

廈門大學



《汇编语言》实验报告

(七)

批注 [a1]: 第几次实验

姓 名 宋浩元

学 号 37220232203808

学 院 信息学院

专 业 软件工程

2024 年 12 月

1 实验目的

- (1) 熟悉和掌握高级汇编程序设计的指令与设计方法
- (2) 进一步熟练运用各种程序设计基本结构。

批注 [a2]: 根据实验有关文档, 了解本次实验的目的, 需要达到的目标

2 实验环境

Windows11 环境下的 masm 与 DOSBOX;

批注 [a3]: 编译环境, 机器配置情况

3 实验内容

- (1) 针对下述条件汇编代码, 将之转换为正常的汇编指令, 并采用 debug 进行调试, 结合不同的数据, 验证结果。
`.if(x==5) && (ax!=bx)`
`inc ax .endif`

批注 [a4]: 根据实验有关文档, 提炼主要的实验内容, 列表展示

- (2) 实现下述程序段, 比较汇编程序生成的代码序列有何不同:

```
.while ax!=10  
  
    mov [bx], ax  
    inc bx  
  
    inc ax  
  
.endw
```

```
.repeat mov  
[bx], ax  
bx  
inc bx  
inc ax  
until ax==10
```

- (3) 利用重复汇编方法定义一个数据区, 数据区有 100 个双字, 每个双字的高字

部分依次是 2,4,6,...,200，低字部分都是 0，给出代码截屏并展示内存数据区存储情况。

(4) 利用宏结构完成以下功能：如果变量 `byteX` 中的数据大于 5 时，指令“`ADD AX, AX`”将汇编 10 次，否则什么也不汇编。

(5) 定义一个宏 `LOGICAL`，代表 4 条逻辑运算指令：`AND`、`OR`、`XOR`、`TEST`，注意：需要利用 3 个形式参数，并给出一个宏调用和展开的例子，编写代码，并调试，展示结果。

(6) 定义一个宏 `MOVESTR strN`、`DSTR`、`SSTR`，将 `strN` 个字符从一个字符区 `SSTR` 传送到另一个字符区 `DSTR`，编写代码，并给出具体调用实例。

(7) 将例 4.7 的大写字母转换为小写字母用宏完成

```

; 数据段
string db 'Hello, Everybody!', 0      ; 可以任意给定一个字符串
; 代码段
mov     bx, offset string
again:  mov     al, [bx]                ; 取一个字符
        or      al, al                 ; 是否为结尾符0
        jz      done                 ; 是，退出循环
        cmp     al, 'a'               ; 是否为大写字母 A~Z
        jb      next
        cmp     al, 'z'
        ja      next
        or      al, 20h                ; 是，转换为小写字母（使 D5=1）
        mov     [bx], al              ; 仍保存在原位置
next:   inc     bx
        jmp     again                 ; 继续循环
done:
```

4 实验具体实现

(1)

```

DATAS SEGMENT
    X DW 5
    AX_VAR DW 10
    BX_VAR DW 10
DATAS ENDS

STACKS SEGMENT
    DB 128 DUP(0)
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES, DS:DATAS, SS:STACKS

START:
    ; 初始化段寄存器
    MOV AX, DATAS
    MOV DS, AX
    MOV AX, STACKS
    MOV SS, AX
    MOV SP, 128

    ; 实现条件逻辑 .if(x == 5) && (ax != bx)
    MOV AX, X          ; 将 X 的值加载到 AX
    CMP AX, 5          ; 比较 X 是否等于 5
    JNE ed             ; 如果不等于 5, 跳转到 ed

    MOV AX, AX_VAR
    MOV BX, BX_VAR
    CMP AX, BX         ; 比较 AX 和 BX 是否相等
    JE ed              ; 如果相等, 跳转到 ed

    INC AX              ; AX 加 1

ed:
    ; 程序结束
    MOV AH, 4CH        ; DOS 功能号 4CH: 程序结束
    INT 21H            ; 调用中断 21H

CODES ENDS
    END START

```

在第一个例子中，x 等于五且 ax 等于 bx 所以最后直接跳到 ed。

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
DS=0760 ES=0760 SS=076F CS=0779 IP=0000 NU UP EI PL NZ NA PO NC
0779:0000 B87007 MOV AX,0770
-t
AX=0770 BX=0000 CX=00BB DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=0779 IP=0003 NU UP EI PL NZ NA PO NC
0779:0003 B8D8 MOV DS,AX
-t
AX=0770 BX=0000 CX=00BB DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0779 IP=0005 NU UP EI PL NZ NA PO NC
0779:0005 B87107 MOV AX,0771
-t
AX=0771 BX=0000 CX=00BB DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0779 IP=0008 NU UP EI PL NZ NA PO NC
0779:0008 B8D0 MOV SS,AX
-tt
^ Error
-t
AX=0771 BX=0000 CX=00BB DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=000D NU UP EI PL NZ NA PO NC
0779:000D A10000 MOV AX,[0000] DS:0000=0005
t
AX=0005 BX=0000 CX=00BB DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0010 NU UP EI PL NZ NA PO NC
0779:0010 83F805 CMP AX,+05
-t
AX=0005 BX=0000 CX=00BB DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0013 NU UP EI PL ZR NA PE NC
0779:0013 750F JNZ 0024
t
AX=0005 BX=0000 CX=00BB DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0015 NU UP EI PL ZR NA PE NC
0779:0015 A10200 MOV AX,[0002] DS:0002=000A
-t
AX=000A BX=0000 CX=00BB DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0018 NU UP EI PL ZR NA PE NC
0779:0018 8B1E0400 MOV BX,[0004] DS:0004=000A
```

```

DS=0770 ES=0760 SS=0771 CS=0779 IP=0013  NU UP EI PL ZR NA PE NC
0779:0015 A10200      MOV     AX,I00021      DS:0002=000A
-t
AX=000A BX=0000 CX=00B8 DX=0000 SP=00B0 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0018  NU UP EI PL ZR NA PE NC
0779:0018 8B1E0400      MOV     BX,I00041      DS:0004=000A
-
t
AX=000A BX=000A CX=00B8 DX=0000 SP=00B0 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=001C  NU UP EI PL ZR NA PE NC
0779:001C 3BC3          CMP     AX,BX
-t
AX=000A BX=000A CX=00B8 DX=0000 SP=00B0 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=001E  NU UP EI PL ZR NA PE NC
0779:001E 7404          JZ      0024
-t
AX=000A BX=000A CX=00B8 DX=0000 SP=00B0 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0024  NU UP EI PL ZR NA PE NC
0779:0024 B44C          MOV     AH,4C

```

```

01 DATAS SEGMENT
02     X DW 5
03     AX_VAR DW 10
04     BX_VAR DW 20
05 DATAS ENDS
06
07 STACKS SEGMENT
08     DB 128 DUP(0)
09 STACKS ENDS
10
11 CODES SEGMENT
12     ASSUME CS:CODES, DS:DATAS, SS:STACKS
13
14 START:
15     ; 初始化段寄存器
16     MOV AX, DATAS
17     MOV DS, AX
18     MOV AX, STACKS
19     MOV SS, AX
20     MOV SP, 128
21
22     ; 实现条件逻辑 .if(x == 5) && (ax != bx)
23     MOV AX, X           ; 将 X 的值加载到 AX
24     CMP AX, 5           ; 比较 X 是否等于 5
25     JNE ed             ; 如果不等于 5, 跳转到 ed
26
27     MOV AX, AX_VAR
28     MOV BX, BX_VAR
29     CMP AX, BX          ; 比较 AX 和 BX 是否相等
30     JE ed              ; 如果相等, 跳转到 ed
31
32     INC AX              ; AX 加 1
33
34 ed:
35     ; 程序结束
36     MOV AH, 4CH         ; DOS 功能号 4CH: 程序结束
37     INT 21H            ; 调用中断 21H
38
39 CODES ENDS
40     END START
41

```

成功实现给 ax 加一。

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
-r
AX=FFFF BX=0000 CX=00B8 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=0779 IP=0000 NU UP EI PL NZ NA PO NC
0779:0000 B87007 MOV AX,0770
-t
AX=0770 BX=0000 CX=00B8 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=0779 IP=0003 NU UP EI PL NZ NA PO NC
0779:0003 8ED8 MOV DS,AX
-t
AX=0770 BX=0000 CX=00B8 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0779 IP=0005 NU UP EI PL NZ NA PO NC
0779:0005 B87107 MOV AX,0771
-t
AX=0771 BX=0000 CX=00B8 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0779 IP=0008 NU UP EI PL NZ NA PO NC
0779:0008 8ED0 MOV SS,AX
-t
AX=0771 BX=0000 CX=00B8 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=000D NU UP EI PL NZ NA PO NC
0779:000D A10000 MOV AX,[0000] DS:0000=0005
-t
```

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
AX=0771 BX=0000 CX=00B8 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=000D NU UP EI PL NZ NA PO NC
0779:000D A10000 MOV AX,[0000] DS:0000=0005
-t
AX=0005 BX=0000 CX=00B8 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0010 NU UP EI PL NZ NA PO NC
0779:0010 83F805 CMP AX,+05
-t
AX=0005 BX=0000 CX=00B8 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0013 NU UP EI PL ZR NA PE NC
0779:0013 750F JNZ 0024
-t
AX=0005 BX=0000 CX=00B8 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0015 NU UP EI PL ZR NA PE NC
0779:0015 A10200 MOV AX,[0002] DS:0002=000A
-t
AX=000A BX=0000 CX=00B8 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0018 NU UP EI PL ZR NA PE NC
0779:0018 8B1E0400 MOV BX,[0004] DS:0004=000A
-t
```



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
AX=000A BX=0000 CX=00B5 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0018 NU UP EI PL ZR NA PE NC
0779:0018 8B1E0400 MOV BX,[0004] DS:0004=0014
-t
AX=000A BX=0014 CX=00B5 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=001C NU UP EI PL ZR NA PE NC
0779:001C 3BC3 CMP AX,BX
-t
AX=000A BX=0014 CX=00B5 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=001E NU UP EI NG NZ NA PE CY
0779:001E 7401 JZ 0021
-t
AX=000A BX=0014 CX=00B5 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0020 NU UP EI NG NZ NA PE CY
0779:0020 40 INC AX
-t
AX=000B BX=0014 CX=00B5 DX=0000 SP=0080 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=0771 CS=0779 IP=0021 NU UP EI PL NZ NA PO CY
0779:0021 B44C MOV AH,4C
```

(2)

先完成代码:

```
DATA SEGMENT

DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

START:
    MOV AX, DATA      ; 初始化数据段寄存器
    MOV DS, AX
    mov ax, 0
    .while ax != 10
        MOV [BX], AX
        INC BX
        INC BX
        INC AX
    .endw
    MOV AH, 4CH        ; 正常结束程序
    INT 21H
CODE ENDS
END START
```

```

DATA SEGMENT

DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

START:
    MOV AX, DATA      ; 初始化数据段寄存器
    MOV DS, AX
    ; .repeat 示例
    mov ax, 0
    .repeat
        MOV [BX], AX
        INC BX
        INC BX
        INC AX
    .until ax == 10

    MOV AH, 4CH        ; 正常结束程序
    INT 21H
CODE ENDS
END START

```

然后观察两者的 lst 文件：

```
0000          DATA SEGMENT

0000          DATA ENDS

0000          CODE SEGMENT
          ASSUME CS:CODE, DS:DATA

0000          START:
0000 B8 ---- R      MOV AX, DATA    ; 初始化数据段寄存器
0003 8E D8          MOV DS, AX
0005 B8 0000        mov ax, 0
                  .while ax != 10
000A 89 07          MOV [BX], AX
000C 43             INC BX
000D 43             INC BX
000E 40             INC AX
                  .endw
0014 B4 4C          MOV AH, 4CH    ; 正常结束程序
0016 CD 21          INT 21H
0018              CODE ENDS
          END START
```

Segments and Groups:

N a m e	Size	Length	Align	Combine	Class
CODE	16 Bit	0018	Para		Private
DATA	16 Bit	0000	Para		Private

Symbols:

N a m e	Type	Value	Attr
START	L Near	0000	CODE

0 Warnings
0 Errors

```
0000          DATA SEGMENT
0000          DATA ENDS
0000          CODE SEGMENT
          ASSUME CS:CODE, DS:DATA
0000          START:
0000 B8 ---- R    MOV AX, DATA    ; 初始化数据段寄存器
0003 8E D8        MOV DS, AX
          ; .repeat 示例
0005 B8 0000      mov ax, 0
          .repeat
0008 89 07        MOV [BX], AX
000A 43          INC BX
000B 43          INC BX
000C 40          INC AX
          .until ax == 10
0012 B4 4C        MOV AH, 4CH    ; 正常结束程序
0014 CD 21        INT 21H
0016          CODE ENDS
          END START
```

Segments and Groups:

N a m e	Size	Length	Align	Combine	Class
CODE	16 Bit	0016	Para		Private
DATA	16 Bit	0000	Para		Private

Symbols:

N a m e	Type	Value	Attr
START	L Near	0000	CODE

0 Warnings

这两个代码序列的主要区别在于程序的执行，.while 的代码长一些，因为有两个指令，一个 jmp 指令和 je 指令，而.repeat 的代码在开始没有这些判断。

(3)

```
DATA SEGMENT
    DBL_WORDS LABEL DWORD
    REPT 100
        DW ((($ - DBL_WORDS) / 4 + 1) * 2 ; 高字部分：依次生成 2, 4, 6, ..., 200
        DW 0 ; 低字部分：固定为 0
    ENDM
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

START:
    MOV AX, DATA
    MOV DS, AX

    ; 程序结束
    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

(4)

```

CHECK MACRO VAR
    CMP VAR, 5      ; 比较 VAR 和 5
    JBE SKIP        ; 如果 VAR <= 5, 跳过汇编 ADD 指令
    REPT 10          ; 重复执行 10 次
        ADD AX, AX   ; 指令 ADD AX, AX
    ENDM
SKIP:
ENDM

DATA SEGMENT
    byteX DB 6
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

START:
    MOV AX, DATA
    MOV DS, AX
    MOV AL, byteX
    CHECK AL
    ; 程序结束
    MOV AH, 4CH
    INT 21H

CODE ENDS
END START

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...

Address	Instruction
0771:0000	B87007 MOV AX,0770
0771:0003	BED8 MOV DS,AX
0771:0005	A00000 MOV AL,[0000]
0771:0008	3C05 CMP AL,05
0771:000A	7614 JBE 0020
0771:000C	03C0 ADD AX,AX
0771:000E	03C0 ADD AX,AX
0771:0010	03C0 ADD AX,AX
0771:0012	03C0 ADD AX,AX
0771:0014	03C0 ADD AX,AX
0771:0016	03C0 ADD AX,AX
0771:0018	03C0 ADD AX,AX
0771:001A	03C0 ADD AX,AX
0771:001C	03C0 ADD AX,AX
0771:001E	03C0 ADD AX,AX

当 x 为 4 时:

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
AX=0770 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0760 ES=0760 SS=076F CS=0771 IP=0003  NU UP EI PL NZ NA PO NC
0771:0003 8ED8      MOV     DS,AX
-t

AX=0770 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0005  NU UP EI PL NZ NA PO NC
0771:0005 A00000     MOV     AL,[0000]          DS:0000=04
-t

AX=0704 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0008  NU UP EI PL NZ NA PO NC
0771:0008 3C05      CMP     AL,05
-t

AX=0704 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=000A  NU UP EI NG NZ AC PE CY
0771:000A 7614      JBE     0020
-t

AX=0704 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0020  NU UP EI NG NZ AC PE CY
0771:0020 B44C      MOV     AH,4C

```

当 x 为 6 时:

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
AX=0706 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=000A  NU UP EI PL NZ NA PO NC
0771:000A 7614      JBE     0020
-t

AX=0706 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=000C  NU UP EI PL NZ NA PO NC
0771:000C 03C0      ADD     AX,AX
-t

AX=0E0C BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=000E  NU UP EI PL NZ NA PE NC
0771:000E 03C0      ADD     AX,AX
-t

AX=1C18 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0010  NU UP EI PL NZ AC PE NC
0771:0010 03C0      ADD     AX,AX
-t

AX=3B30 BX=0000 CX=0034 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0012  NU UP EI PL NZ AC PE NC
0771:0012 03C0      ADD     AX,AX
-t

```

(5)

```

01 LOGICAL MACRO OP, REG1, REG2
02     &OP &REG1, &REG2
03 ENDM
04
05 DATA SEGMENT
06     VALUE1 DW 0F0Fh
07     VALUE2 DW 0A0A0h
08     RESULT DW ?
09 DATA ENDS
10
11 CODE SEGMENT
12     ASSUME CS:CODE, DS:DATA
13
14 START:
15     MOV AX, DATA
16     MOV DS, AX
17
18     MOV AX, VALUE1      ; 加载第一个值到 AX
19     MOV BX, VALUE2      ; 加载第二个值到 BX
20
21     ; 使用 LOGICAL 宏
22     LOGICAL AND, AX, BX ; 执行 AX = AX AND BX
23     MOV RESULT, AX      ; 将结果存储到 RESULT
24
25     LOGICAL OR, AX, BX  ; AX = AX OR BX
26     LOGICAL XOR, AX, BX ; AX = AX XOR BX
27     LOGICAL TEST, AX, BX
28
29     ; 结束程序
30     MOV AH, 4CH
31     INT 21H
32
33 CODE ENDS
34     END START
35

```



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
-u
0771:0000 B87007 MOV AX,0770
0771:0003 BED0 MOV DS,AX
0771:0005 A10000 MOV AX,[0000]
0771:0008 BB1E0200 MOV BX,[0002]
0771:000C Z3C3 AND AX,BX
0771:000E A30400 MOV [0004],AX
0771:0011 0BC3 OR AX,BX
0771:0013 33C3 XOR AX,BX
0771:0015 B5C3 TEST AX,BX
0771:0017 B44C MOV AH,4C
0771:0019 CB21 INT 21
0771:001B 0000 ADD [BX*SI],AL
0771:001D 0000 ADD [BX*SI],AL
0771:001F 0000 ADD [BX*SI],AL
-;
```

AND 操作: AX = 0F0Fh AND 0A0Ah = 000Ah

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
-t
AX=0F0F BX=0000 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0008 NU UP EI PL NZ NA PO NC
0771:0008 BB1E0200 MOV BX,[0002] DS:0002=A0A0
-t
AX=0F0F BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=000C NU UP EI PL NZ NA PO NC
0771:000C Z3C3 AND AX,BX
-t
AX=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=000E NU UP EI PL ZR NA PE NC
0771:000E A30400 MOV [0004],AX DS:0004=0000
-d 0000
0770:0000 0F 0F A0 A0 00 00 00 00-00 00 00 00 00 00 00 00 .....
0770:0010 BB 70 07 BE D8 A1 00 00-8B 1E 02 00 Z3 C3 A3 04 .p.....#...
0770:0020 00 0B C3 33 C3 B5 C3 B4-4C CD Z1 00 00 00 00 00 ...3...L.!....
0770:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

OR 操作: AX = 0000h OR 0A0Ah = 0A0Ah

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
-t
AX=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=000E NU UP EI PL ZR NA PE NC
0771:000E A30400 MOV [0004],AX DS:0004=0000
-d 0000
0770:0000 0F 0F A0 A0 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0010 B8 70 07 BE DB A1 00 00-BB 1E 02 00 23 C3 A3 04 .p.....#...
0770:0020 00 0B C3 33 C3 B5 C3 B4-4C CD 21 00 00 00 00 00 ...3....L.!...
0770:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-t
AX=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0011 NU UP EI PL ZR NA PE NC
0771:0011 0BC3 OR AX,BX
-t
AX=A0A0 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0013 NU UP EI NG NZ NA PE NC
0771:0013 33C3 XOR AX,BX
;
```

XOR 操作: AX = 0A0A or 0A0Ah = 0000h

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
-d 0000
0770:0000 0F 0F A0 A0 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0010 B8 70 07 BE DB A1 00 00-BB 1E 02 00 23 C3 A3 04 .p.....#...
0770:0020 00 0B C3 33 C3 B5 C3 B4-4C CD 21 00 00 00 00 00 ...3....L.!...
0770:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0770:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-t
AX=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0011 NU UP EI PL ZR NA PE NC
0771:0011 0BC3 OR AX,BX
-t
AX=A0A0 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0013 NU UP EI NG NZ NA PE NC
0771:0013 33C3 XOR AX,BX
-t
AX=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0770 ES=0760 SS=076F CS=0771 IP=0015 NU UP EI PL ZR NA PE NC
0771:0015 85C3 TEST AX,BX
;
```

TEST 操作: AX = 0000h

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
770:0040 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
770:0050 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
770:0060 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
770:0070 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
t
X=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
S=0770 ES=0760 SS=076F CS=0771 IP=0011 NU UP EI PL ZR NA PE NC
771:0011 0BC3 OR AX,BX
t
X=A0A0 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
S=0770 ES=0760 SS=076F CS=0771 IP=0013 NU UP EI NG NZ NA PE NC
771:0013 33C3 XOR AX,BX
t
X=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
S=0770 ES=0760 SS=076F CS=0771 IP=0015 NU UP EI PL ZR NA PE NC
771:0015 85C3 TEST AX,BX
t
X=0000 BX=A0A0 CX=002B DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
S=0770 ES=0760 SS=076F CS=0771 IP=0017 NU UP EI PL ZR NA PE NC
771:0017 B44C MOV AH,4C
;
```

(6)

```

MOVESR MACRO strN, dstr, sstr
    MOV CX, strN          ; 设置字符传送的计数
    LEA SI, sstr           ; 将源字符串地址加载到 SI
    LEA DI, dstr          ; 将目标字符串地址加载到 DI
    CLD                  ; 确保方向标志位清零
    REP MOUSB             ; 使用 REP MOUSB 指令重复传送字符
ENDM

DATAS SEGMENT
    SSTR DB 'HELLO WORLD!'
    DSTR DB 20 DUP(?)
DATAS ENDS

STACKS SEGMENT
    DB 128 DUP(0)
STACKS ENDS

CODES SEGMENT
    ASSUME CS:CODES, DS:DATAS, SS:STACKS

START:

    MOV AX, DATAS
    MOV DS, AX
    MOV AX, DATAS
    MOV ES, AX

    MOVESR 12, DSTR, SSTR ; 将 SSTR 中的 12 个字符传送到 DSTR

    ; 程序结束
    MOV AH, 4CH
    INT 21H

CODES ENDS
END START

```

可以看到已经实现了 strcpy

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
AX=0770 BX=0000 CX=0002 DX=0000 SP=0000 BP=0000 SI=0000 DI=0016
DS=0770 ES=0770 SS=076F CS=077A IP=0016 NU UP EI PL NZ NA PO NC
077A:0016 F3 REPZ
077A:0017 A4 MOUSB
t

AX=0770 BX=0000 CX=0001 DX=0000 SP=0000 BP=0000 SI=000B DI=0017
DS=0770 ES=0770 SS=076F CS=077A IP=0016 NU UP EI PL NZ NA PO NC
077A:0016 F3 REPZ
077A:0017 A4 MOUSB
t

AX=0770 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=000C DI=0018
DS=0770 ES=0770 SS=076F CS=077A IP=0018 NU UP EI PL NZ NA PO NC
077A:0018 B44C MOV AH,4C
d 0000
0770:0000 4B 45 4C 4C 4F 20 57 4F 52 4C 44 21 4B 45 4C 4C HELLO WORLD!HELL
0770:0010 4F 20 57 4F 52 4C 44 21-00 00 00 00 00 00 00 00 O WORLD!.....
0770:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0770:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0770:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0770:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0770:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
0770:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
```

(7)

```

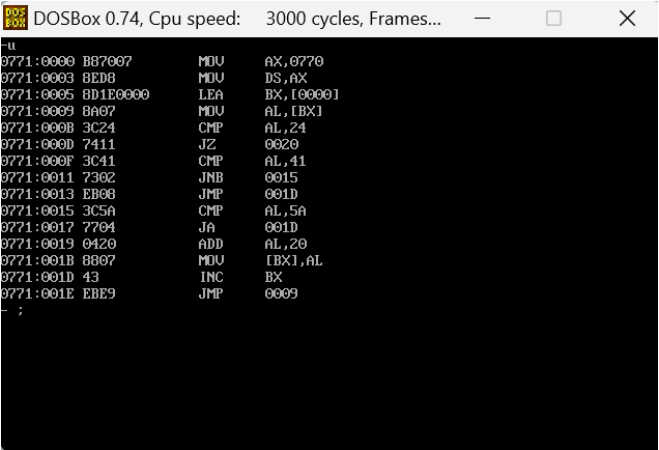
Change    macro    buffer
          lea    bx,buffer
again:    mov    al,[bx]
          cmp    al,'$' ;若到达字符串结尾，退出
          je     done
          cmp    al,'A' ;判断是否大于A
          jae    next1
          jmp    done1
next1:    cmp    al,'Z' ;判断是否小于Z
          ja     done1
          add    al,20h ;满足，大写转小写
          mov    byte ptr [bx],al
done1:    inc    bx ;进入下一个字符
          jmp    again
done:
          endm

DATAS    SEGMENT
          S    db 'Hello Wo1RD!','$'
DATAS    ENDS

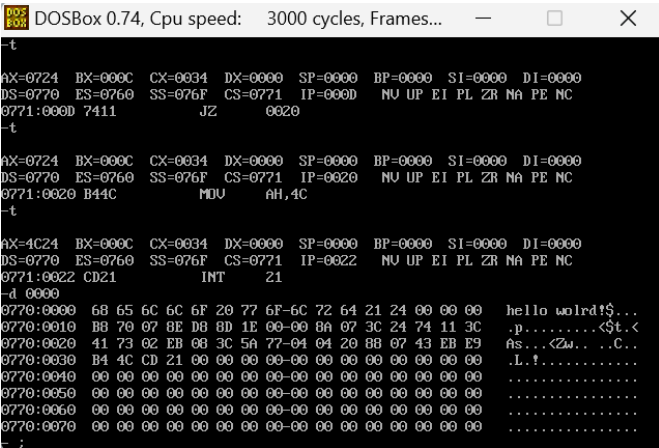
STACKS    SEGMENT
          ;此处输入堆栈段代码
STACKS    ENDS

CODES    SEGMENT
          ASSUME CS:CODES,DS:DATAS,SS:STACKS|
START:
          MOV    AX,DATAS
          MOV    DS,AX
          ;此处输入代码段代码
          Change S
          MOV    AH,4CH
          INT    21H
CODES    ENDS
          END    START

```



可以发现实现了大写转小写。



5 实验分析与总结

- (1) 对汇编语言基础程序编写有更深刻的理解
- (2) 能够运用相关运算完成各类要求
- (3) 进一步学习重复汇编和条件汇编
- (4) 进一步学习宏汇编

批注 [a5]: 针对编译、调试中出现的一些实验结果进行分析，并结合此次实验目的和实验内容，总结实验经验，并针对存在的问题进行阐述和说明，部分疑点可留置上报