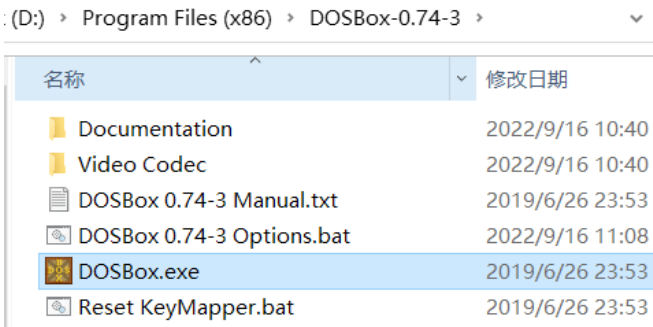
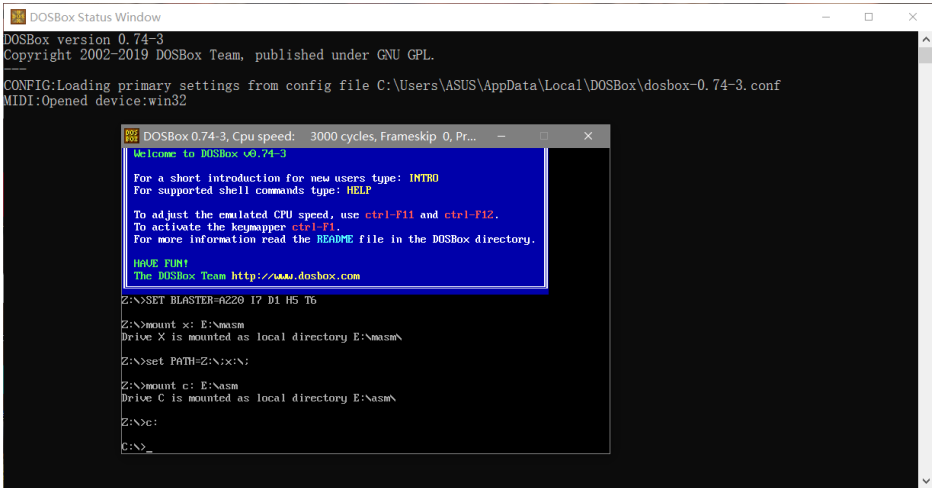


《汇编语言》实验报告 1

班级	2022 秋	实验日期	2022.9.16	实验成绩	
姓名	黄勛	学号	22920212204392		
实验名称	汇编语言第 1 次实验				
实验目的、要求	1) 了解到汇编语言程序的概念以及基本结构 2) 学习了 DOSBox 0.74-3 的环境配置以及基本使用方法 3) 掌握汇编语言程序编写、编译、链接、运行的基本步骤和命令 4) 学习并掌握运用 DEBUG 命令进行程序调试的基本命令 5) 尝试利用 DEBUG 工具进行一些基本的内存修改工作				
实验内容、步骤及结果	<p>1. 下载 DOSBox 0.74-3，先尝试运行模拟 DOS 环境。</p> <div></div> <p>图 1 – 安装的 DOSBox</p> <div></div> <p>图 2 – 运行的 DOSBox 环境界面</p>				

2. 选用 MASM 编辑器,尝试在 DOS 环境中检验是否已配置成功 MASM
具体方法:

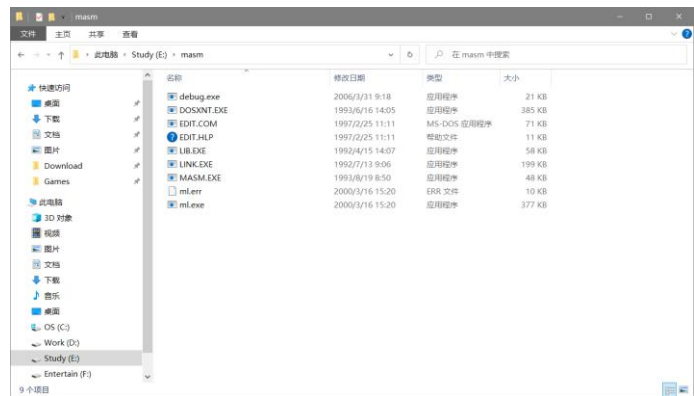


图 3 – 配置的MASM 文件

- 1) 下载 masm 6.15.zip 文件，解压缩后，复制所在的地址
- 2) 先通过挂载的方式将 masm 目录虚拟为 x 盘
- 3) 接下来将虚拟的目录设置为 DOS 环境的环境变量，便于以后直接使用，不用一直切换目录（最后的问题总结中有更详细的说明）

```
Z:\>mount x: E:\masm
Drive X is mounted as local directory E:\masm\

Z:\>set PATH=Z:\;x:\;
```

图 4 – 挂载和配置环境变量

- 4) 此时输入 MASM，查看是否已经配置成功

```
C:\>MASM
Microsoft (R) MASM Compatibility Driver Version 6.11
Copyright (C) Microsoft Corp 1993. All rights reserved.

usage: MASM [option...] source(.asm),[out(.obj)],[list(.lst)],[cref(.crf)]!;!
Run "MASM /H" for more info

C:\>
```

图 5 – 输入 MASM 并正确显示版本号

3. 输入 Helloworld.asm 程序（照着实验材料以及网络 csdn 内容输入）

```
ASM hw.asm X
E: > asm > ASM hw.asm
1  data segment ;数据段
2  | string db 'Hello World$'
3  data ends
4  code segment ;代码段
5  assume cs:code,ds:data
6  start:
7  | mov ax,data ;获取段基址
8  | mov ds,ax ;将段基址送入寄存器
9  | mov dx,offset string
10 | mov ah,9
11 | int 21h
12 | mov ah,4ch
13 | int 21h
14 code ends
15 end start
```

图 6 – 输入的程序

4. 使用 `masm` 与 `link` 命令汇编并运行程序

具体方法:

- 1) 在 PowerShell 中使用 Windows 环境利用 MASM 进行编译, 得到 `obj` 文件
obj 文档一般是 Object 的简写, 是程序编译后的二进制文档, obj 文档可称为目标文档或中间文档。另外 obj 文档只给出了程序的相对地址。



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\ASUS> cd e:\asm
PS E:\asm> MASM /c .\hw.asm
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993. All rights reserved.

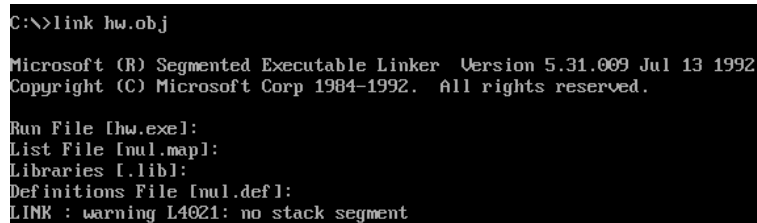
Invoking: ML.EXE /I. /Zm /c /FR /Ta .\hw.asm

Microsoft (R) Macro Assembler Version 6.15.8803
Copyright (C) Microsoft Corp 1981-2000. All rights reserved.

Assembling: .\hw.asm
PS E:\asm>
```

图7- 使用 `cd` 命令切换目录和 `MASM` 命令编译文件

- 2) 在 DOSBox 环境中使用 `link` 命令链接文件, 得到可以打开的 `exe`



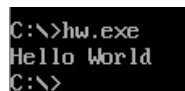
```
C:\>link hw.obj

Microsoft (R) Segmented Executable Linker Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992. All rights reserved.

Run File [hw.exe]:
List File [nul.map]:
Libraries [libl]:
Definitions File [nul.def]:
LINK : warning L4021: no stack segment
```

图8- 链接文件

- 3) 运行可执行文件查看, 已经编译成功



```
C:\>hw.exe
Hello World
C:\>
```

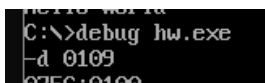
图9- 运行结果

5. 自学“Debug 调试程序”, 了解其基本选项的应用

参考资料: <https://blog.csdn.net/Notzuonotdied/article/details/70888205>

6. 查看“Hello World”字符串所在的内存地址, 使用 `debug` 工具将“W”改为“k”
具体方法:

- 1) 使用 Debug 程序打开 `hw.exe`



```
C:\>debug hw.exe
-d 0109
075C:0109
```

图10- 打开DEBUG

2) 使用 D 命令 (Dump 内存 16 进制显示) 查看内存中的内容

查看指定地址的数据(默认显示 128 个内存单元): **d <段地址>:<偏移地址>**

```
-d 0100
075C:0100  48 65 6C 6C 6F 20 57 6F-72 6C 64 24 00 00 00 00  Hello World$.
075C:0110  B8 6C 07 8E D8 BA 00 00-B4 09 CD 21 B4 4C CD 21  .l.....!.L.!
075C:0120  1E 00 E8 1B 00 E8 18 00-E8 15 00 E8 12 00 E8 0F  .....
075C:0130  00 E8 0C 00 E8 09 00 E8-06 00 E8 03 00 E8 00 00  .....
075C:0140  FA 1E 2E 8E 1E 00 00 A3-7A 13 55 8B EC 8B 46 0A  .....z.U...F.
075C:0150  25 FF BC A3 78 13 8C C0-87 46 04 5D 2D D3 12 51  %...x...F.l-..Q
075C:0160  B1 03 F6 F1 59 C1 E0 02-89 26 76 13 8C 16 74 13  ....Y...&v...t.
075C:0170  2E 8E 16 00 00 8B 26 8C-1F 81 2E 8C 1F 00 01 50  ....&.....P
```

图 11 - 查看内存结果

3) 使用 E 命令 (Enter 修改内存字节) 改写内存中字符串的内容

```
-e 0100 "Hello Korld"
-d 0100
075C:0100  48 65 6C 6C 6F 20 4B 6F-72 6C 64 24 00 00 00 00  Hello Korld$.
075C:0110  B8 6C 07 8E D8 BA 00 00-B4 09 CD 21 B4 4C CD 21  .l.....!.L.!
075C:0120  1E 00 E8 1B 00 E8 18 00-E8 15 00 E8 12 00 E8 0F  .....
075C:0130  00 E8 0C 00 E8 09 00 E8-06 00 E8 03 00 E8 00 00  .....
075C:0140  FA 1E 2E 8E 1E 00 00 A3-7A 13 55 8B EC 8B 46 0A  .....z.U...F.
075C:0150  25 FF BC A3 78 13 8C C0-87 46 04 5D 2D D3 12 51  %...x...F.l-..Q
075C:0160  B1 03 F6 F1 59 C1 E0 02-89 26 76 13 8C 16 74 13  ....Y...&v...t.
075C:0170  2E 8E 16 00 00 8B 26 8C-1F 81 2E 8C 1F 00 01 50  ....&.....P
```

图 12 - 修改内存结果

4) 使用 G 命令 (Go 执行) 继续执行程序, 发现已经修改成功!

```
-g
Hello Korld
Program terminated normally
-
```

图 13 - 修改后运行结果

7. 掌握其他选项的使用: AUDEGHPTQR (下方展示了一部分, 由于目前所学有限还需要一段时间的研究)

1) 用 R 命令查看 CPU 寄存器的内容

```
C:\>debug
-r
查看CPU寄存器的内容
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C81 ES=0C81 SS=0C81 CS=0C81 IP=0100 NU UP EI PL NZ NA PO NC
0C81:0100 FE064404 INC BYTE PTR [0444] DS:0444=CD

C:\>debug
-r
指令的二进制
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C81 ES=0C81 SS=0C81 CS=0C81 IP=0100 NU UP EI PL NZ NA PO NC
0C81:0100 FE064404 INC BYTE PTR [0444] DS:0444=CD
- 当前CS:IP指向地址存放的指令 指令的汇编形式
```

2) 用 R 命令改变 CPU 寄存器的内容

使用 **R <寄存器名称>** 来修改寄存器中的内容, 出现 **:** 时输入要修改的数据, **回车** 完成修改。

```
-r ip
IP 0100 修改IP寄存器的值为0200H
:0200
-r cs
CS 0C81 修改CS寄存器的值为FF00H
:ff00
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C81 ES=0C81 SS=0C81 CS=FF00 IP=0200 NU UP EI PL NZ NA PO NC
FF00:0200 59 POP CX
```

3) 用 D 命令查看 DS 段寄存器指示的内容

```

-r ds
DS 0C81 ← 先设置DS段寄存器的值
:1000
-d ds:0 f ← 然后根据DS查看内存内容
1000:0000 8B 05 26 8B 55 02 F6 44-07 0D 75 0F F6 C6 80 74  Y.&YU.÷D.÷|Çt

```

CS 查看当前代码段所指示的指令

```

-d cs:0
CS 0C81:0000 CD 20 7F 9F 00 9A EE FE-1D F0 4F 03 06 06 8A 03 = Δf.Ü€■.≡0...è.
CS 0C81:0010 06 06 17 03 06 06 02 E8-01 01 01 00 02 FF FF FF .....φ.....
CS 0C81:0020 FF FF FF FF FF FF FF FF-FF FF FF FF EB 05 4E 01 .....δ.N.
CS 0C81:0030 0D 0A 14 00 18 00 01 0C-FF FF FF FF 00 00 00 00 .....ü.....
CS 0C81:0040 07 0A 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
CS 0C81:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20 =!π.....
CS 0C81:0060 20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20 .....
CS 0C81:0070 20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00 .....

```

SS 查看当前栈中的内容

```

-d ss:0
SS 0C81:0000 CD 20 7F 9F 00 9A EE FE-1D F0 4F 03 06 06 8A 03 = Δf.Ü€■.≡0...è.
SS 0C81:0010 06 06 17 03 06 06 02 E8-01 01 01 00 02 FF FF FF .....φ.....
SS 0C81:0020 FF FF FF FF FF FF FF FF-FF FF FF FF EB 05 4E 01 .....δ.N.
SS 0C81:0030 0D 0A 14 00 18 00 01 0C-FF FF FF FF 00 00 00 00 .....ü.....
SS 0C81:0040 07 0A 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
SS 0C81:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 20 20 20 =!π.....
SS 0C81:0060 20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20 .....
SS 0C81:0070 20 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00 .....

```

4) 用 U 命令将机器码翻译为汇编指令

```

-e 1000:0 b8 01 00 b9 02 00 01 c8
-d 1000:0 f
1000:0000 B8 01 00 B9 02 00 01 C8-6F 76 65 72 36 36 36 0F  1...||...öover666.
-u 1000:0
1000:0000 B80100      MOV     AX,0001
1000:0003 B90200      MOV     CX,0002
1000:0006 01C8      ADD     AX,CX
1000:0008 6F        DB      6F
1000:0009 7665      JBE     0070
1000:000B 7236      JB      0043
1000:000D 36        SS:
1000:000E 36        SS:
1000:000F 0F        DB      0F
1000:0010 0443      ADD     AL,43
1000:0012 80E67F     AND     DH,7F
1000:0015 C706C2B0A000     MOV     WORD PTR [8BC21,000A
1000:001B F644070E     TEST    BYTE PTR [SI+071,0E
1000:001F 7506      JNZ     0027

```

5) 用 T 命令执行指令

```

-r cs
CS FF00
:1000
-r ip
IP 0200
-r
AX=FFFF BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C81 ES=0C81 SS=0C81 CS=1000 IP=0000  NV UP EI PL NZ NA PO NC
1000:0000 B80100      MOV     AX,0001
-t
AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C81 ES=0C81 SS=0C81 CS=1000 IP=0003  NV UP EI PL NZ NA PO NC
1000:0003 B90200      MOV     CX,0002
-t
AX=0001 BX=0000 CX=0002 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C81 ES=0C81 SS=0C81 CS=1000 IP=0006  NV UP EI PL NZ NA PO NC
1000:0006 01C8      ADD     AX,CX
-t
AX=0003 BX=0000 CX=0002 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0C81 ES=0C81 SS=0C81 CS=1000 IP=0008  NV UP EI PL NZ NA PE NC
1000:0008 6F        DB      6F

```

6) 用 A 命令写入汇编指令

```
-a 1000:0
1000:0000 mov ax,1
1000:0003 mov bx,2
1000:0006 mov cx,3
1000:0009 add ax,bx
1000:000B add ax,cx
1000:000D add ax,ax
1000:000F
```

开始在指定地址写入汇编指令

输入的是汇编指令，存入内容的是机器指令

输入空内容回车表示输入结束

查看反汇编的效果：

```
-u 1000:0
1000:0000 B80100      MOV     AX,0001
1000:0003 B80200      MOV     BX,0002
1000:0006 B80300      MOV     CX,0003
1000:0009 0108      ADD     AX,BX
1000:000B 01C8      ADD     AX,CX
1000:000D 01C8      ADD     AX,AX
1000:000F 0F      DB      0F
1000:0010 0443      ADD     AL,43
1000:0012 80E67F     AND     DH,7F
1000:0015 C706C28B0A00  MOV     WORD PTR [8BC2],000A
1000:001B F644070E     TEST    BYTE PTR [SI+07],0E
1000:001F 7506      JNZ     0027
```

查看一下根据机器码反汇编出的内容

最后再执行看看效果：

```
-t
AX=0006 BX=0002 CX=0003 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
IS=0C81 ES=0C81 SS=0C81 CS=1000 IP=000D  NO UP EI PL NZ NA PE NC
1000:000D 01C8      ADD     AX,AX
-t
AX=000C BX=0002 CX=0003 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
IS=0C81 ES=0C81 SS=0C81 CS=1000 IP=000F  NO UP EI PL NZ NA PE NC
1000:000F 0F      DB      0F
```

最后一次执行的效果

- 出现的问题：
- 1) 文件名不能超过 8 个字符，在一开始使用“helloworld”作为文件名，在执行过程中出现错误，发现后进行修正。
 - 2) 原本在 DOSBox 中使用 MASM 编译文件发现出现如下错误：

```
C:\>masm hw
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993. All rights reserved.

Invoking: ML.EXE /I. /Zm /c hw.asm

This program cannot be run in DOS mode.
```

最后发现需要在 win 命令行下进行编译。
 - 3) 一开始用 d 命令找不到所需要的字符串，之后发现查看指定地址的内容后，可以使用 d 接着查看，最后在 0100 找到了具体的位置。

总结

在这一次实验中我第一次对汇编语言有一个实际化的认识，在实验过程中虽然遇到了许多困难，但是通过一步步解决问题，我也对计算机的运行原理有了更深刻的认识，并对汇编语言的执行过程有了一个大致的大致的框架，这让我受益匪浅；具体问题的回答我已经附在实验内容中，在这里就不多赘述；在未来我需要对 DEBUG 工具继续研究，对这个程序的内容我还没有探索完全，还要很多指令需要研究和实践；这是一次很有意义的实验。