栈是一个“后进先出LIFO”（或说“先进后出FILO”）的主存区域，位于堆栈段中，SS段寄存器记录其段地址。
- ◇ 堆栈只有一个出口，即当前栈顶；用堆栈指针寄存器SP指定。
- ◇ 堆栈只有两种基本操作：进栈和出栈，对应两条指令PUSH和POP。

图示

# 进栈指令PUSH

- ◇ 进栈指令先使堆栈指针SP减2，然后把一个字操作数存入堆栈顶部

PUSH r16/m16/seg

;  $SP \leftarrow SP - 2$

;  $SS:[SP] \leftarrow r16/m16/seg$

指令实例：

push ax

push [2000h]

演示

# 出栈指令POP

- ◇ 出栈指令把栈顶的一个字<sup>字</sup>传送至指定的目的操作数，然后堆栈指针SP加2

POP r16/m16/seg

; r16/m16/seg ← SS:[SP]

; SP ← SP + 2

指令实例：

pop ax

pop wvar

演示

# 堆栈操作的特点

- ◇堆栈操作的单位是字，进栈和出栈只对字量。
- ◇字量数据向栈顶压入和从栈顶弹出时，同样遵循高高低低原则。
- ◇堆栈操作遵循后进先出原则，但可用**MOV**指令随机存取堆栈中的数据。
- ◇堆栈段是程序中不可或缺的一个内存区，常用来
  - ◆ 临时存放数据
  - ◆ 传递参数
  - ◆ 保存和恢复寄存器

## 2.3.3 标志操作指令

### 1. 标志位操作指令

- ◆ CLC ; 复位进位标志:  $CF \leftarrow 0$
- ◆ STC ; 置位进位标志:  $CF \leftarrow 1$
- ◆ CMC ; 求反进位标志:  $CF \leftarrow \sim CF$
- ◆ CLD ; 复位方向标志:  $DF \leftarrow 0$
- ◆ STD ; 置位方向标志:  $DF \leftarrow 1$
- ◆ CLI ; 复位中断标志:  $IF \leftarrow 0$
- ◆ STI ; 置位中断标志:  $IF \leftarrow 1$





## 2. 标志寄存器传送指令

指令格式：

**LAHF** ; (AH) ← (PSW低字节)

**SAHF** ; (PSW低字节) ← (AH)

**PUSHF** ; (SP) ← (SP-2), (SP+1), (SP) ← (PSW)

**POPF** ; (PSW) ← ((SP)+1, (SP)), (SP) ← (SP)+2

注意

- ① LAHF/SAHF指令是寄存器AH与标志寄存器PSW的低字节之间完成的字节型数据传送。
- ② PUSHF/POPF指令是标志寄存器PSW与堆栈间进行的字型数据传送。
- ③ 指令SAHF和POPF将影响标志位。

15	12	11	10	9	8	7	6	5	4	3	2	1	0
		OF	DF	IF	TF	SF	ZF		AF		PF		CF

## 2.3.4 有效地址传送指令LEA

- ◇ 功能：将存储器操作数的有效地址送至指定的16位通用寄存器。

**LEA r16, mem**

； r16←mem的有效地址EA

例：有效地址的获取

mov bx,400h

mov si,3ch

lea bx,[bx+si+0f62h]

； BX←400H+3CH+0F62H=139EH

## 例: 地址传送与内容传送

; lea.asm

wvar dw 4142h ; 假设偏移地址为04H

...

mov ax,wvar ; 内容传送: AX = 4142H

lea si,wvar ; 地址传送: SI = 0004H

; 等同于 lea si, [0004h]

mov cx,[si] ; 内容传送: CX = 4142H

mov di,offset wvar ; = mov di,0004h

; 利用操作符OFFSET获取变量的有效地址

mov dx,[di] ; 内容传送: DX = 4142H

# 地址传送指令

除LEA外还有两条地址传送指令：

## (1)指针送寄存器和DS

LDS REG, rem

; (REG)  $\leftarrow$  (rem) , (DS)  $\leftarrow$  (rem+2)

## (2)指针送寄存器和ES

LES REG, rem

; (REG)  $\leftarrow$  ( rem ) , (ES)  $\leftarrow$  (rem +2)

## 地址传送指令举例

### [例]

① 指出 **LEA AX, [5678H]** 执行后 **AX** 中的值。

答:  $(AX) = 5678H$

② 已知:  $(DS) = C000H$

$(C2480H) = 1357H$ ,  $(C2482H) = 2468H$

给出指令 **LDS SI, [2480H]** 的执行结果。

答:  $(SI) = 1357H$ ,  $(DS) = 2468H$

### 注意:

指令的源操作数不能使用立即数和通用寄存器, 目的操作数不能使用段寄存器。地址传送指令不影响状态标志位。



# 参考资料



# MOV指令的功能

MOV dest,src



播放



# XCHG指令的功能

XCHG reg,reg/mem



播放

返回



# XLAT指令的功能

XLAT

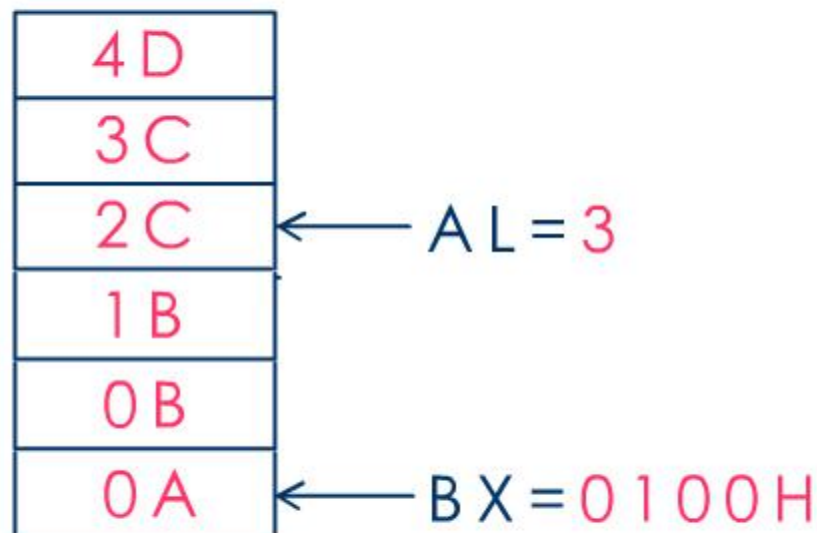
执行前：

AL

BX

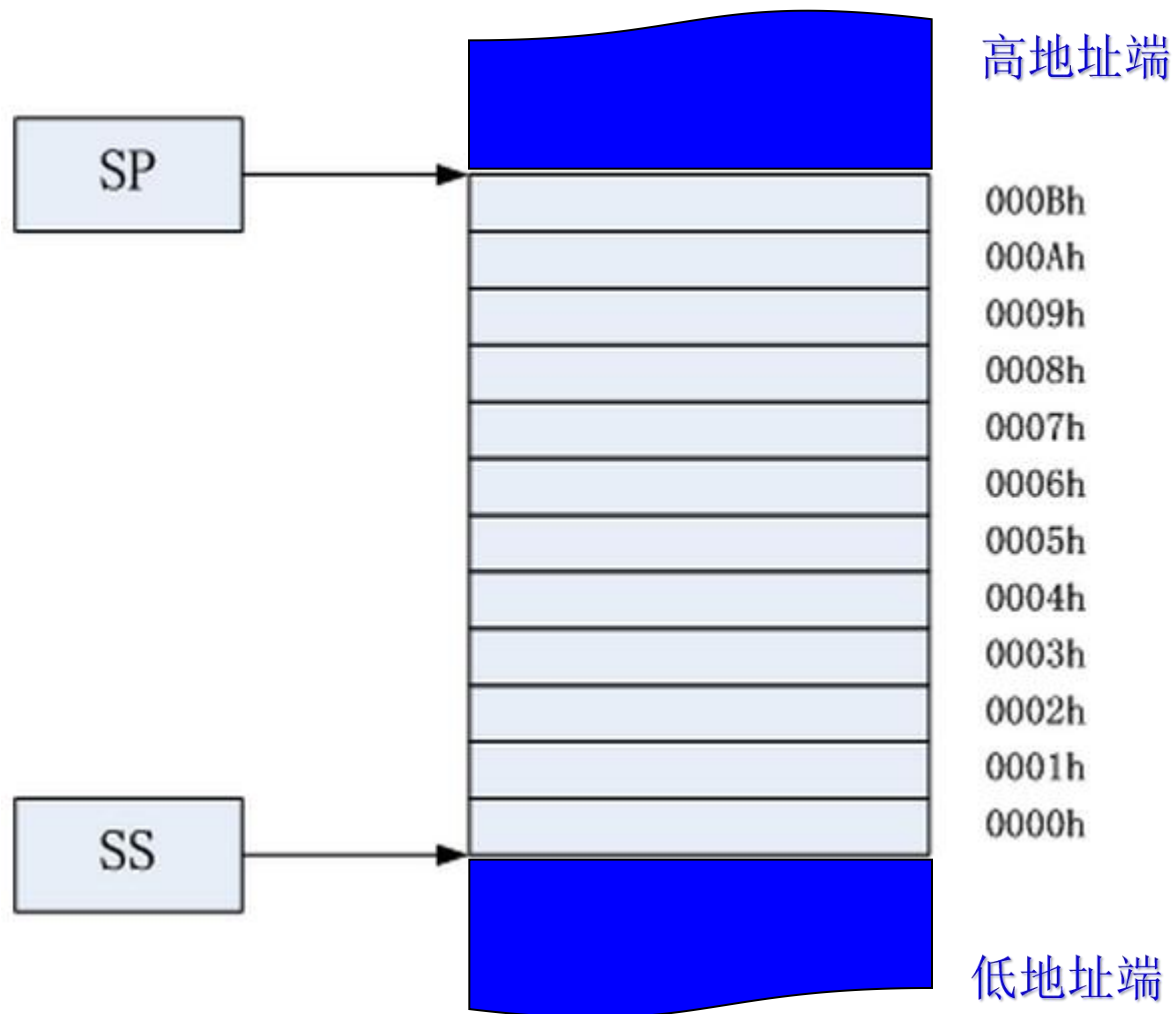
执行后：

AL



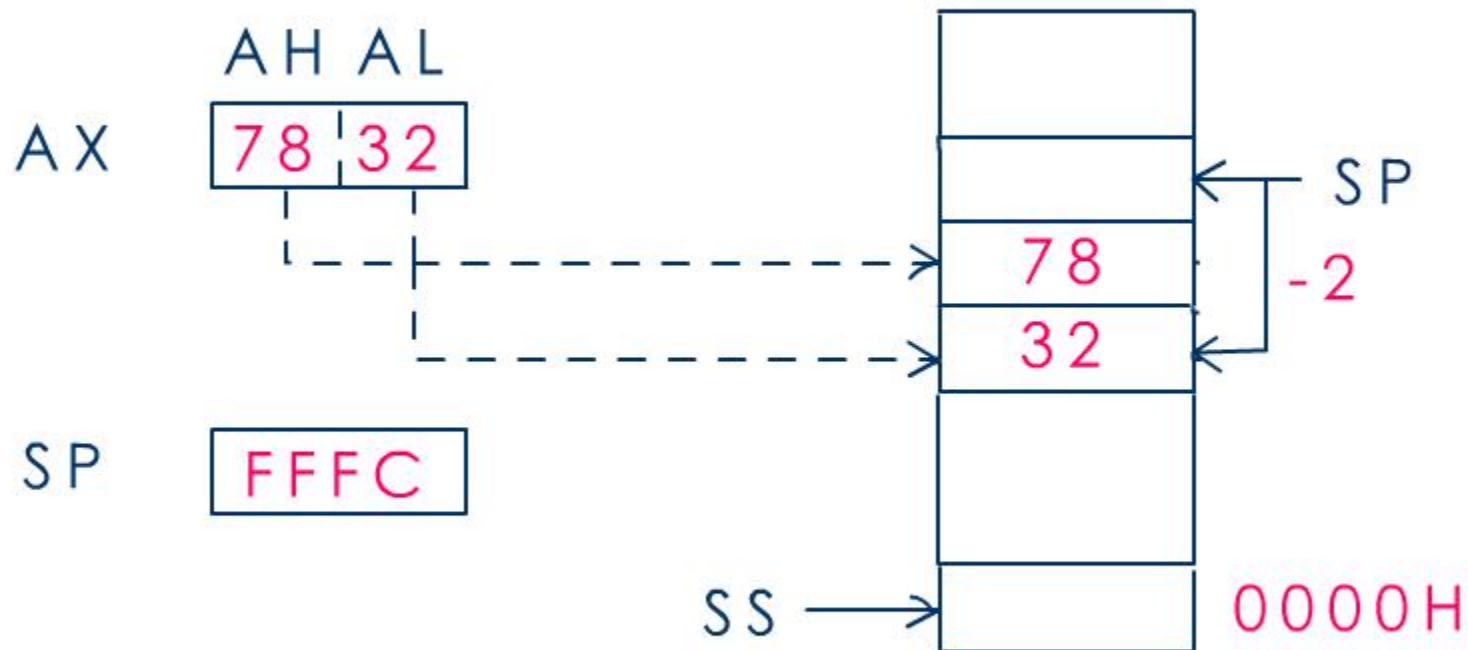
返回

# 堆栈的图示



# 进栈指令PUSH

PUSH AX



播放

返回

# 出栈指令POP

POP AX

