

****

《汇编语言》实验报告

（二）

**姓　　名 宋浩元**

**学　　号 37220232203808**

**学 院 信息学院**

**专 业 软件工程**

**2024年 9 月**

# 实验目的

1) 了解汇编语言程序(源程序)的基本组成部分；  
2) 掌握寻址方式以及传送类指令的工作原理；  
3) 进一步掌握使用 DEBUG 相应命令进行程序修改以及指令的调试与运行。

# 实验环境

Masm for Windows 继承环境

DOSBOX 0.74-3

# 实验内容

1. 将例程 a 汇编连接， 生成可执行文件 Hello.exe； 利用 Debug 调试 Hello.exe，  
   完成下述任务：
2. a) 指出字符串“Hello， World！ ”所在的段地址，以及段内起始地址；  
   b) 如果去掉字符串定义语句（黄色加亮） 部分，程序运行结果如何？

MESS DB ‘Hello World!’ , 0DH, 0AH , 24H

c)如果去掉下述语句，程序汇编， 连接以及运行过程中会有什么变化？

SSEG SEGMENT PARA STACK

DW 256 DUP(?)

SSEG ENDS

d) 下述三条语句执行之后， AX,DS,DX 寄存器的取值各位多少？代表什么含  
义？ 针对你所操作的计算机， MESS 的地址是多少？  
 BEGIN： MOV AX, DSEG  
 MOV DS, AX  
 MOV DX, OFFSET MESS  
e) 如果去掉下述语句，程序执行结果有何变化？为什么？  
 MOV AH， 9  
 INT 21H  
(2) 将例程 b 汇编连接， 生成可执行文件 Cal.exe； 利用 Debug 调试 Cal.exe，完  
成下述任务：

a) 给出 X， Y， Z 的内存地址；  
b) 单步执行该程序，观察寄存器 AL 以及标志寄存器的变化；  
c) 执行完成下述代码后， Z 和 Z1 两个内存变量里面的值各是多少？ 各自代  
表什么含义？  
 IDIV BL  
 MOV Z, AL  
 MOV Z1, AH  
d) 程序中， IMUL 以及 IDIV 只有一个操作数，请问另一个操作数在哪里？  
e) 结合例程 a, 在例程 b 中， 补充代码， 显示下述提示信息： ”the result is :”  
然后再输出结果， 请给出补充完整的程序代码以及运行结果。

(3) 利用 DEBUG 调试程序给出下述指令的运行结果：

|  |  |  |
| --- | --- | --- |
| I)  II) | MOV SP,50FFH MOV DX, SP; | DH=? DL=? |
| III) MOV SI,3040H IV) MOV DL, [SI+2] | ； 建议先利用 E 内存修改命令，将[SI+2]处存入自定义 |  |

非零数据， 之后， 观察指令运行结果。  
V) MOV BX,2030H  
VI) MOV WORD PTR[BX+SI],34  
VII) MOV DI, SI

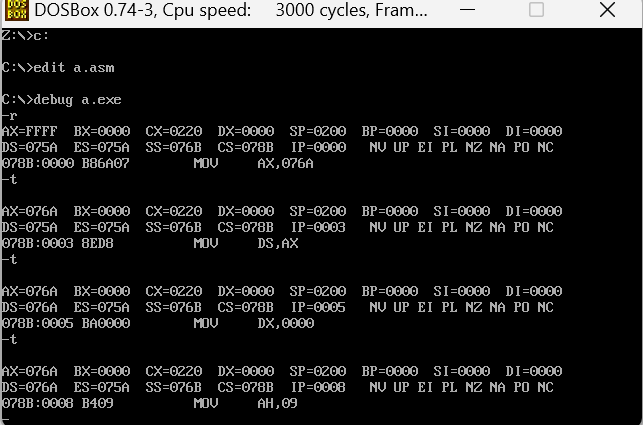
|  |  |
| --- | --- |
| VIII) MOV BP, 2[BX+DI]  IX) LEA AX, [BX+SI+3]  X) PUSH BX  XI) LAHF  XII) XCHG DH, BL | ； 参考题 IV ； 参考题 IV ； SP=? ； Flag=? |

(4) 写出实现下述要求的一条（或几条） 汇编指令，并利用 Debug 程序进行验证。  
⚫ 将一个立即数送入寄存器 BX；  
⚫ 将一个立即数送入段寄存器 DS；  
⚫ 将变址寄存器 DI 送入一个存储单元中；  
⚫ 从存储单元中取出一个数到段寄存器 ES 中；  
  
(5) 使用 Debug 程序运行下述指令，给出运行结果(截屏)，如不能正常运行，请  
解释出错原因（可附上编译报错提示）， 如可以改正，请改正。  
⚫ MOV CX, EDH  
⚫ MOV [SI], 34H  
⚫ MOV [3000H], [2000H]  
⚫ MOV DS,100H  
⚫ MOV CS, [SI]  
⚫ MOV DS, ES

# 实验具体实现

实验一：

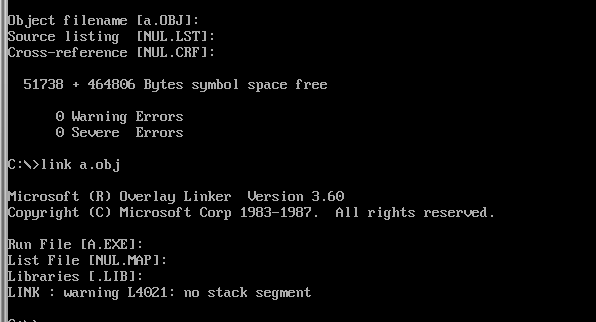
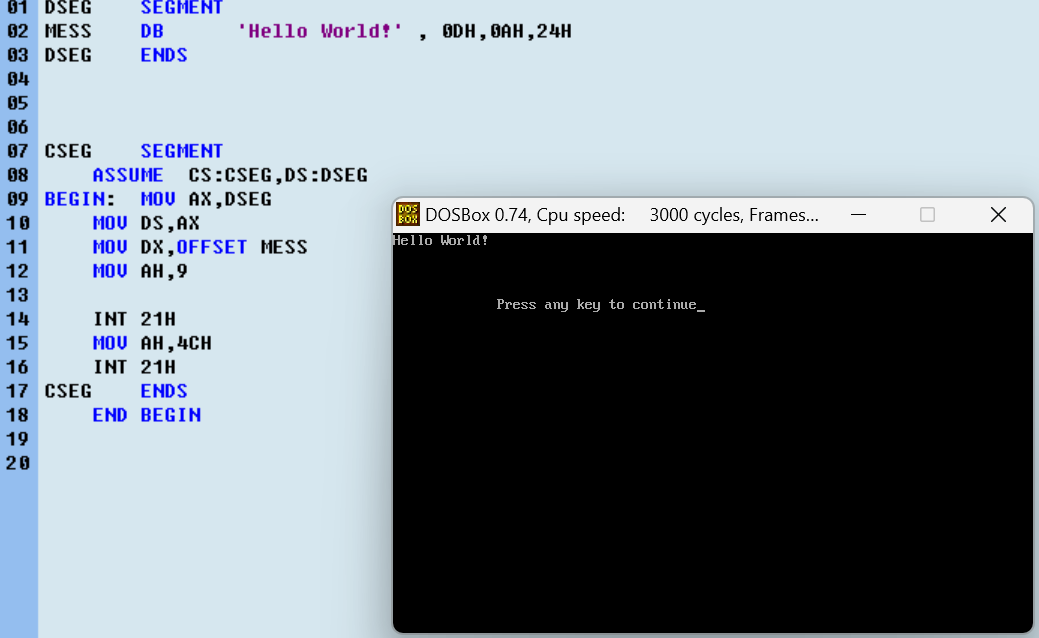
（a）段地址为076A，段起始地址为0000（即offset mess）



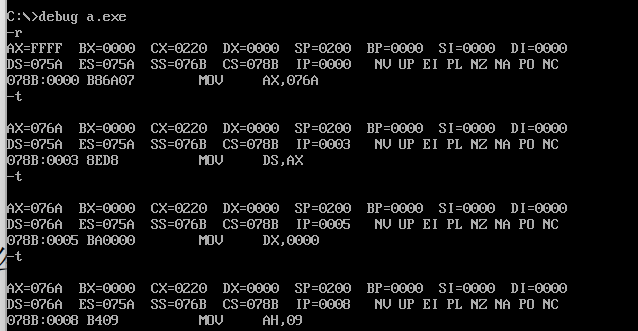
（b）删除后运行出现了乱码情况



1. 删除堆栈定义之后，在link的过程中出现warning，但运行正常



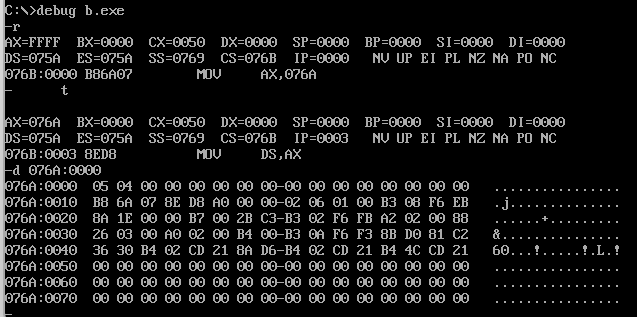
（4）ax = 076A ， DS = 076A ， DX = 0000 ， mess 地址为074A：0000



1. 去掉后程序将不会显示“Hello, World!”消息，并结束运行

实验二：

（1）x：076A:0000 y:076A:0001 z:076A:0002



(2)

MOV AL, X 后，AL = 5

ADD AL, Y 后，AL = 9（无进位，CF=0）

IMUL BL 后（BL=8），AX = 72（AL=72, AH=0），OF=0（无溢出）

SUB AX, BX 后（BX=0005），AX = 67（AL=67, AH=0），CF=0（无借位）

IDIV BL 后（BL=2），AL = 33，AH = 1，ZF=0，SF=0

DIV BL（BL=10）后，AL = 3，AH = 3，ZF=0，SF=0

(3)

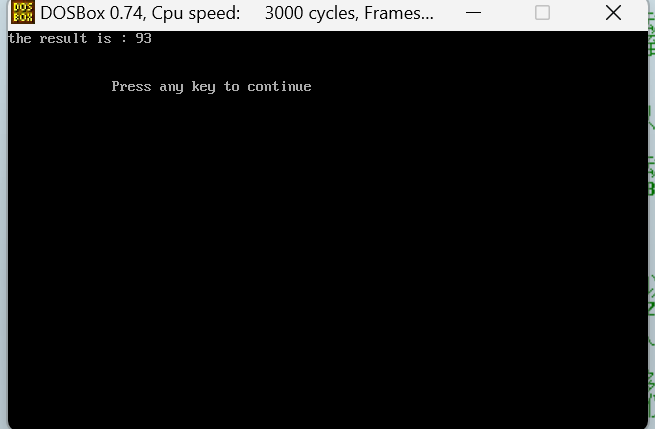
Z 的值是 33，代表 67 除以 2 的商。

Z1 的值是 1，代表 67 除以 2 的余数。

(4)

另一个操作数默认在ax寄存器中。

(5)



实验三：

#### **I) MOV SP,50FFH**

* 结果：SP = 50FFH

#### **II) MOV DX, SP**

结果：DX = 50FFH

#### **III) MOV SI,3040H**

结果：SI = 3040H

#### **IV) MOV DL, [SI+2]**

使用 E 命令在内存地址 3042H 处设置一个45H。

结果：DL = 45H

#### **V) MOV BX,2030H**

* 结果：BX = 2030H

#### **VI) MOV WORD PTR[BX+SI],34**

* 计算地址：BX + SI = 2030H + 3040H = 5070H
* 结果：在内存地址 5070H 和 5071H 处存储值 0034H

#### **VII) MOV DI, SI**

* 结果：DI = SI = 3040H

#### **VIII) MOV BP, 2[BX+DI]**

* 在内存地址 5072H（即 BX+DI+2 = 2030H + 3040H + 2 = 5072H）处设置一个45H。
* 结果：BP = 0045

#### **IX) LEA AX, [BX+SI+3]**

* 计算地址：BX + SI + 3 = 2030H + 3040H + 3 = 5073H
* 结果：AX = 5073H（LEA 指令加载有效地址到目标寄存器）

#### **X) PUSH BX**

* SP 递减 2，并将 BX 的值压入堆栈。
* 新的 SP 值 = 50FFH - 2 = 50FDH

#### **XI) LAHF**

* 将标志寄存器的低8位（FLAGS 的低8位）加载到 AH 寄存器。
* 结果取决于之前的操作，但假设没有改变任何标志位，AH 可能包含之前的标志状态。

#### **XII) XCHG DH, BL**

* 结果：DH = 30H, BL = 50H。

实验四：

1. 从图一可以看出bx由0000 变成了1234
2. 从图一和图二可以看出ax由0000变成了1234
3. 从图二可以看出ds由073F变成了1234
4. 从图三可以看出es由073F变成了1234

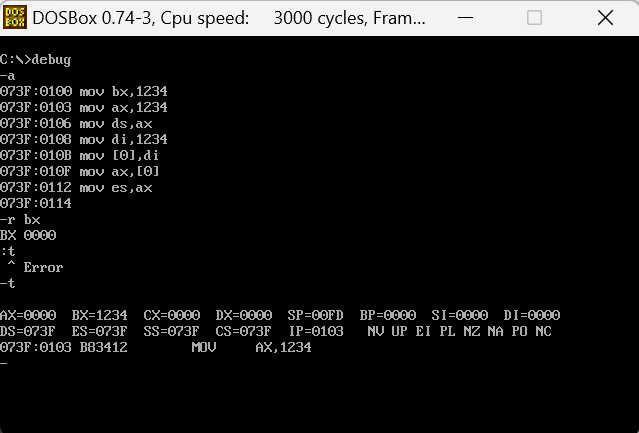


表 1图一

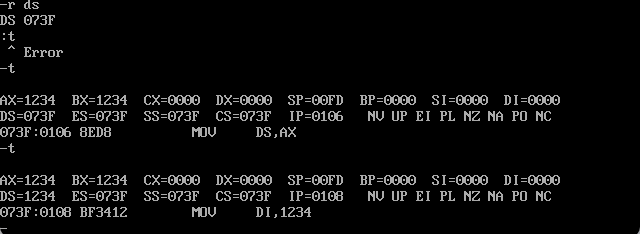


表 2图二

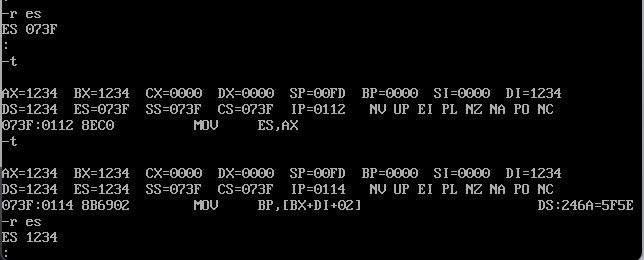


表 3图三

实验五：

（先进入debug写好代码）

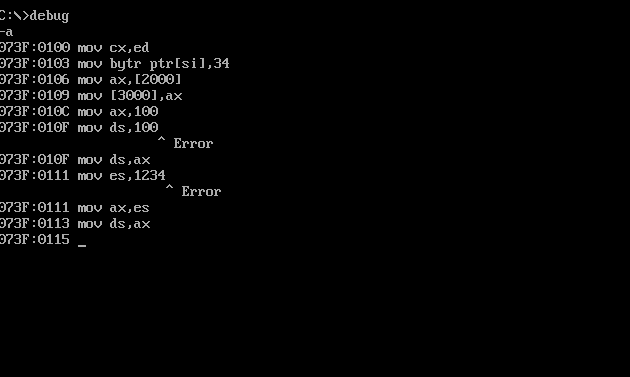


表 4图四

（1）cx从0000变成00ed



表 5图五

（2）需要输入byte ptr才不会报错，可以发现[si]中的值已经变成34h



表 6图六

（3）需要中转，不能直接对段寄存器赋值

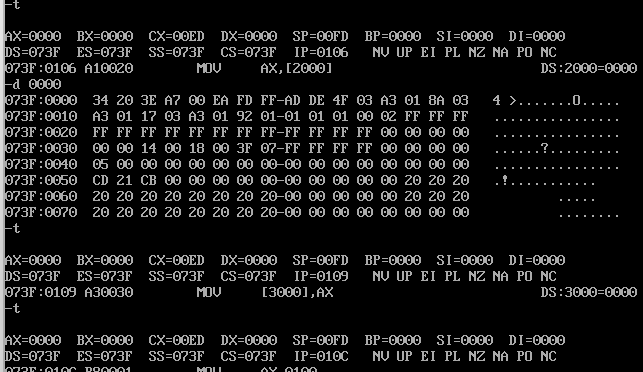


表 7图七

（4）与（3）一样需要中转

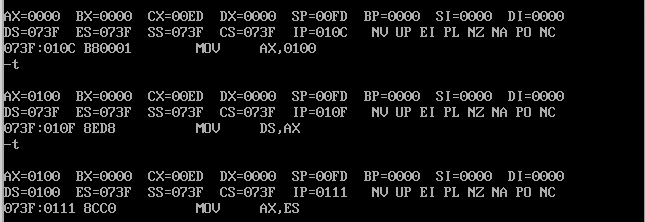


表 8图八

（5）如果输入第四条命令，发现cs的值被改变，但不是期望的0000，而是F000，并且程序会陷入崩溃/死循环？

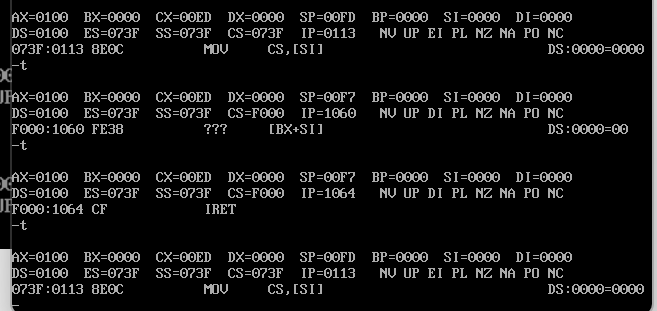
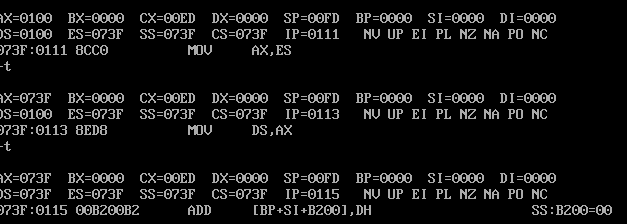


表 9图九

（6）同样需要寄存器中转



# 实验分析与总结...............

1. 学会了掌握寻址方式以及传送类指令的工作原理；如立即数寻址，寄存器寻址，直接寻址，寄存器间接寻址，寄存器相对寻址，基址变址寻址，相对基址变址寻址。
2. 进一步掌握了使用 DEBUG 相应命令进行程序修改以及指令的调试与运行。
3. 了解了汇编语言程序(源程序)的基本组成部分，汇编程序主要由数据段，代码段，堆栈段，段定义和初始化以及指令等组成。
4. 熟悉了集成环境和模拟 dos 环境的使用，了解部分特性。