

Graphical user interface, text

Description automatically generated

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:”

Text

Description automatically generated

So sánh **:** .

* Nếu num1 < num2 thì xuống so sánh với num3
  + 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
  + Nếu num2 < num3 thì num2 là min
  + 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.

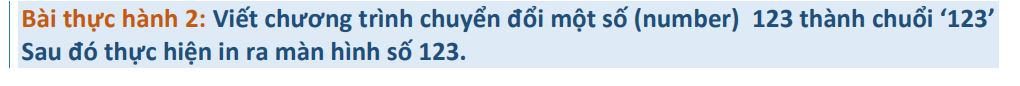
Text

Description automatically generated

* Xuất chuỗi đã khai báo ở đầu chương trình
* Gán smallest = ecx
* Xuất giá trị min ra
* Kết quả :

Text

Description automatically generated with medium confidence



Graphical user interface, text, application

Description automatically generated

Khai báo x để chuyển thành chuỗi và in ra **number\_str** là 1 biến dùng để lưu chuỗi đó

Text

Description automatically generated

Xuất “x=” ra màn hình

Text

Description automatically generated

* 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)

* Eax = 1 , edx=23
* Đổi eax từ số nguyên sang kí tự
* Gán kí tự đầu tiên của biến **number\_str** là : 1

A screenshot of a computer

Description automatically generated with medium confidence

* 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**

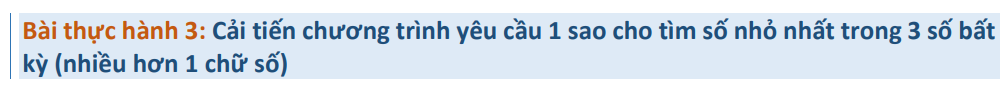
**Text

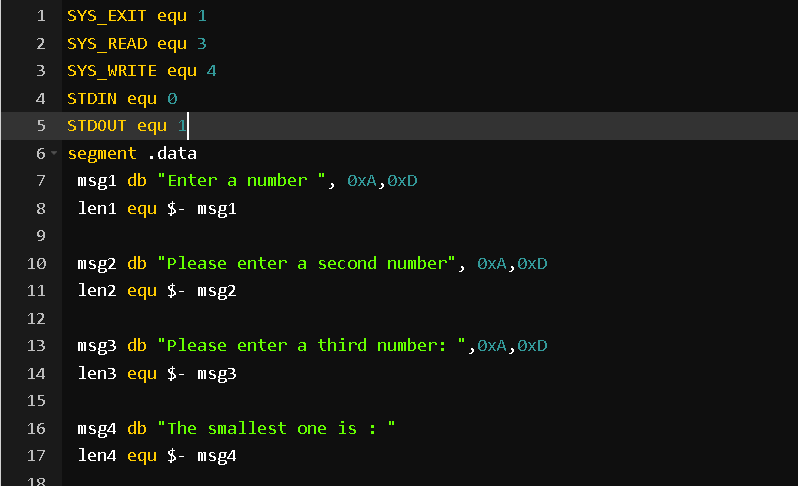
Description automatically generated**

Xuất biến **number\_str** ra và chúng ta có : 123

**Graphical user interface, text

Description automatically generated**

****

****

Khai báo các chuỗi để xuất ra màn hình

**A picture containing text

Description automatically generated**

Các biến dùng để nhập vào và so sánh

**Text

Description automatically generated**

Nhập 2 biến num1 và num2

**Text

Description automatically generated**

Nhập num3

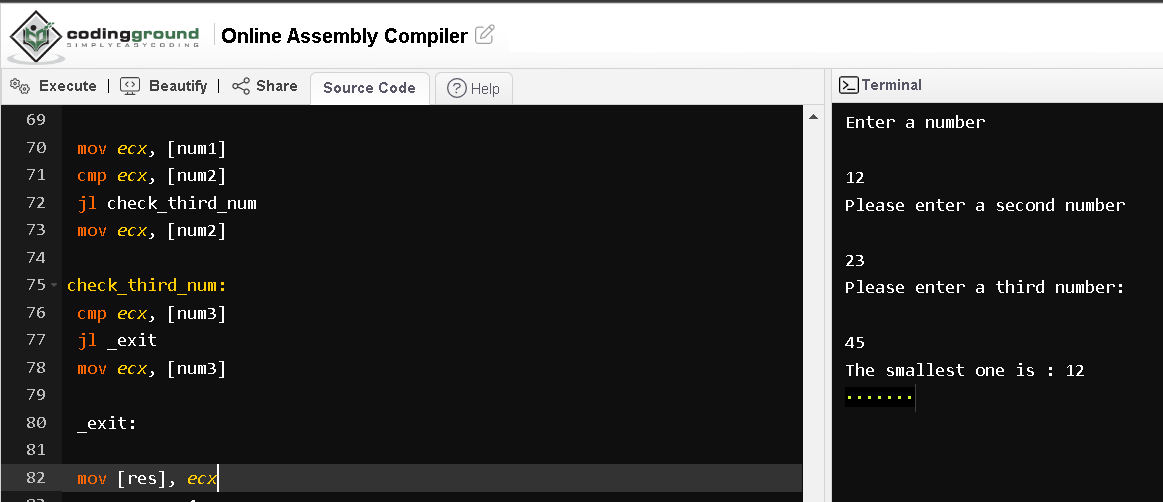


So sánh **:** .

* Nếu num1 < num2 thì xuống so sánh với num3
  + 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
  + Nếu num2 < num3 thì num2 là min
  + Nếu num2 > num3 thì num3 là min và ecx = num3

Xuất “**The smallest one is** : ” (msg4) và **res**

* Kết quả :



Shape, rectangle

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application

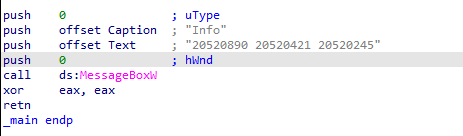
Description automatically generated



Để chèn mã vào tập tin PE thì ta phải có chỗ để chèn vào

**B1**:Dùng HxD để mở file ta cần chèn mã vào và insert thêm 1000 byte vào cuối file

**B2**:Biên dịch chương trình cần chèn dưới chế độ **Release**, **Not Using Precompiled Headers**. Sử dụng IDA Pro để mở file PE và xem mã hợp ngữ của chương trình vừa biên dịch , nó sẽ có dạng như thế này :



push 0 ; 6a 00

push Caption ; 68 X

push Text ; 68 **Y**

push 0 ; 6a 00

call [MessageBoxW] ; ff15 Z

* Dùng IDA mở lên để xem cấu trúc câu lệnh:

**B3** : Chọn vị trí bất kì trong phần mới insert đó để thêm vào :

Ta chọn:

* Địa chỉ **0x00011000** trong vùng nhớ đã được mở rộng (địa chỉ chèn mã vào) – **new\_entry\_point**
* **0x00011040** để lưu trữ **Caption**
* **0x00011060** để lưu trữ **Text**

Ta sử dụng công thức : **Offset = RA – Section RA = VA – Section VA (1)**

* Để có được section RA và Section VA thì ta phải dùng CFF explorer để mở file notepad.exe để xem 2 dữ liệu này
* **section RA**= 00008400
* **Section VA**= 0000B000
* Để tính địa chỉ trong bộ nhớ ảo khi ta load file PE lên (để thay thế địa chỉ này vào địa chỉ entry cũ của chương trình thì chương trình sẽ thực hiện những câu lệnh ta chèn vô đầu tiên)

**new\_entry\_point** = 0x00011000 – 0x00008400 + 0x000B000 = **0x00013C00**

* Tính địa chỉ **caption** và **text**

**0x00011040** – 0x00008400 = X – 0x000B000

Cộng thêm ImageBase, suy ra **X = 0x01013C40**.

Tương tự , **Y = 0x01013C60**

* Sau khi thực hiện được những câu lệnh chúng ta mong muốn thì phải trả về địa chỉ entry cũ của nó để nó hoạt động bình thường

Ta lại có công thức :

**old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA (2)**

* Ta dùng câu lệnh **jump** sau những câu lệnh mà đã chèn vào :
  + Xác định địa chỉ đầu tiên của câu lệnh jump và dùng công thức 1 để tính **jmp\_instruction\_VA**
* old\_entry\_point = AddressOfEntryPoint + ImageBase (dùng CFF Explorer với notepad.exe)
* Sau khi có 2 / 3 tham số
* Dễ dàng tính được **relative\_VA** dùng để chèn vào sau câu lệnh ta muốn thực thi để trở về **old\_entry\_point**

**old\_entry\_point** = **0x0100739D** chính là giátrị AddressOfEntryPoint ban đàu đã cộng ImageBase. (**AddressOfEntryPoint** và **ImageBase** coi trong **CFF Explorer**)

0000739D 01000000

* **jmp\_instruction\_VA (** *là địa chỉ đầu tiên của lệnh jump sau 5 câu lệnh thực thi mà chúng ta chèn vào***)**
  + **Áp dụng công thức 1 để tính :**
    - **RA – Section RA = VA – Section VA**
    - **0x011014 -** 0x00008400 = **jmp\_instruction\_VA** – 0x000B000
    - **jmp\_instruction\_VA** = **0x01013C14** (đã cộng imageBase)

**old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA (2)**

0x0100739D = 0x01013C14 + 5 + **relative\_VA**

Suy ra, **relative\_VA** = 0x0100739D – 5 – 0x01013C14 = **0xFFFF3784**

**B4**: Sử dụng HxD để **chèn** đoạn mã cùng với giá trị **Caption** và **Text** vào Notepad.exe. Lưu lại file

Coi địa chỉ của messagebox (Z) để gọi messagebox

push 0 ; 6a 00

push Caption ; 68 X 403C0101

push Text ; 68 **Y** 603C0101

push 0 ; 6a 00

call [MessageBoxW] ; ff15 Z 68120010

jmp Origianl\_Entry\_Point ; e9 8437FFFF

Điền ngược do little Endian

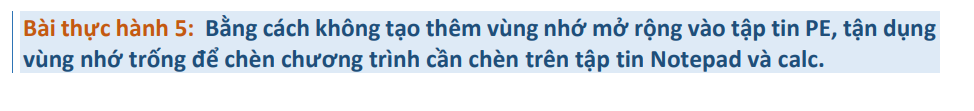
**B5**: Sử dụng CFF Explorer để thay đổi các giá trị **Virtual Size** , **Raw Size** vì ta đã insert 1000 byte vào sau **notepad.exe**

• Trong **Optional Headers**, tăng SizeOfImage lên 0x1000.

• Trong **Optional Headers**, chỉnh sửa **AddressOfEntryPoint** thành **0x00013C00**.

Trong Section Headers, thay đỏi .rsrc Section Header

Lưu lại và chạy sẽ có được kết quả

Bài 5 : 

PHẦN NOTEPAD

Để chèn mã vào tập tin PE thì ta phải tìm chỗ trống cần chèn vào

**B1**:Biên dịch chương trình cần chèn dưới chế độ **Release**, **Not Using Precompiled Headers**. Sử dụng IDA Pro để mở file PE và xem mã hợp ngữ của chương trình vừa biên dịch , nó sẽ có dạng như thế này :

Graphical user interface, application

Description automatically generated

push 0 ; 6a 00

push Caption ; 68 X

push Text ; 68 **Y**

push 0 ; 6a 00

call [MessageBoxW] ; ff15 Z

* Dùng IDA mở lên để xem cấu trúc câu lệnh:

**B3** : Chọn vị trí bất kì trong phần mới insert đó để thêm vào :

Ta chọn:

* Địa chỉ **10D50** trong vùng nhớ đã được mở rộng (địa chỉ chèn mã vào) – **new\_entry\_point**
* **10D90** để lưu trữ **Caption**
* **10DB0** để lưu trữ **Text**

Ta sử dụng công thức : **Offset = RA – Section RA = VA – Section VA (1)**

* Để có được section RA và Section VA thì ta phải dùng **CFF explorer** để mở file notepad.exe để xem 2 dữ liệu này
* **section RA**= 00008400
* **Section VA**= 0000B000
* Để tính địa chỉ trong bộ nhớ ảo khi ta load file PE lên (để thay thế địa chỉ này vào địa chỉ entry cũ của chương trình thì chương trình sẽ thực hiện những câu lệnh ta chèn vô đầu tiên)

**new\_entry\_point** = **10D50** – 0x00008400 + 0x000B000 = 0x**13950**

* Tính địa chỉ **caption** và **text**

**10D90** – 0x00008400 = X – 0x000B000

Cộng thêm ImageBase, suy ra **X = 1013990**.

Tương tự , **Y = 10139B0**

* Sau khi thực hiện được những câu lệnh chúng ta mong muốn thì phải trả về địa chỉ entry cũ của nó để nó hoạt động bình thường

Ta lại có công thức :

**old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA (2)**

* Ta dùng câu lệnh **jump** sau những câu lệnh mà đã chèn vào :
  + Xác định địa chỉ đầu tiên của câu lệnh jump và dùng công thức 1 để tính **jmp\_instruction\_VA**
* old\_entry\_point = AddressOfEntryPoint + ImageBase (dùng CFF Explorer với notepad.exe)
* Sau khi có 2 / 3 tham số
* Dễ dàng tính được **relative\_VA** dùng để chèn vào sau câu lệnh ta muốn thực thi để trở về **old\_entry\_point**

**old\_entry\_point** = **0x0100739D** chính là giátrị AddressOfEntryPoint ban đàu đã cộng ImageBase. (**AddressOfEntryPoint** và **ImageBase** coi trong **CFF Explorer**)

0000739D 01000000

* **jmp\_instruction\_VA (** *là địa chỉ đầu tiên của lệnh jump sau 5 câu lệnh thực thi mà chúng ta chèn vào***) lấy entry\_point + 0x14**
  + **Áp dụng công thức 1 để tính :**
    - **RA – Section RA = VA – Section VA**
    - **10D64 -** 0x00008400 = **jmp\_instruction\_VA** – 0x000B000 (CT 1)
    - **jmp\_instruction\_VA** = **1013964** (đã cộng imageBase)

**old\_entry\_point (trên VA) = jmp\_instruction\_VA + 5 + relative\_VA (2) (addr trên file PE)**

**0x0100739D** = **1013964** + 5 + **relative\_VA**

Suy ra, **relative\_VA** = 0x0100739D – 5 – **1013964** = **0x FFFF3A34**

**B4**: Sử dụng HxD để **chèn** đoạn mã cùng với giá trị **Caption** và **Text** vào Notepad.exe. Lưu lại file

Coi địa chỉ của messagebox (Z) để gọi messagebox

push 0 ; 6a 00

push Caption ; X 68 90390101

push Text ; **Y** 68 B0390101

push 0 ; 6a 00

call [MessageBoxW] ; Z ff15 68120001

jmp Origianl\_Entry\_Point ; e9 343AFFFF

Điền ngược do little Endian

**B5**: Sử dụng CFF Explorer để thay đổi các giá trị **Virtual Size** , **Raw Size**

• Trong **Optional Headers**, chỉnh sửa **AddressOfEntryPoint** thành **13750**.

Lưu lại và chạy sẽ có được kết quả

PHẦN CALC – không thêm byte nè

Để chèn mã vào tập tin PE thì ta phải tìm chỗ trống cần chèn vào

**B1**:Biên dịch chương trình cần chèn dưới chế độ **Release**, **Not Using Precompiled Headers**. Sử dụng IDA Pro để mở file PE và xem mã hợp ngữ của chương trình vừa biên dịch , nó sẽ có dạng như thế này :

Graphical user interface, application

Description automatically generated

push 0 ; 6a 00

push Caption ; 68 X

push Text ; 68 **Y**

push 0 ; 6a 00

call [MessageBoxW] ; ff15 Z

* Dùng IDA mở lên để xem cấu trúc câu lệnh:

**B3** : Chọn vị trí bất kì trong phần mới insert đó để thêm vào :

Ta chọn:

* Địa chỉ **1BF60** trong vùng nhớ đã được mở rộng (địa chỉ chèn mã vào) – **new\_entry\_point**
* **1BF80** để lưu trữ **Caption**
* **1BFA0** để lưu trữ **Text**

Ta sử dụng công thức : **Offset = RA – Section RA = VA – Section VA (1)**

* Để có được section RA và Section VA thì ta phải dùng **CFF explorer** để mở file notepad.exe để xem 2 dữ liệu này
* **section RA**= 13600
* **Section VA**= 16000
* Để tính địa chỉ trong bộ nhớ ảo khi ta load file PE lên (để thay thế địa chỉ này vào địa chỉ entry cũ của chương trình thì chương trình sẽ thực hiện những câu lệnh ta chèn vô đầu tiên)

**new\_entry\_point** = **1BF60** – 13600+ 16000= **1E960**

* Tính địa chỉ **caption** và **text**

**1BF80**– 13600= X – 16000

Cộng thêm ImageBase, suy ra **X = 101E980**.

Tương tự , **Y = 101E9A0**

* Sau khi thực hiện được những câu lệnh chúng ta mong muốn thì phải trả về địa chỉ entry cũ của nó để nó hoạt động bình thường

Ta lại có công thức :

**old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA (2)**

* Ta dùng câu lệnh **jump** sau những câu lệnh mà đã chèn vào :
  + Xác định địa chỉ đầu tiên của câu lệnh jump và dùng công thức 1 để tính **jmp\_instruction\_VA**
* old\_entry\_point = AddressOfEntryPoint + ImageBase (dùng CFF Explorer với notepad.exe)
* Sau khi có 2 / 3 tham số
* Dễ dàng tính được **relative\_VA** dùng để chèn vào sau câu lệnh ta muốn thực thi để trở về **old\_entry\_point**

**old\_entry\_point** = **1012475** chính là giátrị AddressOfEntryPoint ban đàu đã cộng ImageBase. (**AddressOfEntryPoint** và **ImageBase** coi trong **CFF Explorer**)

12475 01000000

* **jmp\_instruction\_VA (** *là địa chỉ đầu tiên của lệnh jump sau 5 câu lệnh thực thi mà chúng ta chèn vào***) lấy entry\_point + 0x14 (1BF60+14)**
  + **Áp dụng công thức 1 để tính :**
    - **RA – Section RA = VA – Section VA**
    - **1BF74 -** 13600= **jmp\_instruction\_VA** – 16000 **(CT 1)**
    - **jmp\_instruction\_VA** = **101E974** (đã cộng imageBase)

**old\_entry\_point = jmp\_instruction\_VA + 5 + relative\_VA (2)**

**1012475** = **101E974** + 5 + **relative\_VA**

Suy ra, **relative\_VA** = **1012475**– 5 – **101E974** = **0x FFFF3AFC**

**B4**: Sử dụng HxD để **chèn** đoạn mã cùng với giá trị **Caption** và **Text** vào Notepad.exe. Lưu lại file

Coi địa chỉ của messagebox (Z) để gọi messagebox

push 0 ; 6a 00

push Caption ; X 68 80E90101 101E980

push Text ; **Y** 68 **A0E90101** 101E9A0

push 0 ; 6a 00

call [MessageBoxW] ; Z ff15 A8110001

jmp Origianl\_Entry\_Point ; e9 FC3AFFFF

Điền ngược do little Endian

**B5**: Sử dụng CFF Explorer để thay đổi các giá trị **Virtual Size** , **Raw Size**

• Trong **Optional Headers**, chỉnh sửa **AddressOfEntryPoint** thành **1E960**

Lưu lại và chạy sẽ có được kết quả