第五章
程序设计的其它方法与技术

一、本章的学习内容：

本章学习汇编语言的高级程序设计技术：

（1）串操作指令的使用格式及功能；

（2）宏指令的定义与调用方式；

（3）模块程序设计方法及连接技术。

通过本章的学习，能提高编程的效率和质量，简化程序设计的工作，这是汇编语言中最具特色的部分。

二、本章的学习重点：

（1）串操作指令MOVS、CMPS、SCAS的使用格式及功能；

（2）简单宏指令的定义与调用方式；

（3）模块程序设计的方法。

三、本章学习的难点：

（1）MOVS与MOV、CMPS与CMP功能上的差别及串操作指令的正确使用方法；

（2）宏指令的定义与调用方式；

（3）模块之间的组合、定位及通信方式；

（4）模块化程序设计技术。

第五章
程序设计的其它方法与技术

5.1 字符串操作指令

5.2 宏功能程序设计

5.1 字符串操作指令

(1) 串传送指令 MOVS

(2) 串比较指令 CMPS

(3) 串搜索指令 SCAS

(4) 取字节/字/双字串指令 LODS

(5) 存储字节/字/双字指令 STOS

(6) 总结

（1） 串传送指令 MOVS

- 格式：**MOVS OPD，OPS** 或
  - **MOVSW** 字串传送
  - **MOVSB** 字节串传送
  - **MOVSD** 双字串传送
- 功能：(1) (DS:[SI]/[ESI]) → ES:[DI]/[EDI]

  (2) 修改串指针，使之指向下一元素。修改方式为：

  i. 当DF＝0时，(SI)/(ESI)增量1(字节操作)或2(字操作)或4(双字操作)

  ii. 当DF＝1时，(SI)/(ESI)减量1(字节操作)或2(字操作)或4(双字操作)

说明：(1) 该指令可带的重复前缀为：**REP**

即当(CX/ECX)≠0时连续传送，直至(CX/ECX)=0。

(2) 源串应在当前数据段，指针为SI/ESI；目的串应在当前附加数据段指针为DI/EDI。

（1） 串传送指令 MOVS

例1：将以STR1为首址的字节存储区中存放的字符串传送到以STR2为首址的字节存储区中。

```
 .386
DATA   SEGMENT   USE16
STR1   DB    'ABCDEFGHIJKLM ……'  ; 输出缓冲区BUF1
COUNT  EQU    $-BUF1              ; BUF1中的字符个数
STR2   DB     COUNT DUP (0)       ; 输入缓冲区BUF2
DATA   ENDS
STACK  SEGMENT   USE16   STACK
       DB    200 DUP(0)
STACK  ENDS
CODE   SEGMENT   USE16
       ASSSUME   DS:DATA,ES:DATA,CS:CODE,SS:SATCK
```

(1) 串传送指令 MOVS

```
START:MOV   AX, DATA
      MOV   DS, AX      ;
      MOV   ES, AX
      LEA   SI, STR1    ;
      LEA   DI, STR2    ;
      MOV   CX, COUNT   ;
      CLD               ;
  REP MOVSB             ;
      MOV   AH, 4CH
      INT   21H
CODE  ENDS
      END   START
```

CX

16

反向操作的源程序:

```
      STD
      LEA  SI,BUF1+COUNT-1
      LEA  DI,BUF2+COUNT-1
      MOV  CX,COUNT
  REP MOVSB
```

"REP MOVE"语句代替了以下程序段

```
P:  MOV  AL, [SI]
    MOV  [DI],AL
    DEC   SI
    DEC   DI
    LOOP   P
```

由于DF=0为默认状态,故须使用STD指令设置DF=1,才能反向操作.

（2）串比较指令　CMPS

- 格式: CMPS　OPD，OPS　或 { CMPSW　字串比较 / CMPSB　字节串比较 / CMPSD　双字串比较 }

- 功能:

(1) (DS: [SI]/[ESI])—(ES: [DI]/[EDI])，即将SI/ESI所指的源串中的一个字节(或字、双字)中的数与DI、EDI所指的目的串中的一个字节(或字、双字)中的据相减，并根据相减的结果设置标志位，结果并不保存。

(2) 修改串指针，使之指向串中的下一个元素。修改方式为:

i. 当DF = 0时，(SI)/(ESI)和(DI)/(EDI)增量1(字节操作)或2(字操作)或4(双字操作)。

ii. 当DF = 1时，(SI)/(ESI)和(DI)/(EDI) 减量1(字节操作)或2(字操作) 或4(双字操作)。

（2）串比较指令　CMPS

说明:(1) 源串放在当前数据段中,指针为SI/ESI;目的串放在当前附加数据段中,指针为DI/EDI;

(2) 该指令可带的重复前缀为:

a. REPE/REPZ　当(CX/ECX)≠0时,如两串对应字符相等继续比较,不相等跳出循环;

b. REPNE/REPNZ　当(CX/ECX)≠0时,如两串对应字符相等继续比较,不相等跳出循环;

例2

（2）串比较指令 CMPS

例2 从键盘输入一字符串至STR1为首址的字节缓冲区中，试比较该串与字节字符串STR2是否相等，相等则0→BX;不等-1→BX。

```
.386
DATA    SEGMENT USE16                    } 输入缓冲区
STR1    DB 80
        DB 0                        ;待比较字符串
        DB 80 DUP(0)                ;待比较字符串长度
STR2    DB ' WAN1、ASM'
COUNT   EQU $-STR2
DATA    ENDS
STACK   SEGMENT USE16 STACK
        DB 200 DUP(0)
STACK   ENDS
CODE    SEGMENT USE16
        ASSUME DS:DATA，ES:DATA，SS:STACK，CS:CODE
```

（2）串比较指令 CMPS

```
START：  MOV  AX，DATA
         MOV  DS，AX          } 当前数据段和当前附加数据段重合
         MOV  ES，AX
         LEA  DX，STR1
         MOV  AH，10          } 输入一串字符至STR1缓冲区
         INT  21H
         MOV  AL，STR1＋1
         CMP  AL，COUNT       比较两串长度是否相等，不等转EXIT
         JNE  EXIT
         LEA  SI，STR1＋2
         LEA  DI，STR2
         MOV  CX，COUNT
  REPZ   CMPSB                ；逐一比较两串的对应字符是否相等
         JNE  EXIT            ；不等，转EXIT
         MOV  BX，0           ；相等，0→BX
         JMP  RETU
EXIT：   MOV  BX，-1
RETU：   MOV  AH，4CH
         INT  21H
CODE     ENDS
END      START
```

（3）串搜索指令 SCAS

- 格式：SCAS OPD 或 { SCASB 字节串搜索 / SCASW 字串搜索 / SCASD 双字串搜索 }

- 功能：(1) 字节操作：(AL)-（ES：[DI]/[EDI]）

  字操作：(AX)-（ES：[DI]/[EDI]）

  双字操作：(EAX)-(ES：[DI]/[EDI])

  (2) 修改串指针使之指向下一元素.修改方式为：

  i. 当DF＝0时，(DI)/(EDI)增量1(字节操作)或2(字操作)或4(双字操作)。

  ii. 当DF＝1时，(DI)/(EDI)减量1(字节操作)或2(字操作)或4(双字操作)。

  (3) 修改循环变量(CX)/(ECX) -1→ CX)/ECX

（3）串搜索指令 SCAS

说明:

(1) 待搜索串一定要是目的串,放在当前附加数据段中,指针为DI/EDI;

(2) 该指令可带的重复指令为:

i. REPZ/REPE 若(CX/ECX)≠0且相等时接着搜索.

ii. REPNZ/REPNE 若(CX/ECX)≠0且不相等时接着搜索.

例3

（3）串搜索指令 SCAS

例3 从键盘输入一串字符至ASS区，试用串搜索指令在该串中搜索子串'AM'出现的次数→BX.

```
.386
DATA   SEGMENT  USE16
ASS    DB       80
       DB   0
       DB   79 DUP(0)
DATA   ENDS
STACK  SEGMENT  USE16  STACK
   DB 200 DUP(0)
STACK  ENDS
CODE   SEGMENT  USE16
   ASSUME   DS：DATA，ES：DATA，SS：STACK, CS：CODE
START：  MOV  AX，DATA
  MOV  DS，AX
  MOV  ES，AX
```

当前数据段和当前附加数据段重合

（3）串搜索指令 SCAS


```
        LEA   DX, ASS
        MOV   AH, 10
        INT   21H
        LEA   DI,ASS+2          ;输入串首址→DI
        MOV   CL,ASS+1          ;输入串长度→CX
        MOVZX CX,CL
        MOV   AL, ' A'
        MOV   BX, 0             ;计数器清零
P:      REPNE SCASB             ;查找字符"A"
      JE    A                 ;查到字符"A"转A处执行
      JMP   OUT1              ;全部查完转结束
A:    CMP   CX,0              ;所有判断完否
        JE   OUT1             ;全部查完转结束
        CMP BYTE PTR [DI], 'M'  ;判断"A"后的字符为"M"否
        JNE B                   ;不是转B处执行
        INC   BX                ;是"AM"计数器加1
B:      JMP   P                 ;未完转P继续判断
OUT1:  MOV   AH, 4CH            ;返回操作系统
        INT   21H
CODE   ENDS
        END   START
```

（4）取字节/字/双字串指令 LODS

- 格式：LODS OPS 或 { LODSB 从字节串中取数 / LODSW 从字串中取数 / LODSD 从双字串中取数 }
- 功能：(1) 字节操作：(DS: [SI]/[ESI]) →AL
  字操作：(DS: [SI]/[ESI]) →AX
  双字操作：DS: [SI]/[ESI]) →EAX

将SI/ESI所指的源串中的一个字节(或字、双字)存储单元中的数据取出送入AL(或AX、EAX)中。

(2) 修改SI/ESI,使之指向下一元素:

i. 当DF = 0时，(SI)/(ESI)增量1(字节操作)或2(字操作) 或4(双字操作)。

ii. 当DF = 1时，(SI)/(ESI)减量1(字节操作)或2(字操作) 或4(双字操作)。

(3) 修改循环变量(CX)/(ECX) -1→ CX)/ECX

（4）取字节/字/双字串指令 LODS

说明:

由于该指令的目的地址为一固定的寄存器，如果带上重复前缀，源串的内容将连续地送入AL(或AX、EAX)中，操作结束后，AL(或AX、EAX)中只保存了串中最后一个元素的值，这是没有多大意义的，因此，该指令一般不带重复前缀。

例4

（4）取字节/字/双字串指令 LODS

例4: 下面阅读一个使用了LODSD的程序段:

```
.386
 DATA   SEGMENT   USE16
BUF     DB   'ABCDEFGHIJKL'
COUNT EQU   $-COUNT
          ⋮
START:  MOV    AX, DATA
        MOV    DS, AX
        MOV    ES, AX
        LEA    SI, BUF
        MOV    CX, COUNT/4
        LODSD                ; 该语句执行后，（EAX）=44434241H
          ⋮
```

（5）存储字节/字/双字指令 STOS

- 格式：STOS OPD 或 { STOSB 往字节串中存数 / STOSW 往字串中存数 / STOSD 往双字串中存数 }
- 功能：

(1) 字节操作：(AL) → ES:[DI]/[EDI]

字操作：(AX) → ES:[DI]/[EDI]

双字操作：(EAX) → ES:[DI]/[EDI]

即将AL(或AX、EAX)中的数据送入DI/EDI所指的目的串中的字节(或字、双字)存储单元中。修改指针DI/EDI，使之指向串中的下一个元素。

(2) 修改串指针,使之指向下一元素,修改方式为:

i. 当DF = 0时，(SI)/(ESI)增量1(字节操作)或2(字操作) 或4(双字操作)。

ii. 当DF = 1时，(SI)/(ESI)减量1(字节操作)或2(字操作) 或4(双字操作)。

说明:该指令执行后，并不影响标志位，因而它一般只带REP重复前缀，用来将一片连续的存储字节(或字)单元置相同的值。

串操作指令总结

(1) 源串指针: DS: SI/ESI 即源串在当前数据段

(2) 目标串的指针: ES: DI/EDI 即目的串在当前附加数据段

(3) 重复计数器: CX/ECX

(4) 中间寄存器: AX/EL/EAX

(5) 传送/比较方向: DF=0, SI/ESI, DI/EDI自动增量(加1/加2/加4).

DF=0, SI/ESI, DI/EDI自动减量(减1/减2/减4).

(6) 指令格式: 带操作数: ×××S

不带操作数: ×××SD (双字操作)

×××SW (字操作)

×××SB (字节操作)

串操作指令总结

(7) 重复前缀:

REP　　重复执行,直到(CS/ECS)=0; (主要是MOVS指令使用)

REPE/REPZ　ZF=1 时重复执行,直到(CS/ECS)=0;

REPNE/REPNZ　ZF≠1时重复执行,直到(CS/ECS)=0;

主要时CMPS,SCAS指令使用

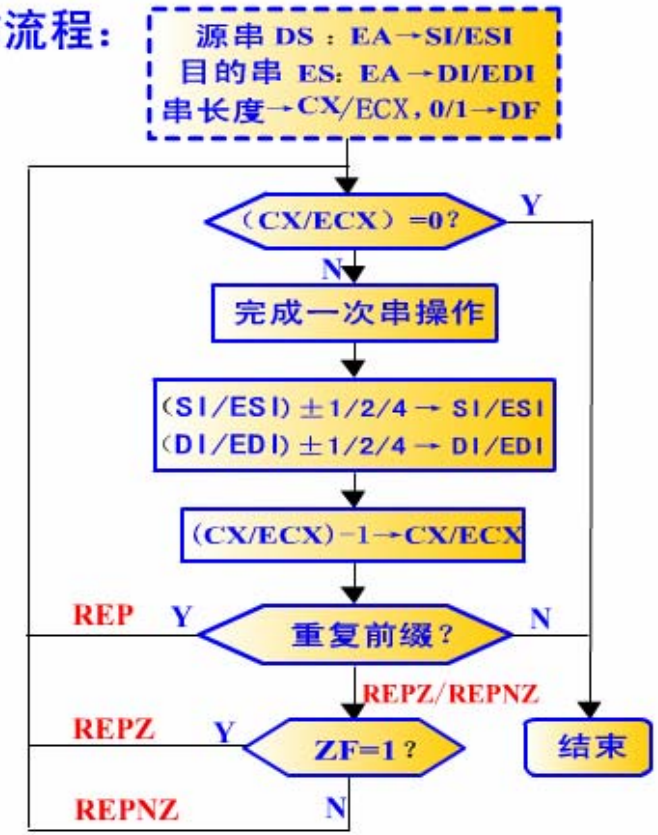
说明:

循环次数(CX)是否为0是在操作之前检测的,因此,当(CX)为0时不会引起串操作.操作终止后,SI,DI均指向下一待操作的EA,方向由DF确定.

工作流程

串操作指令总结

串操作指令工作流程：



5.2 宏功能程序设计

(1) 宏定义

(2) 宏调用与宏扩展

(3) 宏指令中的参数

(4) 宏库的使用

(5) 宏指令与子程序的比较

(1) 宏定义

宏指令在使用之前要先定义.

- 定义格式: 宏指令名　　MACRO　形式参数　[，形式参数]
  宏体
  ENDM

例如: 某一程序中要多次使用DOS9号功能调用:

```
        ⋮
LEA   DX, BUF1
MOV   AH, 9
INT   21H
        ⋮
LAE   DX, IN_BUF
MOV   AH, 9
INT   21H
```

(1) 宏定义

他们之间的差别仅在于输出缓冲区的首址不一样. 如果讲该首址定义成形参, 就可将其写成宏定义:

```
WRITE    MACRO   A
         LEA   DX, A
         MOV   AH, 9
         INT   21H
         ENDM
```

说明:

(1) 宏名字可以与其他变量, 标号, 保留字同名, 汇编程序在处理时宏名字优先级最高. 利用这一特点, 程序员可以设计新的指令系统.

(1) 宏定义

(2) 形参可有可无个数不限,但总字符长度不超过132个,个参数之间用逗号隔开;

(3) ENDM和MACRO必须成对出现;

(4) 宏指令必须先定义后调用,因为它是在汇编期间处理的.

(2) 宏调用与宏扩展

- 调用格式：宏指令名 [实在参数 [，实在参数]……]

对于例1中所作的宏定义，所应做的宏调用为：

```
WRITE   BUF1        ;宏调用
   ⋮
WRITE   IN_BUF      ;宏调用
```

汇编程序的处理方式：

(1) 第一次扫描时，先扫描宏定义，将宏名字、形参、宏体均填入宏定义表中再遇到宏调用时，则嵌入宏体，用实参按位置顺序替换形参，这一过程称为宏扩展。

(2) 第二次扫描时，再将其转换成目标代码.

(2) 宏调用与宏扩展

在汇编列表文件(.lst)中,宏拓展后的宏体语句均在前面冠以"+"以示与其他语言的区别,即拓展后的形式为:

```
+     LEA   DX, BUF1
+     MOV   AH, 9
+     INT   21H
          ⋮
+     LEA   DX, IN_FUT
+     MOV   AH, 9
+    INT   21H
```

(2) 宏调用与宏扩展

说明:

(1) 宏指令名要与原宏定义的名字一致;

(2) 实参与形参应按位置关系一一对应:

i. 实参个数多形参, 多余实参被忽略;

ii. 实参个数小于形参, 缺少的实参被处理成空白(没有字符).

还可以讲DOS9号和10号调用写成一个宏定义:

```
IO      MACRO     A, B
        LEA DX,   A       ;形参A代表输入/输出缓冲区首址
        MOV AH,   B       ;形参B代表DOS调用号
        INT 21H
        ENDM
```

(3) 宏指令中的参数

一、带几个符的参数

在宏调用中，有时实参是一串带间隔符的字符串，为了不致混淆，应该尖括号将它们括起来，说明为一个参数.

例如，每一个程序都需要定义堆栈段,只是堆栈的大小和初值不一样 .可以讲定义堆栈的语句写成宏定义:

```
DEF_STACK    MACRO   A
STACK        SEGMENT  USE16  STACK
             DB   A
STACK        ENDS

             ENDM
```

其中实参为一个重复子句，它中间带有空格，因此要用尖括号括起来

如果需要建立200个字节、初值均为1的堆栈段.宏调用应为:

```
DEF_STACK <200  DUP(1)>
```

(3) 宏指令中的参数

二、宏体中的变量与标号

例1. 以下程序段的功能为:求从AA开始连续BB个奇数(或偶数)的和送AX.

```
AA    =1
BB    =50
      MOV     BX, AA
      MOV     CX, BB
      MOV     AX, 0
NEXT: ADD     AX, BX
      ADD     BX, 2
      LOOP    NEXT
```

其宏定义和宏调用如下:

```
SUM     MACRO   AA,  BB
        MOV     BX,  AA
        MOV     CX,  BB
        MOV     AX,  0
NEXT:   ADD     AX,  BX
        ADD     BX, 2
        LOOP    NEXT
        ENDM
          ⋮
        SUM     1,  50
          ⋮
        SUM     10,  20
```

查看说明　　查看宏定义

(3) 宏指令中的参数

```
+       MOV     BX, 1
+       MOV     CX, 50
+       MOV     AX, 0
+  NEXT: ADD    AX, BX      ;标号NEXT第一次出现
+       ADD     BX, 2
+       LOOP    NEXT
            ⋮
+       MOV     BX, 10
+       MOV     CA, 20
+       MOV     AX, 0
+  NEXT: ADD    AX, BX      ;标号NEXT第二次出现,重复定义
+       ADD     BX, 2
+       LOOP    NEXT
```

(3) 宏指令中的参数

处理方式:

1. 将标号定义成形参,每次调用均用实参代替.但会造成参数过多,给编程带来麻烦;

2.使用伪指令LOCAL,让机器自动生成不同标号.

- 格式: LOCAL 形式参数[,形式参数.....]
- 功能: 宏拓展时,汇编程序自动为其后形参生成特殊顺序号,范围为??0000 - ??FFFFH,并用之取代形参,避免重复定义.

注意:该语句只能做宏体中的第一个语句.

(3) 宏指令中的参数

修改后的宏定义为:

```
SUM     MACRO   AA,  BB
        LOCAL   NEXT
        MOV     BX,  AA
        MOV     CX, BB
        MOV     AX,  0
NEXT:   ADD     AX,  BX
        ADD     BX,  2
        LOOP    NEXT
        ENDM
```

相应的宏拓展为:

```
+       MOV     BX,  1
+       MOV     CX,  50
+       MOV     AX,  0
+??0000: ADD    AX, BX; 标号NEXT转换??0000
+       ADD     BX,  2
+       LOOP??0000
          ⋮
+       MOV     BX,  10
+       MOV     CX,  20
+       MOV     AX,  0
+??0001: ADD    AX,  BX; 标号NEXT转换??0001
+       ADD     BX,  2
+       LOOP??0001
```

(4) 宏库的使用

对于经常使用的宏定义，用户可将它们集中在一起，建成宏库供自己或别人随时调用。由于宏库为文本文件，可用一般编辑程序建立或修改，文件名也可由用户任意指定。

例如：我们利用编辑程序，建立了一个宏库MACRO. LIB:

当程序中需要调用时，应首先将宏库加入自己的源文件中，然后按宏库中各宏定义的规定调用即可。

将宏库加入源文件一起进行汇编可用伪指令INCLUDE实现。

- 语句格式：INCLUDE　文本文件名
- 功　　能：将指定的文本文件从本行起加入汇编，直到该文本文件的最后一行汇编完后，再继续汇编INCLUDE后面的语句。

查看宏库　　宏库调用

(4) 宏库的使用

宏库内容:

```
READ    MACRO   A
        LEA     DX,  A
        MOV     AH,  10
        INT     21H
        ENDM
WRITE   MACRO   A
        LEA     DX,  A
        MOV     AH,  9
        INT     21H
        ENDM
OUT1    MACRO   A
        MOV     DL,  A
        MOV     AH,  2
        INT     21H
        ENDM
CRLF    MACRO
        OUT1    0AH
        OUT1    0DH
        ENDM
STACK0  MACRO   A
STACK   SEGMENT USE16   STACK
        DB      A
STACK   ENDS
        ENDM
```

(4) 宏库的使用

```
        INCLUDE MACRO.LIB;将指定的宏库加入本程序一起汇编
.386
DATA    SEGMENTUSE16
BUF     DB    80, 0, 80    DUP(0)
DATA    ENDS
STACK0 < 200  DUP(0) >
CODE    SEGMENT USE16
        ASSUME  DS: DATA, SS:STACK, CS: CODE
START:  MOV     AX, DATA
        MOV     DS, AX
        READ    BUF         ;输入一字符串→BUF缓冲区
        CRLF                ;输出回车换行
        LEA     SI, BUF+2
        MOV     CL, BUF+1
        MOV     CH, 0
        CLD
```

(4) 宏库的使用

```
Y1:     LODSB               ;从输入串取一字符→AL
        CMP     AL, 'a'
        JB      Y2
        CMP     AL, 'z'
        JA      Y2          ;该字符不为小写字母,则转Y2
        SUB     AL, 20H     ;将小写字母转换为大写字母的ASCII码
Y2:     OUT1    AL
        LOOP    Y1
        CRLF                ;输出回车换行
        MOV     AH, 4CH
        INT     21H
CODE    ENDS
        END     START
```

(5) 宏指令与子程序的比较

不同点:

(1) 处理时间不同;

(2) 处理方式不同;

(3) 目标程序的长度不同;

(4) 执行速度不同;

(5) 参数传递方式不同;

5.3 模块化程序设计

什么是模块？

- 一个以END语句为结束的源程序称为一个模块。一个模块是一个独立的汇编源文件，一个模块汇编后生成一个目标文件（*.obj），或称目标模块。
- 模块化程序是由一个以“END 标号”结束的主模块，和几个以“END”结束的子模块组成。

5.3 模块化程序设计

模块化程序的作用：

- 用于大型程序系统的设计。通过模块的划分，将复杂的问题合理分配到各子模块中去解决，便于多人合作，编程的质量和效率都会提高。
- 此外，通过模块的划分，可以针对不同模块的特性，采用不同的语言编程，有利于发挥各计算机语言的特性。

5.3 模块化程序设计

模块化程序设计主要解决的问题：

- 如何正确地划分模块
- 如何把模块装配在一起

5.3.1 段的组合方式

段定义语句：

- 段名 SEGMENT [定位方式] [组合方式] ['类别']

  …

  段名 ENDS

5.3.1 段的组合方式

1. 定位方式
2. ‘类别’
3. 组合方式

1. 定位方式

定位方式是用来指定各段的起始物理地址的确定方式。有四种不同的选择：

- PARA: 段从能被16整除的地址开始存放。
- WORD: 段从一个偶数地址开始存放。
- BYTE: 对段存放的首地址不作要求。
- PAGE: 段从能被256整除的地址开始存放。

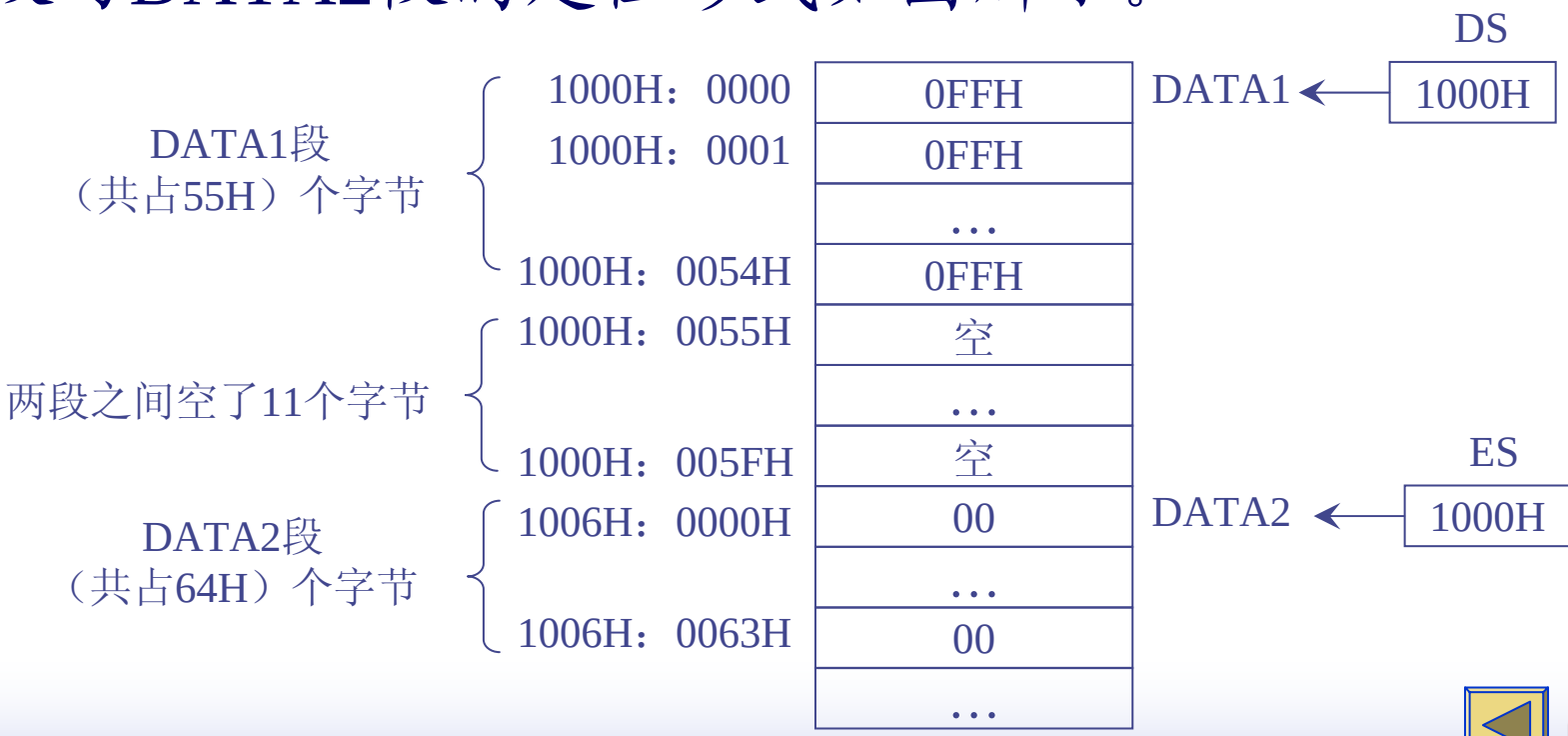
例1

已知程序中的DATAl段和DATA2段的定义如下，如果这两个段分别为当前数据段和当前附加数据段. 试分析下述连接程序对它们边界的处理方法。

```
DATAl   SEGMENT
A   DB  55H  DUP(0FFH)
DATAl   ENDS
DATA2   SEGMENT
        DB  64H  DUP(0)
DATA2   ENDS
```

例1

由于这两个段均未明确指定定位方式，说明它们的定位方式均为PARA。经连接程序连接后，DATAl段与DATA2段的定位形式如图所示。



5.3.1 段的组合方式

1. 定位方式
2. ‘类别’
3. 组合方式

2.‘类别’

‘类别’是用单引号括起来的字符串，该字符串可以是任何合法的名称。连接程序在进行连接处理时，将‘类别’相同的段（它们不一定同名），按出现的先后次序连续存放（但仍是不同名的段），且每段都有自己的首址。

例2

有5个段名不同的段，分属三个不同的‘类别’，连接时。五个段按以下顺序提供给连接程序：

A SEGMENT ‘DATA’
B SEGMENT ‘CODE’
C SEGMENT ‘TO’
D SEGMENT ‘DATA’
E SEGMENT ‘TO’

例2

连接后，在生成的.EXE文件中，各段的相对位置成为

```
A   SEGMENT   ‘DATA’
D   SEGMENT   ‘DATA’
B   SEGMENT   ‘CODE’
C   SEGMENT   ‘TO’
E   SEGMENT   ‘TO’
```

5.3.1 段的组合方式

1. 定位方式
2. ‘类别’
3. 组合方式

3. 组合方式

组合方式向连接程序提供了本段同其它段的组合关系，主要有：

- (1) NONE（不选择）：本段与其它段不发生任何逻辑上的联系，自己有自己的段首址。（隐含的）
- (2) PUBLIC：表示应将本段与其它模块中的同名、同‘类别’段按各模块连接的顺序相邻地连接在一起，组成一个物理段， 大小不超过64K。

  合并的好处：以数据段为例，当组合成一个段后，在子模块中就不用再对DS送首址，方便而又有效。

3. 组合方式

- (3) STACK：仅对堆栈段，功能同PUBLIC。
- (4) COMMON：表示本段与同名、同‘类别’的其它段应具有相同段首址，即相覆盖。长度取决于最长的COMMON段。
- (5) “AT表达式”：表示该段应放在表达式所指定的绝对地址上。
- (6) “MEMORY”：表示该段应放所有段之上（最高地址上）。

5.3.2 通讯方式

1. 局部符号：

仅在定义自己的模块中被访问的符号为局部符号。

2. 公共符号：

不仅被定义自己的模块访问，还要供其它模块访问的符号为公共符号。要用PUBLIC伪指令说明。其格式为：

PUBLIC 符号[，符号，……]

3. 外部符号：

只在模块内访问而不在该模块内定义的符号为外部符号。要用EXTRN伪指令说明。其格式为：

EXTRN 符号：类型[，符号：类型，……]

5.3.3 连接程序（LINK）的功能

1. 将制定的若干个目标模块（.OBJ）和子程序库（.LIB）中的子程序模块连接在一起，生成可执行的文件（.EXE）
2. 产生一个地址分配文件（.MAP）

5.3.4 地址分配文件举例

地址分配文件（.MAP）用以描述执行文件中各段的浮动起始地址、结束地址、所占用空间大小、段名及‘类别，还可列出各模块中所定义的公共符号及其偏移地址，它主要用于程序调试及资料归档。下面列出WANl和WAN2两模块连接后所产生的“.MAP”文件。

例3

```
        NAME    WAN1
        EXTRN   WAN2：NEAR
        PUBLIC  DIV0，COUNT
.386
DATA1   SEGMENT         USE16 PARA PUBLIC ‘D1’
BEG     EQU     1
COUNT   DW      50
DIV0    DW      0
DATA1   ENDS
STACK   SEGMENT         USE16 PARA STACK ‘STACK’
        DB      200 DUP(0)
STACK   ENDS
CODE    SEGMENT         USE16 PARA PUBLIC ‘CODE’
        ASSUME DS: DATA1, CS: CODE, SS: STACK
START:  MOV     AX, DATA1
```

例3

```
        MOV     AX，DATA1
        MOV     DS，AX
        MOV     BX， BEG
        MOV     CX，COUNT
        CALL    WAN2
        MOV     AH，4CH
        INT     21H
CODE    ENDS
        END     START
```

例3

```
        NAME    WAN2
        EXTRN   DIV0：WORD，COUNT：WORD
        PUBLIC  WAN2
.386
DATA2   SEGMENT         USE16 PARA PUBLIC ‘D2’
SUM     DW      -
DATA2   ENDS
STACK   SEGMENT         USE16 PARA STACK ‘STACK’
        DB      200 DUP(0)
STACK   ENDS
CODE    SEGMENT         USE16 PARA PUBLIC  ‘CODE’
        ASSUME  ES:DATA2, CS: CODE, SS: STACK
WAN2    PROC
        MOV     AX，DATA2
        MOV     ES，AX
```

例3

```
        MOV     AX，0
NEXT：  ADD     AX，BX
        ADD     BX，2
        LOOP    NEXT
        MOV     SUM，AX
        MOV     DX，0
        DIV     DS：COUNT
        MOV     DS：DIV0，AX
        RET
WAN2    ENDP
CODE    ENDS
        END
```

例3

Start	Stop	Length	Name	Class
00000H	00003H	00004H	DATA1	D1
00010H	0019FH	00190H	STACK1	STACK
001A0H	001DDH	0003EH	CODE	CODE
001E0H	001E1H	00002H	DATA2	D2

Address	Publics by Name
0000:0000	COUNT
0000:0002	DIV0
001A:0020	WAN2

Address	Publics by Value
0000:0000	COUNT
0000:0002	DIV0
001A:0020	WAN2

Program entry point at 001A:0000

例3

WAN1和WAN2两模块连接的特点如下。

1. 由于两代码段名字、‘类别’相同且组合方式均为PUBLIC，因此，这两个段按出现的顺序合并为一个段，也正因为如此，子程序WAN2可定义为近过程。合并后的段所占字节总数应为原来各段大小之和。
2. 两数据段不同名，不同‘类别’，各自独立，不发生任何关系，每个段都有自己的起始地址。在主模块中为当前数据段，在子模块中为附加数据段。

例3

3. 两堆栈段由于同名、同‘类别’且组合方式为STACK，因此，这两个堆栈段合并成一个段，合并后的段所占字节总数应为原来各段大小之和。
4. 程序的启动地址为001A:0000，即为WAN1模块中END后面所带的符号地址START，它是CODE段中第一条语句的标号，因此，EA等于0。
5. 符号表中列出了各模块中所定义的公共符号及经修改过的值。
6. 以上所提供的信息。将给我们调试多模块程序带来很大的方便。

5.4 源程序综合举例

5.4.1 模块程序设计中的注意事项

- 1. 模块的划分的规则
- 2. 程序文件的命名
- 3. 标号的定义
- 4. 变量和缓冲区的定义
- 5. 模块注释

5.4.2 模块程序设计举例

1. 模块划分的规则

(1) 如果一个程序段被很多模块所公用，则它应是一个独立的模块。

(2) 如果若干个程序段处理的数据是公用的，则这些程序段应放在一个模块中。

(3) 若两个程序段的利用率差别很大，则应分属于两个模块。

(4) 一个模块既不能过大，也不能过小。过大则模块的通用性较差，过小则会造成时间和空间上的浪费。

1. 模块划分的规则

(5) 力求使模块具有通用性，通用性越强的模块利用率越高。

(6) 各模块间应在功能上、逻辑上相互独立，尽量截然分开，特别应避免用转移语句在模块间转来转去。

(7) 各模块间的接口应该简单，要尽量减少公共符号的个数，尽量不共用数据存储单元，在结构或编排上有联系的数据应放在一个模块中，以免互相影响，造成查错困难。

(8) 每个模块的结构应尽量设计成单入口、单出口的形式。这样的程序便于调试、阅读和理解且可靠性高。

5.4 源程序综合举例

5.4.1 模块程序设计中的注意事项

- 1. 模块划分的规则
- 2. 程序文件的命名
- 3. 标号的定义
- 4. 变量和缓冲区的定义
- 5. 模块注释

5.4.2 模块程序设计举例

2. 程序文件的命名

(1) 标识出程序设计者。即在文件名中用一符号作程序员代号。这样，当制作的模块很多时，通过文件名就应能分辨出各个模块的设计者；每个程序员又可在其代码之后加一数字，标识本人编写的不同模块。这样，遇有问题就可快速找到出处，进行追踪。

(2) 简单标识模块功能。在文件名中标识出模块功能可方便调用与连接。

(3) 标识出模块之间的连接关系：即指一个模块在待连接的一串模块中的位置，可在文件名最后以数字表示顺序。

例如：

程序员编号为“C”，本模块的编号为3，功能为“QUEUE”，无须指定连接关系，则其文件名应为：

CQUEUE3.ASM

5.4 源程序综合举例

5.4.1 模块程序设计中的注意事项

- 1. 模块的划分
- 2. 程序文件的命名
- 3. 标号的定义
- 4. 变量和缓冲区的定义
- 5. 模块注释

5.4.2 模块程序设计举例

3. 标号的定义

标号名不仅要统一，最好包含它的功能和所有必备的信息。标号所代表的信息有：

(1) 模块出处：即为程序员的代号和附加编号。

(2) 功能名称：明确的功能定义可以提高模块的再利用价值。功能名称部分长度以三个字母为宜，太长则输入费时。

- 如：DSP 表示屏幕显示功能
- PRN 表示打印功能
- KIN 表示键盘输入功能

(3) 分支代号：程序中的分支代号最理想的是由小而大依序安排，为适应编写程序中的变化，可预留空号以便扩充。

5.4 源程序综合举例

5.4.1 模块程序设计中的注意事项

- 1. 模块的划分
- 2. 程序文件的命名
- 3. 标号的定义
- 4. 变量和缓冲区的定义
- 5. 模块注释

5.4.2 模块程序设计举例

4. 变量和缓冲区的定义

例如：

- 第一个字母用“B”为缓冲器名开头；用“V”为变量名开头；用“S”为字符串名开头。
- 第二个字母可代表定义的类型:
  - Q=QWORD　　如:BQxxxx 缓冲器为4字类型。
  - D=DWORD　　如:BDxxxx 缓冲器为双字类型。
  - W=WORD　　如:VWxxxx 变量为字类型。
  - B=BYTE　　如:VBxxxx 变量为字节类型。
  - S=STRING　　如:Sxxxxx 为字符串（字节类型）。
  - F=FLAG　　如:VBFxxx 该变量为字节型类，用作标志。

5.4 源程序综合举例

5.4.1 模块程序设计中的注意事项

- 1. 模块的划分
- 2. 程序文件的命名
- 3. 标号的定义
- 4. 变量和缓冲区的定义
- 5. 模块注释

5.4.2 模块程序设计举例

5. 模块注释

为方便子模块的调用，在子模块的前面必须有详细的说明。说明的内容包括：子模块功能、调用此子模块的入口参数和出口参数、子模块中所使用的主要变量等。

5.4 源程序综合举例

5.4.1 模块程序设计中的注意事项

- 1. 模块的划分
- 2. 程序文件的命名
- 3. 标号的定义
- 4. 变量和缓冲区的定义
- 5. 模块注释

5.4.2 模块程序设计举例

例1

编写一子程序模块DUMP，使之能将32位寄存器和16位寄存器的内容以有符号十进制数的形式显示出来，显示按“PUSHAD”进栈和“POPAD”出栈的顺序，格式为：

(EDI)= × × × × × × × ×　　(DI)= × × × ×
(ESI)= × × × × × × × ×　　(SI)= × × × ×
(EBP)= × × × × × × × ×　　(BP)= × × × ×
(ESP)= × × × × × × × ×　　(SP)= × × × ×
(EBX)= × × × × × × × ×　　(BX)= × × × ×
(EDX)= × × × × × × × ×　　(DX)= × × × ×
(ECX)= × × × × × × × ×　　(CX)= × × × ×
(EAX)= × × × × × × × ×　　(AX)= × × × ×

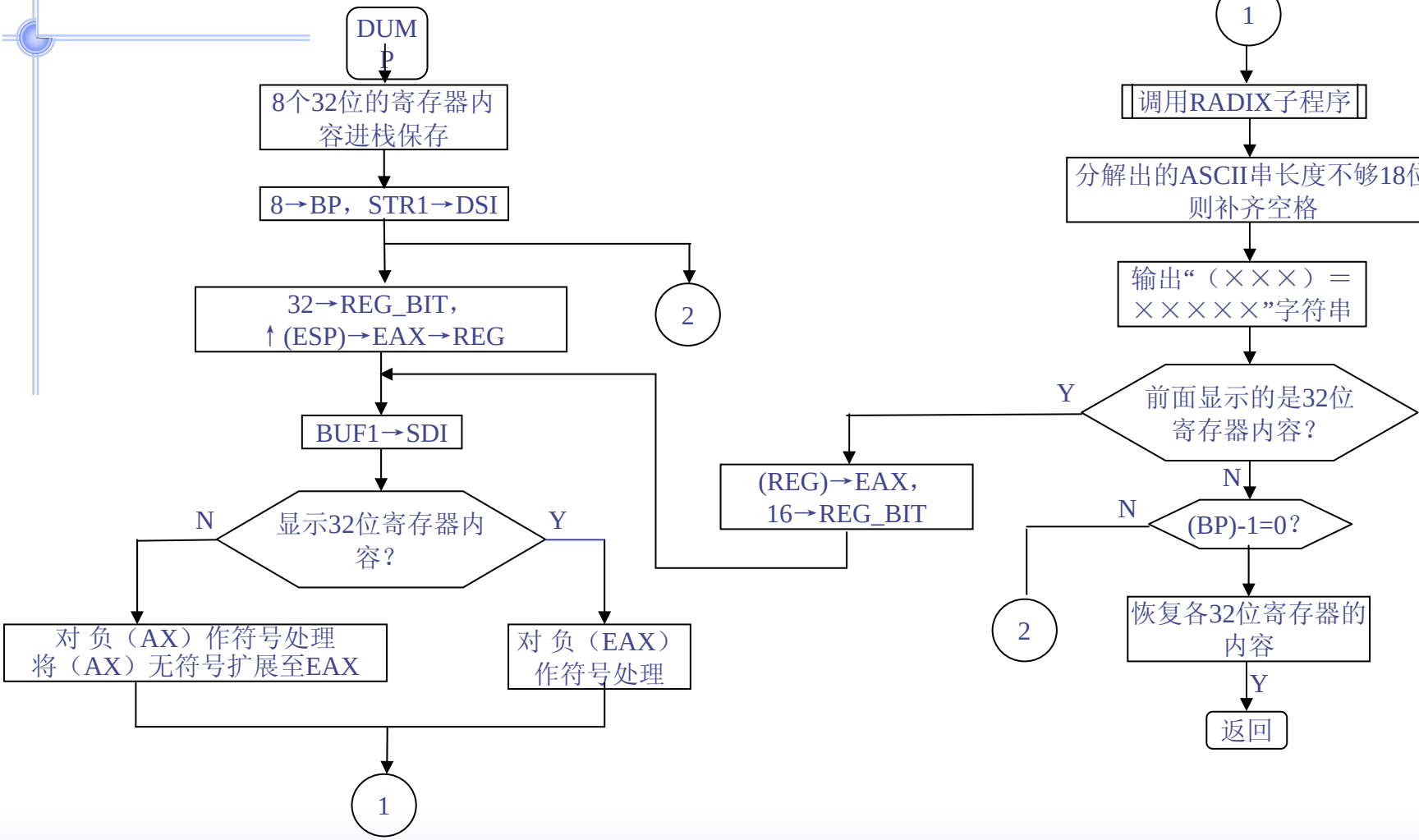
并要求主模块在调用该子模块之前与之后，全部寄存器的内容均不改变。

例1

本例采用的算法是：

- 为了保证在调用该子模块之前与之后全部寄存器的内容均不改变，必须一进入本子模块就将全部32位寄存器的内容用“PUSHAD”进栈保存，在返回主模块之前再恢复各寄存器的内容。
- 整个显示过程需要通过8次循环，一次循环在一行中输出一个32位的寄存器和一个16位寄存器的内容。其方法是：
  - 在每次循环中顺序从堆栈中弹出一个32位寄存器的内容→EAX，然后根据二进制转换成十进制ASCII码输出的算法将(EAX)分解成十进制ASCII码→BUF1输出缓冲区，显示该32位寄存器的内容；
  - 再将(AX)分解成十进制ASCII码→BUF1输出缓冲区，显示该16位寄存器的内容；为了保证按所规定的格式输出，可规定每个寄存器所要显示的字符串长度均为18，不足部分补空格，每行两个显示完后则回车换行，开始下一循环，再在下一行接着显示，这样就能达到每行显示2个并自动对齐的要求。
- 该例的流程图如下所示。

例1


DUMP
8个32位的寄存器内容进栈保存
8→BP，STR1→DSI
32→REG_BIT，
↑(ESP)→EAX→REG
2
BUF1→SDI
显示32位寄存器内容？
N
Y
对负（AX）作符号处理
将（AX）无符号扩展至EAX
对负（EAX）作符号处理
1
1
调用RADIX子程序
分解出的ASCII串长度不够18位则补齐空格
输出“（×××）＝×××××”字符串
前面显示的是32位寄存器内容？
Y
N
(REG)→EAX，
16→REG_BIT
N
(BP)-1=0？
2
恢复各32位寄存器的内容
Y
返回

例1

根据流程图所编写的源程序为：

```
NAME    DUMP
;  子模块DUMP
;  功能:将全部16 /32位数据寄存器的内容按有符号十进制数的形式分8行输出
;  入口参数:  为各寄存器的内容
;  出口参数:  仍为各寄存器的入口值不变
;  所使用的寄存器和主要变量:
;  EAX——保存待分解的数
;  DX ——按除法指令和系统功能调用的规定使用
;  DI ——取寄存器名指针，初值为STR1
;  SI ——送分解出的十进制ASCII码指针，初值为BUF1
;  BX ——RADIX子程序中的进制参数
;  CX ——RADIX子程序中分解出的数字位数计数器
;  STR1 ——寄存器名字符串存储区首址
;  BUF1 ——输出缓冲区首址
;  REG  ——临时保存待输出寄存器的内容
;  REG_BIT——显示16/32位寄存器标志

                PUBLIC   DUMP
.386
WRITE           MACRO    A
                LEA      DX,A
```

例1

```
                MOV     AH, 9
                INT     21H
                ENDM
DATA            SEGMENT             USE16 PARA PUBLIC ‘DATA’
STR1            DB      ‘(EDI)=$’,  ‘ (DI)=$’
                DB      ‘(ESI)=$’,  ‘ (SI)=$’
                DB      ‘(EBP)=$’,  ‘ (BP)=$’
                DB      ‘(ESP)=$’,  ‘ (SP)=$’
                DB      ‘(EBX)=$’,  ‘ (BX)=$’
                DB      ‘(EDX)=$’,  ‘ (DX)=$’
                DB      ‘(ECX)=$’,  ‘ (CX)=$’
                DB      ‘(EAX)=$’,  ' (AX)=$'
BUF1            DB      20 DUP(0)
CRLF            DB      0AH,0DH,'$'
REG             DD      0
REG_BIT         DB      0                   ;  显示16/32位寄存器标志
DATA            ENDS
STACK           SEGMENT             USE16 PARA STACK 'STACK'
                DB      200 DUP(0)
STACK           ENDS
CODE            SEGMENT USE16 PARA PUBLIC 'CODE'
                ASSUME  DS:DATA,SS:STACK,CS:CODE
```

例1

```
DUMP        PROC  FAR
            PUSHAD          ; 保护所有32位寄存器的内容
            MOV     AX,DATA
            MOV     DS,AX
            MOV     BP,8
            MOV     DI,OFFSET STR1
NEW_LINE:   MOV     REG_BIT,32      ; 首先置显示32位寄存器标志
            POP     EAX
            MOV     REG,EAX
ENCORE:     LEA   SI,BUF1
            CMP     REG_BIT,32      ; 是否显示32位寄存器
            JNE     BIT_16          ; 否，转BIT-16处显示16位寄存器的内容
            OR      EAX,EAX
            JNS     PLUS
            NEG     EAX
            MOV     BYTE PTR [SI],'-‘
            INC     SI
            JMP     PLUS
BIT_16:     OR      AX,AX
            JNS     B1
            NEG     AX
            MOV     BYTE PTR [SI],'-'
```

例1

```
            INC     SI
B1:         MOVZX   EAX, AX        ; 将16位无符号数扩展为32位后→EAX
PLUS:       MOV     EBX, 10
            CALL    RADIX          ; 将EAX中的无符号二进制数转换为十进制
ASCII码→BUF1
SPACE:      CMP     SI, 18+OFFSET BUF1
            JE      PP
            MOV     BYTE PTR [SI], ‘ ’ ;  不足18位补空格
            INC     SI
            JMP     SPACE
PP:         MOV     BYTE PTR [SI], ‘$’; 在该串末尾补’$’
            WRITE   [DI]           ; 输出“（×××）=”，长度为7个字节（包
括’$’）
            ADD     DI, 7          ; 指针下移7个字节
            WRITE   BUF1           ; 显示RADIX子程序分解出的字符串
            CMP     REG_BIT,16     ; 是显示的16位寄存器吗?
            JE      LINE_END       ; 是，一行结束
            MOV     EAX, REG       ; 否则，取出开始保存的内容→EAX
            MOV     REG_BIT,16     ; 显示16位寄存器的内容的标志→REG_BIT
            JMP     ENCORE         ; 转显示16位寄存器的内容
LINE_END:   WRITE   CRLF           ; 一行结束，输出回车换行
            DEC     BP             ; 修改循环变量
            JZ      ALL_END        ; 全部显示完，转结束处理
            JMP     NEW_LINE       ; 否则转显示新的一行
```

例1

```
ALL_END:  SUB     ESP, 32        ; 让栈指针回到执行完“PUSHAD”指令时的位置
          POPAD                  ; 恢复各寄存器的初值
          RET                    ; 返回
DUMP      ENDP
; RADIX子程序的功能为：将EAX中的无符号二进制数转换为十进制ASCII码→SI所指的缓冲区中
RADIX     PROC
          PUSH    CX
          PUSH    EDX
          MOV     CX, 0
LOP1:     MOV     EDX, 0
          DIV     EBX
          PUSH    DX
          INC     CX
          OR      EAX, EAX
          JNE     LOP1
LOP2:     POP     AX
          ADD     AL, 30H
          MOV     [SI], AL
          INC     SI
          LOOP    LOP2
          POP   EDX
          POP   CX
```

例1

```
                RET
RADIX           ENDP
CODE            ENDS
                END
```

该模块编写完成后，可供需要的程序调用。下面为调用以上子模块的主程序:

```
                EXTRN    DUMP:FAR
.386
DATA            SEGMENT USE16 PARA PUBLIC 'DATA'
ARR             DW 1111,2222,3333,4444,5555,6666,-7777
DATA            ENDS
STACK           SEGMENT USE16 STACK
                DB 200 DUP(0)
STACK           ENDS
CODE             SEGMENT USE16
                ASSUME DS:DATA,CS:CODE,SS:STACK
START:          MOV      AX,DATA
                MOV      DS,AX
                MOVSX    EAX,ARR
                MOVSX    EBX,ARR+2
                MOVSX    ECX,ARR+4
                MOVSX    EDX,ARR+6
```

例1

```
                MOVSX  ESI,ARR+8
                MOVSX  EDI,ARR+10
                MOVSX  EBP,ARR+12
                CALL   DUMP
                MOV    AH,4CH
                INT    21H
CODE            ENDS
                END        START
```

将它们分别汇编、连接后，运行的结果为:

(EDI)=6666　(DI)=6666
(ESI)=5555　(SI)=5555
(EBP)=-7777　(SI)=-7777
(ESP)=196　(SP)=196
(EBX)=2222　(BX)=2222
(EDX)=4444　(DX)=4444
(ECX)=3333　(CX)=3333
(EAX)=1111　(AX)=1111

说明：按照PUSHAD指令的功能，此处显示的（SP）/（ESP）的值为执行PUSHAD指令之前的位置。

例2

从键盘输入一串以逗号为分隔符的十进制有符号数，然后按从小到大的顺序显示出来(仍以逗号为分隔符)，若输入的数中包括非法数，则停止输入并给出错误提示后返回DOS。

例如：输入的数组为： -180，90，123，-327，10
排序输出结果为：-327，-180，10，90，123

或：输入的数组为： -180，90，9A IS ILLEGAL DIGIT!

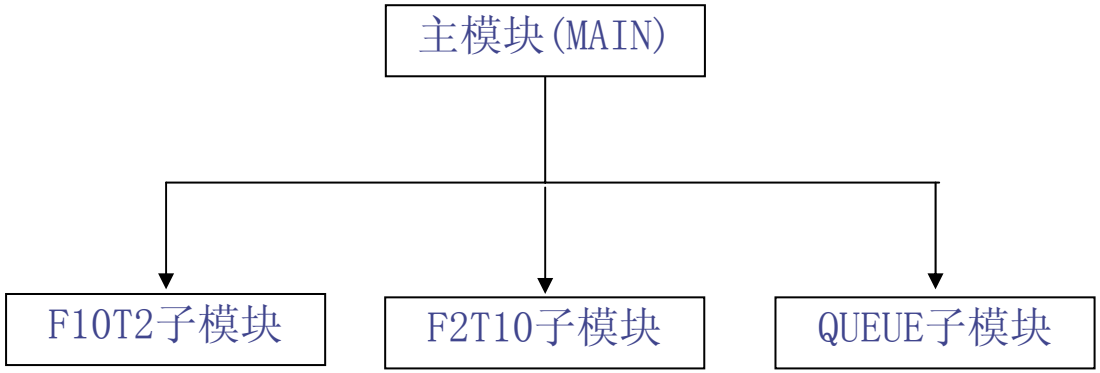
因“9A”为非法数，则结束程序的执行返回DOS，无输出结果。

例2

实现以上功能的方法是：

- 采用1号DOS功能调用输入一个字符，如果该字符不是逗号，也不是回车符，则说明该字符为十进制数字字符，应送入BUF字节缓冲区，再输入下一个字符；
- 如果该字符是逗号，说明一个数已经输完，则应将BUF区中的这个十进制数字串转换成二进制数，送入ARR字存储区(调用F10T2子程序实现)后再输入下一个数；
- 如果该字符是回车符，说明全部数已经输完，再对ARR存储区中的全部有符号数排序(调用QUEUE子程序)，最后将排好序的数通过调用F2T10子程序全部以十进制的形式显示出来(仍以逗号为分隔符)。
- 如果采用模块设计的方法，将F2T10、F10T2、QUEUE三个子程序都写成独立模块的形式且存放在子程序库WAN.LIB中，则主模块对它们的调用关系如下图所示。

例2



例2

主模块中寄存器及主要变量的使用分配如下:

AX: 中间寄存器。

BX: 往ARR字存储区送数指针，初值为ARR。

CX: 以逗号分隔的一个十进制数字串长度计数器，初值为0。

DX: 按系统功能调用的规定使用。

DI: 往BUF字节存储区送输入的字符指针，初值为BUF。

例2

SI：调用F10T2子模块的入口参数，用作从BUF区取字符的指针。

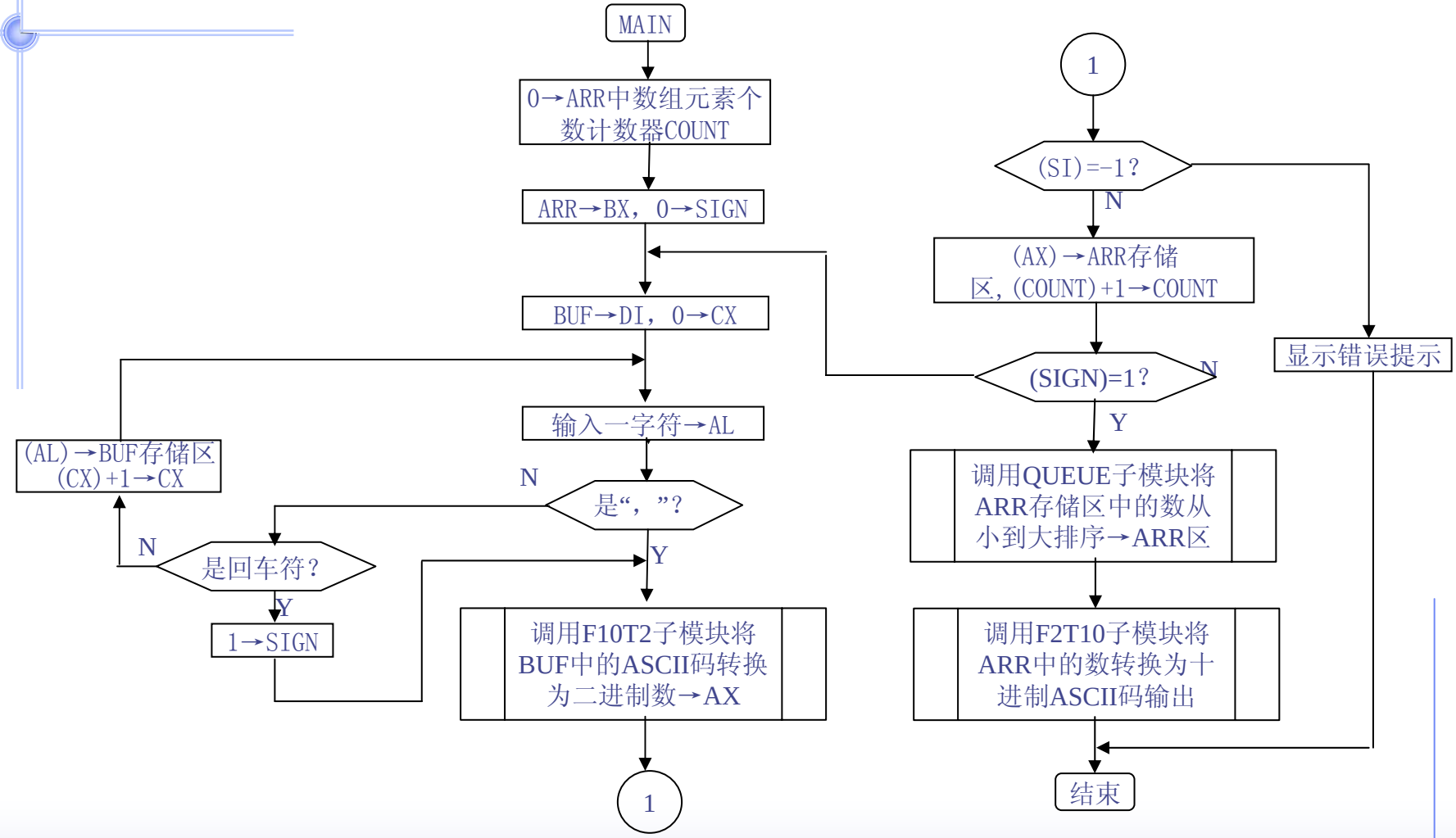
BUF：输入缓冲区首址。

ARR：经转换得到的二进制数组存储区首址。

COUNT：ARR存储区中数组元素个数计数器，初值为0。

SIGN：输入字符串处理结束标志，1：处理结束，0：处理未结束。

例2主模块程序框图


MAIN
0→ARR中数组元素个数计数器COUNT
ARR→BX，0→SIGN
BUF→DI，0→CX
输入一字符→AL
是“，”?
N
Y
是回车符?
N
Y
(AL)→BUF存储区
(CX)+1→CX
1→SIGN
调用F10T2子模块将BUF中的ASCII码转换为二进制数→AX
1
1
(SI)=-1?
N
(AX)→ARR存储区，(COUNT)+1→COUNT
(SIGN)=1?
N
Y
显示错误提示
调用QUEUE子模块将ARR存储区中的数从小到大排序→ARR区
调用F2T10子模块将ARR中的数转换为十进制ASCII码输出
结束

例2

在编写该例的主模块时，调用了宏库MACRO.LIB，该宏库中所包含的宏定义如下所示：

```
READ    MACRO   A
        LEA     DX, A
        MOV     AH, 10      ; 10号系统功能调用
        I       NT      21H
        ENDM
WRITE   MACRO   A
        LEA     DX, A
        MOV     AH, 9       ; 9号系统功能调用
        INT     21H
        ENDM
OUT1    MACRO   A
        MOV     DL, A
        MOV     AH, 2       ; 输出回车换行
        INT 21H
        ENDM
CRLF    MACRO
        MOV     AH, 2
        MOV     DL, 0AH
        INT     21H         ; 2号系统功能调用
        MOV     DL, 0DH
        INT     21H
        ENDM
```

例2

```
STACK0      MACRO   A
            STACK   SEGMENT PARA STACK 'STACK'
            DB      A                ; 定义堆栈
            STACK   ENDS
            ENDM
```

根据图5. 并使用MACRO、LIB宏库所编写出的主模块程序为：

```
            NAME    MAIN
            EXTRN   F10T2:FAR,F2T10:FAR,QUEUE:FAR
            IF1                      ; 将宏库在第一次扫描时加入一起汇编
            INCLUDE MACRO.LIB
            ENDIF
.386
DATA        SEGMENT USE16 PARA PUBLIC 'DATA'
            BUF     DB 8 DUP(0)      ; 输入的一个十进制数字串存储区
            ARR     DB 32 DUP(0)     ; 转换后的二进制数组存储区
            SIGN    DB 0             ; 是否输完标志，1: 已输完，0: 未输完
            COUNT   DW 0             ; 转换后的二进制数组元素个数计数器
            ERROR   DB ' IS ILLEGAL DIGIT!$'
DATA        ENDS
            STACK0  <200 DUP(0)>     ; 调用宏指令定义堆栈
```

例2

```
CODE        SEGMENT USE16  PARA  PUBLIC 'CODE'
            ASSUME  DS:DATA,ES:DATA,CS:CODE,SS:STACK
START:      MOV     AX, DATA
            MOV     DS, AX
            MOV     ES, AX
            LEA     BX, ARR       ; 数组首址ARR→BX
            MOV     COUNT, 0      ; 0→ARR数组元素个数计数器COUNT
            MOV     SIGN, 0       ; 0→输入串处理完标志SIGN
BEG:        LEA     DI, BUF       ; 数字串存储区首址→DI
            MOV     CX, 0         ; 一个十进制数字串长度计数器清0
NEXT0:      MOV     AH, 1         ; 输入一字符→AL
            INT     21H
            CMP     AL, ‘,’       ; 输入的字符是逗号吗?
            JE      DIGIT         ; 是，一个数已输完
            CMP     AL, 0DH       ; 是回车吗?
            JNE     P             ; 是一般字符，转P
            CMP     COUNT,0       ;
            JE      EXIT          ; 一个数也未输，转结束
            INC     SIGN          ; 已全部输完，标志置1
            JMP     DIGIT         ; 转DIGIT准备调用F10T2子模块
P:          STOSB                 ; 将输入的一个字符送BUF区
            INC     CX            ; 输入字符的个数加1
            JMP     NEXT0         ; 转输入下一个字符
DIGIT:      LEA     SI,BUF        ; 待转换数字串首址送SI
```

例2

```
        MOV     DX,16       ; 需要转换为16位二进制数
        CALL    F10T2       ; 调用F10T2子模块
        CMP     SI,-1
        JE      ERR         ; 如为非法数转错处处理
        MOV     [BX],AX     ; 转换出的十六位二进制数送ARR区
        ADD     BX,2
        INC     COUNT       ; 计数器加1
        CMP     SIGN,1      ; 是否全部输完
        JE      END0        ; 是转排序
        JMP     BEG         ; 否则转输入下一个数
END0:   MOV     CX,COUNT    ; 待排序数的个数送CX
        LEA     SI,ARR      ; 待排序数组首址送SI
        MOV     BP,1        ; 对有符号数排序标志置1
        CALL    QUEUE       ; 调用排序子模块QUEUE对有符号数排序
        CRLF
        MOV     BX,COUNT    ; 待输出数的个数送BX
        LEA     SI,ARR      ; 待输出数组的首址送SI
OUT2:   MOV     AX,[SI]     ; 取一个待输出数送AX
        ADD     SI,2
        MOV     CX,16       ; 待输出数是16位二进制数
        CALL    F2T10       ; 调用F2T10子模块将AX中的数输出
```

例2

```
                OUT1    ‘,’         ；输出逗号作隔离符
                DEC     BX
                JNE     OUT2        ；未输完转OUT2
                JMP     EXIT        ；输完转结束
ERR:            WRITE   ERROR       ；对非法数给出错误提示
EXIT:           MOV     AH,4CH      ；返回DOS
                INT     21H
CODE            ENDS
                END     START
```

十换二子模块F10T2的程序为：

```
                NAME    F10T2
；子模块F10T2
；功能：将十进制ASCII码转换为有符号或无符号二进制数→AX/EAX，如果该数溢出或
；    为非法数，则-1→SI。其中，16位无符号数的范围为0～65535，16位有符号
；    数的范围-32768～＋32767，如果是-65535～－32769，将作为1～32767处理，
；     如果是＋32768～＋65535将作为-32768～-1处理。32位无符号数的范围为0～
；    4294967295，32位有符号数的范围为-2147483648～＋2147483647，如果是
```

例2

```
;      -4294967295～－2147483649将作为1～2147483647处理，如果是2147483648～
;     4294967295，将作为-2147483648～－1处理。
; 入口参数：SI——指向待转换的十进制ASCII码存储区首址
;         CX——存放该十进制ASCII码串的长度
;         DX——存放16位或32位标志
; 出口参数：转换后的二进制数→AX/EAX。如果是非法数或溢出数，则-1→SI
; 所使用的寄存器和变量：
;         BX——中间寄存器
;         SIGN——正负数标记
                PUBLIC   F10T2
DATA            SEGMENT USE16 PARA PUBLIC ′ DATA′
                SIGN     DB ?
DATA            ENDS
STACK           SEGMENT USE16 PARA STACK ′ STACK′
                DB 100 DUP(?)
STACK           ENDS
CODE            SEGMENT USE16 PARA PUBLIC ′ CODE′
                ASSUME CS: CODE，DS: DATA，SS: STACK
F10T2           PROC     FAR
      ┆         ;  过程体请见4. 5. 5节的例3
F10T2           ENDP
CODE            ENDS
                END
```

例2

二换十子模块F2T10的程序为：

```
                NAME    F2T10
; 子模块F2T10
; 功能: 将AX/EAX中的有符号二进制数转换成十进制ASCII输出，其中，子程序RADIX
;     的功能为将EAX中的无符号二进制数按(BX)所规定的基制转换成ASCII码送
;     入SI所指示的缓冲区中
; 调用F2T10的入口参数:
;     AX/EAX——存放待转换的二进制数
;     DX——存放16位或32位标志
; 调用F2T10的出口参数: F2T10子程序无出口参数。
; 所使用的变量:
;   BUF——存放转换后的10进制ASCII码数字串的字节缓冲区
; 调用RADIX的入口参数:
;     EBX——存放要转换的数制基数
;     EAX——存放待转换的32位无符号二进制数
;     SI ——存放转换后的P进制ASCII码数字串的字节缓冲区首址
; 调用RADIX的出口参数:
;     所求P进制ASCII码数字串按高位在前、低位在后的顺序存放在以SI为指针
;   的字节缓冲区中。
;     SI——指向字节缓冲区中最后一个ASCII码的下一个字节处
; 所使用的寄存器:
;     CX ——P进制数字入栈、出栈时的计数器
;     EDX—按除法指令和系统功能调用的规定使用
```

例2

```
                PUBLIC   F2T10，RADIX
DATA            SEGMENT USE16 PARA STACK ′ DATA′
                BUF      DB 12 DUP(?)
DATA            ENDS
STACK           SEGMENT USE16 PARA STACK ′ STACK′
                DB 200 DUP(0)
STACK           ENDS
CODE            SEGMENT USE16 PARA PUBLIC ′ CODE′
    ASSUME      CS: CODE，DS: DATA，ES: DATA，SS: STACK
F2T10           PROC FAR
      ┆         ; 过程体请见4. 5. 6节的例5
F2T10           ENDP
RADIX           PROC    FAR
      ┆         ; 过程体请见4. 5. 5节的例4
RADIX           ENDP
CODE            ENDS
    END
```

排序子模块QUEUE的程序为：

```
                NAME QUEUE
; 子模块QUEUE
; 功能: 将一组十六位无符号或有符号二进制数按从小到大的顺序排列后仍存储在原存储区中
; 入口参数:
; SI——从数组存储区取数指针，初值为该存储区首址
```

例2

```
; CX——待排序数组元素个数
; BP——对无符号或有符号数排序标记，（BP）= 0：对无符号数排序;
;                    （BP）= 1：对有符号数排序;
; 所使用的寄存器:
; DI——中间寄存器，保存数组首址
; AX——存放待比较数
; DX——内循环计数器，CX为外循环计数器

; 本模块不破坏AX、DX、DI的内容
; 出口参数：排好序的数仍存储在原数组存储区中
                PUBLIC  QUEUE
STACK           SEGMENT  SUE16
PARA            STACK ′ STACK′
                DB  200  DUP(0)
STACK           ENDS
CODE            SEGMENT  USE16  PARA  PUBLIC ′ CODE′
                ASSUME  SS:  STACK，CS:  CODE
QUETE           PROC     FAR
      ┆         ; 过程体请见习题4. 10
QUEUE           ENDP
CODE            ENDS
                END
```

例3

编写将十六进制代码加密和解密的程序，要求实现的功能为：

（1）密码表存放有初始密码，但允许用户重新输入密码（密码为16个大写字母）；

（2）密码表建立好后，即可进行加密和解密，要求如下：

- # ；显示命令提示符，等待用户键入加密或解密命令。
- # E ；E命令为等待用户从键盘输入待加密的十六进制代码(最后以∧Z作结束)，加密后送入密文字节存储区IN_CHAN。
- # D ；对IN_CHAN中的密文解密后送明文字节存储区OBJECT。#↙ ；结束操作，返回DOS。

例3

- 为了达到保密的目的，可以对数字代码作加密处理，在需要时再进行解密。
- 本题使用下面最简单的加密方法:
  - 如果要对十六进制代码作加密处理，可以先建立一密码表，例如，将十六进制的16个数字0～F与大写字母A～Z建立关系，这个关系如何对应可由用户任意指定，从而形成密码表。
  - 十六进制数字: 0 1 2 3 4 5 6 7 8 9 A B C D E F
    密码表: N J K A D C E P B I O F M L G H
  - 这样，对每一位十六进制数均可按这个对应关系，利用XLAT指令将其换成对应的大写字母存放在密文表中，从而达到加密的目的。

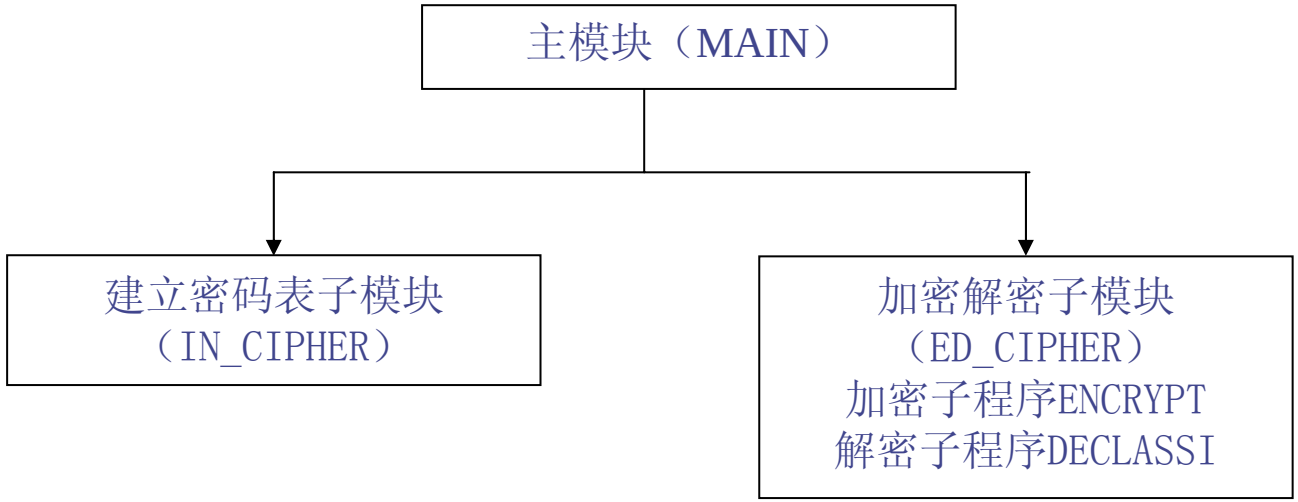
例3

- 为了将加密后的数据解密，可建立解密表，解密表的内容应根据以上密码表与十六进制数字的对应关系由下式运算出来:
  - 十六进制数字在解密表中的位移量 = 对应密码表中字母的ASCII码 − 41H
  - 例如：3在解密表中的位移量 = A的ASCII码 − 41H = 0
    1在解密表中的位移量 = J的ASCII码 − 41H = 9
  - 因此，因此在解密表中，3为第一个元素；1为第10个元素，如此类推，最后得到解密表为:
    3，8，5，4，6，0BH，0EH，0FH，9，1，2，
    0DH，0CH，0，0AH，7

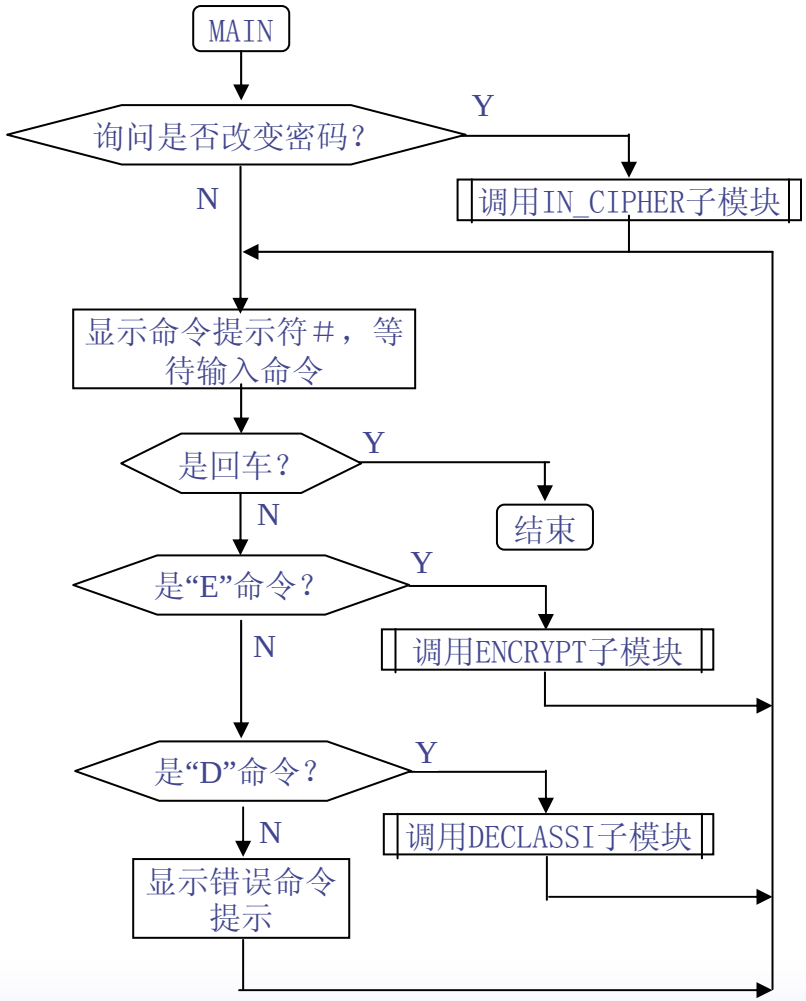
例3

- 有了解密表，当需要对每一位加密后的代码(加密后的密文全部成为大写字母形式)解密时，只要将其ASCII码值减41H，然后用XLAT指令查解密表，得到的即为原始的十六进制数字代码。
- 根据题目的要求，本例由主模块（MAIN）和两个子模块组成，一个为建立密码表子模块、一个为加密解密处理子模块，后者由加密子程序ENCRYPT和解密子程序DECLASSI组成。

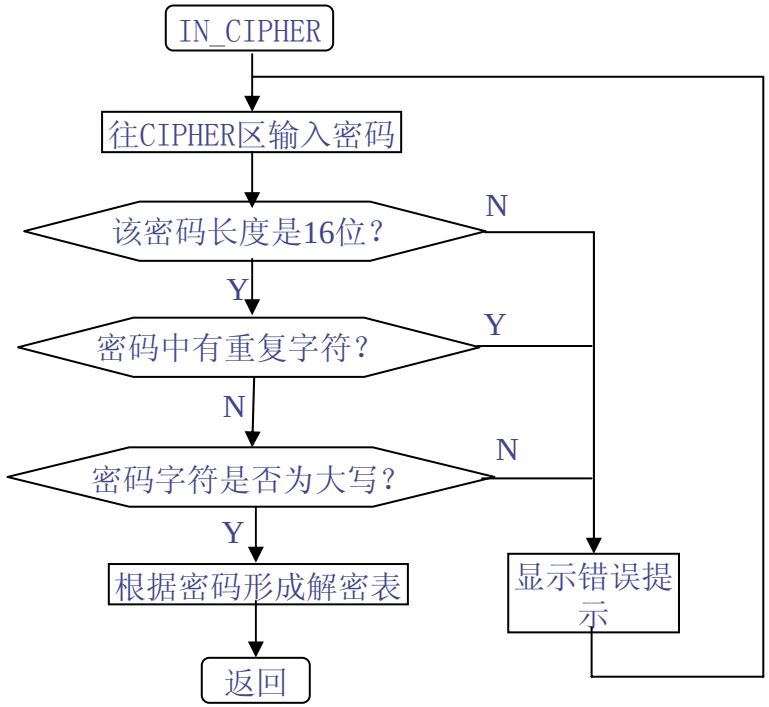
例3模块结构图


主模块（MAIN）
建立密码表子模块
（IN_CIPHER）
加密解密子模块
（ED_CIPHER）
加密子程序ENCRYPT
解密子程序DECLASSI

例3主模块处理流程图


MAIN
询问是否改变密码?
Y
N
调用IN_CIPHER子模块
显示命令提示符#，等待输入命令
是回车?
Y
N
结束
是“E”命令?
Y
N
调用ENCRYPT子模块
是“D”命令?
Y
N
调用DECLASSI子模块
显示错误命令提示

例3 IN_CIPHER子模块流程图



例3


```
                NAME    MAIN
;  主模块MAIN
;  功能为:
;    当用户需要重新输入密码时调用IN_CIPHER子模块;
;    加密和解密命令处理:
;      #     ; 显示命令提示符，等待用户键入加密或解密命令。
;      # E   ; 调用子模块ED_CIPHERE中的子程序ENCRYPT;
;      # D   ; 调用子模块ED_CIPHERE中的子程序DECLASSI;
;      #↙    ; 结束操作，返回DOS。
;  所使用的寄存器:
;     AX、DX按DOS 1、2、9、10号功能调用规定使用。

                EXTRN     IN_CIPHER:FAR,ENCRYPT:FAR,DECLASSI:FAR
IO              MACRO   A, B
                LEA       DX,A
                MOV     AH,B
                INT     21H
                ENDM
.386
DATA            SEGMENT             USE16 PARA PUBLIC 'DATA'
STR0                    DB          0AH,0DH,'Cipher has existed , whether do you modify?
$'
ERR1                    DB          'Illegal commant !$'
CRLF                    DB          0AH, 0DH,'$'
DATA            ENDS
```

例3

```
STACK       SEGMENT          USE16 STACK 'STACK'
            DB       200 DUP(0)
STACK       ENDS
CODE        SEGMENT          USE16 PARA PUBLIC 'CODE'
            ASSUME  CS:CODE,DS:DATA,SS:STACK,ES:DATA
START:      MOV      AX,DATA
            MOV      DS,AX
            MOV      ES,AX
            IO       STR0,9
            MOV      AH,1
            INT      21H
            CMP      AL,‘Y’
            JE       Z          ;  确定是否需要改变密码，如改变调用IN_CIPHER
            CMP      AL,‘y’
            JE       Z
            JMP      COMMA
Z:          CALL     IN_CIPHER
COMMA:      IO       CRLF,9
            MOV      DL,‘#’
            MOV      AH,2       显示命令提示符‘#’
            INT      21H
            MOV      AH,1
            INT      21H
```

例3

```
CMP  AL,0DH
     JE   EXIT
CMP  AL,'E'
     JNE  DD0
     IO   CRLF,9
     CALL ENCRYPT
     JMP  COMMA      ；转输入新命令
DD0:   CMP  AL,'D'
     JNE  ER1        如为‘D’命令调用DECLASSI 子程序解密
     CALL DECLASSI
     JMP  COMMA      ；转输入新命令
ER1:   IO   ERR1,9     ；不是‘E’也不是‘D’命令则显示出错
     JMP  COMMA      ；转输入新命令
EXIT:  MOV  AH,4CH
     INT  21H
CODE   ENDS
     END  START
NAME   IN_CIPHER
；子模块IN_CIPHER
；功能：建立对十六进制数字串加密的密码（由十六位大写字母组成）并送入密码表
；      CIPHER，同时生成解密码送入解密表DEC_CIPHER。
；入口参数：无。
```

例3


```
;  出口参数: 公共变量CIPHER和DEC_CIPHER。其中:
;          CIPHER——密码表首址，CIPHER+2中存放对十六进制数字加密的密码;
;          DEC_CIPHER——解密表首址，存放对加密的密文进行解密的解密码。
;  所使用的寄存器和主要变量:
;     SI——从加密表取字符指针;
;     DI——往解密表送字符指针;
;     DX——检查密码大写字符不重复外循环变量;
;     CX——串操作指令SCAS执行重复次数;
;     BX——存放密码表位移量;
;     AX——按DOS功能调用规定使用;
;     CIPHER——密码表（按输入缓冲区定义），以CIPHER + 2开始的16个字节中存
;           放初始密码;
;     DEC_CIPHER——解密表（16个字节），初值为与密码表中初始密码对应的解密
;           码值。

                PUBLIC   IN_CIPHER,CIPHER,DEC_CIPHER
IO              MACRO    A,B
                LEA      DX,A
                MOV      AH,B
                INT      21H
                ENDM
```

例3

```
.386
DATA        SEGMENT  USE16  PARA PUBLIC 'DATA'
STR0        DB       0AH,0DH,'PLEASE INPUT CIPHER:$'
CIPHER      DB       17,0, 'NJKADCEPBIOFMLGH'                ; 密码表
DEC_CIPHER  DB       3,8,5,4,6,0BH,0EH,0FH,9,1,2,0DH,0CH,0,0AH,7   ; 解密表
ERR0        DB       0AH,0DH,'INPUT IS WRONG，ENCORE！$'
CRLF        DB       0AH,0DH,'$'
DATA        ENDS
STACK       SEGMENT USE16 PARA STACK 'STACK'
            DB       200 DUP(0)
STACK       ENDS
CODE        SEGMENT USE16 PARA PUBLIC 'CODE'
            ASSUME  DS:DATA,CS:CODE,SS:STACK,ES:DATA
IN_CIPHER   PROC     FAR
BEGIN:      IO       STR0, 9
            IO       CIPHER, 10     ; 输入16位密码（大写字母A-Z且不得重复）
至密码表
            CMP      CIPHER+1,16
            JNE      ERR
; 以下为检查输入的16位密码中是否有重复字符
            LEA      SI,CIPHER+2   ; 取输入密码串首址→SI
            MOV      DX,16         ; 串长度→DX
CHECK1:     LODSB                  ; 从输入密码串中取一字符→AL，(SI)指向下一字符
            DEC  DX                ; 其后剩下的字符串长度→DX
            JE   CHE2              ; 已全部检查完了转CHE2
```

例3

```
                MOV     CX, DX          ; SCAS指令执行的重复次数
                MOV     DI, SI          ; 为执行SCAS指令置指针
                REPNZ   SCASB           ; 将(AL)与其后字符逐一比较是否有相等字符
                JE      ERR             ; 相等，则说明出现了重复字符，转出错处理
                JMP     CHECK1          ; 均不等，说明此字符在串中未重复出现，转检
查下一字符
CHE2:           LEA     SI, CIPHER+2    ; 密码表首址→SI
                LEA     DI, DEC_CIPHER  ; 解密表首址→DI
                MOV     CX, 0           ; 解密值
CHECK2:         MOV     BL, [SI]        ; 取一个密码字符→BL
                INC     SI
                MOV     BH, 0           ; 将BL扩展为16位
                CMP     BL, 'A'
                JB      ERR
                CMP     BL, 'Z'
                JA      ERR
                SUB     BL, 'A'         ; 是，形成解密表位移量
                MOV     [BX+DI], CL     ; 将解密值送入解密表
                INC     CL              ; 解密值增1
                CMP     CL,10H          ; 检查解密值是否到十六进制的最大值10H
                JNE     CHECK2          ; 否，转形成下一解密值
END0:           RET
ERR:            IO      ERR0,9          ; 显示错误提示
                JE      BEGIN           ; 转BEGIN重输
IN_CIPHER       ENDP
CODE            ENDS
                END
```

例3


```
                NAME    ED_CIPHER
;  子模块ED_CIPHER
;  子模块ED_CIPHER包含两个子程序：ENCRYPT和DECLASSI。
;   ENCRYPT子程序的功能为：从键盘输入待加密的十六进制字符串，经加密后送
;     入密文表IN_CHAR中。
;     入口参数：外部变量CIPHER+2为密码表首址;
;     出口参数：密文表IN_CHAR存放已加密的密文，供解密时用。
;  所使用的寄存器和主要变量:
;     DI——往密文表送加密后的密文指针;
;     BX——按XLAT指令规定保存密码表首址;
;     AX、DX按DOS功能调用规定使用;
;     IN_CHAR——密文表，即十六进制数字串经加密后的存储区。
;   DECLASSI子程序的功能为：将密文表IN_CHAR中存放的密文解密后存放在明文
;     表OBJECT中。
;     入口参数：外部变量DEC_CIPHER为解密表首址;
;           IN_CHAR为密文表首址;
;     出口参数：明文表OBJECT存放已解密的明文。
;  所使用的寄存器和主要变量:
;     SI——从密文表中取密文指针;
;     DI——往明文表中送解密后的明文指针;
;     BX——按XLAT指令规定保存解密表首址;
;     AX、DX按DOS功能调用规定使用;
;     OBJECT ——明文表，即将密文表中的密文解密成明文后的数据存储区。
```

例3

```
                PUBLIC   ENCRYPT,DECLASSI
                EXTRN    CIPHER:BYTE,DEC_CIPHER:BYTE
                IO       MACRO   A,B
                LEA      DX,A
                MOV      AH,B
                INT      21H
                ENDM
.386
DATA            SEGMENT USE16 PARA PUBLIC 'DATA'
OBJECT          DB       40 DUP(0)              ; 解密后的明文数据存储区
IN_CHAR         DB       40 DUP(0)      ; 密文表（加密后的十六进制串存储区）
CRLF            DB       0AH,0DH,'$'
ERR0            DB       0AH,0DH,'ILLEGAL DIGIT !',0AH,0DH,'$'
DATA            ENDS
STACK           SEGMENT USE16 STACK 'STACK'
                DB       200 DUP(0)
STACK           ENDS
CODE            SEGMENT USE16 PARA PUBLIC 'CODE'
                ASSUME  CS:CODE,DS:DATA,SS:STACK,ES:DATA
ENCRYPT         PROC     FAR
                LEA      DI,IN_CHAR    ; 密文表首址→DI
                LEA      BX,CIPHER+2   ; 密码表首址→BX
IN_TXT:         MOV      AH,1          ; 输入一位待加密的十六进制数→AL
                INT      21H
                CMP      AL,1AH        ; 是结束符∧Z 吗?
                JNE      GM
```

例3

```
                STOSB           ；是，将∧Z送入加密表
                RET             ；返回主程序GM:
                CMP     AL,‘0’
                JB      ER0
                CMP     AL,‘9’
                JBE     NUM
                CMP     AL,‘A’          是16进制数字字符吗？否则转ER0作出错处理
                JB      ER0
                CMP     AL,‘F’
                JA      ER0
                SUB     AL,7
NUM:            SUB     AL,30H
                XLAT    CIPHER+2    ；查密码表将16进制数转换为密文
                STOSB           ；将密文存入密文表
                JMP     IN_TXT      ；转IN_TXT准备再输入密文
ER0:            IO      ERR0,9      ；显示错误提示
                JMP     IN_TXT      ；转IN_TXT准备再输入密文
ENCRYPT         ENDP
```

例3

```
DECLASSI    PROC    FAR
            LEA     SI,IN_CHAR    ; 密文表首址→SI
            LEA     DI,OBJECT     ; 明文表首址→DI
            LEA     BX,DEC_CIPHER ; 解密表首址→BI
NEXT0:      LODSB                 ; 取一密文字符→AL
            CMP     AL,1AH        ; 是否为结束符∧Z？
            JE      RET3          ; 是，转结束
            SUB     AL,41H        ; 否，形成解密表位移量→AL
            XLAT    DEC_CIPHER    ; 从解密表转换为明文→AL
            STOSB                 ; 明文送明文表
            JMP     NEXT0         ; 转处理下一密文字符
RET3:       STOSB                 ; 结束符∧Z送明文表
            RET                   ; 返回主程序
DECLASSI    ENDP
CODE        ENDS
            END
```

总结

1. 串操作指令
2. 宏功能程序设计
3. 模块化程序设计

1. 串操作指令

串传送指令：

- MOVS OPD，OPS/MOVSB/MOVSW

串比较指令：

- CMPS OPD，OPS/CMPSB/CMSW

串查找指令：

- SCAS OPD/SCASB/SCASW

从源串中取数指令：

- LODS OPS/LODSB/LODSW

从目的串中送数指令

- STOS OPD/STOSB/STOSW

总结

1. 串操作指令
2. 宏功能程序设计
3. 模块化程序设计

2. 宏功能程序设计

- 宏指令的定义与调用方式；带间隔符实参的表示方法；
- 宏体中变量和标号；为避免重复定义要用LOCAL；
- 宏定义与子程序的区别。
- 宏库。

总结

1. 串操作指令
2. 宏功能程序设计
3. 模块化程序设计

3. 模块化程序设计

- 定位、组合方式;
- 局部符号、公共符号、外部符号;
- 一般程序改写成子程序、子程序改写成独立子模块的方法;
- 模块化程序设计的方法

作业

1. 试编写一程序，将以变量BUF为首址的100个字节存储单元清零。
2. 设有一个稀疏数组a　i(i=1，2，…，1000)存放在以A为起始地址的字存储区中，现要求将数组加以压缩，使其中的非0元素仍按序存放在A存储区中，而0元素不再出现，试用串操作指令编写实现上述功能的程序。
3. 在STR1串中寻找字符‘A’，若找到则删除，且分两行显示删除前和删除后的STR1串，若找不到，则显示“FIND FAIL”。试用串操作指令编写实现以上功能的程序。
4. 设在TAB字节存储区中存放着0～9的平方值，试编写一程序实现以下功能：从键盘上输入一个数x(0≤x≤9)，试利用平方表计算它的平方值并将结果在显示器上显示出来。其显示格式为：

作业


```
2*2 = 4↙
5*5 = 25↙
A  IS ILLEGAL  DIGIT !↙
9*9 = 81↙
↙
A >
```

5. 某程序设计语言的关键字是大写字母组成的字符串，这些关键字按顺序表的方式存放在以STRIN为首址的字节存储区中，每一个字节存储单元存放一个字符，每个关键字均以0结尾，试利用串操作指令编制一程序，对给定的大写字母中(存放在以SUBS为首址的字节存储区中)查关键字表，确定串是否为该语言的关键字。

6. 编写一宏定义IO，要求实现9号与10号系统调用的功能。
请将以下语句按宏定义IO的要求用宏调用语句代替。

```
LEA  DX，BUF1
MOV  AH，9
INT  21H
 ┆
 LEA  DX，BUF2
MOV  AH，10
INT  21H
```

作业

7. 已知宏定义如下：

```
XCHG0  MACRO  A，B
         MOV   AL，A
XCHG   AL，B
MOV    A，AL
         ENDM
OPP    MACRO  P1，P2，P3，P4
         XCHG0  P1，P4
         XCHG0  P2，P3
         ENDM
```

试说明宏定义OPP的功能，并展开以下宏调用：

```
          ⋮
B    DB 10，20，30，40
          ⋮
OPP  B，B + 1，B + 2，B + 3
```

8. 已知以BCD为首址的字节存储区存放着以升序排列的4个未压缩的BCD码，现需要将它们以降序输出且之间用空格隔开，试利用串操作指令和前面各题中的宏定义编写程序。

作业

9. 已知宏定义如下：

```
OUT_CHR MACRO A
        MOV   DL,A
        MOV   AH,2
        INT   21H
        ENDM
OUT_STR MACRO A,B,C
        LOCAL C
        MOV   CX,B
C:      OUT_CHR A
        LOOP  C
        ENDM
```

调用以上宏定义实现以下功能：从键盘输入一个数（1－9），紧跟其后显示相同个数的'*'后返回DOS；若输入的不是1－9，则给出错误提示后重输，直到输入正确为止。

作业

*10. 有若干个字符存放在以BUF为首址的缓冲区，现需要在第20个字符后面插入10个字符，并将插入后缓冲区的内容全部显示出来，试编制其程序。

要求： 采用条件汇编，如果字符串长度＜20，则不用插入，显示错误提示。只有当串长度≥20时，才进行插入操作。

尽量使用前面已定义过的宏指令和串操作指令。

作业

11. 一位密码分析专家需要对字母A，B，…，Z在某件密码中出现的次数进行统计。假设该密码已用ASCII编码存放在以CODEM为首址的缓冲区中，最后用“*”号表示其结束，试编一程序：统计各字母在密码中出现的次数(设不会溢出)并填入左下表中。

A
字母A出现的次数
B
字母B出现的次数
…
Z
字母Z出现的次数

TAB

作业

将各字母及出现的次数在显示终端上显示出来，其显示格式要求为：

A␣␣␣␣945　　；表示字母A出现了$(945)_{10}$次

B␣␣␣␣1001

C␣␣␣␣359

⋮

Z␣␣␣␣51

说明：可调用子程序库WAN.LIB中的子模块F2T10，其调用说明见5.4节的例2。

作业

12. 有一计算机交通监视系统记录一次交通情况调查，每通过一辆机动车记录一个字符“V”，每通过一辆自行车记录一个字符“B”，每隔10分钟记录一个字符“T”，记录好的信息已存放在以BUFFER为首址的缓冲区中，现在需要统计从第一个字符“T”开始连续10分钟里所通过的自行车和机动车的数目，并在显示器上显示出来，其显示格式要求为：

AMOUNT OF VEHICLES: × × × × ×

AMOUNT OF BIKES: × × × × ×

试编其程序。

说明：可调用子程序库WAN.LIB中的F2T10子模块，其调用说明请见5.4节的例2。

作业

13. 已知某校1000名同学的英文比赛成绩连续地存放在以ENGLI为首址的字存储区中，其存放形式如右图所示。现需查询某一学号学生的成绩与名次，其显示格式为：

```
   学号    分数     名次
00050    98       2
01021    70       21
61011    NOT      FOUND  OUT
00705    88       10
↙
A >
```

试编其程序。

说明：可以调用子程序库WAN.LIB中的子模块F2T10和F10T2，其模块调用说明请见5.4节的例2。