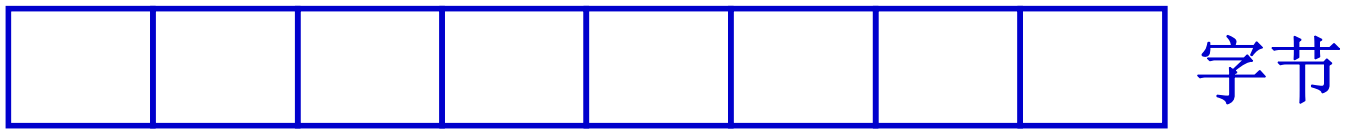
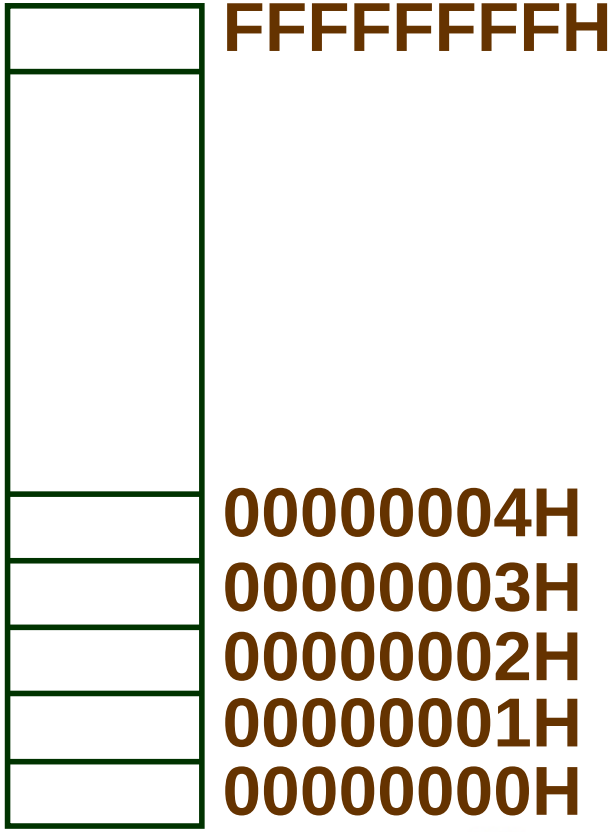
多字节数据的存储顺序

字节编址的主存储器

- 最小的存储单位：二进制位（比特位bit）
  - 最常用的存储单位：字节（Byte）
- 8个二进制位是一个字节
  - 一个存储单元保存一个字节量数据
  - 一个存储单元对应一个存储器地址





多字节数据的存储顺序

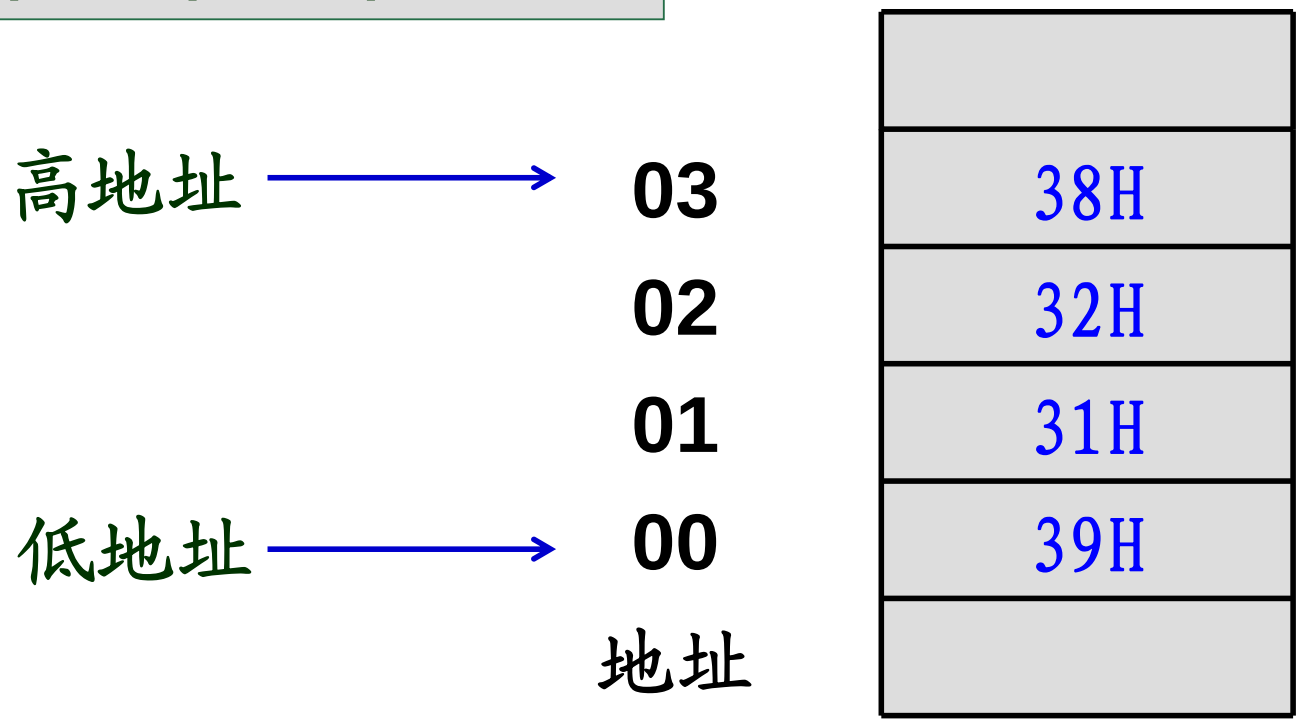
- 变量保存于字节编址的主存储器中
- 每个存储单元保存一个8位、字节量数据
- 多个字节数据顺序逐个存放在主存相邻单元

```
bvar6    byte 39h,31h,32h,38h
wvar6    word 3139h,3832h
dvar6    dword 38323139h
```

```
9128
9128
9128
```

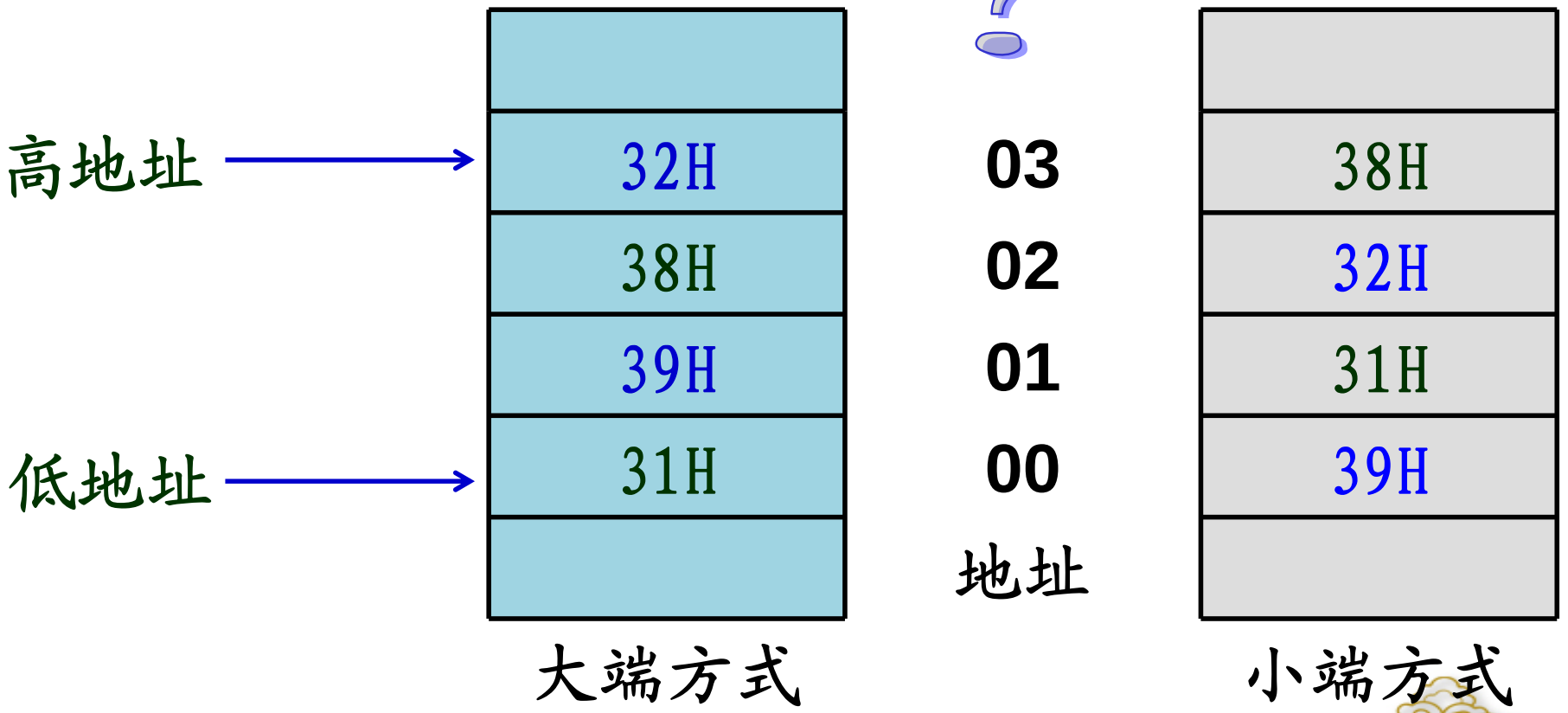
字节量数据的顺序存储

```
bvar6    byte 39h,31h,32h,38h
```



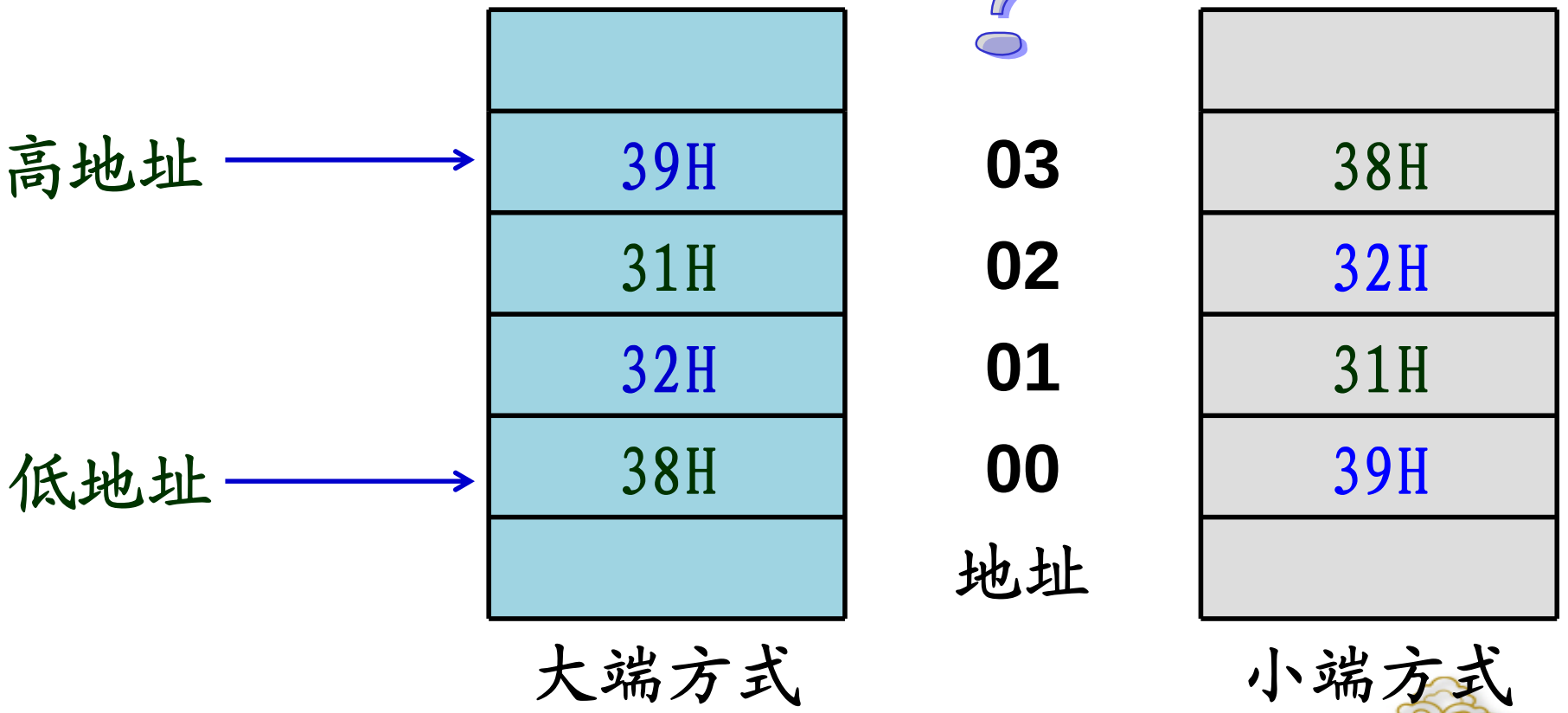
字量数据的存储顺序

wvar6　word 3139h,3832h



双字量数据的存储顺序

dvar6　　dword 38323139h



小端存储和大端存储

- 小端方式（Little Endian）
  - 高字节数据保存在高地址存储单元
  - 低字节数据保存在低地址存储单元

高对高、低对低

- 大端方式（Big Endian）
  - 高字节数据保存在低地址存储单元
  - 低字节数据保存在高地址存储单元

高对低、低对高

How to open an egg,
from the little end or the big end ?

80x86采用小端方式存储多字节数据

高对高、低对低

字节量：[00405090H]=39H

字量：[00405090H]=3139H

双字量：[00405090H]=38323139H

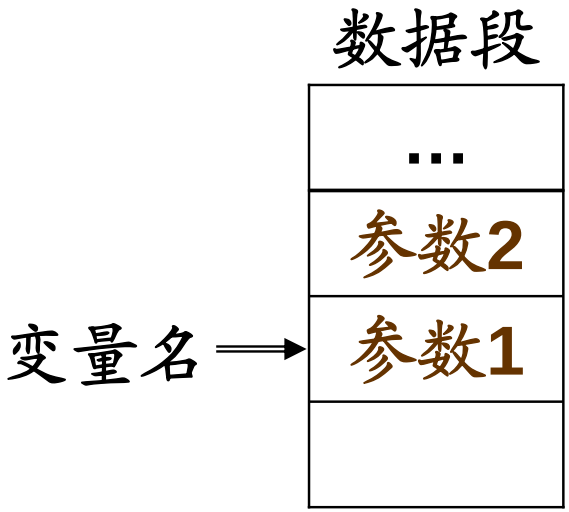
本讲总结

- 主存储器采用字节编址
  - 一个存储单元保存一个字节量数据
  - 一个存储单元对应一个存储器地址
- 对于N个字节的数据（N≥2）
  - 存储在N个连续的存储单元、具有N个存储器地址
  - 以最低地址表达该数据地址
  - 存储采用小端方式：“高对高、低对低”
    或者采用大端方式：“高对低、低对高”

变量的地址属性

变量属性

- 变量定义　**变量名　变量定义伪指令　参数, 参数...**
- 变量定义可以
  - 分配存储空间
  - 赋初值
  - 创建变量名
- 定义后的变量名具有两类属性



（1）地址属性：首个变量所在存储单元的逻辑地址

（2）类型属性：变量定义的数据单位

变量的地址属性及地址操作符

- 变量的地址属性指所在存储单元的逻辑地址
  - 含有段基地址和偏移地址
- 通过地址操作符获得变量的地址属性值

[ ]	括起的表达式作为存储器地址指针
$	返回当前偏移地址
OFFSET 变量名	返回变量名所在段的偏移地址
SEG 变量名	返回段基地址（实地址存储模型）

变量地址属性程序—1

```
                ;数据段
00000000        12 34
bvar            byte 12h,34h
                org $+10
                ;当前地址(00000002H)加10，等于0000000CH
0000000C        0001 0002 0003 0004 0005 0006
                0007 0008 0009 000A
array           word 1,2,3,4,5,6,7,8,9,10
```

机器指令

汇编语句

当前地址 0000002H

相对地址

变量地址属性程序—2

```
00000020   5678
wvar       word 5678h
=00000016   arr_size = $-array
           ;计算出当前到ARRAY变量所占存储空间
=0000000B  arr_len = arr_size/2
           ;（除以类型值）计算出变量个数
00000022   9ABCDEF0
dvar       dword 9abcdef0h
```

当前地址 00000022H

array的地址 0000000CH

数据段使用变量名
代表其偏移地址

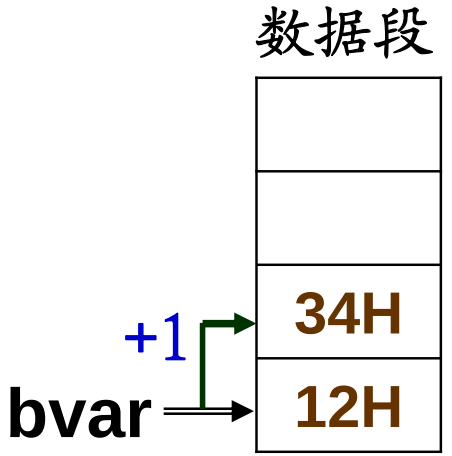
变量地址属性程序—3

```
                ;代码段
00000000  A0 00000000 R
                mov al, bvar
                ;bvar等同于[bvar]
00000005  8A 25 00000001 R
                mov ah, bvar+1
```

bvar byte 12h,34h

AL=12H

AH=34H



代码段使用变量名

代表其首个数据(变量值)

变量名加减常量

指向首个数据的前后单元

变量地址属性程序—4

0000000B 66|8B 1D 00000022 R

mov bx, wvar[2]

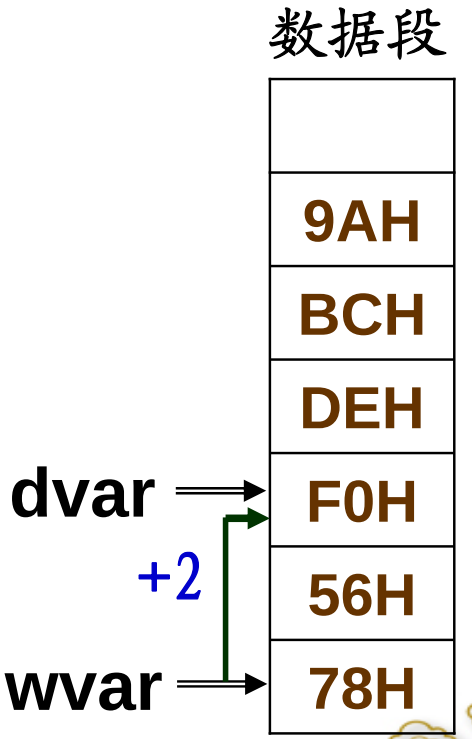
```
wvar  word 5678h
dvar  dword 9abcdef0h
```

BX=DEF0H

变量名[n] = 变量名+n

常量n表示n个存储单元

指向首个数据的前后单元



变量地址属性程序—5

```
00000012   B9 0000000B
           mov ecx, arr_len
00000017   BA 00000017 R
           mov edx, $
           ;$表示当前（指令）地址
```

ECX=0000000BH

EDX=代码段地址+17H

```
=0000000B    arr_len = arr_size/2
        ;（除以类型值）计算出变量个数
```

变量地址属性程序—6

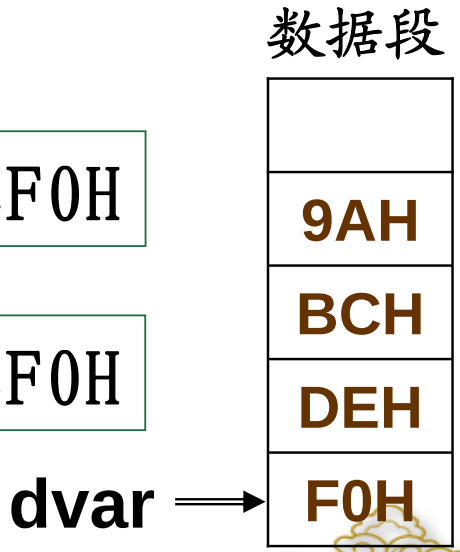
```
0000001C   BE 00000022 R
           mov esi, offset dvar
00000021   8B 3E
           mov edi, [esi]
           ;通过地址获得变量值
00000023   8B 2D 00000022 R
           mov ebp, dvar
           ;直接获得变量值
```

ESI=数据段地址+22H

```
00000022    9ABCDEF0
dvar   dword 9abcdef0h
```

EDI=9ABCDEF0H

EBP=9ABCDEF0H



变量地址属性程序—7

```
00000029    E8 00000000 E
            call disprd                ;显示通用寄存器内容
```

子程序名	DISPRD
功能说明	显示8个32位通用寄存器内容

运行结果

```
EAX=00003412, EBX=7FFDDEF0, ECX=0000000B, EDX=00401017
ESI=00405022, EDI=9ABCDEF0, EBP=9ABCDEF0, ESP=0013FFC4
```

变量地址属性程序

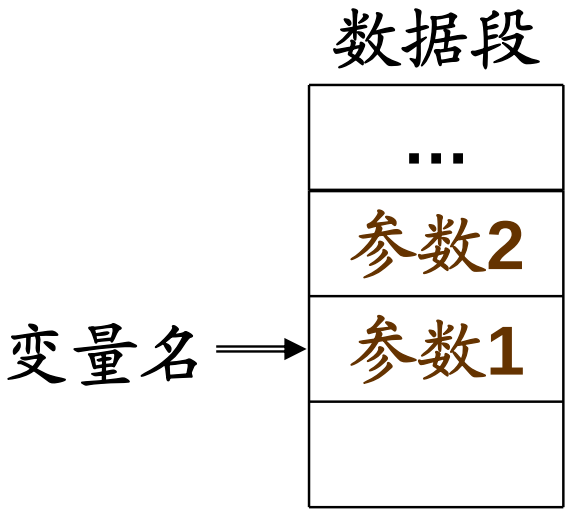
```
                ;数据段
bvar            byte 12h,34h
                org $+10
array           word 1,2,3,4,5,6,7,8,9,10
wvar            word 5678h
arr_size        = $-array
arr_len         = arr_size/2
dvar            dword 9abcdef0h
```

```
;代码段
mov al, bvar
mov ah, bvar+1
mov bx, wvar[2]
mov ecx, arr_len
mov edx, $
mov esi, offset dvar
mov edi, [esi]
mov ebp, dvar
```

变量的类型属性

变量属性

- 变量定义　**变量名　变量定义伪指令　参数, 参数...**
- 变量定义可以
  - 分配存储空间
  - 赋初值
  - 创建变量名
- 定义后的变量名具有两类属性



（1）地址属性：首个变量所在存储单元的逻辑地址

（2）类型属性：变量定义的数据单位

变量的类型属性

- 变量的类型属性表示变量定义的数据单位
- 通过类型操作符获得变量的类型属性值

变量定义	类型名	类型值（字节数）
字节变量定义BYTE	BYTE	1
字变量定义WORD	WORD	2
双字变量定义DWORD	DWORD	4

TYPE 变量名

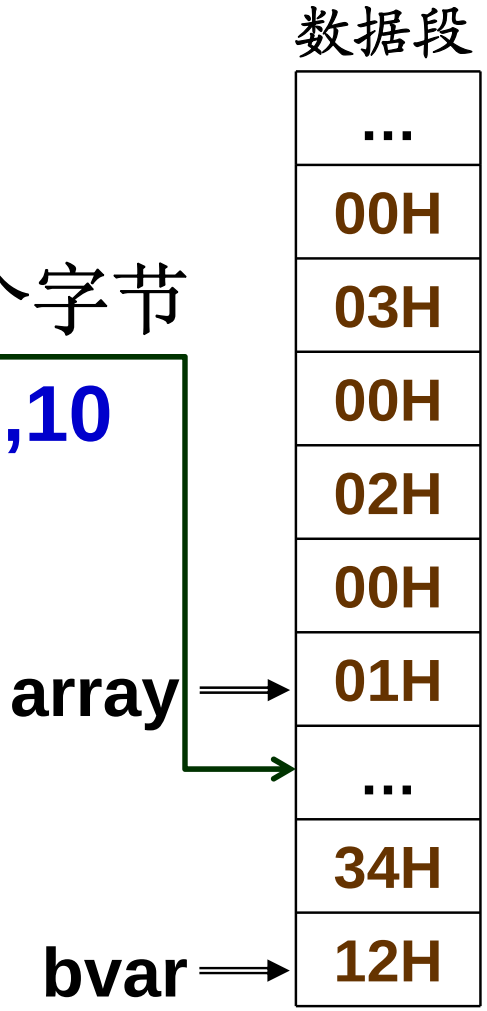
类型操作符

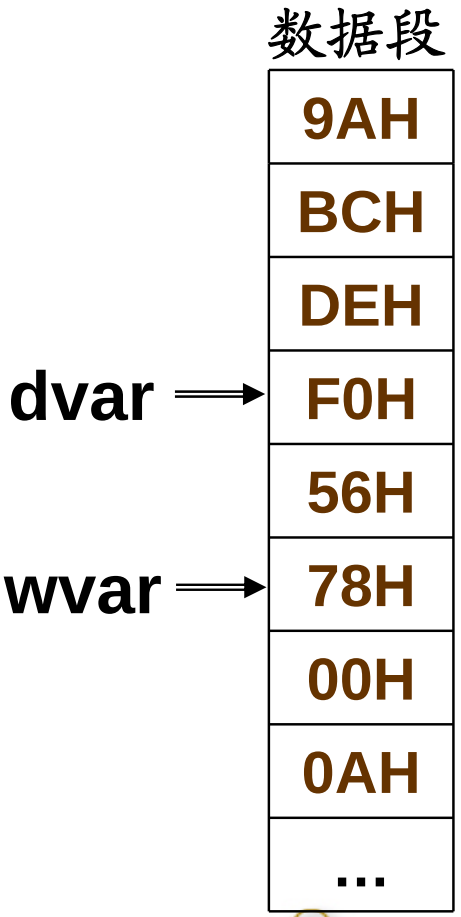
- 类型操作符使用变量名的类型属性

类型名 PTR 变量名	将变量名按照指定的类型使用
TYPE 变量名	返回占用字节空间的字量数值
LENGTHOF 变量名	返回整个变量的数据项数
SIZEOF 变量名	返回整个变量占用的字节数

变量类型属性程序—数据段

```
        ;数据段
bvar    byte 12h,34h
        org $+10        ;间隔10个字节
array   word 1,2,3,4,5,6,7,8,9,10
wvar    word 5678h
arr_size = $-array
arr_len  = arr_size/2
dvar    dword 9abcdef0h
```





变量类型属性程序—代码段1

;代码段

mov eax, dword ptr array

被定义为字类型

以双字类型访问

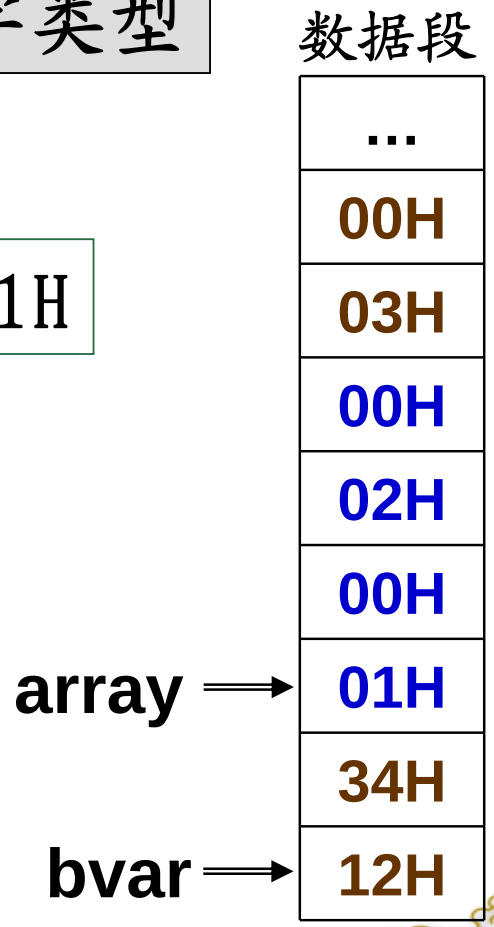
EAX=00020001H

寄存器具有确定的类型属性

8位寄存器是字节类型byte

16位寄存器是字类型word

32位寄存器是双字类型dword



变量类型属性程序—代码段2

```
mov ebx, type bvar
;获得字节变量类型值
```

EBX=00000001H

```
mov ecx, type wvar
;获得字变量类型值
```

ECX=00000002H

```
mov edx, type dvar
;获得双字变量类型值
```

EDX=00000004H

类型值就是每个数据所占的字节数

字节类型byte为1、字类型word为2、双字类型dword为4

变量类型属性程序—代码段3

mov esi, lengthof array

;获得变量的数据个数（项数）

ESI=0000000AH

mov edi, sizeof array

;获得变量所占的存储空间个数

EDI=00000014H

```
        ;数据段
array  word 1,2,3,4,5,6,7,8,9,10
```

变量类型属性程序—代码段4

```
mov ebp, arr_size
call disprd
;显示寄存器内容
```

EBP=00000016H

```
array  word 1,2,3,4,5,6,7,8,9,10
wvar   word 5678h
arr_size      = $-array
```

运行结果

```
EAX=00020001, EBX=00000001, ECX=00000002, EDX=00000004
ESI=0000000A, EDI=00000014, EBP=00000016, ESP=0013FFC4
```

变量类型属性程序

```
                ;数据段
bvar            byte 12h,34h
                org $+10
array           word 1,2,…,9,10
wvar            word 5678h
arr_size        = $-array
arr_len         = arr_size/2
dvar            dword 9abcdef0h
```

```
;代码段
mov eax, dword ptr array
mov ebx, type bvar
mov ecx, type wvar
mov edx, type dvar
mov esi, lengthof array
mov edi, sizeof array
mov ebp, arr_size
call disprd
```