

## 电工电子实验中心 实验报告

课程名称：\_\_\_\_微机原理与接口技术实验\_\_\_\_

实验名称：\_\_\_\_四则运算、数据统计、代码转换、数据块移动\_\_\_\_

姓名：\_\_\_\_陈伟\_\_\_\_学号：\_\_\_\_161710223\_\_\_\_

评定成绩：\_\_\_\_审阅教师：\_\_\_\_黄晓晴\_\_\_\_

实验时间：\_\_\_\_2019 年 12 月 6 日\_\_\_\_

南京航空航天大学

## 目录

一、四则运算.....	2
1.1 实验目的要求.....	2
1.2 实验任务.....	2
1.3 实验代码.....	2
1.4 探究内容（选做）.....	4
1.5 实验的运行数据及分析.....	4
1.6 实验讨论及心得体会.....	5
二、数据统计.....	6
2.1 实验目的要求.....	6
2.2 实验任务.....	6
2.3 实验代码.....	6
2.4 探究内容（选做）.....	8
2.5 实验的运行数据及分析.....	14
2.6 实验讨论及心得体会.....	15
三、代码转换.....	16
3.1 实验目的要求.....	16
3.2 实验任务.....	16
3.3 实验代码.....	16
3.4、探究内容（选做）.....	18
3.5 实验的运行数据及分析.....	21
3.6 实验讨论及心得体会.....	21
四、数据块移动.....	22
4.1 实验目的要求.....	22
4.2 实验任务.....	22
4.3 实验代码.....	23
4.4 探究内容（选做）.....	25
4.5 实验的运行数据及分析.....	29
4.6 实验讨论及心得体会.....	30

# 一、四则运算

## 1.1 实验目的要求

1. 熟悉汇编语言程序的框架结构，掌握顺序结构的编程方法。
2. 熟悉 Tddebug 调试环境和 Turbo Debugger 的使用。
3. 理解 X86 内存数据的组织方式。
4. 理解基本的内存寻址方式。

## 1.2 实验任务

完成 32 位无符号数的加法、减法，16 位乘以 16 位，32 位除以 16 位除法的四则运算练习。

## 1.3 实验代码

```

DATA SEGMENT
A    DW    1234H,5678H    ;被加数
B    DW    0FEDCH,123AH  ;加数
C    DW    2    DUP(0)    ;预留和
AD   DW    0FEDCH,0BA98H ;被减数
BD   DW    1234H,789AH   ;减数
CD   DW    2    DUP(0)    ;预留差
A1   DW    0D678H        ;被乘数
B1   DW    0012H         ;乘数
C1   DW    2    DUP(0)    ;预留积
A2   DW    5678H,0234H   ;被除数
B2   DW    0F234H        ;除数
C2   DW    2    DUP(0)    ;预留商,余数
DATA ENDS

STACK1 SEGMENT STACK
      DB    100 DUP(0)
STACK1 ENDS

CODE SEGMENT
      ASSUME CS:CODE,DS:DATA,SS:STACK1

START PROC FAR

```

```
PUSH DS ;标准序
MOV AX,0
PUSH AX
MOV AX,DATA
MOV DS,AX

MOV AX,A ;32 位加 32 位
ADD AX,B
MOV C,AX
MOV AX,A+2
ADC AX,B+2 ;用 ADC 考虑到 CF
MOV C+2,AX

MOV AX,AD ;32 位减 32 位
SUB AX,BD
MOV CD,AX
MOV AX,AD+2
SBB AX,BD+2 ;用 SBB 考虑到 CF
MOV CD+2,AX

MOV AX,A1
MUL B1
MOV C1,AX ;将 AX 中保存的结果低 16 位存放到 C1 中
MOV C1+2,DX ;将 DX 中保存的结果高 16 位存放到 C1+2 中

MOV DX,A2+2 ;DX 存放高 16 位
MOV AX,A2 ;AX 存放低 16 位
DIV B2
MOV C2,AX ;保存商
MOV C2+2,DX ;保存余数

RET
START ENDP

CODE ENDS
END START
```

## 1.4 探究内容（选做）

### 1) 若需进行有符号数的运算，需要注意什么问题？如何实现？

加法、减法：步骤同有符号数一样，因为计算机进行加减法运算是不区分有符号数和无符号数的，不过最后要关注 OF 是否为 1，为 1 溢出

乘法：被乘数存放在 AX 中，但是要用 IMUL 指令，结果存放在 DX:AX 中

除法：被除数放在 DX:AX 中，但是要用 IDIV 指令，商送 AX，余数送 DX

### 2) 上述实验中，我们在 80X86 的实模式中实现了 32 位的四则运算。例如，乘法运算为 16 位乘以 16 位，运算结果为 32 位。请思考如何利用 32 位指令，实现 64 位的四则运算？

由于 80x86 里面没有 32 位寄存器，所以本题不做代码演示，只阐述思路。

加法：用 ADD 指令低 32 位相加，再用 ADC 指令高 32 位相加

减法：用 SUB 指令低 32 位相减，再用 SBB 指令高 32 位相减

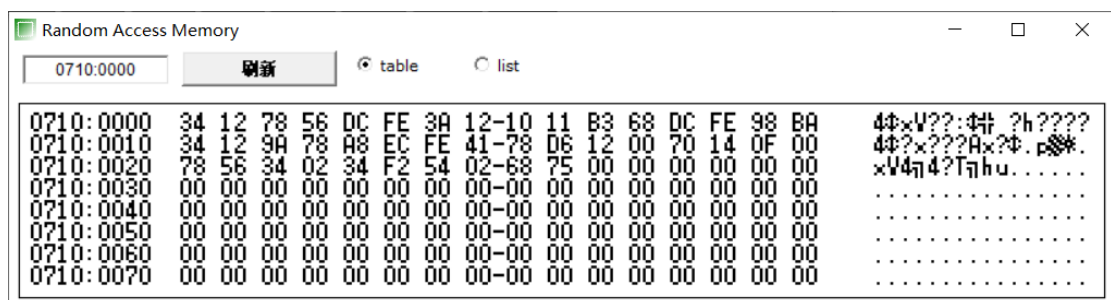
乘法：将被乘数放在 EAX 中，再用 MUL 指令与乘数相乘，结果高 32 位存放在 EDX，低 32 位存放在 EAX

除法：将被除数高 32 位放在 EDX 中，低 32 位放在 EAX 中，再用 DIV 指令与除数相除，商送 EAX，余数送 EDX

## 1.5 实验的运行数据及分析

### 1. 实验数据记录

软件环境，emu8086



DS:0000	34	12	78	56	DC	FE	3A	12
DS:0008	10	11	B3	68	DC	FE	98	BA
DS:0010	34	12	9A	78	A8	EC	FE	41
DS:0018	78	D6	12	00	70	14	0F	00
DS:0020	78	56	34	02	34	F2	54	02
DS:0028	68	75						

2. 数据分析:

和: 68B31110H 差: 41FEECA8H 积: 000F1470H 商: 0254H 余数: 7568H

## 1.6 实验讨论及心得体会

本次实验加深了我对 ADD、ADC、SUB、SBB、MUL、DIV 指令的使用，了解了乘除法指令中 DX、AX 存放值的内容，这次实验也加深了对大小端存放数据的掌握，并且学会了如何规范的写汇编代码

## 二、数据统计

### 2.1 实验目的要求

1. 熟悉汇编语言程序的框架结构，掌握顺序结构的编程方法。
2. 熟悉 Td debug 调试环境和 Turbo Debugger 的使用。
3. 理解 X86 内存数据的组织方式。
4. 理解基本的内存寻址方式。

### 2.2 实验任务

本实验要求通过求某数据区内负数的个数来表现循环程序的结构形式。要求实验程序在数据段中存放一组数据，分类统计数据中正数、负数和零的个数，并分别存入内存变量 Positive、Negative 和 Zero 中。将所有数据累加求和，存入 SUM 中。

### 2.3 实验代码

#### DATA SEGMENT

```
NUM      DB 12H,88H,82H,89H,33H,90H,0H,10H,0BDH,01H
Positive  DB DUP (0)
Negative  DB DUP (0)
Zero      DB DUP (0)
SUM       DW 2 DUP (0)
```

#### DATA ENDS

#### STACK1 SEGMENT STACK

```
DB 100 DUP(0)
```

#### STACK1 ENDS

#### CODE SEGMENT

```
ASSUME CS:CODE,DS:DATA,SS:STACK1
```

#### START PROC FAR

```
PUSH DS
MOV  AX, 0
PUSH AX
MOV  AX, DATA
MOV  DS, AX
```

```
MOV CX,10           ;循环 10 次
LEA SI,NUM
MOV BX,0
LAB1:
MOV DL,[SI]
CMP DL,0           ;判断 num 是否为 0
JG LAB2           ;num>0,跳转到 LAB2
JL LAB3           ;num<0,跳转到 LAB3
INC ZERO          ;num=0,ZERO 加 1
JMP LAB4
LAB2:
INC Positive      ;负数+1
JMP LAB4
LAB3:
INC Negative      ;正数+1
LAB4:
MOV AL,[SI]
CBW              ;符号拓展, 8 位拓展为 16 位
ADD SUM,AX       ;+sum
ADC [SUM+2],0
INC SI
LOOP LAB1        ;循环

RET

START ENDP
CODE ENDS
END START
```



## 2.4 探究内容（选做）

代码：两个选做题写在了一个代码里，先执行完输出统计个数再进行排序，最后输出排序结果

```
DATA SEGMENT
    NUM      DB 12H, 88H, 82H, 89H, 33H, 90H, 0H, 10H, 0BDH, 01H
    Positive DB DUP (0)
    Negative DB DUP (0)
    Zero     DB DUP (0)
    SUM      DW 2 DUP (0)
    result   DB 0, 0, 'H', 0DH, 0AH, '$'
    results  DB 0, 0, 0, 0, 'H', 0DH, 0AH, '$'
    result1  DB "Positive:$"
    result2  DB "Negative:$"
    result3  DB "Zero:$"
    result4  DB "SUM:$"
    result5  DB "SortResult:$"
    resultsort DB 0, 0, 'H', ' ', '$'
DATA ENDS
```

```
STACK1 SEGMENT STACK
    DB 100 DUP(0)
STACK1 ENDS
```

```
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:STACK1
```

```
START PROC FAR
BEGIN:
    PUSH DS
    MOV AX, 0
    PUSH AX
    MOV AX, DATA
    MOV DS, AX

    MOV CX, 10           ;循环 10 次
    LEA SI, NUM
    MOV BX, 0
LAB1:
    MOV DL, [SI]
```

```

    CMP DL, 0                ;判断 num 是否为 0
    JG LAB2                 ;num>0, 跳转到 LAB2
    JL LAB3                 ;num<0, 跳转到 LAB3
    INC ZERO                ;num=0, ZERO 加 1
    JMP LAB4

LAB2:
    INC Positive            ;负数+1
    JMP LAB4

LAB3:
    INC Negative            ;正数+1

LAB4:
    MOV AL, [SI]
    CBW                    ;符号拓展, 8 位拓展为 16 位
    ADD SUM, AX             ;+sum
    ADC [SUM+2], 0
    INC SI
    LOOP LAB1               ;循环

    MOV DX, OFFSET result1  ;显示 Positive
    MOV AH, 9
    INT 21H
    MOV BL, Positive
    MOV AL, BL
    SHR AL, 4
    CALL ToASCII            ;把高四位转换为对应的 ASCII 码
    MOV [result], AL
    MOV AL, BL
    CALL ToASCII            ;把低四位转换为对应的 ASCII 码
    MOV [result+1], AL
    MOV DX, OFFSET result
    MOV AH, 9
    INT 21H

    MOV DX, OFFSET result2  ;显示 Negative
    MOV AH, 9
    INT 21H
    MOV BL, Negative
    MOV AL, BL
    SHR AL, 4
    CALL ToASCII
    MOV [result], AL

```

```
MOV AL, BL
CALL ToASCII
MOV [result+1], AL
MOV DX, OFFSET result
MOV AH, 9
INT 21H
```

```
MOV DX, OFFSET result3      ;显示 Zero
MOV AH, 9
INT 21H
MOV BL, Zero
MOV AL, BL
SHR AL, 4
CALL ToASCII
MOV [result], AL
MOV AL, BL
CALL ToASCII
MOV [result+1], AL
MOV DX, OFFSET result
MOV AH, 9
INT 21H
```

```
MOV DX, OFFSET result4      ;显示 Sum
MOV AH, 9
INT 21H
```

```
MOV BX, SUM
MOV AL, BL
SHR AL, 4
CALL ToASCII
MOV [results+2], AL
MOV AL, BL
CALL ToASCII
MOV [results+3], AL
SHR BX, 8
MOV AL, BL
SHR AL, 4
CALL ToASCII
MOV [results], AL
MOV AL, BL
CALL ToASCII
MOV [results+1], AL
```

```

MOV  DX, OFFSET results
MOV  AH, 9
INT  21H

XOR  AX, AX                ;选择排序算法
MOV  BX, OFFSET NUM        ;I=0
MOV  SI, 0
FORI:
    MOV  DI, SI
    INC  DI                ;J=I+1
FORJ:
    MOV  AL, [BX+SI]
    CMP  AL, [BX+DI]       ;A[i]与A[j]比较
    JLE  NEXTJ             ;A[i]小于等于A[j]跳转
    XCHG AL, [BX+DI]       ;A[i]与A[j]交换
    MOV  [BX+SI], AL
NEXTJ:
    INC  DI                ;J=J+1
    CMP  DI, 10            ;J<10 跳转
    JB  FORJ
NeXTI:
    INC  SI                ;I=I+1
    CMP  SI, 9
    JB  FORI              ;I<9 时跳转

MOV  DX, OFFSET result5    ;显示排序结果
MOV  AH, 9
INT  21H
MOV  CL, 10
MOV  SI, OFFSET NUM
PutNum:
    MOV  BL, [SI]
    MOV  AL, BL
    SHR  AL, 4
    CALL ToASCII
    MOV  [resultsort], AL
    MOV  AL, BL
    CALL ToASCII
    MOV  [resultsort+1], AL
    MOV  DX, OFFSET resultsort
    MOV  AH, 9

```

```

    INT 21H
    INC SI
    LOOP PutNum
    MOV AH, 4CH
    INT 21H
    RET
START ENDP

ToASCII PROC
    AND AL, 0FH
    ADD AL, '0'
    CMP AL, '9'
    JBE LAB5
    ADD AL, 7
LAB5:
    RET
ToASCII ENDP

CODE ENDS
    END BEGIN

```

## 分析:

1) 若需要将程序得到的结果显示在屏幕上, 如何处理?

解决: 以输出 Positive 这个结果(单字节数)为例, 首先将数据临时存入 BL 中, 再将 BL 赋值给 AL, 将 AL 右移 4 位, 先处理高位, 调用写好的子函数将高 4 位转换为 ASCII 码(先将 AL 和 0FH 进行与操作), 再将 BL 赋值给 AL, 处理低 4 位, 将低 4 位转化为 ASCII 码, 最后输出结果, 其他操作类似

核心代码:

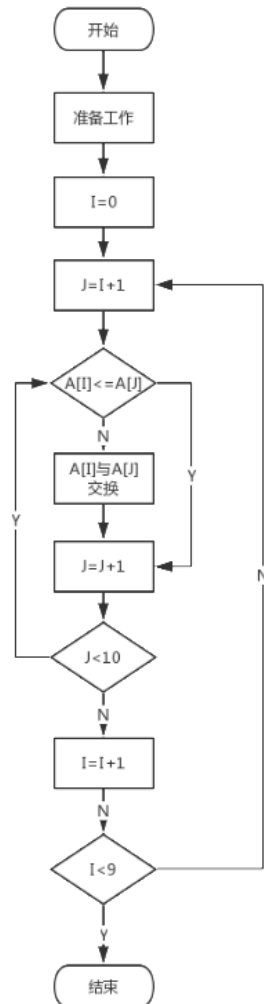
```

MOV DX, OFFSET result1      ;显示Positive
MOV AH, 9
INT 21H
MOV BL, Positive
MOV AL, BL
SHR AL, 4
CALL ToASCII                 ;把高四位转换为对应的ASCII码
MOV [result], AL
MOV AL, BL
CALL ToASCII                 ;把低四位转换为对应的ASCII码
MOV [result+1], AL
MOV DX, OFFSET result
MOV AH, 9
INT 21H

```

## 2) 利用某种排序算法，对原始数据进行排序。

利用如下的选择排序算法即可，控制好判断条件



核心代码：

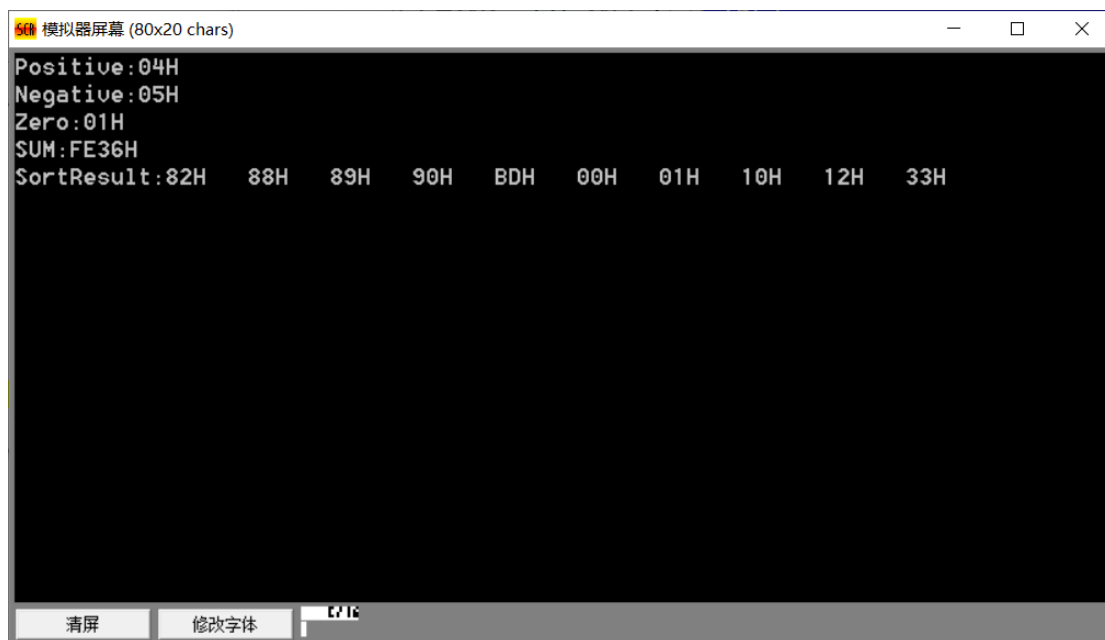
<code>XOR AX,AX</code>	;	选择排序算法
<code>MOV BX,OFFSET NUM</code>	;	I=0
<code>MOV SI,0</code>		
<code>FORI:</code>		
<code>MOV DI,SI</code>		
<code>INC DI</code>	;	J=I+1
<code>FORJ:</code>		
<code>MOV AL,[BX+SI]</code>		
<code>CMP AL,[BX+DI]</code>	;	A[i]与A[j]比较
<code>JLE NEXTJ</code>	;	A[i]小于等于A[j]跳转
<code>XCHG AL,[BX+DI]</code>	;	A[i]与A[j]交换
<code>MOV [BX+SI],AL</code>		
<code>NEXTJ:</code>		
<code>INC DI</code>	;	J=J+1
<code>CMP DI,10</code>	;	J<10跳转
<code>JB FORJ</code>		
<code>NextI:</code>		
<code>INC SI</code>	;	I=I+1
<code>CMP SI,9</code>	;	I<9时跳转
<code>JB FORI</code>		

运行结果：

(1) 统计结果显示如下

(2) 由于数是有符号数，所以最终排序结果正确，

82H < 88H < 89H < 90H < BDH < 00H < 01H < 10H < 12H < 33H



## 2.5 实验的运行数据及分析

### 1. 实验数据记录

软件环境，emu8086

Random Access Memory

0710:0000

刷新

☒ table

☐ list

0710:0000	12	88	82	89	33	90	00	10-BD	01	04	05	01	36	FE	04	\$?????.#?r  r6?
0710:0010	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	.....
0710:0020	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	.....
0710:0030	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	.....
0710:0040	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	.....
0710:0050	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	.....
0710:0060	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	.....
0710:0070	00	00	00	00	00	00	00	00-00	00	00	00	00	00	02	00	.....7.

DS:0000	12	88	82	89	33	90	00	10
DS:0008	BD	01	04	05	01	36	FE	04

### 2. 数据分析：

正数个数：04H      负数个数：05H      0 的个数：01H      SUM：FE36H

## 2.6 实验讨论及心得体会

本次实验完成了数据统计，主要用到了判断条件和 LOOP 循环，由于题中给的是有符号数，所以判断大小应该用有符号数判断大小指令（JG、JL 等），最后由于 SUM 是字，而其余数是字节存放的，也就是用 8 位寄存器进行存放，在拓展的时候用到了 CBW 指令，但是一开始对指令不熟悉，不知道只能对 AL 进行操作，结果存在了问题，最后进行修正后完成了实验，实验中也加深了对有符号数相加的掌握。



## 三、代码转换

### 3.1 实验目的要求

1. 掌握 ASCII 码转换的基本方法。
2. 学会 INT21 功能调用，掌握人机对话的设计方法。
3. 进一步熟悉 Td debug 调试环境和 Turbo Debugger 的使用。

### 3.2 实验任务

从键盘输入小写字母(最多 20 个)，以 “.” 号作为结束标志，将其变换成相应的大写字母输出在屏幕上。

### 3.3 实验代码

CRLF MACRO

```
MOV DL, 0DH
MOV AH, 02H
INT 21H
MOV DL, 0AH
MOV AH, 02H
INT 21H
ENDM
```

DATA SEGMENT

```
MES1 DB 'PLEASE INPUT THE SMALL LETTER, ENDED WITH ".":$'
MES2 DB 'THE CAPITAL LETTER IS:$'
SMALL DB 50 ;预留键盘输入缓冲区长度为 50 个
      DB 0 ;预留实际键盘输入字符数的个数
      DB 50 DUP(0)
CAPITAL DB 50 DUP(0) ;预留大写字母缓冲区长度为 50 个
DATA ENDS
```

STACK1 SEGMENT STACK

```
DB 100 DUP (0)
STACK1 ENDS
```

CODE SEGMENT

```
ASSUME CS:CODE, DS:DATA, SS:STACK1
START PROC FAR
```

```

PUSH    DS
MOV     AX, 0
PUSH    AX
MOV     AX, DATA
MOV     DS, AX
MOV     AH, 9
MOV     DX, OFFSET MES1 ;输出提示信息 MES1
INT     21H
CRLF

MOV     AH, 0AH
LEA     DX, SMALL;接收小写字符串
INT 21H
CRLF                                ;宏调用

LEA     BX,  SMALL+2
LEA     DI,  CAPITAL
MOV     CX,  20                ;最多 20 个字符
LAB:    MOV     AL,  [BX]
CMP     AL,  2EH                ;是否遇到句号.
JE      KE
SUB     AL,  20H                ;转为大写,ASCLL-20H
MOV     [DI], AL
INC     BX
INC     DI
LOOP    LAB

KE:     MOV     AL, '$'          ;大写字符串后加"$"
MOV     [DI], AL
MOV     DX, OFFSET MES2 ; 输出提示信息 MES2
MOV     AH, 9
INT     21H
CRLF

MOV     DX, OFFSET CAPITAL
MOV     AH, 9 ; 输出大写字符串
INT     21H
RET
START ENDP
CODE ENDS
END START

```

## 3.4、探究内容（选做）

代码：

CRLF MACRO

MOV DL, 0DH

MOV AH, 02H

INT 21H

MOV DL, 0AH

MOV AH, 02H

INT 21H

ENDM

DATA SEGMENT

MES1 DB 'PLEASE INPUT THE SMALL LETTER, ENDED WITH ".":\$'

MES2 DB 'THE CAPITAL LETTER IS:\$'

SMALL DB 50 ;预留键盘输入缓冲区长度为 50 个

DB 0 ;预留实际键盘输入字符数的个数

DB 50 DUP(0)

CAPITAL DB 50 DUP(0) ;预留大写字母缓冲区长度为 50 个

DATA ENDS

STACK1 SEGMENT STACK

DB 100 DUP (0)

STACK1 ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:STACK1

START PROC FAR

PUSH DS

MOV AX, 0

PUSH AX

MOV AX, DATA

MOV DS, AX

MOV AH, 9

MOV DX, OFFSET MES1 ;输出提示信息 MES1

INT 21H

CRLF

MOV AH, 0AH

LEA DX, SMALL;接收小写字符串

INT 21H

```

CRLF                                ;宏调用

LEA    BX,  SMALL+2
LEA    DI,  CAPITAL
MOV     CX,  20                    ;最多 20 个字符
LAB:   MOV     AL,  [BX]
CMP     AL,  2EH                    ;是否遇到句号.
JE      KE
CMP     AL,  'a'                    ;是否>=a
JL      NEXT
CMP     AL,  'z'                    ;是否<=z
JG      NEXT
SUB     AL,  20H                    ;转为大写,ASCLL-20H
NEXT:
MOV     [DI], AL
INC     DI
INC     BX
LOOP    LAB

KE:    MOV     AL,  '$'                ;大写字符串后加"$"
MOV     [DI], AL
MOV     DX,  OFFSET MES2 ; 输出提示信息 MES2
MOV     AH,  9
INT     21H
CRLF

MOV     DX,  OFFSET CAPITAL
MOV     AH,  9 ; 输出大写字符串
INT     21H
RET
START ENDP
CODE ENDS
END START

```

## 分析:

1) 若从键盘输入的字符非小写字母，其ASCII码减去20H后输出为无关字符，  
如何在程序中避免以上问题？

**分析:**加入判断语句，如果字符为小写则将 ASCII 减去 20H, 否则原样保留

## 核心代码

```

        LEA     BX,  SMALL+2
        LEA     DI,  CAPITAL
LAB:     MOV     CX,  20          ;最多20个字符
        MOV     AL,  [BX]
        CMP     AL,  2EH         ;是否遇到句号.
        JE      KE
        CMP     AL,  'a'        ;是否>=a
        JL      NEXT
        CMP     AL,  'z'        ;是否<=z
        JG      NEXT
        SUB     AL,  20H         ;转为大写,ASCLL-20H
NEXT:    MOV     [DI], AL
        INC     DI
        INC     BX
        LOOP    LAB

```

## 思考题运行结果

```

模拟器屏幕 (80x20 chars)
PLEASE INPUT THE SMALL LETTER, ENDED WITH ".":
asdfgh444GGGfdgfd.
THE CAPITAL LETTER IS:
ASDFGH444GGGFDGFD

```

### 3.5 实验的运行数据及分析

实验结果

软件环境，emu8086



### 3.6 实验讨论及心得体会

本次完成的是将小写字母转为大写字母的实验，这个实验比较简单，主要是将小写字母 ASCII 码值减 20H 即可，实验中的问题主要是对 INT 21H 的 0AH 功能

(3) 键盘输入

入口：AH=0AH

调用参数：DS:DX=输入缓冲区地址，首字节为缓冲区字节长度，第二字节为实际输入的字符计数

一开始掌握的不是很好。

DX 的前两个字节分别为缓冲区字节长度和实际输入的字符计数，第三个字节开始才为输入的值，由于对此掌握不到位，一开始找错了输入值，最终 debug 成功。

实验中 CRLF 采用了宏定义，优化了代码结构，使得代码更为简洁

## 四、数据块移动

### 4.1 实验目的要求

1. 进一步掌握主程序、子程序设计方法。
2. 掌握人机对话的设计方法
3. 进一步熟悉 Tddebug 调试环境和 Turbo Debugger 的使用。

### 4.2 实验任务

本实验要求将指定数据区的数据搬移到另一块内存空间中,并通过子程序调用的方法将搬移的数据显示在屏幕上。

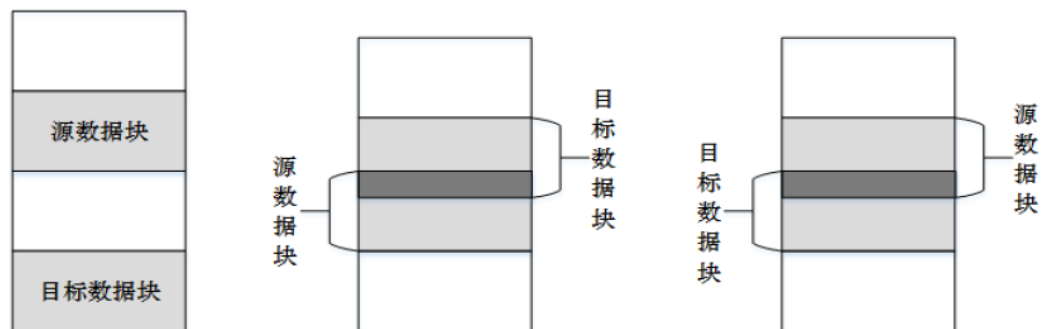


图 2-4-1 源数据块和目标数据块在存储器中的位置示意

源数据块和目标数据块在存储中的位置可能有三种情况,如图 2-4-1 所示。对于两个数据块分离的情况,数据的传送从数据块的首地址开始,或者从数据块的末地址开始均可。但对于有部分重叠的情况,则要加以分析,否则重叠部分会因搬移而遭到破坏。

所以搬移过程可以通过以下两个方式完成:

当源数据块首地址>目标块首址时,从数据块的首地址开始传送数据;

当源数据块首地址<目标块首址时,从数据块的末地址开始传送数据。

### 4.3 实验代码

```

STACK1 SEGMENT STACK
    DW 256 DUP(0)
STACK1 ENDS
DATA    SEGMENT
MES1    DB 'The data in buf2 are:',0AH,0DH,'$'
BUF1                                          DB
11H,22H,33H,44H,55H,66H,77H,88H,99H,0AAH,0BBH,0CCH,0DDH,0EEH,0FFH,00H
BUF2    DB 20H DUP(0)
DATA    ENDS

CODE    SEGMENT
    ASSUME CS:CODE,DS:DATA
START:  MOV AX,DATA
        MOV DS,AX
        MOV SI,OFFSET BUF1
        MOV DI,OFFSET BUF2
        MOV CX,16                ;循环 16 次
        CMP SI,DI
        JG  LAB1                ;SI>DI
        JL  LAB2                ;SI<DI

LAB1:                                       ;从前往后移动
        MOV BL,[SI]
        MOV [DI],BL
        INC SI
        INC DI
        LOOP LAB1
        JMP NEXT

LAB2:  ADD SI,15                    ;BUF1 末地址
        ADD DI,15                  ;BUF2 和 BUF1 等长的末地址
LAB3:  MOV BL,[SI]                ;从后往前移动
        MOV [DI],BL
        DEC SI
        DEC DI
        LOOP LAB3
NEXT:
        CALL PUTSTR
        RET

```



PUTSTR PROC NEAR

MOV DX,OFFSET MES1

MOV AH,9

INT 21H

MOV CX,16

MOV SI,OFFSET BUF2

PutNum:

MOV BL,[SI]

MOV AL,BL

SHR AL,4

CALL ToASCII

;将高 4 位转换为 ASCII 码

MOV DL,AL

MOV AH,2

INT 21H

MOV AL,BL

CALL ToASCII

;将低 4 位转换为 ASCII 码

MOV DL,AL

MOV AH,2

INT 21H

MOV DL,'H'

MOV AH,2

INT 21H

MOV DL,' '

MOV AH,2

INT 21H

INC SI

LOOP PutNum

MOV AH,4CH

INT 21H

RET

PUTSTR ENDP

ToASCII PROC NEAR

AND AL,0FH

ADD AL,'0'

CMP AL,'9'

JBE LAB5

ADD AL,7

LAB5:

RET

ToASCII ENDP

CODE ENDS

END START

## 4.4 探究内容（选做）

代码：

```

STACK1 SEGMENT STACK
    DW 256 DUP(0)
STACK1 ENDS
DATA    SEGMENT
MES1    DB 'The data in buf2 are:', 0AH, 0DH, '$'
BUF1                                          DB
11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H, 99H, 0AAH, 0BBH, 0CCH, 0DDH, 0EEH, 0FFH, 00
H
BUF2    DB 20H DUP(0)
MES2    DB 0DH, 0AH, "SortResult:$"
result  DB 0, 0, 'H', '$'
DATA    ENDS

CODE    SEGMENT
    ASSUME CS:CODE, DS:DATA
START:  MOV AX, DATA
        MOV DS, AX
        MOV SI, OFFSET BUF1
        MOV DI, OFFSET BUF2
        MOV CX, 16                ;循环 16 次
        CMP SI, DI
        JG  LAB1                  ;SI>DI
        JL  LAB2                  ;SI<DI

LAB1:                                ;从前往后移动
        MOV BL, [SI]
        MOV [DI], BL
        INC SI
        INC DI
        LOOP LAB1
        JMP NEXT

LAB2:    ADD SI, 15                ;BUF1 末地址
        ADD DI, 15                ;BUF2 和 BUF1 等长的末地址
LAB3:    MOV BL, [SI]              ;从后往前移动
        MOV [DI], BL
        DEC SI
        DEC DI

```

```

        LOOP LAB3
NEXT:
        CALL PUTSTR
        CALL SORT

        MOV  AH, 4CH
        INT  21H

        RET

SORT PROC  NEAR
        XOR  AX, AX                ;选择排序算法
        MOV  BX, OFFSET BUF1      ;I=0
        MOV  SI, 0
FORI:
        MOV  DI, SI
        INC  DI                    ;J=I+1
FORJ:
        MOV  AL, [BX+SI]
        CMP  AL, [BX+DI]          ;A[i]与A[j]比较
        JLE  NEXTJ                ;A[i]小于等于A[j]跳转
        XCHG AL, [BX+DI]          ;A[i]与A[j]交换
        MOV  [BX+SI], AL
NEXTJ:
        INC  DI                    ;J=J+1
        CMP  DI, 16               ;J<16 跳转
        JB  FORJ
NeXTI:
        INC  SI                    ;I=I+1
        CMP  SI, 15
        JB  FORI                  ;I<15 时跳转
        MOV  DX, OFFSET MES2      ;显示排序结果
        MOV  AH, 9
        INT  21H
        MOV  CL, 16
        MOV  SI, OFFSET BUF1
PutNum:
        MOV  BL, [SI]
        MOV  AL, BL
        SHR  AL, 4
        CALL ToASCII
        MOV  [result], AL
        MOV  AL, BL

```

```

    CALL ToASCII
    MOV  [result+1], AL
    MOV  DX, OFFSET result
    MOV  AH, 9
    INT  21H
    INC  SI
    LOOP PutNum
    RET
SORT ENDP
PUTSTR PROC  NEAR
    MOV  DX, OFFSET MES1
    MOV  AH, 9
    INT  21H
    MOV  CX, 16
    MOV  SI, OFFSET BUF2
PutNum1:
    MOV  BL, [SI]
    MOV  AL, BL
    SHR  AL, 4
    CALL ToASCII                      ;将高 4 位转换为 ASCII 码
    MOV  DL, AL
    MOV  AH, 2
    INT  21H
    MOV  AL, BL
    CALL ToASCII                      ;将低 4 位转换为 ASCII 码
    MOV  DL, AL
    MOV  AH, 2
    INT  21H
    MOV  DL, 'H'
    MOV  AH, 2
    INT  21H
    MOV  DL, ' '
    MOV  AH, 2
    INT  21H
    INC  SI
    LOOP PutNum1
    RET
PUTSTR ENDP
ToASCII PROC  NEAR
    AND  AL, 0FH
    ADD  AL, '0'
    CMP  AL, '9'
    JBE  LAB5
    ADD  AL, 7

```

```

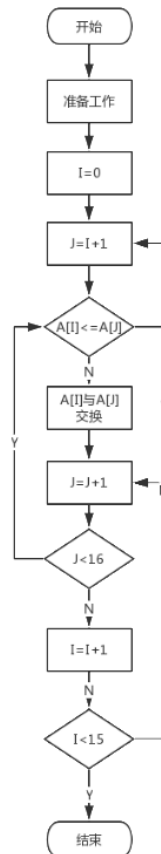
LAB5:
    RET
ToASCII ENDP
CODE ENDS
    END START

```

### 分析:

1) 利用某种排序算法，对原始数据进行排序后输出至屏幕。

算法流程图:



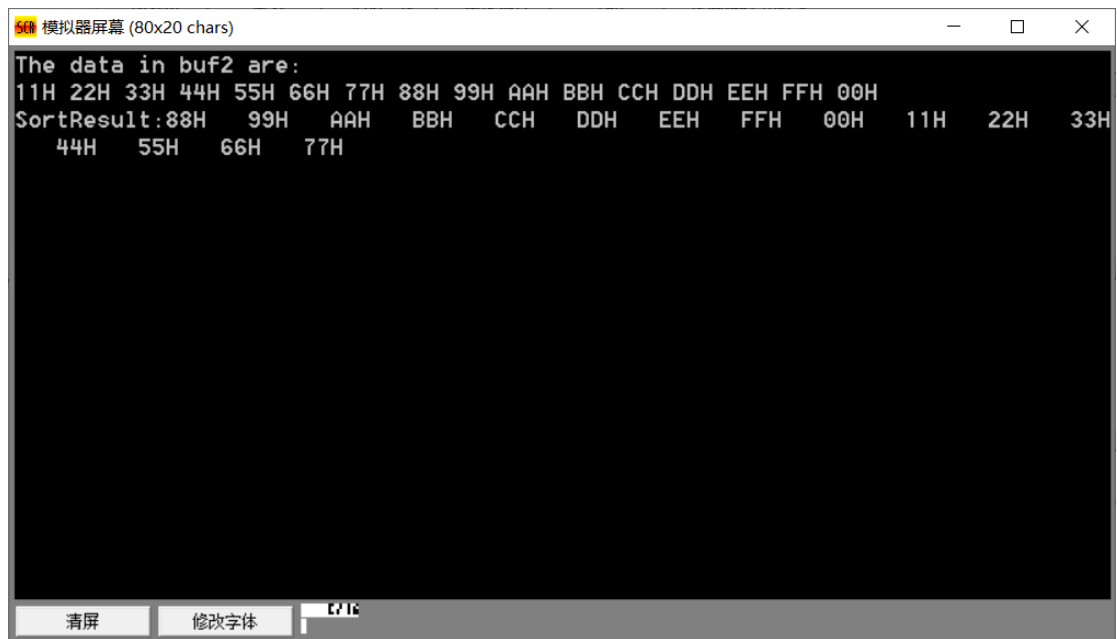
### 核心代码:

```

SORT PROC NEAR
    XOR AX,AX                ;选择排序算法
    MOV BX,OFFSET BUF1      ;I=0
    MOV SI,0
FORI:
    MOV DI,SI
    INC DI                    ;J=I+1
FORJ:
    MOV AL,[BX+SI]
    CMP AL,[BX+DI]           ;A[i]与A[j]比较
    JLE NEXTJ                ;A[i]小于等于A[j]跳转
    XCHG AL,[BX+DI]          ;A[i]与A[j]交换
    MOV [BX+SI],AL
NEXTJ:
    INC DI                    ;J=J+1
    CMP DI,16                ;J<16跳转
    JB FORJ
NEXTI:
    INC SI                    ;I=I+1
    CMP SI,15
    JB FORI                  ;I<15时跳转

```

## 思考题运行结果：

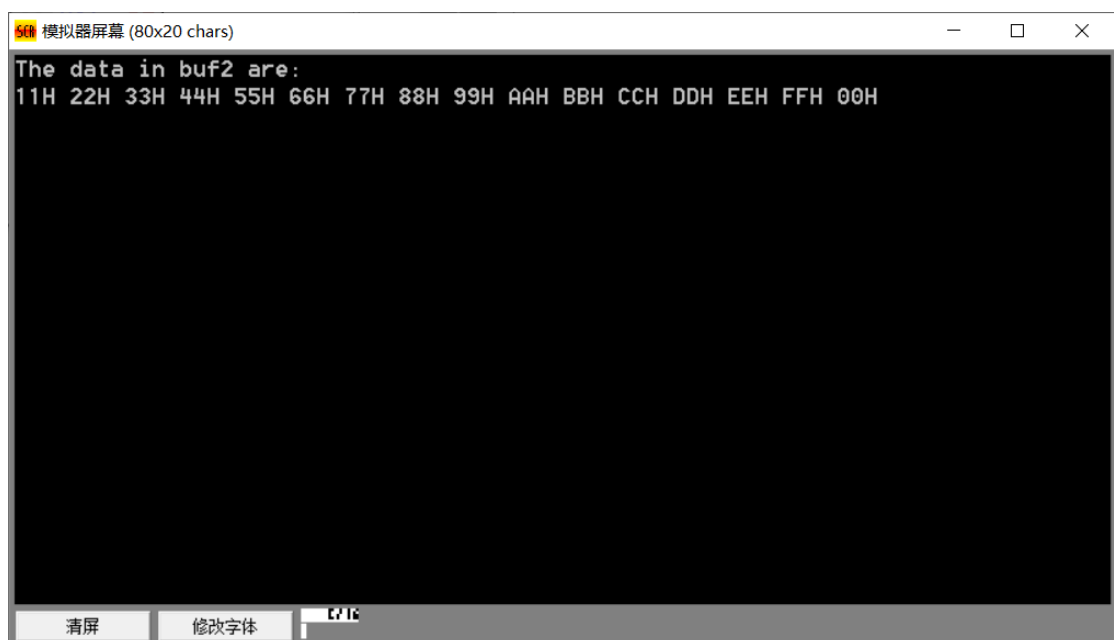


The data in buf2 are:  
11H 22H 33H 44H 55H 66H 77H 88H 99H AAH BBH CCH DDH EEH FFH 00H  
SortResult:88H 99H AAH BBH CCH DDH EEH FFH 00H 11H 22H 33H  
44H 55H 66H 77H

模拟器屏幕 (80x20 chars)

清屏 修改字体

## 4.5 实验的运行数据及分析



The data in buf2 are:  
11H 22H 33H 44H 55H 66H 77H 88H 99H AAH BBH CCH DDH EEH FFH 00H

模拟器屏幕 (80x20 chars)

清屏 修改字体

## 4.6 实验讨论及心得体会

本次实验为代码块的移动，实验难度不大，简单的将源数据块搬移到目标数据块即可，搬移前比较下首地址大小，以方便判断移动方向(从前到后、从后到前)。实验一开始采用了 MOVSB 指令，但是不知为何 emu8086 中该指令执行有误，只好换最基本的写法，多用 MOV，最终实现了功能。本次实验学会了子程序的书写以及代码块移动的方向判断。本实验同样实现了排序结果的输出。