

汇编语言程序设计课程实验报告

实验 1

姓名		院系	计算机工程与科学学院	学号	
实验目的：					
1. 熟悉汇编语言编程环境； 2. 了解汇编语言程序代码框架； 3. 体会高级语言与汇编语言的优缺点； 4. 掌握汇编语言的调试方法。					
实验任务：					
1. C语言实现例子程序； 2. 汇编语言实现例子程序； 3. 实践MASM环境的程序调试方法； 4. 对比上述两具实例的可执行代码长度、两种编程语言各自的优缺点。					

实验步骤：

1. C语言实验例子子程序

运行汇编子程序可知，这个程序的功能上是输入并显示一个字符串的功能，C语言实现如下：

```
1  #include<stdio.h>
2  #include<string.h>
3
4  int main()
5  {
6      char s[100];
7      printf("sentence please: ");
8      gets(s);
9      printf("%s", s);
10     return 0;
11 }
```

运行效果如下：

```
PS D:\Math\huibian> & 'c:\Users\Lenovo\.vs
IEngine-In-sxbj4spc.bis' '--stdout=Microsof
.4xg' '--dbgExe=D:\Mingw-w64\mingw64\bin\gd
sentence please: I love SHU
I love SHU
PS D:\Math\huibian> []
```

结果显示很好的使用C语言模拟汇编例子程序的效果，完成实验任务1。

2. 汇编语言实现例子程序；

在vs code中安装masm/tasm拓展，输入例子程序汇编代码，效果如下：

```

1 data segment
2     mess db 'sentence please: $'
3     stor db 50 dup(?)
4 data ends
5 code segment
6     assume cs:code,ds:data
7 start:
8     mov ax,data
9     mov ds,ax
10    lea dx,mess
11    mov ah,9
12    int 21h
13    lea di,stor
14    mov cx,0
15 rotate: mov ah,1
16           int 21h
17           cmp al,0dh
18           jz output
19           cmp al,61h
20           jb return
21           cmp al,7ah
22           ja return
23           sub al,20h
24 return: mov [di],al
25           inc di
26           inc cx
27           jmp rotate
28 output: mov dl,0dh
29           mov ah,2
30           int 21h
31           mov dl,0ah
32           mov ah,2
33           int 21h
34           lea di,stor
35 again:  mov dl,[di]
36           mov ah,2
37           int 21h
38           inc di
39           loop again
40           mov ah,4ch
41           int 21h
42 code ends
43     end start

```

运行效果如下：

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
asm\MASM-v6.11""
Drive C is mounted as local directory c:\Users\Lenovo\AppData\Roaming\Code\User\
globalStorage\xsro.masm-tasm\MASM-v6.11\

Z:\>mount d "c:\Users\Lenovo\AppData\Roaming\Code\User\globalStorage\xsro.masm-t
asm\workspace""
Drive D is mounted as local directory c:\Users\Lenovo\AppData\Roaming\Code\User\
globalStorage\xsro.masm-tasm\workspace\

Z:\>d:

D:\>set PATH=C:\MASM

D:\>masm D:\TEST.ASM; >>C:\74367.LOG
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993. All rights reserved.

D:\>link D:\TEST; >>C:\74367.LOG

D:\>D:\TEST
sentence please: I love shu

I LOVE SHU
Do you need to keep the DOSBox [Y,N]?_
```

效果与使用 C 语言模拟的一致，完成实验任务 2。

3. 实践 MASM 环境的程序调试方法。

在 vs code 中点击鼠标右键，点击调试当前运行程序，可以得到以下界面。

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount c "c:\Users\Lenovo\AppData\Roaming\Code\User\globalStorage\xsro.masm-t
asm\MASM-v6.11""
Drive C is mounted as local directory c:\Users\Lenovo\AppData\Roaming\Code\User\
globalStorage\xsro.masm-tasm\MASM-v6.11\

Z:\>mount d "c:\Users\Lenovo\AppData\Roaming\Code\User\globalStorage\xsro.masm-t
asm\workspace""
Drive D is mounted as local directory c:\Users\Lenovo\AppData\Roaming\Code\User\
globalStorage\xsro.masm-tasm\workspace\

Z:\>d:

D:\>set PATH=C:\MASM

D:\>masm D:\TEST.ASM; >>C:\93140.LOG
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993. All rights reserved.

D:\>link D:\TEST.OBJ; >>C:\93140.LOG

D:\>debug D:\TEST.exe
- ^_
```

随后使用 8086DEBUG 中的部分命令进行操作。

1) 追踪指令 T，执行下一条机器指令，效果如下：

```

D:\>debug D:\TEST.exe
-T
AX=076C BX=0000 CX=0099 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076B CS=0771 IP=0003 NU UP EI PL NZ NA PO NC
0771:0003 8ED8          MOV     DS,AX
- ▲▲

```

可以看到显示出执行的第一条指令 MOV DS,AX，由于指令很多，所以需要多次使用 T 指令才能执行完程序。

2) R 指令

R 指令，查看 CPU 寄存器中的内容，效果与 T 指令相同，在 R 指令后面加上寄存器的名称，可以修改对应寄存器的内容，效果如下：

```

-R
AX=076C BX=0000 CX=0099 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076B CS=0771 IP=0003 NU UP EI PL NZ NA PO NC
0771:0003 8ED8          MOV     DS,AX
-r ax
AX 076C
:076A
-R
AX=076A BX=0000 CX=0099 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076B CS=0771 IP=0003 NU UP EI PL NZ NA PO NC
0771:0003 8ED8          MOV     DS,AX
- ▲

```

3) D 指令，查看内存信息，效果如下：

```

0771:0000 B8 6C 07 8E D8 8D 16 00-00 B4 09 CD 21 8D 3E 12 .l.....!>.
0771:0010 00 B9 00 00 B4 01 CD 21-3C 0D 74 10 3C 61 72 06 .....!<.t.<ar.
0771:0020 3C 7A 77 02 2C 20 88 05-47 41 EB E8 B2 0D B4 02 <zw., ..GA.....
0771:0030 CD 21 B2 0A B4 02 CD 21-8D 3E 12 00 8A 15 B4 02 .!.....!>.....
0771:0040 CD 21 47 E2 F7 B4 4C CD-21 0A 00 03 09 46 0A 5D .!G...L.!...F.l
0771:0050 A1 7A 13 1F 07 55 8B EC-F7 46 06 00 02 5D 74 01 .z...U...F...lt.
0771:0060 FB CF 58 FF 36 74 13 FF-36 76 13 6A FF FF 36 78 ..X.6t..6v.j..6x
0771:0070 13 0E 68 6C 16 93 FF 36-78 13 8E 06 1C 00 26 FF ..hl...6x.....&.
- ▲a

```

4) E 指令，修改内存单元内容，用法：-E 地址 内容表 将指定地址开始的内存单元中的内容替换成内容表中的内容。效果如下：

```

-D 076C:0000
076C:0000 73 65 6E 74 65 6E 63 65-20 70 6C 65 61 73 65 3A sentence please:
076C:0010 20 24 00 00 00 00 00 00-00 00 00 00 00 00 00 00 $.
076C:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076C:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076C:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076C:0050 B8 6C 07 8E D8 8D 16 00-00 B4 09 CD 21 8D 3E 12 .l.....!>.
076C:0060 00 B9 00 00 B4 01 CD 21-3C 0D 74 10 3C 61 72 06 .....!<.t.<ar.
076C:0070 3C 7A 77 02 2C 20 88 05-47 41 EB E8 B2 0D B4 02 <zw., ..GA.....
-E 076C:0000 73 65 66 67
-D 076C:0000
076C:0000 73 65 66 67 65 6E 63 65-20 70 6C 65 61 73 65 3A sefgence please:
076C:0010 20 24 00 00 00 00 00 00-00 00 00 00 00 00 00 00 $.
076C:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076C:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076C:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076C:0050 B8 6C 07 8E D8 8D 16 00-00 B4 09 CD 21 8D 3E 12 .l.....!>.
076C:0060 00 B9 00 00 B4 01 CD 21-3C 0D 74 10 3C 61 72 06 .....!<.t.<ar.
076C:0070 3C 7A 77 02 2C 20 88 05-47 41 EB E8 B2 0D B4 02 <zw., ..GA.....
- ▲▲a

```

5) R 指令，检查和修改寄存器内容，首先可以检查 CPU 内部所有寄存器内容和标志位信

息。效果如下：

```
-R
AX=FFFF BX=0000 CX=0099 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076B CS=0771 IP=0000  NU UP EI PL NZ NA PO NC
0771:0000 B86C07      MOV     AX,076C
```

查阅资料可得各个标志位含义如下表：

标志名	置位	复位
溢出Overflow（是/否）	OV	NV
方向Direction（减量/增量）	DN	UP
中断Interrupt（允许/屏蔽）	EI	DI
符号Sign（负/正）	NG	PL
零Zero（是/否）	ZR	NZ
辅助进位Auxiliary Carry（是/否）	AC	NA
奇偶Parity（偶/奇）	PE	PO
进位Carry（是/否）	CY	NC

- 6) U 指令，由 exe 文件反汇编得到机械指令。格式：-U[地址]该命令从指定地址开始，反汇编（指把机器代码翻译成汇编语句）32 个字节，若地址省略，则从上一个 U 命令的最后一条指令的下一个单元开始显示 32 个字节。效果如下：

```
-U
7 start: 0771:0000 B86C07      MOV     AX,076C
8   mov ax,data 0771:0003 8ED8      MOV     DS,AX
9   mov ds,ax 0771:0005 8D160000    LEA     DX,[0000]
10  lea dx,mess 0771:0009 B409      MOV     AH,09
11  mov ah,9 0771:000B CD21      INT     21
12  int 21h 0771:000D 8D3E1200    LEA     DI,[0012]
13  lea di,stor 0771:0011 B90000    MOV     CX,0000
14  mov cx,0 0771:0014 B401      MOV     AH,01
15 rotate: mov ah,1 0771:0016 CD21      INT     21
```

- 7) 退出命令 Q
退出 DEBUG 模式，结束程序。
由以上内容，我初步实践了 MASM 环境的程序调试方法，完成实验任务 3。
4. 通过对比汇编语言和 C 语言，可以观察到 C 语言的语句比汇编语言简短非常多，以下是两种语言的优缺点。

1) C 语言

优点	缺点
易读和易学。C 语言的语法结构更接近自然语言，代码更易于阅读和维护	对于某些场景，特定的优化需要用汇编来实现
可以很方便的移植到不同平台	底层操作和细节控制汇编语言更为强大
有许多第三方库和工具支持	缺乏内建的安全机制，容易产生野指针

- 2) 汇编语言

优点	缺点
精确控制：汇编语言允许开发者直接控制底层硬件和处理器	汇编语言的语法和语义相对较为复杂，不容易理解和学习。
可以实现高度优化的代码，针对特定的硬件平台和应用需求进行性能调整	可读性和可维护性差，维护汇编代码更加困难和耗时
在逆向工程和安全领域，汇编语言是非常重要的工具，可以帮助分析和理解底层系统	可移植性较差，需要针对不同的硬件平台进行修改和适配。

5. 实验体会

汇编语言不同于高级程序设计语言，在计算机组成原理课中我初步接触到了汇编语言，当真正使用到它来编程，写一个简单的小程序时仍是觉得具有较大的挑战性，很多在高级语言上一条代码就能解决的问题再汇编语言却需要“绕很多弯路”，接触时间不长，一时间没有适应汇编语言的逻辑操作，我相信经过通过后面汇编语言的学习，我会进一步增加我对计算机的理解。