



# 汇编语言伪指令



# 学习路线

- ◇ 在学习了基本指令和常用系统功能调用的基础上，我们就可以开始编写简单的汇编语言程序了。（第一个程序：**Hello World**）
- ◇ 通过程序的分析，学会用于程序结构描述和常量及变量定义的伪指令。
- ◇ 通过程序调试工具和命令的学习，更加深入灵活地学习新的指令和程序设计方法。

# 第一个源程序文件wj0301.asm

; wj0301.asm

.model small

.stack

.data

string db 'Hello, Assembly !',0dh,0ah,'\$'

.code

start: mov ax,@data

mov ds,ax

mov dx,offset string

mov ah,9

int 21h

mov ax,4c00h

int 21h

end start

存储模式



## 3.1 汇编语言源程序格式

- ◇ 完整的汇编语言源程序由段组成。
- ◇ 一个汇编语言源程序可以包含若干个代码段、数据段、附加段或堆栈段，段与段之间的顺序可随意排列。
- ◇ 能独立运行的程序至少包含一个代码段，并指示程序执行的起始点，一个程序只有一个起始点。
- ◇ 所有的可执行性语句必须位于某一个代码段内，说明性语句可根据需要位于任一段内。
- ◇ 通常，程序还需要一个堆栈段。

# 汇编语言的语句格式

(1)执行性语句——包含处理器指令的语句。

标号: 指令助记符 目标操作数,源操作数 ;注释

**begin:** mov ax, @data ;将数据段段地址送入AX

(2)说明性语句——即伪指令语句，指示源程序如何汇编，变量、子程序怎样定义等。

名字 伪指令助记符 参数, 参数, ... ;注释

**string db 'Hello, Assembly !',0dh,0ah,'\$'**

# 标号、名字与标识符

- ◇ 标号是反映硬指令位置（逻辑地址）和属性的标识符，后跟一个冒号分隔。
- ◇ 名字是反映伪指令位置（逻辑地址）和属性的标识符，后跟空格或制表符分隔，没有冒号。
- ◇ 标识符（Identifier）是名字和标号的统称，一般最多由31个字母、数字及规定的特殊符号（如 \_、\$、?、@）组成，不能以数字开头。默认情况下，汇编程序不区别标识符中的字母大小写。
- ◇ 一个源程序中，每个标识符的定义是唯一的，且不能是汇编语言采用的保留字。

# 保留字

保留字（Reserved Word）是汇编程序已经使用的标识符（也称为关键字），主要有：

- ◇ 指令助记符——例如：MOV、ADD
- ◇ 伪指令助记符——例如：DB、DW
- ◇ 操作符——例如：OFFSET、PTR
- ◇ 寄存器名——例如：AX、CS
- ◇ 预定义符号——例如：@data

## 操作数和参数

- ◇ 指令的操作数可以是立即数、寄存器和存储单元。

例：`mov ah,9`

- ◇ 伪指令的参数可以是常数、变量、表达式等，多个参数之间用逗号分隔。

例：`string db 'Hello, Assembly !$'`



# 注释

- ◇ 语句中由分号 “;” 开始的部分为注释内容，用以增加源程序的可读性
- ◇ 必要时，一个语句行也可以由分号开始作为阶段性注释
- ◇ 汇编程序在翻译源程序时将跳过注释，不对它们做任何处理

# 分隔符

- ◇ 语句的4个组成部分要用分隔符分开
- ◇ 标号后用冒号，注释前用分号
- ◇ 操作数之间和参数之间使用逗号分隔
- ◇ 其他部分通常采用空格或制表符
- ◇ 多个空格和制表符的作用与一个相同
- ◇ **MASM**支持续行符 “\”

## 3.2 常量和变量

- ◇ 汇编语言的数据可以简单分为常量和变量。
- ◇ 常量可以作为指令的操作数或伪指令的参数。
- ◇ 变量是存储于寄存器和内存中的在程序运行过程中可变的量。

### 3.2.1 常量

常量表示一个固定的值，它又分成多种形式。

1. 常数
2. 字符串
3. 符号常量
4. 数值表达式

# 1. 常数

- ◇ 用10、16、2和8进制形式表示的数值。
- ◇ 不同进制的数据用后缀字母加以区分，默认不加后缀字母的是十进制数。

十进制	由数字0~9组成，以字母D(d)结尾 (缺省情况可以省略)	100, 255D
十六进制	由数字0~9、A~F组成，以字母H(h)结尾，以字母开头的常数需要加一个前导0	64H, 0FFH 0B800H
二进制	由0和1两个数字组成，以字母B(b)结尾	01100100B
八进制	由数字0~7组成，以字母Q(q)结尾	12Q

## 2. 字符串

- ◇ 字符串常量是用单引号或双引号括起来的单个字符或多个字符。
- ◇ 例如：
  - ‘d’
  - ‘AB’
  - “Hello, Assembly !”
- ◇ 机内用每个字符对应的**ASCII**码表示和存储。

### 3. 符号常量

- ◇ 符号常量使用标识符表达一个数值。
- ◇ **MASM**提供等价机制，用来为常量定义符号名。
- ◇ 符号定义伪指令有等价“**EQU**”和等号“**=**”。

符号名 **EQU** 数值表达式

符号名 **EQU** <字符串>

符号名 **=** 数值表达式

- ◇ **EQU**用于数值等价时不能重复定义符号名，但“**=**”允许有重复赋值。例如：

**X = 7**           ； 等效于：**X EQU 7**

**X = X+5**       ； “**X EQU X+5**”是错误的

## 4. 数值表达式

- ◇ 数值表达式是由运算符连接的各种常量所构成的表达式。
- ◇ 汇编程序在汇编过程中计算表达式，最终得到一个确定的数值，所以也是常量。
- ◇ 表达式在程序运行前的汇编阶段计算，所以组成表达式的各个部分必须在汇编时就能确定。
- ◇ 汇编语言支持多种运算符（表3.3）。
- ◇ 我们经常使用的是加减乘除（+ - \* /）。
- ◇ 例如：

`mov ax, 3*4+5` ； 等价于： `mov ax, 17`



# 汇编语言运算符

运算符类型	运算符号及说明
算数运算符	+, -, *, /、MOD
逻辑运算符	AND、OR、XOR、NOT
移位运算符	SHL、SHR
关系运算符	EQ、NE、GT、LT、GE、LE



### 3.2.2 变量

- ◇ 汇编语言程序中的变量实质上是主存中存储的数据，主存单元的物理地址是固定不变的，但其中存放的数据可以改变。
- ◇ 变量需要事先定义才能使用。
- ◇ 变量定义伪指令为变量申请存储空间，并可以同时为相应的存储单元初始化。
- ◇ 变量定义以后，可以利用变量名引用变量的值。

# 1. 变量的定义

## ◇ 变量定义语句格式：

变量名 伪指令 初值表

变量名是用户自定义的标识符，表示初值表首元素的逻辑地址，常称为符号地址。变量名也可以没有。

## ◇ 初值表是用逗号分隔的参数，主要由常量、数值表达式或“？”组成。其中“？”表示未赋初值。

## ◇ 多个存储单元如果初值相同，可以用复制操作符 **DUP** 进行定义：

重复次数 **DUP**(重复参数)

## ◇ 变量定义伪指令有 **DB**、**DW**、**DD** 等（表3.4）

实例

## 1. 变量的定义

str db 'Hello, world!',0dh,0ah,'\$'

buf db 10

db ?

db 10 dup(0)

mov al,buf+1



## (1) 字节变量的定义DB (Define Byte)

- ◇ 格式：变量名 DB 初值表
- ◇ DB伪指令以字节为单位分配一个或多个内存单元，并可以将它们初始化为指定值。
- ◇ 初值表中的每个数据一定是字节量，可存储以下信息：
  - ✧ 0~255的无符号数
  - ✧ -128~+127有符号数
  - ✧ 字符串常数

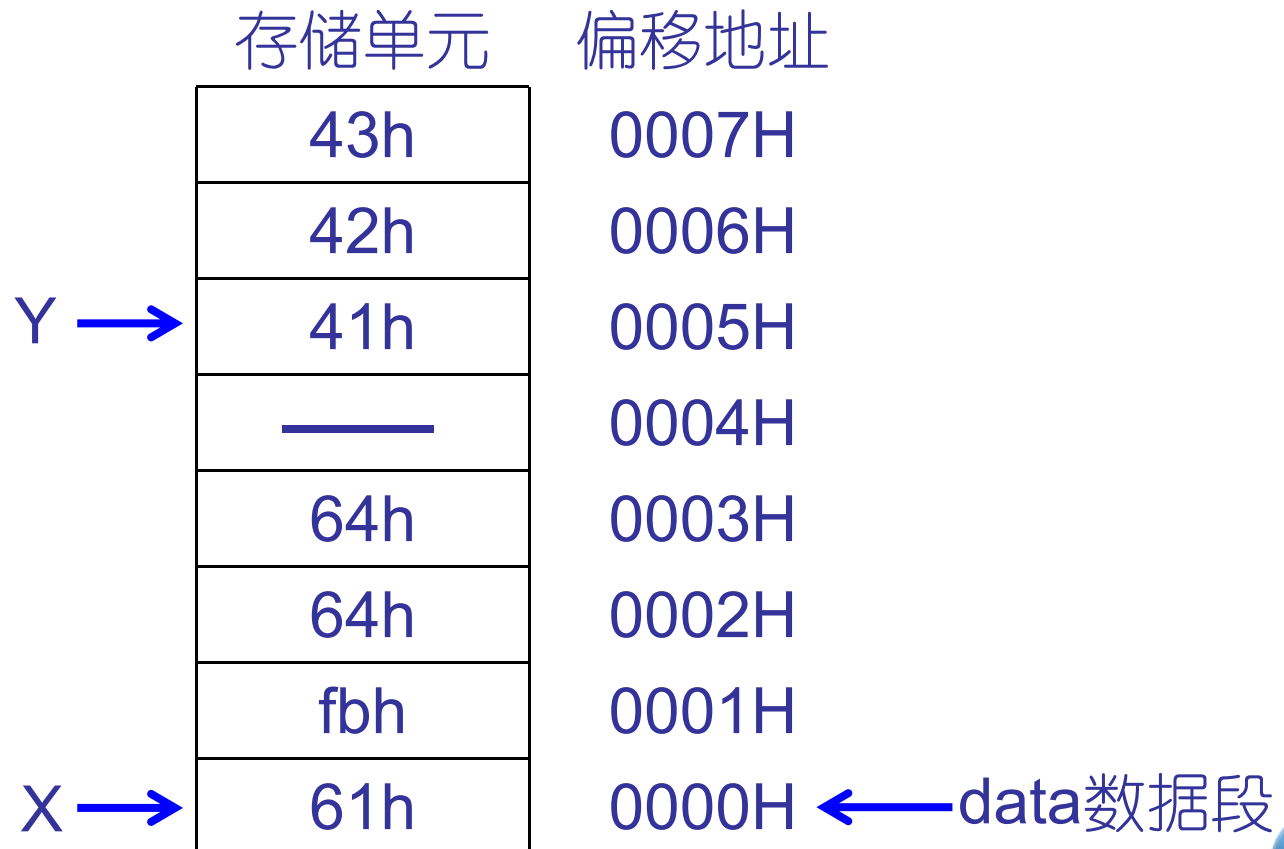
# 字节变量定义实例

.data

X db 'a',-5

db 2 dup(100),?

Y db 'ABC'



# 字节变量的应用

mov al,X

；此处X表示它的第1个数据，故 $AL \leftarrow 'a'$

dec X+1

；对X为始的第2个数据减1，故成为-6

mov Y,al

；现在Y这个字符串成为 'aBC'

.data  
X db 'a',-5  
    db 2 dup(100),?  
Y db 'ABC'

	存储单元	偏移地址
	43h	0007H
	42h	0006H
Y →	41h	0005H
	——	0004H
	64h	0003H
	64h	0002H
	fbh	0001H
X →	61h	0000H

## (2) 字变量的定义DW (Define Word)

- ◇ 格式：变量名 DW 初值表
- ◇ DW伪指令以字为单位分配内存单元，并可以将它们初始化为指定值。
- ◇ 初值表中每个数据均是字量，可用于存储以下信息：
  - ✧ 一个段地址
  - ✧ 一个偏移地址
  - ✧ 两个字符
  - ✧ 0~65535之间的无符号数
  - ✧ -32768~+32767之间的有符号数



# 字变量定义实例

.data

count

maxint

number

array

dw 8000h,?, 'AB'

equ 64h

dw maxint

dw maxint dup(0)

存储单元

偏移地址

	00	
	00	000AH
	00	
array →	00	0008H
	00	
number →	64h	0006H
	41h	
	42h	0004H
	—	
	—	0002H
	80h	
count →	00h	0000H



### (3) 双字变量的定义DD (Define Double word)

- ◇ 格式：变量名 DD 初值表
  - ◇ DD伪指令以双字为单位分配内存空间，并可以将它们初始化为指定值。
  - ◇ 初值表中每个数据是一个32位的双字量，可用来存储下列各种类型的数据：
    - ◆ 有符号或无符号的32位整数
    - ◆ 远指针：16位段地址+16位的偏移地址
- vardd DD 0,?,12345678h
- farpoint DD 00400078h

## 2. 变量的应用

- ◇ 变量名是存储单元的符号地址（逻辑地址）。
- ◇ 程序代码中
  - ◆ 通过变量名引用其指向的首个数据
  - ◆ 通过变量名加减位移量可存取位于该变量名所标识数据前后的数据。

例题3.2 wj0302.asm

### 3. 变量的定位

- ◇ 定位伪指令**ORG**控制数据或代码在内存的位置。
- ◇ 格式：**ORG** 参数
- ◇ **ORG**伪指令将当前偏移地址指针指向参数表达的偏移地址。例如：  
**ORG 100h**       ； 从100H处安排数据或程序  
**ORG \$+10**       ； 偏移地址加10，即跳过10个字节空间
- ◇ 符号“\$”称作汇编计数器，表示当前偏移地址值

Position.asm

### 3.2.3 名字和标号的属性

- ◇ 名字和标号是用户自定义的标识符。名字指向一条伪指令，标号指向一条指令。
- ◇ 名字和标号一经使用便具有两类属性：
  - (1) 地址属性：名字和标号是变量或指令所在存储单元的逻辑地址，包括段地址和偏移地址；
  - (2) 类型属性：变量名的类型可以是 **BYTE**（字节）、**WORD**（字）和 **DWORD**（双字）等；标号、段名、子程序名的类型可以是 **NEAR**（近）和 **FAR**（远），分别表示段内或段间调用。
- ◇ 汇编程序提供有关的操作符，以便获取这些属性值

# 1. 地址操作符

地址操作符用于获取名字或标号的段地址和偏移地址

表3-5 常用的地址操作符

[ ]	将括起的表达式作为存储器地址指针
\$	当前偏移地址
:	段前缀，采用指定的段地址寄存器
OFFSET 名字/标号	返回名字或标号的偏移地址
SEG 名字/标号	返回名字或标号的段地址

实例

## 地址操作符应用实例

.model small

.data

org \$+10

str db 'Hello, world!',0dh,0ah,'\$'

.code

mov ax,[bx+si+6]

mov ax,cs:[si]

mov ax, seg str

mov dx, offset str



## 2. 类型操作符

- ◇ 类型操作符对名字或标号的类型属性进行设置。

表3-6 常用的类型操作符

类型操作符	作用
类型名 <b>PTR</b> 名字/标号	将名字或标号按指定的类型使用
<b>THIS</b> 类型名	用于创建采用当前地址但为指定类型的操作数
<b>SHORT</b> 标号	将标号作为短转移处理
<b>TYPE</b> 名字/标号	返回一个字量数值，表明名字或标号的类型
<b>LENGTHOF</b> 变量名	返回整个变量的数据项数（元素数）
<b>SIZEOF</b> 变量名	返回整个变量占用的字节数



# 类型操作符应用实例

alpha dw 1234h

beta dw 50 dup(0)

adr1 equ this word

adr2 db 15h,26h,37h

mov al, byte ptr alpha ; al=34h

mov ax,adr1 ; ax=2615h

mov bx,type alpha ; bx=2

mov ax,length beta ; ax=50, 内存必须由dup分配

mov bx,size beta ; bx=100, 内存必须由dup分配





本小节到此结束  
谢谢！

# 简化段定义格式



```
.model small      ; 定义程序的存储模式（小型模式）
.stack            ; 定义堆栈段（默认是1KB空间）
.data            ; 定义数据段
.....          ; 数据定义
.code            ; 定义代码段
start: mov ax,@data ; 程序开始点
      mov ds,ax     ; 设置DS指向用户定义的数据段
      .....        ; 程序代码
      mov ax,4c00h
      int 21h       ; 程序终止点，返回DOS
      .....        ; 子程序代码
      end start     ; 汇编结束，同时指明程序起始点start
```



# 汇编语言程序的开发过程（附录B）

文本编辑器，如 记事本

编辑

源程序：文件名.asm

错误

汇编程序，如 ML.EXE

汇编

目标模块：文件名.obj

错误

连接程序，如 LINK.EXE

连接

可执行文件：文件名.exe

错误

调试程序，如 DEBUG.EXE

调试

应用程序

错误

## 开发过程1：源程序的编辑

- ◇ 源程序文件要以**ASM**为扩展名
- ◇ 源程序文件的形成（编辑）可以通过任何一个文本编辑器实现：
  - ◆ **DOS**中的全屏幕文本编辑器**EDIT**
  - ◆ 其他程序开发工具中的编辑环境
  - ◆ **Windows**中的记事本**Notepad**

## 开发过程2：源程序的汇编（MASM 6.x）

- ◇ 汇编是将源程序翻译成由机器代码组成的目标模块文件的过程
- ◇ MASM 6.x提供的汇编程序是ML.EXE：  
  
ML /c wj0301.asm
- ◇ 源程序中没有语法错误，MASM将自动生成一个目标模块文件（wj0301.obj）；否则MASM将给出相应的错误信息。这时应根据错误信息，重新编辑修改，再次汇编

## 开发过程2：源程序的汇编（生成列表文件）

- ◇ 汇编过程中，可以通过参数选择生成列表文件（**.LST**）。列表文件是一种文本文件，含有源程序和目标代码，对我们学习汇编语言程序设计和发现错误很有用
- ◇ 汇编程序**ML.EXE**可带其他参数，为了生成列表文件，命令是：

**ML /c /FI wj0301.asm**

该命令产生：模块文件**wj1301.obj**

列表文件**wj1301.lst**

## 开发过程3：目标模块的连接

- ◇ 连接程序能把一个或多个目标文件和库文件合成一个可执行程序（.EXE、.COM文件）：

**LINK wj0301.obj;**

- ◇ 如果没有严重错误，**LINK**将生成一个可执行文件（**wj0301.exe**）；否则将提示相应的错误信息。这时需要根据错误信息重新修改源程序文件后再汇编、链接，直到生成可执行文件
- ◇ **ML.EXE**汇编程序（**MASM 6.x**）可自动调用**LINK**连接程序，实现汇编和连接的依次进行

**ML wj0301.asm**



## 开发过程4：可执行程序的调试

- ◇ 经汇编、连接生成的可执行程序在操作系统下只要输入文件名就可以运行：

wj0301

- ◇ 操作系统装载该文件进入主存，并开始运行
- ◇ 如果出现运行错误，可以从源程序开始排错，也可以利用调试程序帮助发现错误
- ◇ 采用**DEBUG.EXE**调试程序：

**DEBUG wj0301.exe**

# 1. 存储模式 (Memory Model)

- ◇ 存储模式决定了一个程序的规模，也确定了子程序调用、指令转移和数据访问等的缺省属性
- ◇ 当使用简化段定义的源程序格式时，必须有存储模式.**MODEL**语句，且位于所有简化段定义语句之前。其格式为：

## .MODEL 存储模式

- ◇ .MODEL语句确定了程序采用的存储模式，MASM有7种可以选择，如表3.1所示



本课程学习过程中，均采用小型模式SMALL

## 2. 逻辑段的简化定义

### .STACK [大小]

；堆栈段定义伪指令.**STACK**创建一个堆栈段，段名是：**STACK**。可选的“大小”参数指定堆栈段所占存储区的字节数，默认是1KB（=1024=400H字节）

### .DATA

；数据段定义伪指令.**DATA**创建一个数据段，段名是：**\_DATA**。数据段名可用**@DATA**预定义标识符表示

### .CODE [段名]

；代码段定义伪指令.**CODE**创建一个代码段，可选的“段名”参数指定该代码段的段名。如果没有给出段名，则采用默认段名



一个段的开始自动结束前面的一个段



简化段定义伪指令之前，需有存储模式语句

### 3. 程序开始

- ◇ 为了指明程序开始执行的位置，需要使用一个标号（例题中采用了start标识符）
- ◇ 连接程序会根据程序起始点正确地设置**CS**和**IP**值，根据程序大小和堆栈段大小设置**SS**和**SP**值
- ◇ 连接程序没有设置**DS**和**ES**值。程序如果使用数据段或附加段，必须明确给**DS**或**ES**赋值
- ◇ 大多数程序需要数据段，程序的执行开始应是：

```
start:  mov ax,@data      ; @data表示数据段的段地址
        mov ds,ax         ; 设置DS
```

## 4. 程序终止

- ◇ 应用程序执行结束，应该将控制权交还操作系统
- ◇ 汇编语言程序设计中，有多种返回DOS的方法，但一般利用DOS功能调用的4CH子功能实现，它需要的入口参数是AL = 返回数码（通常用0表示程序没有错误）
- ◇ 于是，应用程序的终止代码就是：

```
mov ax,4c00h
```

```
int 21h
```

## 5. 汇编结束

- ◇ 汇编结束表示汇编程序到此结束将源程序翻译成目标模块代码的过程
- ◇ 源程序的最后必须有一条END伪指令  
**END [标号]**
- ◇ 可选的“标号”参数指定程序开始执行点，连接程序据此设置CS和IP值（例题中采用了start标识符）

——不要糊涂——  
**程序终止和汇编结束是两码事**

## 6. 可执行程序的结构

### ◇ DOS操作系统支持两种可执行程序结构

#### 1. EXE程序

- ◆ 程序可以有多个代码段和多个数据段，程序长度可以超过64KB
- ◆ 通常生成EXE结构的可执行程序

#### 2. COM程序

- ◆ 只有一个逻辑段，程序长度不超过64KB
- ◆ 需要满足一定条件才能生成COM结构的可执行程序（MASM 6.x需要采用TINY模式）



## 表3.1 存储模式

02:57 PM  
3/31

存储模式	特点
<b>TINY</b> (微型模式)	COM类型程序，只有一个不超过64KB的逻辑段（MASM 6.x支持）
<b>SMALL</b> (小型模式)	小应用程序，只有一个代码段和一个数据段（含堆栈段），每段不大于64KB
<b>COMPACT</b> (紧凑模式)	代码少、数据多的程序，只有一个代码段，但有多多个数据段
<b>MEDIUM</b> (中型模式)	代码多、数据少的程序，可有多多个代码段，只有一个数据段
<b>LARGE</b> (大型模式)	大应用程序，可有多多个代码段和多个数据段（静态数据小于64KB）
<b>HUGE</b> (巨型模式)	更大应用程序，可有多多个代码段和多个数据段（对静态数据没有限制）
<b>FLAT</b> (平展模式)	32位应用程序，运行在32位80x86CPU和Windows 9x或NT环境





## 例题3.2 变量的定义

； 数据段

**bvar1 db 100,01100100b,64h,'d'**

； 字节变量：不同进制表达同一个数值，内存中有4个64H

**minint = 5**

； 符号常量：minint数值为5，不占内存空间

**bvar2 db -1,minint,minint+5**

； 内存中数值依次为FFH,5,0AH

**db ?,2 dup(20h)**

； 预留一个字节空间，重复定义了2个数值20H

**wvar1 dw 2010h,4\*4**

； 字变量：两个数据是2010H、0010H，共占4个字节

**wvar2 dw ?**

；wvar2是没有初始化的字变量

## 例题3.2 变量的定义（续）

dvar dd 12347777h,87651111h,?

；双字变量：2个双字数据，一个双字空间

abc db 'a','b','c',? ；定义字符，实际是字节变量

maxint equ 0ah ；符号常量：maxint = 10

string db 'ABCDEFGHJIJ'

；定义字符串：使用字节定义DB伪指令

crlfs db 13,10,'\$'

；回车符0DH、换行符0AH和字符'\$' = 24H

array1 dw maxint dup(0)

；10个初值为0的字量，可以认为是数组

array db 2 dup(2,3,2 dup(4))


；6个字节内容依次为：02 03 04 04 02 03 04 04

## 例题3.2 变量的应用

； 代码段

mov dl,bvar1	； DL = 100
dec bvar2+1	； bvar2+1 = 4
mov abc[3],dl	； abc = ' abcd'
mov ax,word ptr dvar[0]	； 取双字到DX.AX
mov dx,word ptr dvar[2]	
add ax,word ptr dvar[4]	； 加双字到DX.AX
adc dx,word ptr dvar[6]	
mov word ptr dvar[8],ax	； 保存双字的求和结果
mov word ptr dvar[10],dx	

## 例题3.2 变量的应用（续）



```
mov cx,maxint      ; CX = 10
mov bx,0            ; BX = 0
again: add string[bx],3 ; string每个数值加3
inc bx
loop again          ; 循环
lea dx,abc          ; 从abc开始
mov ah,9            ; 09H号DOS功能调用
int 21h
; 显示结果: abcdDEFGHIJKLM
```



## 2. 类型操作符

- ◇ 格式：类型名 PTR 名字/标号
- ◇ 其中类型名可以是BYTE、WORD、DWORD（依次表示字节、字、双字）等，或者是NEAR、FAR（分别表示近、远），还可以是由结构、记录等定义的类型。



# 汇编语言的特点

- ◇ 汇编语言是一种以处理器指令系统为基础的低级程序设计语言，它采用助记符表示指令操作码，采用标识符号表示指令操作数。
- ◇ 利用汇编语言编写程序的主要优点是可以直接、有效地控制计算机硬件，因而容易创建代码序列短小、运行快速的可执行程序。
- ◇ 在有些应用领域，汇编语言的作用是不容置疑和无可替代的。
- ◇ 汇编程序设计的过程与其他高级语言程序设计大致相同。