BÁO CÁO THỰC HÀNH 1

Môn học: Thực Hành Cơ Chế Hoạt Động Của Mã Độc

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

GVHD: Nguyễn Hữu Quyền

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.N21.ATCL.1

STT	Họ và tên	MSSV	Email
1	Phan Hữu Luân	20521585	
2	Phạm Ngọc Lợi	20521560	
3	Nguyễn Trần Đức An	20520373	

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1		100%
2		100%
3		100%
4		90%
5		60%

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

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

- Đầu tiên ý tưởng bài toán như sau : cho 3 số a,b,c cho trước (được gán giá trị trong chương trình), đầu tiên ta so sánh a với b
  - Nếu a < = b thì a sẽ được đem đi so sánh với c
    - Nếu a < = c thì a là số nhỏ nhất
    - Nếu a > c thì c là số nhỏ nhất
  - Ngược lại nếu a > b thì b sẽ được đem đi so sánh với c
    - Nếu b <= c thì b là số nhỏ nhất
    - Nếu b > c thì c là số nhỏ nhất
- Hiện thực ý tưởng bằng mã assembly :
  - Đầu tiên ta sẽ khai báo các biến cần thiết trong **section. Data**:

```
section .data
msg db "so nho nhat la : "
len equ $- msg
a dd '9'
b dd '2'
c dd '3'
```

  - Trong phần bss, ta khai báo **min** để chứa giá trị min

```
segment .bss
min resb 2
```

  - Phần code trong **section.Text**

```
section .text
global _start
_start:
mov ecx, [a]
cmp ecx, [b]
jle check_a_and_b
mov ecx, [b]
check_a_and_b:
cmp ecx, [c]
jle _result
mov ecx, [c]
_result:
mov [min], ecx
```

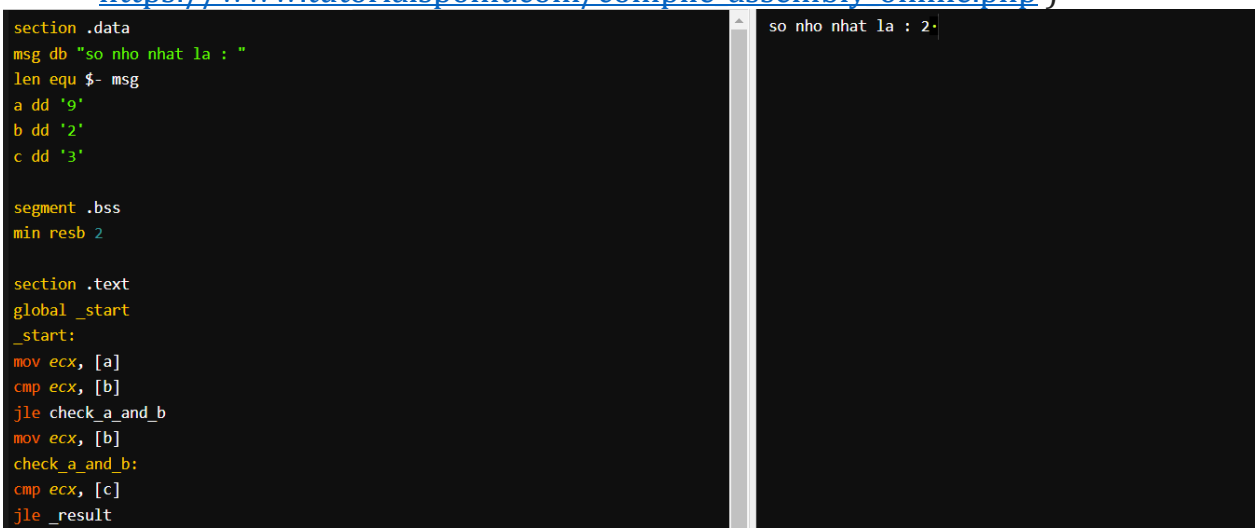
- Cuối cùng là các câu lệnh khai báo các thủ tục in ra màn hình cần thiết của system call và thực thi system call đó.

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

mov ecx,min
mov edx, 2
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)

int 0x80 ;call kernel
mov eax, 1
int 0x80
```

- Full Code và kết quả :
  - Thực hiện online trên : https://www.tutorialspoint.com/compile_assembly_online.php)



- **Code** :

```
section .data
msg db "so nho nhat la : "
len equ $- msg
a dd '9'
b dd '2'
c dd '3'

segment .bss
```

```
min resb 2

section .text
global _start
_start:
mov ecx, [a]
cmp ecx, [b]
jle check_a_and_b
mov ecx, [b]
check_a_and_b:
cmp ecx, [c]
jle _result
mov ecx, [c]
_result:
mov [min], ecx
mov ecx,msg
mov edx, len
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)
int 0x80 ;call kernel

mov ecx,min
mov edx, 2
mov ebx,1 ;file descriptor (stdout)
mov eax,4 ;system call number (sys_write)

int 0x80 ;call kernel
mov eax, 1
int 0x80
```

Câu 2 : **Viết chương trình chuyển đổi một số (number) 123 thành chuỗi '123'**

Sau đó thực hiện in ra màn hình số 123 (sử dụng gợi ý) :

- **Ý tưởng :** đầu tiên ta sẽ tách từng đơn vị (từng số) của số cho trước bằng cách chia cho 10 lấy phần dư , tiếp theo đó ta chuyển đổi ascii char và in ra màn hình chuỗi số .
- Hiện thực code assembly theo ý tưởng :
  - Phần **section. Data** và khai báo :

```
%assign SYS_EXIT 1
%assign SYS_WRITE 4
%assign STDOUT 1
;;; ----------------------------------------------------
;;; data section
;;; ----------------------------------------------------
section .data
x db 123
msgX db "x = "
```

- Phần **section. Text** chứa code :

```
section .text
global _start
_start:
;;; display x
mov ecx, msgX
mov edx, 4
call _printString
mov eax, 0
mov al, byte[x]
call _printDec
;;; ; exit
mov ebx, 0
mov eax, 1
int 0x80
_printString:
push eax
push ebx
mov eax,SYS_WRITE
mov ebx,STDOUT
int 0x80
pop ebx
pop eax
ret
```

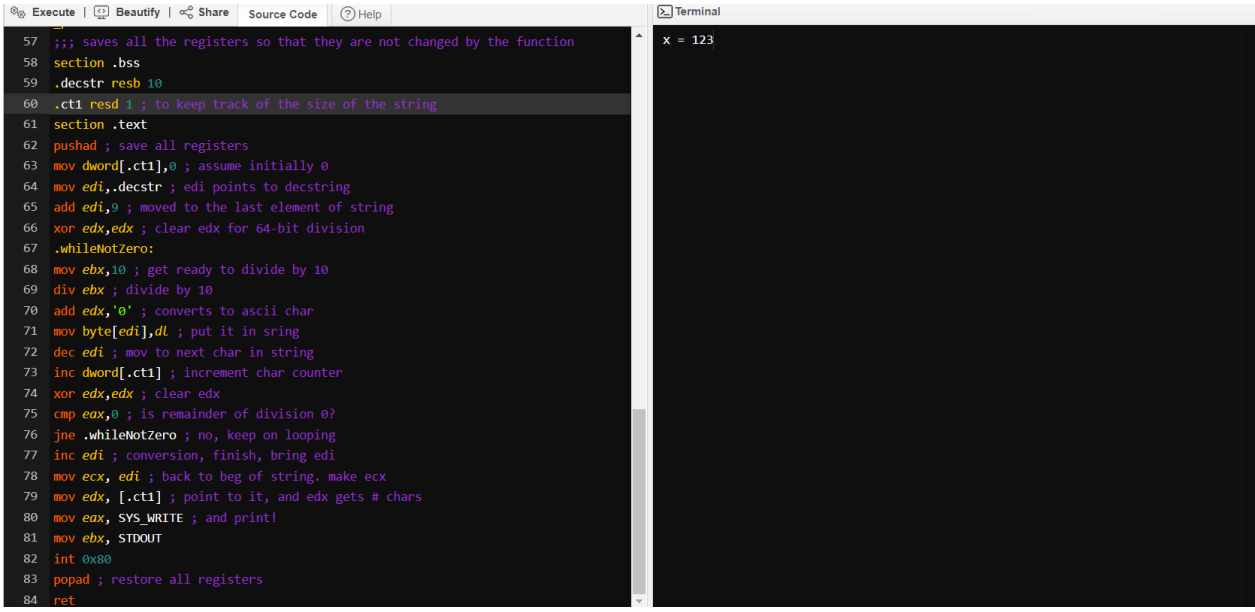
- Chuẩn bị các hàm để phục vụ cho việc xuất thông tin

```
_println:
section .data
.nl db 10
section .text
push ecx
push edx
mov ecx, .nl
mov edx, 1
call _printString
pop edx
pop ecx
ret
```

- Hàm println

```
_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
int 0x80
popad ; restore all registers
ret
```

  - Code chính dùng để convert sang dạng ascii char và xuất ra màn hình và cũng chứa các bước setup các thanh ghi.
- **Kết quả** :
  - Thực hiện online trên : https://www.tutorialspoint.com/compile_assembly_online.php



- **Full code :**

```
%assign SYS_EXIT 1
%assign SYS_WRITE 4
%assign STDOUT 1
;;; ----------------------------------------------------
;;; data section
;;; ----------------------------------------------------
section .data
x db 123
msgX db "x = "
;;; ----------------------------------------------------
;;; code section
;;; ----------------------------------------------------
section .text
global _start
_start:
;;; display x
mov ecx, msgX
mov edx, 4
call _printString
mov eax, 0
mov al, byte[x]
call _printDec
;;; ; exit
mov ebx, 0
mov eax, 1
int 0x80
_printString:
push eax
push ebx
```

```
mov eax,SYS_WRITE
mov ebx,STDOUT
int 0x80
pop ebx
pop eax
ret
;;; ; ------------------------------------------------------------
;;; ; _println put the cursor on the next line.
;;; ;
;;; ; Example:
;;; ; call _println
;;; ;
;;; ; REGISTERS MODIFIED: NONE
;;; ; ------------------------------------------------------------
_println:
section .data
.nl db 10
section .text
push ecx
push edx
mov ecx, .nl
mov edx, 1
call _printString
pop edx
pop ecx
ret
_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
int 0x80
popad ; restore all registers
ret
```

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

- **Ý tưởng :** vẫn là ý tưởng cũ, ta điều chỉnh để thêm bước nhập số và điều chỉnh kích thước của các số.
- **Thay đổi :**
  - Phần section data :

```
section  .data

   msg db "so nho nhat la :  "
   len equ $- msg
   input db 'moi nhap so : '
   lenip equ $-input


segment .bss
   min resd 10
   a resd 10
   b resd 10
   c resd 10
```

  - Trong phần **section. text** có thêm phần nhập và lưu trữ input cho từng biến :

```
_start:

   mov eax, 4
   mov ebx, 1
   mov ecx, input
   mov edx, lenip
   int 80h

   mov eax, 3
   mov ebx, 2
   mov ecx, a
   mov edx, 5          ;5 bytes
   int 80h
; set up so dau tien
   mov eax, 4
   mov ebx, 1
   mov ecx, input
   mov edx, lenip
   int 80h


   mov eax, 3
   mov ebx, 2
   mov ecx, b
   mov edx, 5          ;5 bytes
   int 80h
; set up so thu hai


   mov eax, 3
   mov ebx, 2
   mov ecx, c
   mov edx, 5          ; 5 bytes
   int 80h
; set up so thu ba
```

- **Kết quả** :
  - Thực hiện online trên : https://www.tutorialspoint.com/compile_assembly_online.php
  - Kết quả

```
section  .data

   msg db "so nho nhat la :  "
   len equ $- msg
   input db 'moi nhap so : '
   lenip equ $-input
```

```
moi nhap so : 33
moi nhap so : 3
moi nhap so : 333
so nho nhat la :  3
```

- **Full code :**

```
section  .data

  msg db "so nho nhat la :  "
  len equ $- msg
  input db 'moi nhap so : '
  lenip equ $-input


segment .bss
  min resd 10
  a resd 10
  b resd 10
  c resd 10

section  .text
  global _start

_start:

  mov eax, 4
  mov ebx, 1
  mov ecx, input
  mov edx, lenip
  int 80h

  mov eax, 3
  mov ebx, 2
  mov ecx, a
  mov edx, 5        ;5 bytes
  int 80h
; set up so dau tien
  mov eax, 4
  mov ebx, 1
  mov ecx, input
  mov edx, lenip
  int 80h


  mov eax, 3
  mov ebx, 2
  mov ecx, b
  mov edx, 5        ;5 bytes
  int 80h
; set up so thu hai
```

```
  mov eax, 4
  mov ebx, 1
  mov ecx, input
  mov edx, lenip
  int 80h


  mov eax, 3
  mov ebx, 2
  mov ecx, num3
  mov edx, 5        ; 5 bytes
  int 80h
; set up so thu ba

  mov   ecx, [a]
  cmp   ecx, [b]
  jle    check_a_and_b
  mov   ecx, [b]

check_a_and_b:

  cmp   ecx, [c]
  jle    _result
  mov   ecx, [c]

_result:

  mov   [smallest], ecx
  mov   ecx,msg
  mov   edx, len
  mov   ebx,1
  mov   eax,4
  int   0x80

  mov   ecx,smallest
  mov   edx, 2
  mov   ebx,1
  mov   eax,4
  int   0x80

  mov   eax, 1
  int   80h
```

HẾT