《微机原理与接口技术》实验报告

|  |  |
| --- | --- |
| 姓 名： | 郑德凯 |
| 学 号： | U202217216 |
| 专业班级： | 软件2202班 |
| 实验名称： | 第1次实验 |
| 实验日期： | 2024.9.30 |

备注：

（1）请将报告电子版发到邮箱 MrSuInterfaceWork@163.com，文件名：姓名-学号-班级-微机原理-第X次实验.docx。

（2）提交的内容：文档，实验源代码文件（有几个任务就提交几个源代码，源码文件命名方式：Test0103.asm，第1次实验的第3个任务）

（3）邮件的主题和文档名同名。

（4）文档排版统一为小四仿宋，行间距离1.5倍行距。

（5）提交日期：下一次实验之前。

## 一、实验目的

1)熟悉汇编程序开发(和调试)环境；

2)熟悉汇编程序基本结构和分段概念；

3)熟悉8086指令系统常⻅指令；

4)熟悉8086寻址⽅式和应⽤；

## 二、实验内容

1)实验1.1 寄存器读写(和调试)实验

2)实验1.2 ⼀维数值数组处理实验

3)实验1.3 字符串处理实验

4)实验1.4 ⼆维数组处理实验

5)实验1.5 位处理实验

## 三、实验过程和疑难问题解决

### 3.1 寄存器读写(和调试)实验

#### 3.1.1 源代码

CODE SEGMENT

ASSUME CS:CODE, DS:CODE ;

START:

; 给 AX、BX、CX、DX 赋初始值

MOV AX, 1234H ; AX = 1234H

MOV BX, 5678H ; BX = 5678H

MOV CX, 9ABCH ; CX = 9ABCH

MOV DX, 4321H ; DX = 4321H

; 交换 AX 和 BX 的值

PUSH AX

MOV AX, BX

POP BX

; 交换 CX 和 DX 的值

PUSH CX

MOV CX, DX

POP DX

; 交换 AX 的高 8 位与 DX 的低 8 位的值

XCHG AH, DL

; 程序结束

MOV AX, 4C00h ; 正常退出 DOS

INT 21h

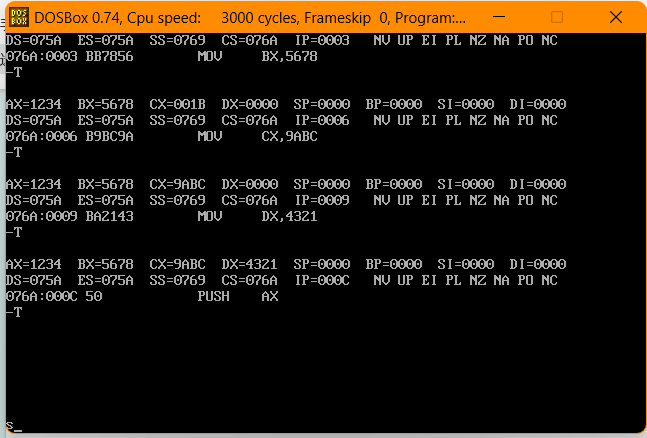
CODE ENDS

END START

#### 3.1.2 观察到的现象

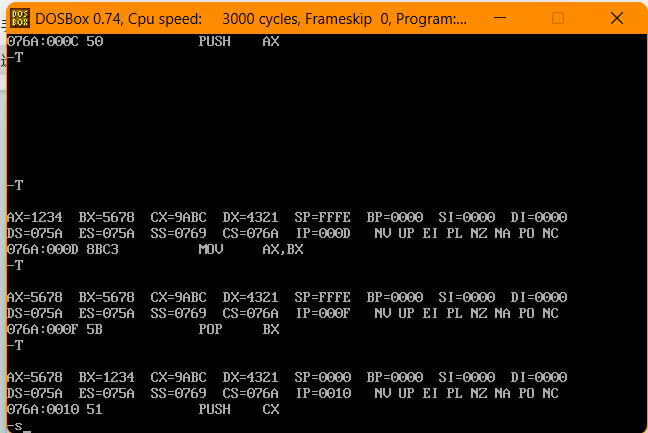
在进入debug调试模式后，通过-T一步步来对代码进行运行

在图3.1.2.1中，可以看到首先对AX，BX，CX，DX进行赋值



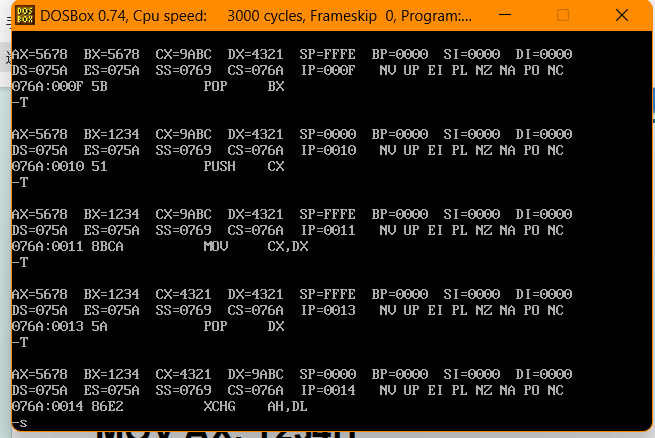
***图3.1.2.1***

在图3.1.2.2中，可以看到AX和BX发生了调换



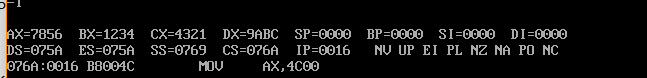
***图3.1.2.2***

在图3.1.2.3中，CX和DX发生调换



***图3.1.2.3***

最后，在图3.1.2.4中，AX的高八位和低八位发生调换



***图3.1.2.4***

### 3.2 ⼀维数值数组处理实验

#### 3.2.1 源代码

DATA SEGMENT

NUMBERS DW 0001H, 0002H, 0003H, 0004H, 0005H, 0006H, 0007H, 0008H

DW 0009H, 000AH, 000BH, 000CH, 000DH, 000EH, 000FH, 0010H

SUM DW 0FFFFH ; 用于存放和的变量

DATA ENDS

STACK SEGMENT

DW 16 DUP(0) ; 创建堆栈，大小为16字

STACK ENDS

CODE SEGMENT

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

START:

; 初始化数据段

MOV AX, DATA

MOV DS, AX

; 初始化堆栈段

MOV AX, STACK

MOV SS, AX

MOV SP, 20H ; 设置栈指针

; 求和部分

MOV AX, 0 ; 清零 AX，准备求和

MOV BX, 0 ; BX 指向 NUMBERS 数组的起始地址

MOV CX, 16 ; 计数器设置为 16

S1:

ADD AX, [NUMBERS + BX] ; 将 NUMBERS 中的值加到 AX

PUSH [NUMBERS + BX] ; 将当前值压入栈中

ADD BX, 2 ; BX 增加 2，指向下一个字

LOOP S1 ; 循环，直到 CX 为 0

; 保存总和

MOV SUM, AX ; 将 AX 的值存入 SUM

; 逆序部分

MOV BX, 0 ; BX 重置为 0

MOV CX, 16 ; 计数器重新设置为 16

S2:

POP AX ; 从栈中弹出值到 AX

MOV [NUMBERS + BX], AX ; 将 AX 的值存储回 NUMBERS 数组

ADD BX, 2 ; BX 增加 2，指向下一个字

LOOP S2 ; 循环，直到 CX 为 0

; 程序结束

MOV AX, 4C00h ; 正常退出 DOS

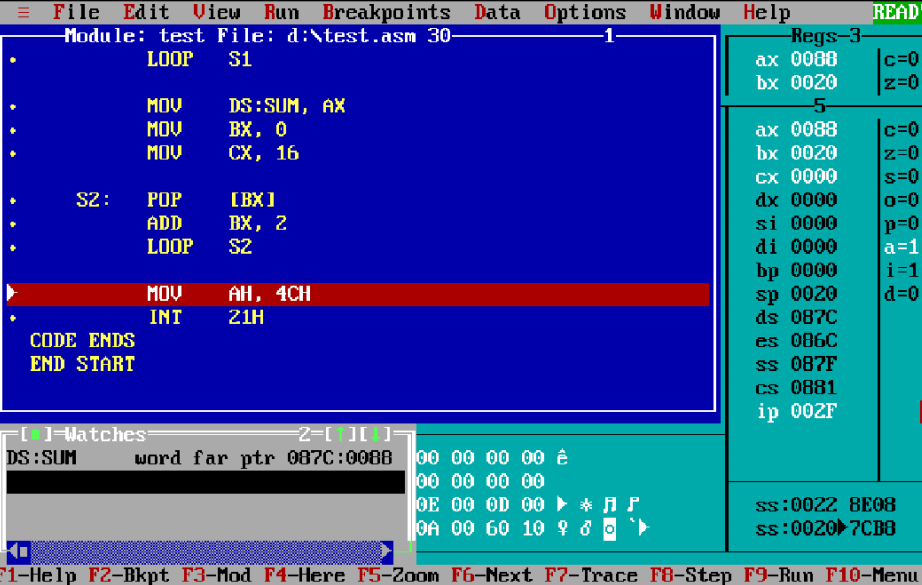
INT 21h

CODE ENDS

END START

#### 3.2.2 观察到的现象

按F2设置断点之后，按F9执⾏整个程序，会⾃动暂停，此时再次通过监视窗⼝查看，可以发现DS段的数据均逆序存储；在Watches窗⼝添加 SUM值，可以查看到SUM的值为0088H,符合预期结果



***图3.2.2.1***

### 3.3 字符串处理实验

#### 3.3.1 源代码

DATA SEGMENT

MaxLength DB 100

ActualLength DB ?

STRING DB 100 DUP(?)

ChangeRow DB 0DH, 0AH, '$'

UpperCase DB 100 DUP(?)

DATA ENDS

STACK SEGMENT

DW 128 DUP(0)

STACK ENDS

CODE SEGMENT

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

MAIN:

MOV AX, DATA

MOV DS, AX

INPUT:

MOV DX, OFFSET MaxLength

MOV AH, 10

INT 21H

; 转换为大写

MOV BX, OFFSET STRING

MOV DI, OFFSET UpperCase

MOV CH, 0

MOV CL, ActualLength ; CX = ActualLength

TO\_UPPER:

MOV AL, [BX]

CALL TurnCapital

MOV [DI], AL

INC BX

INC DI

LOOP TO\_UPPER

; 添加终止字符到 UpperCase

MOV BYTE PTR [DI], 24H

OUTPUT\_UPPER:

MOV DX, OFFSET ChangeRow

MOV AH, 9

INT 21H

MOV DX, OFFSET UpperCase

MOV AH, 9

INT 21H

; 转换为小写, CX = ActualLength

MOV BX, OFFSET STRING

MOV CH, 0

MOV CL, ActualLength

TO\_LOWER:

MOV AL, [BX]

CALL TurnSmall

MOV [BX], AL

INC BX

LOOP TO\_LOWER

; 添加终止字符到 STRING

MOV BYTE PTR [BX], 24H

OUTPUT\_LOWER:

MOV DX, OFFSET ChangeRow

MOV AH, 9

INT 21H

MOV DX, OFFSET STRING

MOV AH, 9

INT 21H

END\_MAIN:

MOV AH, 4CH

INT 21H

TurnCapital PROC

; AL中的字符转为大写

CMP AL, 'a'

JL DONE\_CAP

CMP AL, 'z'

JG DONE\_CAP

SUB AL, 20H ; 大写 = 小写 - 20H

DONE\_CAP:

RET

TurnCapital ENDP

TurnSmall PROC

; AL中的字符作大小写转换

CMP AL, 'A'

JL DONE\_SMALL

CMP AL, 'Z'

JG DONE\_SMALL

ADD AL, 20H

DONE\_SMALL:

RET

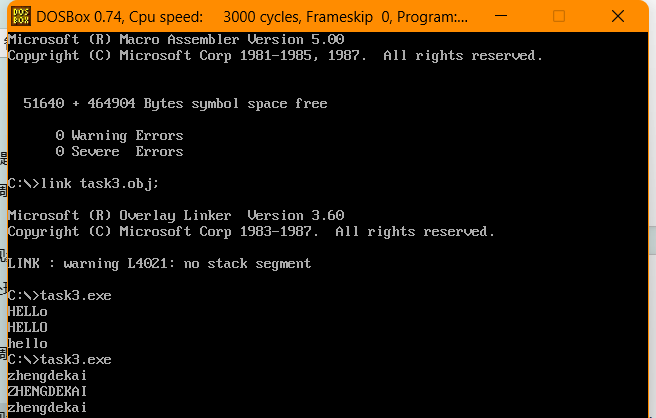
TurnSmall ENDP

CODE ENDS

END MAIN

#### 3.3.2 观察到的现象

在代码运行后，可以看到，输出了大写和小写



### 3.4 位处理实验

#### 3.4.1 源代码

DATA SEGMENT

COUNT DW 8 ; 数组中的元素数量

NUMBERS DW 1234H, 5678H, 9ABCH, 0DEF0H, 1111H, 2222H, 3333H, 4444H ; 数字数组

RESULT DW ? ; 占位符，用于存放符合条件的数的个数

DATA ENDS

STACK SEGMENT

DW 16 DUP(0) ; 栈初始化，分配16个字

STACK ENDS

CODE SEGMENT

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

START:

MOV AX, DATA ; 初始化数据段

MOV DS, AX

MOV AX, STACK ; 初始化栈段

MOV SS, AX

MOV SP, 20H ; 设置堆栈指针

; 初始化 BX 指向数组的第一个元素

MOV BX, OFFSET NUMBERS

ADD BX, 2 ; 跳过 COUNT，指向第一个数值

; 将 COUNT 的值加载到 CX 中，用于计数循环

MOV CX, [OFFSET NUMBERS]

; 初始化 DI 指向 RESULT，用于存储符合条件的数的个数

LEA DI, RESULT

XOR DX, DX ; DX用于存储符合条件的数的个数，初始化为0

PROCESS:

; 将当前数字加载到 AX 中

MOV AX, [BX]

; 检查 D3 位是否为1

TEST AX, 0008H ; 测试 D3 位（位3）

JZ SKIP ; 如果D3位不为1，跳过

; D3位为1，增加计数

INC DX

; 取反 D3 位

XOR AX, 0008H ; 取反 D3 位

MOV [BX], AX ; 将取反后的数存回原数组位置

SKIP:

; 移动到数组的下一个数字

ADD BX, 2

LOOP PROCESS ; 如果 CX 未减为 0，则继续循环

; 将计数结果存入 RESULT

MOV [DI], DX

; 程序结束

MOV AH, 4CH

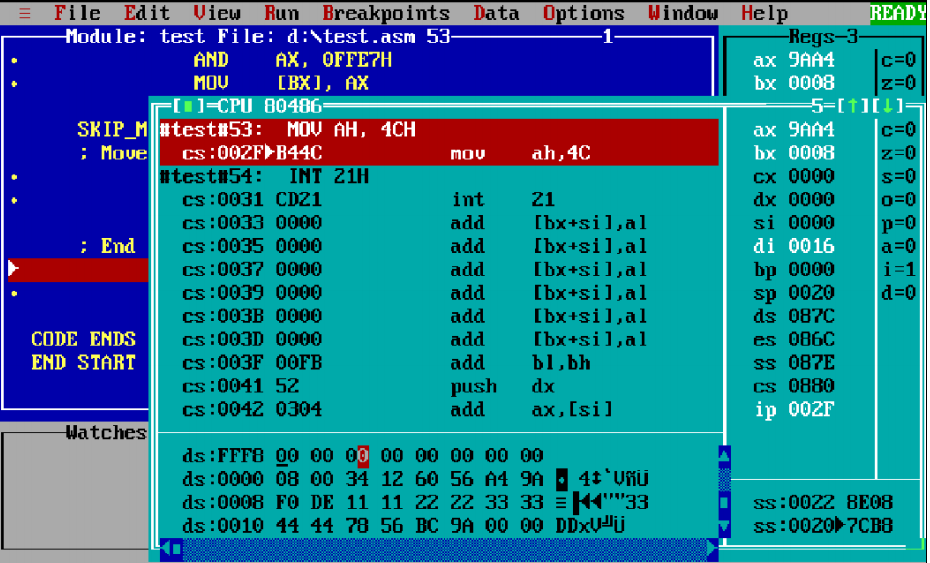
INT 21H

CODE ENDS

END START

#### 3.4.2 实验中的现象

运⾏结束后，可以发现D3,D4位为0的数据已经完成了修改，并且复制到了代码段的最后（5678H、9ABCH）



***图3.4.2.1***

## 四、收获

通过微机原理实验一，我更加深入地理解了汇编程序的基本结构和分段机制，尤其是8086指令系统中的常用指令在实际应用中的使用方法。同时，我也掌握了8086的多种寻址方式，特别是在处理一维数组、字符串、二维数组以及位操作方面的应用。这次实验不仅加深了我对汇编语言编程的理解，还为后续的学习奠定了坚实的基础。