



# 第四章

## 汇编语言程序设计

# 内容

- 4-0 概述
- 4-1 汇编语言程序格式
- 4-2 **MSAM**中的表达式
- 4-3 伪指令语句
- 4-4 **DOS**系统功能调用和**BIOS**中断调用
- 4-5 程序设计方法
- 4-6 宏汇编和条件汇编

## 4-0 概述

- 汇编语言是面向**CPU**指令系统的，用指令助记符、符号地址、标号等构成的程序设计语言。
- 用汇编语言编写的程序称为汇编语言（源）程序。汇编语言程序要比用机器指令编写的程序容易理解和维护，但**CPU**不能直接识别。
- 汇编语言程序的主要特点
  - 与机器相关，能够直接针对和根据系统硬件特性编写高质量的应用程序。
  - 程序目标代码占有内存少，执行速度快，效率高，适用于实时控制、实时数据处理和实时通信。
  - 程序编制和调试复杂，不易移植，无法跨平台。

● **汇编程序**：能将汇编语言程序翻译成CPU能识别机器指令序列的翻译程序。

两者易混淆

● 汇编程序的主要作用和功能

①将源程序翻译成机器语言程序。

②按用户要求自动分配存储区域（如程序代码区、数据区等）。

③自动地把各种进制数 → 二进制数。

④把字符 → ASCII码。

⑤计算表达式的值。

⑥自动对源程序进行检查，指出语法错误（如非法格式、未定义的助记符、标号、漏掉操作数等）。

## ● 汇编程序种类

①基本汇编：**ASM**（也称为小汇编）

②宏汇编：**MASM**（有各种版本，还支持宏操作，条件汇编，协处理命令等）。

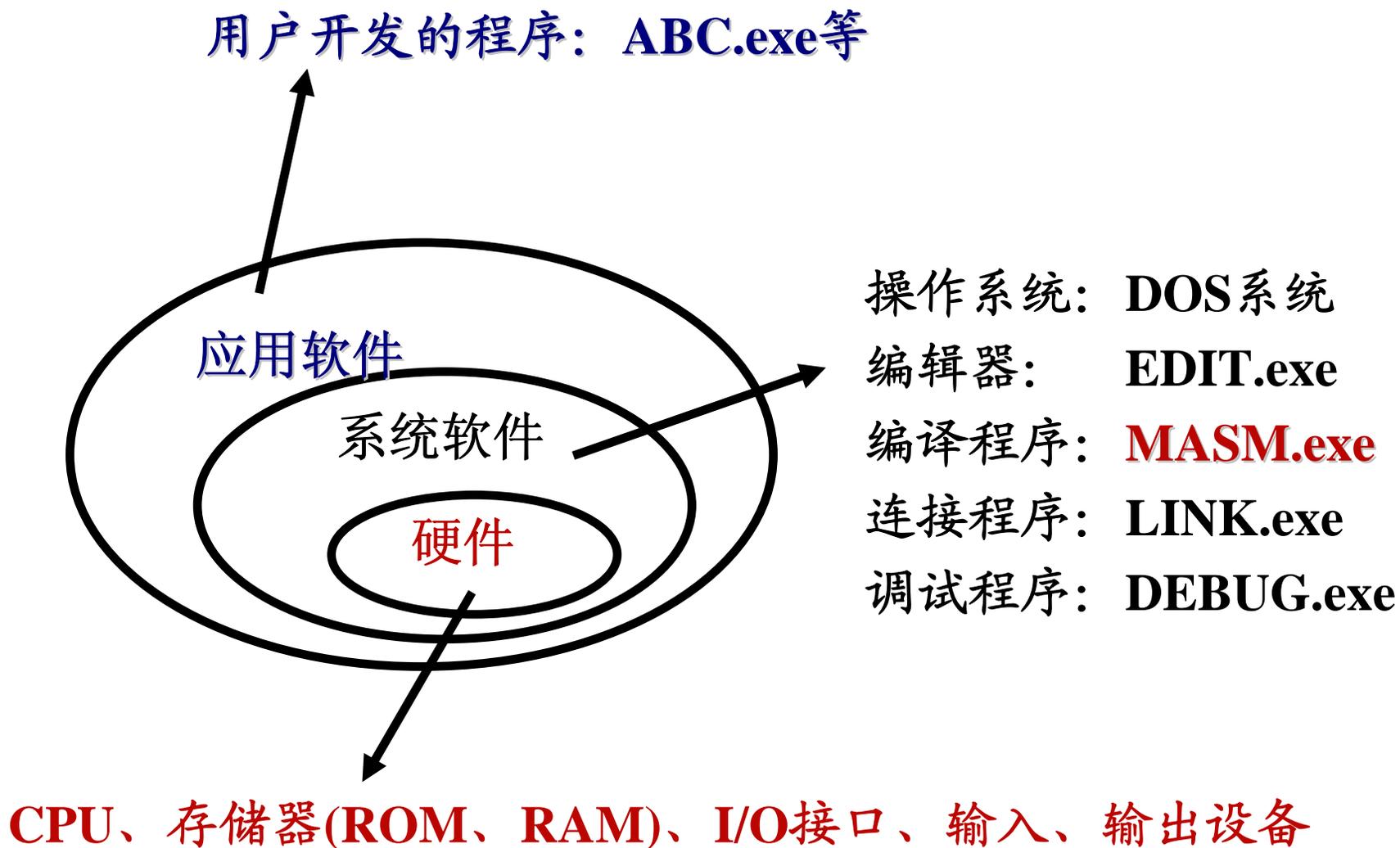
③**TASM: Turbo Assembler**，功能同**MASM**，汇编速度更快。

④**OPTASM: Optimizing Assembler**，一种优化的**MASM**。

## ● 调试工具

● **DEBUG**、**CV (CodeView)**、**TD (Turbo Debug)** 等。

# 汇编语言程序编写环境（上机实验）



# 汇编语言上机过程

D:>EDIT ABC.asm

D:>MASM ABC

有语法错，回EDIT下改程序

D:>LINK ABC

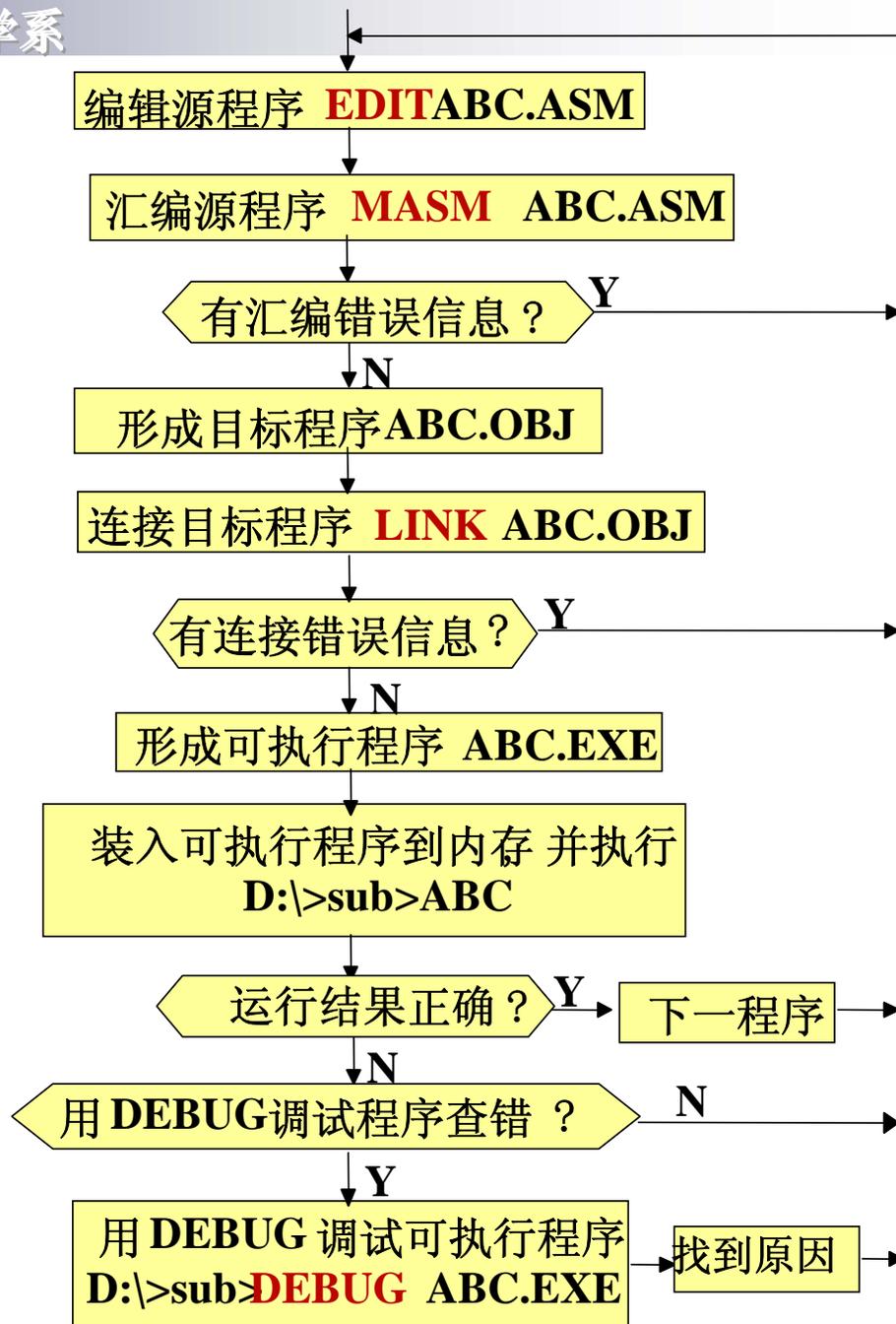
有错，回在EDIT下改程序

D:>ABC

运行结果错，回EDIT下改程序

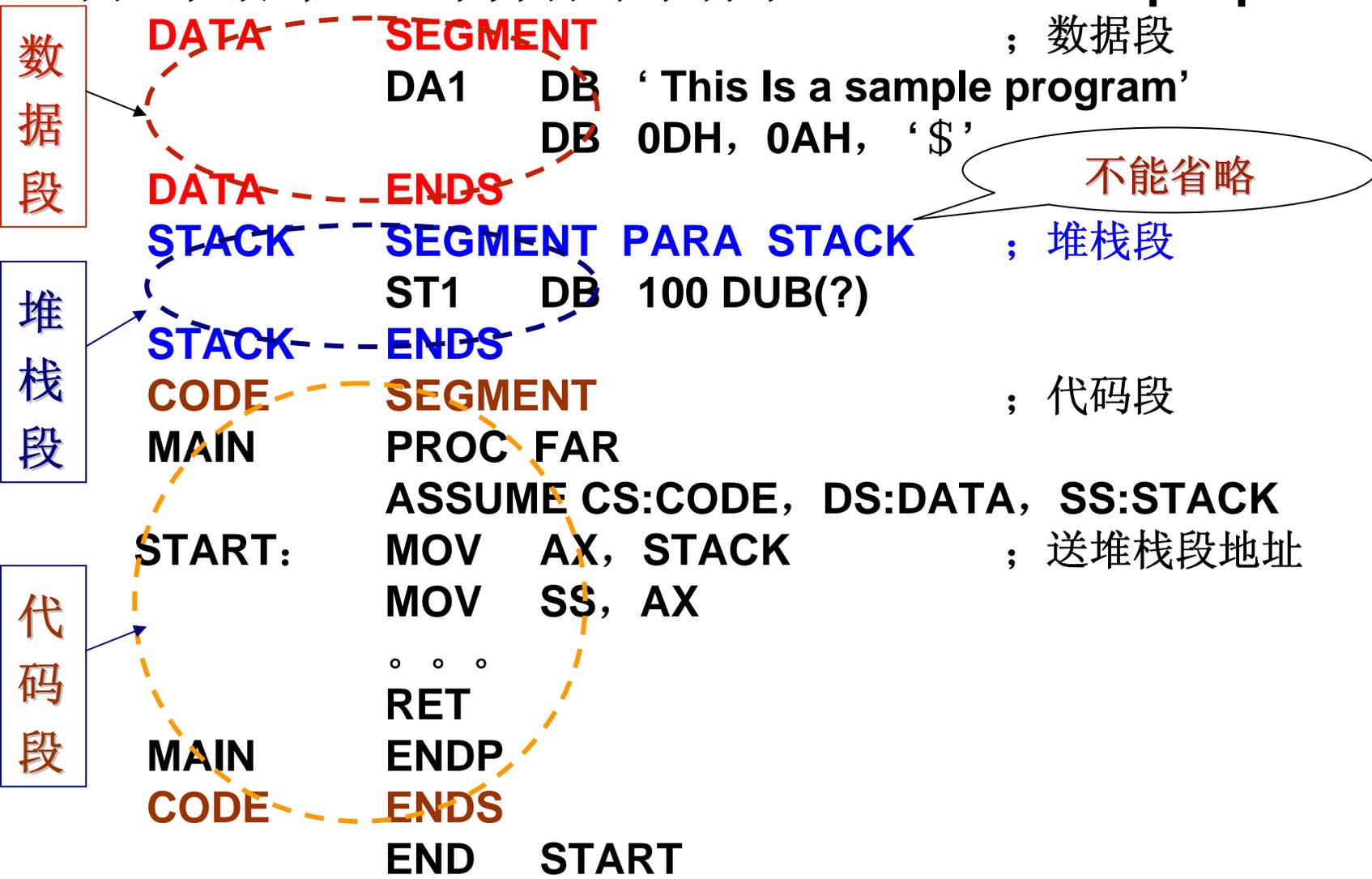
或在DEBUG下调试，找原因。

D:>DEBUG ABC.exe



# 汇编语言程序示例

例：在屏幕上显示并打印字符串“**This Is a sample pro...**”



## ● 汇编语言程序组成和特点

1. 源程序由若干个逻辑段组成，并按逻辑段组织源程序，包括：
  - ① 代码段——必不可少，源程序至少有一个代码段。
  - ② 堆栈段——建立一个堆栈区，以存放中断时的断点地址，子程序调用时断点地址及子程序间传递参数。
  - ③ 数据段和附加数据段——用来在内存中建立一个适当容量的工作区用以存放数据。
2. 每段由段定义伪指令**SEGMENT**开始，以**ENDS**，用段名区分不同的段。段定义语句的格式为：

```
;  
-----  
段名1      SEGMENT      ; 一个段的开始  
           语句1  
           语句2  
  
段名1      .....  
           ENDS         ; 一个段的结束  
;  
-----  
段名2      SEGMENT      ; 另一个段的开始  
  
           .....  
段名2      ENDS         ; 另一个段的结束  
;  
-----  
           .....      ; 其它段  
;  
-----  
           END          ; 源代码结束
```

- 在代码段起始处，用**ASSUME**伪指令说明各个段**Reg**与逻辑段的关系，并在程序起始处设置段**Reg**的初值（代码段**CS**除外，其地址由计算机自动分配）。设置段**Reg**的具体语句是：

```
MOV  AX, STACK      ; 获取堆栈段地址
MOV  SS, AX         ; 堆栈段基址送SS
MOV  AX, DATA      ; 获取数据段地址
MOV  DS, AX         ; 数据段基址送DS
```

注：STACK和DATA是段定义伪指令的段名

- 每段由若干语句行组成，每行只有一条语句且不能超过**128**个字符，允许有后续行。
- 整个源程序以**END**结束，它通知汇编程序停止汇编。**END**后的**START**标号为程序运行时的起始地址。

# 内容

- 4-0 概述
- 4-1 汇编语言程序格式
- 4-2 MSAM中的表达式
- 4-3 伪指令语句
- 4-4 DOS系统功能调用和BIOS中断调用
- 4-5 程序设计方法
- 4-6 宏汇编和条件汇编

- 汇编语言程序含三种指令语句：指令性语句、伪指令语句和宏指令语句。
  - (1) 指令性语句：可执行语句，对应**CPU**指令系统规定的一条指令。汇编时产生一一对应的机器目标代码。
  - (2) 伪指令语句：又称为指示性语句。汇编时给汇编程序提供**与硬件有关的**汇编信息，并指示汇编程序进行汇编操作，本身并不产生目标代码。
  - (3) 宏指令语句：是一种特殊的伪指令，实际上是用一条伪指令代替多条指令，以简化书写。有关宏指令将在**4-6**介绍。

# (1) 指令性语句

- 一般格式:

[指令标号:] 指令助记符 [操作数1][, 操作数2][; 注释]

- 标号——为本条指令符号地址，表示该指令的目标代码在内存中的存放地址。标号可以省略，但是标号后面必须带冒号“:”。
- 指令助记符——关键字，不能省略
- 操作数——有些指令没有，有些指令有1个或多个。多个操作数之间用逗号“,”隔开。
- 注释——可以省略，前面用“;”标注，汇编不处理。

## (2) 伪指令语句

- 一般格式:

[名字] 伪指令指示符 [操作数[, ... , 操作数]][; 注释]

- 名字——可为变量名、过程名、段名、常量名、宏名等。名字可省略，名字后面不能带冒号“:”。
- 伪指令指示符——汇编程序**MASM**规定的符号。常用的有变量定义语句、符号定义语句、...等等，伪指令将在**4-3**介绍。
- 操作数——根据伪指令要求，有些没有，有些有**1**个或多个，多个操作数之间用“,”隔开。操作数可以是常数、变量、字符串、表达式等。
- 注释——与指令性语句相同。

# 关于指令和伪指令语句中的操作数

- 操作数可以是常数、寄存器、存储器、变量、标号或表达式等。其中常数、变量和标号是三种基本数据项。
  - 1) **常数**: 无属性的确定的数据, 可用多种进制表示, 其中可打印的**ASCII**码必须用单引号' '括起来。
  - 2) **变量**: 常指存放在**M**中的数值, 在程序运行过程中可被修改。变量有三个属性:
    - (1) 段值 (**SEGMENT**);
    - (2) 段偏移地址 (**OFFSET**);
    - (3) 类型 (**TYPE**): 即长度属性。有字节型变量 (**BYTE**)、字变量 (**WORD**) 及双字变量 (**DWORD**) 等。

**3) 标号:** 指令语句的地址符号, 可作为**JMP**指令和调用指令**CALL**的目标操作数, 以确定程序转向的目标地址。标号有三个属性:

(1) 段值 (**SEGMENT**): 标号所在段的段基址。

(2) 段内偏移地址 (**OFFSET**): 偏移地址

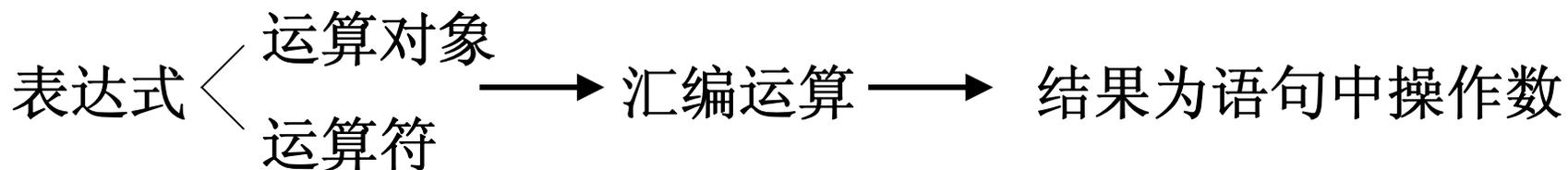
(3) 类型: 指转移指令的可转移距离, 即距离属性。其中:

- 近标号(**NEAR**), 指针长度**2**字节, 只能段内调用或转移。**NEAR**可以省略不写。
- 远标号(**FAR**), 指针长度**4**字节, 可作为其它代码段的目标地址, 实现段间调用或转移。

# 内容

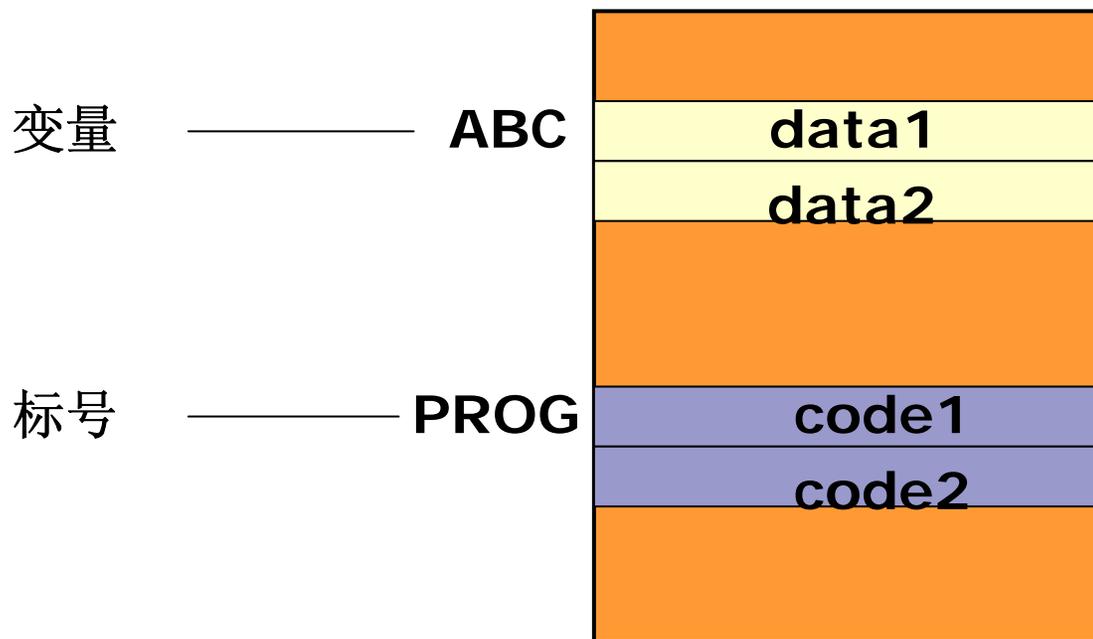
- 4-0 概述
- 4-1 汇编语言程序格式
- **4-2 MSAM中的表达式**
- 4-3 伪指令语句
- 4-4 DOS系统功能调用和BIOS中断调用
- 4-5 程序设计方法
- 4-6 宏汇编和条件汇编

**表达式**：由常数、变量或标号和运算符连接而成的式子



运算对象：常数、变量、标号

结果操作数：常数或地址（变量或标号）



# ● 宏汇编**MASM**中使用了六类运算符

(教材**P129**表**4-1**)

1. 算术运算符

2. 逻辑运算符

3. 关系运算符

也称为分析运算符

4. 数值返回运算符

5. 修改属性运算符

6. 其它运算符

也称为合成运算符

# 1. 算术运算符——

**+**、**-**、**×**、**/**、**MOD**、**SHL**、**SHR**

例：**MOV AX, 6×8** ；汇编后生成 **MOV AX, 48**

- 数值表达式中可使用所有算术运算符
- 地址表达式仅使用‘+、-’两种算术运算符  
地址表达式常用标号或变量±常量形式，  
运算结果仍为标号或变量，其3个属性中的类型及段基址属性不变，仅修改了偏移量属性。

例：**PLACE+2×3**；若**PLACE**与某存贮单元地址相关，则**PLACE+2×3**仍与某存贮单元相关。

- 算术运算符不影响标志位

例：**MOV AL, 80H+90H**；结果**AL=10H, CF=0**

## 2. 逻辑运算符 —— AND、OR、XOR、NOT

- 实现操作数按位逻辑操作，只能对常数操作，得到结果也是常数。

例：MOV AX, 80h OR 70h ; 汇编后 MOV AX, 0F0h

- **注：**逻辑运算符与逻辑指令助记符形同意不同。
  - 作为指令助记符时，是在程序运行时被执行，操作对象可以是寄存器或存储器操作数。
  - 作为运算符时，是在程序汇编时由汇编程序计算的，计算结果充当指令的某一个操作数或构成操作数的部分。（P130例4-5）

### 3. 关系运算符 ——

#### **EQ、NE、LT、GT、LE、GE**

- 在两个无符号操作数之间进行大小关系比较。
  - 两个操作数必须同是数值或同是一个段内的两个存储器地址。
  - 关系不成立（为假）则结果为“0”（假：**0**）；
  - 若关系成立（为真）则结果为全“1”（真：**0FFH**或者**0FFFFH**）；
  - 结果值在汇编时获得。

## 4. 数值返回运算符——

### OFFSET、SEG、TYPE、LENGTH、SIZE

- 用于获取段基址、偏移量或类型、长度等属性；
- 说明（**OFFSET**和**SEG**略）：
  - **TYPE**用于取标号或变量的类型：标号类型返回值为**NEAR**(-1)或**FAR**(-2)；变量类型返回值为1(字节)、2(字)、4(双字)或8(四字)。
  - **LENGTH**：当变量中使用**DUP**时，**LENGTH**返回该变量所包含的基本单元数；其它变量返回1。
  - **SIZE**用于取分配给变量的字节个数。显然有：

$$\text{SIZE} = \text{LENGTH} \times \text{TYPE}$$

## 5. 修改属性运算符—— 段Reg名: 、PTR、 **THIS**、**HIGH**、**LOW**、**SHORT**

- 用于修改变量和标号（即所谓存储器类型操作数）得属性（段属性、偏移地址属性和类型属性等）。

- 说明:

- 段Reg名+“:”——即段超越

- **PTR**——“新类型 **PTR** 原操作数”

临时修改，仅  
在当前所在的  
指令中有效。

- **THIS**——“变量/标号 **EQU THIS** 新类型”，详见P134。

- **HIGH**、**LOW**通常只对字操作数（常数）有效，将一个  
字或地址表达式分离出高位字节和低位字节；

- **SHORT**——“**SHORT** 标号”，限制转移范围在-128~  
+127之间。

## 6. 其它运算符——

( )、[ ]、·、< >、**MASK**、**WIDTH**

### ● 说明：

- ( ) 改变运算优先级。建议：尽可能多地利用 ( ) 确定优先级。
- [ ]——1) 地址表达式；如：MOV CL [BX]；  
2) 表示多重变量（数组）的下标值。
- ·、< >——在结构中专用，在4-3中介绍
- **MASK**、**WIDTH**——在记录中专用，在4-3中介绍。

## 汇编运算符总结:

算术运算符	——	<b>+</b> 、 <b>-</b> 、 <b>*</b> 、 <b>/</b> 、 <b>MOD</b> 、 <b>SHL</b> 、 <b>SHR</b>
逻辑运算符	——	<b>AND</b> 、 <b>OR</b> 、 <b>XOR</b> 、 <b>NOT</b>
关系运算符	——	<b>EQ</b> 、 <b>NE</b> 、 <b>LT</b> 、 <b>LE</b> 、 <b>GT</b> 、 <b>GE</b>
数值返回运算符	——	<b>OFFSET</b> 、 <b>SEG</b> 、 <b>TYPE</b> 、 <b>LENGTH</b> 、 <b>SIZE</b>
修改属性运算符	——	段Reg名: <b>,</b> <b>PTR</b> 、 <b>THIS</b> 、 <b>HIGH</b> 、 <b>LOW</b> 、 <b>SHORT</b>
其它运算符	——	<b>( )</b> 、 <b>[ ]</b> 、 <b>·</b> 、 <b>&lt; &gt;</b> 、 <b>MASK</b> 、 <b>WIDTH</b>

7. 运算符和操作符的优先权等级 **P136表4-3**

优 先 级		运 算 符 和 操 作 符
高 ↓	1	LENGTH, SIZE, WIDTH, MASK, ( ), [ ], < >
	2	PTR, OFFSET, SEG, TYPE, THIS, 段寄存名: (加段前缀)
	3	HIGH, LOW (操作数高、低字节)
	4	+, - (单目)
	5	*, /, MOD, SHL, SHR
	6	+, - (双目)
	7	EQ, NE, LT, LE, GT, GE
低	8	NOT
	9	AND
	10	OR, XOR
	11	SHORT

# 汇编语言程序示例：多个连续字节单元的累加

**DATA SEGMENT** ;定义数据段

**D1 DB 5 DUP ( ? )** ;定义连续的字节单元

**CC EQU \$ - D1** ;元计数, CC等于常量5

**SUM DW 0** ;定义字变量SUM, 保留结果

**DATA ENDS** ;数据段定义结束

**STACK SEGMENT PARA STACK 'STACK'** ;定义堆栈段

**DB 100 DUP ( ? )** ;100字节的栈空间

**STACK ENDS** ;堆栈段定义结束

**CODE SEGMENT** ;定义代码段

**ASSUME DS:DATA, SS:STACK, CS:CODE** ;对各段进行说明

**BGN: MOV AX, DATA** ;DS初始化为DATA

**MOV DS, AX** 指令

**LEA SI, D1** ;预置源数据的指针

定义存储单元的数据

定义堆栈

作堆栈用的存储单元

MOV CX, CC	; 预置次数
LODS BYTE PTR [SI]	; 取一个数并修改指针
MOV AH, 0	; 清AX的高字节
CLC	; 清CF位
AGAN: ADC SUM, AX	; 累加
LODS BYTE PTR [SI]	; 再取数并修改指针
LOOP AGAN	; 控制循环
MOV AH, 4CH	; 完成, 利用DOS中断
INT 21H	; 调用返回DOS
CODE ENDS	; 代码段结束
END BEGIN	; 整个程序结束

段定义

汇编程序  
结束

# 内容

- 4-0 概述
- 4-1 汇编语言程序格式
- 4-2 MSAM中的表达式
- 4-3 伪指令语句
- 4-4 DOS系统功能调用和BIOS中断调用
- 4-5 程序设计方法
- 4-6 宏汇编和条件汇编

# 伪指令语句类型

1. 数据定义语句
2. 表达式赋值语句
3. 段定义语句
4. 段分配语句
5. 过程定义语句
6. 程序开始和结束语句

基本常用语句

- I. 群定义语句
- II. 结构定义语句
- III. 记录定义语句

复杂数组定义语句

# 一、数据定义语句

- 定义存储器单元的伪指令。为程序分配指定数目的存储单元，并根据情况进行初始化。
- 这类伪指令类似高级语言中的变量定义，因此指令中定义的标识符就是所谓的变量名。
- 格式1：变量名 助记符 操作数，操作数，…
- 格式2：变量名 助记符 **n DUP** (操作数，… )

助记符：	<b>DB</b>	定义字节	<b>DW</b>	定义字
	<b>DD</b>	定义双字	<b>DQ</b>	定义四字
	<b>DT</b>	定义十字节		

变量名—符号表示,可省略。可作为其后的第一个字节存储单元的符号地址。

操作数—常数,字符串,变量,标号,表达式

**n DUP( )**—**n**为整数,表示括号中操作数重复次数。

●注意:

- ① 定义多字节字符串必须用**DB**, **DW**只允许包含两个字符。
- ② 在数据段预先定义足够的不确定值变量,为保存运算结果预留存储空间。
- ③ **DW**和**DD**可将变量或标号的逻辑地址存入存储器(教材P-138~139、例4-25)。

## 例1:

```
STT DB 'How are you? '
```

```
M1 DW 10 DUP ( ? )
```

```
M2 DB 4 DUP ( 1, 3 DUP ( 0AH ) ) ; DUP嵌套
```

```
ADR1 DD STT ; STT段址和偏移量→ADR1
```

```
MOV AX, WORD PTR ADR1+2 ; 高位地址是STT段址
```

```
MOV DS, AX
```

```
MOV SI, WORD PTR ADR1 ; 低位地址是STT偏移量
```

## 汇编后

**M1**为20个字节的连续地址变量，**M1**为第一个字节的首地址。

**M2**为: 1, 0AH, 0AH, 0AH, 1, 0AH, 0AH, 0AH,  
1, 0AH, 0AH, 0AH, 1, 0AH, 0AH, 0AH

**DS=SEG STT**

**SI=OFFSET STT**

例2: 设数据段的起始地址为02000H

**DATA SEGMENT**

**DBYTE DB 10, 10H**

**DWORD DW 100, 100H**

**DDWORD DD 12345678H**

**DQWORD DQ 1234567890ABCDEFH**

**DBS DB 'AB'**

**DWS DW 'AB'**

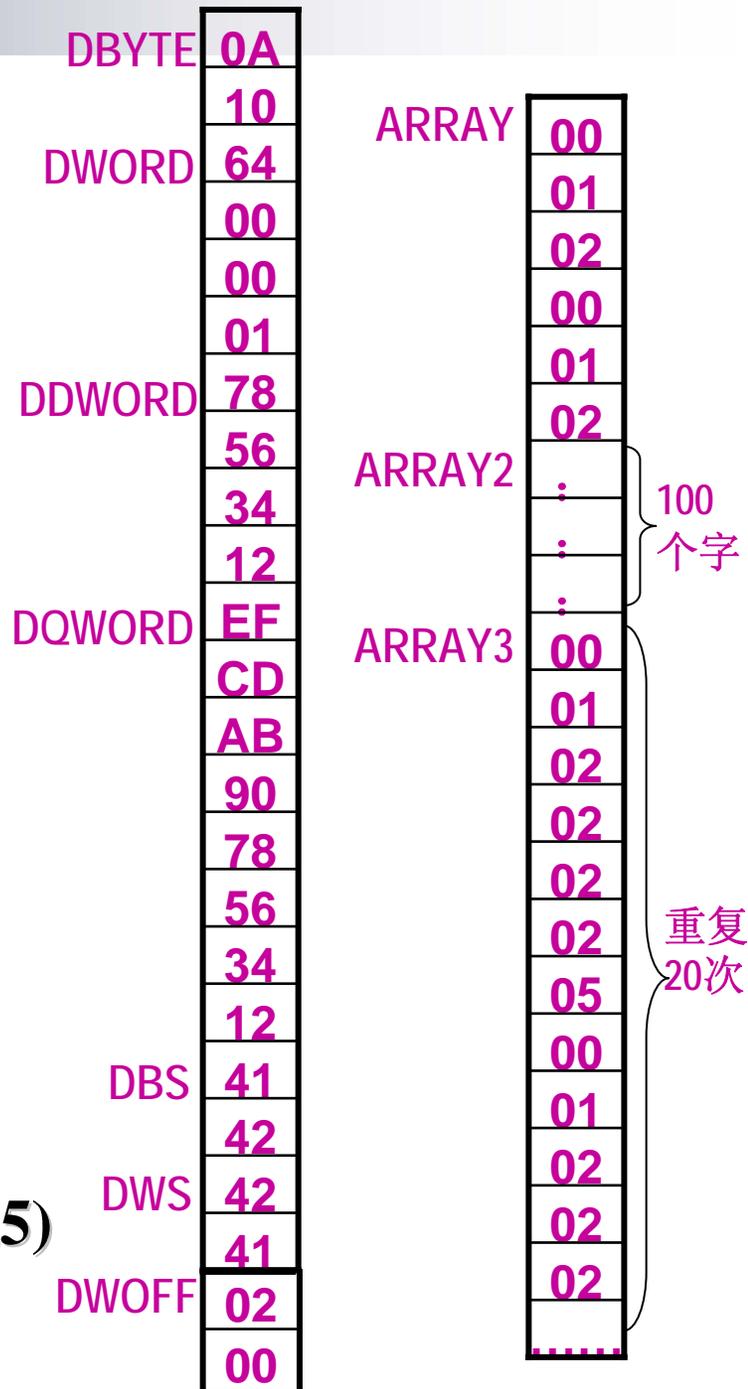
**DWOFF DW OFFSET DWORD**

**ARRAY DB 2 DUP (0, 1, 2)**

**ARRAY2 DW 100 DUP (?)**

**ARRAY3 DB 20 DUP(0,1,4 DUP(2),5)**

**DATA ENDS**



```

DATA SEGMENT
    X1    DB    12H, 0ABH
    X2    DB    '12AB'
    X3    DW    12ABH
    X4    DW    'AB'
    X5    DD    X1
    X6    DW    X2
    X7    DD    12345678H
    CNT   EQU   $-X6
    Y1    DB    CNT DUP (?), 1, 2 DUP (0FFH)
DATA ENDS
;设DS=1234H
;这里不能用DW
;尽量少用
;X1的段基址:偏移量
;X2的偏移量
;常量6, 不占存储单元
    
```

DATA ENDS

X1	12H 0ABH
X2	31H 32H 41H 42H
X3	0ABH 12H
X4	42H 41H

X5	00H 00H 34H 12H
X6	02H 00H
X7	78H 56H 34H 12H

Y1	?
	01H
	0FFH
	0FFH
	?
	01H
0FFH	
0FFH	
?	
01H	
0FFH	
0FFH	

共重复6次

## 二、表达式赋值语句

- 表达式赋值语句有以下两种，均不占内存。

### 1) 赋值伪指令

**格式：符号 EQU 表达式** ；

给右边的表达式定义一个符号名，一经定义在同一程序中不能再重新定义。

指令**PURGE** 符号可以解除对符号的赋值。

### 2) 等号伪指令

**格式：符号 = 表达式**

与**EQU**功能相同，区别在于左边符号允许重新定义。

(教材**P139~P140**，例**4-27、4-28**)

## 三、段定义和段分配语句

### 1) 段定义语句

格式：段名 **SEGMENT** 定位类型 组合 ‘分类名’

....

段名 **ENDS**

**A) 定位类型**—表示对段起始边界的要求，即对段首地址定位。此参数缺省，则定位于节边界。

① **PAGE--**定位于页边界( $n*256$ ) (**\*\*\*0000000B**)

② **PARA**—定位于节边界( $n*16$ ) (**\*\*\*\*0000B**)

③ **WORD--**定位于字边界 (**\*\*\*\*0B**)

④ **BYTE--**定位于存储器任何字节

**B) 组合类型：**告诉连接程序本段与（其它模块的）其它段的关系。

- (1) **NONE**—本段与其它段在逻辑上不发生关系。
- (2) **PUBLIC**—连接程序把几个模块的同名段相邻地连接成一个逻辑段，次序由连接命令指定，地址由低到高同时满足定位要求。
- (3) **COMMON**—该段在连接时与其它模块的同名段有相同的起始地址，在内存中以覆盖方式存放，连接长度为各分段的最大长度。**COMMON**段使得不同模块之间的数据可以共享。
- (4) **AT**—以表达式的值为该段基址(代码段不允许使用)。

(5) **STACK**—指定堆栈段，在堆栈段不能省略(\*)。

(6) **Memory**—作用类似与**COMMON**。

**C) 类别或分类名**。长度小于**40**字符并以‘ ’括起来。汇编时将同类别的段集中组成段组。

- 段定义语句允许嵌套，但是各个逻辑段不得交叉。

(教材**P140~141**)

## 2) 段分配语句

- 格式:

**ASSUME CS:段名, DS:段名 [, SS:段名] [, ES:段名]**

- 功能和使用方法:

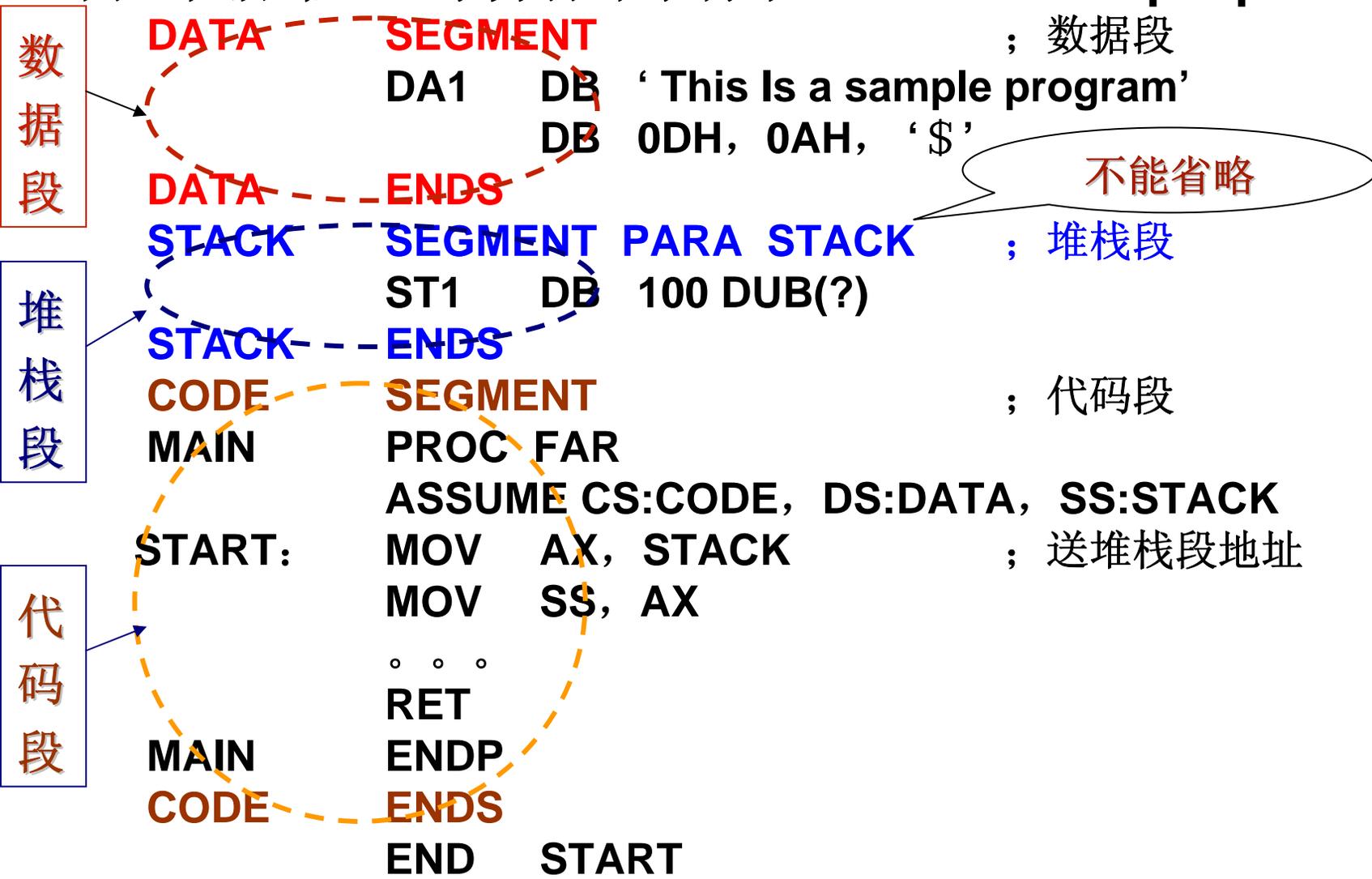
- 定义4个逻辑段，指明段与段寄存器的关系。
- 段名必须是**SEGMENT...ENDS**定义过的。
- **ASSUME...NOTHING**取消前面**ASSUME**指定的段寄存器。
- 四个段不一定全部定义，**代码段和数据段必须定义**。
- **ASSUME**只是指定某段分配给何寄存器，并不能将段地址装入段寄存器。仅**CS**在分配时自动装入。

### 3) 组定义语句**GROUP**

- 格式： 组名 **GROUP** 段名[, 段名.....]
- **GROUP**将程序中若干不同名的段集合成一个组，并赋予一个组名，使它们都装在一个**64KB**的物理段中。
- 这时组内不同类型的段运行时共用一个段寄存器，组内各段间的跳转都可以看作段内跳转。

# 汇编语言程序示例

例：在屏幕上显示并打印字符串“**This Is a sample pro...**”



## 四、过程定义语句

- 格式： 过程名 **PROC** 属性

....

**RET N**

过程名 **ENDP**

- 说明：

- 过程名三属性：段基地址、偏移地址、距离。
- **RET N**返回后从栈顶开始再弹出**N**个字节丢弃。
- 一个过程可能有多个**RET**，即过程有多个出口。
- 过程允许嵌套和递归调用（被调用的过程在返回前又调用过程本身）。
- 调用过程时注意保护，退出时注意恢复。

## 五、程序开始和结束语句

### 1) **NAME**伪指令

- 格式: **NAME** 程序名
- 功能和使用方法:
  - 为源程序汇编输出的目标模块赋名。
  - **NAME**为伪指令助记符, 置于程序开始处, 在输出源程序列表文件时, 将在每页开头打出程序名。若省略汇编则将源文件名作为目标模块的名字。

### 2) **TITLE**伪指令

- 格式: **TITLE** 文本名; 文本名赋予目标模块
- 功能和使用方法: 同**NAME**

### 3) 指定地址伪指令ORG

- 在汇编程序中，有一个软件计数器(LC)——地址计数器。在每个段定义开始处，它被复位为0。当汇编程序逐行扫描语句时，用LC保存当前指令的地址。
- 格式1：  
**ORG 表达式** ; 表达式值(0~65535)→ LC
- 格式2：  
**ORG \$+表达式** ; LC当前值\$+表达式值→ LC
- 功能：强行指定地址计数器LC的当前值，在段内改变它以后的代码或数据存放的偏移地址。
- **ORG**语句之后段内所有的**代码或数据**均以表达式（常量）的值为**起始偏移量**连续存放，除非遇到另一个**ORG**语句。

## 4) END语句

- 格式：**END** 标号名
- 功能和使用方法：
  - 指示源程序结束。
  - **END**放在程序最后一行。每个模块只有一个**END**，汇编程序碰到**END**停止编译，
  - 标号名是程序启动地址。

## 六、结构定义语句(\*)

- 对于具有类似数据库表结构的复杂数组，汇编程序提供了结构定义伪指令以及记录定义伪指令。
- 结构定义和使用三个步骤：**1)** 结构定义；**2)** 副本预置；**3)** 结构的使用。
- 与数据库的类比关系：
  - 结构——数据库中的某个表结构
  - 结构副本——表中的一条记录
  - 变量——某条记录中的一个字段

# 1) 结构定义

- 格式:

结构名    **STRUC**

(用**DB**、**DW**、**DD**等定义的结构中的数据变量)

结构名    **ENDS**

- 功能: 把不同类型的数据放在同一个数据结构中  
(类似于定义数据库表的各个字段)

- 结构中的变量类型

- 简单变量: 只有一个元素, 被引用时可以修改。

- 多重变量: 包括多个元素, 被引用时不能修改, 保持结构定义时初值不变。

- 字符串变量：变量为字符串。被引用时可以用同样长度的字符串对其进行修改（置换）。
- 多重结构：变量本身又是另外一个结构。

### 例4-36：定义一个数据表格**TAB**的结构

```
TAB    STRUC  
  
    T1    DB 'ABCD'           ; 字符串  
  
    T2    DW ?                ; 简单变量  
  
    T3    DW SEG L1           ; 简单变量  
  
    T4    DW 2 DUP (0)        ; 多重变量  
  
    T5    DW 1122H, 3344H     ; 多重变量  
  
TAB    ENDS
```

## 2) 结构副本预置

- 结构定义后，汇编时不产生目标代码，也不分配存储空间。只有在预置结构副本之后，汇编才给每个副本分配空间，结构中的变量才与存储单元发生关联。
- 格式中允许修改的变量（简单变量和字符串变量）在不同的结构副本中可以被修改成不同的值。
- 预置结构副本语句格式有两种：
- 格式1：  
副本名 结构名 <元素值，元素值， ...> ; 注释
- 格式2：  
副本名 结构名 **N DUP** (<元素值，元素值， ...>) ; 注释
- < >是结构副本预置专用运算符，表示将结构中可以修改的变量改为< >中的数值。< >中无数据表示不修改。

### 例4-37：结构副本元素值的表示：

**< >** ; 副本对结构定义的所有变量不修改。

**<20H>** ; 第一个变量改为**20H**，后续不变。

**<, 352AH>** ; 第二个变量改为**352AH**，其余不变。

**<'OK', , , 0DH>** ; 修改第一和第四个变量，其余不变。

说明：只有简单变量和字符串变量才能被修改。

### 例4-38：对例4-36定义的**TAB**预置4个结构副本

**ONE      TAB   < >** ; **ONE**—结构副本名

**TWO      TAB   < 'STOP' >** ; **TAB**—结构名

**THERE    TAB   < , 0FH, SEG L2>**

**FOUR     TAB   5 DUP ( < 'EFGH', 55H > )**

- 问题：多个结构副本中的不同变量如何表示？
  - 1) 结构副本名 · 变量名。例如 **TWO · T1** 表示在 **TWO** 副本中的 **T1** 变量。
  - 2) 在类似 **FOUR** 的结构副本，**FOUR · Tx[n]** 用于区别多个相同结构副本中的变量。其中：**Tx** 是变量名；**n** 为下标值，表示当前结构副本首地址与第一个相同的结构副本首地址（第一个字节）之间的字节距离。
- 预置后的结构副本中的变量的三属性：
  - ① 段属性：副本中各变量与预置结构副本语句同在一段；
  - ② 偏移量：副本起始字节偏移量 + 本变量相对偏移量；
  - ③ 类型：由结构定义时确定。

假设：**DS=1000H**，结构副本**ONE**的偏移量为**2000H**。

**4**个预置的结构副本在存储器中的地址分配如图。

**ONE TAB < >**

12000H	'B'	'A'	ONE-T1
	'D'	'C'	
	?	?	ONE-T2
	SEG L1		ONE-T3
	0	0	ONE-T4
	0	0	
	11	22	ONE-T5
	33	44	

**TWO TAB <'STOP' >**

12010H	'T'	'S'	TWO-T1
	'P'	'O'	
	?	?	TWO-T2
	SEG L1		TWO-T3
	0	0	TWO-T4
	0	0	
	11	22	TWO-T5
	33	44	

**THERE TAB <,0FH,SEG L2 >**

12020H	'B'	'A'	THERE-T1
	'D'	'C'	
	00	0F	THERE-T2
	SEG L2		THERE-T3
	0	0	THERE-T4
	0	0	
	11	22	THERE-T5
	33	44	

FOUR TAB 5 DUP (< 'EFGH', 55H >)

12030H	'F'	'E'	FOUR-T1[0]
	'H'	'G'	
	00	55	FOUR-T2[0]
	SEG L1		FOUR-T3[0]
	0	0	FOUR-T4[0]
	0	0	
	11	22	FOUR-T5[0]
	33	44	
12040H	'F'	'E'	FOUR-T1[10H]
	'H'	'G'	
	00	55	FOUR-T2[10H]
	....		...
12070H	'F'	'E'	FOUR-T1[40H]
	'H'	'G'	
	00	55	FOUR-T2[40H]
	SEG L1		FOUR-T3[40H]
	0	0	FOUR-T4[40H]
	0	0	
	11	22	FOUR-T5[40H]
	33	44	

## ● 结构的使用

- 在汇编语言源程序中，可以通过结构副本名、运算符“.”和下标[ ]对结构副本中的变量进行寻址和操作。

例4-39: **MOV TWO · T2, AX**

例4-40: **MOV BX, FOUR · T5[0]**

- 汇编程序在对源程序进行编译时计算出变量的物理地址，生成**CPU**可识别的目标代码。

## 七、外部伪指令及对准指令

### 1) 外部伪指令

外部伪指令用于实现程序和数据在多个模块之间的共享。

① 格式：**PUBLIC** 名字 [, 名字, ...] ;

● 说明：名字可以是标号、变量名、常数、**过程名**或由EQU（或=）伪指令定义的符号名。

● 功能：由**PUBLIC**说明的名字是全局的，可以在其它模块中使用（共享）。

② 格式: **EXTRN** 名字:类型 [, 名字:类型, ...]

● 说明:

名字: 与**PUBLIC**的要求相同;

类型: 名字为变量则类型为**BYTE**、**WORD**等; 名字为标号、过程, 则类型为**NEAR**、**FAR**, 名字为常数, 则类型为**ABS**。

● 功能: 表明本模块中使用的名字在别的模块中定义过, 调用别的模块中使用**PUBLIC**定义过的名字。**EXTRN**和**PUBLIC**配对使用, 并且类型一致。(P151例4-42)

## 2) 对准伪指令

- 格式: **EVEN** ;

- 功能: 将下一条语句的地址调整在偶地址上。

例4-43

```
DATA SEGMENT
    X1    DB 0DH
        EVEN
    X2    DW 100 DUP(?)
DATA ENDS
```

加EVEN后X2从偶数开始存放，可提高存储器访问速度。

- **\$**——汇编程序表示**当前地址计数器的值**。

### 3) LABEL伪指令

- 格式：名称 LABEL 类型属性
- 功能：给下一语句中已定义的变量或标号另取一名字，并可重定义类型和属性。使同一变量或标号在不同地方被调用时可采用不同的名字，具有不同的属性。
- 说明：
  - 名称：下行语句中变量或标号的别名
  - 类型：与变量连用属性可修改BYTE，WORD等；与标号连用属性可修改FAR，NEAR。

- **LABEL**与**变量**连用示例：给下一个变量起一个别名，类型属性修改为**BYTE**、**WORD**等。

### 例4-45:

```

DATB    LABEL BYTE           ; DATB为DATW的
                                   ; 别名，类型为字节
DATW    DW 3031H, 3233H      ; DATW类型为字
        MOV AL, DATB[0]      ; 31H→AL
        MOV BX, DATW[1]      ; 3330H→BX
        MOV BX, DATW[2]      ; 3233H→BX

```

### 例4-46: 堆栈段中经常使用**LABEL**

```

STACK   SEGMENT STACK 'STACK'
        DW 100 DUP ( ? )
        TOP LABEL WORD      ; 定义100个字的堆栈，栈底
STACK   ENDS                ; 名为TOP，类型为字。

```

- **LABEL**与**标号**连用示例：给下一语句的标号起一个别名，距离类型属性修改为**NEAR**或**FAR**。

### 例4-47:

```
DISF LABEL FAR  
DISN: MOV AX, [SI]
```

标号**DISF**和**DISN**指向同一条指令，**DISF**是**DISN**的别名，但是距离属性改为**FAR**，可以使用段间远调用。

## 八、80286以后新增的伪指令

80286以后增加了一些伪指令，用于：

- 80X86指令系统选择
- 编程模式选择
- 汇编语言存储模型选择

(参见教材P153~156)

# 习题4—1

**P206:**

**1**

**2**

**3**

**5**

# 内容

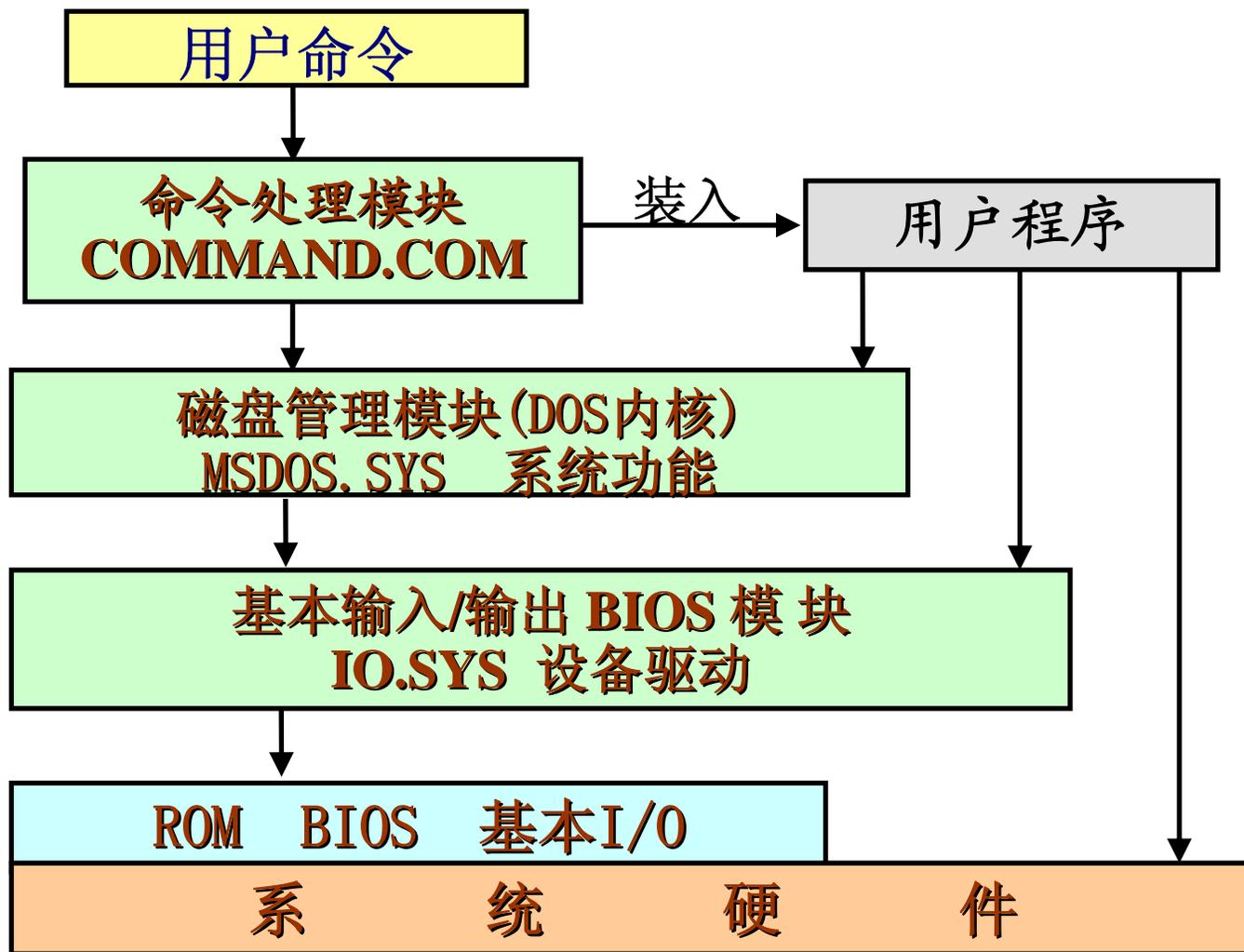
- 4-0 概述
- 4-1 汇编语言程序格式
- 4-2 MSAM中的表达式
- 4-3 伪指令语句
- 4-4 DOS系统功能调用和BIOS中断调用
- 4-5 程序设计方法
- 4-6 宏汇编和条件汇编

# DOS: Disk Operating System

## ● DOS功能

- 文件管理：在磁盘上建立或者删除文件、文件读写和文件检索。
- 设备管理：对显示器、键盘、打印机、磁盘及异步通信等I/O设备管理。
- **ROM BIOS**：基本I/O系统（**FE000H—FFFFFFH**），一部分固化于系统内存ROM中的部分管理软件。
- 用户可调用的子程序：**DOS 2.0—87个**，**DOS3.0—98个**，**DOS6.2—100多个**。
- 调用方式：软件中断调用——**INT n**

# DOS系统层次结构



## 层次特点（自底向上）：

- **ROM BIOS**中的中断子程序

使用**IN/OUT**指令直接控制外设，实现与外设之间的输入/输出操作，以软件形式向其上层提供服务。

- **IO.SYS**中的中断子程序

（称基本输入/输出**BIOS**模块）调用**ROM BIOS**的基本**I/O**功能，向**DOS.SYS**提供设备驱动服务。

- **DOS.SYS**中的中断子程序

（称**DOS**内核模块）调用**IO.SYS**，实现对外设的控制。与标准外设有两层隔离：**IO.SYS**和**ROM BIOS**。

- 其中，**21H类型**的中断子程序提供了丰富的系统服务，称**21类型**的中断调用为**DOS系统（功能）调用**。

- 在汇编语言程序设计中，用户可通过使用**BIOS**和及基本**DOS**系统提供的这些功能模块子程序（中断子程序调用），来编制直接管理和控制计算机硬件设备的底层软件，主要是完成**I/O**操作。

## ● 用户编程原则

- ① 尽可能使用**DOS**的系统功能调用，提高程序可移植性。
- ② 在**DOS**功能不能实现情况下，考虑用**BIOS**功能调用。
- ③ 在**DOS**和**BIOS**的中断子程序都不能解决问题时，才使用**IN/OUT**指令直接控制硬件。

# 1) 调用BIOS/DOS功能子程序的基本方法

BIOS/DOS的每个功能子程序都对应着一个子程序文件。使用时，直接用一条软中断指令INT n（n称为中断类型号）。BIOS中的n=5~1FH .....；DOS软中断中的 n =20H, 21H, 23H~2AH.....

## ①常用DOS功能调用列表

类型号	中断功能	类型号	中断功能
20H	程序结束	21H	请求DOS功能调用
22H	结束地址	23H	中止(Ctrl-Break)处理
24H	关键性错误处理	25H	磁盘顺序读
26H	磁盘顺序写	27H	程序结束且驻留内存
28H	DOS内部使用	29~2EH	DOS内部保留
2FH	DOS内部使用	30~3FH	DOS内部保留

## ②常用BIOS功能调用列表

类型号	中断功能	类型号	中断功能
00 H	被零除	11 H	设备检测
01 H	单步	12 H	存储容量
02 H	不可屏蔽	13 H	磁盘I/O
03 H	断点	14 H	通信I/O
04 H	溢出	15 H	盒式磁带I/O
05 H	打印屏幕	16 H	键盘I/O
06 H	保留	17 H	打印机I/O
07 H	保留	18 H	ROM BASIC
08 H	日时钟	19 H	引导
09 H	键盘	1A H	日时钟
0A H	保留	1B H	Ctrl-Break
0B H	串口2	1C H	定时器报时
0C H	串口1	1D H	显示器参数
0D H	硬盘	1E H	软盘参数
0E H	软盘	1F H	图形字符扩展
0F H	打印机	40 H	保留给软盘
10 H	显示器	41 H	硬盘参数

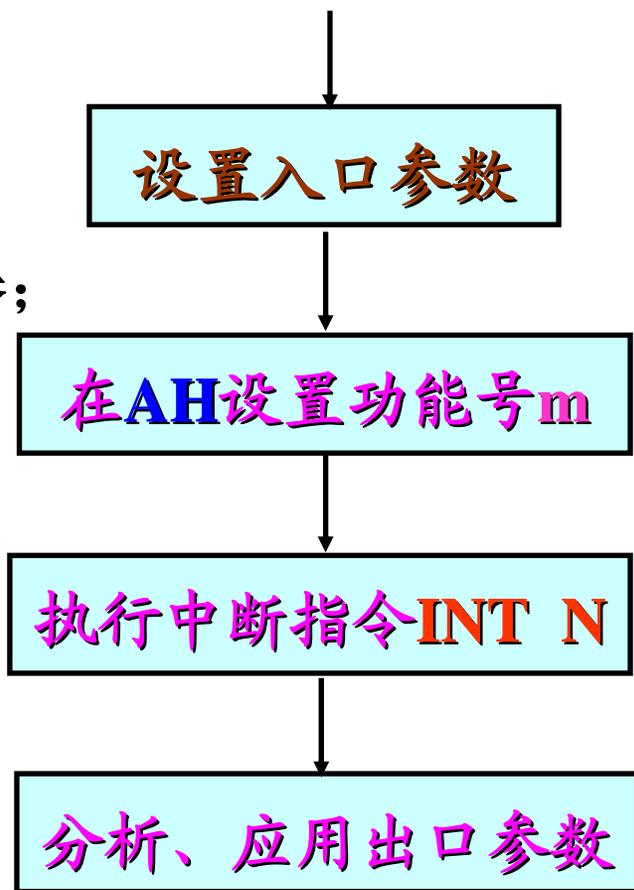
## 2) 使用步骤:

- ① 子程序入口参数送规定寄存器;
- ② 子程序编号(功能调用号)送AH寄存器;
- ③ 发软中断命令: **INT n**

## 3) 使用时注意事项

- ① 有的软中断号 **n** 对应一个子程序,调用时无需以上②;
- ② 有的软中断号 **n** 对应若干个子程序,必须严格按以上顺序执行。

- 常用的有: **INT 10H; INT 16H; INT 21H**



例1：将一个ASCII字符显示于屏幕当前光标的位置。

使用**BIOS**的中断类型号**10H**、功能调用号**0EH**：

**MOV AL, '?'** ; 要显示的字符送入**AL**

**MOV AH, 0EH** ; 功能号送入**AH**

**INT 10H** ; 调用**10H**软中断

例2：有的子程序不需要入口参数，这时①可略去。

**MOV AH, 4CH** ; 没有入口参数，只有调用号

**INT 21H** ; 调用**21H**软中断对应的**4CH**  
; 号子程序（带返回码返回  
; **DOS**, **AL**中是返回码）

- 在所有**DOS**功能调用中，将**INT 21H**软中断命令来实现的所有子程序调用称为**DOS系统功能调用**，它体现了**DOS**的**核心功能**。
- **INT 21H**对应**100**多个子程序（参见教材附录**F**）常用如下：
  - ① 单字符输入（**01H**、**07H**、**08H**功能）
  - ② 显示单字符（**02H**、**05H**功能）
  - ③ 单字符输入或显示（**06H**功能）
  - ④ 字符串输入（**0AH**功能）
  - ⑤ 显示字符串（**09H**功能）
  - ⑥ 检测键盘状态（**0BH**功能）
  - ⑦ 保存中断向量（**35H**功能）
  - ⑧ 设置中断向量（**25H**功能）

## DOS系统功能调用与BIOS中断调用的区别

- ① 调用BIOS中断程序比调用DOS的复杂一些，但运行速度快，功能更强，操作更加“细腻”；
- ② DOS系统功能调用只适用于DOS环境，而BIOS功能调用不受任何操作系统的约束；
- ③ DOS系统功能调用在更高层次上提供了与BIOS类同的功能。
- ④ 但是某些功能只有BIOS才具有。

## 二、DOS系统功能（INT 21H）调用

要求掌握的DOS系统功能调用：

### 1) 键盘输入

- 单字符输入；
- 字符串输入：

### 2) 显示器输出

- 单字符输出：
- 字符串输出：

### 3) DOS打印功能调用

### 4) 日期与时间的设置和读取

### 5) DOS返回：功能号—4CH

# 1. DOS键盘功能调用

通过**DOS**功能调用，读从键盘输入的键码到**AL**。有多个功能调用与键盘输入有关（**P158表4-7**）

## 1) 单字符输入

调用号**01H**、**06H**、**07H**、**08H**都能够实现，区别如下：

- **01H**：从键盘输入一字符并回显，检查**ctrl-Break**
- **08H**：从键盘输入一字符**不**回显，检查**ctrl-Break**
- **07H**：从键盘输入一字符**不**回显，**不**检查**ctrl-Break**
- **06H**：键盘输入/屏幕输出。
  - ✦ 当参数**DL=0FFH**时，从键盘输入，若**ZF=0**，**AL**为键码；若**ZF=1**，无键按下，**AL**不是键码。
  - ✦ 当参数**DL≠0FFH**时，表示屏幕输出。

**例4-51：交互式程序根据用户输入数字跳转**

```
KEY:   MOV  AH, 1           ; 等待从键盘输入
         INT  21H           ; 读入键码→AL
         CMP  AL, '1'       ; 键码='1'?
         JE   ONE
         CMP  AL, '2'       ; 键码='2'?
         JE   TWO
         CMP  AL, '3'       ; 键码='3'?
         JE   THERE
         ...
ONE    ...
         ...
TWO   ...
         ...
```

例：用不带回显功能输入密码，以回车结束。

```
input:  MOV  AH, 07H    ; 7号功能调用
        INT  21H      ; 键码输入→AL
        MOV  [DI], AL  ; 存入缓冲区
        CMP  AL, 0DH   ; 是回车符?
        JE   check    ; 是，转检查密码
        INT  21H      ; 不是继续输入
        MOV  [DI+1], AL ; 存入下一个缓冲区
        ... ..
check:  ...
        ...
```

## 2) 字符串输入 (0AH功能)

入口参数 **DS:DX**=缓冲区的首地址

**[DS:DX]**=设置输入字符串的最大长度

功能号 **AH = 0AH**

类型号 **21H**

出口参数 **[DS:DX+1]**=实际键入字符数(不含回车符),  
从**[DS:DX+2]**开始顺序存放键入的字符串, 回车符**0DH**  
为串尾最后一字符。

- **功能:** 等待从键盘输入字符串, 并存入设定的缓冲区内, 同时回显字符串, 光标随着移动, 回车符使光标回到行首。
- **注意事项:** 应按要求先定义缓冲区, 再调用。

例：应用0AH功能输入字符串，最大串长为11。

<b>buff</b>	<b>SEGMENT</b>	;定义缓冲区	对应的另外形式语句
<b>max</b>	<b>DB 11</b>	;定义限制最多输入个数	◆ <b>max DB 11</b>
<b>Lenth</b>	<b>DB ?</b>	;用于存放实际输入个数	◆ <b>DB ?</b>
<b>stri</b>	<b>DB 11 DUP(?)</b>	;用于存放输入的字符串	◆ <b>DB 11 DUP(?)</b>
<b>buff</b>	<b>ENDS</b>		
<b>code</b>	<b>SEGMENT</b>		
	<b>ASSUME CS:code, DS:buff</b>		
<b>start:</b>	<b>MOV AX, buff</b>	;置缓冲区地址于DS:DX	
	<b>MOV DS, AX</b>		
	<b>LEA DX, max</b>		
	<b>MOV AH, 0AH</b>		
	<b>INT 21H</b>	;调0AH输入功能，max后面	
	<b>MOV CH, 0</b>	;的lenth是实际串长	◆ <b>MOV CH, 0</b>
	<b>MOV CL, lenth</b>	;取字符串长度放CX中	◆ <b>MOV CL, max+1</b>
	<b>LEA BX, stri</b>	;取字符串首址于BX中	◆ <b>LEA BX, max+2</b>
	<b>MOV AL, [BX]</b>	;应用输入字符	
	<b>.....</b>		
<b>code</b>	<b>ENDS</b>		

## ● 0AH功能执行过程

- ① 若[DS:DX]字节单元的值为0，则不等待从键盘输入，结束调用。
- ② 若[DS:DX]字节单元的内容大于0，则等待从键盘输入，并把输入键码顺序存放在DS:DX+2开始的单元，按回车键表示结束输入。当按下键的个数超过[DS:DX]中的值，发出警告声“嘟嘟”，不再接收输入的数据，直到输入回车键。
- ③ 输入结束后，实际输入的字符个数(不含回车键)→[DS:DX+1]，结束调用。

## ● 0AH功能注意事项

- ① 回车符0DH作为最后输入的字符存放串尾，但不计数。实际最多能输入的字符数=设定的最大串长-1（回车符占1个）。
- ② 执行完0AH功能后，DS和DX的值不变，仍指向缓冲区的首地址。
- ③ 整个缓冲区的大小应为：设定的最大串长度+2。

### 3) 检测键盘状态 (0BH功能)

入口参数: 无; 功能号: **AH = 0BH**; 类型号: **21H**

出口参数: **AL = 0FFH/0**, 表示有/无键按下。

- 功能: 检测键盘状态 (通过检测键盘缓冲区实现)
- 例: 利用**0BH**功能实现按键退出循环。

```
LOOP:    ...
          ...
          ...
          MOV  AH, 0BH      ; 检测是否有键按下
          INT  21H
          CMP  AL, 0FFH    ; 教材上为INC AL/JNZ LOOP
          JNZ  LOOP        ; 无键按下继续循环
          ...
          MOV  AH, 4CH      ; 4CH号功能, 返回DOS
          INT  21H
```

## 2. DOS显示功能调用

### 1) 显示单字符（02H功能）

入口参数—**DL**=要显示字符的**ASCII** 码

功能号——**AH=02H**    类型号—**21H**

出口参数—无

- 实现功能：在显示器上显示指定字符，光标随动；

例：显示字符A

```
MOV  DL, 'A'  
MOV  AH, 02H  
INT  21H
```

例：使光标回到下一行的行首。

```
MOV  DL, 0DH      ;显示回车符  
MOV  AH, 02H  
INT  21H  
MOV  DL, 0AH      ;显示换行符  
MOV  AH, 02H  
INT  21H
```

## 2) 显示字符串 (09H功能)

- 入口参数

- ① 定义要显示的字符串，字符串尾应为'\$'，作为结束显示的标志。

- ② **DS:DX** = 字符串的首地址。

- 功能号—**AH = 09H**；      类型号—**21H**

- 出口参数—无

- 功能：显示字符串，遇 '\$' 停止显示，光标随动。

# 例：显示字符串'Welcome To USTC'后返回DOS

```
Data SEGMENT ; 定义显示的子字符串
Stri DB 'Welcome To USTC', '$'
Data ENDS
Code SEGMENT
    ASSUME CS:Code, DS:Data
start: MOV AX, data ; 置缓冲区地址于DS:DX
      MOV DS, AX
      LEA DX, stri
      MOV AH, 09H ; 调显示功能
      INT 21H
      MOV AH, 4CH ; 返回DOS
      INT 21H
Code ENDS
END start
```

例：利用DOS系统功能调用实现人机对话。根据屏幕上显示的提示信息，从键盘输入字符串并存入内存缓冲区。

```

DATA    SEGMENT
BUF     DB  100           ; 定义输入缓冲区长度
          DB  ?             ; 保留为填入实际输入的字符个数
          DB 100 DUP (?)   ; 准备接收键盘输入信息
MESG    DB 'WHAT IS YOUR NAME ? $' ; 要显示的提示信息
DATA    ENDS
CODE    SEGMENT

          ... ..
START: MOV  AX, DATA
          MOV  DS, AX
          ...
          MOV  DX, OFFSET MESG
          MOV  AH, 9           ; 屏幕显示提示信息
          INT  21H
          MOV  DX, OFFSET BUF
          MOV  AH, 10          ; 即0AH功能, 接收键盘输入
          INT  21H
          ...

```

### 3. DOS打印功能调用

#### 1) DOS打印功能（05H功能）

入口参数—**DL**=要打印内容  
（包括打印控制码）

功能号—**AH=05H**

类型号—**21H**

出口参数—无

教材P162例4-56

#### 2) 特殊打印命令

控制打印格式

教材P162例4-57

字符码	功能
<b>08H</b>	空格
<b>09H</b>	水平制表 <b>TAB</b>
<b>0AH</b>	换行
<b>0BH</b>	垂直制表 <b>TAB</b>
<b>0CH</b>	换页
<b>0DH</b>	回车

字符码	功能
<b>0FH</b>	设置紧缩格式
<b>0EH</b>	设置扩展格式
<b>12H</b>	取消紧缩格式
<b>14H</b>	取消扩展格式
<b>1BH 30H</b>	设置每英寸 <b>8</b> 行
<b>1BH 32H</b>	设置每英寸 <b>6</b> 行
<b>1BH 45H</b>	设置加重打印
<b>1BH 46H</b>	取消加重打印

## 4. 日期和时间设置

### 1) 设置日期 (2BH功能)

入口参数: **CX**←年号, **DH**←月份, **DL**←日期(1~31)

功能号——**AH=2BH**

类型号——**21H**

功能: 设置系统日期

出口参数: **AL=0**

;设置成功, 日期有效

**AL=0FFH**

;无效日期, 设置不成功

### 2) 取得日期 (2AH功能)

入口参数: 无

功能号——**AH=2AH**

类型号——**21H**

功能: 读取系统日期

出口参数: 存放格式同设置日期格式, 均为二进制数。

### 3) 设置时间 (2DH功能)

入口参数: **CH**←时(0~23), **CL**←分(0~59),

**DH**←秒(0~59), **DL**←百分之一秒(0~99)

功能号——**AH=2DH**

类型号——**21H**

功能: 设置系统时间

出口参数: **AL=0**

;设置成功, 时间有效

**AL=0FFH**

;无效时间, 设置不成功

### 4) 取得时间 (2CH功能)

入口参数: 无

功能号——**AH=2CH**

类型号——**21H**

功能: 读取系统时间

出口参数: 存放格式同设置时间格式, 均为二进制数。

## 三、BIOS中断调用

### ● BIOS

- 驻留在**ROM**中，通过系统加电自检、引导装入**IO**设备的管理程序以及接口控制模块。
- 提供多种中断调用服务，方便用户程序开发。

### ● 与**DOS**系统功能调用比较

- 相同：用户可以直接通过指令预置参数，然后中断调用**BIOS**中的程序：
- 不同：**1) BIOS**中断调用的功能更加“细腻”；  
**2) BIOS**中断调用与操作系统无关。

## ❖ BIOS功能调用类型

类型号	中断功能	类型号	中断功能
00 H	被零除	11 H	设备检测
01 H	单步	12 H	存储容量
02 H	不可屏蔽	13 H	磁盘I/O
03 H	断点	14 H	通信I/O
04 H	溢出	15 H	盒式磁带I/O
05 H	打印屏幕	16 H	键盘I/O
06 H	保留	17 H	打印机I/O
07 H	保留	18 H	ROM BASIC
08 H	日时钟	19 H	引导
09 H	键盘	1A H	日时钟
0A H	保留	1B H	Ctrl-Break
0B H	串口2	1C H	定时器报时
0C H	串口1	1D H	显示器参数
0D H	硬盘	1E H	软盘参数
0E H	软盘	1F H	图形字符扩展
0F H	打印机	40 H	保留给软盘
10 H	显示器	41 H	硬盘参数

# (一) 键盘字段调用 (INT 16H)

(教材P166表4-12)

AH	功能	返回参数
0	从键盘读一个字符	AL=字符, AH=扫描码
1	读键盘缓冲区字符	ZF=0/1时, AL=字符缓冲区 / 缓冲区为空
2	取特殊功能键状态 (KBFLAG)	AL=KBFLAG, KBFLAG(B7~B0)为1依次代表Ins、Caps、Num、Scroll、Alt、Ctrl、Left和Right键被按下 (激活)。

(对比DOS调用: INT 21H的1、8、6、7号调用)

说明: 扫描码是CPU从键盘接口读到的与键所在的行列位置有关的编码。而ASCII码则是接口程序根据扫描码通过查表方式进行翻译转换的结果。

## (二) 显示和屏幕设置中断调用 (INT 10H)

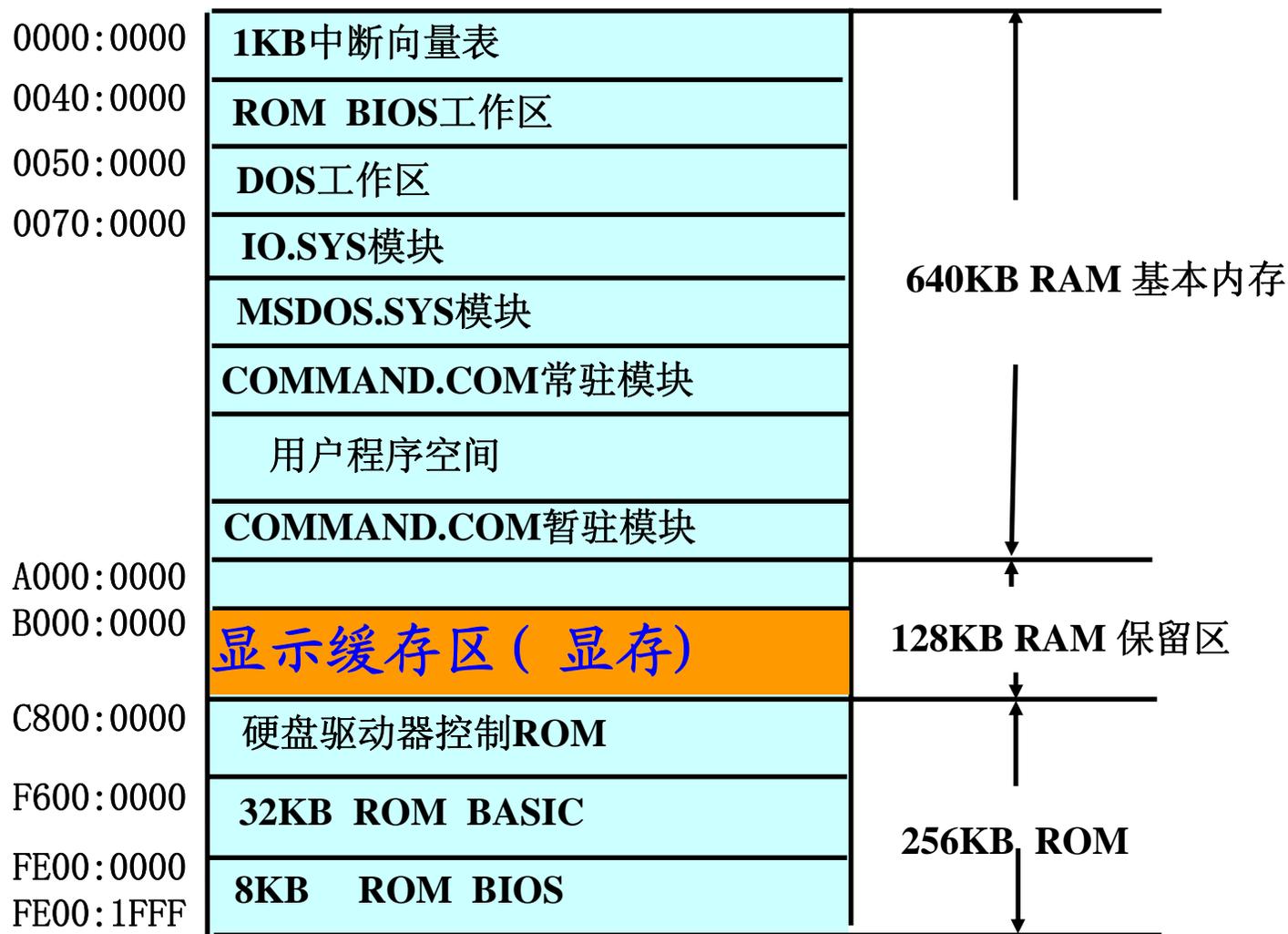
(教材P166表4-13)

AH	功能	入口参数	出口参数
0	初始化CRT	AL=CRT工作方式	
1	置光标类型	CX=光标开始、结束行	
...	...	...	...
8	读光标处字符属性	BH=页号	AH=属性, AL=字符
...	...	...	...
0FH	读当前状态		AH=字符列数

(对比DOS调用: INT 21H的2、6、9号调用)

# ● DOS环境下的显示器属性

屏幕上各象素的显示信息，存放在显示缓冲区(显存)中



## 显示器有两种显示方式:

### ● 文本方式:

- 将屏幕划分为若干行和列，在每个网格位置上显示象素，
- 一个字符是一个象素。

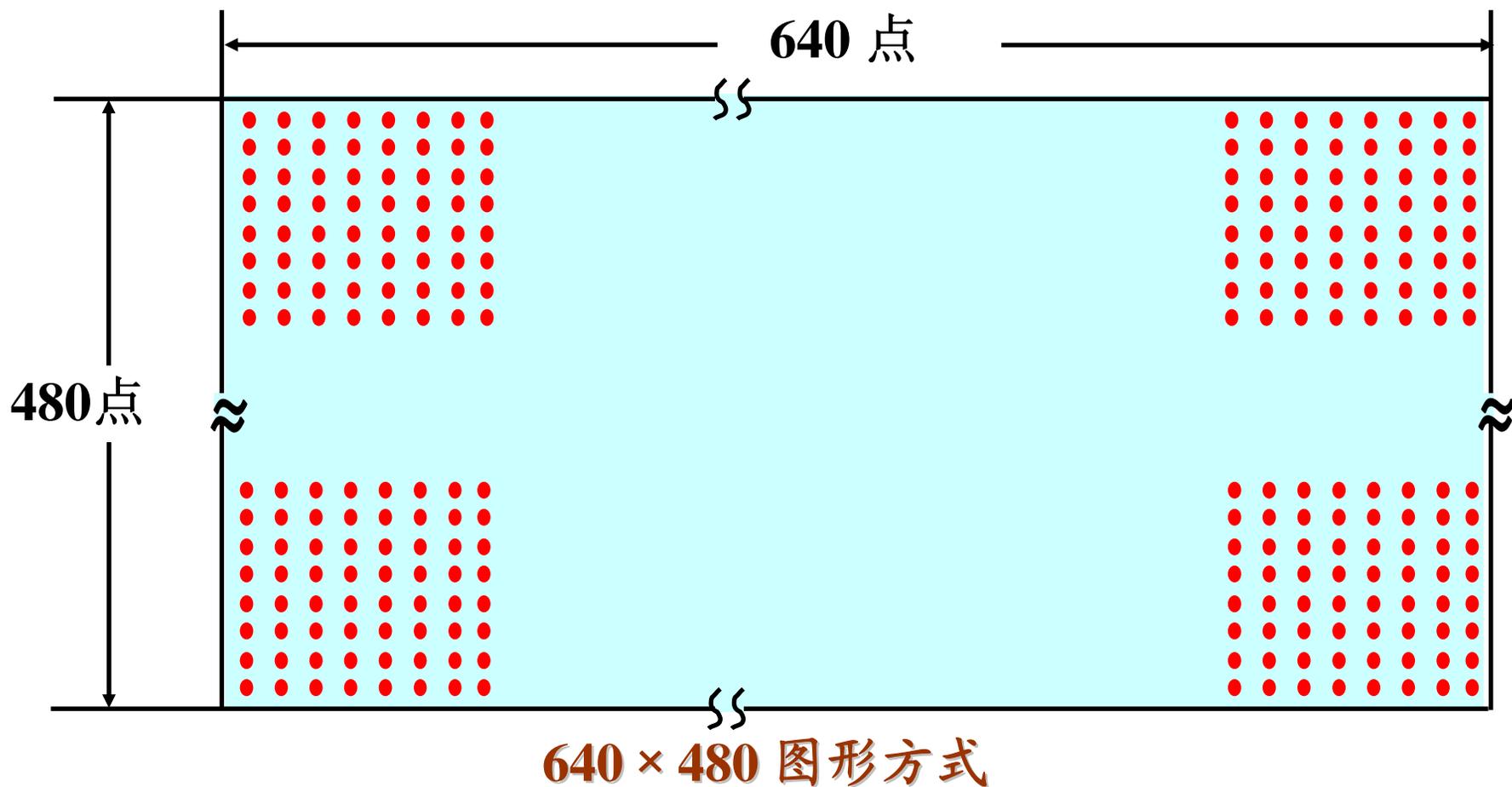
### ● 图形方式:

- 将屏幕划分为  $m \times n$  的点阵，在每个点的位置显示象素，
- 一个点是一个象素。



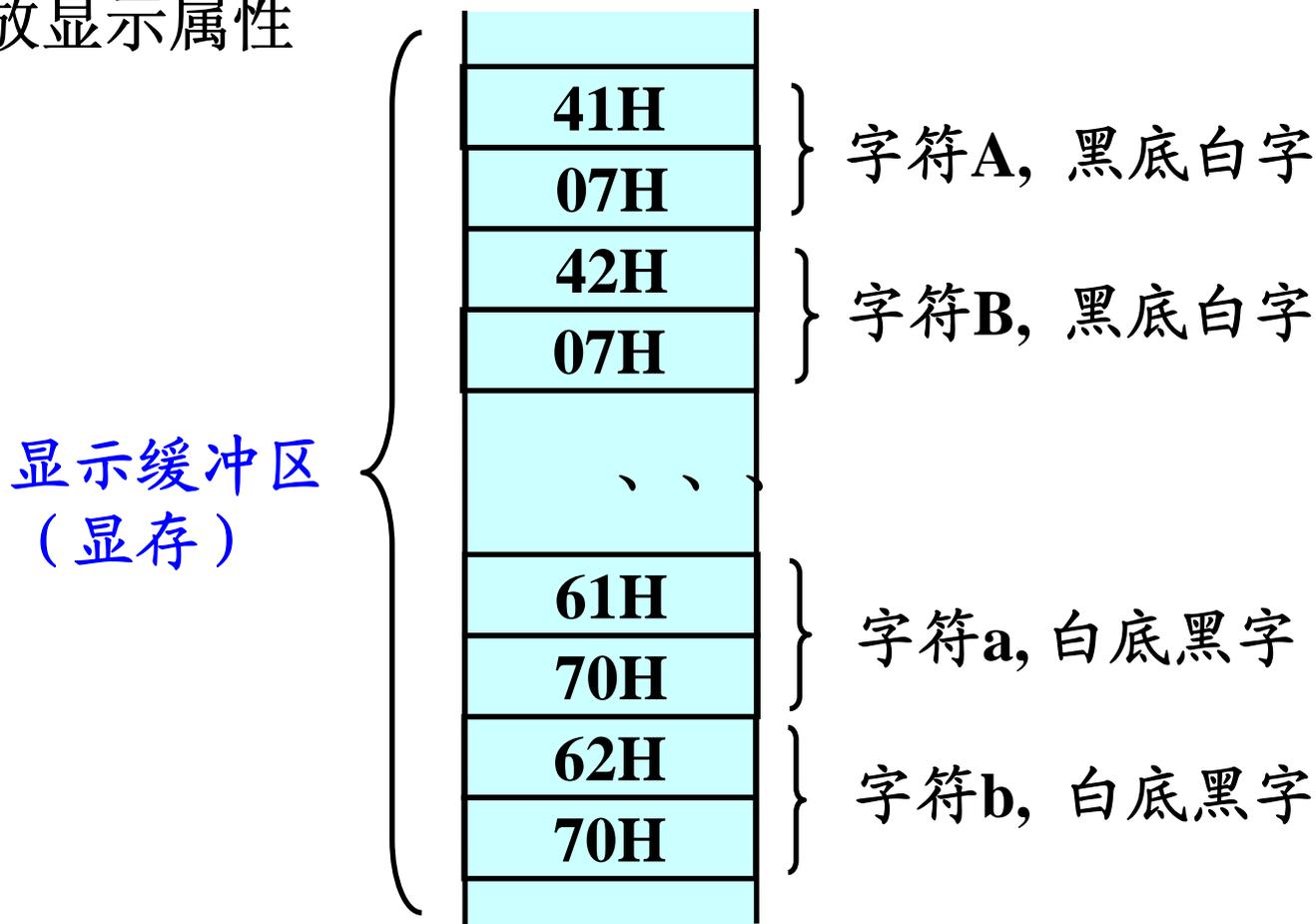
## ● 图形方式:

- 将屏幕划分为  $m \times n$  的点阵，在每个点的位置显示象素，一个点是一个象素。

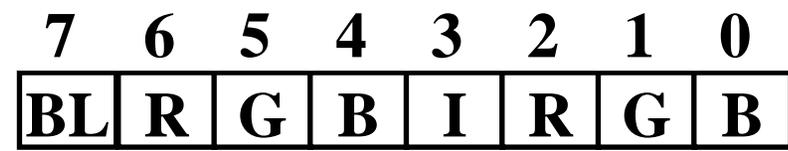


● 在文本方式下，对应屏幕上的每个字符，在显示缓冲区中占用两个单元：

- 一个存放**ASCII**码
- 一个存放显示属性



# 属性字节的含义:



闪烁

背景

前景

0 不闪烁  
1 闪烁

8种

- 000 黑
- 001 蓝
- 010 绿
- 100 红
- 111 白
- ...

16种

- 0000 黑
- 0010 绿
- 0100 红
- 0111 灰白
- 1000 灰
- 1010 浅绿
- 1100 浅红
- 1111 白
- ...

例 **10000111B** 或 **87H**  
表示黑底白字, 闪烁

**01110000B** 或 **70H**  
表示白底黑字, 不闪烁

- 根据显存大小，可存储若干页的字符象素。
- 例：**16KB** 显存能存储
  - **80 × 25**方式，**4页( 0 ~ 3 )**，  
 **$80 \times 25 \times 2 \times 4 = 16,000\text{Byte}$**
  - **40 × 25**方式，**8页( 0 ~ 7 )**，  
 **$40 \times 25 \times 2 \times 8 = 16,000\text{Byte}$**

# 1. 设置显示方式

- 入口参数 **AL = 显示方式值**

00	40×25	黑白文本方式
01	40×25	彩色文本方式
02	80×25	黑白文本方式
03	80×25	彩色文本方式
04	320×320	彩色图形方式
- 功能号 **AH = 00H**
- 类型号 **10H**
- 出口参数 无
- 实现功能 将显示方式设置为指定形式

## 2. 清屏、清窗口功能

● 入口参数 **AL = 0**

**CH** = 窗口左上角行号

**CL** = 窗口左上角列号

**DH** = 窗口右下角行号

**DL** = 窗口右下角列号

**BH** = 窗口属性

● 功能号 **AH = 06H 或 AH = 07H**

● 类型号 **10H**

● 出口参数 无

● 实现功能 按给定属性清除指定的窗口内容



- 例1：将显示方式设置为 **80×25** 彩色文本方式

```
MOV AL, 03H
```

```
MOV AH, 00
```

```
INT 10H
```

- 例2：清屏并将其属性置为反白(白底黑字)显示。

```
MOV AL, 0 ;清屏功能
```

```
MOV BH, 70H ;白底黑字
```

```
MOV CH, 0 ;左上角 行号
```

```
MOV CL, 0 ;左上角 列号
```

```
MOV DH, 24 ;右下角 行号
```

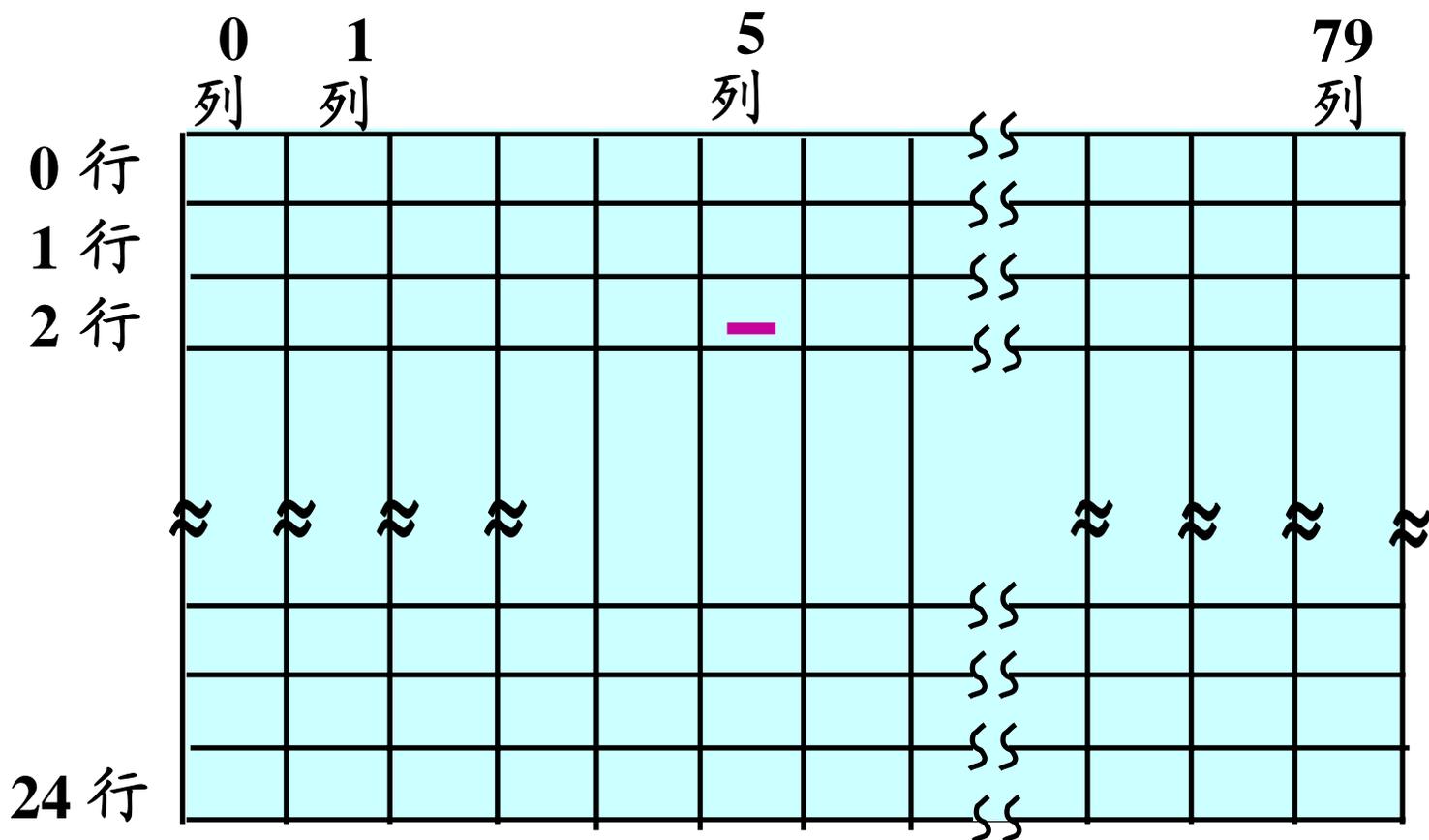
```
MOV DL, 79 ;右下角 列号
```

```
MOV AH, 6 ;功能号
```

```
INT 10H ;中断调用
```

### 3. 置光标位置

- 光标控制开始显示的位置，
- 计算机有专门的硬件控制光标的显示大小、位置。
- 光标只在文本方式中出现，在图形方式下光标消失。



# 置光标位置

例：将光标设置在**2行5列**位置

入口参数

DH = 行号

DL = 列号

BH = 页号

功能号

AH = 02H

类型号

10H

出口参数

无

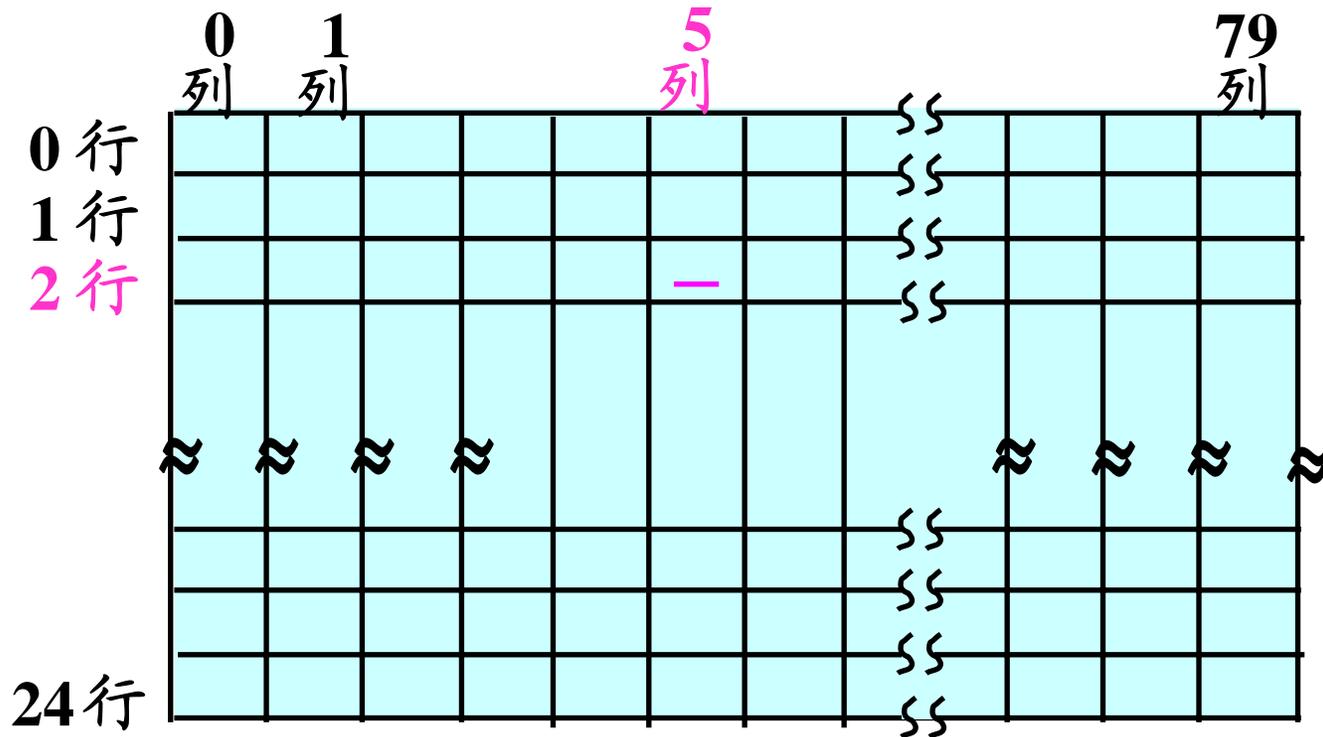
```
MOV DH, 2
```

```
MOV DL, 5
```

```
MOV BH, 0
```

```
MOV AH, 2
```

```
INT 10H
```



### (三) 打印中断调用 (INT 17H)

(教材P167表4-14)

AH	功能	入口参数	出口参数
0	指定字符在指定打印机上打印	<b>DX</b> =打印机号, <b>AL</b> =待打印字符	<b>AH</b> =打印机状态
1	初始化打印机	<b>DX</b> =打印机号 <i>AL = 初始化命令</i>	<b>AH</b> =打印机状态
2	读打印机状态	<b>DX</b> =打印机号	<b>AH</b> =打印机状态

(对比DOS调用: INT 21H的5号调用)

说明: 1) **DX**最多指定三台打印机(三个并行口), 机号**0~2**。

2) **AH**返回的打印机状态字节 (**B7~B0**) 的含义见教材P168图4-8。

## (四) 时间设置与读取 (INT 1AH)

(教材P168表4-15)

AH	功能	入口参数	出口参数
0	读取时间		CH:CL=时:分
1	设置时间	CH:CL=时:分 DH:DL=秒:1/100秒	
2	读时钟 (AT机以后)	(BCD码) →	CH:CL=时:分 DH:DL=秒:1/100秒
6	置报警时间 (AT机以后)	CH:CL=时:分 DH:DL=秒:1/100秒	← (BCD码)
7	消除报警		

(对比DOS调用: INT 21H的2CH和2DH号调用)

## (五) 串行通信功能调用 (INT 14H)

(教材P169表4-16)

AH	功能	入口参数	出口参数
0	初始化串口	AL=初始化参数 DX=通信口号 0—COM0, 1—COM1	AH=通信口状态 AL=调制解调器状态
1	向串口写字符	AL=所写字符 DX=通信口号 0—COM0, 1—COM1	AH7=0, 成功 AH7=1, 失败 AH6~0, 通信口状态
2	从串口读字符	DX=通信口号 0—COM0, 1—COM1	AH7=0, 成功 AH7=1, 失败 AH6~0, 通信口状态
6	取串口状态	DX=通信口号 0—COM0, 1—COM1	AH=通信口状态 AL=调制解调器状态

## ● 说明:

- 1) **AH=0**时对串口初始化, 初始化参数**AL**的含义:

<b>AL:</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
	设置波特率			奇偶校验设置		Stop位长度		字长选择

- 2) 返回参数: 通信口状态字节**AH**中的各位含义 (等同于**INS8250**串行接口芯片中的**LSR**)

<b>AH:</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
	<b>0</b>	<b>TSRE</b>	<b>THRE</b>	<b>BI</b>	<b>FE</b>	<b>PE</b>	<b>OE</b>	<b>DR</b>

调制解调器状态字节**AL**中的各位含义等同于**INS8250**串行接口芯片中的**MSR** (具体内容在第十章介绍)。

# 内容

- 4-0 概述
- 4-1 汇编语言程序格式
- 4-2 MSAM中的表达式
- 4-3 伪指令语句
- 4-4 DOS系统功能调用和BIOS中断调用
- 4-5 程序设计方法
- 4-6 宏汇编和条件汇编

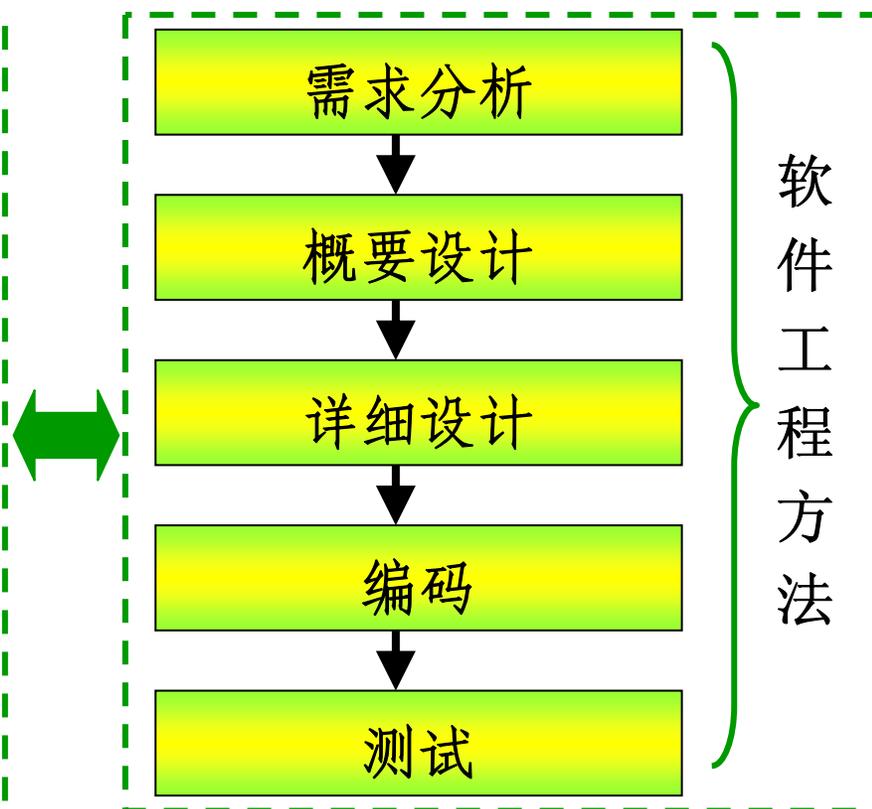
# 程序设计基本要求

- 实现基本功能，能够正确和正常运行；
- 结构模块化，简明、简捷、易读、易于调试、易于维护；
- 数据组织合理，充分发挥**CPU**、**Reg**和存储器的作用，占用系统资源少；
- 代码量小、执行速度快。

# 程序设计的一般方法与步骤

- 模块化设计方法：“自顶向下，逐步细化”
- 汇编语言与高级语言的程序设计步骤基本相同

- ① 分析问题并抽象出数学模型。
- ② 确定最佳算法。
- ③ 画出程序结构框图和流程图。
- ④ 合理分配内存工作单元和寄存器，并了解I/O接口地址。
- ⑤ 编程并调试（有时需用注释行说明，便于阅读和维护）



# 程序结构

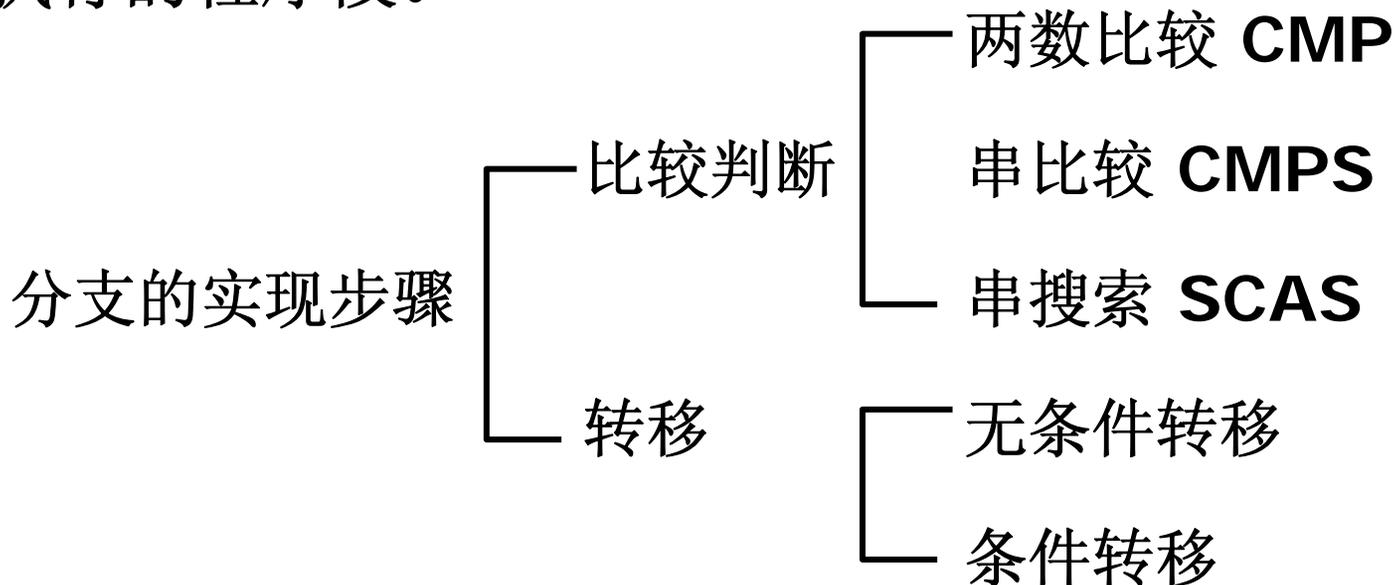
- 教材介绍四种结构：
  1. 顺序结构
  2. 分支结构
  3. 循环结构
  4. 子程序结构
- 前三种是基本的程序结构，奠定了实现任何复杂程序的基础。

## 1) 顺序结构

- 直线程序。按事件发展的先后，选择合适的指令有序地加以组合。

## 2) 分支结构

- 程序在顺序执行过程中，根据不同的计算结果或者不用的条件，由计算机自动判断和选择下一步所要执行的程序段。

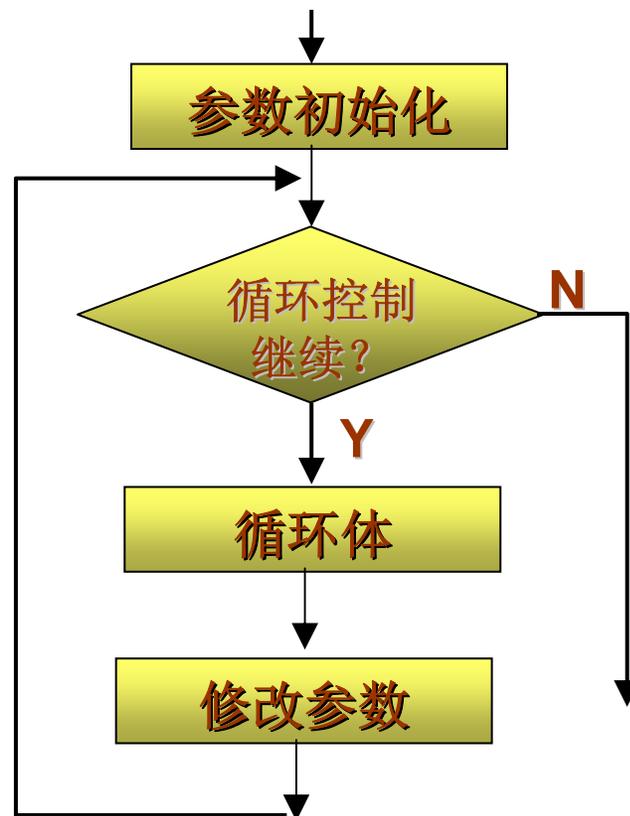
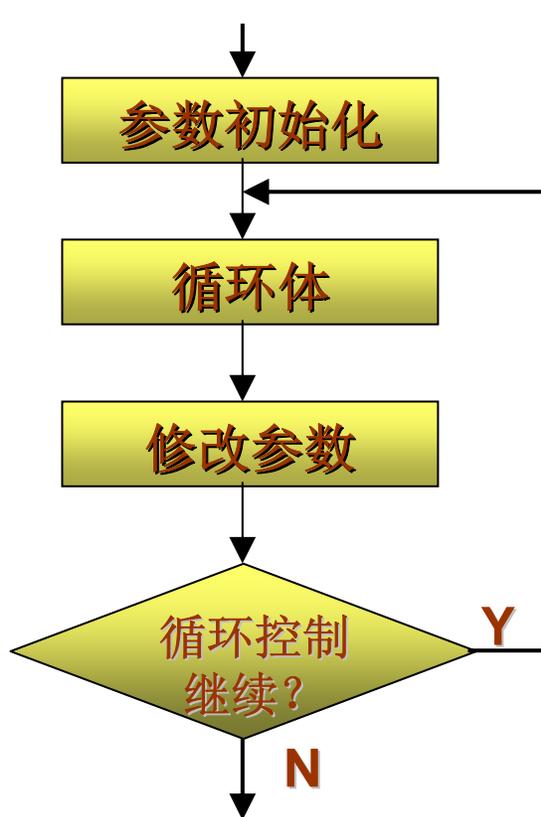


### 3) 循环结构

- 循环程序是在满足某些条件时对一段程序的重复执行，一般由四部分组成：
  - ① **初始化**：设置循环记数值(次数)和变量初值；
  - ② **循环体**：循环核心部分，满足条件时需执行的程序段；
  - ③ **修改参数**：修改源和目的操作数地址；
  - ④ **循环控制**：修改计数器值，判断控制条件，决定是否继续或者结束循环。

## 判断的两种形式:

- (1) **先执行后判断 (do—while)**: 先执行循环体至少一次, 再判断是否结束。适用于次数固定的循环。
- (2) **先判断后执行**: 先判断结束条件, 再决定是否执行循环体。适用于次数不固定的循环。



- **教材P176例4-69：先执行，后判断。**
  - **流程图P178图4-16**
  - **注意：**
    - ① **BX**中有**4**位十六进制数
    - ② **HEX**数‘**0**’~‘**9**’和‘**A**’~‘**F**’的**ASCII**码不连续
    - ③ 显示**1**位字符用功能**2**的**DOS**调用。
- **教材P176例4-70：先判断，后执行。**

## ● 循环控制方法

● 每个循环程序必须选择一个循环控制条件，控制循环的运行和结束。常用控制条件（方法）有：

- 1) 计数控制——循环次数已知，每循环一次加/减1。
- 2) 条件控制——循环次数未知，须根据条件控制循环。
- 3) 状态控制——根据事先设置或实时检测的状态来控制循环。
- 4) 逻辑尺控制——多次循环过程中分别做不同的操作时，可设置位串（**逻辑尺**）控制循环。逻辑尺是自左至右对应执行次数的二进制位串。每次循环逻辑尺左移一位，再根据**CF=1**或**0**执行不同的操作。

## 4) 子程序设计和调用技术

- 子程序不是一种基本的程序结构，但在程序设计时合理使用子程序是实现模块化程序设计的重要技巧。
- 有两种子程序：
  - ① 重复使用的程序段或具有通用性、便于共享的程序段（键盘处理、代码转换）；
  - ② 中断处理子程序：中断处理随机产生，对其处理只能采用子程序的形式。

## ● 与子程序有关的术语

- 子程序嵌套：子程序中调用别的子程序称为嵌套，只要堆栈空间允许，嵌套层次不限。
- 子程序递归调用：子程序调用该子程序本身称为递归调用。
- 可重入子程序：能够被中断并可再次被中断程序调用的子程序。
- 可重定位子程序：全部采用相对地址、可重定位在内存任意区域的子程序。

## (1) 子程序必须用过程定义语句定义

一般采用**FAR**属性，同一代码段可用**NEAR**，调用时的属性应与过程的属性对应。

```
CALL FAR PTR PROG
```

```
...
```

```
ENDP
```

## (2) 使用**CALL**须注意：

①保护断点

②保护寄存器内容

③主程序和子程序之间参数传递

### (3) 子程序必要的说明

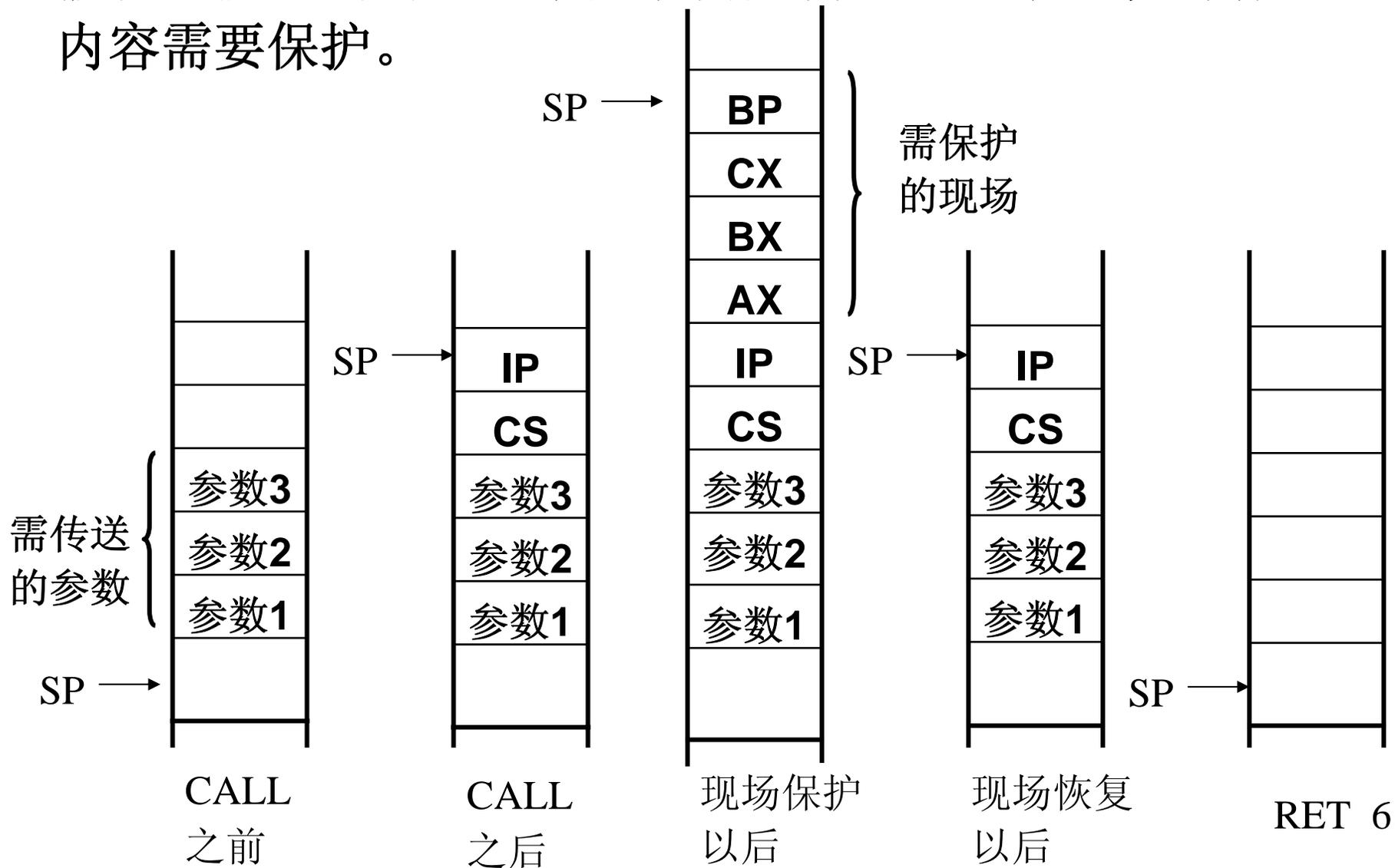
- ① 功能描述：名称、功能及性能；
- ② 子程序用到的寄存器及存储器单元，以便在调用时进行保护；
- ③ 入口参数和出口参数；
- ④ 子程序调用其它子程序的名称。

### ● 主程序和子程序之间参数传递

- a. 寄存器传递。传递速度快，适合参数较少的场合。
- b. 存储器传递。须建参数表，适合参数较多的场合
- c. 堆栈传递。参数较多时可用，适合子程序嵌套和子程序递归调用。

# 利用堆栈传递参数以及保护现场的过程示意图

假设：段间调用，三个参数需要传送，四个**16**位寄存器的内容需要保护。



# 习题4—2

**P206:**

**6、 7**

**9、 11**

**13、 15**

**18**（递归调用，选作）

**19**

**21**（只要求做前半题）

# 内容

- 4-0 概述
- 4-1 汇编语言程序格式
- 4-2 MSAM中的表达式
- 4-3 伪指令语句
- 4-4 DOS系统功能调用和BIOS中断调用
- 4-5 程序设计方法
- 4-6 宏汇编和条件汇编

# 一、宏汇编

- **宏**：源程序中一段独立的代码段（一连串的操作）。该段代码被定义为“宏(指令)”之后，即可用宏代替原来的代码，可以在程序中反复调用。
- 宏可以看作是汇编语言源程序的编写者自定义的组合指令
- 宏只存在汇编语言源程序中，在经过汇编后，宏就被展开。
- *学习要点：宏调用中的参数传递。*



## 2) 宏调用

- 格式：宏指令名 实参[1, 实参2...]
- 说明：
  - 已定义的宏可被调用，调用时实参与形参必须对应。

## 3) 宏展开

- 汇编程序在对汇编源程序进行汇编时，对每一个宏调用进行展开，即用宏体代替源程序中的宏指令名，用实参代替宏指令中的形参，每条插入的宏体指令前带上“+”符号。
- **P198例4-83**

## 4) 宏调用中的参数传递

- 调用时，用实参代替形参。实参可以是数字、寄存器或(部分)操作码。
- P198例4-85: 任意寄存器右移任意次 (<256次)**

宏定义:

```
SHIFT MACRO N, M  
    MOV CL, N  
    SAR M, CL  
ENDM
```

宏调用1:

```
SHIFT 4, AL
```

宏调用2:

```
SHIFT 6, DI
```

宏展开1:

```
+MOV CL, 4  
+SAR AL, CL
```

宏展开2:

```
+MOV CL, 6  
+SAR DI, CL
```

- 宏定义可用“部分操作码”作为参数，但在宏定义体中必须用‘&’作为分割符。

- P199例4-86**

宏定义:

```
SHIFT MACRO N, M, P
    MOV CL, N
    SAR M, CL
    S&P M, CL
ENDM
```

宏调用1:

```
SHIFT 3, AX, HR
```

宏调用2:

```
SHIFT 5, BH, AL
```

宏展开1:

```
+MOV CL, 4
+SAR AX,CL
+SHR AX, CL
```

宏展开2:

```
+MOV CL, 5
+SAR BH,CL
+SAL BH, CL
```

## 5) 宏定义嵌套

- 宏定义中可以宏调用，但是必须先定义后调用。

例4-89，先定义宏**DBF**，然后在宏**DBFS**定义中调用**DBF**

```
DBF  MACRO P, Q
    MOV BX, P
    ADD AX, Q
    ENDM
```

```
DBFS MACRO X1, X2, X3
    PUSH AX
    PUSH BX
    DBF X1, X2
    MOV X3, AX
    POP BX
    POP AX
    ENDM
```

宏调用：

```
DBFS 3AC0H, BX, [3AC4H]
```

宏展开：

```
+ PUSH AX
```

```
+ PUSH BX
```

```
+ DBF X1, X2 ; 不占内存
```

```
+ MOV BX, 3AC0H ; DBF宏展开
```

```
+ ADD AX, BX
```

```
+ MOV [3AC4H], AX
```

```
+ POP BX
```

```
+ POP AX ... ..
```

- 宏定义中允许嵌套，即宏定义体中又包括宏定义。
- 例4-90

```
DEFM MACRO MACN, OPER
```

```
MACN MACRO A, B, C
```

```
    PUSH AX
```

```
    MOV AX, A
```

```
    OPER AX, B
```

```
    MOV C, AX
```

```
    POP AX
```

```
    ENDM
```

```
ENDM
```

← **MACN**是内层宏定义名，也是外层宏定义的形参。

← **OPER**是外层的形参，在调用时，实参可以使用不同的操作码，从而实现不同的功能

## 6) 其它宏指令

### ① 取消宏指令

格式: **PURGE** 宏指令名[1, 宏指令名2, ...]

*P.201 例4-91:* 宏指令名**SUB**与指令助记符重名, 出现重名时**宏指令优先**, 使得同名指令或伪指令失效。宏调用后使用**PURGE**取消宏定义。

### ② 重复执行宏指令语句

格式: **REPT** 表达式 ; 表达式为重复次数  
宏体  
**ENDM**

*P.202 例4-92*



## 7) 宏指令与子程序的区别

- ① 完成操作的主体不同      (过程:CPU, 宏:汇编程序)
- ② 执行速度不同              (过程:速度较慢, 宏:快)
- ③ 占用存储空间不同        (过程:占用少, 宏:占用多)
- ④ 灵活性不同                (过程:灵活性较低, 宏:灵活性高)

——形参使得宏更有魅力

## 二、条件汇编

- 条件汇编

- 汇编程序对给定条件进行测试，并根据结果决定是否对某段程序进行汇编处理。

- 格式：

**IF**            条件（表达式）

                 指令体1                            ; **T**汇编指令体1

**ELSE**        指令体2                            ; **F**汇编指令体2

**ENDIF**

## ● 条件汇编的几种形式（P204表）

语句	条件	说明
<b>IF</b>	表达式	表达式值不等于 <b>0</b> 时为 <b>T</b> （条件满足）
<b>IFE</b>	表达式	表达式值等于 <b>0</b> 时为 <b>T</b> （条件满足）
<b>IFDEF</b>	符号	符号已定义或被说明为外部符号为 <b>T</b>
<b>IFDEFE</b>	符号	符号未定义或未说明为外部符号为 <b>T</b>
<b>IFB</b>	参数	参数为空时为 <b>T</b>
<b>IFNB</b>	参数	参数不为空时为 <b>T</b>
<b>IFIDN</b>	字符串 <b>1</b> , 字符串 <b>2</b>	两个字符串相同时为 <b>T</b>
<b>IFDIF</b>	字符串 <b>1</b> , 字符串 <b>2</b>	两个字符串不同时为 <b>T</b>

## 程序设计题：

用**8086**汇编语言设计一**ATM**机存取款软件的用户界面。要求进系统后屏幕显示“欢迎使用本系统”，另起行显示“输入密码：”。

允许三次输入密码，出错后可重新输入，三次输入都出错则返回**DOS**。密码输入正确即进入本人帐号，并在屏幕上显示：

“**1 返回DOS 2 查阅余额 3 存款 4 取款**”

然后根据所按的**1, 2, 3, 4**键选择所需工作方式。

以上各方式有如下功能：

**1 返回DOS-- 退出小系统，返回DOS**

**2 查阅余额-- 显示本人帐号内余额（元）**

**3 存款-- 输入存款数目，即与余额相加**

**4 取款-- 输入取款数目，即与余额相减**

按**2, 3, 4**键即进入相应方式，以**ENTER**键结束，结束后再显示：

“**1 返回DOS 2 查阅余额 3 存款 4 取款**”

说明：①屏幕显示可用英文，②帐号内要求存款额不小于十进制**6**位数。

# 习题4—3

**P206:**

**23**

**24**

**25 (选作题)**