BÁO CÁO BÀI TẬP

**Môn học: Cơ chế hoạt động của mã độc**

**Kỳ báo cáo: Buổi 01 (Session 01)**

**Tên chủ đề: ÔN TẬP NGÔN NGỮ ASSEMBLY & CHÈN MÃ VÀO TẬP TIN PE**

*GV: Nghi Hoàng Khoa*

*Ngày báo cáo: 13/03/2023*

**Nhóm: 09**

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT230.N21.ANTN

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Bùi Kim Ngân | 20520648 | 20520648@gm.uit.edu.vn |
| 2 | Nguyễn Bình Thục Trâm | 20520815 | 20520815@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** | **Người đóng góp** |
| 1 | Bài thực hành 1 | 100% | Trâm |
| 2 | Bài thực hành 2 | 100% | Ngân |
| 3 | Bài thực hành 3 | 100% | Trâm |
| 4 | Bài thực hành 5 | 100% | Trâm |

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

BÁO CÁO CHI TIẾT

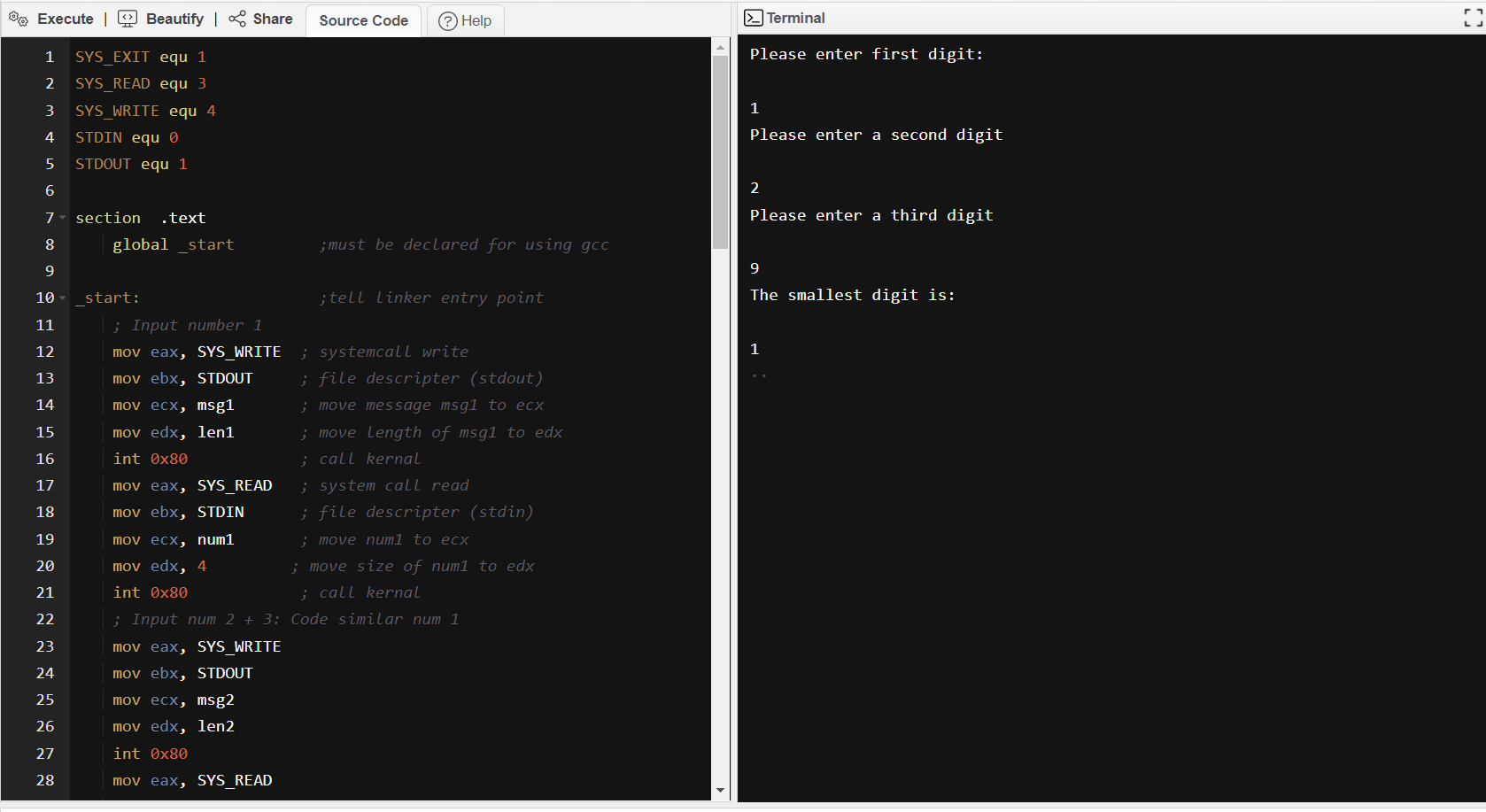
## Bài thực hành 01

## Tài nguyên: <https://www.tutorialspoint.com/compile_assembly_online.php>

## Mô tả/mục tiêu: Viết một đoạn chương trình tìm số nhỏ nhất trong 3 số (1 chữ số) a,b,c cho trước.

## Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích):

Kết quả chạy chương trình:



Đoạn code thực hiện:

SYS\_EXIT equ 1

SYS\_READ equ 3

SYS\_WRITE equ 4

STDIN equ 0

STDOUT equ 1

section .text

    global \_start         ;must be declared for using gcc

\_start:                  ;tell linker entry point

    ; Input number 1

    mov eax, SYS\_WRITE  ; systemcall write

    mov ebx, STDOUT     ; file descripter (stdout)

    mov ecx, msg1       ; move message msg1 to ecx

    mov edx, len1       ; move length of msg1 to edx

    int 0x80            ; call kernal

    mov eax, SYS\_READ   ; system call read

    mov ebx, STDIN      ; file descripter (stdin)

    mov ecx, num1       ; move num1 to ecx

    mov edx, 4         ; move size of num1 to edx

    int 0x80            ; call kernal

    ; Input num 2 + 3: Code similar num 1

    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, 4

    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, 4

    int 0x80

    ; num1 -> eax, num2 -> ebx; num3 -> ecx

    ; Find the smallest

    mov   eax, [num1]

    sub   eax, '0'

    mov   ebx, [num2]

    sub   ebx, '0'

    mov   ecx, [num3]

    sub   ecx, '0'

    ; Compare num1 & num2

    cmp   eax, ebx

    jl    check\_third\_num ; if num1 < num2 => Compare num1 & num3

    mov   eax, ebx ; else move num2 -> eax => Compare num2 & num3

    check\_third\_num: ; Compare numX & num 3

    cmp   eax, ecx

    jl    \_exit     ; if numX < num3 => numX min => exit

    mov   eax, ecx  ; else num3 min => move num3 -> eax

    \_exit: ; exit function

    add   eax, '0'  ; add '0' to to convert the result from decimal to ASCII

    mov   [smallest], eax ; move result to smallest

    mov   ecx, msg      ; move message msg to ecx

    mov   edx, len      ; move length of message msg to edx

    mov   ebx,1         ;file descriptor (stdout)

    mov   eax,4         ;system call number (sys\_write)

    int   0x80          ;call kernel

    mov   ecx, smallest ; move result to ecx for printing

    mov   edx, 4       ; move size of result to edx

    mov   ebx,1         ;file descriptor (stdout)

    mov   eax,4         ;system call number (sys\_write)

    int   0x80          ;call kernel

    mov   eax, 1        ; systemcall EXIT

    int   80h           ; call kernal

section .data

    msg1 db "Please enter first digit: ", 0xA,0xD   ; define message msg1

    len1 equ $- msg1 ; define len1 = length of message msg1

    msg2 db "Please enter a second digit", 0xA,0xD   ; define message msg2

    len2 equ $- msg2 ; define len2 = length of message msg2

    msg3 db "Please enter a third digit", 0xA,0xD   ; define message msg3

    len3 equ $- msg3 ; define len3 = length of message msg3

    msg db "The smallest digit is: ", 0xA,0xD ; define message msg

    len equ $- msg  ; define len = length of message msg

segment .bss

    num1 resb 4     ; define num1  with size = 4

    num2 resb 4     ; define num2 type d with size = 4

    num3 resb 4     ; define num3 type d with size = 4

    smallest resb 4 ; define smallest type d with size = 4

Giải thích: Ý tưởng cụ thể trên đoạn code này là người dùng sẽ nhập vào 3 số bất kì có 1 chữ số, sau đó em sẽ chuyển chúng thành mã ASCII và so sánh với nhau. Quá trình so sánh diễn ra như sau: So sánh 2 số đầu tiên trước, nếu num1 < num2 thì so sánh tiếp num1 và num3 để tìm ra số nhỏ nhất. Ngược lại, nếu num1 >= num2 thì sẽ so sánh num2 với num3 để tìm ra số nhỏ nhất.

Cụ thể giải thích từng câu lệnh em đã chú thích trong đoạn code.

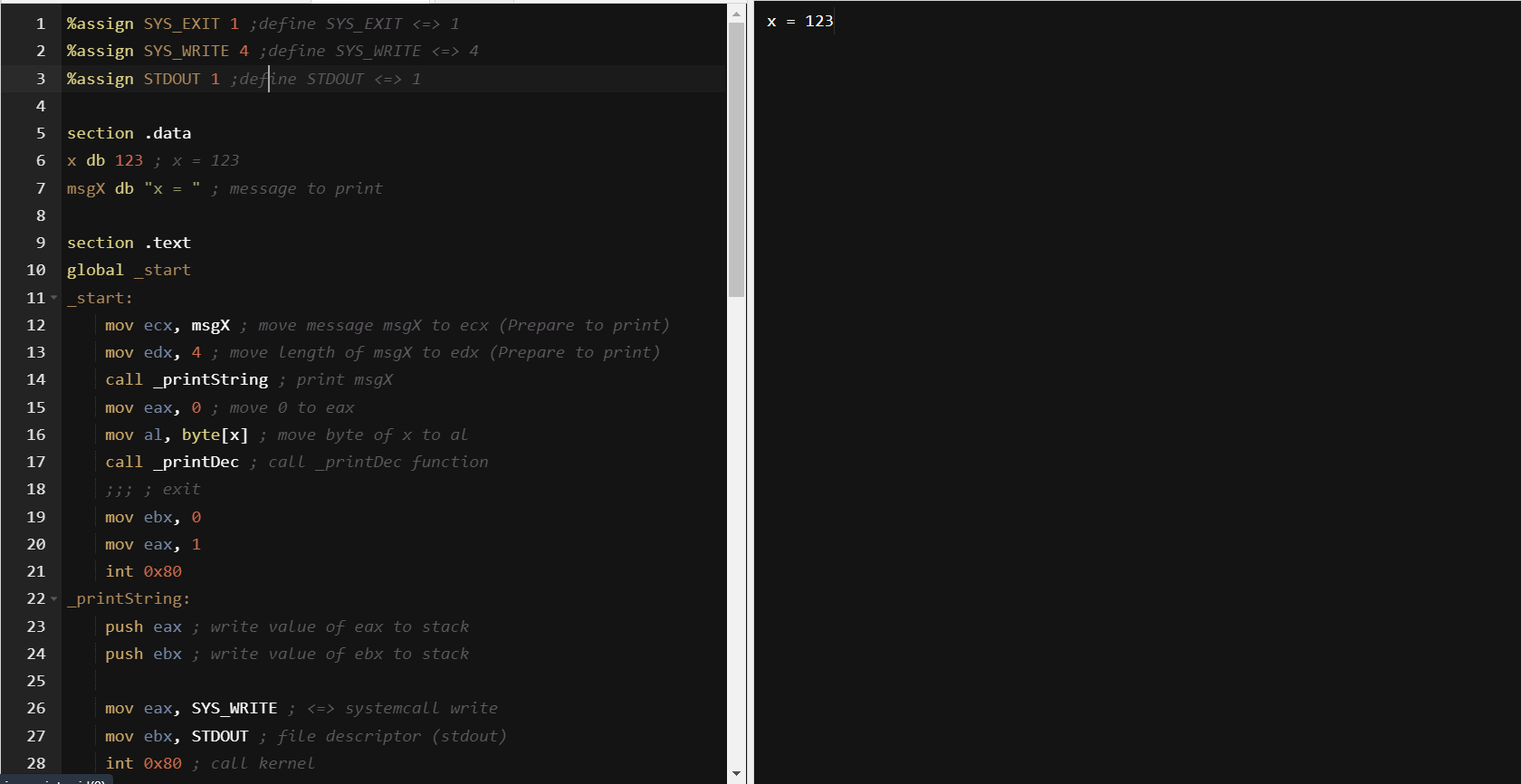
## Bài thực hành 02

## Tài nguyên: <https://www.tutorialspoint.com/compile_assembly_online.php>

## Mô tả/mục tiêu: Viết chương trình chuyển đổi một số (number) 123 thành chuỗi ‘123’ Sau đó thực hiện in ra màn hình số 123.

## Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích):

Kết quả chạy chương trình:



Code:

%assign SYS\_EXIT 1 ;define SYS\_EXIT <=> 1

%assign SYS\_WRITE 4 ;define SYS\_WRITE <=> 4

%assign STDOUT 1 ;define STDOUT <=> 1

section .data

x db 123 ; Define x = 123

msgX db "x = " ; Define message msgX

section .text

global \_start

\_start:

    mov ecx, msgX ; move message msgX to ecx (Prepare to print)

    mov edx, 4 ; move length of msgX to edx (Prepare to print)

    call \_printString ; print msgX

    mov eax, 0 ; move 0 to eax

    mov al, byte[x] ; move byte of x to al

    call \_printDec ; call \_printDec function

    ;;; ; exit

    mov ebx, 0

    mov eax, 1

    int 0x80

\_printString:

    push eax ; write value of eax to stack

    push ebx ; write value of ebx to stack

    mov eax, SYS\_WRITE ; <=> systemcall write

    mov ebx, STDOUT ; file descriptor (stdout)

    int 0x80 ; call kernel

    pop ebx ; pop value out from stack to ebx

    pop eax ; pop value out from stack to eax

    ret ; return

println:

    section .data

    .nl db 10 ; new line character

    section .text

    push ecx ; save value of ecx to stack

    push edx ; save value of edx to stack

    mov ecx, .nl ; move address of .nl to ecx

    mov edx, 1 ; move 1 to edx

    call \_printString ; call \_prinString function

    pop edx ; pop value out from stack to edx

    pop ecx ; pop value out from stack to ecx

    ret ; return

\_printDec:

    ;;; saves all the registers so that they are not changed by the function

    section .bss

    .decstr resb 10 ; define .decstr variable, size = 10 bytes

    .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

    int 0x80

    popad ; restore all registers

    ret

Giải thích đoạn code: Đầu tiên, ta có được số 123 cho sẵn, để có thể chuyển nó thành string, ta cần phải tách các chữ số ra bằng cách tuần tự chia số cho 10, sau đó chuyển các chữ số này sang mã ASCII và ghép lại thành chuỗi “123”. Chi tiết các bước làm đã được chú thích trong đoạn code.

## Bài thực hành 03

## Tài nguyên: <https://www.tutorialspoint.com/compile_assembly_online.php>

## Mô tả/mục tiêu: : Cải tiến chương trình yêu cầu 1 sao cho tìm số nhỏ nhất trong 3 số bất kỳ (nhiều hơn 1 chữ số)

## Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích):

Kết quả chạy chương trình:

Text

Description automatically generated

Code:

SYS\_EXIT equ 1

SYS\_READ equ 3

SYS\_WRITE equ 4

STDIN equ 0

STDOUT equ 1

section .text

    global \_start         ;must be declared for using gcc

\_start:                  ;tell linker entry point

    ; Input number 1

    mov eax, SYS\_WRITE  ; systemcall write

    mov ebx, STDOUT     ; file descripter (stdout)

    mov ecx, msg1       ; move message msg1 to ecx

    mov edx, len1       ; move length of msg1 to edx

    int 0x80            ; call kernal

    mov eax, SYS\_READ   ; system call read

    mov ebx, STDIN      ; file descripter (stdin)

    mov ecx, num1       ; move num1 to ecx

    mov edx, 12         ; move size of num1 to edx

    int 0x80            ; call kernal

    ; Input num 2 + 3: Code similar num 1

    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, 12

    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, 12

    int 0x80

    ; num1 -> eax, num2 -> ebx; num3 -> ecx

    ; Find the smallest

    mov   eax, [num1]

    sub   eax, '0'

    mov   ebx, [num2]

    sub   ebx, '0'

    mov   ecx, [num3]

    sub   ecx, '0'

    ; Compare num1 & num2

    cmp   eax, ebx

    jl    check\_third\_num ; if num1 < num2 => Compare num1 & num3

    mov   eax, ebx ; else move num2 -> eax => Compare num2 & num3

    check\_third\_num: ; Compare numX & num 3

    cmp   eax, ecx

    jl    \_exit     ; if numX < num3 => numX min => exit

    mov   eax, ecx  ; else num3 min => move num3 -> eax

    \_exit: ; exit function

    add   eax, '0'  ; add '0' to to convert the result from decimal to ASCII

    mov   [smallest], eax ; move result to smallest

    mov   ecx, msg      ; move message msg to ecx

    mov   edx, len      ; move length of message msg to edx

    mov   ebx,1         ;file descriptor (stdout)

    mov   eax,4         ;system call number (sys\_write)

    int   0x80          ;call kernel

    mov   ecx, smallest ; move result to ecx for printing

    mov   edx, 12       ; move size of result to edx

    mov   ebx,1         ;file descriptor (stdout)

    mov   eax,4         ;system call number (sys\_write)

    int   0x80          ;call kernel

    mov   eax, 1        ; systemcall EXIT

    int   80h           ; call kernal

section .data

    msg1 db "Please enter first digit: ", 0xA,0xD   ; define message msg1

    len1 equ $- msg1 ; define len1 = length of message msg1

    msg2 db "Please enter a second digit", 0xA,0xD   ; define message msg2

    len2 equ $- msg2 ; define len2 = length of message msg2

    msg3 db "Please enter a third digit", 0xA,0xD   ; define message msg3

    len3 equ $- msg3 ; define len3 = length of message msg3

    msg db "The smallest digit is: ", 0xA,0xD ; define message msg

    len equ $- msg  ; define len = length of message msg

segment .bss

    num1 resd 3     ; define num1 type d with size = 12

    num2 resd 3     ; define num2 type d with size = 12

    num3 resd 3     ; define num3 type d with size = 12

    smallest resd 3 ; define smallest type d with size = 12

Giải thích: Ở đây em cũng làm tương tự với bài 1, nhưng thay đổi kiểu dữ liệu của các số và kết quả. Em tăng size của dữ liệu lên để có thể nhập vào và thực hiện so sánh được với các số có nhiều chữ số hơn. Ở đây tối đa có thể có 3 chữ số.

## Bài thực hành 04

## Tài nguyên: CFF Explorer, HxD, IDA Pro

## Mô tả/mục tiêu: 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.

## Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích)

Do máy tính em yếu nên quay video cứ được khoảng 5 phút là nó giật lag và không làm bài được (Em chuyển từ máy ảo sang máy thật cũng không khá hơn) nên em xin phép thầy quay video đoạn chèn chương trình vào hai tập tin và chạy ra kết quả. Còn phần tính toán để tạo ra đoạn mã em sẽ trình bày cụ thể bên dưới ạ.

Trong cả hai bài, ta sẽ cần tìm một đoạn vùng nhớ trống có sẵn trong tập tin (Vùng nhớ có giá trị 00…) đủ để chèn chương trình vào. Sau đó, ta sẽ ghi chú lại địa chỉ để lưu hợp ngữ, Caption và Text của chương trình tương tự như ở bài 4.

Chương trình cần chèn vào sẽ có 5 câu lệnh:

Text

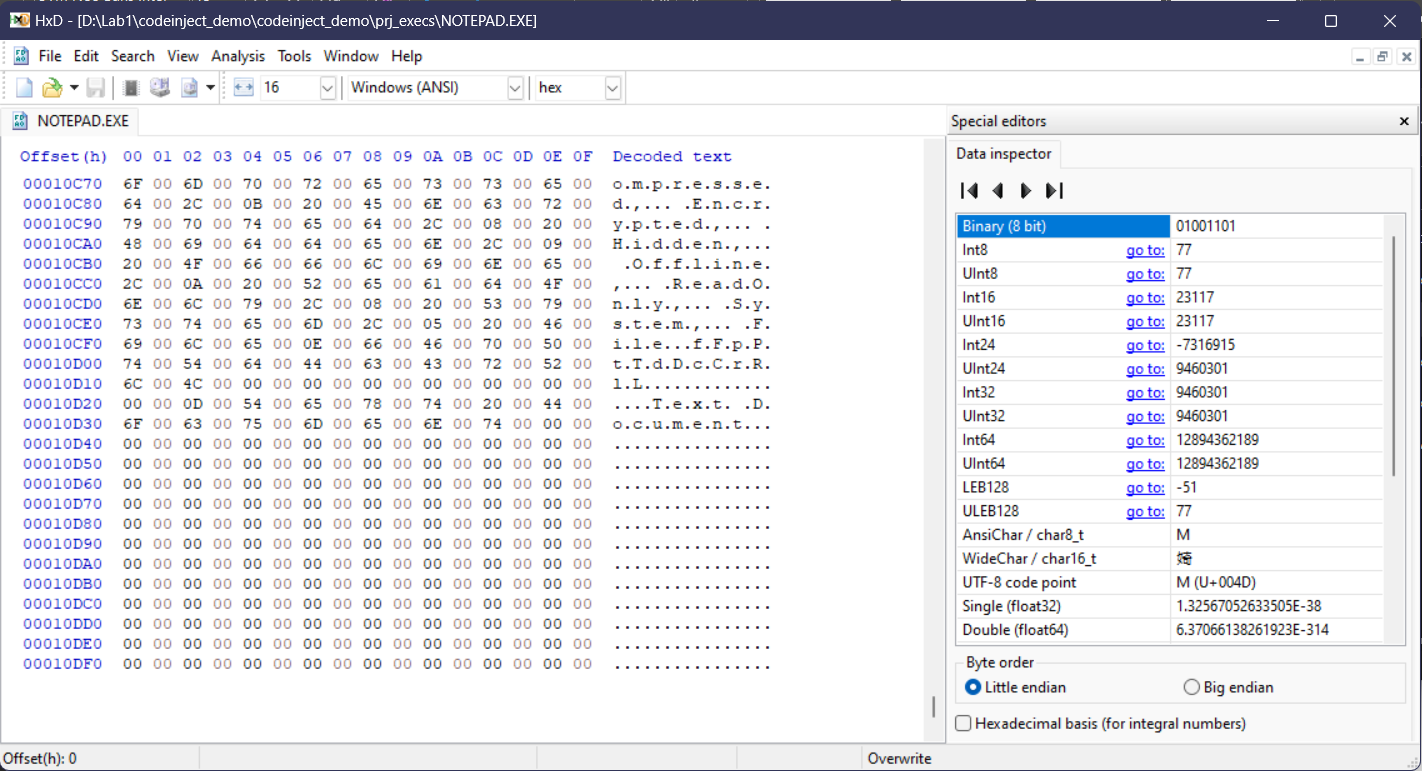
Description automatically generated

Đối với tập tin Notepad.exe, em đã dùng HxD và tìm thấy vùng nhớ trống phù hợp như sau:

Mã hợp ngữ: 0x10D50

Caption: 0x10D70

Text: 0x10DA0



Đối với tập tin cal.exe, em đã dùng HxD và tìm thấy vùng nhớ trống phù hợp như sau:

Graphical user interface, application

Description automatically generated

Tiếp theo sau đó, để áp dụng công thức: **RA – Section RA = VA – Section VA** ta cần phải tìm được các giá trị Section RA và Section VA. Hai giá trị này bên tập tin Notepad.exe đã được tìm trong câu trước, em sẽ sử dụng lại. Còn ở tập tin cal.exe, em sử dụng CFF Explorer để tìm hai giá trị này

Graphical user interface, application

Description automatically generated



Sau đó, em áp dụng công thức để tính được các giá trị X, Y tương tự với bài trước đó.

Notepad.exe:

0x10D70 – 0x00008400 = X – 0x000B000

=> X = 0x13970 + imageBase = 0x01013970

0x10DA0 – 0x00008400 = Y – 0x000B000

=> Y = 0x139A0 + imageBase = 0x010139A0

new entry point = 0x10D50 – 0x00008400 + 0x000B000 = 0x13950

=> + imageBase = 0x1013950

Cal.exe:

0x1BF70 – 00013600 = X – 00016000

=> X = 0x1E970 + 0x01000000 = 0x101E970

0x1BFA0 – 00013600 = Y – 00016000

=> Y = 0x1E9A0 + 0x01000000 = 0x101E9A0

new entry point = 0x1BF50 - 00013600 + 00016000 = 0x1E950

=> + imageBase = 0x101E950

Tiếp theo, em tiếp tục dùng CFF Explorer để xem được giá trị Address Of Entry Point và giá trị này sẽ tương đương với old\_entry\_point trong công thức ***old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA***:

Graphical user interface, table

Description automatically generated

*(Của file Notepad.exe giống như bài trước. Của file cal.exe = 0x12475)*

Nếu đặt lệnh jmp sau 5 câu lệnh thì jmp \_instruction\_VA = new\_entry\_point + imageBase + 14.

Notepad.exe: ***relative\_VA = old\_entry\_point - jmp\_instruction\_VA - 5 = FFFF3A34***

cal.exe: ***relative\_VA = old\_entry\_point - jmp\_instruction\_VA - 5 = FFFF3B0C***

Và cuối cùng, giá trị Z của cả 2 là địa chỉ của hàm MessageBoxW được import từ thư viện USER32.dll, ta có thể dễ dàng dùng IDA Pro để xem 2 giá trị này.

Notepad.exe: Z = 01001268

cal.exe: Z = 010011A8

Cuối cùng, ta sẽ đảo các bit theo đúng thứ tự little Endian để chèn chương trình vào trong hai tập tin đã cho theo đúng thứ tự các câu lệnh:

**Mã hợp ngữ:**

Notepad.exe: 6A 00 68 A0 39 01 01 68 70 39 01 01 6A 00 FF 15 68 12 00 01

E9 34 3A FF FF

Cal.exe: 6A 00 68 A0 E9 01 01 68 70 E9 01 01 6A 00 FF 15 A8 11 00 01

E9 0C 3B FF FF

**Text**: 4D 00 53 00 53 00 56 00

**Caption**: 32 00 30 00 35 00 32 00 30 00 38 00 31 00 35 00 2D 32 00 30 00 35 00 32 00 30 00 36 00 34 00 38 00

Video thực hiện bước chèn và chạy chương trình: <https://youtu.be/lijV-4p3jE4>

---

***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 Bài thực hành/câu hỏi phức tạp hơn, có đóng góp xây dựng.

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

**HẾT**

1. Ghi nội dung công việc, các Bài thực hành trong bài Thực hành [↑](#footnote-ref-1)