存储器寻址

寻址方式（Addressing）

- 通过地址访问数据或指令
- 数据寻址：

  指令执行过程中，

  访问所需要操作的数据(操作数)

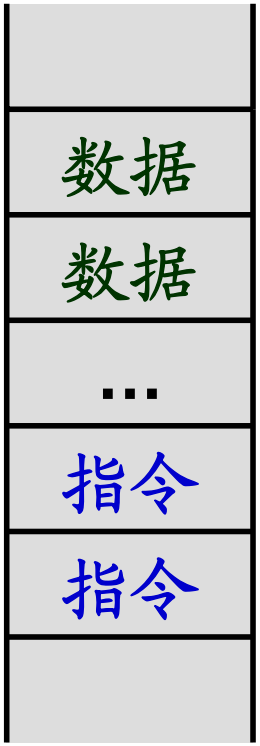
操作数在哪儿呢？

- 指令寻址：

  一条指令执行后，

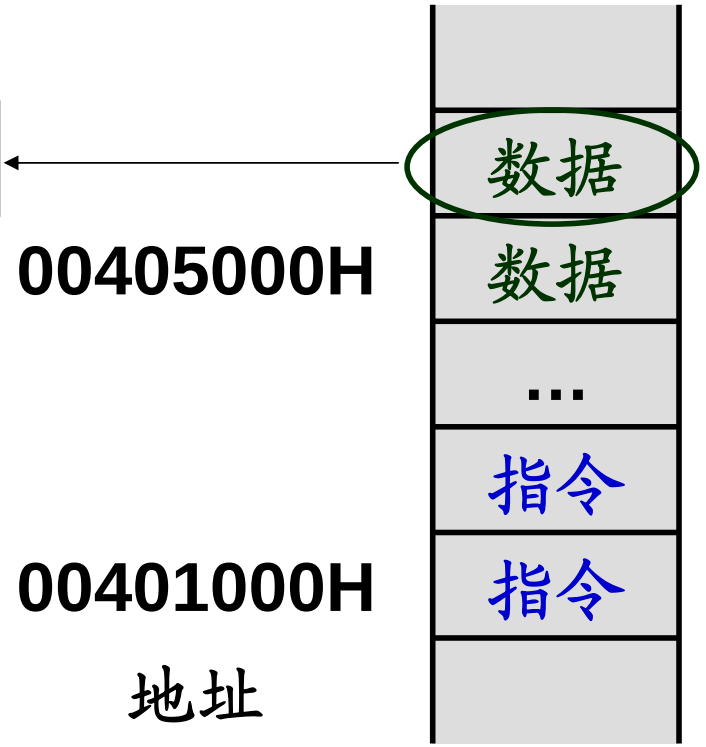
  确定执行的下一条指令的位置

指令又在哪儿呢？



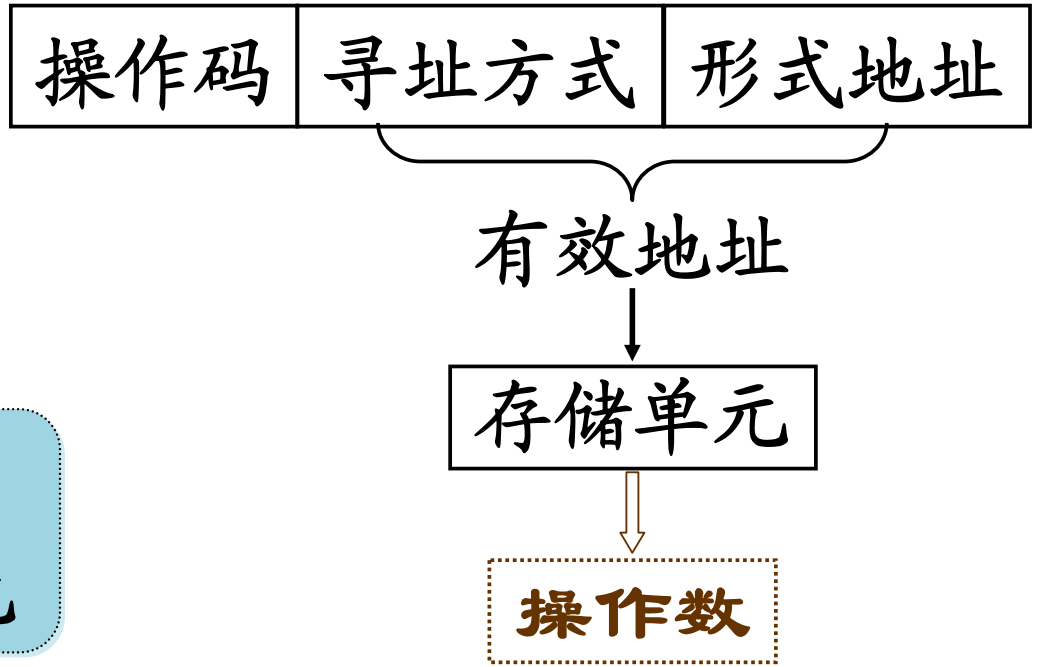
操作数在那儿！

数据来自主存储器→存储器寻址



存储器寻址

- 操作数在主存中，通过存储器地址指示
  - 指令代码表达形式地址
  - 由形式地址结合规则
    经过计算得到有效地址**EA**
    （**Effective Address**）



处理器将有效地址转换
为物理地址访问存储单元

存储器的逻辑地址

- 编程时，存储器地址使用逻辑地址
  - 段寄存器指示段基地址
    - 绝大部分情况使用默认规定，无需表达
    - 有时需要显式说明（称为段超越）
  - 偏移地址由各种存储器寻址方式计算
    - 这被称为有效地址**EA**

逻辑地址 = 段基地址：偏移地址

段寄存器指示段基地址

- 绝大部分情况使用默认规定，无需表达
  - 读取指令，一定是代码段**CS**
  - 堆栈操作，针对堆栈段**SS**
  - 读写数据，默认在数据段**DS**
- 有时需要显式说明（称为段超越）
  - 使用段超越指令前缀（段寄存器名后跟冒号）

CS:　DS:　SS:　ES:　FS:　GS:

段寄存器的默认和超越

访问存储器的方式	默认	可超越	偏移地址
读取指令	CS	无	EIP
堆栈操作	SS	无	ESP
一般数据访问(下列除外)	DS	CS ES SS FS GS	有效地址EA
EBP/ESP基址的数据访问	SS	CS ES DS FS GS	有效地址EA
串指令的源操作数	DS	CS ES SS FS GS	ESI
串指令的目的操作数	ES	无	EDI

32位有效地址的组成

基址寄存器＋变址寄存器×比例＋位移量

任何8个32位通用寄存器之一

1，2，4或8

除ESP外的任何32位通用寄存器之一

8或32位有符号值

EBX+ESI*4+80h

16位有效地址的组成

基址寄存器＋变址寄存器＋位移量

BX+SI+80h

32位存储器寻址

基址寄存器＋变址寄存器×比例＋位移量

{无, EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP} ＋ {无, EAX, EBX, ECX, EDX, ESI, EDI, EBP} × {1, 2, 4, 8} ＋ {8位位移量, 32位位移量}

组合成多种存储器寻址方式

本讲总结

- 存储器寻址
  - 操作数在主存中，通过存储器地址指示
  - 指令代码表达形式地址
- 形式地址结合规则计算得到有效地址EA

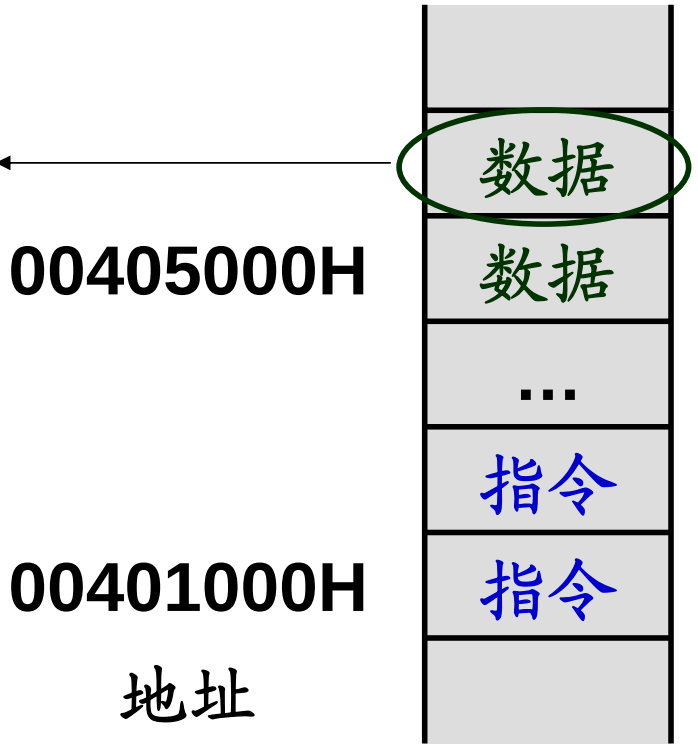
有效地址EA
= 基址寄存器
+ 变址寄存器 × 比例
+ 位移量

符号	含义
m8	8位存储器单元
m16	16位存储器单元
m32	32位存储器单元
mem	存储器操作数

存储器的直接寻址

操作数在那儿！

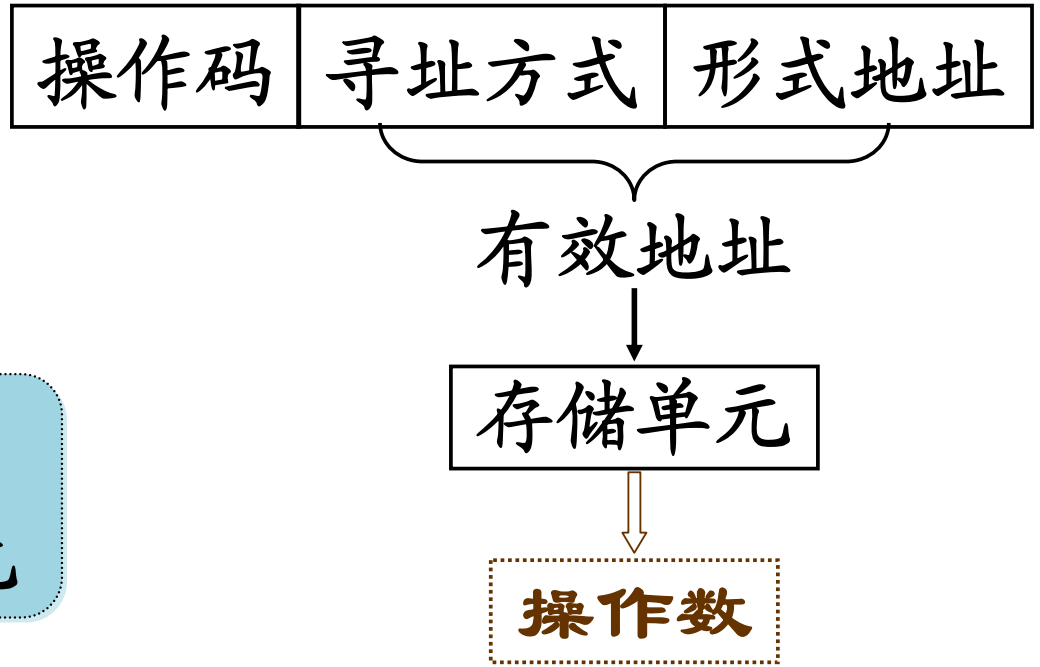
数据来自主存储器→存储器寻址



存储器寻址

- 操作数在主存中，通过存储器地址指示
  - 指令代码表达形式地址
  - 由形式地址结合规则
    经过计算得到有效地址**EA**
    （**Effective Address**）

处理器将有效地址转换
为物理地址访问存储单元



32位存储器寻址

基址寄存器＋变址寄存器×比例＋位移量

{无, EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP} ＋ {无, EAX, EBX, ECX, EDX, ESI, EDI, EBP} × {1, 2, 4, 8} ＋ {8位位移量, 32位位移量}

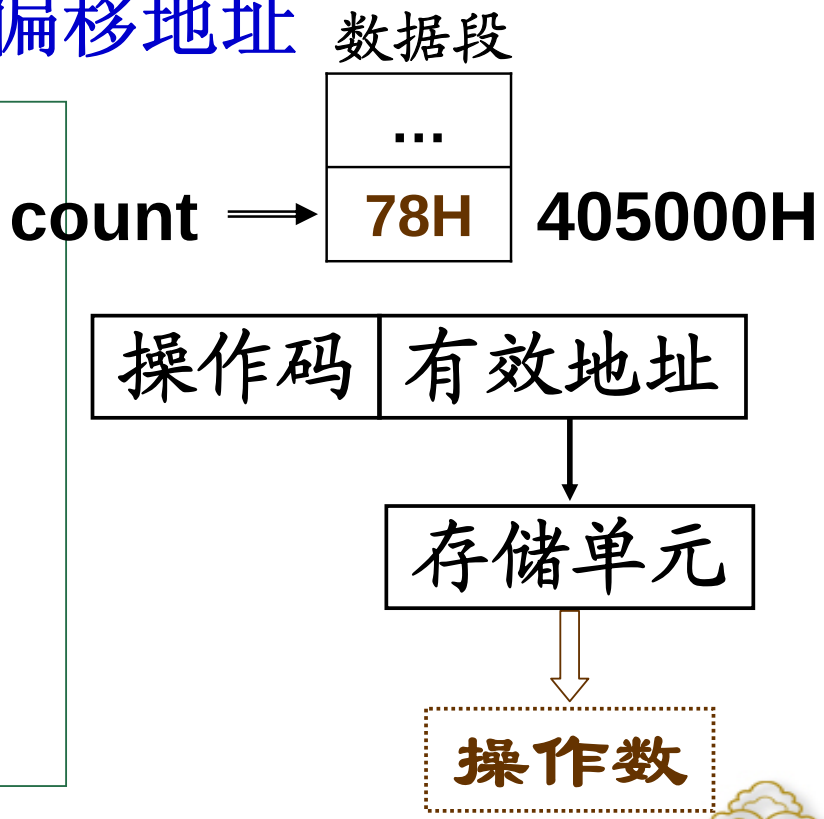
只有位移量部分

组合成多种存储器寻址方式

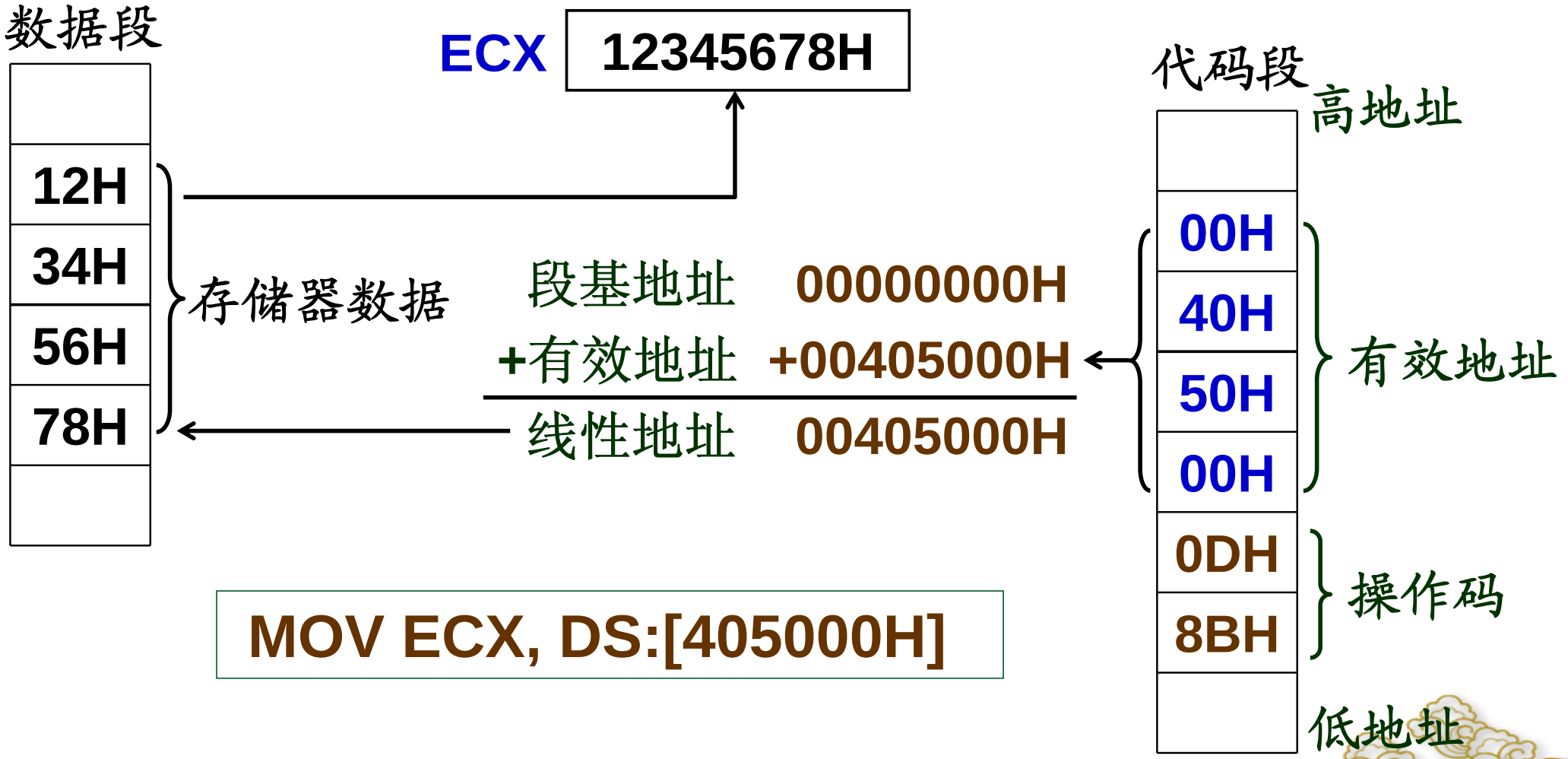
直接寻址

- 有效地址只有位移量部分，直接包含在指令代码中
  - 用变量名（或加中括号）表示偏移地址

```
MOV ECX,COUNT
MOV ECX,[COUNT]
MOV ECX, DS:[405000H]
;指令代码：8B 0D 00 50 40 00
;操作码：8B 0D
;操作数：有效地址 00405000H
```

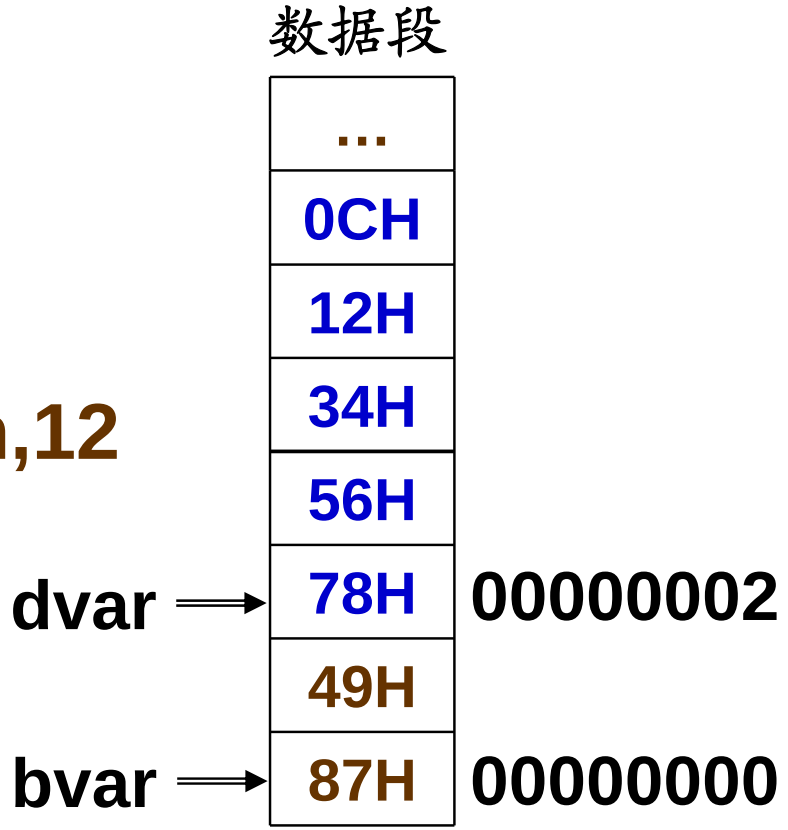


存储器直接寻址示意图


数据段
ECX
12345678H
代码段
高地址
12H
34H
56H
78H
存储器数据
段基地址 00000000H
+有效地址 +00405000H
线性地址 00405000H
00H
40H
50H
00H
有效地址
0DH
8BH
操作码
低地址
MOV ECX, DS:[405000H]

存储器直接寻址程序—1

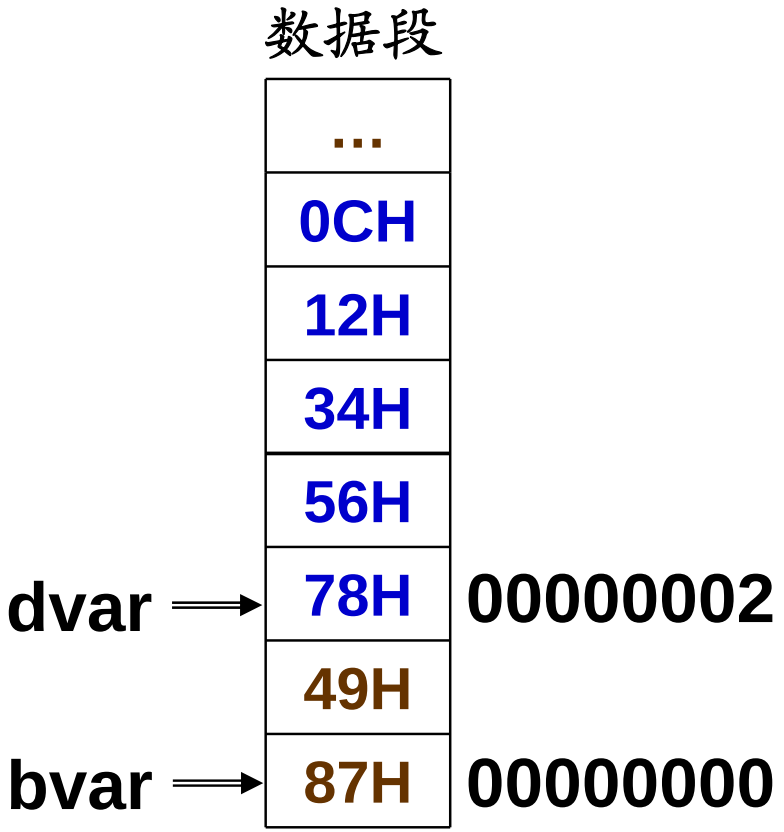
```
                    ;数据段
00000000   87 49
           bvar    byte 87h,49h
00000002 12345678 0000000C
           dvar    dword 12345678h,12
```



存储器直接寻址程序—2

```
                ;代码段
00000000   8A 0D 00000000 R
           mov cl, bvar
00000006   8B 15 00000002 R
           mov edx, dvar
```

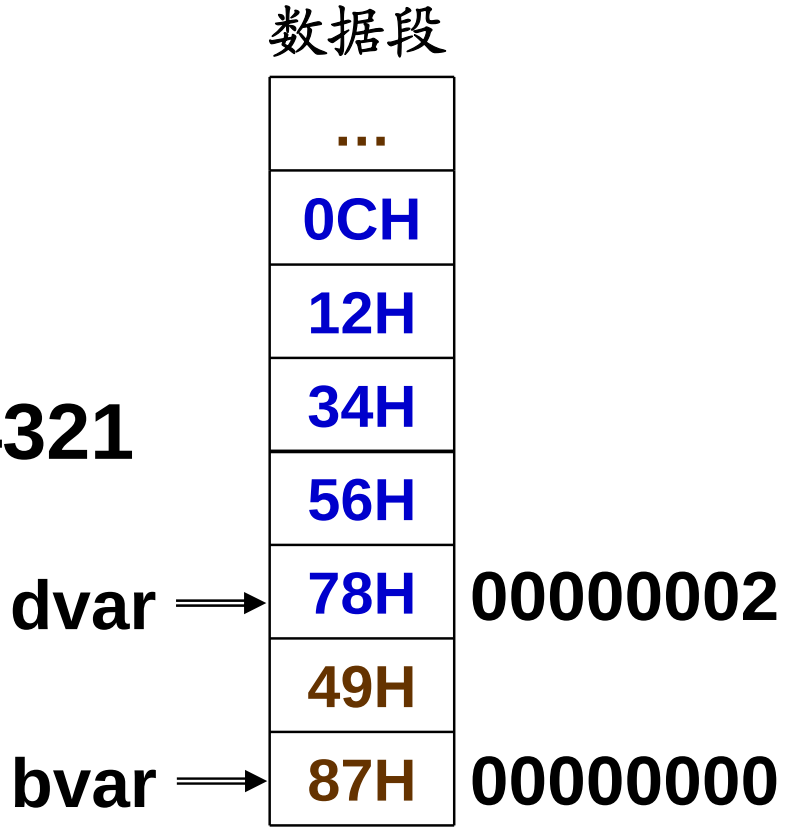
主存操作数常通过变量形式引用
一般不需要使用段超越前缀指令



存储器直接寻址程序—3

```
0000000C   88 35 00000001 R
           mov bvar+1,dh
00000012   66|89 15 00000004 R
           mov word ptr dvar+2,dx
00000019   C7 05 00000002 R 87654321
           mov dvar,87654321h
```

源操作数和目的操作数均可以采用存储器操作数，但不能同时使用



存储器直接寻址程序—4

mov dvar+4,dvar

eg0210.asm(13) : error A2070:

invalid instruction operands

错误语句的行号

错误编号

错误信息

- 常见语法错误原因
  - 拼写错误、多余的空格、遗忘的后缀字母或前导0、不正确的标点、太过复杂的常量或表达式
  - 操作数类型不匹配、2个存储器操作数……

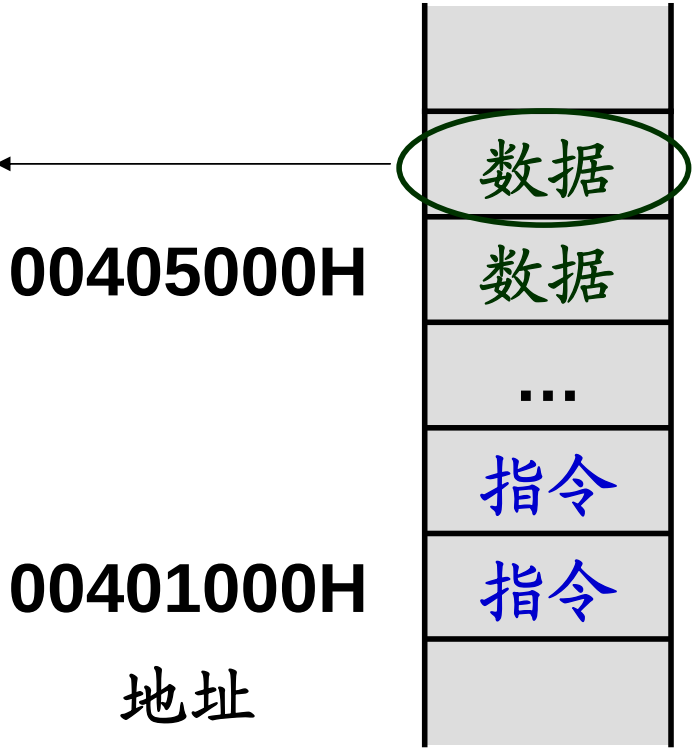
本讲总结

- 存储器的直接寻址
  - 指令代码中直接给出有效地址
- 直接寻址常用于存取变量
  - 直接使用变量名表达：变量名
  - 变量名加或减一个常量：变量名+n　变量名-n
  - 或者用中括号括起变量名：[变量名]　变量名[n]
- 直接寻址的操作数具有变量定义的类型
  - 有时需要通过PTR操作符进行类型转换

存储器的寄存器间接寻址

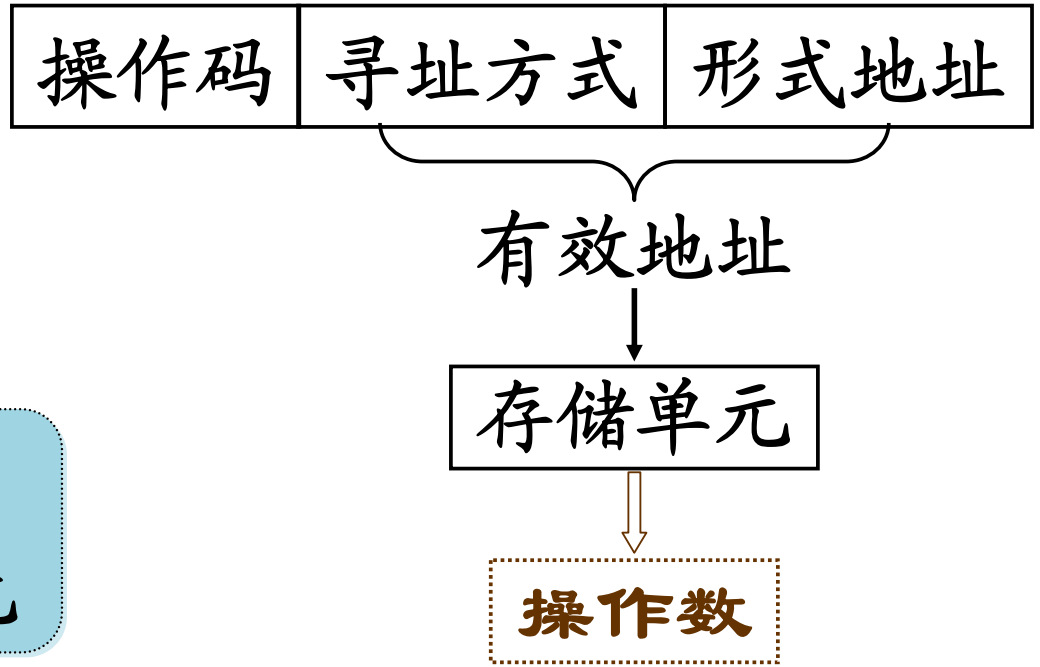
操作数在那儿！

数据来自主存储器→存储器寻址



存储器寻址

- 操作数在主存中，通过存储器地址指示
  - 指令代码表达形式地址
  - 由形式地址结合规则
    经过计算得到有效地址**EA**
    （**Effective Address**）



处理器将有效地址转换
为物理地址访问存储单元

32位存储器寻址

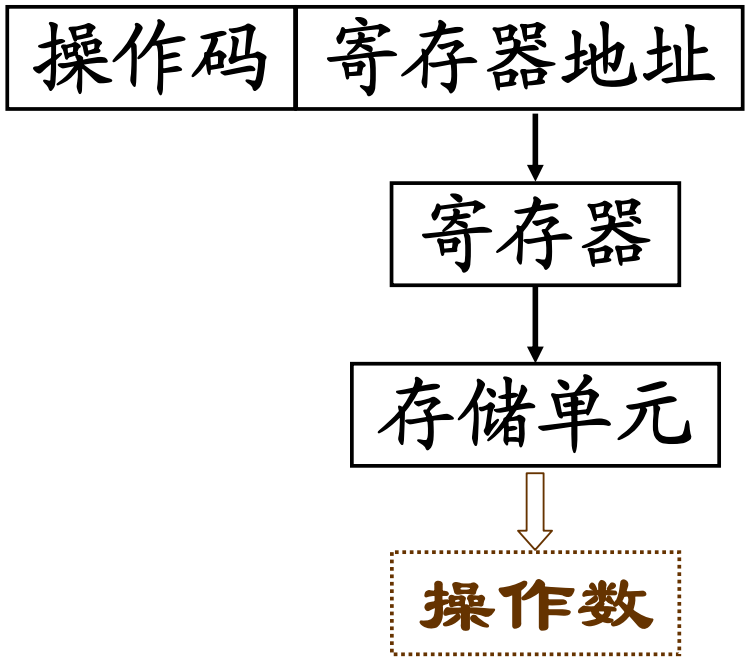
基址寄存器＋变址寄存器×比例＋位移量

- 基址寄存器：无 / 只有寄存器部分（EBX、ECX、EDX、ESI、EDI、EBP、ESP）
- ＋
- 变址寄存器：无 / EAX、EBX、ECX、EDX、ESI、EDI、EBP
- ×
- 比例：1、2、4、8
- ＋
- 位移量：8位位移量 / 32位位移量

组合成多种存储器寻址方式

寄存器间接寻址

- 有效地址存放在寄存器中
  (寄存器内容＝偏移地址＝有效地址)
  - 用中括号括起寄存器表达




```
mov edx,[ebx]     ;双字量传送
mov cx,[esi]      ;字量传送
mov [edi],al      ;字节量传送
```

寄存器间接寻址的数据
由另一个操作数的寄存器或变量类型决定

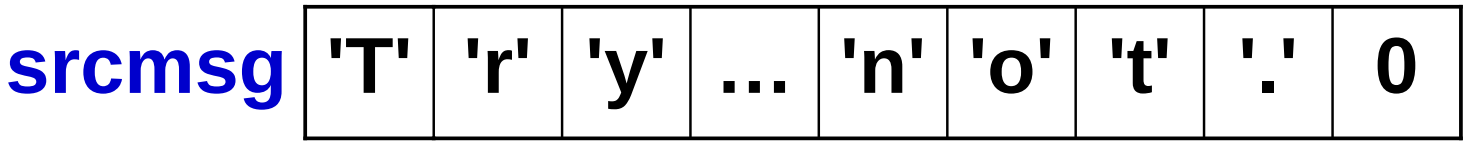
寄存器间接寻址未说明数据类型

```
MOV [EBX],100      ;错误，数据类型不明确
    mov byte ptr [ebx],100
    ;正确：byte ptr说明是字节操作
    mov word ptr [ebx],100
    ;正确：word ptr说明是字操作
    mov dword ptr[ebx],100
    ;正确：dword ptr说明是双字操作
```

立即数也无类型
故需要显式说明

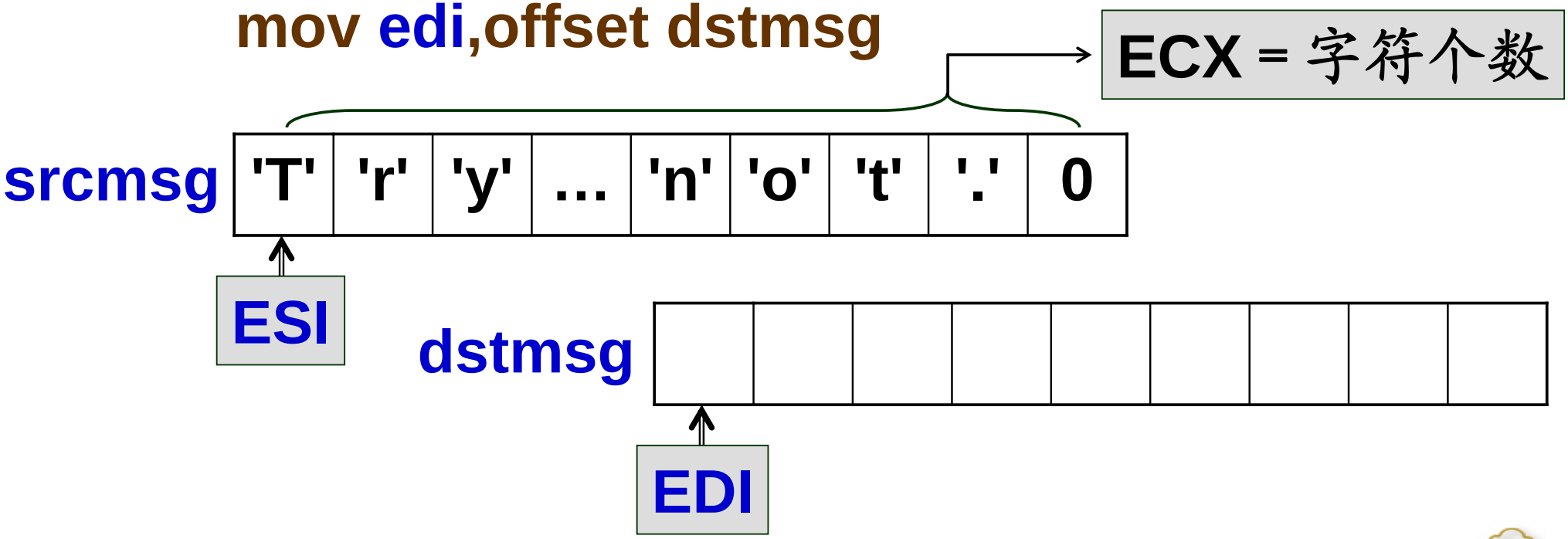
字符串复制

```
;数据段
srcmsg  byte 'Try your best, why not.',0
dstmsg  byte sizeof srcmsg dup(?)
```



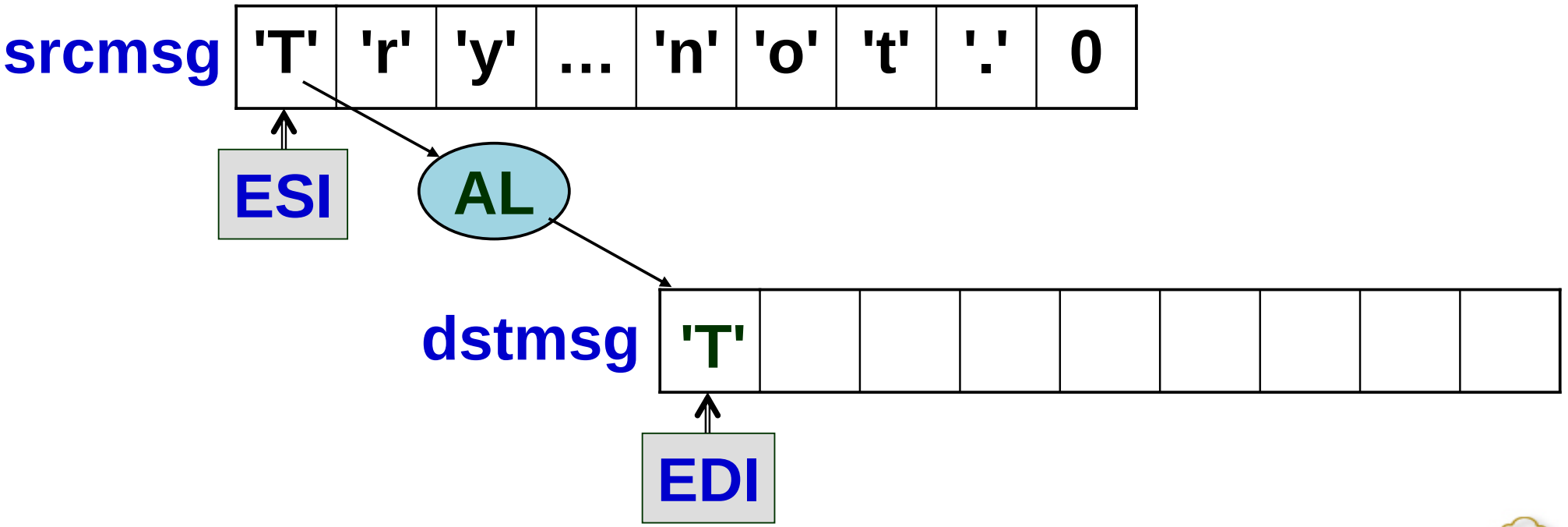
寄存器指向字符

```
mov ecx,lengthof srcmsg
mov esi,offset srcmsg
mov edi,offset dstmsg
```



字符传送

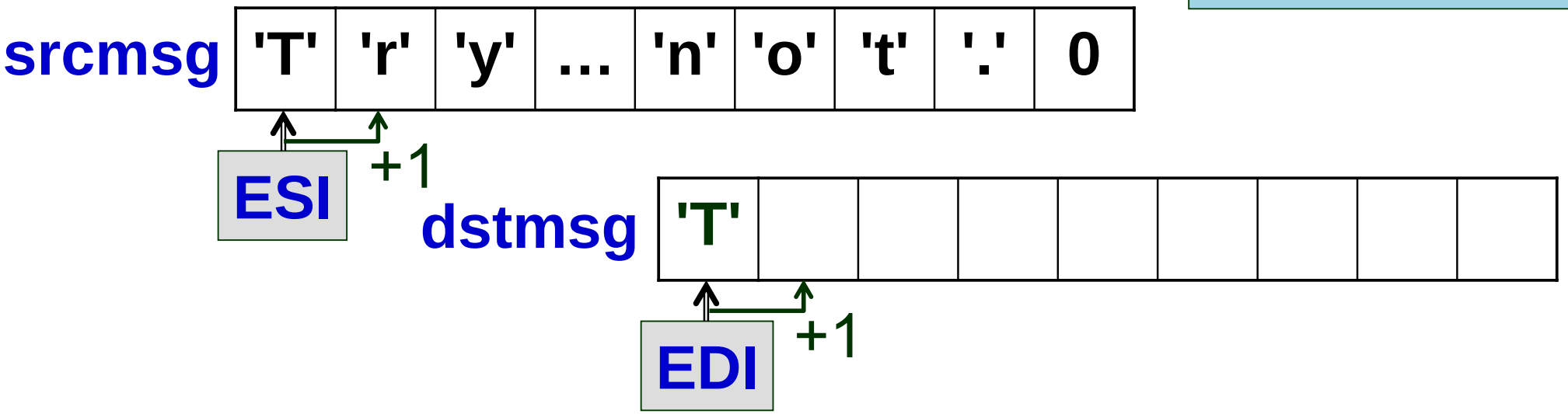
```
mov al, [esi]        ;取源串一个字符送AL
mov [edi], al        ;将AL传送给目的串位置
```



指向下一个字符

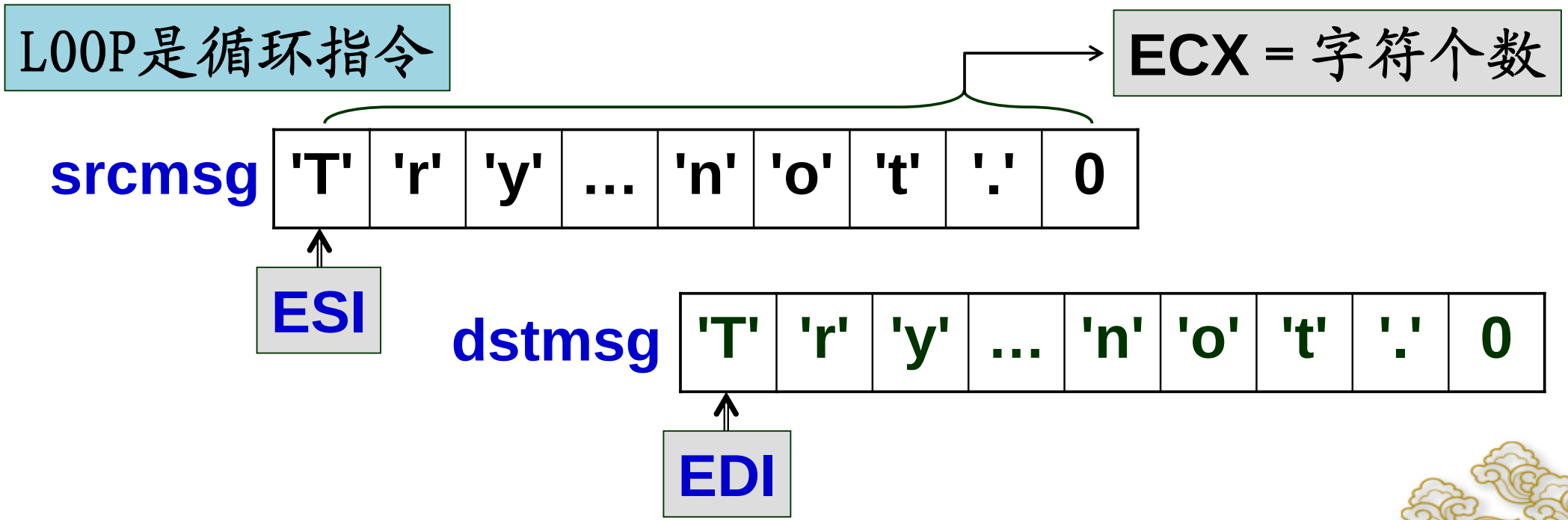
```
add esi,1          ;源串指针加1
add edi,1          ;目的串指针加1
```

ADD是加法指令



字符串复制：逐个字符传送

```
again:   mov al, [esi]      ;传送一个字符
         ...                ;指向下个字符
         loop again         ;字符个数减1，不为0继续
```



目的字符串显示

```
mov eax,offset dstmsg
call dispmsg        ;显示目的字符串内容
```

操作过程

```
D:\MASM>make32 eg0211
……
D:\MASM>eg0211.exe
Try your best, why not.
```

寄存器间接寻址程序

```
        ;数据段
srcmsg  byte 'Try your best, why not.',0
dstmsg  byte sizeof srcmsg dup(?)
        ;代码段
        mov ecx,lengthof srcmsg
        mov esi,offset srcmsg
        mov edi,offset dstmsg
again:  mov al, [esi]
        mov [edi], al
        add esi,1
        add edi,1
        loop again
        mov eax,offset dstmsg
        call dispmsg
```

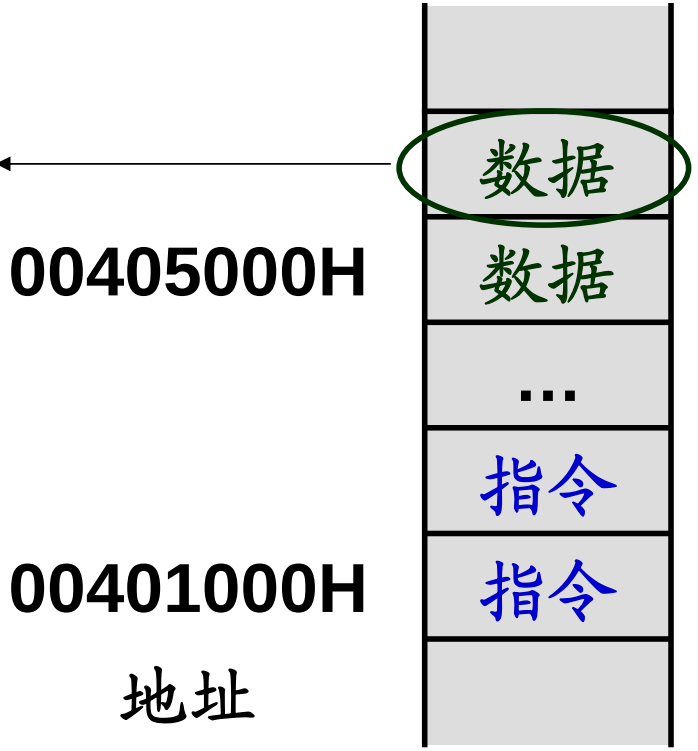
本讲总结

- 存储器的寄存器间接寻址
  - 有效地址通过寄存器提供
- 方便对数组的元素或字符串的字符进行操作
  - 数组(字符串)的首（尾）地址赋值给寄存器
  - 增减寄存器值指向不同的元素(字符)
- 寄存器间接寻址的操作数没有类型
  - 其类型由另一个操作数的寄存器或变量类型决定
  - 若另一个操作数也无类型（立即数），需显式说明

存储器的寄存器相对寻址

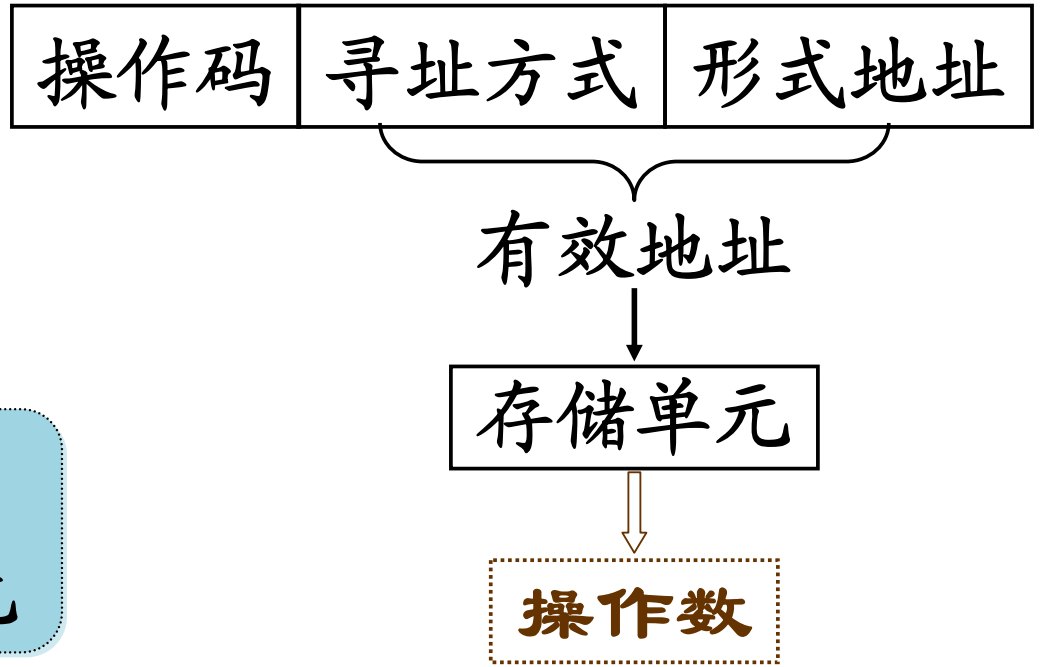
操作数在那儿！

数据来自主存储器→存储器寻址



存储器寻址

- 操作数在主存中，通过存储器地址指示
  - 指令代码表达形式地址
  - 由形式地址结合规则
    经过计算得到有效地址**EA**
    （**Effective Address**）



处理器将有效地址转换
为物理地址访问存储单元

32位存储器寻址

基址寄存器＋变址寄存器×比例＋位移量

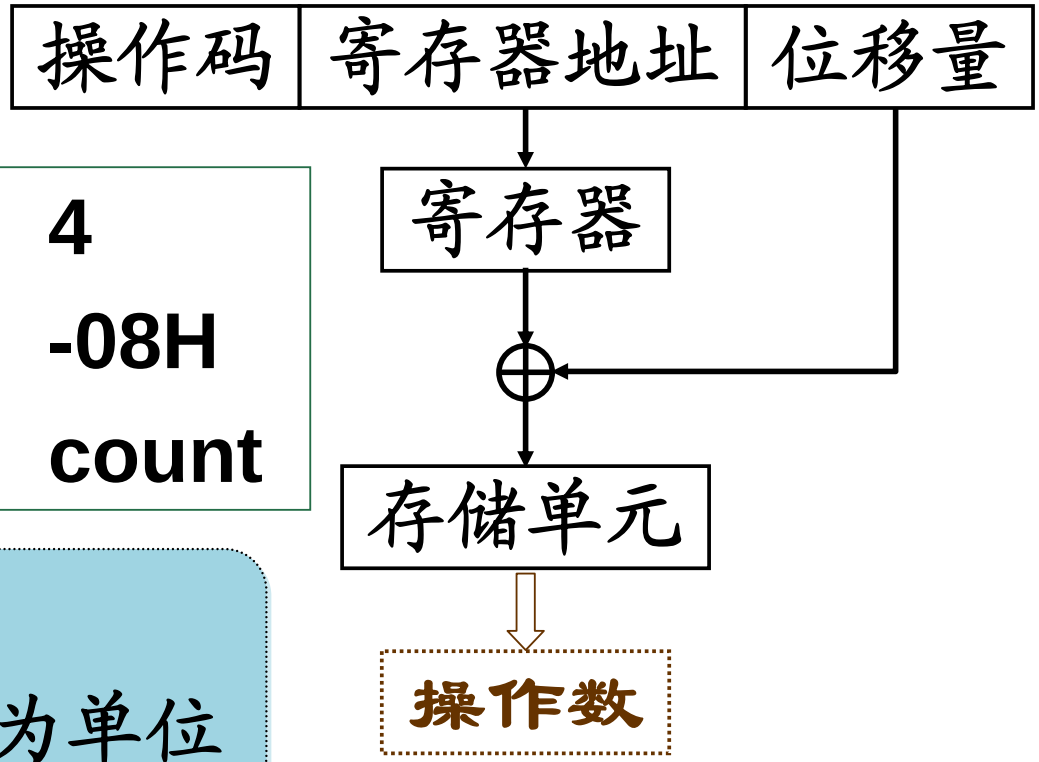
{无, EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP} ＋ {无, EAX, EBX, ECX, EDX, ESI, EDI, EBP} × {1, 2, 4, 8} ＋ {8位位移量, 32位位移量}

寄存器和位移量两部分组成

组合成多种存储器寻址方式

寄存器相对寻址

- 有效地址是寄存器内容与位移量之和
  - 寄存器要用中括号括起




```
mov esi,[ebx+4]      ;位移量：4
mov edi,[ebp-08h]    ;位移量：-08H
mov eax,count[esi]   ;位移量：count
```

主存储器采用字节编址
地址的加减是以主存字节单元为单位

寄存器相对寻址的多种表达形式

mov esi, [ebx+4] ⟶ **4[ebx]**　**[4][ebx]**

;位移量：4，无类型

mov edi, [ebp-08h] ⟶ **[-8h][ebp]**

;位移量：-08H，无类型

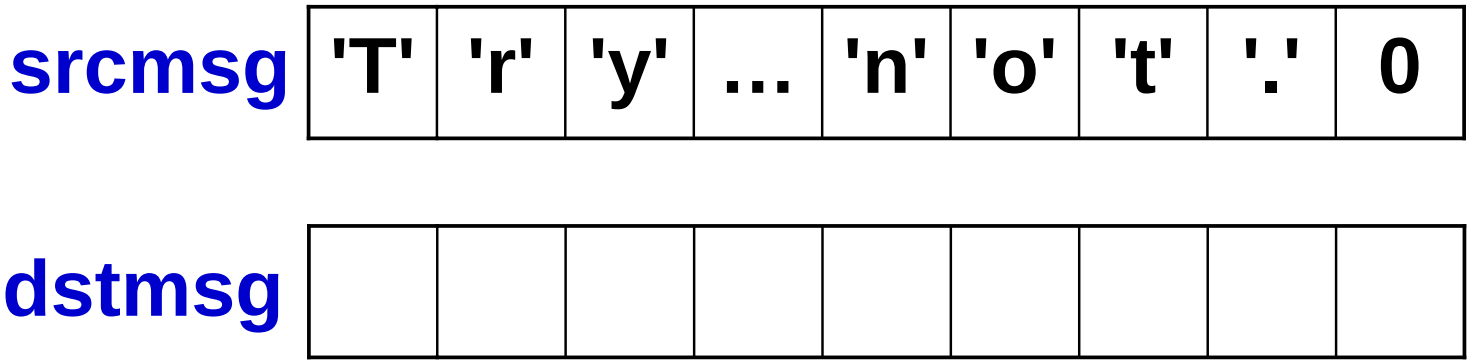
mov eax, count[esi] ⟶ **[esi+count]**　**[count][esi]**

;位移量：count，count定义的类型

;表示变量所在的偏移地址用作相对寻址的位移量

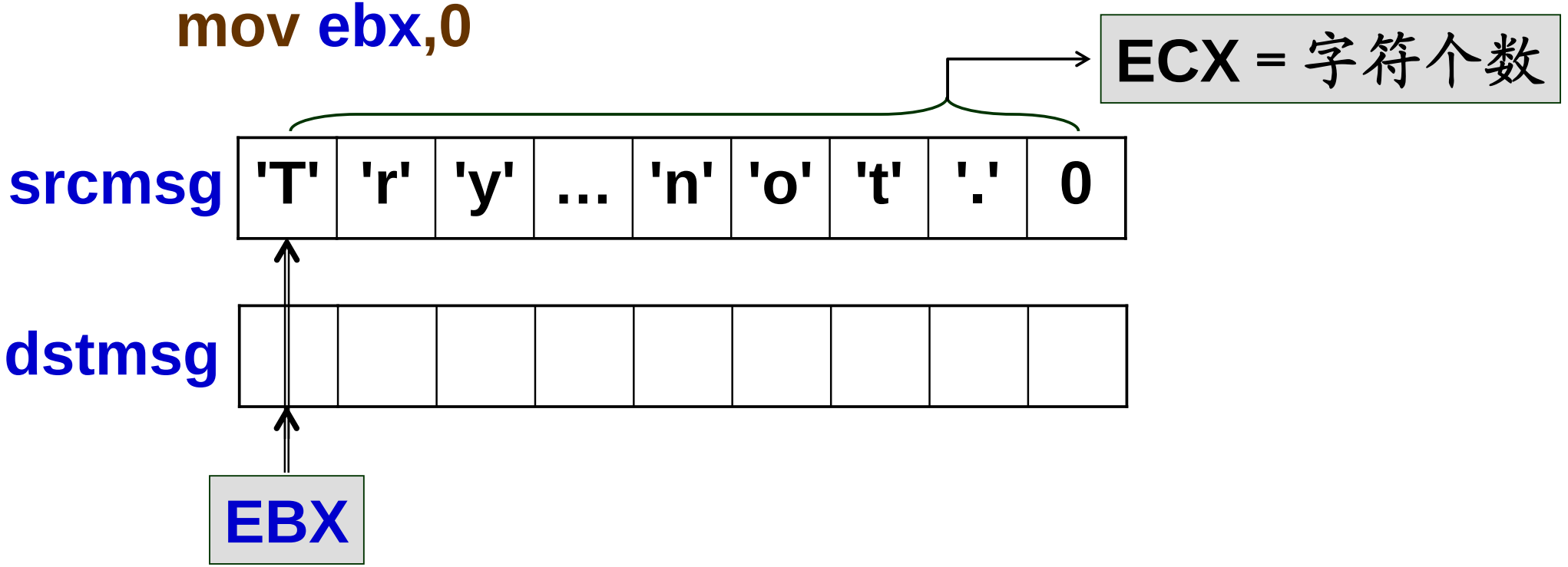
字符串复制

```
;数据段
srcmsg  byte 'Try your best, why not.',0
dstmsg  byte sizeof srcmsg dup(?)
```



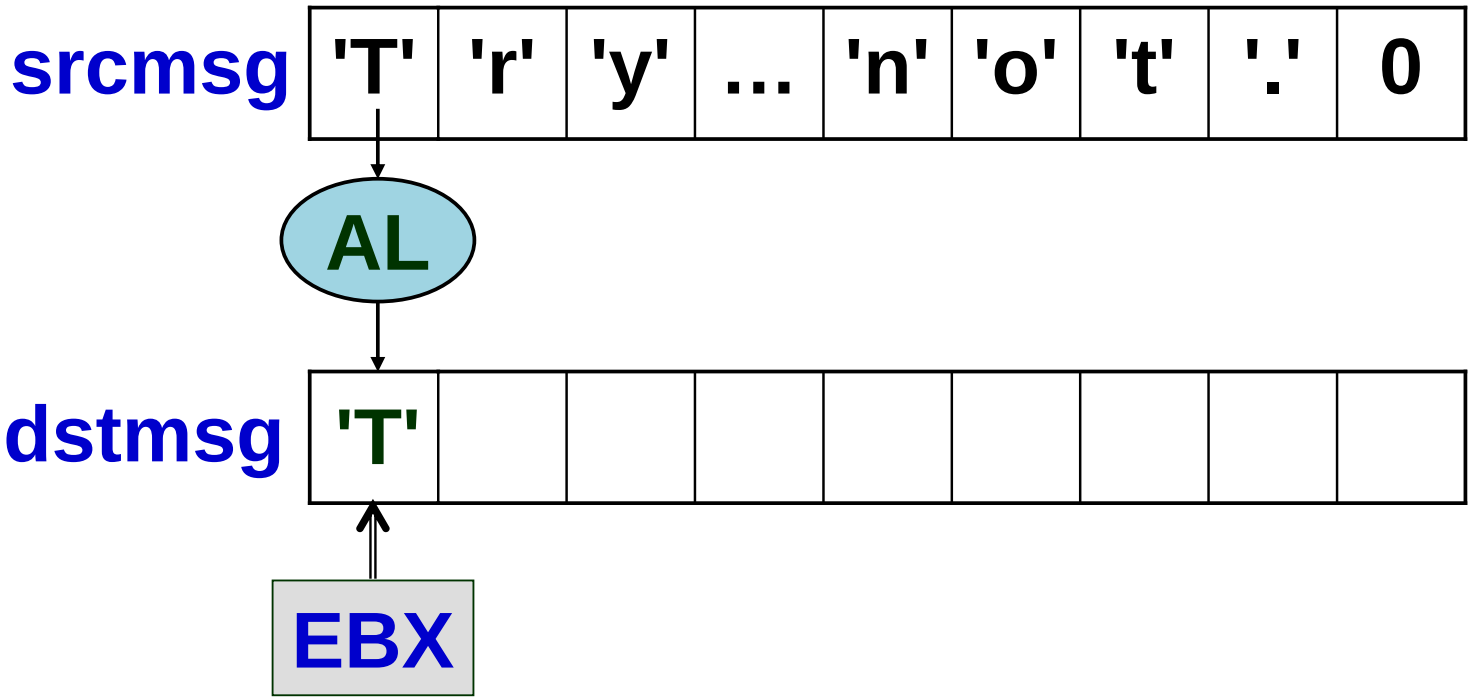
寄存器指向字符位置

```
mov ecx,lengthof srcmsg
mov ebx,0
```



字符传送

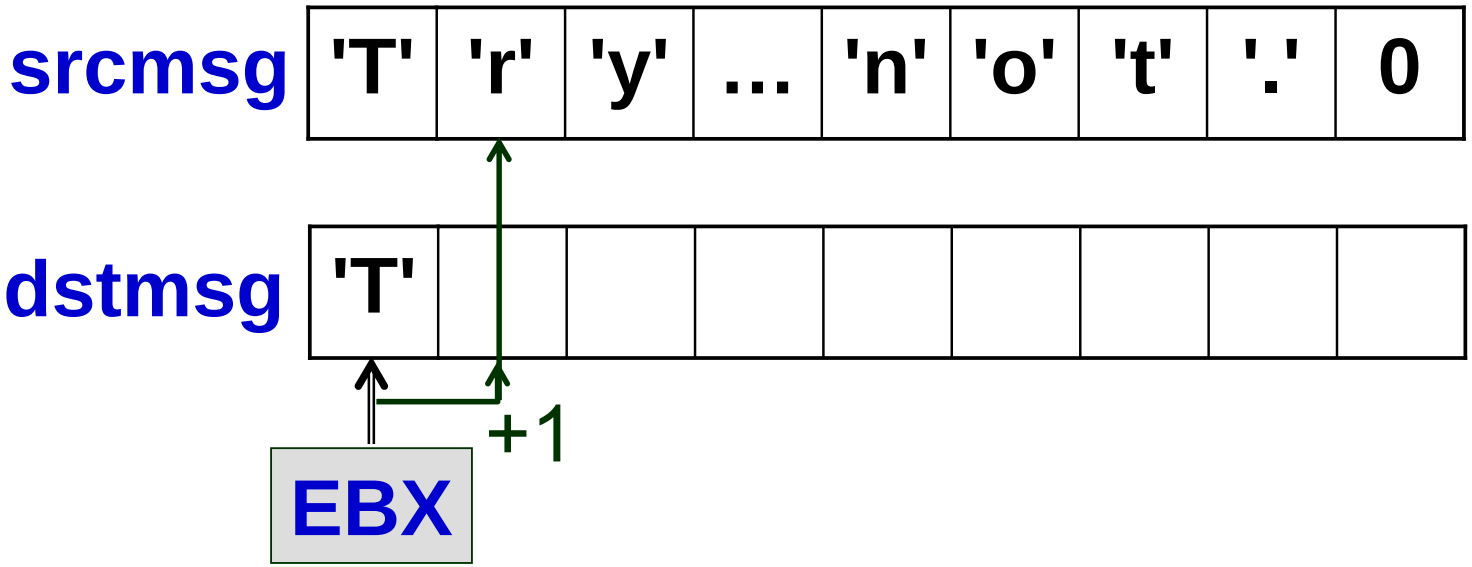
```
mov al, srcmsg[ebx]    ;取源串一个字符送AL
mov dstmsg[ebx], al    ;将AL传送给目的串位置
```



指向下一个字符

```
add ebx,1                    ;加1
```

ADD是加法指令

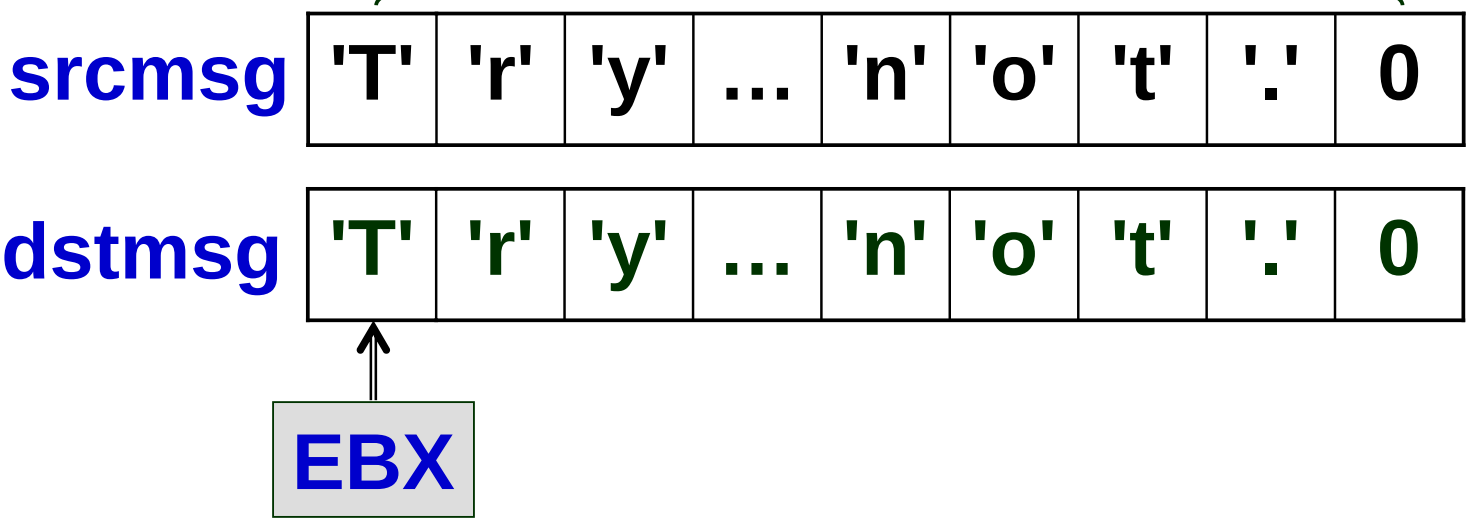


字符串复制：逐个字符传送

```
again:  mov al, srcmsg[ebx]   ;传送一个字符
        ...                   ;指向下个字符
        loop again            ;字符个数减1，不为0继续
```

LOOP是循环指令

ECX = 字符个数



目的字符串显示

```
mov eax,offset dstmsg
call dispmsg        ;显示目的字符串内容
```

操作过程

```
D:\MASM>make32 eg0212
……
D:\MASM>eg0212.exe
Try your best, why not.
```

寄存器相对寻址程序

```
        ;数据段
srcmsg  byte 'Try your best, why not.',0
dstmsg  byte sizeof srcmsg dup(?)
        ;代码段
        mov ecx,lengthof srcmsg
        mov ebx,0
again:  mov al, srcmsg[ebx]
        mov dstmsg[ebx], al
        add ebx,1
        loop again
        mov eax,offset dstmsg
        call dispmsg
```

本讲总结

- 存储器的寄存器相对寻址
  - 有效地址通过寄存器内容与位移量相加获得
- 方便对数组的元素或字符串的字符进行操作
  - 数组(字符串)的首地址作为位移量
  - 寄存器赋值0，或者元素(字符)个数
  - 增减寄存器值指向不同的元素(字符)
- 位移量是数字常量，寄存器相对寻址的操作数没有类型
  - 位移量用变量名表达，则具有变量的类型