MOV指令

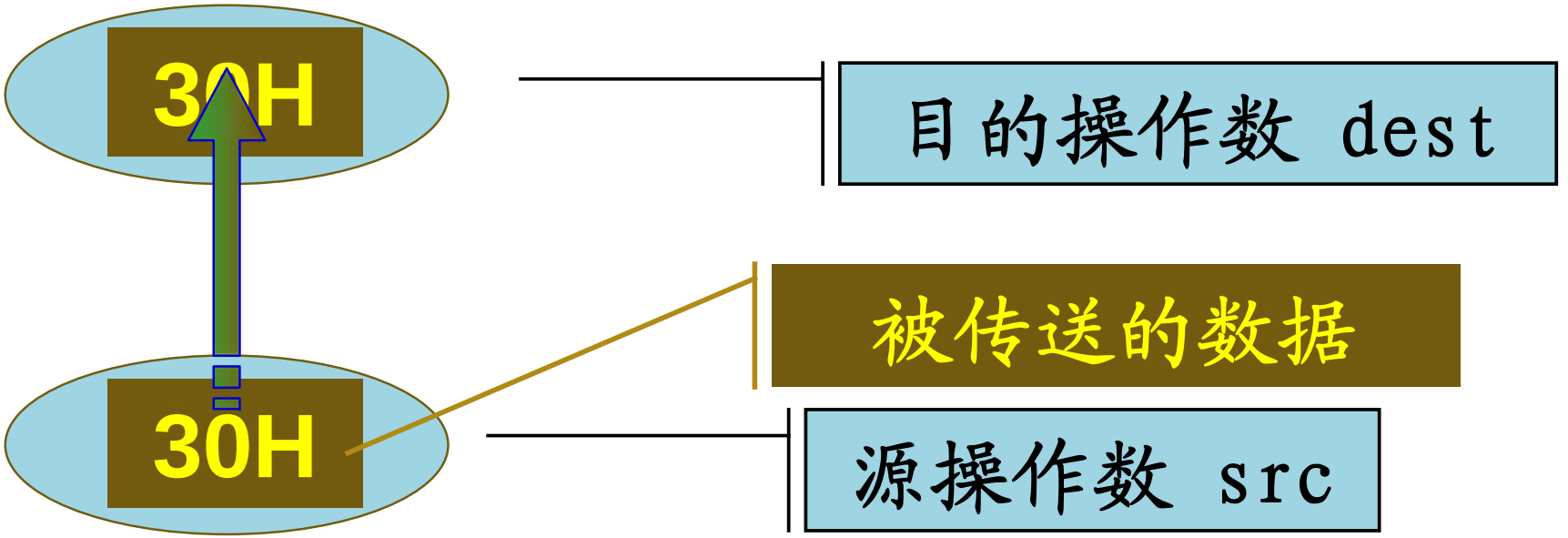
数据传送类指令

- 数据传送
  - 把数据从一个位置传送到另一个位置
  - 计算机中最基本的操作
  - 程序设计中最常使用的指令
- 除标志寄存器传送指令外，均不影响状态标志

MOV　XCHG　PUSH　POP　LEA

MOV指令的功能

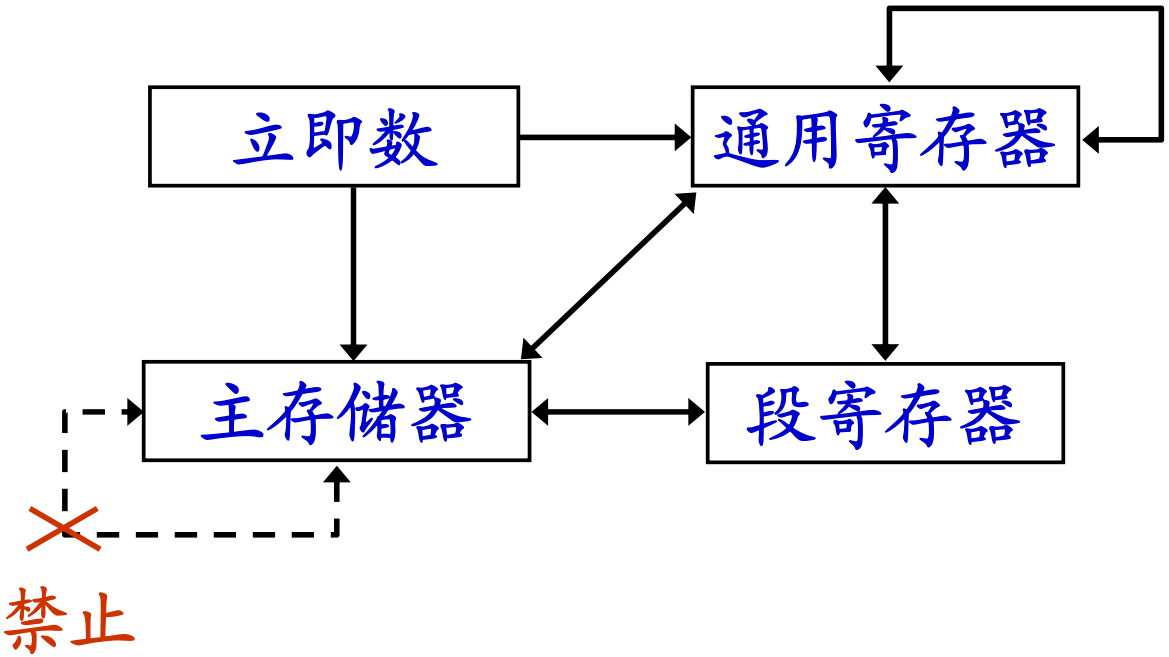
- 提供方便灵活的通用数据传送操作



传送指令MOV（move）

- 把一个字节、字或双字的操作数

从源位置传送至目的位置



MOV reg/mem,imm

MOV reg/mem/seg,reg

MOV reg/seg,mem

MOV r16/m16,seg

IA-32指令支持3种数据长度

- **8位（字节）数据，byte类型**

  mov al,200

- **16位（字）数据，word类型**

  mov ax,[ebx]

- **32位（双字）数据，dword类型**

  mov eax,dvar

32位通用寄存器:

EAX EBX ECX ...

16位通用寄存器:

AX BX CX DX ...

8位通用寄存器:

AH AL BH BL ...

16位段寄存器:

DS CS SS ES ...

立即数传送

- 寄存器reg为目的操作数

```
mov al,200
;8位立即数i8
mov ax,200
;16位立即数i16
mov eax,200
;32位立即数i32
```

- 存储器mem为目的操作数

```
mov bvar,byte ptr 200
;8位立即数i8
mov [ebx],word ptr 200
;16位立即数i16
mov [esi+8],dword ptr 200
;32位立即数i32
```

MOV reg/mem,imm

寄存器传送

- 寄存器reg为目的操作数

```
mov al,ah
;8位通用寄存器r8
mov ax,bx
;16位通用寄存器r16
mov eax,edx
;32位通用寄存器r32
```

- 存储器mem为目的操作数

```
mov bvar,cl
;8位通用寄存器r8
mov [ebx],cx
;16位通用寄存器r16
mov [esi+8],edi
;32位通用寄存器r32
```

MOV reg/mem/seg,reg

- 段寄存器seg为目的操作数

```
mov ds,bx
```

存储器传送

- 寄存器reg为目的操作数

```
mov dl,bvar
;8位储存器m8
mov dx,[ebx]
;16位存储器m16
mov edx,dvar[edi]
;32位存储器m32
```

- 段寄存器seg为目的操作数

```
mov ds,wvar
;16位存储器m16
mov es,[ebx]
;16位存储器m16
mov ss,[ebp+8]
;16位存储器m16
```

MOV reg/seg,mem

（16位）段寄存器传送

- 寄存器**r16**为目的操作数

  mov ax,ds

  mov dx,es

  mov si,fs

  mov di,gs

- 存储器**m16**为目的操作数

  mov wvar,ds

  mov [ebx],ss

  mov [esi-8],cs

  mov [ebp+8],cs

MOV r16/m16,seg

传送指令MOV（move）

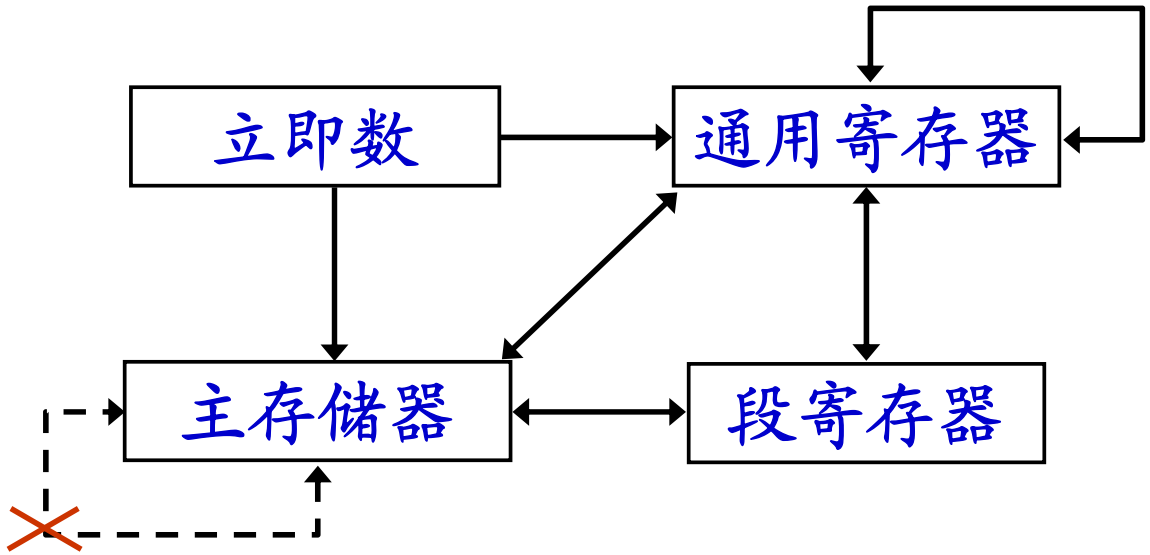
- 把一个字节、字或双字的操作数
  从源位置传送至目的位置

MOV reg/mem,imm

MOV reg/mem/seg,reg

MOV reg/seg,mem

MOV r16/m16,seg



并非任意传送！

LEA指令

数据传送类指令

- 数据传送
  - 把数据从一个位置传送到另一个位置
  - 计算机中最基本的操作
  - 程序设计中最常使用的指令
- 除标志寄存器传送指令外，均不影响状态标志

MOV　XCHG　PUSH　POP　LEA

地址传送指令LEA（Load Effective Address）

- 地址传送指令获取存储器操作数的地址

LEA r16/r32,mem

;r16/r32←mem的有效地址EA（不需类型一致）

;IA-32使用32位地址，保存于32位通用寄存器r32

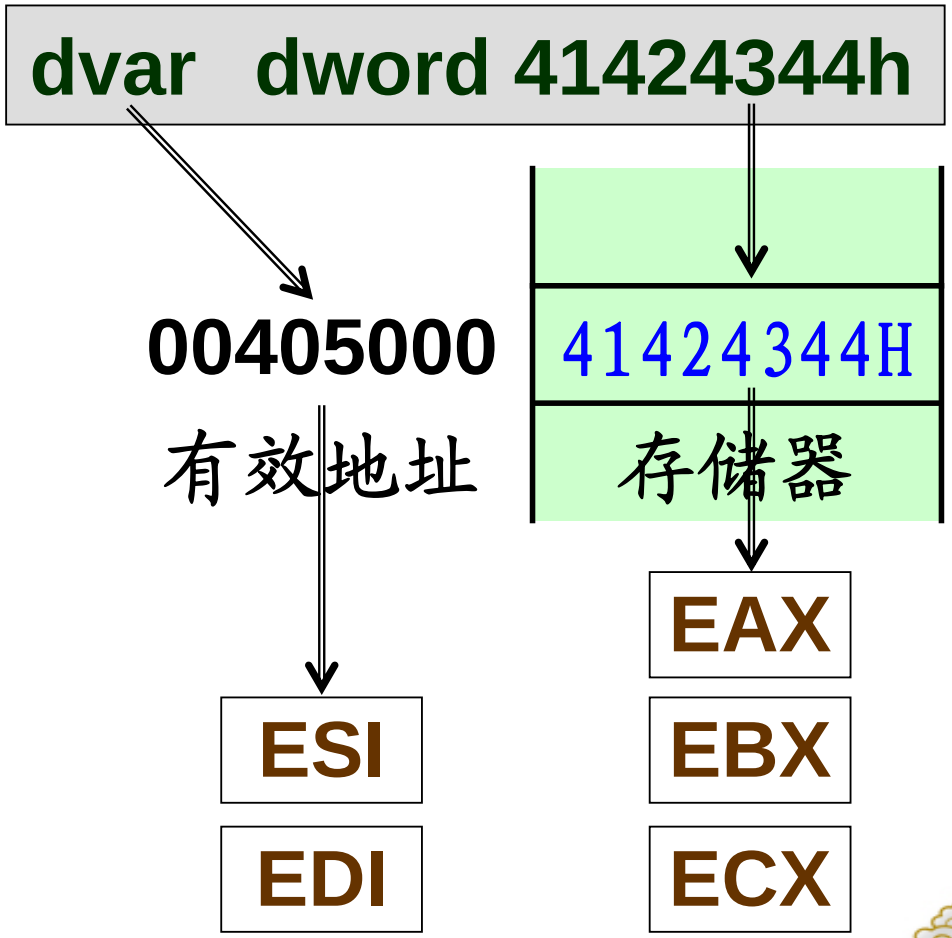
LEA指令类似地址操作符OFFSET的作用

- LEA指令在指令执行时计算出偏移地址
  - OFFSET操作符在汇编阶段取得变量的偏移地址
- OFFSET无需在执行时计算、指令执行速度更快
  - LEA指令能获取汇编阶段无法确定的偏移地址

```
lea   edi, var
mov edi, offset var
```

地址传送程序

```
mov eax,dvar
lea esi,dvar
mov ebx,[esi]
mov edi,offset dvar
mov ecx,[edi]
```

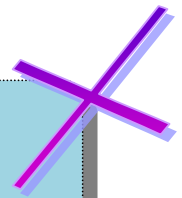


用LEA指令实现运算功能

```
lea edx,[esi+edi*4+100h]
;EDX＝ESI＋EDI×4＋100H
```

- LEA指令在计算地址时，可进行加和移位操作
  - 编译器将其用于实现加法，或者乘以2、4和8
- 不能用OFFSET操作符实现

```
mov edx, offset [esi+edi*4+100h]
```



本讲总结

- LEA指令获得存储器操作数的有效地址
  - 在LEA指令执行时计算地址
  - 对任何存储器寻址方式都可用
- 对存储器的直接寻址
  - 建议用OFFSET操作符在汇编阶段获得地址
- 对存储器的其他寻址
  - 只能使用LEA指令获得地址

PUSH和POP指令

数据传送类指令

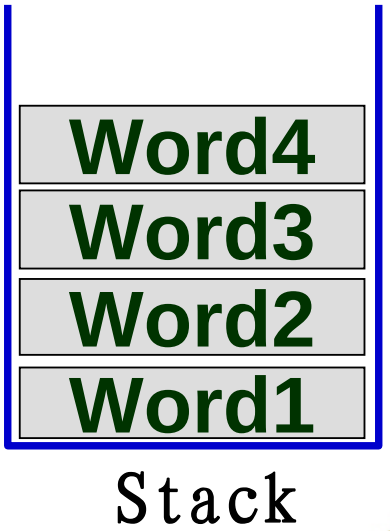
- 数据传送
  - 把数据从一个位置传送到另一个位置
  - 计算机中最基本的操作
  - 程序设计中最常使用的指令
- 除标志寄存器传送指令外，均不影响状态标志

MOV　XCHG　PUSH　POP　LEA

PUSH和POP是堆栈操作指令

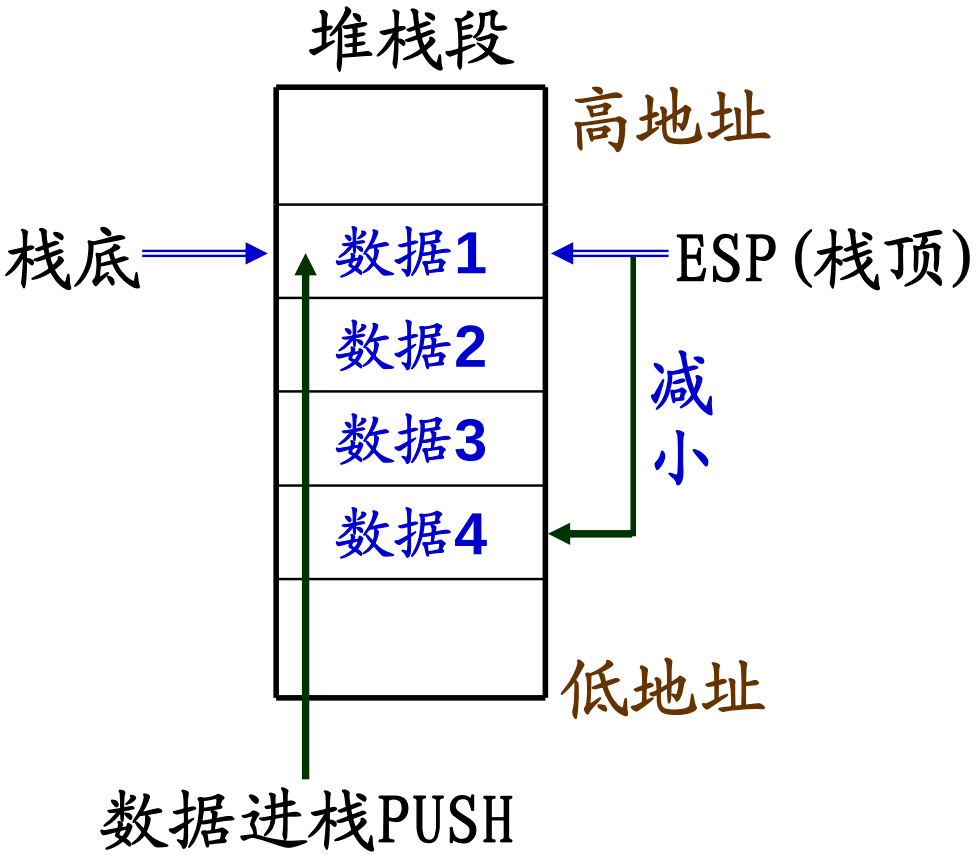
- 堆栈一个特殊的存储区域
  - 存取原则：先进后出**FILO（First In Last Out）**

  也可称为：后进先出**LIFO （ Last In First Out）**
  - 具有两种基本的数据传输操作
    - 数据压进堆栈　**PUSH**
    - 数据弹出堆栈　**POP**

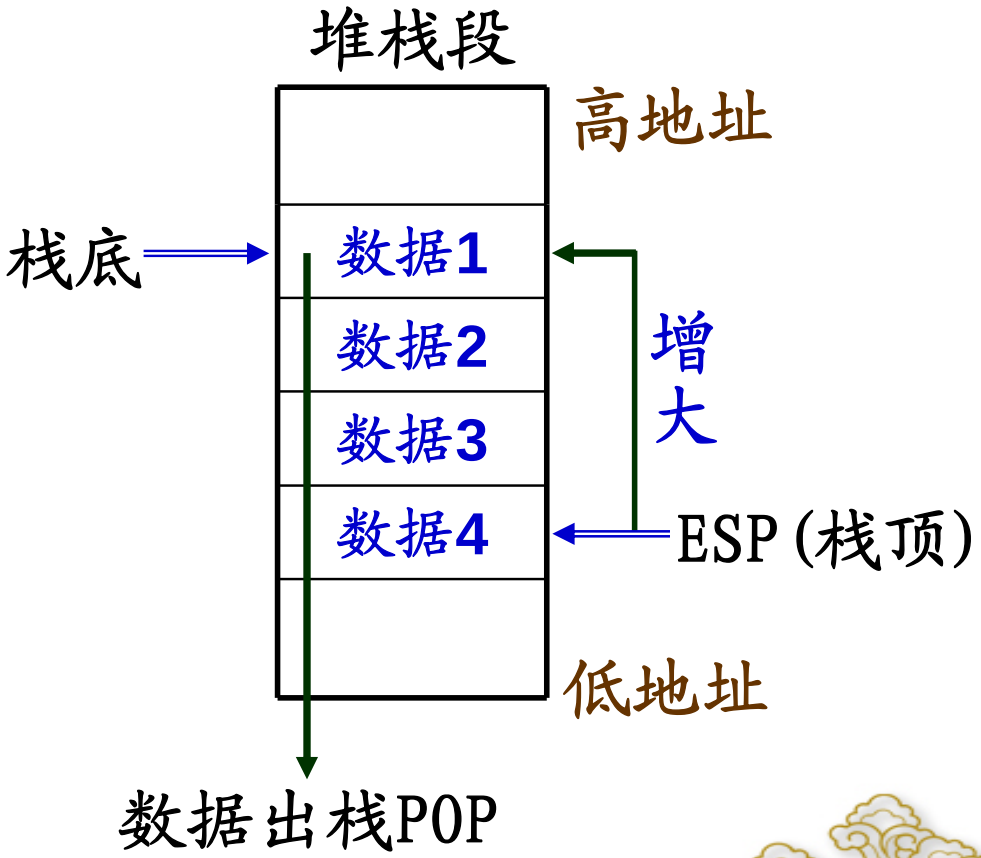


IA-32处理器的堆栈向下生长

数据压入堆栈、ESP逐渐减小



数据弹出堆栈、ESP逐渐增大



进栈指令PUSH

PUSH r16/m16/i16/seg

; ① ESP＝ESP－2　　② SS:[ESP]＝r16/m16/i16/seg

PUSH r32/m32/i32

; ① ESP＝ESP－4　　② SS:[ESP]＝r32/m32/i32

- 先将**ESP**减小作为当前栈顶
- 后将源操作数传送到当前栈顶

- 以字或双字为单位操作
  - 进栈字量数据前，**ESP**减2
  - 进栈双字量数据前，**ESP**减4

push eax

进栈PUSH操作

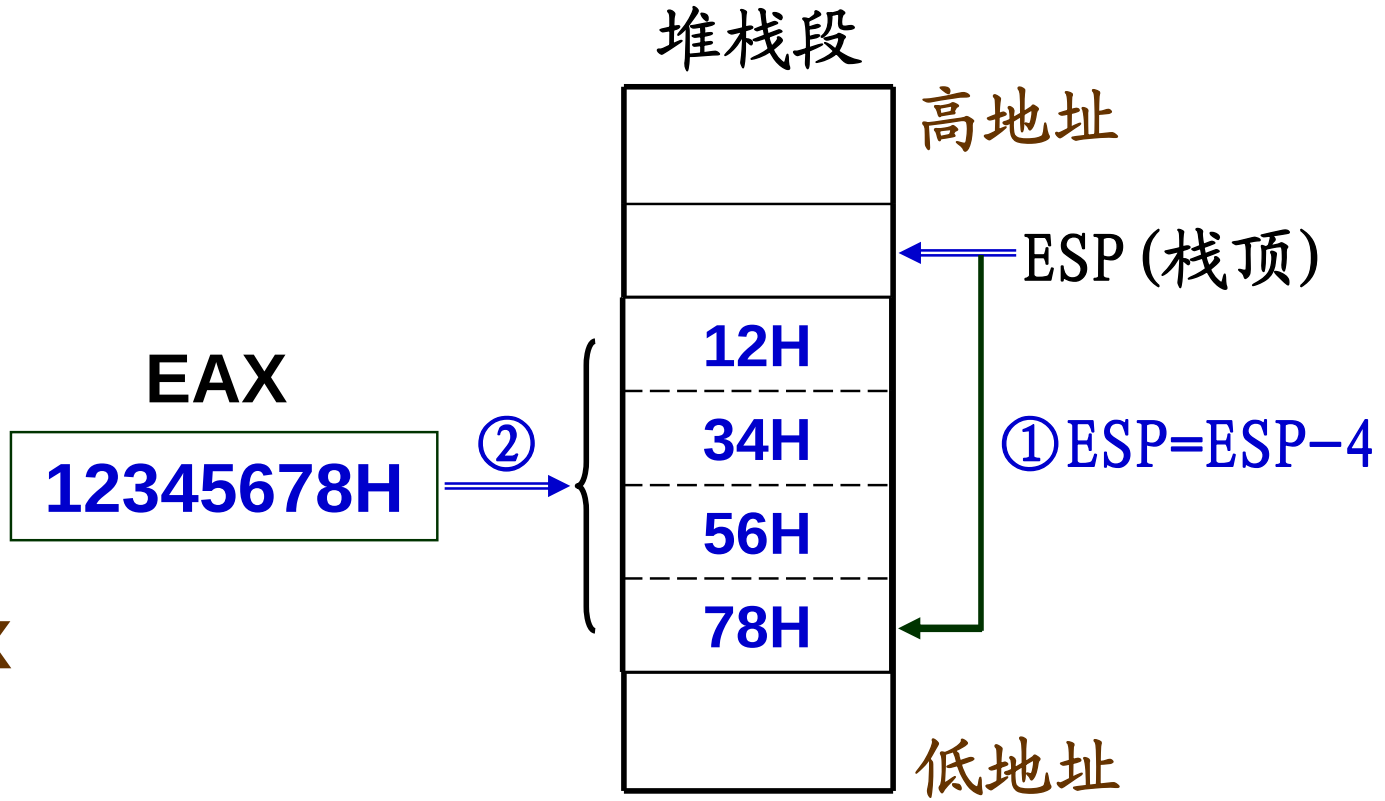
;进栈指令

push eax

;等同于如下2条指令

① **sub esp,4**

② **mov [esp],eax**



出栈指令POP

POP r16/m16/seg

; ① r16/m16/seg＝SS:[ESP]　② ESP＝ESP＋2

POP r32/m32

; ① r32/m32＝SS:[ESP]　② ESP＝ESP＋4

- 先将栈顶数据传送到目的操作数
- 后**ESP**增加作为当前栈顶

- 以字或双字为单位操作
  - 出栈字量数据后，**ESP**加**2**
  - 出栈双字量数据后，**ESP**加**4**

pop eax

出栈POP操作

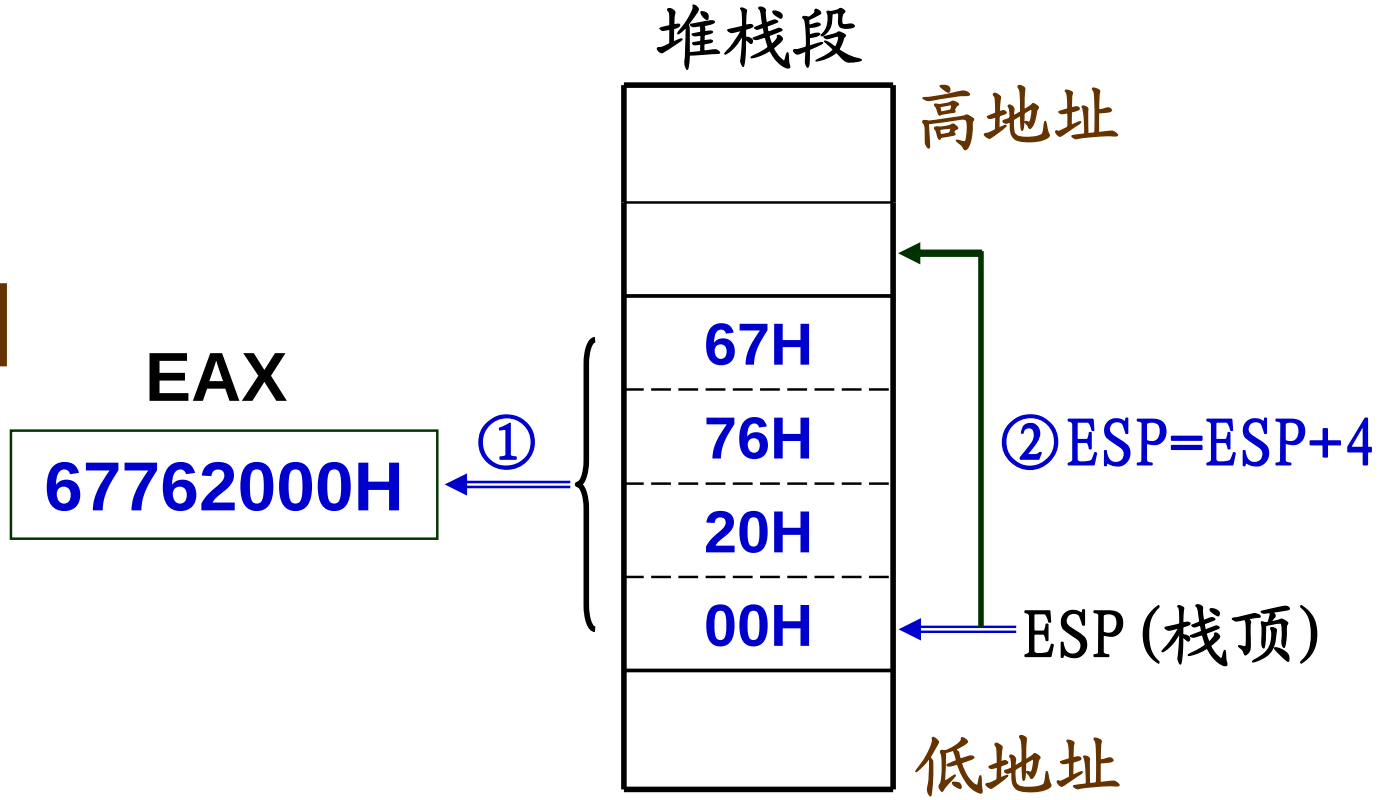
;进栈指令

pop eax

;等同于如下2条指令

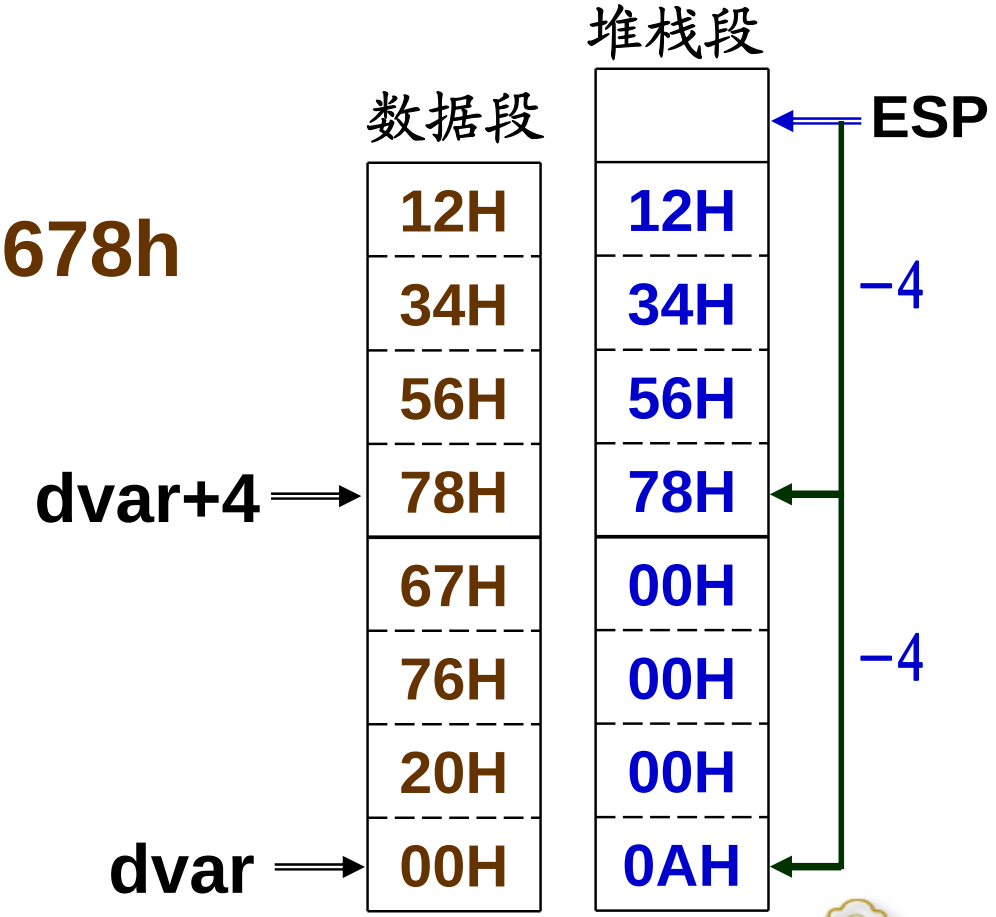
① **mov eax,[esp]**

② **add esp,4**



堆栈操作程序—1

```
        ;数据段
ten     = 10
dvar    dword 67762000h,12345678h
        ;代码段
        mov eax,dvar+4
        push eax
        push dword ptr ten
;将立即数以双字量压入堆栈
```



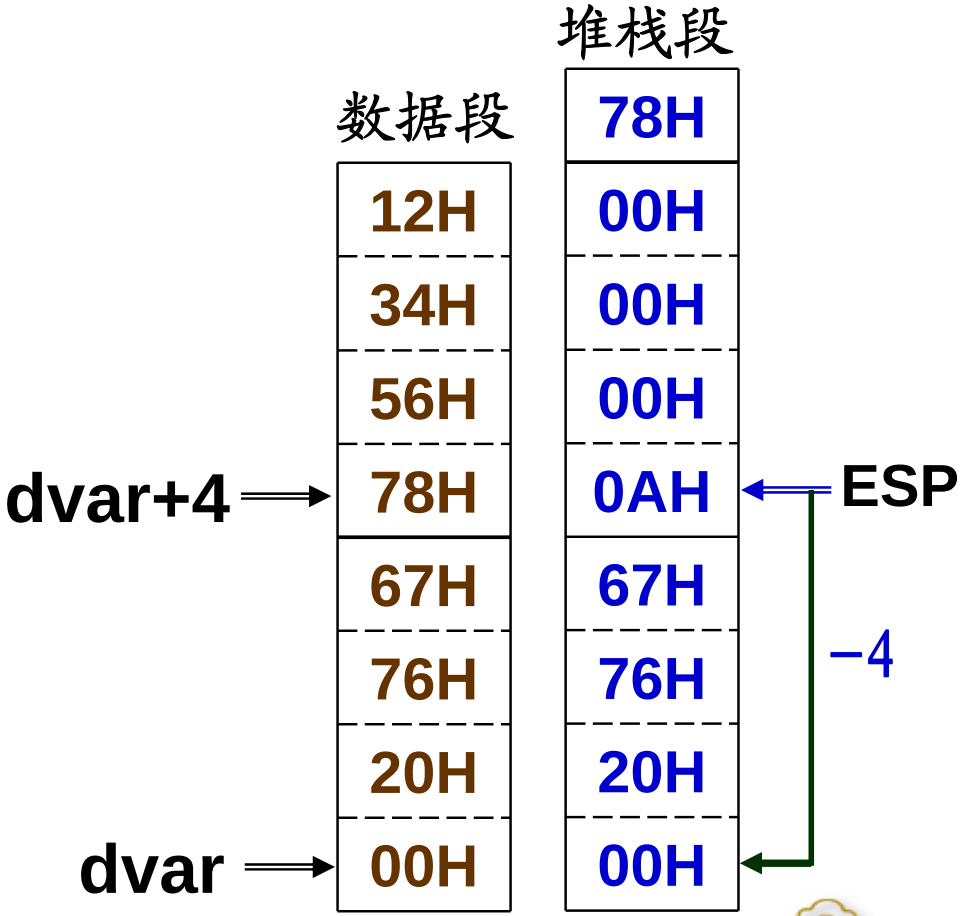
堆栈操作程序—2

push dvar

pop eax

;栈顶数据弹出到**EAX**

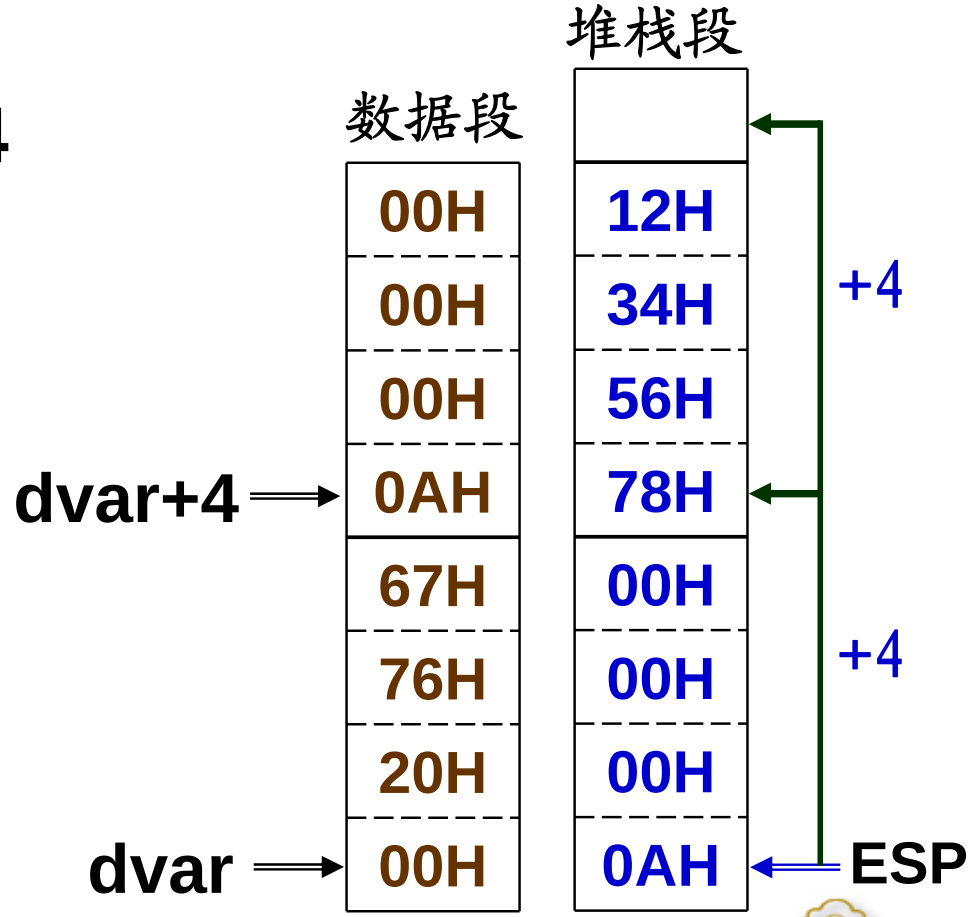
EAX = 67762000H



堆栈操作程序—3

```
pop dvar+4
;栈顶数据弹出到DVAR＋4
mov ebx,dvar+4
pop ecx
;栈顶数据弹出到ECX
```

EAX = 67762000H
EBX = 0000000AH
ECX = 12345678H



本讲总结

- 堆栈是一个后进先出**LIFO**存取原则的存储区域
  - 数据进栈（压入）使用**PUSH**指令
  - 数据出栈（弹出）使用**POP**指令
- 应用中要明确当前栈顶