1) Dada a seguinte tabela abaixo:

kelvin dq 90,380,250,371

celsius dq 0, 0, 0, 0

Faça um programa que transfira para a tabela (vetor) celsius o equivalente em graus centigrados sabendo que

celsius = kelvin - 273

Imprima a tabela kelvin e o equivalente em celsius.

```
section .data
    kelvin dq 90,380,250,371
    celsius dq 0, 0, 0, 0

main:
    push   rbp
    mov    rbp, rsp
    sub    rsp, 48
    mov    DWORD PTR [rbp-32], 90
    mov    DWORD PTR [rbp-28], 380
    mov    DWORD PTR [rbp-24], 250
    mov    DWORD PTR [rbp-20], 371
    mov    DWORD PTR [rbp-4], 0
    jmp    .L2
.L3:
    mov    eax, DWORD PTR [rbp-4]
    cdqe
    mov    eax, DWORD PTR [rbp-32+rax*4]
    lea    edx, [rax-273]
    mov    eax, DWORD PTR [rbp-4]
    cdqe
    mov    DWORD PTR [rbp-48+rax*4], edx
    add    DWORD PTR [rbp-4], 1
.L2:
    cmp    DWORD PTR [rbp-4], 3
    jle    .L3
    mov    DWORD PTR [rbp-4], 0
    jmp    .L4
.L5:
    mov    eax, DWORD PTR [rbp-4]
    cdqe
    mov    edx, DWORD PTR [rbp-48+rax*4]
    mov    eax, DWORD PTR [rbp-4]
    cdqe
    mov    eax, DWORD PTR [rbp-32+rax*4]
    mov    esi, eax
    mov    edi, OFFSET FLAT:.LC0
    mov    eax, 0
    call   printf
    add    DWORD PTR [rbp-4], 1
```

```
.L4:
    cmp     DWORD PTR [rbp-4], 3
    jle     .L5
    mov     eax, 0
    leave
    ret
```

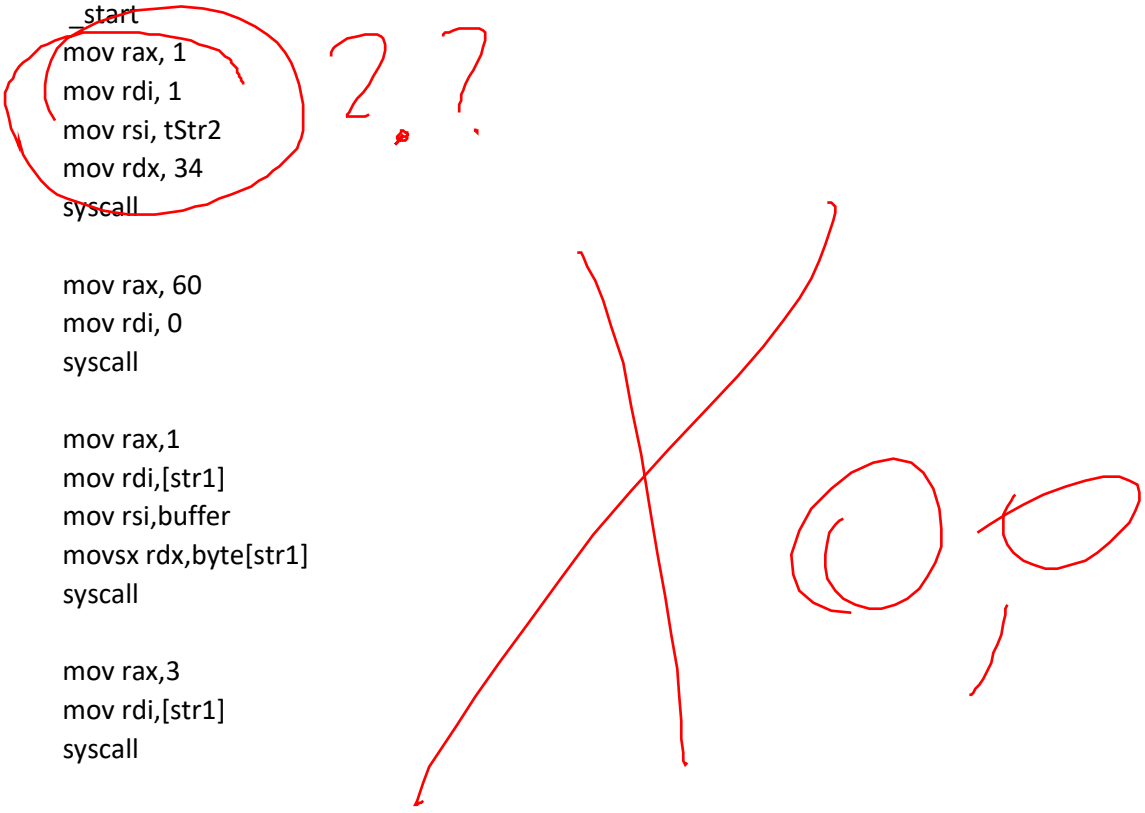
2) Desenvolva o codigo em assembly solicitando uma função fat(5), apresentando o código da mesma em assembly. A seguir, imprima o resultado na tela usando a função externa da linguagem C printf. Mostre ainda como seria a compilação da mesma.

```
extern printf

section .data
main:
    push    rbp
    mov     rbp, rsp
    cmp     DWORD PTR [rbp-12], 0
    js      .L2
    mov     DWORD PTR [rbp-8], 1
    mov     eax, DWORD PTR [rbp-12]
    mov     DWORD PTR [rbp-4], eax
    jmp     .L3
    call printf
.L4:
    mov     eax, DWORD PTR [rbp-8]
    imul    eax, DWORD PTR [rbp-4]
    mov     DWORD PTR [rbp-8], eax
    sub     DWORD PTR [rbp-4], 1
.L3:
    cmp     DWORD PTR [rbp-4], 0
    jg      .L4
    jmp     .L7
.L2:
    mov     eax, 0
    jmp     .L6
.L7:
    mov     eax, 0
.L6:
    pop     rbp
    ret
```

3) Dada as seguintes variáveis na section .data:
str1: db ’Computação é difícil, se não estudar",0
str2: db ’difícil’,0
str3: db ’fácil, se’,0
tStr2 db 0
tStr3 db 0
FUP que:

1) Utilize as instruções de manipulação de string (MOVS, STOS, LOADS, CMPS, etc) em assembly, para calcular o tamanho da str2 e str3, guardando nas variáveis tStr2 e tStr3.
2) Localize a posição da str2 em str1 através de instruções de manipulação de strings.
3) Retire de str1 a string str2, inserindo na posição a string str2.
4) Grave em um arquivo ’exame.txt’ a str1 modificada.

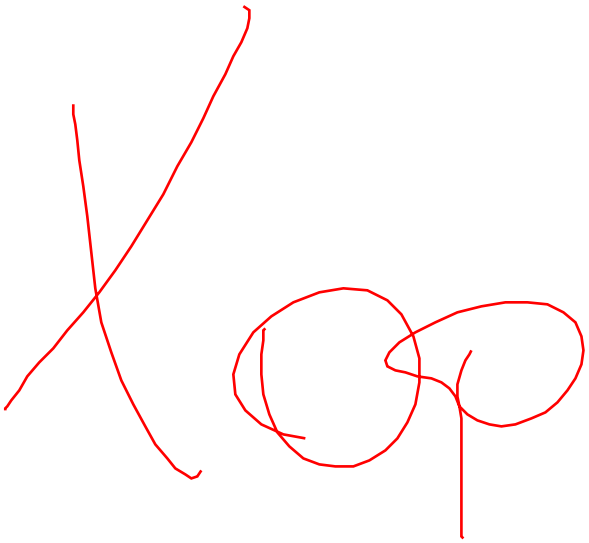
```
section .data
    str1: db ’Computação é difícil, se não estudar",0
    str2: db ’difícil’,0
    str3: db ’fácil, se’,0
    tStr2 db 0
    tStr3 db 0

    buffer times 255 db 0
    arquivo db "exame.txt",0

global _start
    _start
    mov rax, 1
    mov rdi, 1
    mov rsi, tStr2
    mov rdx, 34
    syscall

    mov rax, 60
    mov rdi, 0
    syscall

    mov rax,1
    mov rdi,[str1]
    mov rsi,buffer
    movsx rdx,byte[str1]
    syscall

    mov rax,3
    mov rdi,[str1]
    syscall
```



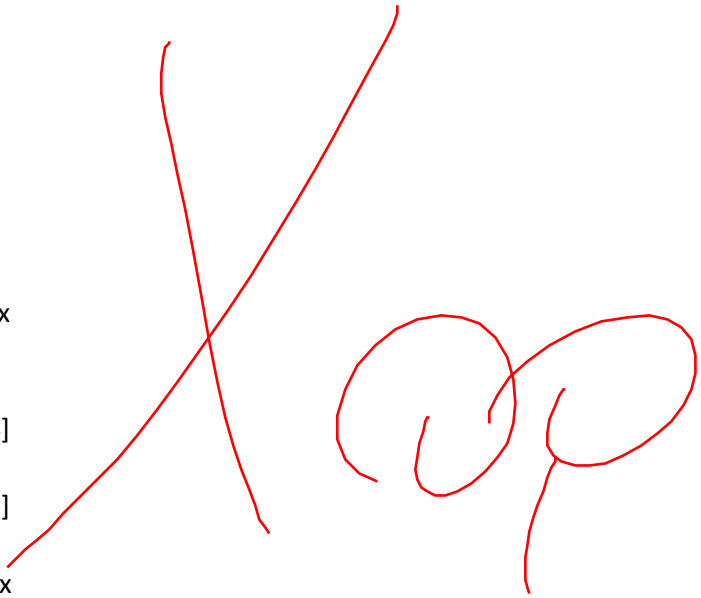
4) De um exemplo de como seria uma chamada de função com 16 bits. Crie uma função Soma para três números.

```
section .data
.LC0:
	.string "Num1: ", 10
.LC1:
	.string "Num2: ", 10
.LC2:
	.string"Num3: ", 10

main:
	push	rbp
	mov	rbp, rsp
	sub	rsp, 112
	mov	edx, 7
	mov	esi, OFFSET FLAT:.LC0
	mov	edi, 1
	call	write
	lea	rax, [rbp-55]
	mov	edx, 15
	mov	rsi, rax
	mov	edi, 0
	call	read
	mov	DWORD PTR [rbp-28], eax
	mov	eax, DWORD PTR [rbp-28]
	sub	eax, 1
	mov	DWORD PTR [rbp-32], eax
	mov	edx, 7
	mov	esi, OFFSET FLAT:.LC1
	mov	edi, 1
	call	write
	lea	rax, [rbp-70]
	mov	edx, 15
	mov	rsi, rax
	mov	edi, 0
	call	read
	mov	DWORD PTR [rbp-28], eax
	mov	eax, DWORD PTR [rbp-28]
	sub	eax, 1
	mov	DWORD PTR [rbp-36], eax
	mov	DWORD PTR [rbp-8], 0
	mov	DWORD PTR [rbp-12], 0
	mov	DWORD PTR [rbp-4], 0
	jmp	.L2

.L3:
```

```
        mov     edx, DWORD PTR [rbp-8]
        mov     eax, edx
        sal     eax, 2
        add     eax, edx
        add     eax, eax
        mov     edx, eax
        mov     eax, DWORD PTR [rbp-4]
        cdqe
        movzx   eax, BYTE PTR [rbp-55+rax]
        movsx   eax, al
        sub     eax, 48
        add     eax, edx
        mov     DWORD PTR [rbp-8], eax
        add     DWORD PTR [rbp-4], 1
.L2:
        mov     eax, DWORD PTR [rbp-4]
        cmp     eax, DWORD PTR [rbp-32]
        jl      .L3
        mov     DWORD PTR [rbp-4], 0
        jmp     .L4
.L5:
        mov     edx, DWORD PTR [rbp-12]
        mov     eax, edx
        sal     eax, 2
        add     eax, edx
        add     eax, eax
        mov     edx, eax
        mov     eax, DWORD PTR [rbp-4]
        cdqe
        movzx   eax, BYTE PTR [rbp-70+rax]
        movsx   eax, al
        sub     eax, 48
        add     eax, edx
        mov     DWORD PTR [rbp-12], eax
        add     DWORD PTR [rbp-4], 1
.L4:
        mov     eax, DWORD PTR [rbp-4]
        cmp     eax, DWORD PTR [rbp-36]
        jl      .L5
        mov     edx, DWORD PTR [rbp-8]
        mov     eax, DWORD PTR [rbp-12]
        add     eax, edx
        mov     DWORD PTR [rbp-40], eax
        mov     BYTE PTR [rbp-85], 0
        mov     DWORD PTR [rbp-16], 0
        mov     edx, DWORD PTR [rbp-40]
        movsx   rax, edx
        imul    rax, rax, 1717986919
```



```
        shr     rax, 32
        sar     eax, 2
        mov     esi, edx
        sar     esi, 31
        sub     eax, esi
        mov     ecx, eax
        mov     eax, ecx
        sal     eax, 2
        add     eax, ecx
        add     eax, eax
        sub     edx, eax
        mov     DWORD PTR [rbp-24], edx
        mov     eax, DWORD PTR [rbp-40]
        movsx   rdx, eax
        imul    rdx, rdx, 1717986919
        shr     rdx, 32
        sar     edx, 2
        sar     eax, 31
        mov     ecx, eax
        mov     eax, edx
        sub     eax, ecx
        mov     DWORD PTR [rbp-20], eax
        jmp     .L6
.L7:
        mov     eax, DWORD PTR [rbp-24]
        lea     ecx, [rax+48]
        mov     eax, DWORD PTR [rbp-16]
        lea     edx, [rax+1]
        mov     DWORD PTR [rbp-16], edx
        mov     edx, ecx
        cdqe
        mov     BYTE PTR [rbp-85+rax], dl
        mov     edx, DWORD PTR [rbp-20]
        movsx   rax, edx
        imul    rax, rax, 1717986919
        shr     rax, 32
        sar     eax, 2
        mov     esi, edx
        sar     esi, 31
        sub     eax, esi
        mov     ecx, eax
        mov     eax, ecx
        sal     eax, 2
        add     eax, ecx
        add     eax, eax
        sub     edx, eax
        mov     DWORD PTR [rbp-24], edx
        mov     eax, DWORD PTR [rbp-20]
```



```
    movsx   rdx, eax
    imul    rdx, rdx, 1717986919
    shr     rdx, 32
    sar     edx, 2
    sar     eax, 31
    mov     ecx, eax
    mov     eax, edx
    sub     eax, ecx
    mov     DWORD PTR [rbp-20], eax
.L6:
    cmp     DWORD PTR [rbp-20], 0
    jne     .L7
    mov     eax, DWORD PTR [rbp-24]
    add     eax, 48
    mov     edx, eax
    mov     eax, DWORD PTR [rbp-16]
    cdqe
    mov     BYTE PTR [rbp-85+rax], dl
    mov     BYTE PTR [rbp-101], 0
    mov     eax, DWORD PTR [rbp-16]
    mov     DWORD PTR [rbp-4], eax
    jmp     .L8
.L9:
    mov     eax, DWORD PTR [rbp-4]
    cdqe
    movzx   eax, BYTE PTR [rbp-85+rax]
    mov     BYTE PTR [rbp-102], al
    lea     rax, [rbp-102]
    mov     edx, 1
    mov     rsi, rax
    mov     edi, 1
    call    write
    sub     DWORD PTR [rbp-4], 1
.L8:
    cmp     DWORD PTR [rbp-4], 0
    jns     .L9
    mov     eax, 0
    leave
    ret
```