Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO BÀI TẬP

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 01 (Session 01)

Tên chủ đề: PE Injection

GV: Nghi Hoàng Khoa

Ngày báo cáo: 17/03/2023

Nhóm: 7

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.N21.ANTN

STT	Họ và tên	MSSV	Email
1	Hoàng Đình Hiếu	20521317	20521317@gm.uit.edu.vn
2	Võ Anh Kiệt	20520605	20520605@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Kịch bản 01/Câu hỏi 01	100%	
2	Kịch bản 02	100%	
3	Kịch bản 03	100%	
4	Kịch bản 04	100%	
5	Kịch bản 05	100%	

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Kịch bản 01

Code tìm số min với 3 số cho sẵn:

Ở bài này phương pháp thực hiện là truyền giá trị số đầu tiên vào ecx xong kiểm tra với số thứ 2 nếu số 1 bé hơn bằng số 2 thì nhảy đến kiểm số 3 và nếu không thì truyền số 2 vào và nhảy đến số thứ 3.

Ở số thứ 3 ta thực hiện tương tự nếu giá trị trong thanh ghi ecx bé hơn hoặc bằng số 3 thì thì truyền vào biến smallest và xuất ra kết quả hoặc truyền số thứ 3 vào ecx vào smallest.

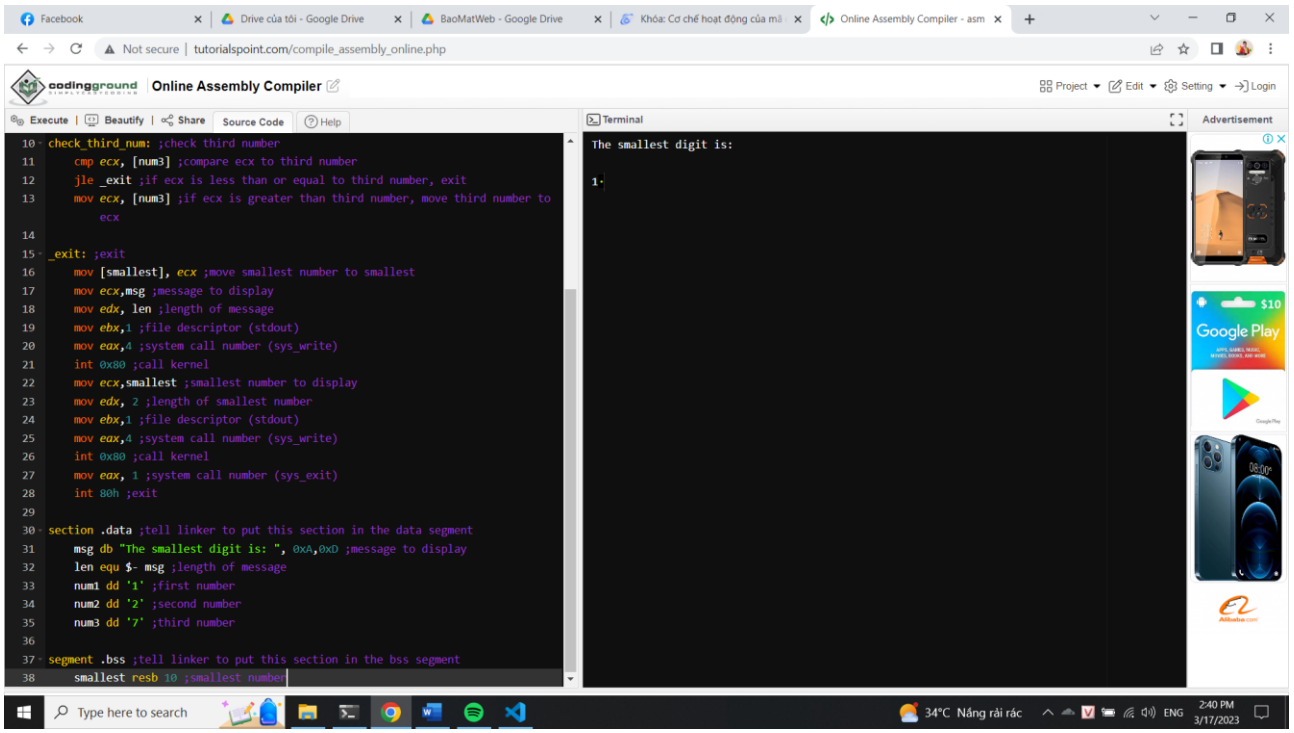
Cuối cùng xuất ra kết quả.

Code của chương trình (giải thích chi tiết được comment trong chương trình)

```
section .text: ;tell linker to put this section in the text segment
    global _start ;tell linker entry point

_start: ;tell linker entry point
    mov ecx, [num1] ;move first number to ecx
    cmp ecx, [num2] ;compare first number to second number
    jle check_third_num ;if first number is less than or equal to second number,
check third number
    mov ecx, [num2] ;if first number is greater than second number, move second
number to ecx

check_third_num: ;check third number
    cmp ecx, [num3] ;compare ecx to third number
    jle _exit ;if ecx is less than or equal to third number, exit
    mov ecx, [num3] ;if ecx is greater than third number, move third number to ecx

_exit: ;exit
    mov [smallest], ecx ;move smallest number to smallest
    mov ecx,msg ;message to display
    mov edx, len ;length of message
    mov ebx,1 ;file descriptor (stdout)
    mov eax,4 ;system call number (sys_write)
    int 0x80 ;call kernel
    mov ecx,smallest ;smallest number to display
    mov edx, 2 ;length of smallest number
    mov ebx,1 ;file descriptor (stdout)
    mov eax,4 ;system call number (sys_write)
    int 0x80 ;call kernel
    mov eax, 1 ;system call number (sys_exit)
```

```
    int 80h ;exit

section .data ;tell linker to put this section in the data segment
    msg db "The smallest digit is: ", 0xA,0xD ;message to display
    len equ $- msg ;length of message
    num1 dd '1' ;first number
    num2 dd '2' ;second number
    num3 dd '7' ;third number

segment .bss ;tell linker to put this section in the bss segment
    smallest resb 10 ;smallest number
```

Kết quả thực thi



2. Kịch bản 02

Ở bài này ta sẽ thực hiện code chuyển đổi số bằng việc chuyển đổi số mapping với số trong bảng mã acii để thực hiện chuyển đổi từ chuỗi số sang chuỗi ký tự và xuất kết quả ra màng hình

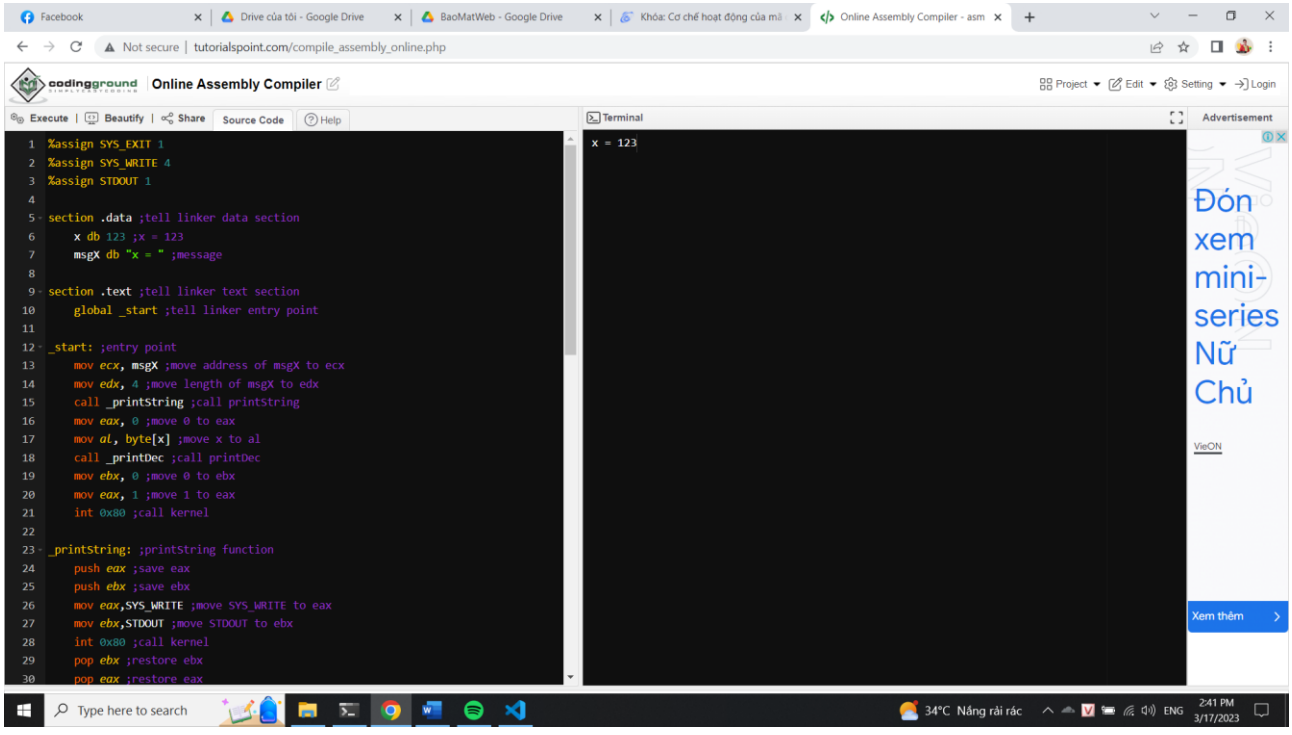
Code thực hiện (giải thích chi tiết được comment trong chương trình)

```
%assign SYS_EXIT 1
%assign SYS_WRITE 4
%assign STDOUT 1

section .data ;tell linker data section
    x db 123 ;x = 123
```

```
    msgX db "x = " ;message

section .text ;tell linker text section
    global _start ;tell linker entry point

_start: ;entry point
    mov ecx, msgX ;move address of msgX to ecx
    mov edx, 4 ;move length of msgX to edx
    call _printString ;call printString
    mov eax, 0 ;move 0 to eax
    mov al, byte[x] ;move x to al
    call _printDec ;call printDec
    mov ebx, 0 ;move 0 to ebx
    mov eax, 1 ;move 1 to eax
    int 0x80 ;call kernel

_printString: ;printString function
    push eax ;save eax
    push ebx ;save ebx
    mov eax,SYS_WRITE ;move SYS_WRITE to eax
    mov ebx,STDOUT ;move STDOUT to ebx
    int 0x80 ;call kernel
    pop ebx ;restore ebx
    pop eax ;restore eax
    ret ;return

_println:
    section .data ;tell linker data section
    .nl db 10 ;newline
    section .text ;tell linker text section
    push ecx ;save ecx
    push edx ;save edx
    mov ecx, .nl ;move address of .nl to ecx
    mov edx, 1 ;move length of .nl to edx
    call _printString ;call printString
    pop edx ;restore edx
    pop ecx ;restore ecx
    ret ;return

_printDec:
    ;;;saves all the registers so that they are not changed by the function
    section .bss
    .decstr resb 10
    .ct1 resd 1 ;to keep track of the size of the string
    section .text
    pushad ;save all registers
    mov dword[.ct1],0 ;assume initially 0
    mov edi,.decstr ;edi points to decstring
```

```
add edi,9 ;moved to the last element of string
xor edx,edx ;clear edx for 64-bit division
.whileNotZero:
mov ebx,10 ;get ready to divide by 10
div ebx ;divide by 10
add edx,'0' ;converts to ascii char
mov byte[edi],dl ;put it in sring
dec edi ;mov to next char in string
inc dword[.ct1] ;increment char counter
xor edx,edx ;clear edx
cmp eax,0 ;is remainder of division 0?
jne .whileNotZero ;no, keep on looping
inc edi ;conversion, finish, bring edi
mov ecx, edi ;back to beg of string. make ecx
mov edx, [.ct1] ;point to it, and edx gets # chars
mov eax, SYS_WRITE ;and print!
mov ebx, STDOUT ;print to stdout
int 0x80 ;call the kernel
popad ;restore all registers
ret
```

Kết quả thực thi



3. Kịch bản 03

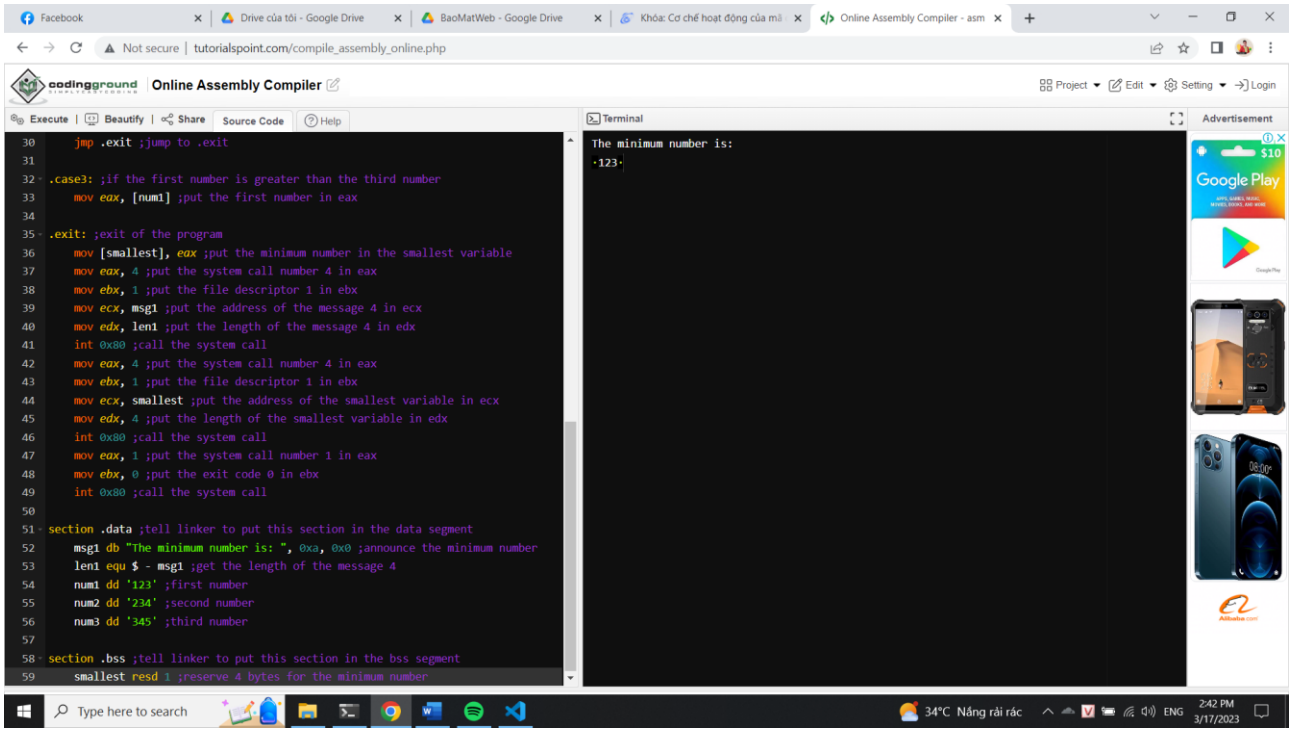
Ở trong trường hợp này cũng như kịch bản 1 ta sẽ thực hiện so sánh từng cặp số với nhau và nhảy vào các case khác nhau để tìm ra số min và thực hiện in ra màng hình, với thuật toán này ta có thể xuất ra nhiều digit khác nhau.

Code thực hiện: (giải thích chi tiết trong phần comment của code)

```
section .text ;tell linker to put this section in the text segment
    global _start ;tell linker to start at _start

_start: ;start of the program
    mov eax, [num1] ;put the first number in eax
    mov ebx, [num2] ;put the second number in ebx
    cmp eax, ebx ;compare the first number and the second number
    jle .case1 ;if the first number is less than or equal to the second number,
jump to .else
    mov eax, [num2] ;put the second number in eax
    mov ebx, [num3] ;put the third number in ebx
    cmp eax, ebx ;compare the second number and the third number
    jle .case2 ;if the second number is less than or equal to the third number,
jump to .case2
    mov eax, [num3] ;put the third number in eax
    jmp .exit ;jump to .exit

.case1: ;if the first number is greater than the second number
    mov eax, [num1] ;put the first number in eax
    mov ebx, [num3] ;put the third number in ebx
    cmp eax, ebx ;compare the first number and the third number
    jle .case3 ;if the first number is less than or equal to the third number, jump
to .case3
    mov eax, [num3] ;put the third number in eax
    jmp .exit ;jump to .exit

.case2: ;if the second number is greater than the third number
    mov eax, [num1] ;put the first number in eax
    mov ebx, [num2] ;put the second number in ebx
    cmp eax, ebx ;compare the first number and the second number
    jle .case3 ;if the first number is less than or equal to the second number,
jump to .case3
    mov eax, [num2] ;put the second number in eax
    jmp .exit ;jump to .exit

.case3: ;if the first number is greater than the third number
    mov eax, [num1] ;put the first number in eax

.exit: ;exit of the program
    mov [smallest], eax ;put the minimum number in the smallest variable
    mov eax, 4 ;put the system call number 4 in eax
    mov ebx, 1 ;put the file descriptor 1 in ebx
    mov ecx, msg1 ;put the address of the message 4 in ecx
    mov edx, len1 ;put the length of the message 4 in edx
    int 0x80 ;call the system call
    mov eax, 4 ;put the system call number 4 in eax
```

```
    mov ebx, 1 ;put the file descriptor 1 in ebx
    mov ecx, smallest ;put the address of the smallest variable in ecx
    mov edx, 4 ;put the length of the smallest variable in edx
    int 0x80 ;call the system call
    mov eax, 1 ;put the system call number 1 in eax
    mov ebx, 0 ;put the exit code 0 in ebx
    int 0x80 ;call the system call

section .data ;tell linker to put this section in the data segment
    msg1 db "The smallest digit is: ", 0xa, 0x0 ;announce the minimum number
    len1 equ $ - msg1 ;get the length of the message 4
    num1 dd '123' ;first number
    num2 dd '234' ;second number
    num3 dd '345' ;third number

section .bss ;tell linker to put this section in the bss segment
    smallest resd 1 ;reserve 4 bytes for the minimum number
```

Kết quả thực thi



4. Kịch bản 04

Đã báo cáo trên lớp

5. Kịch bản 05

Clip thực hiện: https://youtu.be/Yl-_F6b4hbg

Notepad

Tính toán

Infor:

10D50 – 8400 = X – B000

X = 13950

Add image base = 01013950 (50 39 01 01)

Text:

10D60 – 8400 = Y – B000

Y = 13960

Add image base = 01013960 (60 39 01 01)

Code:

New entry point:

10DA0 – 8400 + B000 = 139A0

Old_entry_point = jmp + 5 + R_VA

0100739D = 000139A0 + image base + 14 + 5 + R_VA

R_VA = FFFF39E4 (E4 39 FF FF)

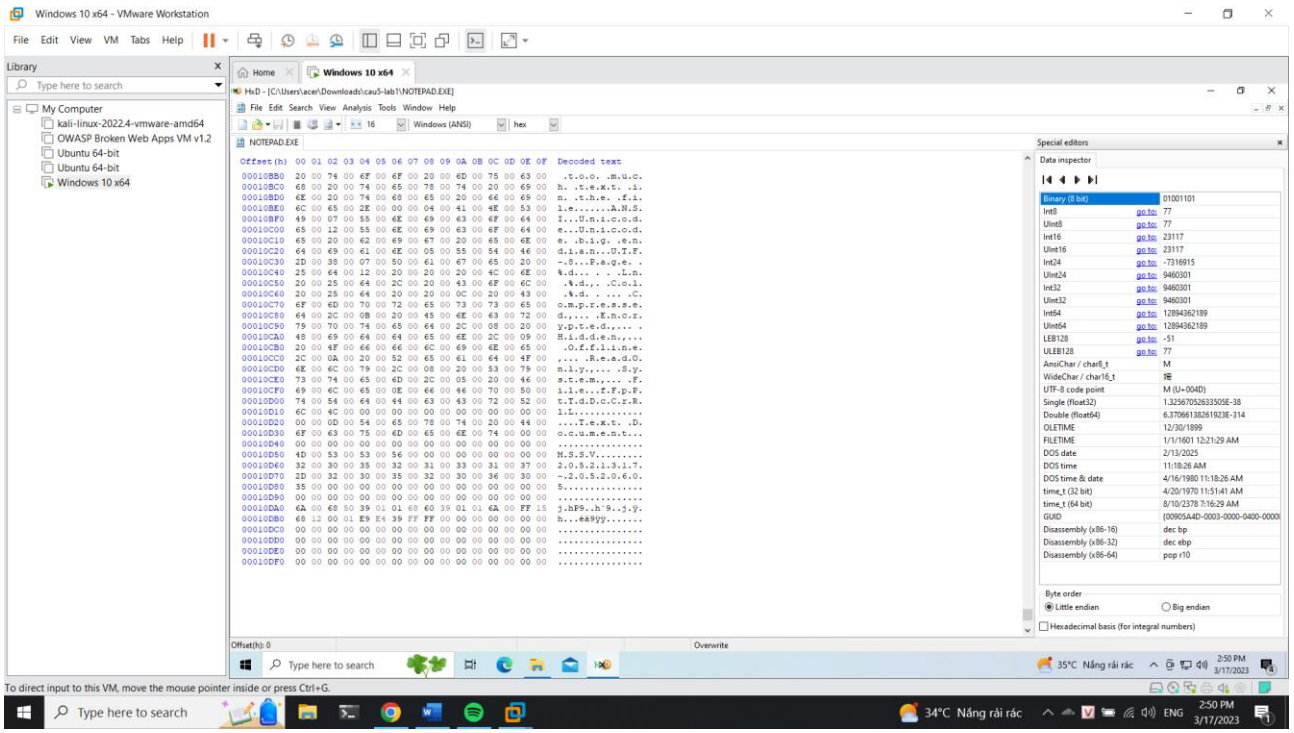
Payload

6A 00 68 50 39 01 01 68 60 39 01 01 6A 00 FF 15 68 12 00 01 E9 E4 39 FF FF

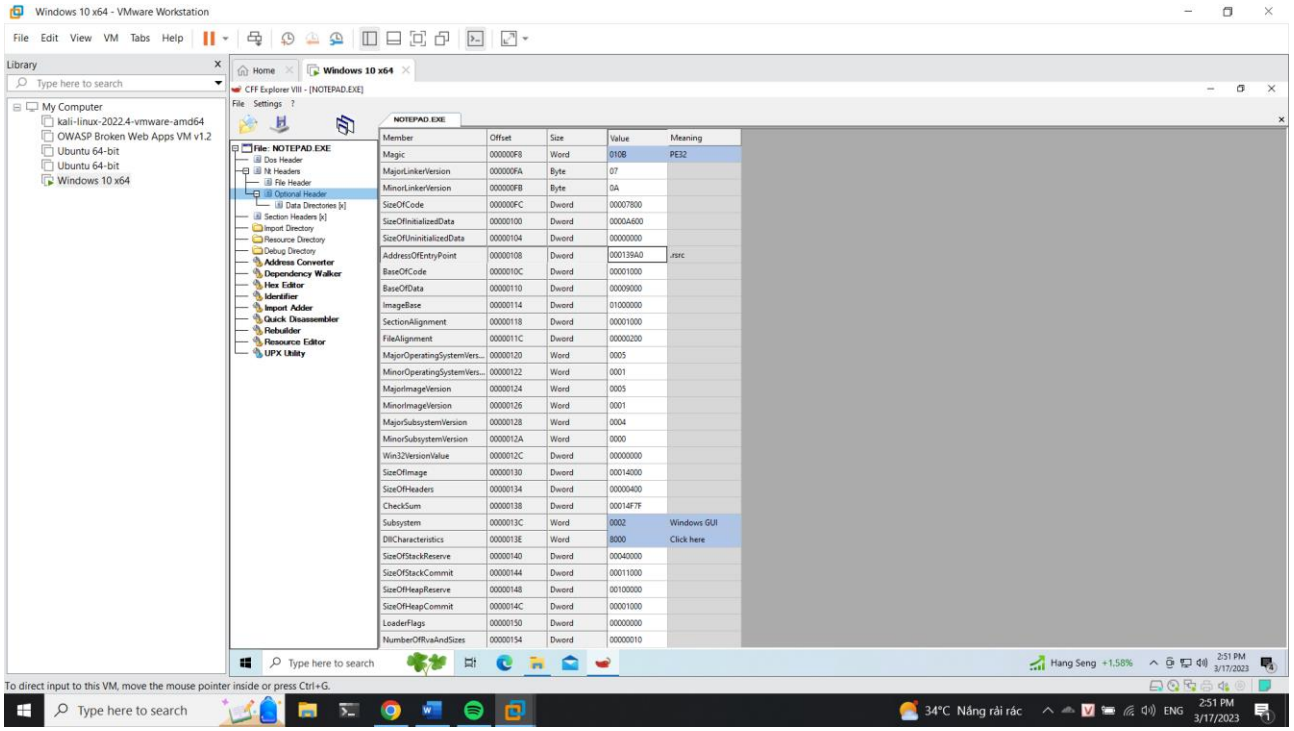
4D 00 53 00 53 00 56 00 (MSSV)

32 00 30 00 35 00 32 00 31 00 33 00 31 00 37 00 2D 00 32 00 30 00 35 00 32 00 30 00 36 00 30 00 35 00 (20521317-20520605)

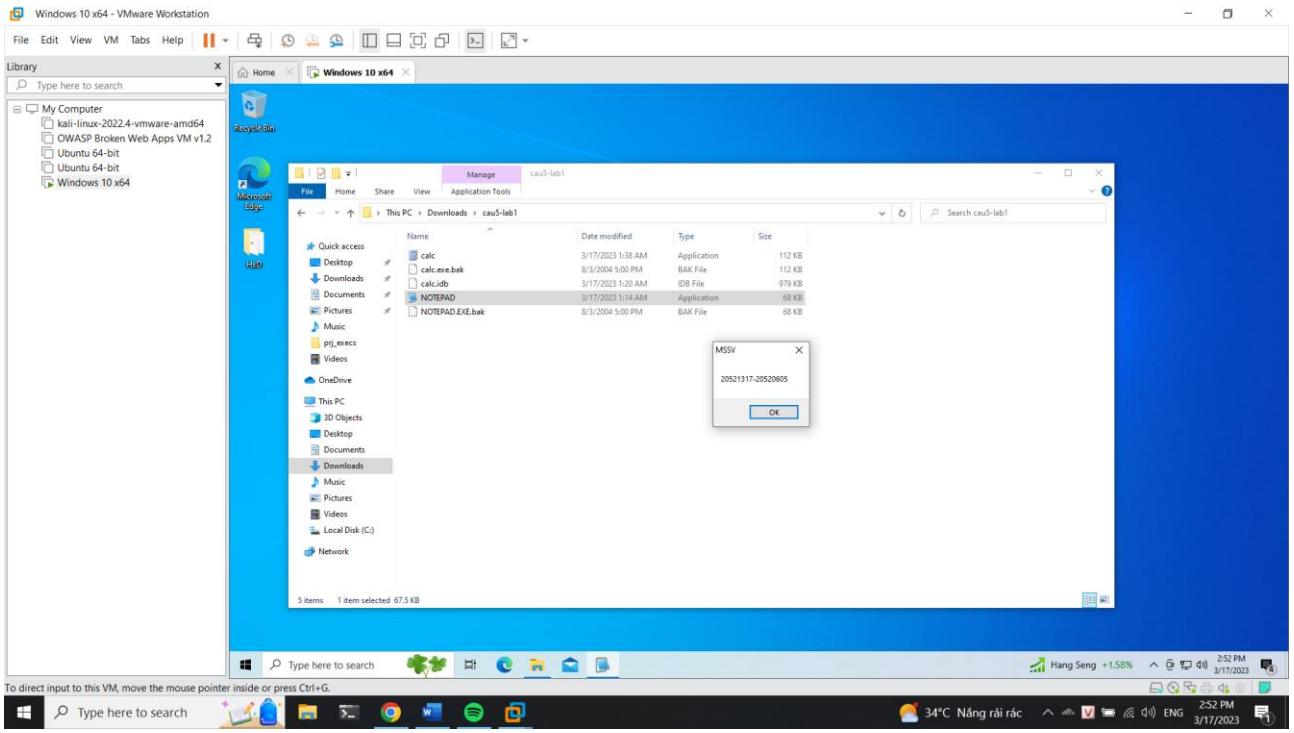
Chèn vào chương trình

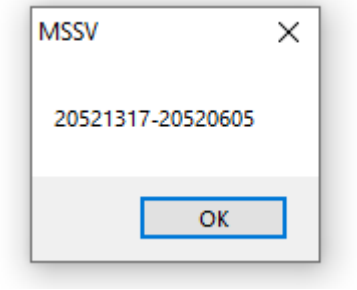


Chỉnh lại Address of entry point



Thực hiện chạy


Windows 10 x64 - VMware Workstation
MSSV
20521317-20520605
OK


MSSV
20521317-20520605
OK

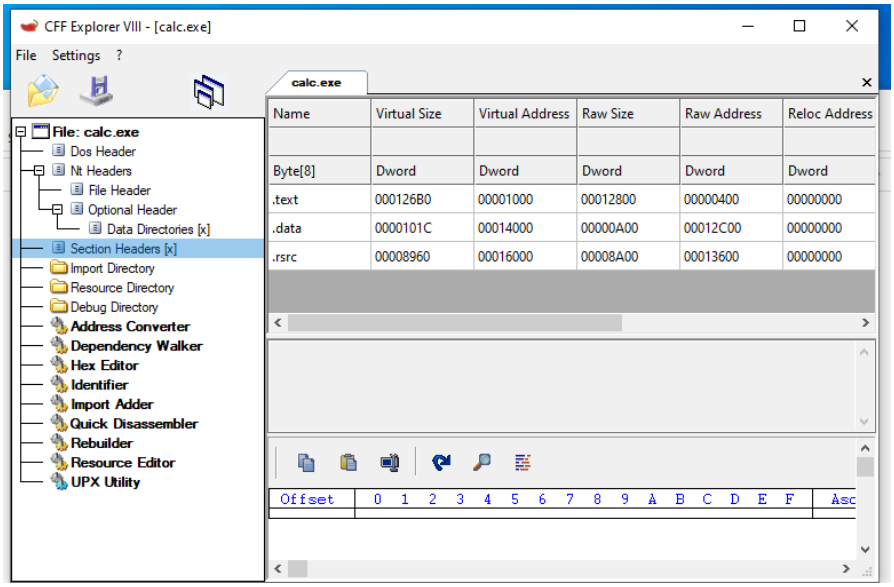
Calc

Kiểm tra các giá trị

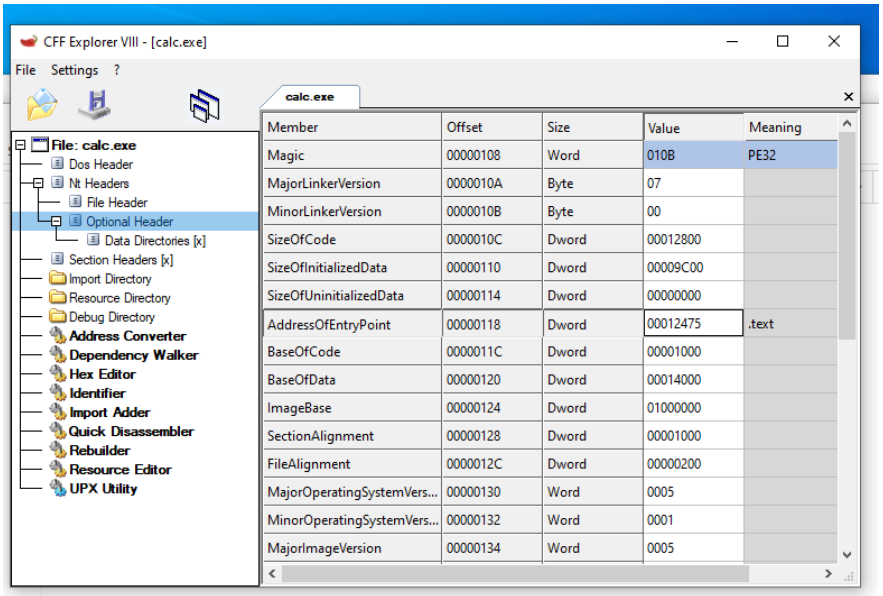
Message BoxW 010011A8

Virtual addresse: 16000

Raw address: 13600



Address of entry point



Tính toán

Info:

1BF70 – 13600 = X – 16000

X = 1E970

Add image base 0101E970 (70 E9 01 01)

Text:

1BF80 – 13600 = Y – 16000

Y = 1E980

Add image base 0101E980 (80 E9 01 01)

Code:

New entry point:

1BF50 – 13600 + 16000 = 1E950

Old entry point = jmp + Virtual address + 5 + R_VA

01012475 = 0001E950 + image base + 14 + 5 + + R_VA

R_VA = FFFF 3B 0C

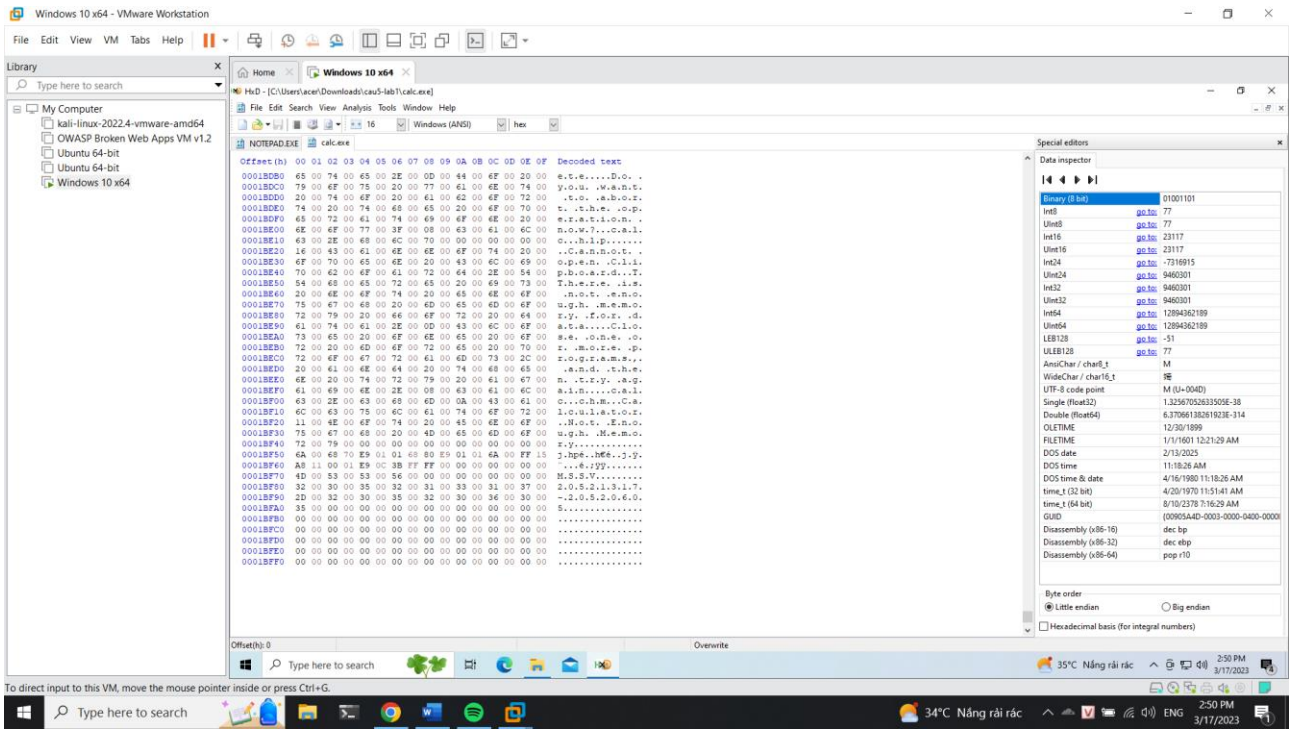
Payload

6A 00 68 70 E9 01 01 68 80 E9 01 01 6A 00 FF 15 A8 11 00 01 E9 0C 3B FF FF

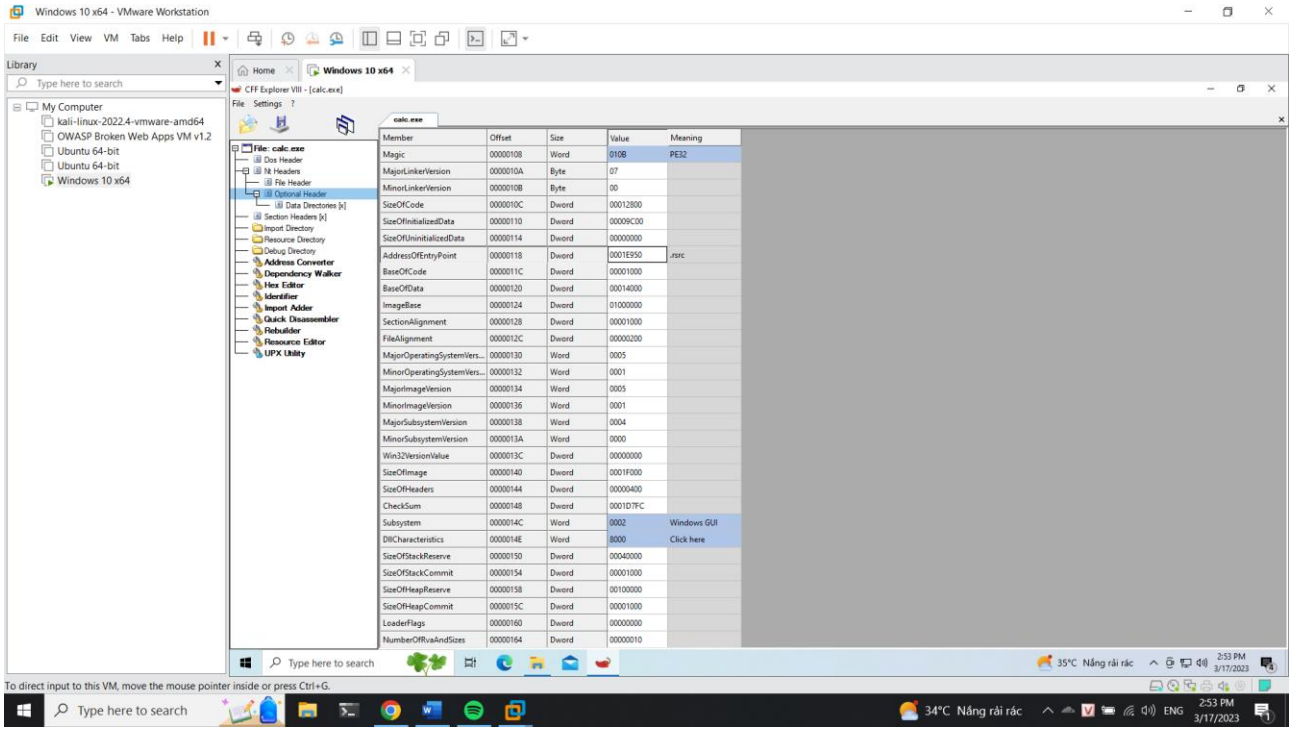
4D 00 53 00 53 00 56 00 (MSSV)

32 00 30 00 35 00 32 00 31 00 33 00 31 00 37 00 2D 00 32 00 30 00 35 00 32 00 30 00 36 00 30 00 35 00 (20521317-20520605)

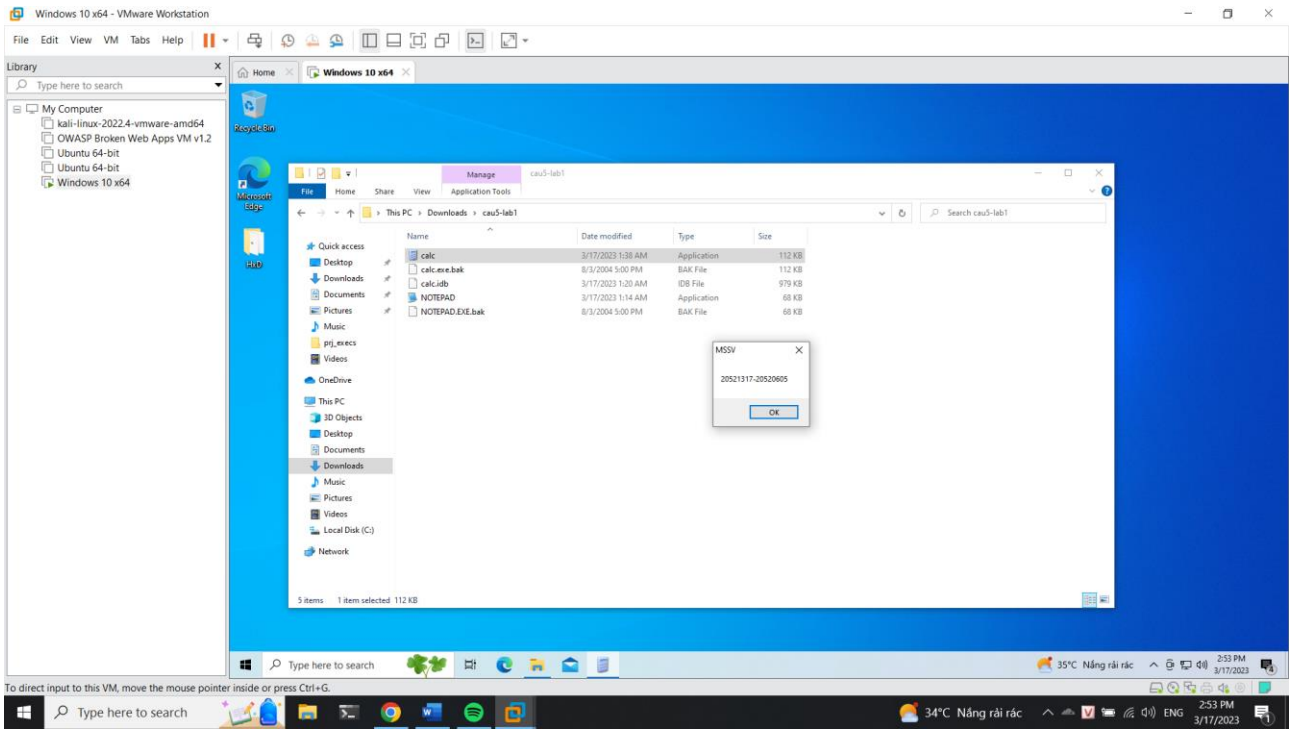
Chèn vào chương trình

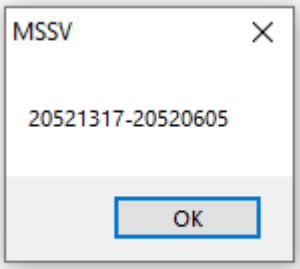


Chỉnh lại Address of entry point



Thực hiện chạy





Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
- Đặt tên theo định dạng: [Mã lớp]-SessionX_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành/Tên Cá nhân đã đăng ký với GV).

  *Ví dụ: [*NT101.K11.ANTT*]-Session1_Group3.*
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Không đặt tên đúng định dạng – yêu cầu, sẽ **KHÔNG** chấm điểm.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá: Sinh viên hiểu và tự thực hiện. Khuyến khích:

- Chuẩn bị tốt.
- Có nội dung mở rộng, ứng dụng trong kịch bản/câu hỏi phức tạp hơn, có đóng góp xây dựng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT