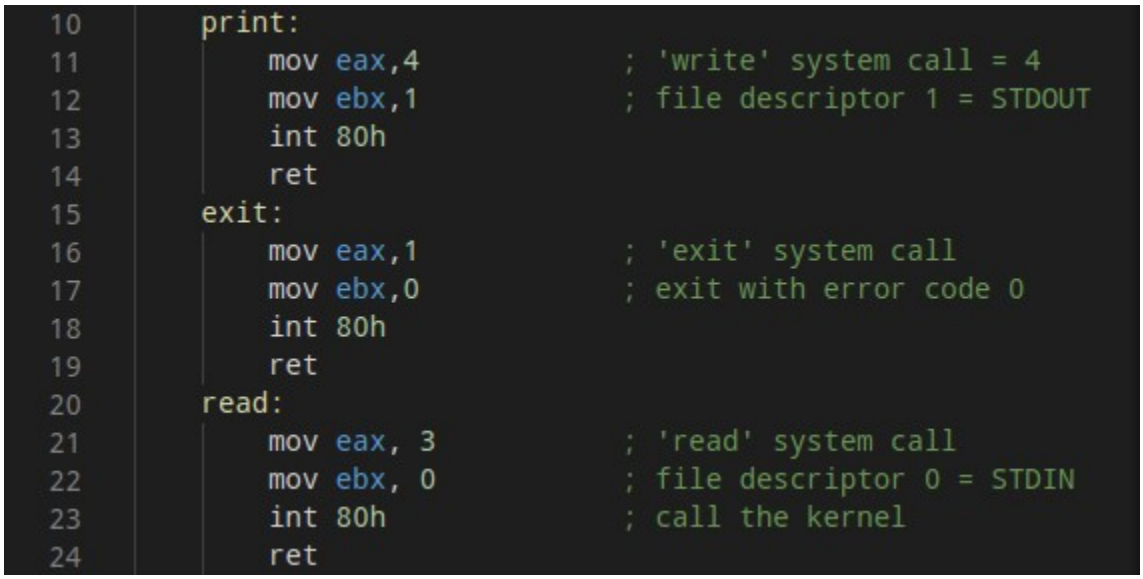
VCS Homework
Week 1 Writeups

Gửi anh Phong mentor đẹp giai

Sau 2 tuần đầu tiên ở VCS và cũng là 2 tuần đầu làm quen với assemby trên winndows và linux, bản thân em đã đúc kết được chút kiến thức cơ bản về assembly. Trong các bài code của em đều có cấu trúc giống nhau và một số quy định trước như sau:

- Linux sử dụng nasm làm assembler. gcc làm linker cho file elf 32 bit. ld làm linker cho file elf 64 bit
- Windows sử dụng masm làm assembler, link (có sẵn trong bộ cài masm) làm linker
- Một số hàm em viết sẵn trong linux:



```
10      print:
11          mov eax,4            ; 'write' system call = 4
12          mov ebx,1            ; file descriptor 1 = STDOUT
13          int 80h
14          ret
15      exit:
16          mov eax,1            ; 'exit' system call
17          mov ebx,0            ; exit with error code 0
18          int 80h
19          ret
20      read:
21          mov eax, 3           ; 'read' system call
22          mov ebx, 0           ; file descriptor 0 = STDIN
23          int 80h              ; call the kernel
24          ret
```

Challenge 1 - Week 1

ASM #1 - Hello World - 100 điểm
Viết chương trình in ra màn hình dòng chữ "Hello, World!"
Yêu cầu:
- Hệ điều hành: Windows & Linux
- Hiểu các thành phần của 1 chương trình assembly, giải thích từng dòng code

Ý tưởng:

Khai báo 1 xâu kí tự “Hello, World!” sau đó in ra màn hình

Cú pháp trong MASM:

```
.data
HelloWorld db "Hello, World!", 0Ah, 0
```

Sử dụng StdOut in xâu ra màn hình

```
push offset HelloWorld      ; print HelloWorld "Hello, World!"
call StdOut
```

Cú pháp trong NASM:

```
section .data
        msg db "Hello, World!",0Ah
        msg_len equ $-msg
```

```
        mov ecx, msg                    ; address of string to output
        mov edx, msg_len                ; number of bytes
        call print
```

Challenge 2 - Week 2

ASM #2 - Echo - 100 điểm
Viết chương trình nhập vào đoạn văn bản, in ra đoạn văn bản vừa nhập
Yêu cầu:
- Hệ điều hành: Windows & Linux
- Chiều dài đoạn văn bản: 32 ký tự

Ý tưởng:

Khai báo biến có độ lớn hơn 33 kí tự, gọi hàm đọc dữ liệu đầu vào từ máy tính sau đó lưu vào biến vừa khai báo. Gọi biến vừa lưu dữ liệu và hàm in ra màn hình console.

Cú pháp trong MASM

Khai báo biến

```
14  max_size equ 33
15  .data?
16      input_text db max_size dup(?)
17  .data
18      msg1 db "Input: ", 0
19      msg2 db "Text: ", 0
20      newline db 0Ah, 0
```

Đọc dữ liệu từ StdIn

```
28      push max_size                   ; size of input_text
29      push offset input_text          ; address of input_text
30      call StdIn                      ; read max_size bytes from StdIn and store in input_text variable
```

Viết dữ liệu ra StdOut

```
34
35      push offset input_text              ; print input_text
36      call StdOut
```

Cú pháp trong NASM

Khai báo biến

```
2   section .data
3       msg db "Input: ", 0Ah
4       msg_len equ $-msg
5       msg2 db "Text: ", 0Ah
6       msg2_len equ $-msg2
7       max_size equ 33
8   section .bss
9       input resb max_size
```

Đọc dữ liệu từ StdIn

```
32      mov ecx, input          ; take input data from STDIN
33      mov edx, 32
34      call read
```

Viết dữ liệu ra StdOut

```
39
40      mov ecx, input          ; print input variable to STDOUT
41      mov edx, 32
42      call print
```

Challenge 3 - Week 1

ASM #3 - Uppercase - 100 điểm
Viết chương trình nhập vào chuỗi văn bản, in ra chuỗi văn bản được in hoa
Yêu cầu:
- Hệ điều hành: Windows & Linux
- Chiều dài đoạn văn bản: 32 ký tự

Ý tưởng:

Thực hiện công việc nhập dữ liệu từ StdIn và lưu vào biến

Duyệt xâu vừa nhập từ đầu xâu. Kiểm tra kí tự là chữ hoa (so sánh với byte ‘A’ và byte ‘Z’) hay chữ thường (so sánh với byte ‘a’ và ‘z’).

- Nếu là chữ hoa thì bỏ qua
- Nếu không là chữ hoa → Kiểm tra chữ thường
- Nếu là chữ thường → Chuyển kí tự thành chữ hoa bằng cách -20h thứ tự trong bảng ascii
- Nếu không là chữ thường → Bỏ qua và tiến đến kí tự tiếp theo
- Nếu kí tự tiếp là NULL → Dừng vòng lặp và trả kết quả

Cú pháp trong NASM:

```
mov ecx, input          ; read user input from STDIN and store to input variable
mov edx, 32
call read
```

Đọc dữ liệu từ StdIn và lưu vào biến input

```
41      mov ecx, max_size ;place length of input text into ecx
42      mov ebp, input  ;place address of input text into ebp
43      dec ebp
44
45      upper:
46          cmp byte [ebp+ecx], 0x41 ;compare char vs 'A'
47          jb next_char    ;if < 'A' -> next_char
48          cmp byte [ebp+ecx], 0x5A ; compare char vs 'Z'
49          ja lower    ;
50          jmp next_char;
51      lower:
52          cmp byte [ebp+ecx], 0x61    ;compare char vs 'a'
53          jb next_char;
54          cmp byte [ebp+ecx], 0x7a    ;compare char vs 'z'
55          ja next_char;
56          sub byte [ebp+ecx], 0x20    ;convert to lowercase
57      next_char:
58          dec ecx
59          jnz upper
60
```

Hàm đổi lowercase thành uppercase

Cú pháp trong MASM:

```
14
15  max_size equ 32                         ; max_size of input/output string
16  .data?
17      text db max_size dup(?)
18  .data
19      msg1 db "Input: ", 0Ah, 0
20      msg2 db "Text (uppercase): ", 0Ah, 0
21      newline db 0Ah, 0
```

Khai báo biến

```
23  .code
24      upper proc
25          cmp byte ptr [ebp+ecx], 41h     ; compare char vs 'A'
26          jb @next_char                   ; if < 'A' -> next_char
27          cmp byte ptr [ebp+ecx], 5Ah     ; compare char vs 'Z'
28          ja @lower                       ; if char is not upper -> check lower
29          jmp @next_char;                 ; else next_char
30      @lower:
31          cmp byte ptr [ebp+ecx], 61h     ; compare char vs 'a'
32          jb @next_char;
33          cmp byte ptr [ebp+ecx], 7ah     ; compare char vs 'z'
34          ja @next_char;                  ; if char not lower -> next-char
35          sub byte ptr [ebp+ecx], 20h     ; else convert to lowercase
36      @next_char:
37          dec ecx                         ; decrease ecx (counter)
38          jnz upper                       ; if ecx != 0 -> loop
39          ret                             ; else return
40      upper endp
```

Hàm đổi lowercase thành uppercase

Challenge 4 - Week 1

ASM #4 - Simple Addition - 100 điểm
Viết chương trình tính tổng 2 số nguyên dương nhập vào từ bàn phím
Yêu cầu:
- Hệ điều hành: Windows & Linux
- Độ lớn tối đa số nhập vào: 31-bit (2^31 - 1)

Ý tưởng:

Nhận dữ liệu người dùng và ghi vào biến. Vì dữ liệu đầu vào là 1 chuỗi kí tự nên em viết 2 hàm atoi (đổi xâu chữ số thành số) và itoa (đổi số thành xâu chữ số). Thực hiện hàm atoi với 2 xâu chữ số đầu vào, làm phép tính cộng như với 2 số dương rồi chuyển kết quả đó sang xâu chữ số để in ra StdOut (sử dụng itoa).

Hàm atoi (eax là kết quả, ecx là kí tự đang duyệt):eax

- eax = 0
- Duyệt từng kí tự của xâu chữ số cần đổi
- Đổi chữ số đang duyệt sang giá trị nguyên bằng cách sub [kí tự], ‘0’
- eax = eax * 10 + ecx
- Khi ecx là NULL → Dừng vòng lặp →Trả kết quả ở eax

Hàm itoa (eax là giá trị nguyên cần đổi):

- Chia eax cho 10
- Phần dư là chữ số cuối cùng
- Đặt eax = thương hay eax = eax / 10
- Thực hiện vòng lặp đến khi eax = 0
- Sau mỗi lần lặp ta được số dư là chữ số tiếp theo thứ tự từ phải sang trái
- eax = 0 →Dừng lặp →Trả kết quả là xâu kí tự

Cú pháp trong NASM:

```
3   section .data
4       msg1 db "Input number A: "
5       msg1_len equ $-msg1
6       msg2 db "Input number B: "
7       msg2_len equ $-msg2
8       msg_error db "Invalid input!", 0Ah
9       msg_error_len equ $-msg_error
10      msg_result db "Sum of A and B: "
11      msg_result_len equ $-msg_result
12      max_size equ 11
13      newLn db 0Ah, 0h
14
15  section .bss
16      number1 resb max_size
17      number1_int resb 8
18      number2 resb max_size
19      number2_int resb 8
20      sum resb max_size
21      sum_int resb 8
22
```

Khai báo biến

```
44      ;;  Input:   edi - input string
45      ;;  Output:  eax - result in integer
46      atoi:
47          xor eax, eax        ; set eax = 0
48      .loop:
49          movzx ecx, byte [edi]   ; ecx = first byte of edi - input string
50          sub ecx, '0'            ; convert num -> int
51          jb .done                ;
52
53          lea eax, [eax*4+eax]    ; eax = eax * 5
54          lea eax, [eax*2+ecx]    ; eax = eax * 5 * 2 + ecx
55          inc edi                 ; next character of string
56          jmp .loop
57      .done:
58          ret
```

Hàm atoi

```
62      itoa:
63          add esi, max_size       ; point to the last of result buffer
64          mov byte [esi], 0h      ; set the last byte of result buffser to null
65          mov ebx, 10             ; ebx = 10
66      .loop:
67          xor edx, edx            ; edx = 0
68          div ebx                 ; ebx = divisor
69          add dl, '0'             ; dl - remainder
70          dec esi                 ; point to the next left byte
71          mov [esi], dl           ; mov remainder to the current byte
72          test eax, eax           ; if (eax == 0) ?
73          jnz .loop               ; if not -> loop
74          mov eax, esi            ; else return result to eax
75          ret
```

Hàm itoa

Cú pháp trong MASM:

```
14
15  max_size  EQU 12                        ; max_size of input/output string
16
17  .data?
18      number1 db max_size dup(?)
19      number2 db max_size dup(?)
20
21  .data
22      msg1 db "Input number A: ", 0h
23      msg2 db "Input number B: ", 0h
24      msg3 db "Sum A + B:      ", 0h
25      sum db max_size dup(?)
26      newline db 0Ah, 0
```

Khai báo biến

```
49      ;;  Input:   edi - input string
50      ;;  Output:  eax - result in integer
51      atoi proc uses ebx ecx edx esi edi
52          xor eax, eax            ; set eax = 0
53      @loop:
54          movzx ecx, byte ptr [edi]   ; ecx = first byte of edi - input string
55          sub ecx, '0'            ; convert num -> int
56          jb @done                ;
57
58          lea eax, [eax*4+eax]    ; eax = eax * 5
59          lea eax, [eax*2+ecx]    ; eax = eax * 5 * 2 + ecx
60          inc edi                 ; next character of string
61          jmp @loop
62      @done:
63          ret
64      atoi endp
```

Hàm atoi

```
;;  Input:  eax - integer
;;          esi - pointer to result buffer
;;  Output: in buffer
itoa proc uses ebx edx esi
    add esi, max_size      ; point to the last of result buffer
    dec esi
    mov byte ptr [esi], 0h      ; set the last byte of result buffser to null
    mov ebx, 10             ; ebx = 10
@loop:
    xor edx, edx            ; edx = 0
    div ebx                 ; divide eax by divisor ebx = 10
    add dl, '0'             ; dl - remainder
    dec esi                 ; point to the next left byte
    mov [esi], dl           ; mov remainder to the current byte
    test eax, eax           ; if (eax == 0) ?
    jnz @loop               ; if not -> loop
    mov eax, esi            ; else return result to eax
    ret
itoa endp
```

Hàm itoa