Introduction to Malware Analysis

Assignment 4 – A Crash Course in X86 Disassembly

10 points

LAB – ASSEMBLY CODE

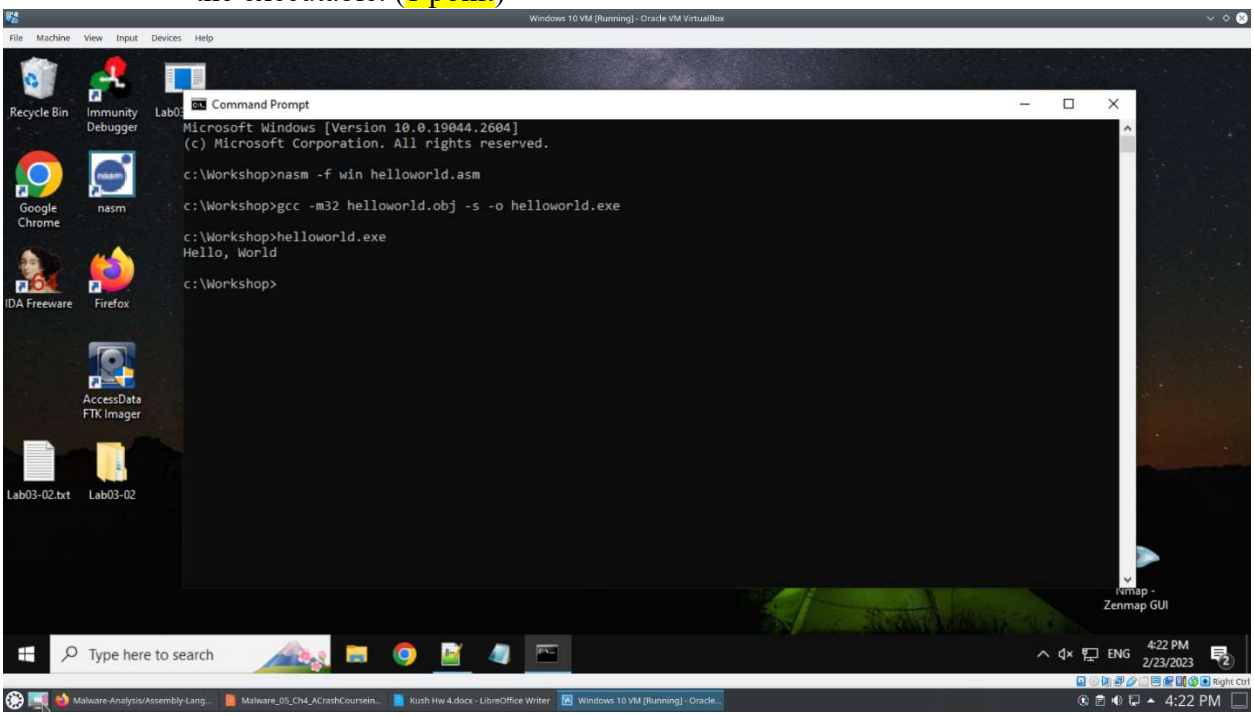
Read this tutorial on assembly language if needed.

Please note: If students copy and paste the assembly code from the slides, it may contain invisible formatting symbols and the compilation may fail.

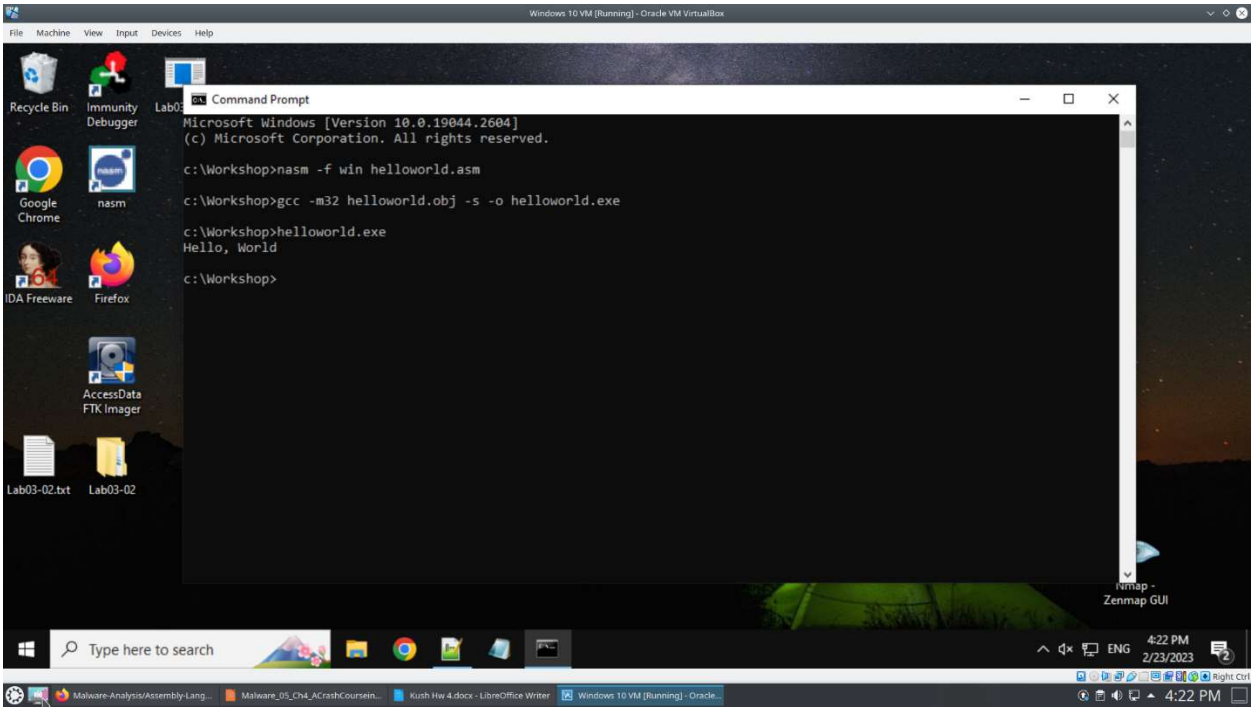
1. Copy the code in helloWorld.asm in the slides.
    a. Please give a note (comment, denoted by a semicolon ;) on each instruction (each line) explaining what it does and paste the code with the notes below. (2 points)

```
extern  _printf                 ;functions from C library
    section .text               ; section called .text
_main:                          ; functions
    push    message             ;push message onto stack
    call    _printf             ;print function is called
    add     esp, 4              ;add 4 to esp
    Ret                         ;return
message:                        ;section called message
    db  'Hello, World', 10, 0   ;string Hello World
```

    b. Provide a screenshot of the commands compiling the code and linking to create the executable. (1 point)



c. Provide a screenshot of the results running the executable. (1 point)

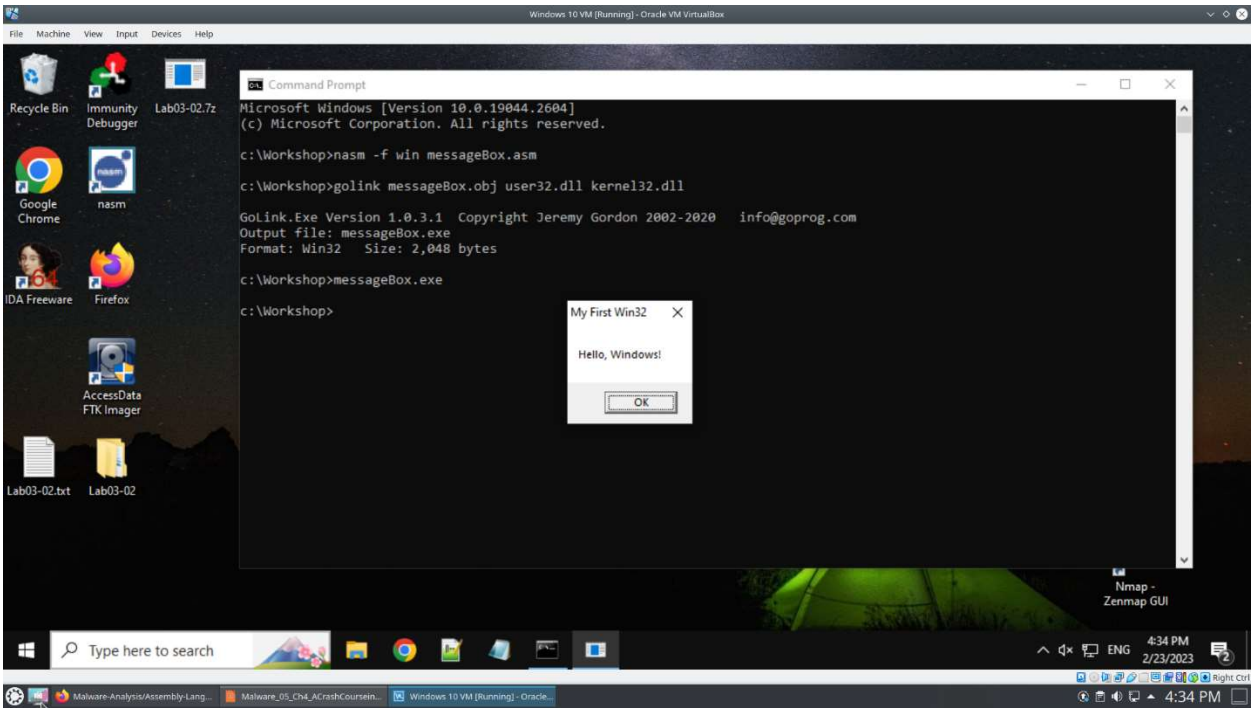


2. Copy the code in messageBox.asm in the slides.
   a. Please give a note (comment, denoted by a semicolon ;) on each instruction (each line) explaining what it does and paste the code with the notes below. (1 point)

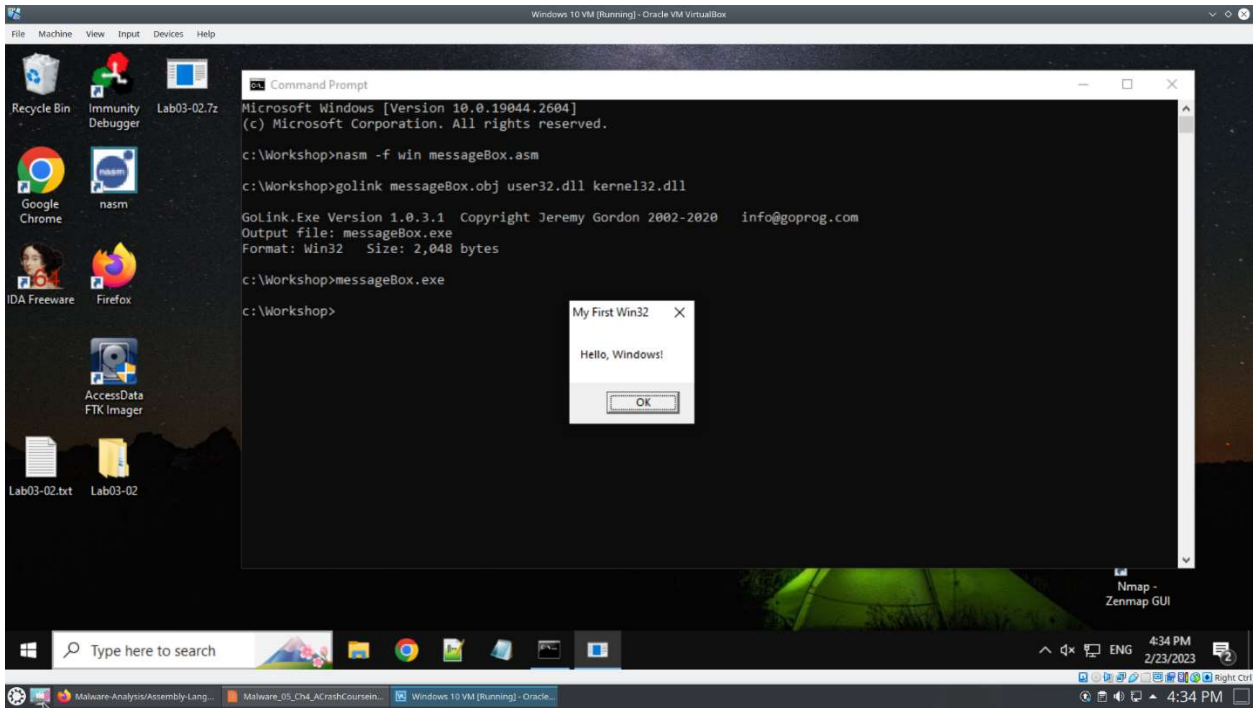
```
global  start
NULL equ 0                          ;equation set to 0
MB_OK       equ 0
extern MessageBoxA ;from user32     ;function from user32
extern ExitProcess ;from kernel32   ;function from kernel32
section .data                       ;section called .data
hello: db       'Hello, Windows!',0 ; string Hello, Windows!
title: db       'My First Win32',0  ;string My First Win32
section .text                       ;section called .text
start:                              ;start functions
        push MB_OK                  ;push MB_OK onto stack
        push title                  ;push title onto stack
        push hello                  ;push hello onto stack
```

```
push NULL                    ;push NULL onto stack
call MessageBoxA             ;call function MessageBoxA
push 0                       ;push 0 onto stack
call ExitProcess             ;call functions ExitProcess
```

b. Provide a screenshot of the commands compiling the code and linking to create the executable. (1 point)



c. Provide a screenshot of the results running the executable. (1 point)

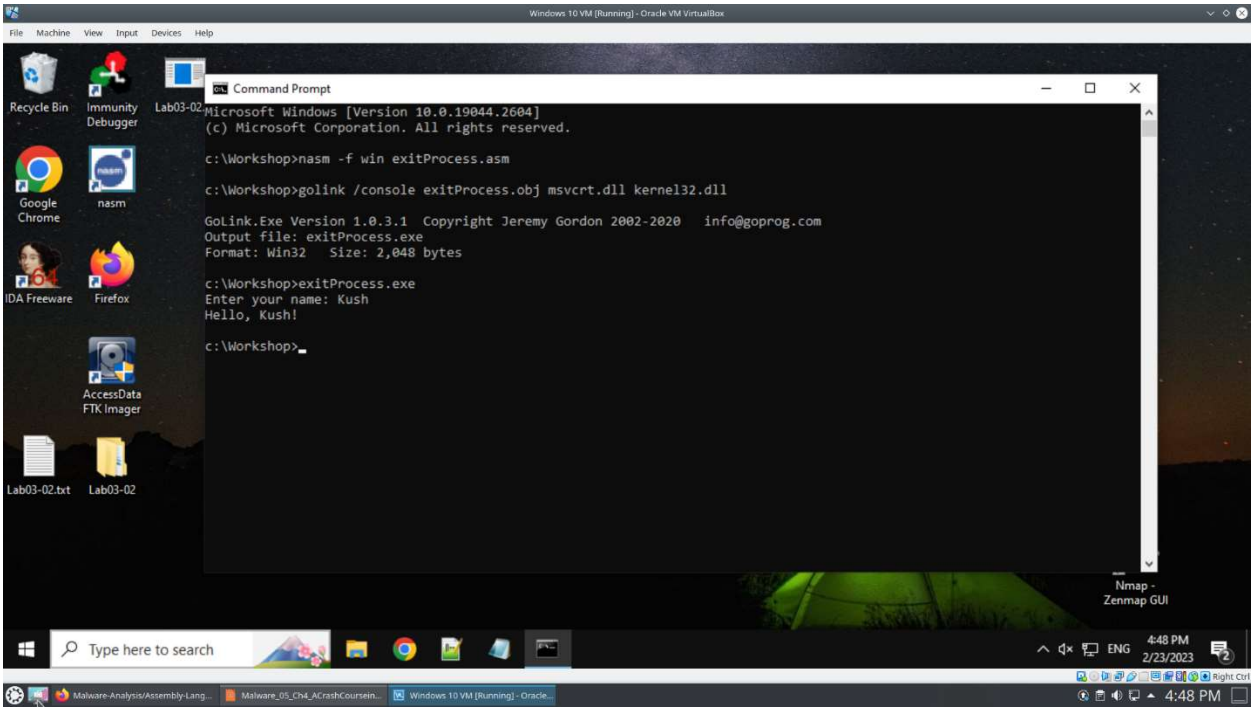


3. Copy the code in exitProcess.asm in the slides.
   a. Please give a note (comment, denoted by a semicolon ;) on each instruction (each line) explaining what it does and paste the code with the notes below. (1 point)

```
global start
extern printf                                   ;printf function from msvcrt
extern scanf                                    ;scanf function from msvcrt
extern ExitProcess                              ;ExitProcess functionsfrom kernel32
section .bss                                    ; section called .bss
name:  resb 100
section .data                                   ;section called .data
prompt:        db 'Enter your name: ',0         ;prompts you to enter name
frmt:   db '%s',0                               ;format of something
greet:  db 'Hello, %s!',0ah,0                   ;program greets you after entering name
section .text                                   ;section called .text
start:                                          ;start function
        push    prompt                          ;push prompt onto stack
        call    printf                          ;called printf function
        add     esp,4                           ;4 is added to esp
```

```
push    name        ;push name onto stack
push    frmt        ;push frmt onto stack
call    scanf       ;calls scanf function
add     esp,8       ; 8 is added to esp
push    name        ;push name onto stack
push    greet       ;push greet onto stack
call    printf      ;calls printf onto stack
add     esp,8       ; 8 is added to esp
push    0           ; 0 is pushed onto stack
call    ExitProcess ; calls ExitProcess function
```

b. Provide a screenshot of the commands compiling the code and linking to create the executable. (1 point)



c. Provide a screenshot of the results running the executable. (1 point)