Assignment 6

Write 64-bit ALP to accept the numbers from user and perform addition of 2 numbers and display the result on screen.  
  
section .data

msg1 db "Enter a digit", 0xA

len1 equ $ - msg1

msg2 db "Please enter a second digit", 0xA

len2 equ $ - msg2

msg3 db "The sum is: ", 0xA

len3 equ $ - msg3

section .bss

num1 resb 2 ; Reserve 2 bytes for the first number

num2 resb 2 ; Reserve 2 bytes for the second number

res resb 1 ; Reserve 1 byte for the result

section .text

global \_start

\_start:

; Print msg1

mov rax, 1 ; syscall: write

mov rdi, 1 ; file descriptor: stdout

mov rsi, msg1 ; pointer to message

mov rdx, len1 ; length of message

syscall

; Read num1

mov rax, 0 ; syscall: read

mov rdi, 0 ; file descriptor: stdin

mov rsi, num1 ; pointer to buffer

mov rdx, 2 ; number of bytes to read

syscall

; Remove the newline character if present (0xA)

mov al, [num1] ; Load first byte of num1

cmp al, 0xA ; Check if it's newline

je .remove\_newline1

jmp .continue1

.remove\_newline1:

mov byte [num1], 0 ; Replace newline with null byte

.continue1:

; Print msg2

mov rax, 1 ; syscall: write

mov rdi, 1 ; file descriptor: stdout

mov rsi, msg2 ; pointer to message

mov rdx, len2 ; length of message

syscall

; Read num2

mov rax, 0 ; syscall: read

mov rdi, 0 ; file descriptor: stdin

mov rsi, num2 ; pointer to buffer

mov rdx, 2 ; number of bytes to read

syscall

; Remove the newline character if present (0xA)

mov al, [num2] ; Load first byte of num2

cmp al, 0xA ; Check if it's newline

je .remove\_newline2

jmp .continue2

.remove\_newline2:

mov byte [num2], 0 ; Replace newline with null byte

.continue2:

; Print msg3

mov rax, 1 ; syscall: write

mov rdi, 1 ; file descriptor: stdout

mov rsi, msg3 ; pointer to message

mov rdx, len3 ; length of message

syscall

; Convert num1 to decimal

movzx rax, byte [num1] ; Load the first byte of num1 into rax (zero-extend)

sub rax, '0' ; Convert ASCII to decimal

; Convert num2 to decimal

movzx rbx, byte [num2] ; Load the first byte of num2 into rbx (zero-extend)

sub rbx, '0' ; Convert ASCII to decimal

; Add the two numbers

add rax, rbx ; Add rax (num1) and rbx (num2)

add rax, '0' ; Convert the sum back to ASCII

; Store the result in res

mov [res], al ; Store the ASCII result in res

; Print the result

mov rax, 1 ; syscall: write

mov rdi, 1 ; file descriptor: stdout

mov rsi, res ; pointer to result

mov rdx, 1 ; number of bytes to write

syscall

; Exit the program

mov rax, 60 ; syscall: exit

xor rdi, rdi ; exit code: 0

syscall