4,1

S) MOV: a instrução copia os dados referido pelo seu segundo operando para a localização referido pelo primeiro operando (MOV OPERANDO1, OPERANDO2)

LEA: a instrução carrega o endereço especificado no seu segundo operando para o registrador especificado no seu primeiro operando. (LEA OPERANDO1, OPERANDO2)

Enquanto movimentações de registradores para registradores são possíveis usando o comando MOV, movimentações de memória para memória não são.

R) RSP é o ponteiro para a pilha, aponta para o topo da pilha RBP é a registrador preservado, geralmente usado para guardar valores antigos do ponteiro de pilha ou para guardar a variável da pilha.

O) ENTER, cria uma stack frame para um procedimento o primeiro operando especifica o tamanho da stack frame e segundo operando fala o nível de aninhamento lexical do procedimento. O nível de aninhamento determina o número de ponteiros para stack frames que são copiados para o área de display da nova stack frame da frame anterior.

LEAVE reverte as ações do ENTER, desfaz cópias do ponteiro da frame no ponteiro de pilha e libera o espaço da pilha usado pelo procedimento para suas variáveis locais

D S T Q Q S S
D L M M J V S

(d) RET transfere o controle do programa para um endereço de retorno localizado no topo da pilha. Geralmente o endereço é colocado na pilha por uma instrução CALL. O operando opcional especifica o número de bytes da pilha que serão liberados, depois que o endereço de retorno é tirado.

RET 16 irá transferir o controle do programa para um endereço de retorno no topo da pilha e depois irá liberar 16 bytes da pilha.

Questão 1:

```
%macro _len 1
        mov rdi, %1
        xor al, al
        mov rcx, -1
        cld
        repnz scasb

        not rcx
        dec rcx
%endmacro

section .data
        printfras1 db "Frase para salvar no arquivo: ", 0
        len_fras1 equ $-printfras1
        arq db "texto10.txt", 0
        str1 times 50 db 0

        tStr1 dq 0

section .text
        global main
main:
        call _printStr1
        call _getFras1
        _len str1
        mov qword [tStr1], rcx
        call _writearq1

        mov rax, 60
        mov rdi, 0
        syscall


_printStr1:
        mov rax, 1
        mov rdi, 1
        mov rsi, printfras1
        mov rdx, len_fras1
        syscall
        ret

_getFras1:
        mov rax, 0
        mov rdi, 0
        mov rsi, str1
        mov rdx, 50
        syscall
        ret

_writearq1:
        mov rax, 2
```

```
mov rdi, arq
mov rsi, 64+1
mov rdx, 0644o
mov rcx, 0
syscall

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov eax, 10


mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
```

```
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10
```

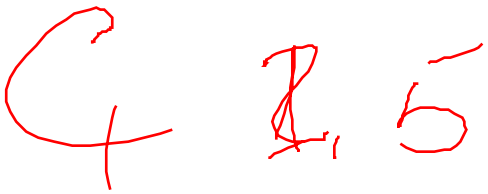
```
push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
```

```
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 2
mov rdi, arq
mov rsi, 1024+1
mov rdx, 0644o
mov rcx, 0
syscall
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str1
mov rdx, qword [tStr1]
syscall

mov rax, 3
pop rdi
syscall

mov rax, 60
mov rdi, 0
syscall

ret
```

Questão 4:

```
%macro _len 1
        mov rdi, %1
        xor al, al
        mov rcx, -1
        cld
        repnz scasb

        not rcx
        dec rcx
%endmacro

section .data
        printfras1 db "Frase para salvar no arquivo 1: ", 0
        len_fras1 equ $-printfras1
        printfras2 db "Frase para salvar no segundo arquivo: ", 0
        len_fras2 equ $-printfras2
        arq db "texto.txt", 0
        str1 times 50 db 0
        str2 times 50 db 0

        tStr1 dq 0
        tStr2 dq 0

section .text
        global _start
_start:
        call _printStr1
        call _getstr1
        _len str1
        mov qword [tStr1], rcx
        call _writestr1

        call _printStr2
        call _getstr2
        _len str2
        mov qword [tStr2], rcx
        call _writestr2

        mov rax, 60
        mov rdi, 0
        syscall


_printStr1:
        mov rax, 1
        mov rdi, 1
        mov rsi, printfras1
        mov rdx, len_fras1
        syscall
        ret
```

```
_getstr1:
        mov rax, 0
        mov rdi, 0
        mov rsi, str1
        mov rdx, 50
        syscall
        ret

_writestr1:
        mov rax, 2
        mov rdi, arq
        mov rsi, 64+1
        mov rdx, 0644o
        mov rcx, 0
        syscall

        push rax
        mov rdi, rax
        mov rax, 1
        mov rsi, str1
        mov rdx, qword [tStr1]
        syscall

        mov rax, 3
        pop rdi
        syscall

        ret

_printStr2:
        mov rax, 1
        mov rdi, 1
        mov rsi, printfras2
        mov rdx, len_fras2
        syscall
        ret

_getstr2:
        mov rax, 0
        mov rdi, 0
        mov rsi, str2
        mov rdx, 50
        syscall
        ret

_writestr2:
        mov rax, 2
        mov rdi, arq
        mov rsi, 1024+1
        mov rdx, 0644o
        mov rcx, 0
        syscall
```



```
mov CL, 10

push rax
mov rdi, rax
mov rax, 1
mov rsi, str2
mov rdx, qword [tStr2]
syscall

mov rax, 3
pop rdi
syscall
ret
```