

**课 程 实 验 报 告**

**课程名称： 计算机系统基础**

**实验名称： 机器级语言理解和编程**

**院 系 ：计算机科学与技术**

**专业班级 ： 本硕博202001**

**学 号 ： U202115666**

**姓 名 ： 刘文博**

**指导教师 ： 许向阳**

**2022 年 10 月 1 日**

**一、实验目的与要求**

(1) 熟练掌握程序开发平台(VS2019) 的基本用法，包括程序的编译、链接和调试；

(2) 熟悉编程的基础知识，包括数据在计算机内的表现形式、寻址方式、常用指令等；

(3) 熟悉程序运行的基本原理；

(4) 熟悉分支、循环程序的结构及控制方法，掌握分支、循环程序的调试方法；

(5) 加深对转移指令及一些常用的汇编指令的理解；

(6) 完成指定功能的程序设计与调试。

**二、实验内容**

**任务4.1 汇编语言程序的编译、链接和调试**

下面汇编语言源程序的功能是：将buf1缓冲区中的12个字节内容拷贝到 buf2中,并显示两个缓冲区中的字符串。

要求使用VS2019进行编译、链接和调试。打开反汇编窗口显示，观察汇编源程序中的语句与反汇编语句之间的关系，分析指令机器码的编码规则。在内存窗口、监视窗口观察数据段的存储结果，说明数据存储规律、各个变量的地址之间的关系等，在寄存器窗口观察寄存器（包括EIP）的值。

汇编语言源程序如下：

.686

.model flat, stdcall

ExitProcess PROTO STDCALL :DWORD

includelib kernel32.lib ; ExitProcess 在 kernel32.lib中实现

printf PROTO C :VARARG

includelib libcmt.lib

includelib legacy\_stdio\_definitions.lib

.DATA

lpFmt db "%s",0ah, 0dh, 0

X DB 10, 255, -1

Y DW 10, 255, -1

Z DD 10, 255, -1

U DW ($-Z)/4

STR1 DB 'Good', 0

P DD X, Y

Q DB 2 DUP (5, 6)

buf1 db 'Hello World',0

buf2 db 12 dup(0) ; 12个字节的空间，初值均为 0

.STACK 200

.CODE

main proc c

MOV ESI,OFFSET buf1

MOV EDI,OFFSET buf2

MOV ECX,0

L1:

MOV EAX, [ESI] ;如果总数不是12个字节，还能每次传送4个字节吗？

MOV [EDI],EAX

ADD ESI, 4

ADD EDI, 4

ADD ECX, 4

CMP ECX,12

JNZ L1

invoke printf,offset lpFmt,OFFSET buf1

invoke printf,offset lpFmt,OFFSET buf2

invoke ExitProcess, 0

main endp

END

**任务4.2 用指定的方式，改写任务4.1中的程序。**

1. 使用变址寻址方式访问 buf1、buf2;
2. 去掉循环，使用直接寻址方式访问 buf1、buf2。

**任务4.3 用汇编语言编写一个输入密码与验证密码的程序。**

提示输入一个最长10位的字符串密码，与程序内保存的字符串密码比较，相同显示“OK!”, 程序退出；不相同显示“Incorrect Password！”，程序退出。

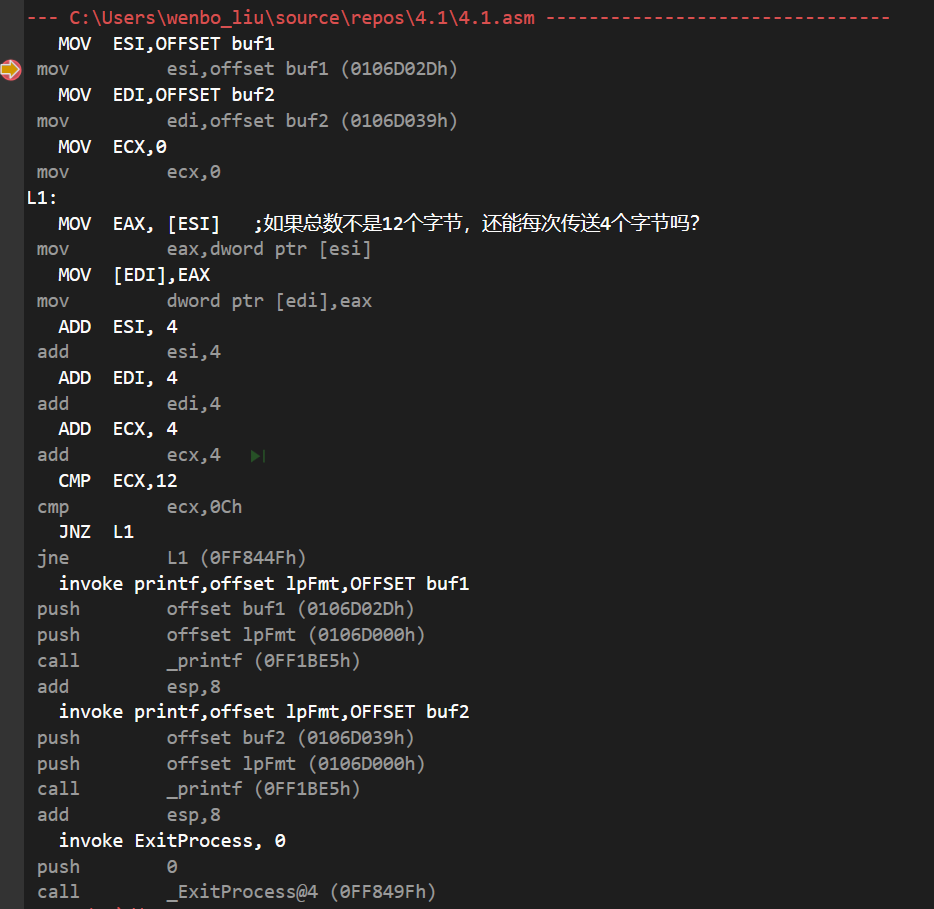
要求：本程序采用分支、循环的程序结构，需要选择合适的字符串比较方法。尝试：采用字节比较、字比较时的程序差异。

**三、实验记录及问题回答**

**（1）实验任务 4.1 的实验记录及问题回答**

**实验记录及问题回答**

反汇编结果如下



如上图所示反汇编指令与汇编原指令大体一致，但在寄存器间接寻址时会特别指出数据的字长（加上dword ptr）。另外invoke指令会具体为将参数进栈再调用函数，最后取消参数占用的堆栈空间的过程。

数据段存储的内容如下



放大后可以看到

X为字节类型 10，255，-1数据表示为0a ff ff

紧接着就是Y 按照小段存储的规律

Y为字类型 10，255，-1数据表示为000a 00ff ffff

紧接着就是Z 按照小段存储的规律

Z为双字类型 10，255，-1数据表示为0000000a 000000ff ffffffff

紧接着就是U 按照小段存储的规律

U为字类型 数据表示为0003 可以看到这与（$-Z）/4的结果，当前地址即U之后的地址减去Z的地址，一共12个字节除4的结果一致

之后的STR1为子类型 'Good', 0 的数据表示为 47 6f 6f 64 00 与Good的ASCII码对应 00与0对应

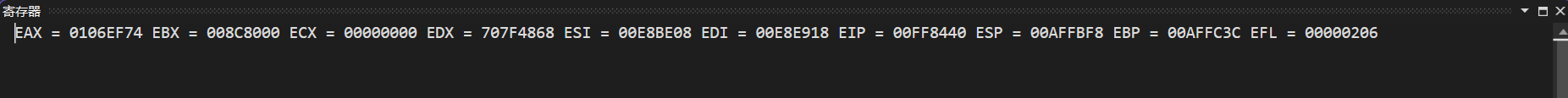
之后的P定义为DD X, Y为双字类型，保存X Y的32位地址，可以看到其数据表示01060d05和开头X的地址相对应，01060d008位X的地址加3字节与Y的地址相对应

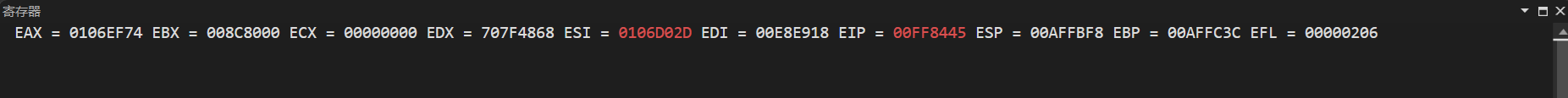
再之后的Q定义为重复2次定义的字节类型的5，6 其数据表示为05 06 05 06

紧接着为buf1 db 'Hello World',0 数据表示为相应的ASCII码，以00做结尾

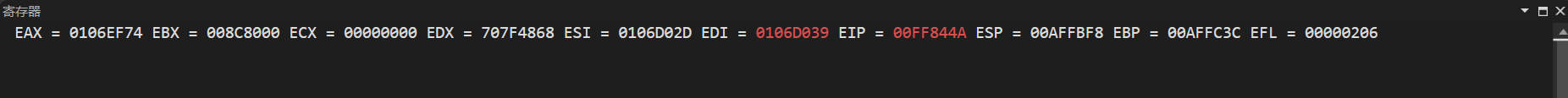
再紧接着为buf2 db 12 dup(0) 数据表示为12个00

寄存器存储的内容如下

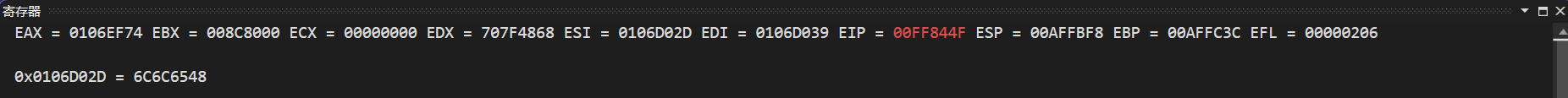
初始值



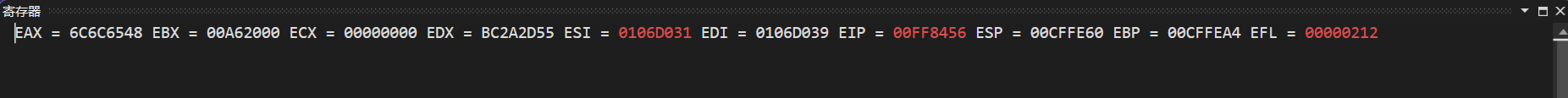
将buf1的地址传给ESI



将buf2的地址传给EDI

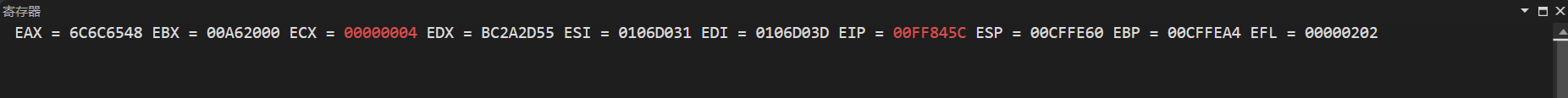


执行了MOV ECX,0 但它原本既是0此处没有标红EIP指向ADD ESI, 4

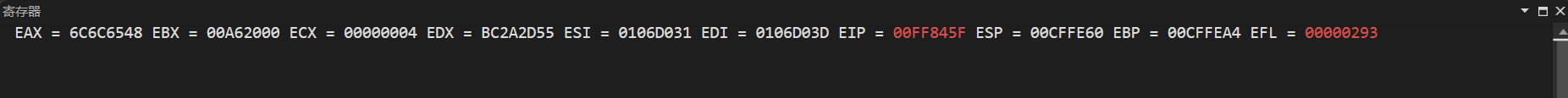


ESI加4 EIP指向ADD EDI, 4

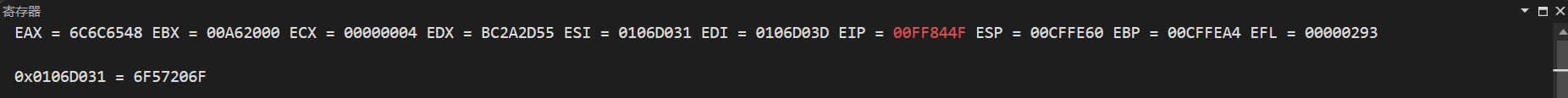
此处省略之后对ADD EDI, 4和ADD ECX, 4指令寄存器变化的追踪



EIP指向CMP ECX,12



执行CMP ECX,12 ELF的值根据比较结果发生改变



执行JNZ L1指令 EIP又指向ADD ESI, 4

此处省略循环过程最终ECX = 0000000C 接着执行了invoke printf,offset lpFmt,OFFSET buf1 invoke printf,offset lpFmt,OFFSET buf2 invoke ExitProcess, 0

Invoke语句涉及多条指令，寄存器变化较为复杂。

**（2）实验任务 4.2 的运行结果**

**运行结果**

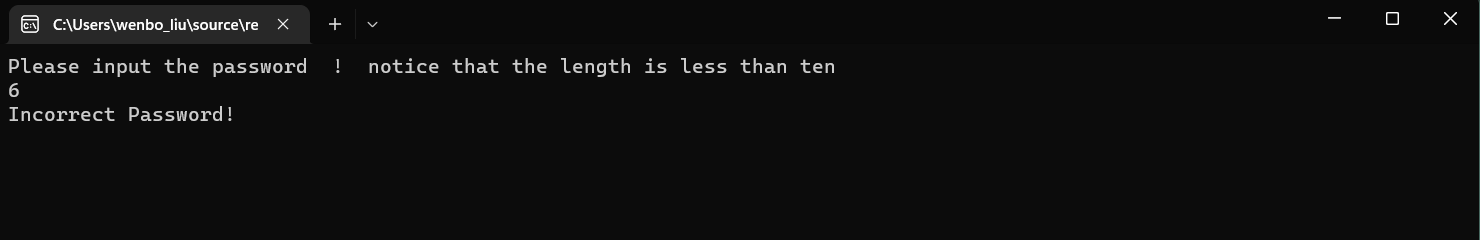


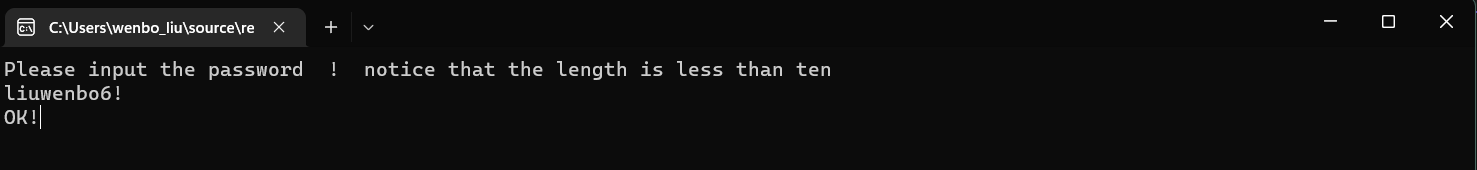
**（3）实验任务 4.3 的算法思想及运行结果**

**算法思想**

通过逐字节比较输入数据与保存的数据，当cmp出现不相等的情况，跳转到incorrect分支，当eax大于10时跳转到ok分支，两个分支最终都以程序退出为结束。

**运行结果**





**四、体会**

通过这次实验，汇编语言编程和反汇编有了更清晰的认识，对数据在内存中的存储以及小端存储的规律有了更深刻的了解，明白了基本的寄存器功能、EIP的改变规则以及多种寻址方式，最后自己编写了汇编程序。编写能够读入字符串并包含循环和分支结构的汇编程序是一个不小的挑战，但最终我完成了这一工作并对函数调用和循环分支结构有了更好的掌握。

**五、源码**

实验任务 4.2 的main 函数的源程序

方式（1）的源代码

.686

.model flat, stdcall

ExitProcess PROTO STDCALL :DWORD

includelib kernel32.lib ; ExitProcess 在 kernel32.lib中实现

printf PROTO C :VARARG

includelib libcmt.lib

includelib legacy\_stdio\_definitions.lib

.DATA

lpFmt db "%s",0ah, 0dh, 0

X DB 10, 255, -1

Y DW 10, 255, -1

Z DD 10, 255, -1

U DW ($-Z)/4

STR1 DB 'Good', 0

P DD X, Y

Q DB 2 DUP (5, 6)

buf1 db 'Hello World',0

buf2 db 12 dup(0) ; 12个字节的空间，初值均为 0

.STACK 200

.CODE

main proc c

MOV ECX,0

L1:

MOV EAX,dword ptr buf1[ECX\*4] ;如果总数不是12个字节，还能每次传送4个字节吗？

MOV dword ptr buf2[ECX\*4],EAX

INC ECX

CMP ECX,4

JNZ L1

invoke printf,offset lpFmt,OFFSET buf1

invoke printf,offset lpFmt,OFFSET buf2

invoke ExitProcess, 0

main endp

END

;任务4.2 用指定的方式，改写任务4.1中的程序。

;(1) 使用变址寻址方式访问 buf1、buf2;

;(2) 去掉循环，使用直接寻址方式访问 buf1、buf2。

方式（2）的源代码

.686

.model flat, stdcall

ExitProcess PROTO STDCALL :DWORD

includelib kernel32.lib ; ExitProcess 在 kernel32.lib中实现

printf PROTO C :VARARG

includelib libcmt.lib

includelib legacy\_stdio\_definitions.lib

.DATA

lpFmt db "%s",0ah, 0dh, 0

X DB 10, 255, -1

Y DW 10, 255, -1

Z DD 10, 255, -1

U DW ($-Z)/4

STR1 DB 'Good', 0

P DD X, Y

Q DB 2 DUP (5, 6)

buf1 db 'Hello World',0

buf2 db 12 dup(0) ; 12个字节的空间，初值均为 0

.STACK 200

.CODE

main proc c

L1:

MOV EAX,dword ptr buf1 ;如果总数不是12个字节，还能每次传送4个字节吗？ 不能

MOV dword ptr buf2,EAX

MOV EAX,dword ptr buf1+4

MOV dword ptr buf2+4,EAX

MOV EAX,dword ptr buf1+8

MOV dword ptr buf2+8,EAX

invoke printf,offset lpFmt,OFFSET buf1

invoke printf,offset lpFmt,OFFSET buf2

invoke ExitProcess, 0

main endp

END

;任务4.2 用指定的方式，改写任务4.1中的程序。

;(1) 使用变址寻址方式访问 buf1、buf2;

;(2) 去掉循环，使用直接寻址方式访问 buf1、buf2。

实验任务 4.3 的源程序

;提示输入一个最长10位的字符串密码，与程序内保存的字符串密码比较，相同显示“OK!”, 程序退出；不相同显示“Incorrect Password！”，程序退出。

;要求：本程序采用分支、循环的程序结构，需要选择合适的字符串比较方法。尝试：采用字节比较、字比较时的程序差异。

;提示输入一个最长10位的字符串密码，与程序内保存的字符串密码比较，相同显示“OK!”, 程序退出；不相同显示“Incorrect Password！”，程序退出。

;要求：本程序采用分支、循环的程序结构，需要选择合适的字符串比较方法。尝试：采用字节比较、字比较时的程序差异。

.686

.model flat, stdcall

ExitProcess PROTO STDCALL :DWORD

includelib kernel32.lib ; ExitProcess 在 kernel32.lib中实现

printf PROTO C :VARARG

scanf PROTO C :VARARG

includelib libcmt.lib

includelib legacy\_stdio\_definitions.lib

.DATA

password db "liuwenbo6!",0

buf1 db "Please input the password ! notice that the length is less than ten ",10,0

input db 11 DUP(0)

lpFmt db "%s",0

buf2 db "Incorrect Password!",0

buf3 db "OK!",0

.STACK 200

.CODE

main proc c

invoke printf,offset lpFmt,offset buf1

invoke scanf,offset lpFmt,offset input

mov eax,0

judge:

mov cl,input[eax]

cmp password[eax],cl

jne incorrect

inc eax

cmp eax,10

jg ok

jmp judge

incorrect:

invoke printf,offset lpFmt,offset buf2

jmp exit

ok:

invoke printf,offset lpFmt,offset buf3

exit:

invoke ExitProcess, 0

main endp

END