Objetivo do código: criar uma sub-rotina que ordene em ordem crescente os elementos de um vetor de tamanho n

Tipo de Sort usado: Bubble Sort

Alunos: Joao Vitor Gagliano de Oliveira, Regi Silva, Pedro Elias

-Código foi comentado seção por seção e linha a linha

-Versão executável também está no GitHub

# **Código**:

;Declaramos funcoes externas

extern printf

extern scanf

;Definimos as constantes

section .data

inputPrompt: db "Enter an array to sort:", 10, 0

inputFormat: db "%s", 0

outputFormat: db "Sorted array: %s", 10, 0

;Alocar espaço para variáveis não inicializadas.

section .bss

data: resb 128

size: resq 1

global main

section .text

main:

mov rbp, rsp

push rbp

;Escrevemos a prompt de input para o usuario

mov rdi, inputPrompt

xor rsi, rsi

xor rax, rax

call printf

;Pegamos o input do usuario

mov rdi, inputFormat

mov rsi, data

xor rax, rax

call scanf

;Chamamos a funcao bubbleSort, que foi implementada mais abaixo no codigo

mov rdi, data

call .bubbleSort

;Printamos o array ordenado resultante

mov rdi, outputFormat

mov rsi, rax

xor rax, rax

call printf

;Saimos do programa

pop rbp

xor rax, rax

ret

;Funcao bubble sort

;Input esta no registrador RDI, que contem o nosso aray

;Output sera alocado no registrador RAX, que vai conter o array ordenado

.bubbleSort:

;Inicializamos o bubble sort

mov rsi, rdi

xor rax, rax

xor rdx, rdx

;Vamos pegar o tamanho do array para saber quantas vezes vamos executar o loop

.length:

lodsb

cmp al, 0

inc rax

jz .length

dec rax

mov qword [size], rax

;preparacao para o primeiro loop zerando os registradores

xor rdx, rdx

xor rax, rax

.loop1:

;Se i >= tamanho, paramos o loop

cmp rdx, [size]

jge .endLoop1

inc rdx

;Setup para o segundo loop

mov rsi, rdi

;Comparamos 2 numeros vizinhos e fazemos outro loop

.loop2:

lodsb

cmp al, 0

jz .endLoop2

mov bl, al

mov byte bh, [rsi]

cmp bh, 0

jz .endLoop2

cmp bl, bh

jg .swap

jmp .loop2

;Trocar 2 numeros vizinhos

.swap:

mov cl, bl

mov bl, bh

mov bh, cl

mov byte [rsi-1], bl

mov byte [rsi], bh

jmp .loop2

;Terminar o loop 2

.endLoop2:

jmp .loop1

;Terminar o loop 1

.endLoop1:

;Saimos da funcao

mov rax, data

ret