汇编程序设计

程序结构

完整的定义格式

- 用方式选择伪指令说明执行该程序的微处理器类型
- 用段定义语句定义每一个逻辑段
- 用ASSUME说明段约定
- 用汇编结束语句说明程序结束

```
        .586                        ; 方式定义
DATA    SEGMENT USE16               ; 定义数据段
        ......
DATA    ENDS

CODE    SEGMENT USE16               ; 定义代码段
        ASSUME  CS:CODE,DS:DATA     ; 说明代码段
BEG:    MOV AX,DATA                 ; BEG为启动地址
        MOV DS,AX
        ......
        MOV AH,4CH
```

```
        INT 21H                         ; 返回DOS
CODE    ENDS
        END BEG                         ; 汇编结束
```

方式选择语句

- · 8086 -> 只汇编8086、8088指令。
- · 286 -> 只汇编8086、8088及80286实模式指令
- · 286P -> 只汇编8086、8088及80286全部指令
- · 386 -> 同·286,且汇编80386实模式指令
- · 386P -> 同·286P,且汇编80386全部指令
- · 486 -> 同·386,且汇编80486实模式指令
- · 486P -> 同·386P,且汇编80486全部指令
- · 586 -> 同·486,且汇编80486实模式指令
- · 586P -> 同·486P,且汇编80586全部指令

通常,方式选择伪指令放在程序的头部,做为源程序的第一条语句。不设置方式选择伪指令与设置·8086是等价的。

段定义语句

- 功能:是逻辑段的定界语句,源程序中每一个逻辑段都必须用段定义语句定界。
- 语句格式

```
段名    SEGMENT     定位参数     链接参数     '分类名'    段长度
        段体
段名    ENDS
```

- 定位参数
    - BYTE字节地址:表明该逻辑段的目标代码可以从任意地址开始依次存放;
    - WORD 字地址:表示该逻辑段的目标代码,从偶地址开始依次存放;
    - PARA (或者缺省) 节地址:表示该逻辑段的目标代码,从一个能被16整除的地址开始依次存放;
    - PAGE 页地址:表示该逻辑段的目标代码,从一个能被256整除的地址开始依次存放。
- 链接参数
    - 链接即为组合，通知链接程序如何将不同的模块中的同名逻辑段组合成一个段
    - PUBLIC：通知链接程序,把不同模块中具有PUBLIC属性的同名段,在满足定位方式前提下,按照指定的链接顺序进行链接,组成一个逻辑段;
    - MEMORY：MEMORY属性和PUBLIC属性是等价的
    - COMMON：
    - STACK：EXE的文件汇编源程序必须有堆栈段，否则链接时发出警告：Warning no stack segment
    - AT：逻辑段在定位时,其段基址等于表达式给出的值
    - 缺省：表明该段为独立逻辑段，不进行组合
- 分类名
    - 由用户定义，长度不超过40个字符的字符串，用单引号括起来

    - 习惯上，数据段用'DATA',代码段用'CODE',堆栈段用'STACK'
- 段长度
    - 80386和80486的新增段参数，只有高版本汇编识别
    - USE16：表示逻辑段长度最大64K，有效寻址16位
    - USE32：表示逻辑段长度可以超过64K，有效寻址32位。
    - 缺省值随编译器有所不同

段约定语句

- 格式

  ASSUME 段寄存器:段名,……,段寄存器:段名

- 功能

  通知汇编程序,寻址逻辑段使用哪一个段寄存器。

返回DOS语句

- 返回DOS最常用的方法是使用DOS系统4CH功能调用,即连续执行以下3条（或2条）指令:
- 代码格式

```
MOV     AH,4CH
MOV     AL,返回码; 不准备组织批处理文件,此条可省
INT        21H
```

汇编程序格式

EXE文件编程

- 此格式允许源程序使用多个逻辑段（包括据段、堆栈段、代码段及其它逻辑段）；
- 每个逻辑段的目标块不超过64K，适合编写大型程序

```
    ;程序实现46H+52H，存放在SUM中

.586
DATA      SEGMENT USE16
        SUM       DB  ?                    ;数据区
        DATA      ENDS
CODE      SEGMENT USE16
ASSUME  CS:CODE,DS:DATA
        BEG:        MOV     AX, DATA       ;设置DS初值
        MOV     DS, AX
        MOV     AL, 46H                    ;46H→AL
        ADD     AL, 52H                    ;46H+52H→AL
```

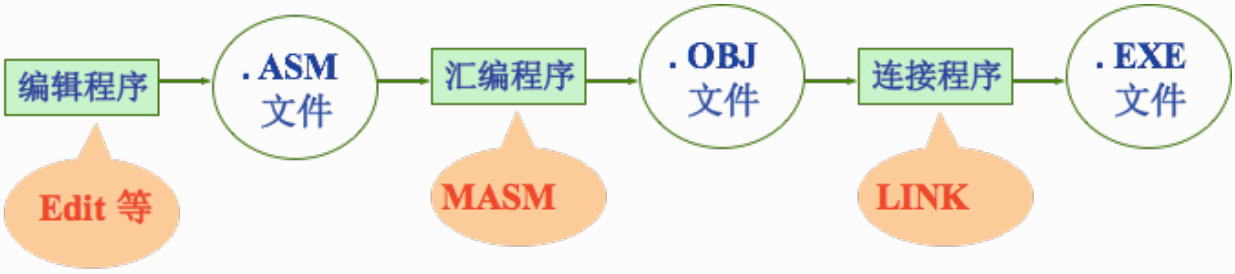
```
        MOV     SUM, AL                         ;AL→SUM
        MOV     AH, 4CH
        INT     21H                             ;返回DOS
        CODE    ENDS
        END     BEG                             ;汇编结束语句
```

COM文件编程

- COM文件的执行级别高于EXE文件,同名的BAT(批处理)文件执行级别最低。
- 源程序只允许使用一个逻辑段,即代码段,不允许设置堆栈段;
- 需使用定位ORG伪指令将程序的启动指令存放在代码段偏移地址为100H的单元。
- 代码段目标块小于64K，适合中小型程序

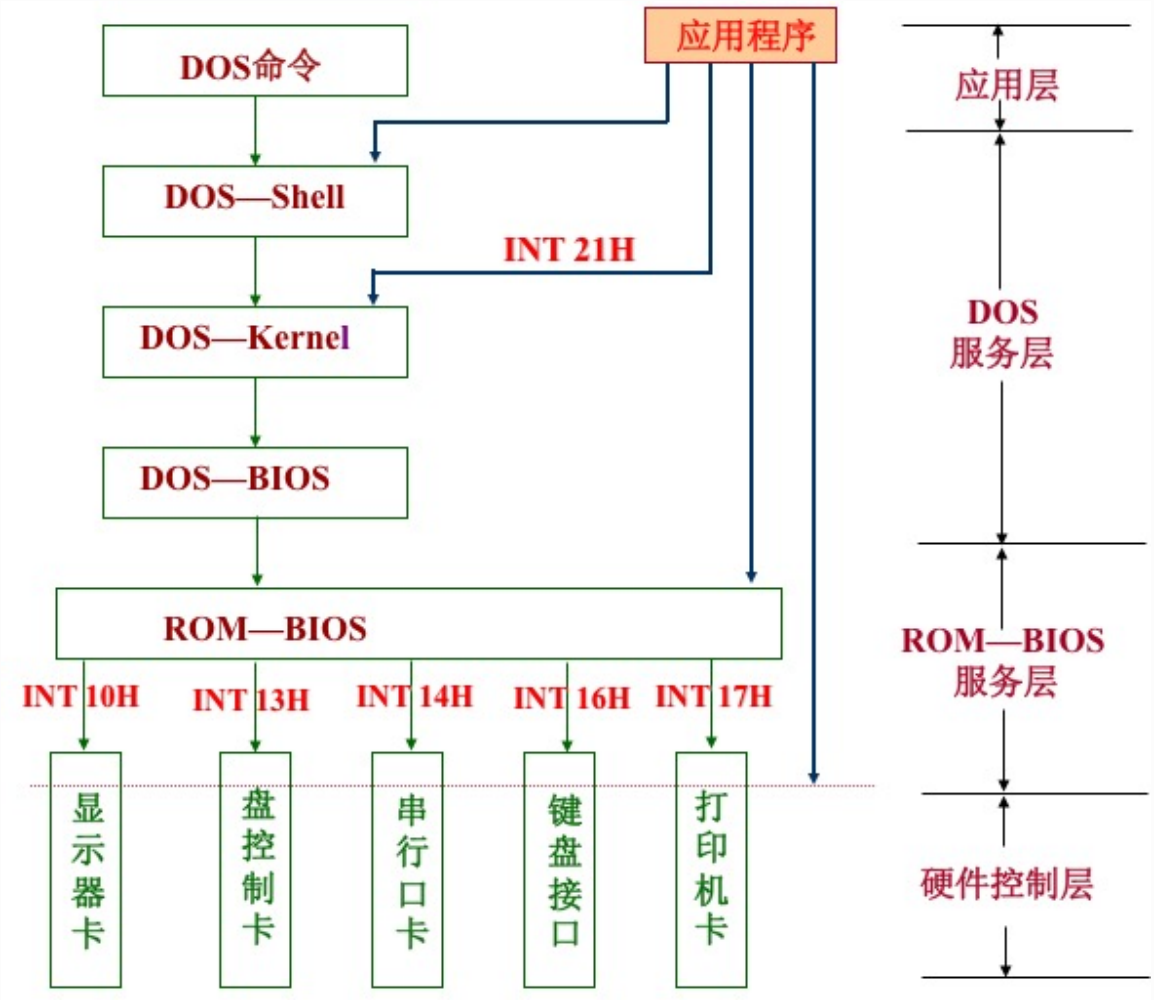
```
    ;下面将上例改造成COM文件的编程格式。

.586
CODE    SEGMENT USE16
ASSUME  CS:CODE
        ORG     100H
        BEG:    JMP     START
        SUM     DB  ?
START:  MOV     AL, 46H                 ;AL←46H
        ADD     AL, 52H                 ;AL←46H+52H
        MOV     SUM, AL                 ;SUM←AL
        MOV     AH, 4CH
        INT     21H                     ;返回DOS
CODE    ENDS
END     BEG
```

汇编开发过程

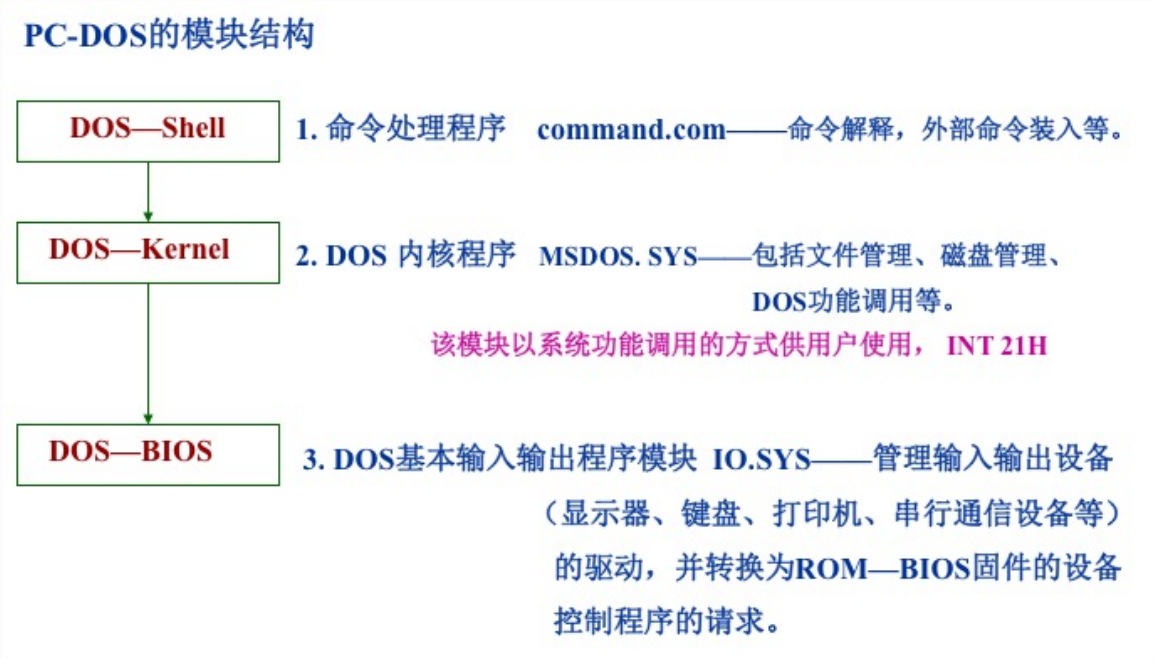


DOS功能调用

- DOS有4个组成部分，其中IBMBIO.COM为基本的I/O设备处理程序，IBMDOS.COM为磁盘文件管理程序。程序调用这些功能都称为系统功能功能调用，其中DOS功能调用即为IBMDOS.COM的调用
- DOS功能结构



- 模块结构说明

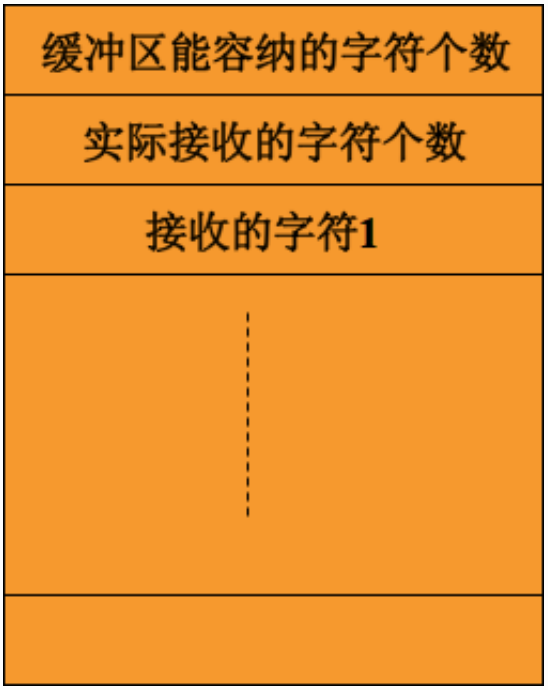


- 调用格式：

```
1. MOV  AH,功能号  ;不同的功能号调用不同的功能
2. 设置入口参数
3. INT  21H
4. 分析出口参数
```

- 功能号
  - 01H
    - 等待键入一个字符，有回显，响应Ctrl C
    - 入口参数：无
    - 出口参数：AL=按键的ASCII码,若AL=0，需要再次调用本功能
  - 02H
    - 显示一个字符，响应CtrL C
    - 入口参数：DL=待显字符的ASCII码
    - 出口参数：无
    - 显示后光标右移，到达最右侧或0AH换行，该功能破坏AL寄存器的内容
  - 07H
    - 等待键入一个字符，无回显，不响应CTRL C
    - 入口参数：无
    - 出口参数：AL=按键的ASCII码,若AL=0，需要再次调用本功能
  - 08H
    - 等待键入一个字符，无回显，响应CTRL C
    - 入口参数：无
    - 出口参数：AL=按键的ASCII码,若AL=0，需要再次调用本功能
  - 09H
    - 显示字符串，响应CTRL C
    - 入口参数：DS:DX=字符串首地址，字符串必须以'$'为结束标志
    - 出口参数：无
    - 该功能破坏AL寄存的内容

- 0AH
  - 等待从键盘输入字符串，并保存在输入数据缓冲区，回显字符
  - 入口参数：DS:DX=字符串首地址
  - 该功能以回车结束，并且将回车以0DH保存,缓冲区能容纳字符个数包括回车
  - 缓冲区保存格式应该如下图：



- 0BH
  - 查询有无键盘输入，响应CTRL C
  - 入口参数：无
  - 出口参数：AL=0 无输入，AL=FFH 有输入
- 4CH
  - 结束正在执行的程序，并返回DOS系统
  - 入口参数：AL=返回码 （或者不设置）
  - 出口参数：无
  - 关闭运行程序打开的文件，并交换内存空间

- 程序实例

```
        ;询问名字并输入名字内容
        ;FILENAME:42.ASM
        .586
DATA    SEGMENT USE16
MESG    DB      'WHAT IS YOUR NAME ? $'
BUF     DB      30
        DB      ?
        DB      30 DUP(?)
DATA ENDS

CODE    SEGMENT USE16
        ASSUME CS:CODE,DS:DATA
```

```
BEG:    MOV     AX,DATA
        MOV     DX,OFFSET   MESG
AGAIN:  MOV     AH,9
        INT     21H
        MOV     AH,0AH
        MOV     DX,OFFSET   BUF
        INT     21H
        MOV     AH,2
        MOV     DL,0AH
        INT     21H
        MOV     BL,BUF+1
        MOV     BH,0
        MOV     SI,OFFSET   BUF+2    ;???传入的是BUF+2的值?
        MOV     BYTE PTR [BX+SI],'$'
        MOV     AH,9
        MOV     DX,OFFSET   BUF+2
        INT     21H
        MOV     AH,4CH
        INT     21H
CODE    ENDS
        END     BEG
```

BIOS功能调用

- 代码格式

```
1. MOV     AH,功能号
2. 设置入口参数
3. INT     n
4. 分析出口参数
```

- 键盘功能调用（中断号n=16H）
  - 00H
    - 读取一个字符，无回显，响应CTRL C，无输入等待
    - 入口参数：无
    - 出口参数：AL=输入字符的ASCII码，若AL=0，AH=输入键扩展码
  - 01H
    - 查询键盘缓冲区
    - 出口参数：Z=0，表示有键入，键代码仍在缓冲区，此时AL=输入ASCII码，AH为扩展码
  - 02H
    - 读取当前转换键状态
    - 出口参数：AL=键盘状态字；8位从高到低分别代表Insert、Caps Lock、Numlock、Scrolllcok、Alt、Ctrl、Left-shift、Right-shift有效
- 文本方式BIOS屏幕功能调用（中断号n=10H）

- 常用的屏显编程DOS功能（2#，9#,...），BIOS功能(0EH#,13H#,...)
- 00H
  - 设置屏幕显示方式
  - 出口参数
    - AL＝0 40×25 黑白文本方式
    - AL＝1 40×25 彩色文本方式
    - AL＝2 80×25 黑白文本方式
    - AL＝3 80×25 彩色文本方式
- 02H
  - 预置光标位置
  - 入口参数：BH=显示页码，DH=行号，DL=列号
- 03H
  - 读取当前光标位置
  - 入口参数：BH=页码
  - 出口参数：CH、CL=光标顶部、底部扫描行号，DH、DL=光标所在行列号
- 05H
  - 设置当前显示页
  - 入口参数：AL=显示存储器页号0~7
  - 出口参数：显示指定显示页的字符
- 08H
  - 读取光标所在位置的字符及其属性
  - 入口参数：BH=页号
  - 出口参数：AH=光标所在位置字符属性，AL=光标所在字符ASCII码
- 0EH
  - 显示一个字符
  - 入口参数：AL=待显示字符的ASCII码
- 13H
  - 显示字符串
  - 入口参数：AL=0~3，BH=显示页码，BL=属性字，CX=串长度，DH、DL=字符串显示的起始行列号，ES:BP=待显示字符首地址

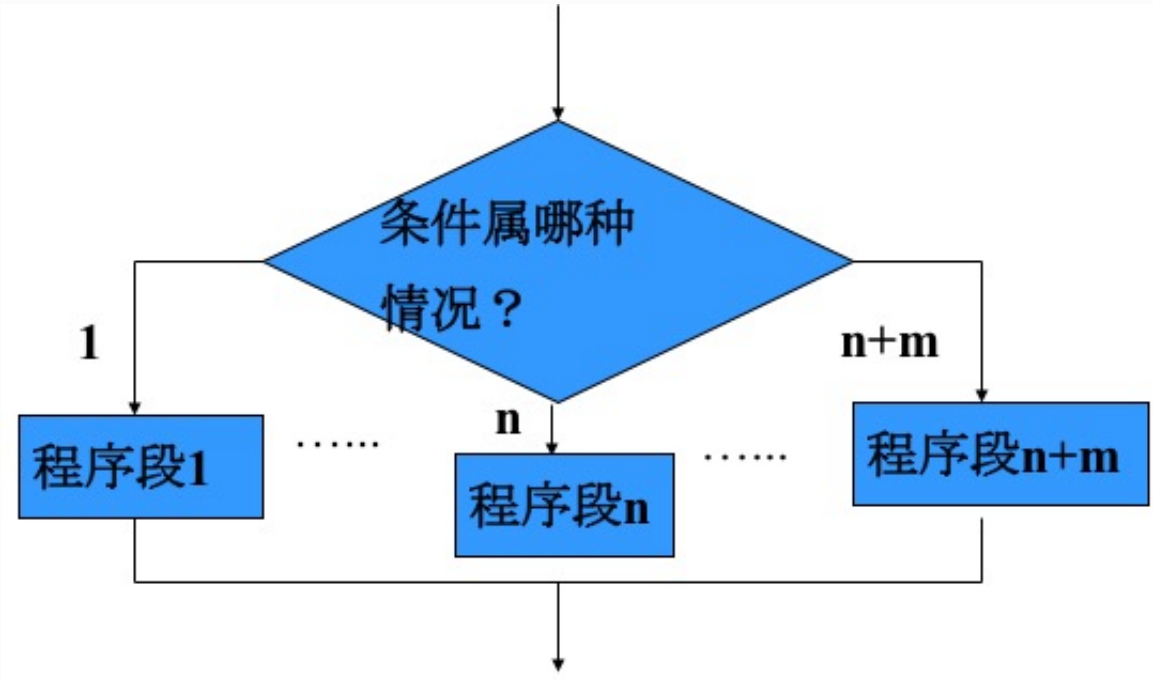
- 实例

```
        ;在屏幕左上角显示HELLO          —黑底灰白字符
        ;在屏幕中央显示 I AM A STUDENT     —红底白字
        .486
DATA    SEGMENT USE16
        MESG1   DB 'HELLO $'
        MESG2   DB 'I AM A STUDENT'
        LL      =$－MESG2
DATA    ENDS
CODE    SEGMENT  USE16
ASSUME  CS:CODE,DS:DATA,ES:DATA
BEG:    MOV   AX,DATA
        MOV   DS,AX
        MOV   ES,AX
        MOV   AX,0003H
```

```
        INT   10H
        MOV   AH,9
        MOV   DX,OFFSET  MESG1
        INT   21H
        MOV   AX,1301H
        MOV   BH,0
        MOV   BL,01001111B
        MOV   CX,LL
        MOV   DH,12
        MOV   DL,(80-LL)/2
        MOV   BP,OFFSET  MESG2
        INT   10H
        MOV   AH,4CH
        INT   21H
CODE    ENDS
END     BEG
```

分支、循环程序设计

- 设计思路
    - 分析题意，确定算法
    - 根据算法画出程序框图
    - 根据框图编写程序
    - 上机调试程序
    - 执行程序



- 常用语句整理
    - loop循环
    - JC

子程序设计

- 相对独立的程序，当程序中要多次完成某一操作时，为了简化整体程序，增强程序可读性，常常把"完成某一操作"设计成一个子程序，供调用。
- 传输参数的方式：

  ① 利用寄存器传送参数

  ② 利用堆栈传送参数

  ③ 利用某个内存单元传送参数
- 代码格式

```
        call xxx

xxx     PROC
        ...     ;子程序内容
        ...
        RET
XXX     ENDS
```

- eg.例.设ARRAY是5个字元素的数组，用子程序计算数组元素的累加和(不计进位)，并将结果存入变量RESULT中。

```
        .586
DATA    SEGMENT USE16
        ARRAY   DW 1111H,2222H,3333H,4444H,5555H
        RESULT  DW ?
DATA    ENDS

CODE    SEGMENT USE16
ASSUME  CS:CODE,DS:DATA
BEG:    MOV AX, DATA
        MOV     DS,AX
        MOV     CX,5            ;数组元素个数→CX
        MOV     BX,OFFSET ARRAY ;数组偏移地址→BX
        CALL    COMPUTE         ;调用子程序
XYZ:    MOV     RESULT, AX      ;处理出口参数
        EXIT:   MOV AH, 4CH
        INT     21H
COMPUTE PROC
        MOV     AX, 0
AGA:    ADD     AX,[BX]         ;求和
        ADD     BX,2
        LOOP    AGA
        RET                     ;返回断点XYZ
COMPUTE ENDP
        CODE    ENDS
        END     BEG
```

宏指令程序设计

- 与子程序设计的区别：同样用于程序调用，但是N次宏指令需要占据的代码内存就是N处，而子程序只占用1处，好处是宏指令的程序执行短，速度快，不需要CPU去调用子程序，直接运行代码就可以了，但是子程序设计可以占用更少的内存；宏指令可以形参和实参结合传递参数
- 定义格式

```
宏名   MACRO    <形参表>
         宏体
     ENDM
```

备注：MACRO、ENDM为宏体的定界语句，每个形式参数之间用逗号分隔，形式参数是没有值的符号，可以通过LOCAL指令消除标号重复定义的语法错误

- Eg.定位显示彩色字符串

```
     ;定位显示彩色字符串
     ;要求置显示器为彩色文本方式，并在：
     ;0行5列    显示   黑底绿色    HELLO
     ;12行36列   显示   黑底红色   WELCOME!
     ;24行66列   显示   黑底黄色   BYE_BYE


     .486
DISP     MACRO   Y, X, VAR, LENGTH, COLOR
         MOV     AH,13H
         MOV     AL,1
         MOV     BH,0                ;选择0页显示屏
         MOV     BL,COLOR            ;属性字(颜色值) →BL
         MOV     CX,LENGTH           ;串长度 →CX
         MOV     DH,Y                ;行号 →DH
         MOV     DL,X                ;列号 →DL
         MOV     BP,OFFSET VAR       ;串有效地址→BP
         INT     10H
         ENDM
EDATA    SEGMENT USE16
         SS1     DB ‘HELLO’
         SS2     DB ‘WELCOME !’
         SS3     DB ‘BYE_BYE’
EDATA    ENDS
CODE     SEGMENT USE16
ASSUME   CS:CODE,ES:EDATA
         MOV     AX,EDATA
         MOV     ES,AX
         MOV     AX,3
         INT     10H
DISP     0,5,SS1,5,2             ;0行5列显示绿色HELLO
```

```
DISP    12,36,SS2,8,4           ;12行36列显示红色WELCOME
DISP    24,66,SS3,7,0EH  ;24行66列显示黄色BYE_BYE
SCAN:   MOV     AH,1
        INT     16H
        JZ      SCAN            ;等待用户键入,无键入转
        MOV     AX,2
        INT     10H             ;恢复80×25黑白文本方式
        MOV     AH,4CH
        INT     21H
        CODE    ENDS
        END     BEG
```

- 宏体中出现的标号称为局部标号,使用LOCAL伪指令后的局部标号允许和源程序中的其它标号、变量重名

```
.486
CMPDISP MACRO   NN
        LOCAL   LAST,NEXT
        MOV     DL,0            ;DL清0
LAST:   CMP     BEN,NN          ;比较
        JC      NEXT            ;BEN < NN 转
        INC     DL              ;DL加1
        SUB     BEN,NN          ;BEN-NN→BEN
        JMP     LAST
NEXT:   ADD     DL,30H
        MOV     AH,2
        INT     21H             ;显示
        ENDM
```