《汇编语言》实验报告 2

班级	2022 秋	实验日期	2022.10.1	实验成绩	
姓名	黄勖	学号	22920212204392		
实验名称	汇编语言第二次实验				
实验目的、要求	1）熟练掌握汇编语言程序的书写、汇编、连接等步骤 2）掌握基本的 debug 命令，并对程序进行基本的调试				
实验内容、步骤及结果	**(一)将给定程序输入，并汇编、连接后生成可执行文件 lab2.exe**				

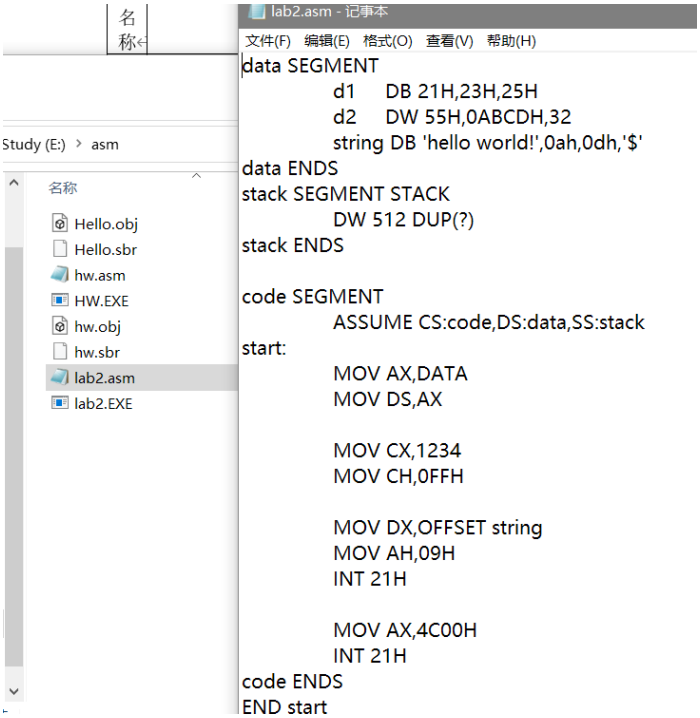

图 1 – 输入的程序

```
PS C:\Windows\system32> cd E:/asm
PS E:\asm> MASM lab2.asm
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1993.  All rights reserved.

 Invoking: ML.EXE /I. /Zm /c /Ta lab2.asm

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

 Assembling: lab2.asm
PS E:\asm>
```

图 2 – 使用 cd 命令切换目录和 MASM 命令编译文件

```
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

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

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

Z:\>mount c: E:\asm
Drive C is mounted as local directory E:\asm\

Z:\>c:

C:\>link lab2.obj

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

Run File [lab2.exe]:

C:\>
```

图 3 – 链接文件

```
C:\>lab2.exe
hello world!
```

图 4 – 运行可执行文件

2.Debug lab2.exe

```
C:\>debug lab2.exe
-
```

图 5 – 执行 debug 指令

(二)将内存中字符串“world”改写成“WORLD”，并显示修改后的结果

①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 0109
075C:0100                                68 65 6C 6C 6F 20 77          hello w
075C:0110  6F 72 6C 64 21 0A 0D 24-00 00 00 00 00 00 00 00   orld!..$........
075C:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075C:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075C:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075C:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075C:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075C:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
075C:0180  00 00 00 00 00 00 00 00-00                        .........
```

图 6 – 查看内存结果

②使用 E 命令（Enter 修改内存字节）改写内存中字符串的内容

```
075C:0180  00 00 00 00
-e 0109 "hello WORLD"
-d 0100
```

图 7 – 修改内存结果

③使用 G 命令（Go 执行）继续执行程序，发现已经修改成功！

```
-g
hello WORLD!

Program terminated normally
-
```

图 8 – 修改后运行结果

(三)展示 3F24+4A2B 和 3F24－4A2B 的计算（用-h 和 add/sub 指令）

Ⅰ（方法 1 – H 指令）H: 执行十六进制算术运算（Hexadecimal）

格式：H 值 1 值 2

值 1、2 为 0—FFFFH 范围内的任意十六进制数。

用途：用来求两个十六进制数的和、差，对结果显示为值 1+值 2 及值 1-值 2。如果值 2＞ 值 1 则显示其补码。

```
-H 3F24 4A2B
894F  F4F9
-
```

图 9 – 使用 H 命令得到的结果（前者为和后者为差）

Ⅱ（方法 2 - add/sub 指令）A:输入汇编指令 T:执行 CS:IP 指向的指令

1） 利用 R 指令先查看各寄存器的内容

```
-R
AX=FFFF  BX=0000  CX=0436  DX=0000  SP=0400  BP=0000  SI=0000  DI=0000
DS=075C  ES=075C  SS=076E  CS=07AE  IP=0000   NV UP EI PL NZ NA PO NC
07AE:0000 B86C07        MOV     AX,076C
```

（可以看出原本 AX 内容为 FFFF，BX 内容为 0000，IP 指向 0000）

2） 利用 R 指令修改 ax,bx 寄存器的内容，使得其中存储我们需要计算的数字

```
-R ax
AX FFFF
:3F24
-R bx
BX 0000
:4A2B
```

（如 -R ax 可以显示 ax 当前值，在“:”后输入新值并回车，在“r ←”后查看修改结果，上图修改了 ax 与 bx 的值为需要计算的数字）

3） 利用 A 指令编写 add 指令，将 ax 与 bx 中的内容相加并存储到 ax 中

```
-a
07AE:0000 add ax,bx
07AE:0002
```

4） 利用 T 指令执行 A 指令编写的代码，可以查看到 ax 的内容变为 894F，即相加的结果

```
-t

AX=894F  BX=4A2B  CX=0436  DX=0000  SP=0400  BP=0000  SI=0000  DI=0000
DS=075C  ES=075C  SS=076E  CS=07AE  IP=0002   OV UP EI NG NZ NA PO NC
07AE:0002 07            POP     ES
```

5） 继续编写计算相减，首先继续重新赋值 ax 与 bx

```
-r ax
AX 894F
:3F24
```

6） 利用 A 指令编写 sub 指令，将 ax 与 bx 中的内容相减并存储到 ax 中

```
-a
07AE:0002 sub ax,bx
07AE:0004
```

7） 利用 T 指令执行 A 指令编写的代码，可以查看到 ax 的内容变为 F4F9，即相减的结果

```
-t

AX=F4F9  BX=4A2B  CX=0436  DX=0000  SP=0400  BP=0000  SI=0000  DI=0000
DS=075C  ES=075C  SS=076E  CS=07AE  IP=0004   NV UP EI NG NZ AC PE CY
07AE:0004 D8B9D204            FDIVR   DWORD PTR [BX+DI+04D2]      DS:4EFD=00
```

(四)在内存中输入 MOV AX, 32H

ADD AX,AX

执行并查看 AX 的变化，修改 AX 的值为 FFFF

① 首先利用 A 指令输入代码

```
-a
07AE:0004 mov ax,32
07AE:0007 add ax,ax
07AE:0009
```

（H 表示 16 进制实际不需要输入，否则会报错）

② 其次使用 T 指令执行两行代码

1）第一次输入 mov 指令将 AX 的值修改为 32（十六进制）

```
-t

AX=0032  BX=4A2B  CX=0436  DX=0000  SP=0400  BP=0000  SI=0000  DI=0000
DS=075C  ES=075C  SS=076E  CS=07AE  IP=0007   NV UP EI NG NZ AC PE CY
07AE:0007 01C0          ADD     AX,AX
```

2）第二次输入 add 指令将 AX 的值进行自增，结果为 64（十六进制）

```
-t

AX=0064  BX=4A2B  CX=0436  DX=0000  SP=0400  BP=0000  SI=0000  DI=0000
DS=075C  ES=075C  SS=076E  CS=07AE  IP=0009   NV UP EI PL NZ NA PO NC
07AE:0009 FFBA0900      ???     [BP+SI+0009]                    SS:0009=0000
```

③ 最后利用 R 指令修改 AX 的值，实验结束

```
-R ax
AX 0064
:FFFF
-R
AX=FFFF  BX=4A2B  CX=0436  DX=0000  SP=0400  BP=0000  SI=0000  DI=0000
DS=075C  ES=075C  SS=076E  CS=07AE  IP=0009   NV UP EI PL NZ NA PO NC
07AE:0009 FFBA0900      ???     [BP+SI+0009]                    SS:0009=0000
```

实验出现的问题：

1）主要出现的问题为在使用 A 指令时输入 mov ax,32H 发现报错

```
-a
07AC:000C mov ax,32h
                  ^ Error
07AC:000C mov ax,32H
                  ^ Error
```

解决方法：H 表示 16 进制实际不需要输入，否则会报错，最后只需要删除 H 即可成功输入汇编指令。

总结	在这一次实验中我对 debug 的使用有了更深的体会，并且我在实际操作中对每一个指令的用途和用法有了更深的认识，通过一步步地解决问题，我对每一个指令的运行模式有了大致的框架，这让我受益匪浅；具体遇到问题的解决方案我已经附在实验内容中，在此就不多赘述；在未来我还要探索 debug 能够解决的更多问题，并在发现问题的过程中继续提高我对汇编语言的掌握能力，这是一次颇有意义的实验！