section .data

    msg1 db "Enter first number: ", 0

    msg2 db "Enter second number: ", 0

    msg\_add db "Addition: ", 0

    msg\_sub db "Subtraction: ", 0

    msg\_and db "Bitwise AND: ", 0

    msg\_or db "Bitwise OR: ", 0

    msg\_xor db "Bitwise XOR: ", 0

    newline db 10, 0

section .bss

    num1 resb 10      ; Buffer for first input

    num2 resb 10      ; Buffer for second input

    result resb 20    ; Buffer for result (increased size)

section .text

    global main

    extern printf, scanf

main:

    ; Prompt for first number

    mov rdi, msg1

    call print\_string

    ; Read first number

    mov rdi, num1

    call read\_input

    call str\_to\_int

    mov r12, rax    ; Store first number in R12

    ; Prompt for second number

    mov rdi, msg2

    call print\_string

    ; Read second number

    mov rdi, num2

    call read\_input

    call str\_to\_int

    mov r13, rax    ; Store second number in R13

    ; Addition

    mov rdi, msg\_add

    call print\_string

    mov rax, r12

    add rax, r13

    call print\_number

    ; Subtraction

    mov rdi, msg\_sub

    call print\_string

    mov rax, r12

    sub rax, r13

    call print\_number

    ; Bitwise AND

    mov rdi, msg\_and

    call print\_string

    mov rax, r12

    and rax, r13

    call print\_number

    ; Bitwise OR

    mov rdi, msg\_or

    call print\_string

    mov rax, r12

    or rax, r13

    call print\_number

    ; Bitwise XOR

    mov rdi, msg\_xor

    call print\_string

    mov rax, r12

    xor rax, r13

    call print\_number

    ; Exit program

    xor rdi, rdi

    mov rax, 60

    syscall

; Function to print a string

print\_string:

    push rdi

    mov rsi, rdi    ; string to print

    mov rdi, 1      ; file descriptor (stdout)

    mov rdx, 20     ; max length

    mov rax, 1      ; syscall: write

    syscall

    pop rdi

    ret

; Function to read input

read\_input:

    mov rsi, rdi    ; destination buffer

    mov rdi, 0      ; file descriptor (stdin)

    mov rdx, 10     ; max input size

    mov rax, 0      ; syscall: read

    syscall

    ret

; Convert string to integer

str\_to\_int:

    xor rax, rax    ; Clear RAX to store the result

    mov rcx, 10     ; Base 10

convert\_loop:

    movzx rbx, byte [rsi]

    cmp rbx, 10     ; Check for newline

    je done\_convert

    cmp rbx, '0'    ; Validate input is a digit

    jl done\_convert

    cmp rbx, '9'

    jg done\_convert

    sub rbx, '0'    ; Convert ASCII to number

    imul rax, rcx   ; Multiply RAX by 10

    add rax, rbx    ; Add digit

    inc rsi

    jmp convert\_loop

done\_convert:

    ret

; Convert number to string and print

print\_number:

    mov rsi, result + 19

    mov byte [rsi], 10  ; Newline at end

    dec rsi

    mov rcx, 10

num\_to\_str:

    xor rdx, rdx

    div rcx

    add dl, '0'

    mov [rsi], dl

    dec rsi

    test rax, rax

    jnz num\_to\_str

    inc rsi

    ; Print number

    mov rdx, 20     ; Buffer size

    sub rdx, rsi

    add rdx, result + 19

    mov rdi, 1      ; file descriptor (stdout)

    mov rax, 1      ; syscall: write

    syscall

    ret