**Name: Muhammad Uzair Attiq**

**Enroll: 01-136242-029**

**Tasks:**

1. Write a program that takes two numbers from user and decide whether those

numbers are equal or not.

**Answer**

**Code:**

section .data

prompt1 db "Enter first number: ", 0

prompt1\_len equ $ - prompt1 - 1

prompt2 db "Enter second number: ", 0

prompt2\_len equ $ - prompt2 - 1

equal\_msg db "The numbers are EQUAL", 10, 0

equal\_len equ $ - equal\_msg - 1

not\_equal\_msg db "The numbers are NOT EQUAL", 10, 0

not\_equal\_len equ $ - not\_equal\_msg - 1

newline db 10, 0

section .bss

input\_buffer1 resb 20

input\_buffer2 resb 20

num1 resq 1

num2 resq 1

section .text

global \_main

\_main:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel prompt1]

mov rdx, prompt1\_len

syscall

mov rax, 0x2000003

mov rdi, 0

lea rsi, [rel input\_buffer1]

mov rdx, 20

syscall

lea rsi, [rel input\_buffer1]

call string\_to\_int

mov [rel num1], rax

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel prompt2]

mov rdx, prompt2\_len

syscall

mov rax, 0x2000003

mov rdi, 0

lea rsi, [rel input\_buffer2]

mov rdx, 20

syscall

lea rsi, [rel input\_buffer2]

call string\_to\_int

mov [rel num2], rax

mov rax, [rel num1]

mov rbx, [rel num2]

cmp rax, rbx

je numbers\_equal

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel not\_equal\_msg]

mov rdx, not\_equal\_len

syscall

jmp exit\_program

numbers\_equal:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel equal\_msg]

mov rdx, equal\_len

syscall

exit\_program:

mov rdi, 0

mov rax, 0x2000001

syscall

string\_to\_int:

push rbx

push rcx

push rdx

xor rax, rax

xor rbx, rbx

convert\_loop:

movzx rbx, byte [rsi]

cmp bl, 10

je convert\_done

cmp bl, 0

je convert\_done

cmp bl, 13

je convert\_done

cmp bl, '0'

jb convert\_done

cmp bl, '9'

ja convert\_done

sub bl, '0'

imul rax, rax, 10

add rax, rbx

inc rsi

jmp convert\_loop

convert\_done:

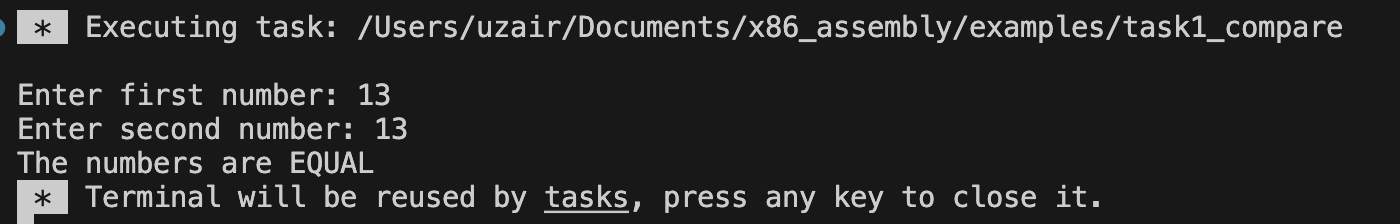
pop rdx

pop rcx

pop rbx

ret

**Output:**



1. Write a program that copies unsigned numbers from one array to another, but it copies numbers that are Greater than any number entered by the user.

**Answer**

**Code:**

section .data

source\_array dq 5, 12, 3, 18, 7, 25, 2, 15, 9, 30

array\_size equ 10

intro\_msg db "Source array: [5, 12, 3, 18, 7, 25, 2, 15, 9, 30]", 10, 0

intro\_len equ $ - intro\_msg - 1

prompt\_msg db "Enter threshold (copy numbers > threshold): ", 0

prompt\_len equ $ - prompt\_msg - 1

result\_msg db "Numbers copied to destination array: ", 0

result\_len equ $ - result\_msg - 1

no\_copy\_msg db "No numbers found greater than threshold", 0

no\_copy\_len equ $ - no\_copy\_msg - 1

space db " ", 0

newline db 10, 0

section .bss

input\_buffer resb 20

dest\_array resq 10

threshold resq 1

copy\_count resq 1

print\_buffer resb 20

section .text

global \_main

\_main:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel intro\_msg]

mov rdx, intro\_len

syscall

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel prompt\_msg]

mov rdx, prompt\_len

syscall

mov rax, 0x2000003

mov rdi, 0

lea rsi, [rel input\_buffer]

mov rdx, 20

syscall

lea rsi, [rel input\_buffer]

call string\_to\_int

mov [rel threshold], rax

mov rcx, 0

mov rdx, 0

mov qword [rel copy\_count], 0

copy\_loop:

cmp rcx, array\_size

jge copy\_done

lea rbx, [rel source\_array]

mov rax, [rbx + rcx\*8]

mov rbx, [rel threshold]

cmp rax, rbx

jle skip\_element

lea rbx, [rel dest\_array]

mov [rbx + rdx\*8], rax

inc rdx

inc qword [rel copy\_count]

skip\_element:

inc rcx

jmp copy\_loop

copy\_done:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel result\_msg]

mov rdx, result\_len

syscall

mov rcx, 0

cmp qword [rel copy\_count], 0

je no\_numbers\_copied

print\_loop:

cmp rcx, [rel copy\_count]

jge print\_done

lea rbx, [rel dest\_array]

mov rax, [rbx + rcx\*8]

call print\_number

inc rcx

cmp rcx, [rel copy\_count]

jge print\_done

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel space]

mov rdx, 1

syscall

jmp print\_loop

no\_numbers\_copied:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel no\_copy\_msg]

mov rdx, no\_copy\_len

syscall

jmp final\_newline

print\_done:

final\_newline:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel newline]

mov rdx, 1

syscall

mov rdi, [rel copy\_count]

mov rax, 0x2000001

syscall

string\_to\_int:

push rbx

push rcx

push rdx

xor rax, rax

xor rbx, rbx

convert\_loop:

movzx rbx, byte [rsi]

cmp bl, 10

je convert\_done

cmp bl, 0

je convert\_done

cmp bl, 13

je convert\_done

cmp bl, '0'

jb convert\_done

cmp bl, '9'

ja convert\_done

sub bl, '0'

imul rax, rax, 10

add rax, rbx

inc rsi

jmp convert\_loop

convert\_done:

pop rdx

pop rcx

pop rbx

ret

print\_number:

push rbp

mov rbp, rsp

push rbx

push rcx

push rdx

push rsi

test rax, rax

jz print\_zero

mov rbx, 10

mov rcx, 0

lea rsi, [rel print\_buffer]

add rsi, 19

mov byte [rsi], 0

convert\_number\_loop:

xor rdx, rdx

div rbx

add dl, '0'

dec rsi

mov [rsi], dl

inc rcx

test rax, rax

jnz convert\_number\_loop

mov rax, 0x2000004

mov rdi, 1

mov rdx, rcx

syscall

pop rsi

pop rdx

pop rcx

pop rbx

pop rbp

ret

print\_zero:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel print\_buffer]

mov byte [rsi], '0'

mov rdx, 1

syscall

pop rsi

pop rdx

pop rcx

pop rbx

pop rbp

ret

**Output:**

A black background with white text

AI-generated content may be incorrect.

1. Write a program that copies signed numbers from one array to another, but it copies numbers that are less than or equal to the number entered by the user.

**Answer**

**Code:**

section .data

source\_array dq -5, 12, -3, 18, 7, -25, 2, 15, -9, 8

array\_size equ 10

intro\_msg db "Source array: [-5, 12, -3, 18, 7, -25, 2, 15, -9, 8]", 10, 0

intro\_len equ $ - intro\_msg - 1

prompt\_msg db "Enter threshold (copy numbers <= threshold): ", 0

prompt\_len equ $ - prompt\_msg - 1

result\_msg db "Numbers copied to destination array: ", 0

result\_len equ $ - result\_msg - 1

no\_copy\_msg db "No numbers found <= threshold", 0

no\_copy\_len equ $ - no\_copy\_msg - 1

space db " ", 0

newline db 10, 0

minus\_sign db "-", 0

section .bss

input\_buffer resb 20

dest\_array resq 10

threshold resq 1

copy\_count resq 1

print\_buffer resb 20

section .text

global \_main

\_main:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel intro\_msg]

mov rdx, intro\_len

syscall

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel prompt\_msg]

mov rdx, prompt\_len

syscall

mov rax, 0x2000003

mov rdi, 0

lea rsi, [rel input\_buffer]

mov rdx, 20

syscall

lea rsi, [rel input\_buffer]

call string\_to\_int

mov [rel threshold], rax

mov rcx, 0

mov rdx, 0

mov qword [rel copy\_count], 0

copy\_loop:

cmp rcx, array\_size

jge copy\_done

lea rbx, [rel source\_array]

mov rax, [rbx + rcx\*8]

mov rbx, [rel threshold]

cmp rax, rbx

jg skip\_element

lea rbx, [rel dest\_array]

mov [rbx + rdx\*8], rax

inc rdx

inc qword [rel copy\_count]

skip\_element:

inc rcx

jmp copy\_loop

copy\_done:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel result\_msg]

mov rdx, result\_len

syscall

mov rcx, 0

cmp qword [rel copy\_count], 0

je no\_numbers\_copied

print\_loop:

cmp rcx, [rel copy\_count]

jge print\_done

lea rbx, [rel dest\_array]

mov rax, [rbx + rcx\*8]

call print\_signed\_number

inc rcx

cmp rcx, [rel copy\_count]

jge print\_done

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel space]

mov rdx, 1

syscall

jmp print\_loop

no\_numbers\_copied:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel no\_copy\_msg]

mov rdx, no\_copy\_len

syscall

jmp final\_newline

print\_done:

final\_newline:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel newline]

mov rdx, 1

syscall

mov rdi, [rel copy\_count]

mov rax, 0x2000001

syscall

string\_to\_int:

push rbx

push rcx

push rdx

xor rax, rax

xor rbx, rbx

mov rcx, 0

cmp byte [rsi], '-'

jne positive\_number

mov rcx, 1

inc rsi

positive\_number:

convert\_loop:

movzx rbx, byte [rsi]

cmp bl, 10

je convert\_done

cmp bl, 0

je convert\_done

cmp bl, 13

je convert\_done

cmp bl, '0'

jb convert\_done

cmp bl, '9'

ja convert\_done

sub bl, '0'

imul rax, rax, 10

add rax, rbx

inc rsi

jmp convert\_loop

convert\_done:

test rcx, rcx

jz positive\_done

neg rax

positive\_done:

pop rdx

pop rcx

pop rbx

ret

print\_signed\_number:

push rbp

mov rbp, rsp

push rbx

push rcx

push rdx

push rsi

test rax, rax

jns print\_positive\_number

push rax

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel minus\_sign]

mov rdx, 1

syscall

pop rax

neg rax

print\_positive\_number:

test rax, rax

jz print\_zero\_signed

mov rbx, 10

mov rcx, 0

lea rsi, [rel print\_buffer]

add rsi, 19

mov byte [rsi], 0

convert\_signed\_loop:

xor rdx, rdx

div rbx

add dl, '0'

dec rsi

mov [rsi], dl

inc rcx

test rax, rax

jnz convert\_signed\_loop

mov rax, 0x2000004

mov rdi, 1

mov rdx, rcx

syscall

pop rsi

pop rdx

pop rcx

pop rbx

pop rbp

ret

print\_zero\_signed:

mov rax, 0x2000004

mov rdi, 1

lea rsi, [rel print\_buffer]

mov byte [rsi], '0'

mov rdx, 1

syscall

pop rsi

pop rdx

pop rcx

pop rbx

pop rbp

ret

**Output:  
A black background with white text

AI-generated content may be incorrect.**