

**汇编语言与逆向技术**

**课程实验报告**

**实验一：HelloWorld**

****

学 院 网络空间安全

专 业 信息安全

学 号 2212998

姓 名 胡博浩

班 级­­­­ 2022级信息安全

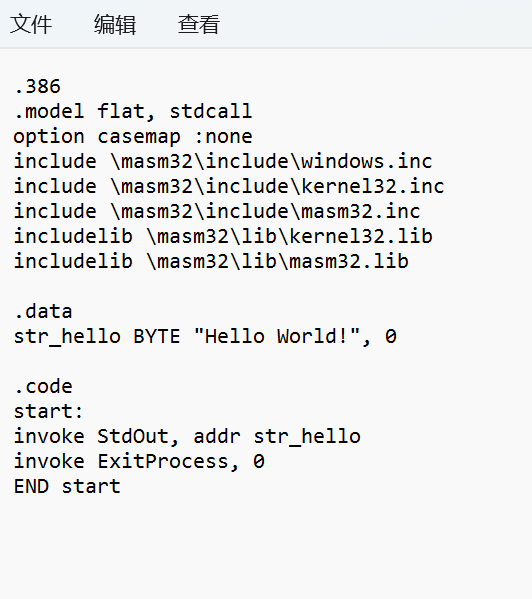
一、实验内容

1. 熟悉 Win32 汇编 MASM32 的编译环境；
2. 命令行输出“HelloWorld”
3. 窗口输出“HelloWorld”

二、实验步骤

**A.**命令行输出“**HelloWorld**”

1.编辑：用编辑软件（Notepad）形成源程序（.asm）,如：hello\_console.asm。将编辑好的代码放到记事本的界面当中，而后将代码保存为.asm格式。

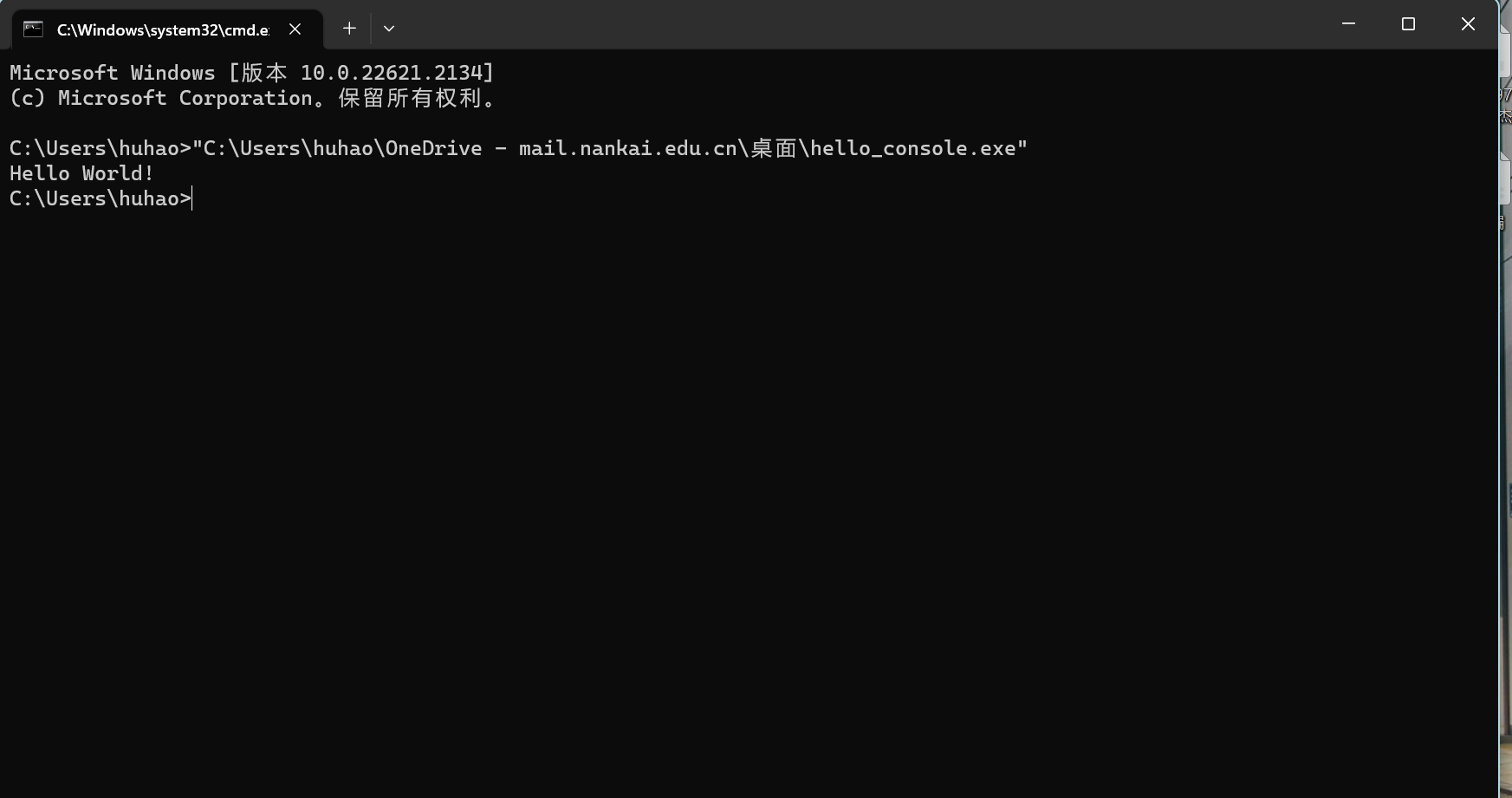


2.编译：用汇编程序（\masm32\bin\ml.exe）对源程序进行汇编，形成目标

文件（.obj），格式如下：“\masm32\bin\ml/c/Zd/coffhello\_console.asm”。

3.连接：用连接程序（\masm32\bin\link.exe）对目标程序进行连接，形成可执行文件（.exe），格式如下：“\masm32\bin\Link /SUBSYSTEM:CONSOLE hello\_console.obj”。

4.执行：找到可执行文件（hello\_console.exe）,输入到cmd中执行可执行文件。程序成功输出helloworld。



B.窗口输出“HelloWorld”

1.编辑：用编辑软件（Notepad）形成源程序（.asm），hello\_window.asm.

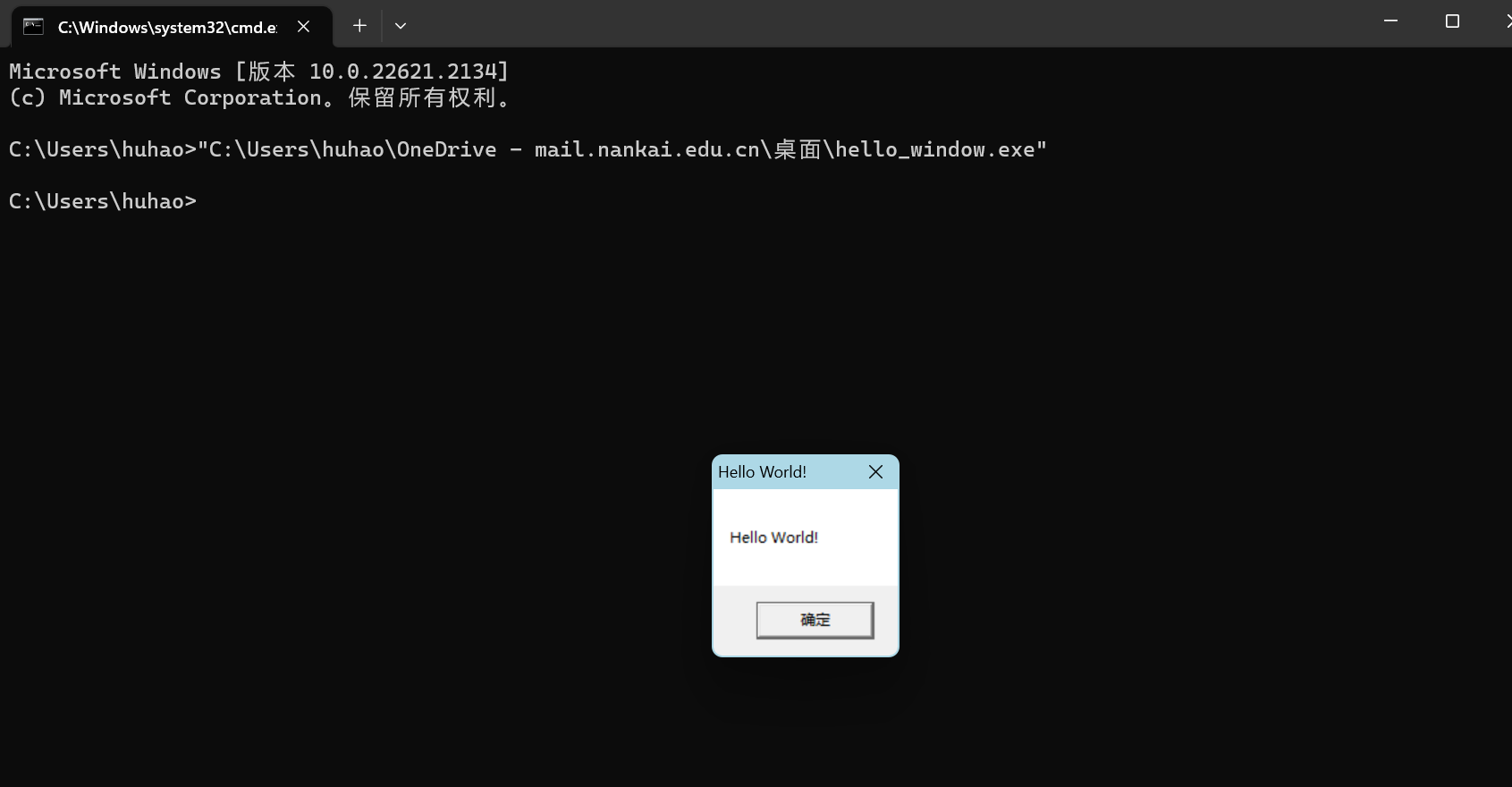


2.编译：用汇编程序（\masm32\bin\ml.exe）对源程序进行汇编，形成目标

文 件（.obj），格式如下：“\masm32\bin\ml/c/Zd/coffhello\_window.asm”。

3.连接：用连接程序（\masm32\bin\link.exe）对目标程序进行连接，形成可执行文件（.exe），格式如下：“\masm32\bin\Link /SUBSYSTEM:WINDOWS hello\_window.obj”。

4.执行：找到可执行文件（hello\_window.exe）,输入到cmd中执行可执行文件。程序成功在窗口输出helloworld。

三、实验命令以及代码解析

1.汇编命令和参数的解析：（以 hello\_console.asm 文件命令为例）

“**\masm32\bin\ml /c /Zd /coff hello\_console.asm**”

①\masm32\bin\ml 代表打开盘中 masm32 中 bin 文件夹的 ml 应用程序。

②/c 代表仅进行编译，而不自动进行链接。

③/coff 代表产生的 obj.文件格式为 COFF 格式。

④/Zd 代表行号调试信息。

⑤hello\_console.asm 是操作对象文件名。

**“\masm32\bin\link/SUBSYSTEM:CONSOLEhello\_console.obj”**

①“\masm32\bin\link” 代表打开盘中 masm32 中 bin 文件夹的 link 应用程序

②SUBSYSTEM:CONSOLE由连接器的“/subsystem:”选项参数确定具体设置哪个程序入口点，告诉操作系统如何运行编译生成的.exe 文件。可以指定四种方式：“CONSOLE|WINDOWS|NATIVE|POSIX”

③hello\_console.obj 是操作对象文件名

2.汇编程序解析

**A.hello\_console.asm**

①.386 指明指令集。

②.modelflat,stdcall 定义了程序的内存模型和调用规则。.modelflat表示使用平坦内存模型，.stdcall 表示使用标准调用规则。

③option casemap :none 这行代码设置不区分大小写。

④include\masm32\include\windows.inc

include\masm32\include\kernel32.inc

include\masm32\include\masm32.inc

includelib\masm32\lib\kernel32.lib

includelib \masm32\lib\masm32.lib

用于包含所需的头文件和链接库。其中，windows.inc 包含了Windows API 的定义，kernel32.inc 包含了Windows内核API的定义，masm32.inc 包含了 MASM32库的定义。includelib 语句告诉编译器链接相应的库文件。

⑤.data str\_hello BYTE "Hello World!", 0 在 .data 部分，程序定义了一个字符串变量 str\_hello，内容为 "Hello World!"，并在末尾加上了一个空字符（字符串结束标志）。

⑥.code start 在 .code 部分，程序的主函数开始，起始标签是 start。

⑦invokeStdOut,addrstr\_hello 这行代码调用了一个函数 StdOut，该函数的作用是将指定的字符串输出到标准输出（通常是控制台）。invoke 是MASM 提供的宏，用于简化函数调用。addrstr\_hello 获取 str\_hello 字符串的内存地址，并作为参数传递给 StdOut 函数。

⑧invokeExitProcess,0 这行代码调用了 ExitProcess 函数，该函数用于退出当前进程。传递的参数是退出码，通常0表示正常退出。

⑨ENDstart END 语句标志着程序的结束。start是程序的入口点，也就是程序开始执行的地方。

**B.hello\_window.asm**

①.386 指定汇编器生成的代码为80386（或以上）处理器的指令。

②.model flat, stdcall 定义内存模型为平坦模型（flat），调用规则为stdcall，即参数从右向左入栈，被调用者负责清理堆栈。

③option casemap :none 设置标识符不区分大小写。

④include\masm32\include\windows.inc

include\masm32\include\kernel32.inc

include \masm32\include\user32.inc

包含WindowsAPI的头文件，windows.inc 包含了基本的WindowsAPI定义， kernel32.inc 包含了Windows内核API的定义，user32.inc 包含了用户界面

API的定义。

⑤includelib\masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

使用链接库 kernel32.lib 和 user32.lib，这些库包含了对应的API函数的实现。

⑥.data str\_hello BYTE "Hello World!", 0 在 .data 部分，定义了一个以NULL结尾的字符串，内容为 "Hello World!"。

⑦.code start:在 .code 部分，程序的主函数开始，起始标签是 start。

⑧invoke MessageBox, NULL, addr str\_hello, addr str\_hello, MB\_OK 调用 MessageBox 函数，该函数用于显示一个消息框。第一个参数是消息框的父窗口句柄（这里为NULL，表示没有父窗口），第二个和第三个参数是消息框中显示的文本（都指向 str\_hello），第四个参数指定消息框的样式（这里为 MB\_OK，表示消息框只有一个“确定”按钮）。

⑨invoke ExitProcess, 0 调用 ExitProcess 函数，该函数用于终止当前进程。传递的参数是退出码（这里为0，表示正常退出）。

⑩END语句标志着程序的结束。start是程序的入口点，也就是程序开始执行的地方。