

实验 7 键盘操作与显示

一、实验目的

熟悉指令系统，掌握常用指令的用法；通过实验加深对各种寻址方式的理解；能熟练使用 DEBUG 中的命令对指令进行反汇编，观察并了解机器代码。

二、实验题

1. 把数据段中 1 维数组 AA_1 变量地址中连续 7 个数 (1,3,5,7,2,4,6) 读出，把每个数加 2 后再存入到数据段中 BB_1 数组开始的标号地址中去，并显示出 BB_1 数组中每个数来 (之间用空格分开)，参考程序如下：

```
DATA SEGMENT
    ORG 0100H
    AA_1    DB      1,3,5,7,2,4,6
    ORG 0150H
    BB_1    DB      7 dup(?)
    COUNT  DW      7
DATA ENDS
CSEG SEGMENT
    ASSUME CS:CSEG,DS:DATA
START:MOV    AX, DATA
        MOV   DS, AX
        MOV   CX, COUNT
        LEA   SI, AA_1
        LEA   DI, BB_1
LP1:    MOV   AL, [SI]
        ADD   AL, 2
        MOV   [DI], AL
        INC   SI
        INC   DI
        LOOP  LP1
        LEA   SI, BB_1
        MOV   CX, COUNT
DISP:   MOV   DL, [SI]
```

```

ADD DL, 30H
MOV AH, 02
INT 21H
MOV DL, ' '
MOV AH, 2
INT 21H
INC SI
LOOP DISP
MOV AH, 4CH
INT 21H
CSEG ENDS
END START

```

考试 2-3 道主观编程题

➤ 运行结果

```

(pt2) D:\code\EXPERIMENTAL_REPORT\汇编语言\E7_键盘操作与显示\c1>debug c1.exe
-u
2063:0000 B84D20      MOV     AX,204D
2063:0003 8ED8          MOV     DS,AX
2063:0005 8B0E5701      MOV     CX,[0157]
2063:0009 8D360001      LEA     SI,[0100]
2063:000D 8D3E5001      LEA     DI,[0150]
2063:0011 8A04          MOV     AL,[SI]
2063:0013 0402          ADD     AL,02
2063:0015 8805          MOV     [DI],AL
2063:0017 46            INC     SI
2063:0018 47            INC     DI
2063:0019 E2F6          LOOP    0011
2063:001B 8D365001      LEA     SI,[0150]
2063:001F 8B0E5701      MOV     CX,[0157]
-g
3 5 7 9 4 6 8

```

请回答以下问题：

(1) 指令 MOV AH, 02H 中 02H 的含义是？

- MOV AH, 02H 指令将立即数 02H 存入 AH 寄存器显示字符输出功能。02H 功能会将 DL 寄存器中的 ASCII 字符显示到标准输出设备（通常是屏幕）上。

(2) 指令 ADD DL, 30H 的作用是什么？

➤ 指令 ADD DL, 30H 的作用是将 DL 寄存器中的数值转换为对应的 ASCII 码字符。

- 30H 是 ASCII 码中数字 '0' 的十六进制表示
- 当 DL 中存储的是一个 0-9 的数值时，加上 30H 后会变成对应数字的 ASCII 码

注意：这种转换方法只适用于单个十进制数字（0-9）。对于大于 9 的数字，这种简单的加 30H 方法会产生错误的字符。

(3) 除了参考程序中用的访问方式，还可以用什么方式访问 AA_1 数组里的元

素？

参考程序中使用的是基于寄存器间接寻址方式（使用 SI 作为指针）来访问 AA_1 数组元素。除此之外，还可以使用以下方式：

- 直接寻址：

```
MOV AL, AA_1[0]    ; 访问第一个元素
MOV AL, AA_1[1]    ; 访问第二个元素
```

- 基址寻址：

```
MOV BX, OFFSET AA_1
MOV AL, [BX]        ; 访问第一个元素
MOV AL, [BX+1]      ; 访问第二个元素
```

- 变址寻址：

```
MOV SI, 0
MOV AL, AA_1[SI]    ; 访问第一个元素
INC SI
MOV AL, AA_1[SI]    ; 访问第二个元素
```

- 基址加变址寻址：

```
MOV BX, OFFSET AA_1
MOV SI, 1
MOV AL, [BX+SI-1]   ; 访问第一个元素
MOV AL, [BX+SI]     ; 访问第二个元素
```

2. 编写程序，从键盘接收一个小写字母，然后找出它的前导字符和后续字符，再按顺序显示这三个字符。把源程序附在下面，并把 debug 单步调试的截图附在下面。

没有数据段的设置,直接编写代码段

键盘输入,ASCII 码放在 AL 里,显示放在 DL 里

MOV AH,1

INT 21H

CMP AL,'a'

JB 退出

和'z'比较

➤ 程序源码

```
CSEG SEGMENT
    ASSUME CS:CSEG
    ORG 100H
START:
    ; 从键盘接收一个小写字母
    MOV AH,1
    INT 21H

    MOV DL,AL
    DEC DL

    ; 设置循环次数
    MOV CX,3      ; 设置循环次数

LOOP1:
    ; 检查输入是否为小写字母
    CMP DL,'a'
    JB EXIT      ; 小于'a'就退出
    CMP DL,'z'
    JA EXIT      ; 大于'z'就退出

    MOV AH,2
    INT 21H

    ADD DL,1

    LOOP LOOP1    ; CX 减1, 若不为0 则跳转到LOOP1 继续循环

EXIT:
    MOV AH,4CH
    INT 21H
```

```
CSEG ENDS
      END START
```

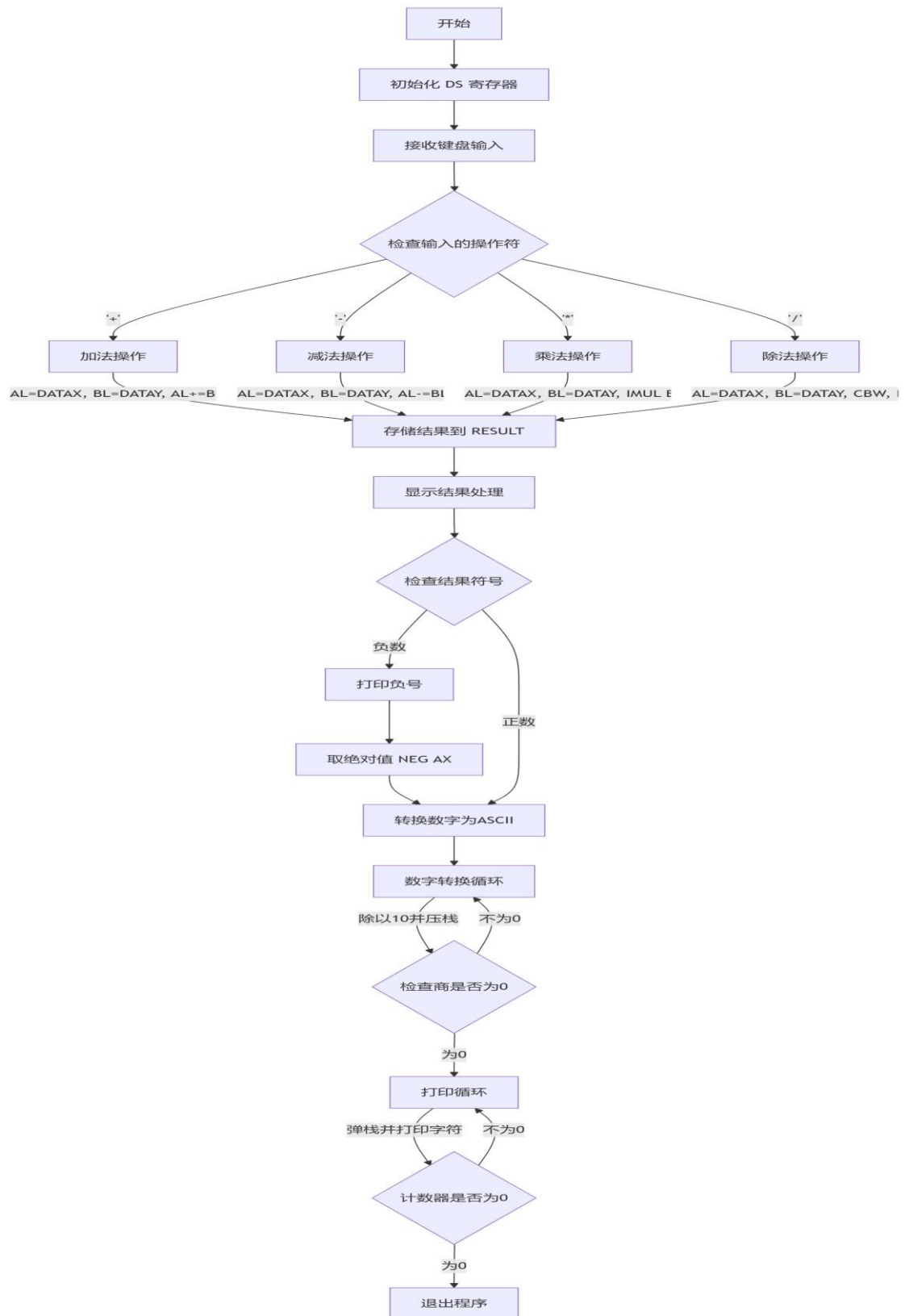
➤ 运行结果

```
(pt2) D:\code\EXPERIMENTAL_REPORT\汇编语言\E7_键盘操作与显示\c2>debug c2.exe
-u
204D:0100 B401      MOV     AH,01
204D:0102 CD21      INT     21
204D:0104 8AD0      MOV     DL,AL
204D:0106 FECA      DEC     DL
204D:0108 B90300     MOV     CX,0003
204D:010B 80FA61     CMP     DL,61
204D:010E 720E      JB      011E
204D:0110 80FA7A     CMP     DL,7A
204D:0113 7709      JA      011E
204D:0115 B402      MOV     AH,02
204D:0117 CD21      INT     21
204D:0119 80C201     ADD     DL,01
204D:011C E2ED      LOOP    010B
204D:011E B44C      MOV     AH,4C
-g
tstu
Program terminated normally
-
```

3. 已知 DATAX 和 DATAY 单元各存放一个带符号字节数据，从键盘上接收加（+）、减（-）、乘（*）或除（/）符号，然后完成相应运算，把结果显示在屏幕上。

3 个变量(带符号)

➤ 程序思路



➤ 程序源码

DATA SEGMENT

DATAH DB 50 ; 带符号字节数据, 负数

```

    DATAY DB 10      ; 带符号字节数据, 正数
    RESULT DB 0
    BUF DB 4 DUP(?)
DATA ENDS

CSEG SEGMENT
    ASSUME CS:CSEG,DS:DATA
START:
    MOV AX,DATA
    MOV DS,AX

    ;从键盘接收+,-,*,/
    MOV AH,1
    INT 21H

    CMP AL,'+'
    JE ADD_OP
    CMP AL,'-'
    JE SUB_OP
    CMP AL,'*'
    JE MUL_OP
    CMP AL,'/'
    JE DIV_OP

ADD_OP:
    MOV AL,DATAH
    MOV BL,DATAY
    ADD AL,BL
    MOV RESULT,AL
    JMP SHOW_RESULT

SUB_OP:
    MOV AL,DATAH
    MOV BL,DATAY
    SUB AL,BL
    MOV RESULT,AL
    JMP SHOW_RESULT

MUL_OP:
    ; 字节乘法:
    ; (AL) * (OPS8) → AX
    MOV AL,DATAH
    MOV BL,DATAY
    IMUL BL

```

```
MOV RESULT,AL
JMP SHOW_RESULT
```

DIV_OP:

```
MOV AL,DATAX
MOV BL,DATAY
CBW ;将AL 符号扩展到AX
IDIV BL
MOV RESULT,AL
JMP SHOW_RESULT
```

; 难点在打印, 将每次除以 10, 把余数入栈, 然后出栈, 打印

SHOW_RESULT:

```
MOV AL, RESULT
CBW
CMP AL, 0
JGE SHOW
PUSH AX ; 保存原始AL
MOV DL, '-'
MOV AH, 2
INT 21H
POP AX ; 恢复AL
NEG AL
CBW
```

SHOW:

```
MOV SI,0 ; SI 为BUF 索引
MOV BX,10
```

CONV_LOOP:

```
MOV DX,0
DIV BX
ADD DL,'0'
MOV BUF[SI],DL
INC SI
CMP AX,0
JNZ CONV_LOOP
```

PRINT_LOOP:

```
DEC SI
MOV DL,BUF[SI]
MOV AH,2
INT 21H
CMP SI,0
```



```

    JNZ PRINT_LOOP
    JMP EXIT

EXIT:
    MOV AH,4CH
    INT 21H

CSEG ENDS
    END START

```

➤ 运行结果

```

    DATAX DB 6      ; 带符号字节数据, 负数
    DATAY DB -2     ; 带符号字节数据, 正数

```

我们设置 X,Y 分别为 6 和-2

+*/对应不同的结果(最左边为操作数)

```

(pt2) C:\Users\tk\AppData\Roaming\Cursor\User\globalStorage\xsro.masm-tasm\workspace>TEST
+4
(pt2) C:\Users\tk\AppData\Roaming\Cursor\User\globalStorage\xsro.masm-tasm\workspace>TEST
-8
(pt2) C:\Users\tk\AppData\Roaming\Cursor\User\globalStorage\xsro.masm-tasm\workspace>TEST
*-12
(pt2) C:\Users\tk\AppData\Roaming\Cursor\User\globalStorage\xsro.masm-tasm\workspace>TEST
/-3

```

➤ 遇到和解决的问题

- 当 RESULT 为负数时,我在打印结果的时候我们需要先打印'-'号,INT 21H 会把 AL 的值重新设置
- 使用 IDIV BX 命令前需要我们将 AX 使用 CBW 命令将 AL 扩展为 AX

➤ 使用 AI 改进的方案

```

; 简单计算器: 对 DATAX 和 DATAY 中的有符号字节数据进行四则运算
; 支持加(+)、减(-)、乘(*)、除(/)四种运算符
; 操作数直接存储在 DATAX 和 DATAY 中

```

```

DATA SEGMENT
    DATAX DB 6      ; 第一个操作数

```

```

    DATAY    DB    -2            ; 第二个操作数
    MSG_OP   DB    0DH,0AH,'Input operator (+, -, *, /): $'
    MSG_RES  DB    0DH,0AH,'Result: $'
    MSG_ERR  DB    0DH,0AH,'Division by zero! $'
DATA    ENDS

CODE    SEGMENT
    ASSUME  CS:CODE, DS:DATA
START:
    MOV    AX, DATA
    MOV    DS, AX

    ; 显示操作符提示消息
    LEA    DX, MSG_OP
    MOV    AH, 09H
    INT    21H

    ; 输入操作符
    MOV    AH, 01H
    INT    21H
    MOV    BL, AL                ; 保存操作符在 BL 中

    ; 显示结果提示消息
    LEA    DX, MSG_RES
    MOV    AH, 09H
    INT    21H

    ; 根据操作符执行相应运算
    CMP    BL, '+'
    JE     DO_ADD
    CMP    BL, '-'
    JE     DO_SUB
    CMP    BL, '*'
    JE     DO_MUL
    CMP    BL, '/'
    JE     DO_DIV
    JMP    EXIT                  ; 非法操作符直接退出

DO_ADD:
    MOV    AL, DATAX
    ADD    AL, DATAY
    JMP    SHOW_RESULT

DO_SUB:

```

```

    MOV AL, DATAX
    SUB AL, DATAY
    JMP SHOW_RESULT

DO_MUL:
    MOV AL, DATAX
    IMUL DATAY      ; AL * DATAY -> AX
    JMP SHOW_AX

DO_DIV:
    MOV AL, DATAX
    CBW             ; 符号扩展 AL->AX
    MOV BL, DATAY
    CMP BL, 0
    JNE DIV_OK
    LEA DX, MSG_ERR ; 除数为0, 显示错误
    MOV AH, 09H
    INT 21H
    JMP EXIT

DIV_OK:
    IDIV BL         ; AX / BL -> AL(商), AH(余数)
    JMP SHOW_RESULT

SHOW_RESULT:
    CBW             ; AL->AX, 准备显示

SHOW_AX:
    ; AX 中为结果, 带符号显示
    CALL PRINT_SIGNED_AX
    JMP EXIT

; 打印AX 中的有符号整数 (-128~32767)
PRINT_SIGNED_AX PROC
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    CMP AX, 0
    JGE PRINT_DEC
    ; 负数, 先输出 '-'
    MOV DL, '-'
    MOV AH, 02H
    INT 21H
    NEG AX

```

```

PRINT_DEC:
    MOV CX, 0
    MOV BX, 10
CONV_LOOP:
    XOR DX, DX
    DIV BX
    PUSH DX
    INC CX
    CMP AX, 0
    JNZ CONV_LOOP

PRINT_LOOP:
    POP DX
    ADD DL, '0'
    MOV AH, 02H
    INT 21H
    LOOP PRINT_LOOP

; 输出换行
    MOV DL, 0DH
    MOV AH, 02H
    INT 21H
    MOV DL, 0AH
    MOV AH, 02H
    INT 21H

    POP DX
    POP CX
    POP BX
    POP AX
    RET
PRINT_SIGNED_AX ENDP

EXIT:
    MOV AH, 4CH
    INT 21H
CODE ENDS
END START

```

➤ 调试过程

- 接收操作符

```

AX=0170 BX=0000 CX=00A1 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0007 NU UP EI PL NZ NA PO NC
0771:0007 CD21 INT 21
-p
*
AX=012A BX=0000 CX=00A1 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0009 NU UP EI PL NZ NA PO NC
0771:0009 3C2B CMP AL,2B

```

● 转到并执行乘法分支程序

```

AX=012A BX=0000 CX=00A1 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0013 NU UP EI PL ZR NA PE NC
0771:0013 7420 JZ 0035
-p
AX=012A BX=0000 CX=00A1 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0035 NU UP EI PL ZR NA PE NC
0771:0035 A00000 MOV AL,[0000] DS:0000=06
-p
AX=0106 BX=0000 CX=00A1 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0038 NU UP EI PL ZR NA PE NC
0771:0038 8A1E0100 MOV BL,[0001] DS:0001=FE
-p
AX=0106 BX=00FE CX=00A1 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=003C NU UP EI PL ZR NA PE NC
0771:003C F6EB IMUL BL
-p
AX=FFF4 BX=00FE CX=00A1 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=003E NU UP EI PL ZR NA PE NC
0771:003E A20200 MOV [0002],AL DS:0002=00
- ;

```

● INT 21 执行中断命令打印 DL 命令后 AL 的值发生改变我们使用 pop 命令恢

复 AX 的值

```

AX=FFF4 BX=00FE CX=00A1 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=005B NU UP EI NG NZ NA PO NC
0771:005B B22D MOV DL,2D
-p
AX=FFF4 BX=00FE CX=00A1 DX=002D SP=FFFE BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=005D NU UP EI NG NZ NA PO NC
0771:005D B402 MOV AH,02
-p
AX=02F4 BX=00FE CX=00A1 DX=002D SP=FFFE BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=005F NU UP EI NG NZ NA PO NC
0771:005F CD21 INT 21
-p
AX=022D BX=00FE CX=00A1 DX=002D SP=FFFE BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0061 NU UP EI NG NZ NA PO NC
0771:0061 5B POP AX
-p
AX=FFF4 BX=00FE CX=00A1 DX=002D SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0062 NU UP EI NG NZ NA PO NC
0771:0062 F6DB NEG AL
- ;

```

其中 AX=FFF4H=-12D

10.255.252.170