Assignment 8

Write 64-bit ALP to perform multiplication of two 8-bit hexadecimal number with successive addition  
  
section .data

num1 db 0x03 ; First number (example: 0x03)

num2 db 0x04 ; Second number (example: 0x04)

result dq 0 ; Store the result (64-bit for safety)

msg db "Result: ", 0 ; Message to display

section .bss

result\_str resb 20 ; Buffer to hold the result string

section .text

global \_start ; Entry point for the program

\_start:

; Load numbers into registers

movzx rax, byte [num1] ; Load num1 into RAX (zero-extended)

movzx rbx, byte [num2] ; Load num2 into RBX (zero-extended)

; Initialize registers

xor rcx, rcx ; RCX will hold the sum (initialize to 0)

xor rdx, rdx ; RDX will be used as a counter

; Perform successive addition

mov rdx, rbx ; Load num2 into RDX as the counter

multiply\_loop:

test rdx, rdx ; Check if counter (RDX) is 0

jz done ; If RDX == 0, exit the loop

add rcx, rax ; Add num1 (RAX) to RCX

dec rdx ; Decrement the counter (RDX)

jmp multiply\_loop ; Repeat the loop

done:

; Store the result

mov [result], rcx ; Save the result in memory

; Convert the result to a string

mov rax, rcx ; Load result into RAX for conversion

mov rdi, result\_str ; Buffer to store string

call int\_to\_string ; Convert integer to string

; Print the message

mov rax, 1 ; Syscall for write

mov rdi, 1 ; File descriptor (stdout)

mov rsi, msg ; Address of the message

mov rdx, 8 ; Length of the message

syscall

; Print the result

mov rax, 1 ; Syscall for write

mov rdi, 1 ; File descriptor (stdout)

mov rsi, result\_str ; Address of the result string

mov rdx, 20 ; Maximum length of the string

syscall

; Exit the program

mov rax, 60 ; Syscall number for exit

xor rdi, rdi ; Return code 0

syscall

; Function: int\_to\_string

; Converts the number in RAX to a null-terminated string in RDI

int\_to\_string:

xor rcx, rcx ; Clear RCX (to count digits)

mov rbx, 10 ; Divisor for base-10

convert\_loop:

xor rdx, rdx ; Clear RDX (to store remainder)

div rbx ; Divide RAX by 10, quotient in RAX, remainder in RDX

add dl, '0' ; Convert remainder to ASCII

push rdx ; Store digit on stack

inc rcx ; Increment digit count

test rax, rax ; Check if quotient is 0

jnz convert\_loop ; If not, continue

mov rsi, rdi ; Destination pointer for string

write\_digits:

pop rdx ; Get digit from stack

mov byte [rsi], dl ; Write digit to memory

inc rsi ; Move to next character

loop write\_digits ; Repeat for all digits

mov byte [rsi], 0 ; Null-terminate the string

ret