

BÁO CÁO THỰC HÀNH

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 01 (Session 01)

Tên chủ đề: Ôn tập ngôn ngữ assembly và chèn mã vào tập tin PE

GVHD: Nghi Hoàng Khoa

Ngày báo cáo: 19/03/2024

Nhóm: 09

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.021.ANTN

STT	Họ và tên	MSSV	Email
1	Hà Thị Thu Hiền	21522056	21522056@gm.uit.edu.vn
2	Phạm Ngọc Thơ	21522641	21522641@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá
1	Bài thực hành 01	100%
2	Bài thực hành 02	100%
3	Bài thực hành 03	100%
4	Bài thực hành 04 (đã báo cáo tại lớp)	100%
5	Bài thực hành 05	100%

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

BÁO CÁO CHI TIẾT

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

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

- Code:

```
section .text
    global _start          ; Khai báo _start là global để có thể sử dụng với gcc

_start:                   ; Điểm nhập của trình liên kết
    mov     ecx, [num1]    ; Di chuyển giá trị của num1 vào thanh ghi ecx
    cmp     ecx, [num2]    ; So sánh giá trị của num1 và num2
    jl      check_third_num ; Nếu num1 < num2, nhảy đến nhãn check_third_num
    mov     ecx, [num2]    ; Nếu không, di chuyển giá trị của num2 vào thanh ghi ecx

check_third_num:          ; Nhãn cho điểm kiểm tra số thứ ba

    cmp     ecx, [num3]    ; So sánh giá trị của ecx và num3
    jl      _exit          ; Nếu ecx < num3, nhảy đến nhãn _exit
    mov     ecx, [num3]    ; Nếu không, di chuyển giá trị của num3 vào thanh ghi ecx

_exit:                    ; Nhãn cho điểm kết thúc

    mov     [min_num], ecx ; Lưu giá trị nhỏ nhất vào min_num
    mov     ecx, msg       ; Di chuyển địa chỉ của msg vào thanh ghi ecx
    mov     edx, len       ; Di chuyển độ dài của msg vào thanh ghi edx
    mov     ebx, 1         ; File descriptor (stdout)
    mov     eax, 4         ; System call number (sys_write)
    int     0x80           ; Gọi kernel để in ra chuỗi

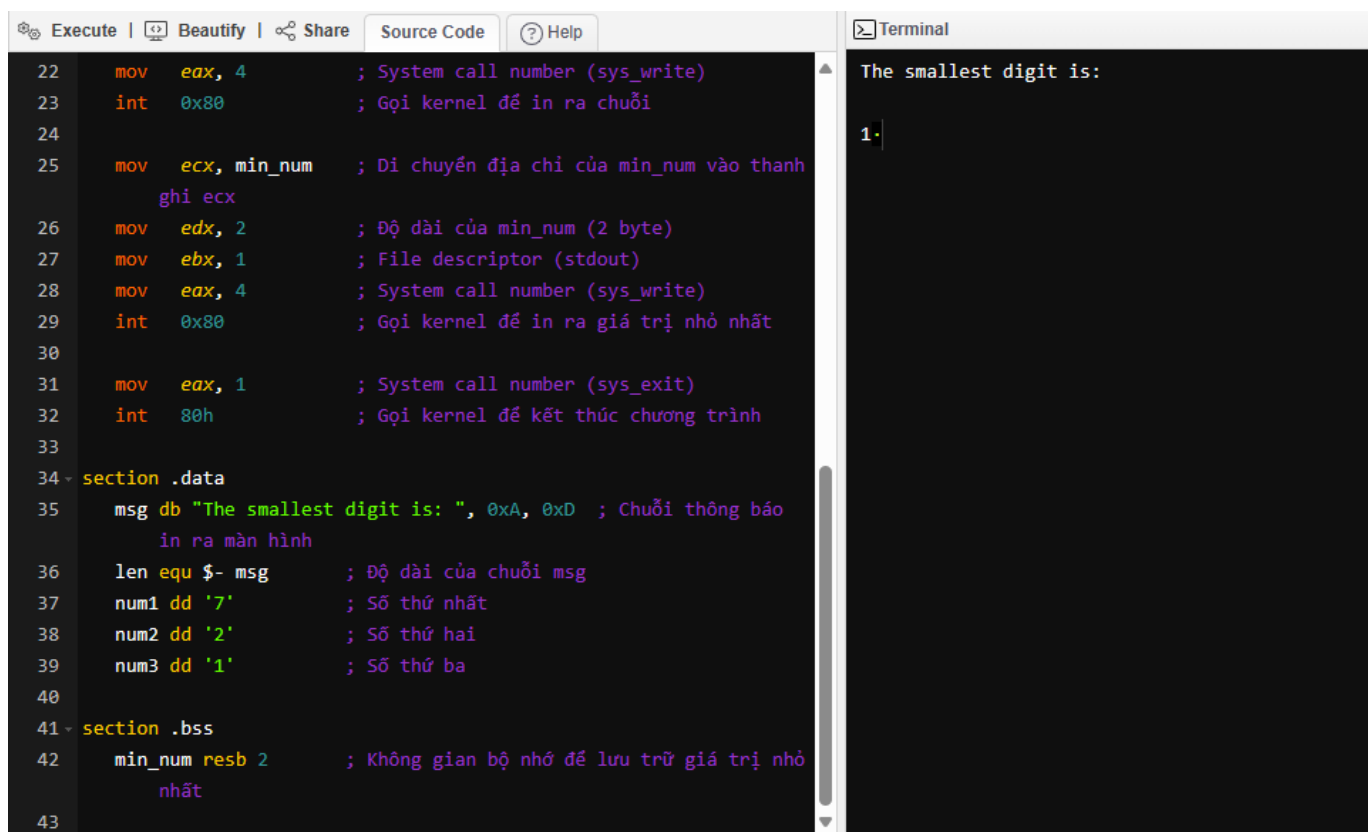
    mov     ecx, min_num   ; Di chuyển địa chỉ của min_num vào thanh ghi ecx
    mov     edx, 2         ; Độ dài của min_num (2 byte)
    mov     ebx, 1         ; File descriptor (stdout)
    mov     eax, 4         ; System call number (sys_write)
    int     0x80           ; Gọi kernel để in ra giá trị nhỏ nhất

    mov     eax, 1         ; System call number (sys_exit)
    int     80h           ; Gọi kernel để kết thúc chương trình

section .data
    msg db "The smallest digit is: ", 0xA, 0xD ; Chuỗi thông báo in ra màn hình
    len equ $- msg        ; Độ dài của chuỗi msg
    num1 dd '7'           ; Số thứ nhất
    num2 dd '2'           ; Số thứ hai
    num3 dd '1'           ; Số thứ ba

section .bss
    min_num resb 2        ; Không gian bộ nhớ để lưu trữ giá trị nhỏ nhất
```

- Kết quả:



```

22  mov     eax, 4          ; System call number (sys_write)
23  int     0x80           ; Gọi kernel để in ra chuỗi
24
25  mov     ecx, min_num    ; Di chuyển địa chỉ của min_num vào thanh
                           ghi ecx
26  mov     edx, 2          ; Độ dài của min_num (2 byte)
27  mov     ebx, 1          ; File descriptor (stdout)
28  mov     eax, 4          ; System call number (sys_write)
29  int     0x80           ; Gọi kernel để in ra giá trị nhỏ nhất
30
31  mov     eax, 1          ; System call number (sys_exit)
32  int     80h            ; Gọi kernel để kết thúc chương trình
33
34  section .data
35      msg db "The smallest digit is: ", 0xA, 0xD ; Chuỗi thông báo
                           in ra màn hình
36      len equ $- msg      ; Độ dài của chuỗi msg
37      num1 dd '7'         ; Số thứ nhất
38      num2 dd '2'         ; Số thứ hai
39      num3 dd '1'         ; Số thứ ba
40
41  section .bss
42      min_num resb 2      ; Không gian bộ nhớ để lưu trữ giá trị nhỏ
                           nhất
43

```

Terminal output: The smallest digit is: 1

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

Ý tưởng của bài này là mỗi chữ số của số nguyên 123 được chia ra, sau đó chuyển đổi thành ký tự ASCII tương ứng, lưu trữ vào chuỗi kết quả và in ra màn hình:

- Code:

```

1  %assign SYS_EXIT 1
2  %assign SYS_WRITE 4
3  %assign STDOUT 1
4
5  section .data
6      x db 123          ; x = 123
7      msgX db "x = "    ; message
8
9  section .text
10     global _start      ; tell linker entry point
11
12     _start:             ; entry point
13         mov ecx, msgX    ; move address of msgX to ecx
14         mov edx, 4       ; move length of msgX to edx
15         call _printString ; call printString()
16         mov eax, 0       ; move 0 to eax
17         mov al, byte [x] ; move x to al
18         call _printDec   ; call printDec()
19         mov eax, SYS_EXIT ; move 1 to eax (exit syscall number)
20         xor edi, edi     ; clear edi
21         syscall          ; syscall for exit
22
23     _printString:        ; printString function
24         push eax          ; save eax
25         push ebx          ; save ebx
26         mov eax, SYS_WRITE ; move SYS_WRITE to eax
27         mov ebx, STDOUT   ; move STDOUT to ebx
28         int 0x80          ; call kernel

```

```

29     pop ebx           ; restore ebx
30     pop eax           ; restore eax
31     ret
32
33     _printDec:
34     section .bss
35     .decstr resb 10    ; reserve space for decimal string
36     .ct1 resd 1        ; to keep track of the size of the string
37     section .text
38     pushad            ; save all registers
39     mov dword [.ct1], 0 ; assume initially 0
40     mov edi, .decstr   ; edi points to decstring
41     add edi, 9         ; moved to the last element of string
42     xor edx, edx      ; clear edx for 64-bit division
43     .whileNotZero:
44     mov ebx, 10        ; get ready to divide by 10
45     div ebx            ; divided by 10
46     add dl, '0'        ; converts to ascii char
47     mov [edi], dl      ; put it in string
48     dec edi            ; move to next char in string
49     inc dword [.ct1]   ; increment char counter
50     xor edx, edx      ; clear edx
51     test eax, eax      ; check if quotient is zero
52     jnz .whileNotZero ; if not, continue loop
53     inc edi            ; conversion, finish, bring edi
54     mov ecx, edi       ; back to beginning of string, make ecx
55     mov edx, [.ct1]    ; point to it, and edx gets # chars
56     mov eax, SYS_WRITE ; and print!

```

```

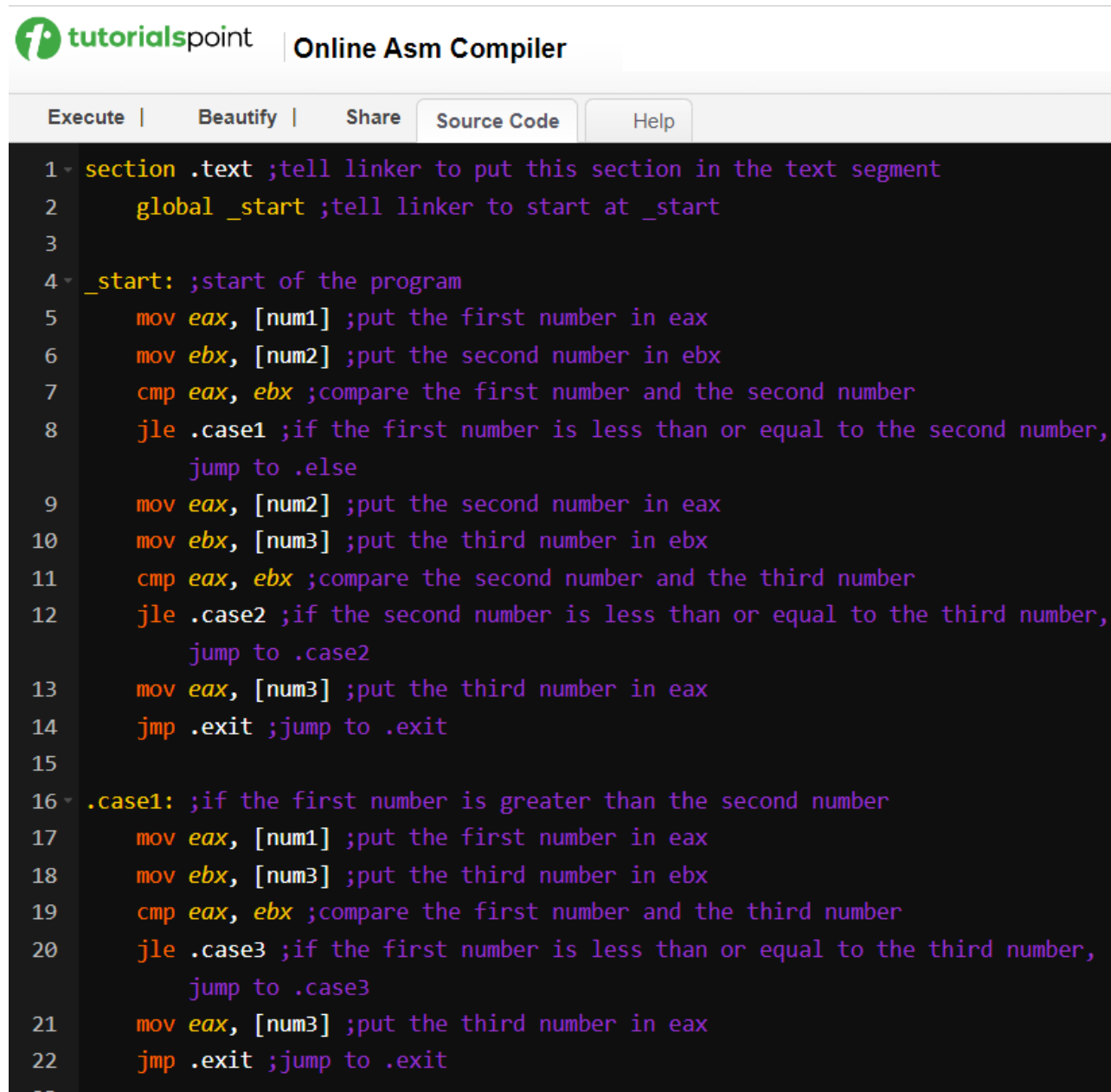
mov ebx, STDOUT    ; print to stdout
int 0x80           ; call the kernel
popad              ; restore all registers
ret

```

- Kết quả:

<pre> 1 %assign SYS_EXIT 1 2 %assign SYS_WRITE 4 3 %assign STDOUT 1 4 5 section .data 6 x db 123 ; x = 123 7 msgX db "x = " ; message 8 9 section .text 10 global _start ; tell linker entry point 11 </pre>	<pre>x = 123</pre>
--	--------------------

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



The screenshot shows the 'Online Asm Compiler' interface with the 'Source Code' tab selected. The code is written in assembly and aims to find the minimum of three numbers (num1, num2, num3) and store it in the EAX register. The code uses conditional jumps (jle) to compare the numbers and jumps to different cases based on the results. The cases are labeled .case1, .case2, and .case3. The code ends with a jump to .exit.

```
1 section .text ;tell linker to put this section in the text segment
2     global _start ;tell linker to start at _start
3
4 _start: ;start of the program
5     mov eax, [num1] ;put the first number in eax
6     mov ebx, [num2] ;put the second number in ebx
7     cmp eax, ebx ;compare the first number and the second number
8     jle .case1 ;if the first number is less than or equal to the second number,
        jump to .else
9     mov eax, [num2] ;put the second number in eax
10    mov ebx, [num3] ;put the third number in ebx
11    cmp eax, ebx ;compare the second number and the third number
12    jle .case2 ;if the second number is less than or equal to the third number,
        jump to .case2
13    mov eax, [num3] ;put the third number in eax
14    jmp .exit ;jump to .exit
15
16 .case1: ;if the first number is greater than the second number
17     mov eax, [num1] ;put the first number in eax
18     mov ebx, [num3] ;put the third number in ebx
19     cmp eax, ebx ;compare the first number and the third number
20     jle .case3 ;if the first number is less than or equal to the third number,
        jump to .case3
21     mov eax, [num3] ;put the third number in eax
22     jmp .exit ;jump to .exit
23
```

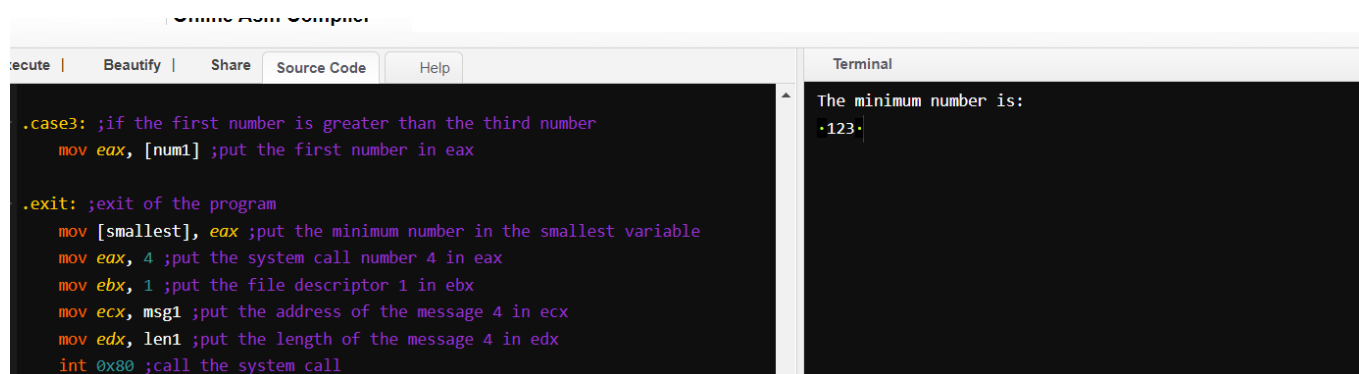


Execute | Beautify | Share | Source Code | Help

```
23
24 .case2: ;if the second number is greater than the third number
25     mov eax, [num1] ;put the first number in eax
26     mov ebx, [num2] ;put the second number in ebx
27     cmp eax, ebx ;compare the first number and the second number
28     jle .case3 ;if the first number is less than or equal to the second number,
        jump to .case3
29     mov eax, [num2] ;put the second number in eax
30     jmp .exit ;jump to .exit
31
32 .case3: ;if the first number is greater than the third number
33     mov eax, [num1] ;put the first number in eax
34
35 .exit: ;exit of the program
36     mov [smallest], eax ;put the minimum number in the smallest variable
37     mov eax, 4 ;put the system call number 4 in eax
38     mov ebx, 1 ;put the file descriptor 1 in ebx
39     mov ecx, msg1 ;put the address of the message 4 in ecx
40     mov edx, len1 ;put the length of the message 4 in edx
41     int 0x80 ;call the system call
42     mov eax, 4 ;put the system call number 4 in eax
43     mov ebx, 1 ;put the file descriptor 1 in ebx
44     mov ecx, smallest ;put the address of the smallest variable in ecx
45     mov edx, 4 ;put the length of the smallest variable in edx
46     int 0x80 ;call the system call
47     mov eax, 1 ;put the system call number 1 in eax
48     mov ebx, 0 ;put the exit code 0 in ebx
49     int 0x80 ;call the system call
50
```

```
section .data ;tell linker to put this section in the data segment
msg1 db "The minimum number 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
```



```

.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
    
```

Terminal

```

The minimum number is:
123
    
```

4. Bài thực hành 5: Bằng cách không tạo thêm vùng nhớ mở rộng vào tập tin PE, tận dụng vùng nhớ trống để chèn chương trình cần chèn trên tập tin Notepad và calc.

Link video báo cáo: <https://youtu.be/yKPLDSJZWNg>

a. Notepad:

- Xem vùng nhớ trống ở cuối file, chọn 0x00010D50 lưu caption và 0x00010D60 lưu text:

```










00010D40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010D50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010D60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010D70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010D80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010D90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010DA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010DB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010DC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010DD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010DE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00010DF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
    
```

- Tìm các giá trị còn thiếu:

```

push 0 ;                6a 00
push Caption ;          68 X
push Text ;             68 Y
push 0 ;                6a 00
call [MessageBoxW] ;    ff15 Z
jmp Origianl_Entry_Point ; e9 relative_VA
    
```

- Tìm Z:

 01001208	PeekMessageW	USER32
 0100123C	SendDlgItemMessageW	USER32
 01001240	SendMessageW	USER32
 01001268	MessageBoxW	USER32
 01001294	DispatchMessageW	USER32
 01001298	TranslateMessage	USER32
 010012A0	IsDialogMessageW	USER32
 010012A4	PostMessageW	USER32
 010012A8	GetMessageW	USER32

- Tìm X:


```
Offset = RA - Section RA = VA - Section VA (1)
=> 0x00010D50 - 0x00008400 = X - 0x0000B000
=> X = 0x00010D50 - 0x00008400 + 0x0000B000 = 0x00013950
=> X = 0x01013950 (50 39 01 01)
```

- Tìm Y:

```
Y = 0x00010D60 - 0x00008400 + 0x0000B000 + 0x01000000 = 0x01013960 (60 39 01 01)
```

- Tìm relative_VA:

```
new_entry_point = 0x00010DA0 - 0x00008400 + 0x0000B000 = 0x000139A0
jmp_instruction_VA = 0x010139A0 + 0x14 = 0x010139B4
relative_VA = old_entry_point - jmp_instruction_VA - 5 - imagebase
              = 0x100739D - 0x010139B4 - 5 - 0x01000000
              = FFFF39E4 (E4 39 FF FF)
```

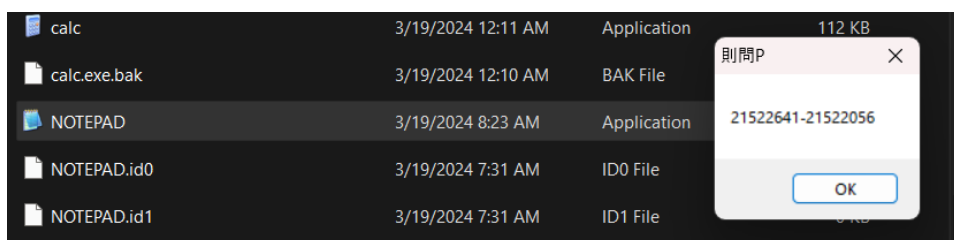
- Tiến hành chèn vào notepad:

```
00010D30  6F 00 63 00 75 00 6D 00 65 00 6E 00 74 00 00 00  o.c.u.m.e.n.t...
00010D40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00010D50  47 52 4F 55 50 09 00 00 00 00 00 00 00 00 00 00  GROUP.....
00010D60  32 00 31 00 35 00 32 00 32 00 36 00 34 00 31 00  2.1.5.2.2.6.4.1.
00010D70  2D 00 32 00 31 00 35 00 32 00 32 00 30 00 35 00  -.2.1.5.2.2.0.5.
00010D80  36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  6.....
00010D90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00010DA0  6A 00 68 50 39 01 01 68 60 39 01 01 6A 00 FF 15  j.hp9..h`9..j.ỹ.
00010DB0  68 12 00 01 E9 E4 39 FF FF 00 00 00 00 00 00 00 00  h...éä9ÿÿ.....
00010DC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00010DD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00010DE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00010DF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

- Chỉnh AddressOfEntryPoint:

SizeOfInitializedData	00000100	Dword	0000A600	
SizeOfUninitializedData	00000104	Dword	00000000	
AddressOfEntryPoint	00000108	Dword	000139A0	.rsrc
BaseOfCode	0000010C	Dword	00001000	

- Kiểm tra kết quả:



b. Calc:

- Dùng HxD để xem khoảng trống cuối file. Có thể chọn vị trí trống tùy ý nhưng em sẽ chọn 0x0001BF70 để lưu caption, và 0x0001BFB0 để lưu text:

```

0001BE80 72 00 79 00 20 00 66 00 6F 00 72 00 20 00 64 00 r.y. .f.o.r. .d.
0001BE90 61 00 74 00 61 00 2E 00 0D 00 43 00 6C 00 6F 00 a.t.a.....C.l.o.
0001BEA0 73 00 65 00 20 00 6F 00 6E 00 65 00 20 00 6F 00 s.e. .o.n.e. .o.
0001BEB0 72 00 20 00 6D 00 6F 00 72 00 65 00 20 00 70 00 r. .m.o.r.e. .p.
0001BEC0 72 00 6F 00 67 00 72 00 61 00 6D 00 73 00 2C 00 r.o.g.r.a.m.s.,.
0001BED0 20 00 61 00 6E 00 64 00 20 00 74 00 68 00 65 00 .a.n.d. .t.h.e.
0001BEE0 6E 00 20 00 74 00 72 00 79 00 20 00 61 00 67 00 n. .t.r.y. .a.g.
0001BEF0 61 00 69 00 6E 00 2E 00 08 00 63 00 61 00 6C 00 a.i.n.....c.a.l.
0001BF00 63 00 2E 00 63 00 68 00 6D 00 0A 00 43 00 61 00 c...c.h.m...C.a.
0001BF10 6C 00 63 00 75 00 6C 00 61 00 74 00 6F 00 72 00 l.c.u.l.a.t.o.r.
0001BF20 11 00 4E 00 6F 00 74 00 20 00 45 00 6E 00 6F 00 ..N.o.t. .E.n.o.
0001BF30 75 00 67 00 68 00 20 00 4D 00 65 00 6D 00 6F 00 u.g.h. .M.e.m.o.
0001BF40 72 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00 r.y.....
0001BF50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BF60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BF70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BF80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BF90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BFA0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BFB0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BFC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BFD0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BFE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0001BFF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

- Đoạn mã hợp ngữ cần chèn:

```

push 0 ;                6a 00
push Caption ;          68 X
push Text ;             68 Y
push 0 ;                6a 00
call [MessageBoxW] ;    ff15 Z
jmp Origanl_Entry_Point ; e9 relative_VA

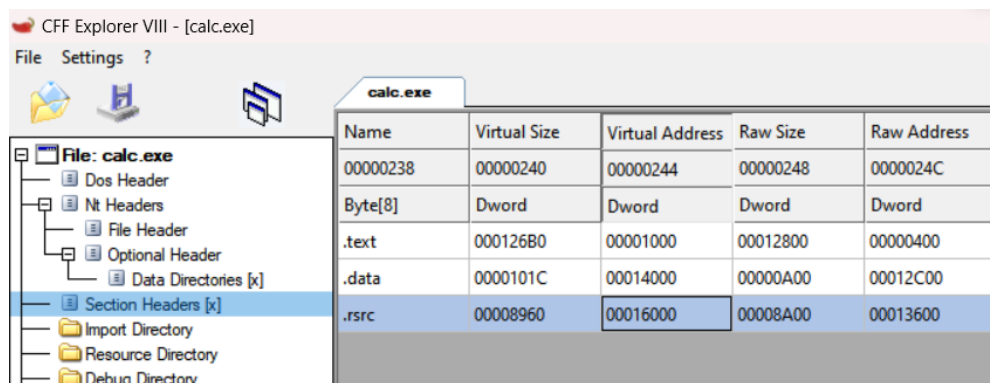
```

- Giá trị Z chính là địa chỉ của hàm MessageBoxW được import từ thư viện USER32.dll. Trong IDA Pro, mở calc.exe, chọn **View -> Open Subviews -> Imports** và ta thấy địa chỉ của hàm **MessageBoxW** chính là: 010011A8.

Vậy Z = **010011A8**.

Address	Ordinal	Name	Library
010010C4		PostQuitMessage	USER32
01001104		MessageBeep	USER32
0100113C		SendMessageW	USER32
01001188		DispatchMessageW	USER32
0100118C		TranslateMessage	USER32
01001198		IsDialogMessageW	USER32
0100119C		GetMessageW	USER32
010011A8		MessageBoxW	USER32

- Tìm X theo công thức: $\text{Offset} = \text{RA} - \text{Section RA} = \text{VA} - \text{Section VA}$ (1)
- => Cần tìm section VA, section RA để tính được X:



Name	Virtual Size	Virtual Address	Raw Size	Raw Address
00000238	00000240	00000244	00000248	0000024C
Byte[8]	Dword	Dword	Dword	Dword
.text	000126B0	00001000	00012800	00000400
.data	0000101C	00014000	00000A00	00012C00
.rsrc	00008960	00016000	00008A00	00013600

Tìm được section RA = 0x00013600, section VA = 0x00016000

Giá trị X có thể được tìm dựa vào công thức (1):

$$0x0001BF70 - 0x00013600 = X - 0x00016000$$

$$\Rightarrow X = 0x0001E970$$

Cộng thêm ImageBase, suy ra X = **0x00101E970**. Tương tự, Y = **0x0101E9B0**.

Như vậy, đoạn code này thực hiện chức năng như mong đợi và có địa chỉ mới là:

$$\text{new_entry_point} = 0x0001BF50 - 0x00013600 + 0x00016000 = 0x0001E950.$$

$$\Rightarrow \text{AddressOfEntryPoint} = 0x0101E950$$

- Thiết lập lệnh quay về AddressOfEntryPoint ban đầu:
- Nếu đặt lệnh jmp sau 5 câu lệnh ở bước 2 thì

$$\text{jmp_instruction_VA} = 0x0101E950 + 0x14 = 0x101E964.$$

$$\text{old_entry_point} = \text{AddressOfEntryPoint} + \text{ImageBase (xem bằng CFF)} = 0x12475 + 0x1000000 = 0x1012475$$

$$\Rightarrow \text{relative_VA} = \text{old_entry_point} - \text{jmp_instruction_VA} - 5 = \mathbf{0xFFFF3B0C}$$

- Đoạn mã cần chèn hoàn chỉnh là (thứ tự little endian):

```
push 0 ;                6a 00
push Caption ;          68 70E901010
push Text ;             68 B0E90101
push 0 ;                6a 00
call [MessageBoxW] ;    ff15 A8110001
jmp Origanl_Entry_Point ; e9 0C3BFFFF
```

- Chèn vào calc.exe:

```

0001BF50  6A 00 68 70 E9 01 01 68 B0 E9 01 01 6A 00 FF 15  j.hpé..h°é..j.ÿ.
0001BF60  A8 11 00 01 E9 0C 3B FF FF 00 00 00 00 00 00 00  ..é.;ÿÿ.....
0001BF70  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0001BF80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0001BF90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0001BFA0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0001BFB0  32 00 31 00 35 00 32 00 32 00 36 00 34 00 31 00  2.1.5.2.2.6.4.1.
0001BFC0  2D 00 32 00 31 00 35 00 32 00 32 00 30 00 35 00  -.2.1.5.2.2.0.5.
0001BFD0  36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  6.....
0001BFE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....

```

- Hiệu chỉnh các tham số trong PE header:
 - Trong Optional Headers, chỉnh sửa AddressOfEntryPoint thành 0x0001E950:

AddressOfEntryPoint	00000118	Dword	0001E950	.rsrc
---------------------	----------	-------	----------	-------

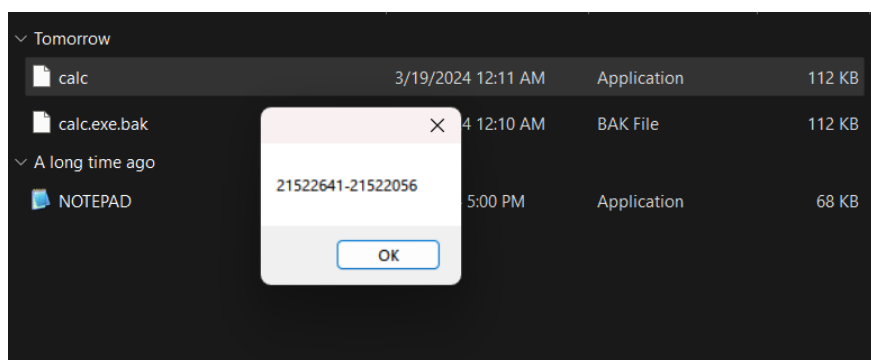
- Payload:

6A 00 68 70 E9 01 01 68 B0 E9 01 01 6A 00 FF 15

A8 11 00 01 E9 0C 3B FF FF

32 00 31 00 35 00 32 00 32 00 36 00 34 00 31 00 2D 00 32 00 31 00 35 00 32 00 32
00 30 00 35 00 36 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (MSSV)

- Kết quả:



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 **.DOCX và .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 đã đăng ký với GVHD-TH).
Ví dụ: [NT101.K11.ATCL]-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 bài Lab.**
- 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 được bài thực hành. Khuyến khích:

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

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

HẾT