**Código fatorial não recursivo:**

.data # Assembly não é sensitivo a a tab

.align 0

str1: .asciz "Entre com um número: " # Com \0 no final

str2: .asciz "Resultado: "

# --> label (rótulo)

.text

.align 2 # Definindo todas as instruções em múltiplos de 2

.globl main

main:

addi a7, zero, 4

la a0, str1

ecall

addi a7, zero, 5 # sys call de scanf()

# o li é um addi. O la é convertido em 2 verdadeiras

ecall

# o valor digitado tá no a0

add s0, a0, zero # --> n

addi s1, zero, 1 # ---> fat

# fazendo o for:

loop:

beq zero, s0, fim

mul s1, s1, s0

addi s0, s0, -1

j loop

fim:

addi a7, zero, 4

la a0, str2

ecall

addi a7, zero, 1

addi a0, s1, 0

ecall

**EXERCÍCIOS LISTA 2**

**CÓDIGO EXERCÍCIO 1:**

.data

array: .word 7, 5, 2, 1, 1, 3, 4

.text

.align 2

.globl main

main:

li t0, 0 # i

li t1, 6 # j

li t6, 1 # i+1

li s11, 6 # max -1

li s12, 7 # max

la s2, array # endereço base do array

li s3, 4 # armazena o valor de 4 bytes para ser usado nas operações

loop\_j:

# PEGANDO OS VALORES DA MATRIZ!

addi t3, t1, -1 # t3 é o índice j-1

mul t3, t3, s3 # byte offset

add t4, s2, t3 # t4 agora armazena o valor do índice j-1 ---> endereço base mais índice

sw s0, 0(t4) # s0 é array[j-1]

addi t4, t1, 0 # t4 (reciclar) é o índice j

mul t4, t4, s3

add t5 , s2, t4

sw s1, 0(t5) # s1 é o array[j]

ble s0, s1, fim\_loop\_interno

# FAZENDO O SWAP!

addi s4, zero, s0 # s4 vai ser o registrador aux que vai receber j-1

addi s0, zero, s1# j-1 recebe j

addi s1, zero, s4 # j recebe aux

# PROCEDIMENTO DE SAÍDA DO FOR DO J

j fim\_loop\_interno

fim\_loop\_interno:

# PROCEDIMENTO DO INICIO FOR DO J

addi t1, t1, -1 # j = j-1

beq t1, t6, loop\_i # j == i+1 ?

j loop\_j

loop\_i:

addi t0, t0, 1

addi t6, t6, 1

beq t0, s11, impressao

j loop\_j

impressao:

addi t0, zero, -4

beq t0, zero, fim

add s10, s2, t0

addi a7, zero, 1

lw a0, 0(s10)

ecall

j impressao

fim:

addi a7, zero, 10

ecall

loop:

# Verifica se o índice é igual ao número de elementos.

beq t2, t1, end\_loop # Se t2 == t1, termina o loop.

# Carrega o elemento atual do array.

lw a1, 0(a0) # Carrega o elemento em a1.

# \*\*Aqui você chamaria a syscall ou a função para imprimir a1.\*\*

# Como exemplo, vamos apenas incrementar o ponteiro do array.

# Incrementa o ponteiro do array para o próximo inteiro (4 bytes à frente).

addi a0, a0, 4

# Incrementa o índice.

addi t2, t2, 1

# Retorna para o início do loop.

j loop

end\_loop:

# Termina o programa (assumindo syscall para sair).

# O código exato aqui depende do seu ambiente de execução.

# li a7, 10 # Para Newlib e algumas simulações Linux: syscall de saída.

# ecall