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

Nhóm: 8

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

STT	Họ và tên	MSSV	Email
1	Nguyễn Đình Luân	21521105	21521105@gm.uit.edu.vn
2	Trần Thanh Triều	21522713	21520713@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	0%	

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

Bài 1: Viết một đoạn chương trình tìm số nhỏ nhất trong 3 số (1 chữ số) a,b,c cho trước.

Khai báo 3 biến muốn so sánh num1,num2,num3 ,biến smallest lưu giá trị bé nhất trong 3 biến này và msg là chuỗi "The smallest digit is:"

```
section .data
 msg db "The smallest digit is: ", 0xA,0xD
 len equ $- msg
 num1 dd '6'
 num2 dd '7'
 num3 dd '8'
segment .bss
 smallest resb 2
```

So sánh : .

- Nếu num1 < num2 thì xuống so sánh với num3
    - o Nếu num1 < num3 thì num1 là số bé nhất và nhảy đến hàm exit
    - o Num 1 > num3 thì num3 là số bé nhất thì gán ecx = num3 rồi cũng đến hàm exit
- Nếu num1> num2 thì gán ecx= num2 và so sánh num2 với num3
    - o Nếu num2 < num3 thì num2 là min
    - o Nếu num2 > num3 thì num3 là min và ecx = num3

Jl : lệnh jump được thực thi khi giá trị so sánh của lệnh cmp bé hơn.

```

 section .text
 global _start ;must be declared for using gcc
_start: ;tell linker entry point
 mov ecx, [num1]
 cmp ecx, [num2]
 jl check_third_num
 mov ecx, [num2]

check_third_num:
 cmp ecx, [num3]
 jl _exit
 mov ecx, [num3]

 _exit:
```

- Xuất chuỗi đã khai báo ở đầu chương trình

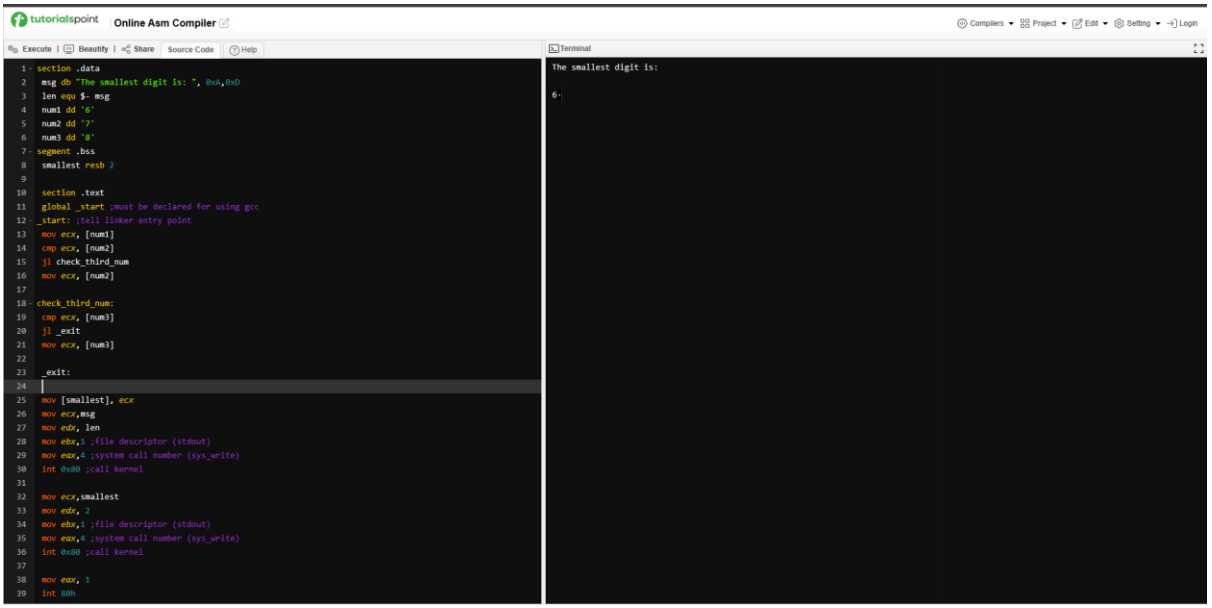
- Gán smallest = ecx

- Xuất giá trị min ra

ð Kết quả :

```
25  mov [smallest], ecx
26  mov ecx,msg
27  mov edx, len
28  mov ebx,1 ;file descriptor (stdout)
29  mov eax,4 ;system call number (sys_write)
30  int 0x80 ;call kernel
31
32  mov ecx,smallest
33  mov edx, 2
34  mov ebx,1 ;file descriptor (stdout)
35  mov eax,4 ;system call number (sys_write)
36  int 0x80 ;call kernel
37
38  mov eax, 1
39  int 80h
```

Kết quả:



Bài 2: Viết chương trình chuyển đổi một số (number) 123 thành chuỗi ‘123’ Sau đó thực hiện in ra màn hình số 123.

Khai báo x để chuyển thành chuỗi và in ra number_str là 1 biến dùng để lưu chuỗi đó

```
1   %assign SYS_EXIT 1
2   %assign SYS_WRITE 4
3   %assign STDOUT 1
4
5   ;;; ------------------------------------------------------
6   ;;; data section
7   ;;; ------------------------------------------------------
8   section .data
9   x         db        123
10  msgX      db        "x = "
11  len_msgX equ        $- msgX
12
13  segment .bss
14  number_str resb     4
15
```

Xuất “x=” ra màn hình

```
;;; xuất X= ra màn hình
mov ecx, msgX
mov edx, len_msgX
mov eax, SYS_WRITE
mov ebx, STDOUT
int 0x80
```

- Di chuyển giá thị của x vào thanh ghi al , 100 vào ebx .

- Thực hiện eax / ebx bằng “div ebx”

- Sau khi chia thì eax sẽ lưu giá trị nguyên của phép chia , edx sẽ lưu phần dư

(123 / 100 = 1 dư 23)

ð Eax = 1 , edx=23

- Đổi eax từ số nguyên sang kí tự

- Gán kí tự đầu tiên của biến number_str là : 1

```
;; pass dividend to %eax                %eax=123
mov al, [x]

;;;;;;;;;;;;;;;;;;;;;;;;

;;pass divisor to (%ebx = 100)
mov ebx, 100

;; eax = 1 , edx = 23
cdq
div ebx

;; remainder in %edx, result in %eax

;; eax = "1"
add eax, '0'

;; save number to memory for printing
;; %ebx = $number_str
mov byte [number_str], al
```

- Gán eax = edx (=23)

- Xóa giá trị hiện tại của edx

- Gán ebx = 10 để tiếp tục thực hiện phép chia

- Tương tự thì ta có (eax = 2 , edx =3)

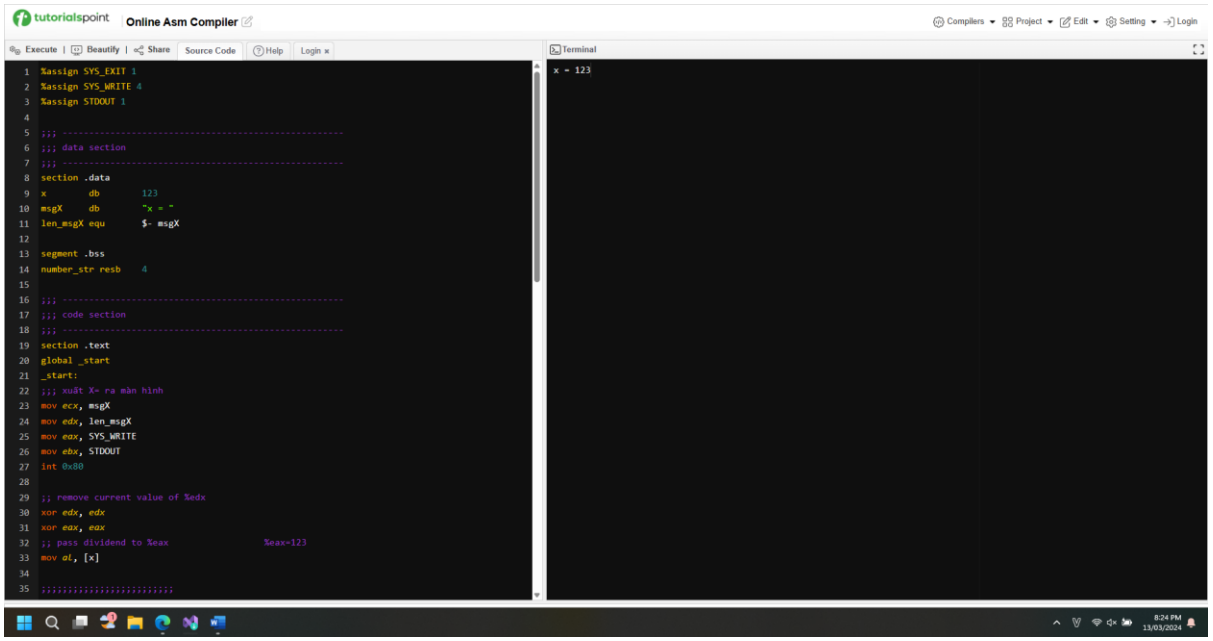
- Đổi cả 2 số nguyên 2 và 3 sang kí tự và gán vào kí tự thứ 2 và 3 của number_str

```
;; eax = 23
mov eax, edx

;; edx = 0
xor edx, edx

;; ebx = 10
mov ebx, 10

;; eax : ebx
cdq
div ebx
;; => eax = 2, edx = 3

;; save digits to memory
add eax, '0'
add edx, '0'
mov byte [number_str + 1], al
mov byte [number_str + 2], dl
```

Xuất biến number_str ra và chúng ta có : 123

```
mov eax, SYS_WRITE
mov ebx, STDOUT
mov ecx, number_str
mov edx, 3
int 0x80

xor eax,eax
xor ebx,ebx
xor ecx,ecx
xor edx,edx

mov eax, SYS_EXIT
xor ebx, ebx
int 0x80
```

Kết quả:



Bài 3: Cải tiến chương trình yêu cầu 1 sao cho tìm số nhỏ nhất trong 3 số bất kỳ (nhiều hơn 1 chữ số)

Khai báo các chuỗi để xuất ra màn hình

```
SYS_EXIT equ 1
SYS_READ equ 3
SYS_WRITE equ 4
STDIN equ 0
STDOUT equ 1
segment .data
 msg1 db "Enter a number ", 0xA,0xD
 len1 equ $- msg1

 msg2 db "Please enter a second number", 0xA,0xD
 len2 equ $- msg2

 msg3 db "Please enter a third number: ",0xA,0xD
 len3 equ $- msg3

 msg4 db "The smallest one is : "
 len4 equ $- msg4

```

Các biến dùng để nhập vào và so sánh

```
segment .bss
 num1 resb 10
 num2 resb 10
 num3 resb 10
 res resb 10
```

Nhập 2 biến num, num2 và num 3

```
mov eax, SYS_WRITE
mov ebx, STDOUT
mov ecx, msg1
mov edx, len1
int 0x80
mov eax, SYS_READ
mov ebx, STDIN
mov ecx, num1
mov edx, 10
int 0x80

mov eax, SYS_WRITE
mov ebx, STDOUT
mov ecx, msg2
mov edx, len2
int 0x80
mov eax, SYS_READ
mov ebx, STDIN
mov ecx, num2
mov edx, 10
int 0x80


mov eax, SYS_WRITE
mov ebx, STDOUT
mov ecx, msg3
mov edx, len3
int 0x80
mov eax, SYS_READ
mov ebx, STDIN
mov ecx, num3
mov edx, 10
int 0x80
```

So sánh : .

- Nếu num1 < num2 thì xuống so sánh với num3

o Nếu num1 < num3 thì num1 là số bé nhất và nhảy đến hàm exit

§ Num 1 > num3 thì num3 là số bé nhất thì gán ecx = num3 rồi cũng đến hàm exit

- Nếu num1> num2 thì gán ecx= num2 và so sánh num2 với num3

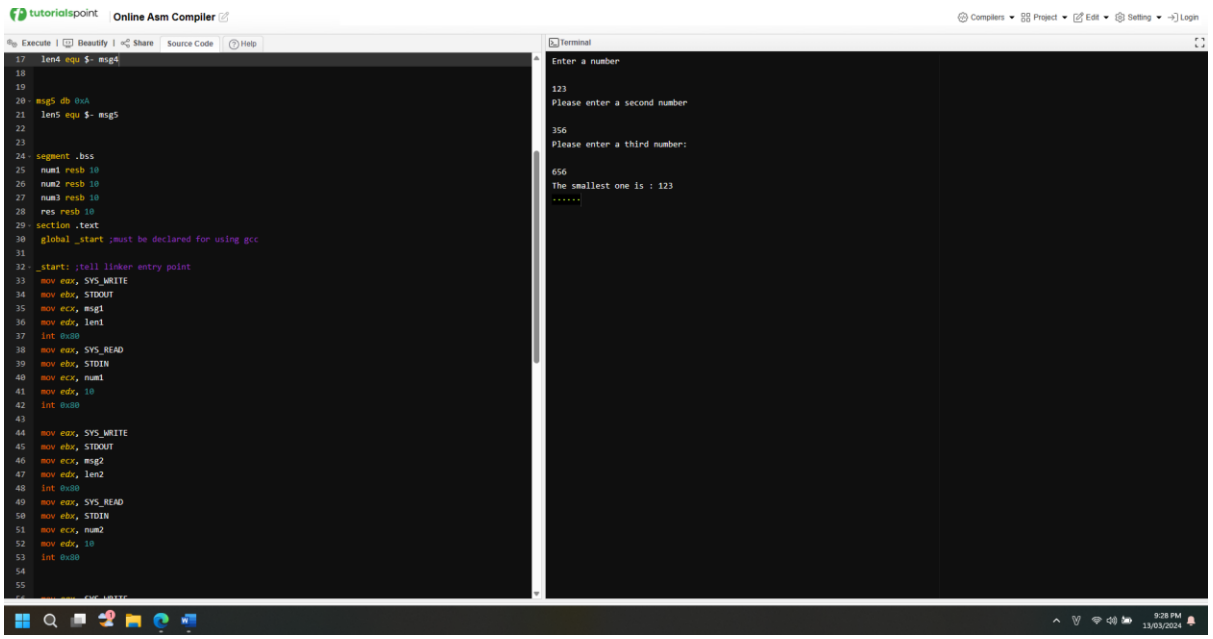
o Nếu num2 < num3 thì num2 là min

o Nếu num2 > num3 thì num3 là min và ecx = num3

Xuất "The smallest one is : " (msg4) và res

```
mov ecx, [num1]
cmp ecx, [num2]
jl check_third_num
mov ecx, [num2]

check_third_num:
cmp ecx, [num3]
jl _exit
mov ecx, [num3]

_exit:

mov [res], ecx
mov ecx,msg4
mov edx, len4
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)
int 0x80 ;call kernel

mov ecx,res
mov edx, 10
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)
int 0x80 ;call kernel

mov eax, 1
int 80h
```

Kết quả:



Bài 4:

ImageBase = 0x01000000

AddressOfEntryPoint = 0x0000739D

Old = ImageBase+ AddressOfEntryPoint = 0x0100739D

RA = 00008400

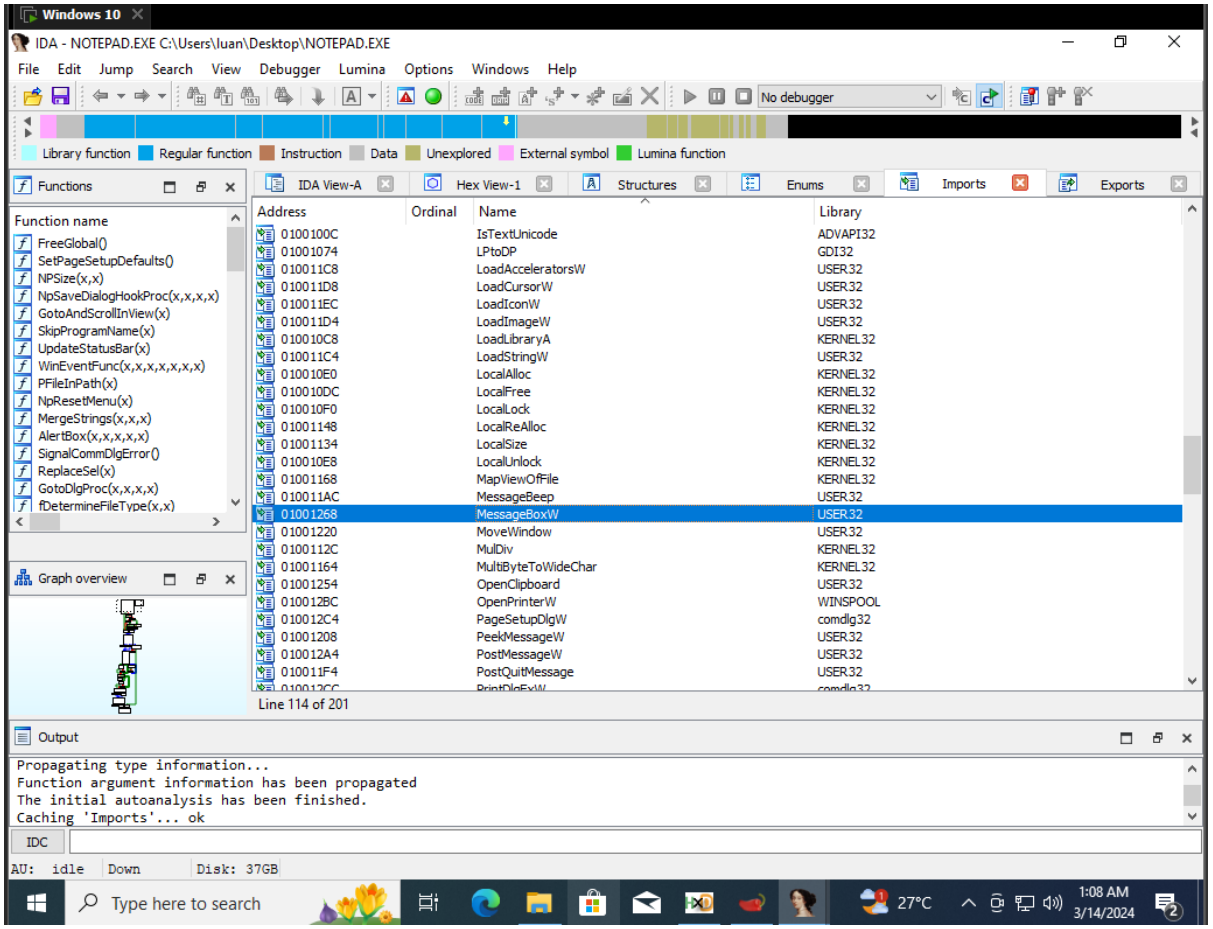
VA = 0000B000

Về cơ bản, chương trình gồm 5 dòng lệnh (sử dụng chức năng hexview để xem mã hex của từng lệnh).

push 0 ;6a 00
push Caption ; 68 X
push Text ; 68 Y
push 0 ;6a 00
call [MessageBoxW] ; ff15 Z

Để chèn đoạn code này vào Notepad.exe, ta phải đi tìm các giá trị (X, Y, Z) phù hợp.

Giá trị Z chính là địa chỉ của hàm MessageBoxW được import từ thư viện USER32.dll. Trong IDA Pro, mở Notepad.exe, chọn View -> Open Subviews -> Imports và ta thấy địa chỉ của hàm MessageBoxW chính là 01001268



Trong HxD, ta chọn địa chỉ 0x00011000 trong vùng nhớ đã được mở rộng (bước C1) để lưu trữ mã hợp ngữ, 0x00011040 để lưu trữ Caption và 0x00011060 để lưu trữ Text.

Offset = RA – Section RA = VA – Section VA
Giá trị X có thể được tìm dựa vào công thức (1) :
0x00011040 – 0x00008400 = X – 0x000B000 X = 0x00013C40
Cộng thêm ImageBase, suy ra X = 0x01013C40. Tương tự , Y = 0x01013C60.

Như vậy, đoạn code này thực hiện chức năng như mong đợi và có địa chỉ mới là: new_entry_point = 0x00011000 – 0x00008400 + 0x000B000 = 0x00013C00

Để chương trình Notepad.exe tiếp tục được thực thi sau khi đã chạy đoạn code trên, ta cần chèn dòng lệnh quay về AddressOfEntryPoint cũ ngay sau đoạn code ở bước 2. jmp relative_VA

Đói với lệnh jmp, đích đến (old_entry_point) sẽ được tính bằng cách cộng giá trị relative_VA vào thanh ghi PC khi lệnh được thực thi. Bởi vì PC luôn trỏ đến vị trí đầu của câu lệnh kế tiếp cho nên cần phải tính 5 bytes của câu lệnh jmp nữa. Ta có công thức sau

old_entry_point = jmp_instruction_VA + 5 + relative_VA (2)

Néu đặt lệnh jmp sau 5 câu lệnh ở bước 2 thì jmp_instruction_VA = 0x01013C14

old_entry_point = 0x0100739D chính là giátriAddressOfEntryPoint ban đầu đã cộng ImageBase.

Suy ra, relative_VA = 0x0100739D – 5 – 0x01013C14 = 0xFFFF3784.

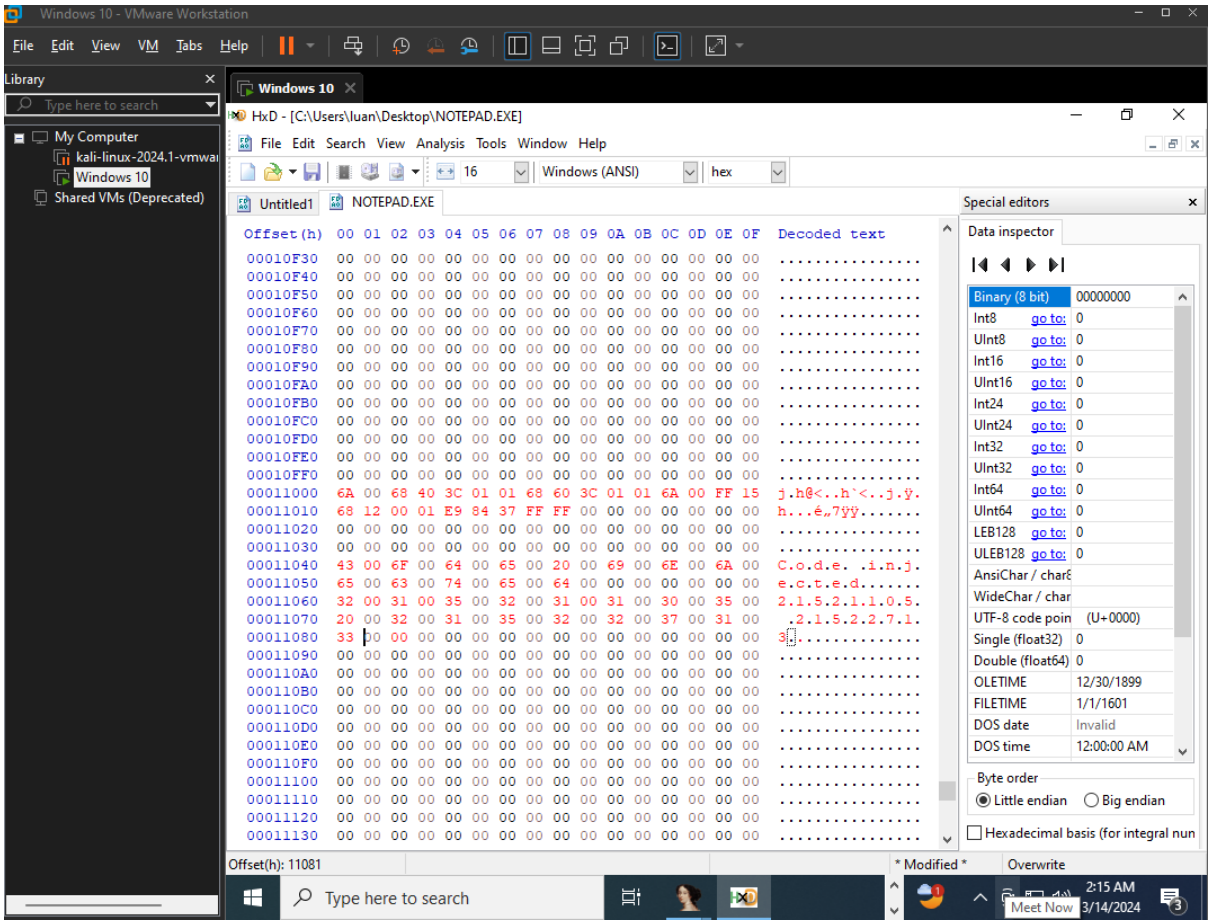
Ta có hợp ngữ hoàn chỉnh

push 0 ; 6a 00
push Caption ; 68 403C0101
push Text ; 68 603C0101
push 0 ; 6a00
call [MessageBoxW] ; ff15 68120001
jmp Origianl_Entry_Point ; e9 8437FFFF

Text: 32 31 35 32 31 31 30 35 20 32 31 35 32 32 37 31 33

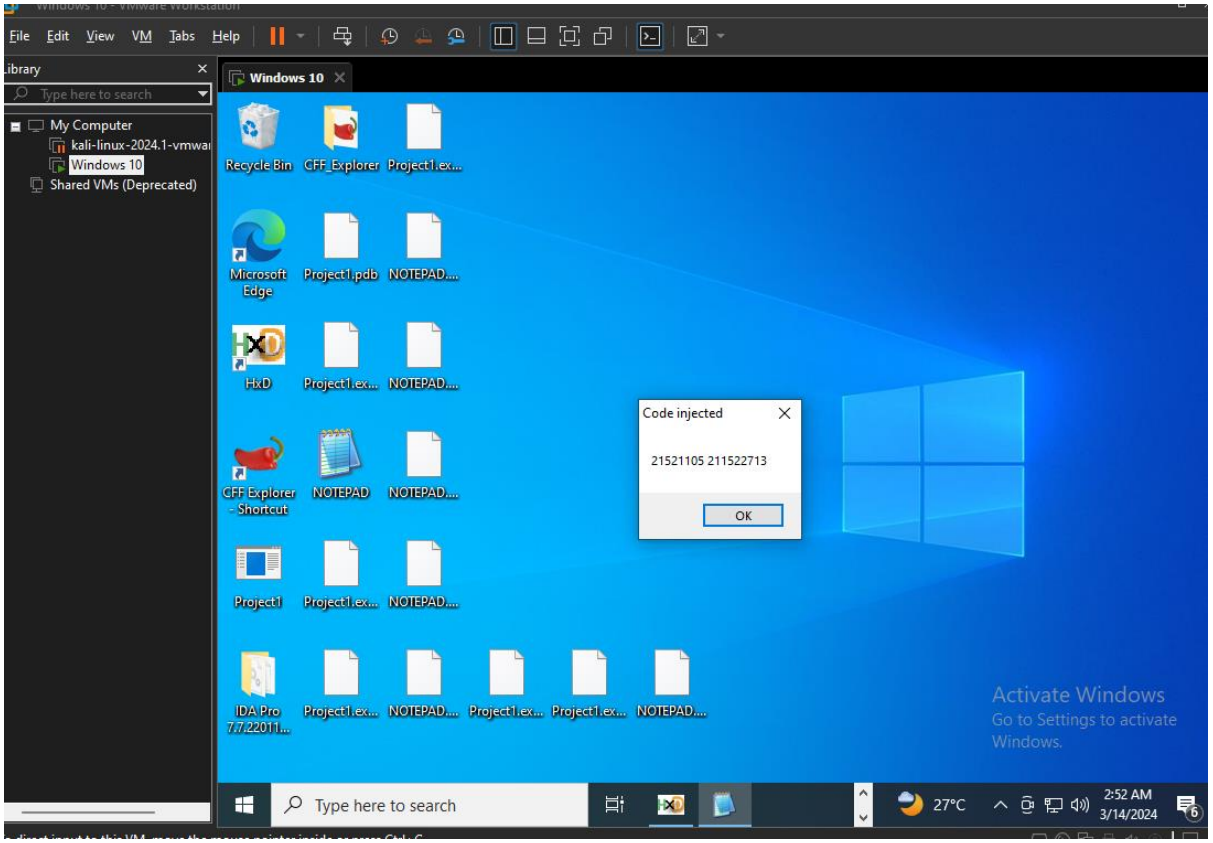
Caption: 43 6f 64 65 20 69 6e 6a 65 63 74 65 64

Chèn vào Notepad.exe:



5. Hiệu chỉnh các tham số trong PE header

Kết quả:



Link video: https://youtu.be/CoGK1AYoTBU