第5章 汇编语言程序设计

5.1 顺序结构程序设计
5.2 分支结构程序设计
5.3 循环结构程序设计
5.4 子程序设计

汇编语言常见形式

5.5 模块化程序设计

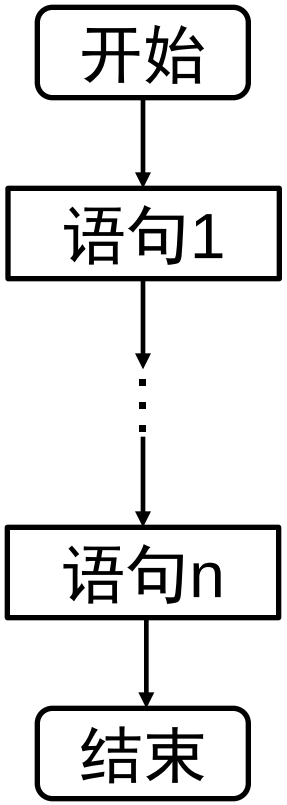
设计方法：模块化和结构化

5.6 常用DOS中断调用

区别：模块相互联系又相对独立，可分别编程、调试

5.1 顺序结构程序设计

✓完全按先后顺序逐条执行指令序列的程序，既不跳转也不循环，从头到尾一条一条执行语句，流程图如下：



5.1 顺序结构程序设计

例5.1 键盘输入任一按键，用十进制数输出相应按键的ASCII值。

```
CODE  SEGMENT
        ASSUME CS:CODE
START:MOV AH,1          AL
        INT 21H         ;读入任意按键
        MOV AH,0        ;AX保存按键的ASCII值
        MOV BL,100      被除数AX, AH余数, AL商
        DIV BL          ;ASCII除以100,取百位数字
        MOV CL,AL       ;CL存放百位数字
```

5.1 顺序结构程序设计

✓ADD CL,30H ;CL存放百位数字的ASCII

✓MOV AL,AH MOV AH,0 MOV BL,10

✓DIV BL ;ASCII值除以10，取十位数字

✓ADD AL,30H ADD AH,30H 被除数AX, AH余数, AL商

✓MOV BX,AX ;AX存放十位数字和各位数字

✓MOV AH,2 显示输出字符, DL为其ASCII码

✓MOV DL,13 INT 21H MOV DL,10 INT 21H 0AH, 0DH

✓MOV DL,CL INT 21H MOV DL,BL INT 21H

✓MOV DL,BH INT 21H MOV AH,4CH INT 21H

✓CODE ENDS END START

5.1 顺序结构程序设计

例5.2 编写程序，计算Z=((W-X)/10*Y)的平方值，R为相处所得余数，其中W，X，Y均为八位有符号二进制数。

- ✓ DATA SEGMENT
- ✓ W DB -128
- ✓ X DB 127
- ✓ Y DB -100
- ✓ R DB 0
- ✓ Z DD 0
- ✓ DATA ENDS

5.1 顺序结构程序设计　Z=((W-X)/10*Y)

✓CODE SEGMENT

✓　　ASSUME CS:CODE,DS:DATA,SS:STACK

✓BEGIN:MOV AX,DATA　MOV DS,AX

✓　　MOV AL,X　CBW　　X转为字，先放AX，再放BX

✓　　MOV BX,AX　MOV AL,W　CBW　　W转为字，放AX

✓　　SUB AX,BX　MOV BL,10　IDIV BL　　被除数AX

✓　　MOV R,AH　IMUL Y　IMUL AX　　字节AL、字乘法AX

余数AH

✓　　MOV WORD PTR Z,AX

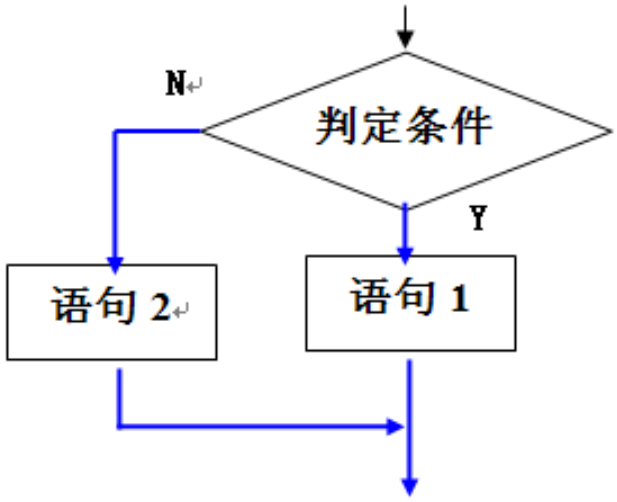
✓　　MOV WORD PTR Z+2,DX

✓　　MOV AX,4CH　INT 21H

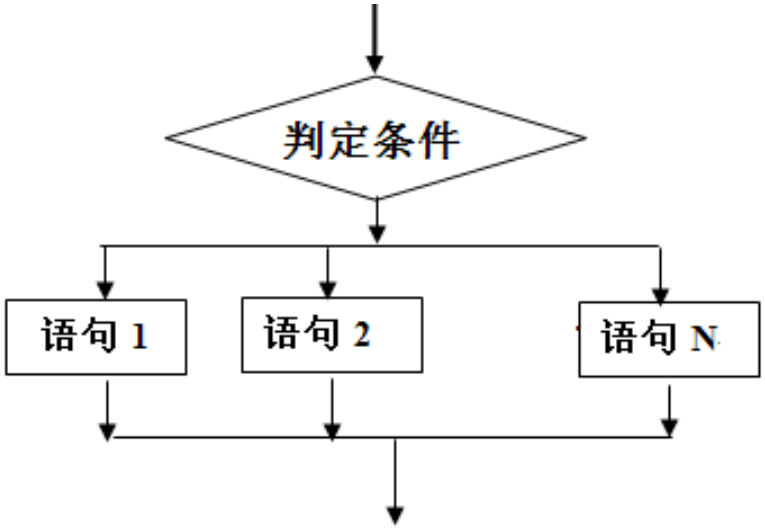
✓CODE ENDS　END BEGIN

5.2 分支结构程序设计

- ✓需要对不同情况或条件作出不同处理，这样的程序需要采用分支结构。
- ✓两种形式：二路分支机构和多路分支结构。



(1) IF THEN ELSE



(2) CASE

特点：只能执行一个分支

实现：比较和条件转移指令

5.2 分支结构程序设计：二路分支

例5.3 从键盘输入一位数字，判断其奇偶性，并在屏幕输出一个标志，若为奇数，则输出1，否则输出0。

判断该数字的ASCII码的最低位

```
CODE   SEGMENT
       ASSUME  CS：CODE
BEGIN:MOV AH, 01H  ;调用DOS中断的1号子功能
       INT 21H        ;AL←键入数字ASCII
       CLC            ;CF清0      P87
       RCR AL, 1      ;AL最低位移入CF      P72
       JNC EVN        ;根据CF判断输入数奇偶性      P81
```

5.2 分支结构程序设计：二路分支

✓MOV AL, 31H ;奇数，AL←1的ASCII码

✓JMP DISP

✓EVN：MOV AL, 30H ;偶数，AL←0的ASCII码

✓ MOV BL, AL

✓DISP：MOV AH, O2H ;调用DOS的2号子功能

✓ MOV DL, 0AH INT 21H;输出换行

✓ MOV DL, 0DH INT 21H;输出回车

✓ MOV DL, BL INT 21H;输出标志字符

✓ MOV AH, 4CH ;返回DOS

✓ INT 21H

✓ CODE ENDS END BEGIN

5.2 分支结构程序设计：二路分支

例5.4 从键盘上输入0到9中任一自然数，求其立方值,若输入数据不是0－9，显示“INPUT ERROR!”。

乘法/立方表

查表法：X，字单元TAB + 2*X

✓DATA SEGMENT

✓ INPUT DB “PLEASE INPUT X(0...9)：$”

✓ TAB DW 0,1,8,27,64,125,216,343,512,729

✓ X DB ?

✓ Y DW ?

✓ INERR DB 0DH，0AH， “INPUT ERROR!$“

✓DATA ENDS

5.2 分支结构程序设计：二路分支

✓CODE SEGMENT

✓ ASSUME CS:CODE,DS:DATA,SS:STACK

✓BEGIN: MOV AX, DATA MOV DS, AX

✓ MOV DX, OFFSET INPUT MOV AH, 9 INT 21H 9功能

✓ MOV AH, 1 INT 21H CMP AL, '0' JB LERR P64/82 1功能

✓ CMP AL, '9' **JA** LERR AND AL, 0FH 错误P67屏蔽高4位

✓ MOV X, AL ADD AL, AL MOV BL, AL 自然数

✓ MOV BH, 0 MOV AX, TAB[BX] MOV Y, AX

✓EXIT: MOV AH, 4CH INT 21H

✓LERR: MOV DX,OFFSET INERR MOV AH,9 9功能/错误

✓ INT 21H JMP EXIT

✓CODE ENDS END BEGIN

5.2 分支结构程序设计：多路分支

例5.5 任意给定X值（-128≤X≤127），

$$Y=\begin{cases}1 & \text{当}X>0\\ 0 & \text{当}X=0\\ -1 & \text{当}X<0\end{cases}$$

用二路分支实现多路分支

- ✓DATA SEGMENT
- ✓ X DB -18
- ✓ Y DB ?
- ✓DATA ENDS
- ✓CODE SEGMENT
- ✓ ASSUME CS: CODE, DS: DATA

5.2 分支结构程序设计：多路分支

✓START:MOV AX, DATA MOV DS, AX

✓MOV AL, X

✓CMP AL,0;此处可用SUB AL,0或OR/AND AL,AL

✓JGE BIGE P83

✓MOV AL, 0FFH ；（AL）<0，AL←-1

✓JMP OUTY ;使用无条件转移指令，保证同一出口

✓BIGE：JE OUTY ；（AL）=0 P81

✓ MOV AL, 1 ；（AL）>0，AL←1

✓OUTY：MOV Y, AL 结果放在Y中

✓ MOV AH, 4CH ；返回DOS

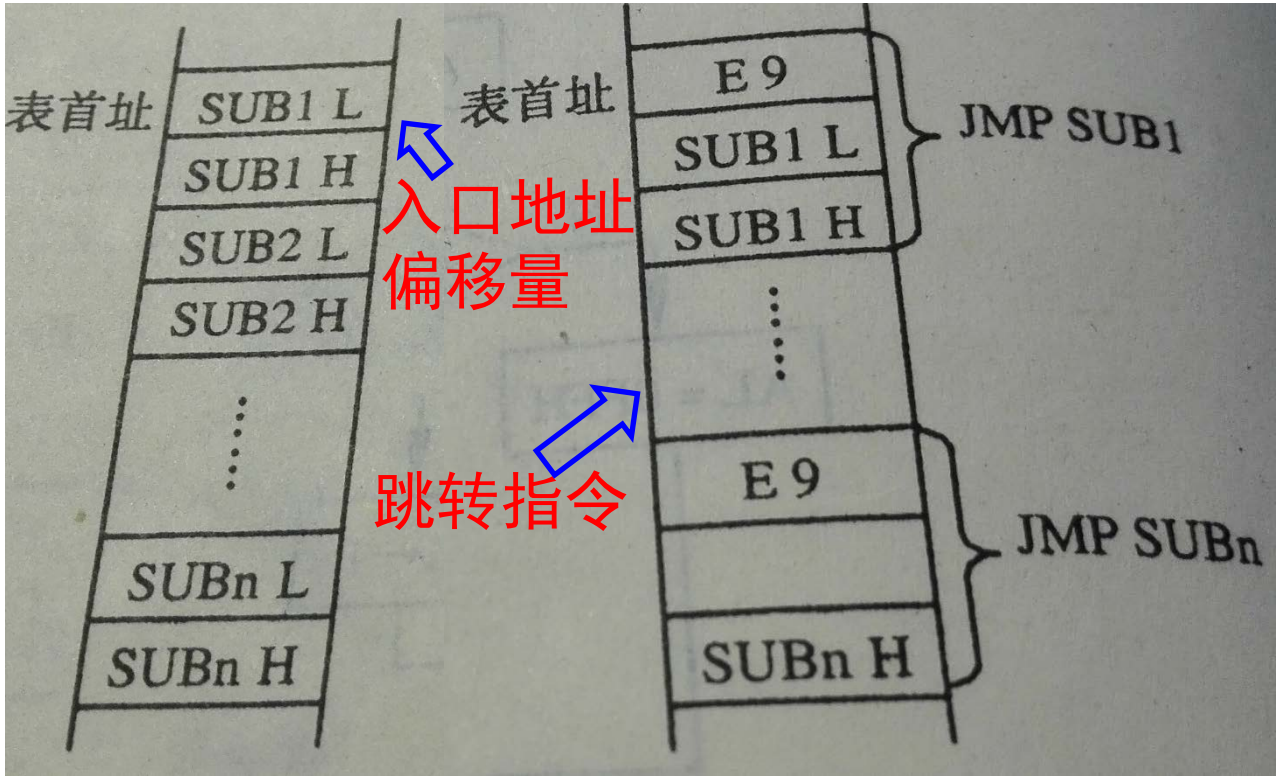
✓ INT 21H

✓CODE ENDS ENDS START

5.2 分支结构程序设计：多路分支

✓用跳转表实现多路分支：

✓假设n路分支，每路分支的入口地址组成一个表，叫跳转表。



表地址=表首址+偏移量

左：表地址=表首址+(k-1)*2 (k为1, …, n)

右：表地址=表首址+(k-1)*3 (k为1, …, n)

5.2 分支结构程序设计：多路分支

例5.6 设某程序具有8种功能，它们的入口地址分别为SUB1～SUB8，用表内存放入口地址的方法编程，执行第3种功能。

```
✓DATA  SEGMENT
✓BASE  DW
  SUB1,SUB2,SUB3,SUB4,SUB5,SUB6,SUB7,SUB8
✓BN     DB 3  ;取功能号3
✓DATA  ENDS
✓STACK  SEGMENT  PARA  STACK
✓          DW 30 DUP (0)
✓STACK  ENDS
```

5.2 分支结构程序设计：多路分支

✓COSEG SEGMENT

✓ASSUME CS：COSEG，DS：DATA，SS：STACK

✓BEGIN: MOV AX, DATA MOV DS, AX

3 ✓MOV AL, BN MOV AH, 0 ;AX←（BN）

K-1 ✓DEC AL SHL AL, 1 ;取功能3的表内偏移量 P69, (k-1)*2

✓MOV BX,OFFSET BASE ADD BX,AX; 3的有效地址

✓MOV AX, [BX] JMP AX ;转SUB3执行

✓SUB1: ┆ JMP EXIT ┆ SUB2: ┆ JMP EXIT

✓SUB3: ┆ JMP EXIT ┆ SUB8: ┆ JMP EXIT ┆

✓EXIT: MOV AH, 4CH INT 21H

✓COSEG ENDS ENDS BEGIN

5.2 分支结构程序设计：多路分支

例5.7 对于例5.6采用表内存放无条件转移指令的方法编程，执行功能3。

- ✓DATA SEGMENT
- ✓BN DB 3 ;取功能号3
- ✓DATA ENDS
- ✓STACK SEGMENT PARA STACK
- ✓ DW 30 DUP (0)
- ✓STACK ENDS
- ✓BEGIN: MOV AX, DATA MOV DS, AX

5.2 分支结构程序设计：多路分支

错误

- ✓MOV BH, 0 MOV BL, BN ;(BL) = 3
- ✓DEC BL ;(BL) = 2 MOV AL, BL ;(AL) = 2
- ✓SHL BL, 1 ;(BL) = 4
- ✓ADD BL, AL ;(BL)乘3，取功能3的表内偏移量 (k-1)*2
- ✓ADD BX, OFFSET BASE JMP BX
- ✓BASE:JMP SUB1 ;表首地址为BASE
- ✓JMP SUB2 JMP SUB3 JMP SUB4 JMP SUB5
- ✓JMP SUB6 JMP SUB7 JMP SUB8
- ✓SUB1: ┆ JMP NEXT SUB2: ┆ JMP NEXT
- ✓SUB3: ┆ JMP NEXT ┆ SUB8: ┆ JMP NEXT ┆
- ✓NEXT: MOV AH, 4CH INT 21H

5.2 分支结构程序设计：多路分支

✓在跳转表的每一个字地址或指令前开辟一个关键字单元，用于存放该字地址或指令的关键字地址或指令的关键字值。

根据关键字实现多路分支

例5.8

✓DATA　SEGMENT

✓BASE　DB　1 ;关键字

✓DW SUB1;外设服务程序入口

✓DB　2 DW　SUB2　DB　4　DW　SUB3　┆

✓DB　80H　DW　SUB8

✓DATA　ENDS

5.2 分支结构程序设计：多路分支

- ✓MAIN　PROC　FAR
- ✓BEGIN:　MOV　AX, DATA　MOV　DS, AX
- ✓READ:　MOV　AH, 1 ;读功能号　INT　21H
- ✓SUB AL,30H JE　READ　MOV　BX, OFFSET BASE

跳回

- ✓RSH: CMP　AL, [BX] ;与关键字比较
- ✓JE BRCH INC BX INC BX INC BX;进下一关键字
- ✓JMP RSH
- ✓BRCH: MOV　CX, [BX+1]　JMP　CX
- ✓SUB1：……
- ✓SUB2：……

5.3 循环结构程序设计

把有规则重复执行的程序段称为循环。包含循环程序段的程序称为循环程序结构。

4个组成部分：

✓(1) 初始化部分：初始状态，计数器初值；

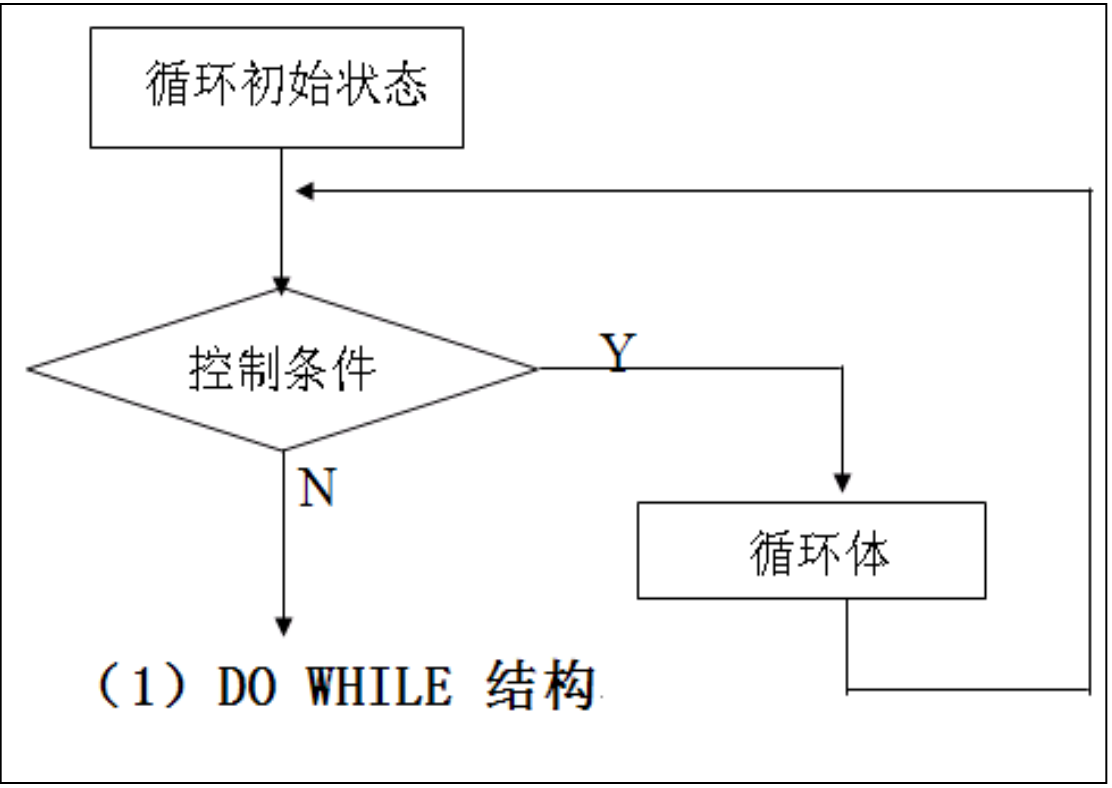
✓(2) 循环体：主体部分，特定功能程序段；下次循环准备，如修改计数器的值；

✓(3) 循环控制部分：判断，是否继续或退出

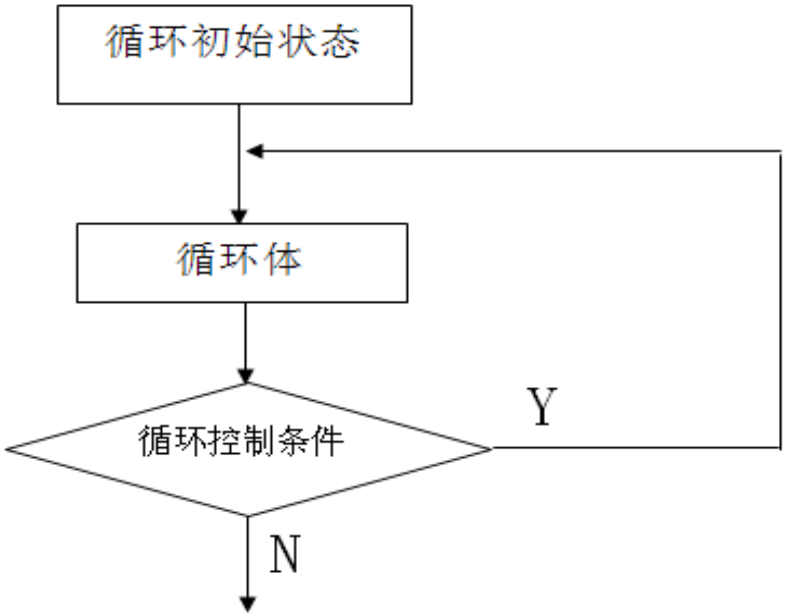
✓(4) 循环结束部分：结束时的数据处理、结果存放等操作。

5.3 循环结构程序设计

- ✓循环控制：“死循环”；方法，包括计数器控制法、条件控制法、逻辑尺控制法等。
- ✓循环结构：当型和直到型



（1）DO WHILE 结构



（2）DO UNTIL 结构

5.3 循环结构程序设计：单重循环

✓当一个循环结构内不再包含循环结构时，称为单重循环结构；否则，为多重循环或嵌套循环结构。

循环次数已知的循环结构：

例5.9 以X为首地址的存储单元中存放20个无符号数，试从中找出最大值送入Y单元中。

分析题意

✓DATA SEGMENT

✓X DB 11,22,13,24,56,76,78,55,64,44

✓ DB 56,64,63,45,43,32,47,89,99,69

✓Y DB ？

✓DATA ENDS

5.3 循环结构程序设计：单重循环

✓CODE SEGMENT

✓ ASSUME CS：CODE，DS：DATA

✓MAIN PROC FAR

✓START:PUSH DS MOV AX，0 PUSH AX

✓MOV AX，DATA MOV DS，AX

✓MOV AL，X MOV BX，OFFSET X 第一个数放在AL中

✓MOV CX，19 ;设置比较次数

2 ✓L1:INC BX CMP AL,[BX] JAE L2 XCHG AL,[BX] P82

✓L2:LOOP L1 MOV Y，AL RET P85，CX≠0

✓MAIN ENDP CODE ENDS END START

5.3 循环结构程序设计：单重循环

例5.10 已知BUF中存放N个数，编程将大于等于0的数放入BUF1中,小于0的数放入BUF2中。

- ✓ DATA SEGMENT
- ✓ BUF DW -2,5,-3,6,100,0,-20,-9,8,-110,20,0
- ✓ N=($-BUF)/2 数的个数，P94
- ✓ BUF1 DW N DUP(?)
- ✓ BUF2 DW N DUP(?)
- ✓ DATA ENDS

5.3 循环结构程序设计：单重循环

✓CODE SEGMENT

✓ASSUME CS：CODE，DS：DATA，SS：STACK

✓BEGIN:MOV AX, DATA MOV DS, AX

✓ LEA BX,BUF LEA SI, BUF1 LEA DI, BUF2 偏移地址

✓ MOV CX, N ;设置循环次数→CX

✓ LOPA: MOV AX, [BX] CMP AX, 0 JGE L1 P83

✓ MOV [DI], AX ADD DI, 2 JMP NEXT

✓ L1:MOV [SI], AX ADD SI, 2

✓ NEXT:ADD BX, 2 DEC CX JNE LOPA 下一个数，P81

✓ MOV AH,4CH INT 21H 不等于0转

✓CODE ENDS END BEGIN

5.3 循环结构程序设计：单重循环

循环次数未知的循环结构：

例5.11 已知在以BUF为首地址的字节存储区中，存放着一个以“$”作结束标志的字符串。试编写程序，在CRT上显示该字符串，并要求将小写字母以大写字母形式显示出来。

分析题意：判断$；小写字母(-20H)

- ✓ DATA SEGMENT
- ✓ BUF DB′ADD AX, bx sub CX,10 mov DX, 1234H END$′
- ✓ DATA ENDS

5.3 循环结构程序设计：单重循环

✓ LEA BX, BUF

循环 ✓LOPA:MOV DL,[BX] CMP DL,“$” JE EXIT 是否相等

✓ CMP DL, “a” JB N CMP DL, “z“ JA N 书上错误

✓ SUB DL,20H ;(DL)－20H→DL，小写变大写

✓N： MOV AH，2 2功能

✓ INT 21H ;显示DL中的字符

✓ INC BX ;BX增1，指向下一字符

✓ JMP LOPA ;转LOPA继续循环

✓EXIT: MOV AH，4CH INT 21H

✓CODS ENDS END BEGIN

条件控制法

5.3 循环结构程序设计：单重循环

例5.12 从自然数1开始累加，直到累加和大于500为止，　条件
统计被累加的自然数的个数，并把统计的个数送入单元N中，把累加和送入单元SUM中。

✓DATA SEGMENT
✓　　N DW ?
✓　　SUM DW ?
✓DATA ENDS
✓STACK SEGMENT PARA STACK ‘STACK’
✓　　DW 2OO DUP (?)
✓STACK ENDS

5.3 循环结构程序设计：单重循环

✓CODE SEGMENT

✓MAIN PROC FAR

✓ASSUME CS：CODE, DS：DATA, SS：STACK

✓START:PUSH DS MOV AX, *0* PUSH AX

✓ MOV AX, DATA MOV DS, AX

✓ MOV AX, *0* MOV BX, *0*

✓L: INC BX ADD AX, BX CMP AX, 500

✓ JBE L AX≤500

✓ MOV N, BX MOV SUM, AX RET

✓MAIN ENDP CODE ENDS END START

5.3 循环结构程序设计：单重循环

逻辑尺控制法实现循环结构：

例5.13 屏幕间隔显示”Y”, ”N”两个字符(各显示4次)

```
✓START: MOV  CX, 8        逻辑尺10101010，0为N，1为Y
✓       MOV  AH, 2        2功能
✓       MOV  bL, 10101010B
✓L1:    SHL  bL,1  JNC  L2  MOV  DL, "Y"    CF
✓       INT  21H   jmp  l3
✓L2:    MOV  DL, "N"   INT  21H
✓l3:    LOOP L1
✓       MOV AH,4CH  INT 21H
✓CODE   ENDS   END   START
```

5.3 循环结构程序设计：多重循环

多重循环：又称循环的嵌套，即循环之内套循环。应仔细考虑各重循环的控制条件及其采用的方法，相互之间不可混淆。

- ✓(1)内循环必须完整地包含在外循环内，内外循环不能相互交叉；
- ✓(2) 内循环位置任意，但应避免混淆；
- ✓(3) 多个内循环可拥有同一外循环，它们之间可嵌套可并列；
- ✓(4) 可从内循环直接跳到外循环，逆向不可；
- ✓(5) 通过外循环再次进入内循环时，内循环初始条件必须重新设置。

5.3 循环结构程序设计：多重循环

例5.14 设有4个学生参加5门课的考试，其中4名学生的学号和各科成绩存放在字数组SCORE中，计算每个学生的平均成绩且存入字数组AVERAGE中。

✓DATA　SEGMENT
✓SCORE DW 1，68，78，85，72，83
✓　　DW 2，72，83，88，91，78
✓　　DW 3，99，91，88，76，77
✓　　DW 4，76，77，80，87，85
✓AVERAGE DW 4 DUP （？）
✓DATA　ENDS

5.3 循环结构程序设计：多重循环

✓CODE SEGMENT

✓ASSUME CS：CODE，DS：DATA

✓START:MOV AX, DATA MOV DS, AX

✓MOV BX,OFFSET SCORE MOV DI,OFFSET AVERAGE

✓ MOV CX, 4 外循环计数 保护外循环计数，CX=?

✓L1:MOV AX,0 ADD BX，2 PUSH CX MOV CX, 5

总✓L2:ADD AX, [BX] ADD BX，2 LOOP L2 CX 内循环计数

✓MOV DL, 5 DIV DL MOV [DI], AL ADD DI, 2

✓POP CX LOOP L1 商放入AVERAGE

✓MOV AH, 4CH INT 21H

✓CODE EDNS END START

5.3 循环结构程序设计：多重循环

例5.15 已知有N个元素存放在以Buf为首址的字节存储区中,用冒泡排序法将它们按从小到大的顺序排列在Buf存储区中。

冒泡法思路是相邻两个数比较，将小的数调到前头，依此类推。

✓STACK SEGMENT STACK
✓ DB 200 DUP(0)
✓STACK ENDS
✓DATA SEGMENT
✓BUF DB 30H,10H,40H,20H,50H,70H,60H,90H,80H,0,0FFH
✓N=$-BUF DATA ENDS

5.3 循环结构程序设计：多重循环

✓CODE SEGMENT

✓ ASSUME CS:CODE,DS:DATA,SS:STACK

✓BEGIN:MOV AX,DATA MOV DS,AX

✓ MOV SI,1 元素的序号

✓LOPI:MOV DI,SI INC DI MOV AL,[BUF+SI-1] 第1个元素值

✓LOPJ: CMP AL,[BUF+DI-1] JBE NEXT 与第2个元素比较

✓ XCHG [BUF+DI-1],AL MOV [BUF+SI-1],AL 小数第1元素

✓NEXT: INC DI CMP DI,N JBE LOPJ 第3个元素

✓ INC SI CMP SI,N-1 JBE LOPI 冒泡1次后取下个元素

✓MOV AH,4CH INT 21H

✓CODE ENDS END BEGIN

5.4 子程序设计

- ✓子程序(或过程)：功能相同结构类似，仅某些变量初值不同的程序段；独立出来，按一定的格式编写，成为可以被其他程序多次调用的程序模块。
- ✓主程序、子程序

调用与返回：

- ✓调用指令：CALL OPD (子程序名/入口地址)
- ✓说明：返回地址(CALL指令后第一条指令的地址，称为断点地址，即当前指令CALL的CS：IP的内容)压入堆栈保护，以便子程序返回；然后子程序入口地址送入CS：IP；结束后，断点地址从堆栈中弹回CS：IP中。

5.4 子程序设计

调用方式：段内直接、段内间接、段间直接、段间间接（位置关系和寻址方式）

- ✓(1) 段内直接: CALL 目标地址; SP<-(SP)-2, (SP)<-(IP); IP<-OPD 例5.16
- ✓(2) 段内间接: CALL REG/MEM; SP<-(SP)-2, (SP)<-(IP); IP <- REG/MEM
- ✓(3) 段间直接: CALL 目标地址; SP<-(SP)-2, (SP)<-(CS), SP<-(SP)-2, (SP)<-(IP); CS<-子程序段基址, IP<-OPD 例5.17
- ✓(4) 段间间接: CALL 双字MEM; SP<-(SP)-2, (SP)<-(CS), SP<-(SP)-2, (SP)<-(IP); CS<-OPD的高16位, IP<-OPD的低16位

5.4 子程序设计

- ✓子程序返回指令RET: RET [n]
- ✓说明:
- ✓(1) 段内返回和段间返回;
- ✓(2) n为任选项，为立即数且为偶，此时: SP<-(SP)+n个字节，废除N个参数
- ✓段内返回：IP<-([SP]), SP<-(SP)+2
- ✓段间返回：IP<-([SP]), SP<-(SP)+2
- ✓ CS<-([SP]), SP<-(SP)+2

5.4 子程序设计

设计方法：　　伪指令

- ✓ 定义：<过程名> PROC <类型属性> 过程体 <过程名> ENDP (地址，属性，CALL/RET属性)　　例5.18/5.19
- ✓ 调用与返回：自定义堆栈，注意堆栈状态
- ✓ 现场的保护和恢复：主程序和子程序共用寄存器；一进入子程序后就应把子程序中所使用到的寄存器内容保护到堆栈中，而在退出前再把堆栈中的内容恢复到原寄存器中。例5.20
- ✓ 那些保存（子程序用到）？那些不必要或不应该（传递参数寄存器）？

5.4 子程序设计

子程序参数传递：

- ✓ 传递参数，返回数据；这种数据传递称为参数传递。
- ✓ 类型：寄存器法、约定单元法、参数表法和堆栈法。
- ✓ 寄存器法：最常用，参数较少情况（个数限制）；传递参数放入寄存器，然后在子程序中从寄存器中取出。例5.21
- ✓ 约定单元法：主程序和子程序在同一源文件中，可把入口参数和出口参数都放到事先约定的共享存储单元中，则子程序可直接访问该变量。例5.22
- ✓ 参数表法：将参数组织成一个参数表，存放在内存或外设端口中，然后用寄存器将表地址传递给子程序；适用于大量参数传递　例5.23

5.4 子程序设计

- ✓堆栈法：适用于参数多、子程序嵌套调用和递归调用的情况。
- ✓实现方法：调用程序在调用子程序之前将参数压入堆栈，子程序在堆栈中取得参数将参数。

例5.24/5.25

5.4 子程序设计

嵌套子程序：

✓一个子程序调用另一个子程序，这种结构就称为子程序的嵌套。

✓嵌套深度：嵌套的层数（不限制，足够的堆栈空间）

✓图5.6：嵌套深度为4

✓注意：寄存器的保护和恢复，避免各子程序之间因寄存器冲突而出错。

✓例5.26

5.4 子程序设计

✓**递归子程序**：

✓一个子程序调用它自己，这种调用称为递归调用，这样的子程序称为递归子程序。

✓注意：频繁使用堆栈，注意堆栈溢出问题；应防止变量及返回地址在递归调用中的冲突。

✓例5.27

5.5 模块化程序设计

基本概念：

✓采用“分而治之”的策略，将一个大的程序模块分割成一些子模块，使得每一个子模块都成为功能单一、结构清晰、接口简洁、容易理解的子程序。

✓子程序是实现模块化程序设计的最佳方法。

✓一个程序可由多个文件组成，一个文件可由多个子程序组成。一个系统有且仅有一个主程序，其他子程序可以单独组织成一个独立的库文件以备调用。

5.6 常用DOS中断调用

- ✓DOS中提供了多个系统功能调用，即中断调用，主要分为：设备管理、文件管理、目录管理和其他功能调用四大类。
- ✓一般过程：将调用号放入寄存器AH中，设置好入口参数，然后执行软中断语句“INT 21H”。
- ✓键盘输入（1号调用）：单个字符的ASCII码送入AL
- ✓显示输出（2号调用）：DL中字符宋显示器显示
- ✓显示字符串（9号调用）：DS：DX所指向的，以$结尾的字符串送显示器显示
- ✓键盘输入字符串（10号调用）：从键盘上往DS：DX所指向的输入缓冲区输入字符串