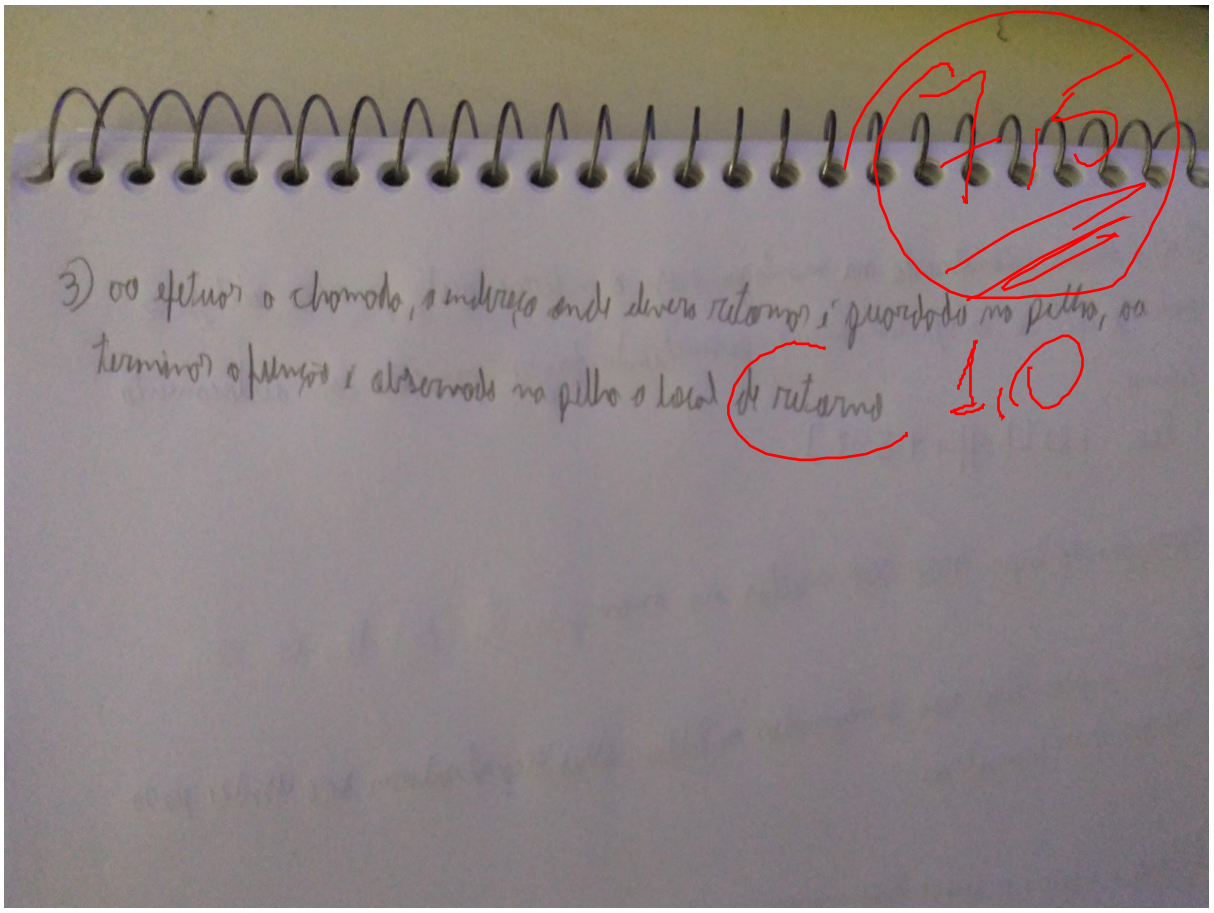


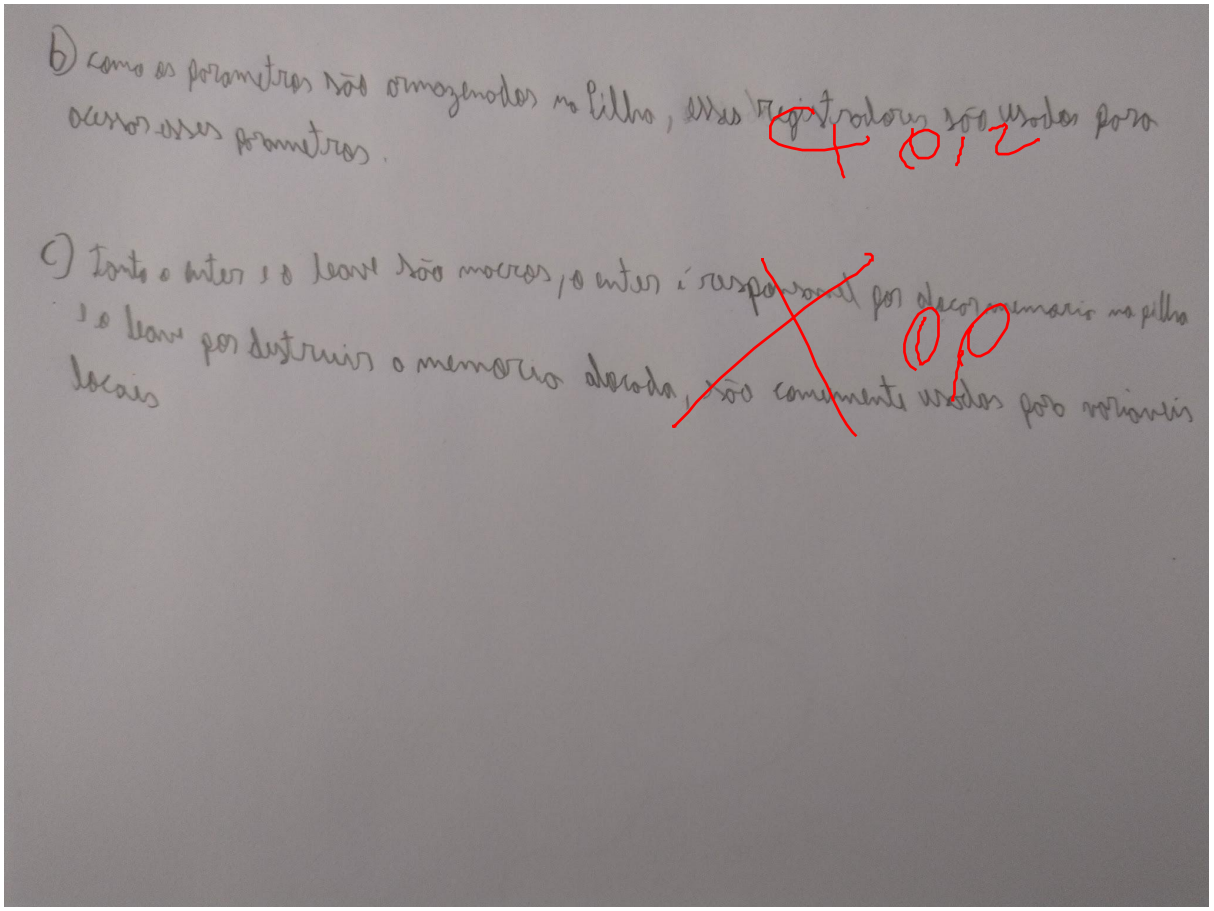
3) Ao efetuar a chamada, o endereço onde deve retornar é guardado na pilha, ao término da função é desempilhado na pilha o local de retorno

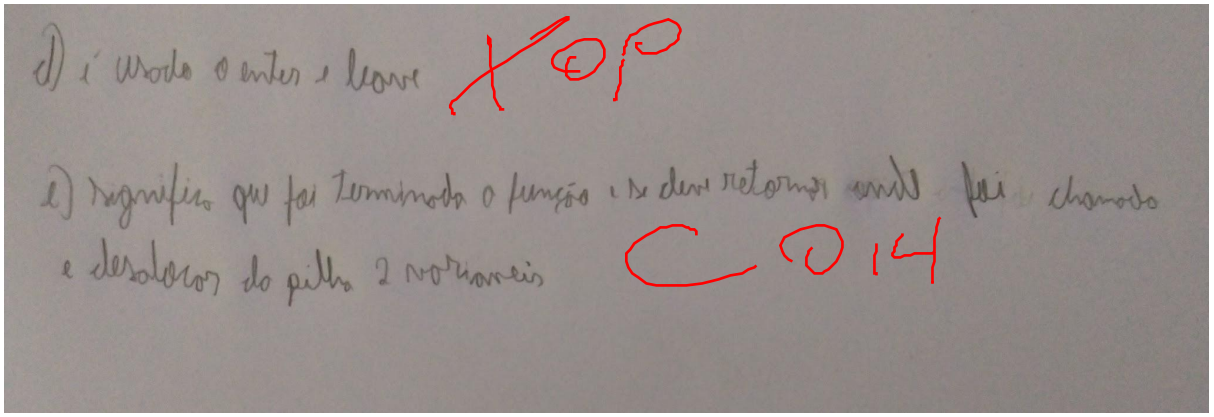
1,0

5) a) o mov é construido na montagem e usa o endereço real, já o leq utiliza o endereço efetivo, ou seja um ponteiro, permitindo fazer operações com deslocamento como:

leq EBX[buffer + ESI]

operações do tipo não são aceitos no mov

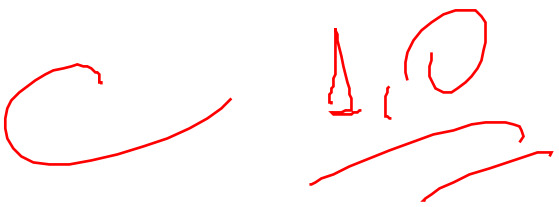




1. (1,0 ponto) FUP que leia um texto do teclado e a seguir grave 10 vezes o mesmo texto em um arquivo "texto10.txt".

```
;---------------------------------------------------------
section .data
        buffer times 255 db 0
        tamlido db 0
        nomeArquivo db "texto10.txt",0
        TAM_NOME_ARQUIVO equ $ - nomeArquivo
        fd      dd 0
        cont db 10
```

```
section .text
        GLOBAL _start
        _start:
        ;lendo do teclado

        mov rax,0   ; codigo syscall read
        mov rdi,0   ; teclado
        mov rsi,buffer
        mov rdx,255
        syscall
        dec rax
        mov byte[buffer+rax],0 ;removendo o \n e colocando o \0 no lugar

        mov byte[tamlido],al ;guardando o tamanho que foi lido




        ;criando o arquivo
        mov rax,2         ; open
        mov rdi,nomeArquivo       ; str nomeArquivo
        mov rsi,1|64        ; modo de criacao O_CREAT e somente escrita
        mov rdx,0o0750      ; permissao
        syscall

        mov byte[fd],al

        escrevendo:
        mov rax,1
        mov rdi,[fd]
        mov rsi,buffer
        movsx rdx,byte[tamlido]
        syscall

        dec byte[cont]
        cmp byte[cont],0
        jne escrevendo

        ;fechando arquivo
        mov rax,3
        mov rdi,[fd]
        syscall
        ;fechando programa
        mov rax,60
        mov rdi,0
        syscall
;-----------------------------------------------------
```



2)

```
;-------------------------------------------------------
extern scanf
extern exit
extern printf

%macro quebralinha 0
        mov rax,1
        mov rdi,1
        mov rsi,10
        mov rdx,1
        syscall
%endmacro

section .data
        entrada db "entre com o item matB[%d,%d]:",0
        saida db "%lf ",0
        fmt db "%lf",0
        linha db 0
        coluna db 0
        cont db 0
        matA dq 1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,8.9

        matB dq 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0

        matC dq 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0

        N_LINHAS equ 3
        N_COLS equ 3

section .text
GLOBAL main
main:
        push rbp ;padrao de abertura
   xor rbx,rbx
   xor rax,rax
   entrada_Matriz:
   mov rdi,entrada
   movsx rsi,byte[linha]
   movsx rdx,byte[coluna]

   call printf
        mov rdi, fmt
   ;mov rsi, qword[matB+rbx]
   mov rax,1
   call scanf

   movsd qword[matB+rbx],xmm0
```

```
    inc rbx
    inc byte[linha]
    cmp byte[linha],3
    jne entrada_Matriz
    mov byte[linha],0
    inc byte[coluna]
    cmp byte[coluna],3
    jne entrada_Matriz
    ;fazendo a soma de matrizes

    mov byte[cont],0
    calculando:
    movsx rbx,byte[cont]
    movsd xmm0,[matC+rbx]
    addsd xmm0,[matA+rbx]
    addsd xmm0,[matB+rbx]
    movsd [matC+rbx],xmm0
    inc byte[cont]
    cmp byte[cont],9
    jne calculando

    mov byte[cont],0
    mov byte[linha],0
    mov byte[coluna],0

    xor rbx,rbx
    saindo:
    mov rdi, saida
    movsd xmm0,qword [matC+rbx] ;
    mov rax,1
    call printf
    inc rbx

    inc byte[linha]
    cmp byte[linha],3
    jne saindo
    mov byte[linha],0
    inc byte[coluna]
    cmp byte[coluna],3
    jne saindo


    pop rbp;
    call exit
;-----------------------------------
```

4. (3,0 pontos) FUP que:
Crie um arquivo
Leia um texto e grave nesse arquivo
Feche o arquivo
Leia um novo texto e também grave nesse arquivo, sem perder o texto anterior (modo append).
Feche o arquivo

```
;------------------------------------------------------------------------
section .data
  TECLADO              equ 0
  TELA                 equ 1
  fd                                  dd 0 ;descritor do arquivo

  buffer times 255 db 0
  tamlido db 0

  nomeArquivo db "arquivoManipulado.txt",0
  TAM_NOME_ARQUIVO equ $ - nomeArquivo




section .text

global _start
  _start:
  ;criando arquivo
  mov rax,2         ; open
  mov rdi,nomeArquivo        ; str nomeArquivo
  mov rsi,1|64        ; modo de criacao O_CREAT e somente escrita
  mov rdx,0o0750       ; permissao
  syscall
  mov byte[fd],al

  ;lendo do teclado
  mov rax,0   ; codigo syscall read
  mov rdi,0   ; teclado
  mov rsi,buffer  ;buffer que ira guardar o texto lido
  mov rdx,255    ;tamanho oo buffer
  syscall
  dec rax
  mov byte[buffer+rax],0;removendo o \n e colocando o \0 no lugar

  mov byte[tamlido],al;guardando o tamanho que foi lido
```

```
;escrevendo no arquivo
mov rax,1
mov rdi,[fd]
mov rsi,buffer
movsx rdx,byte[tamlido]
syscall

;fechando arquivo
mov rax,3
mov rdi,[fd]
syscall

;abrindo como append
mov rax,2         ; open
mov rdi,nomeArquivo        ; str nomeArquivo
mov rsi,1|1024        ; modo de criacao O_CREAT e somente escrita
syscall
mov byte[fd],al

;lendo do teclado o novo texto
mov rax,0   ; codigo syscall read
mov rdi,0   ; teclado
mov rsi,buffer  ;buffer que ira guardar o texto lido
mov rdx,255    ;tamanho o buffer
syscall

dec rax
mov byte[buffer+rax],0;removendo o \n e colocando o \0 no lugar

mov byte[tamlido],al;guardando o tamanho que foi lido

;escrevendo no arquivo
mov rax,1
mov rdi,[fd]
mov rsi,buffer
movsx rdx,byte[tamlido]
syscall

;fechando arquivo
mov rax,3
mov rdi,[fd]
syscall

;fechando programa
mov rax,60
mov rdi,0
syscall
```