```
section .data
   msg_arithmetic db "------Arithmetic operation------", 10
   len_arithmetic equ $-msg_arithmetic
   msg_num1 db 'Enter number 1: ', 10
   len_num1 equ $-msg_num1
   msg_num2 db 'Enter number 2: ', 10
   len_num2 equ $-msg_num2
   msg_sum db "Addition: ", 10
   len_sum equ $-msg_sum
   msg_sub db "Subtraction: ",10
   len_sub equ $-msg_sub
   msg_mul db "Multiplication: ", 10
   len_mul equ $-msg_mul
   msg_div db "Division: ", 10
   len_div equ $-msg_div
   msg_logical db "------Logical operation------", 10
   len_logical equ $-msg_logical
   msg_or db "OR: ", 10
   len_or equ $-msg_or
   msg_and db "AND: ", 10
   len_and equ $-msg_and
   msg_not db "NOT: ", 10
   len_not equ $-msg_not

section .bss
   num1 resb 2
   num2 resb 2
   addition resb 2
   subtraction resb 2
   multiplication resb 2
   division resb 2
   logical_or resb 2
   logical_and resb 2
   logical_not resb 2

section .text
global _start
_start:
   ; display first msg
   mov eax, 4
   mov ebx, 1
   mov ecx, msg_num1
   mov edx, len_num1
   int 80h
```

```
; accepting first number
mov eax, 3
mov ebx, 0
mov ecx, num1
mov edx, 2
int 80h

; display second msg
mov eax, 4
mov ebx, 1
mov ecx, msg_num2
mov edx, len_num2
int 80h

; accepting second no.
mov eax, 3
mov ebx, 0
mov ecx, num2
mov edx, 2
int 80h
;-----------------------------------------------------
; -------------ARITHMETIC OPERATION---------------------
mov eax, 4
mov ebx, 1
mov ecx, msg_arithmetic
mov edx, len_arithmetic
int 80h

; performing operation
mov eax, [num1]
sub eax, '0'
mov ebx, [num2]
sub ebx, '0'
add eax, ebx
add eax,'0'

; move the results to a variable addition
mov [addition], eax
; ------------------------addition-------------------
; display sum message
mov eax, 4
mov ebx, 1
mov ecx, msg_sum
```

```
mov edx, len_sum
int 80h

; displaying sum
mov eax, 4
mov ebx, 1
mov ecx, addition
mov edx, 2
int 80h
; ----------------------subtraction----------------
; display subtraction msg
mov eax, 4
mov ebx, 1
mov ecx, msg_sub
mov edx, len_sub
int 80h

; performing subtraction
mov eax, [num1]
sub eax, '0'
mov ebx, [num2]
sub ebx, '0'
sub eax, ebx
add eax,'0'

; move the results to a variable subtraction
mov [subtraction], eax

; displaying subtraction results
mov eax, 4
mov ebx, 1
mov ecx, subtraction
mov edx, 2
int 80h
; ----------------------multiplication-----------------------
; display subtraction msg
mov eax, 4
mov ebx, 1
mov ecx, msg_mul
mov edx, len_mul
int 80h

; performing subtraction
mov eax, [num1]
```

```
sub eax, '0'
mov ebx, [num2]
sub ebx, '0'
mul ebx
add eax,'0'

; move the results to a variable subtraction
mov [multiplication], eax

; displaying subtraction results
mov eax, 4
mov ebx, 1
mov ecx, multiplication
mov edx, 2
int 80h
; ---------------division--------------------------
 ; display subtraction msg
mov eax, 4
mov ebx, 1
mov ecx, msg_div
mov edx, len_div
int 80h

; performing subtraction
mov al, [num1]
sub al, '0'
mov bl, [num2]
sub bl, '0'
div bl
add al,'0'

; move the results to a variable subtraction
mov [division], al

; displaying subtraction results
mov eax, 4
mov ebx, 1
mov ecx, division
mov edx, 2
int 80h
; ------------------------------------------------
; -------------------LOGICAL OPERATIONS-----------------------
mov eax, 4
mov ebx, 1
```

```
    mov ecx, msg_or
    mov edx, len_or
    int 80h

    ; performing logical or
    mov eax, [num1]
    sub eax, '0'
    mov ebx, [num2]
    sub eax, '0'
    or eax, ebx
    add eax, '0'
    ; moving the result into new variable logical_or
    mov [logical_or], eax
    ; display the result
    mov eax, 4
    mov ebx, 1
    mov ecx, logical_or
    mov edx, 2
    int 80h

    ; -------------------------------------------------------
    ; end
    mov eax, 1
    mov ebx, 0
    int 80h
```

Menu driven program:

```
section .data
    menu db "Menu:", 0
    len_menu equ $-menu
    option1 db "1. Add two numbers",10, 0
    len_1 equ $-option1
    option2 db "2. Subtract two numbers", 10, 0
    len_2 equ $-option2
    option3 db "3. Multiply two numbers", 10, 0
    len_3 equ $-option3
    option4 db "4. Divide two numbers", 10, 0
    len_4 equ $-option4
    option5 db "5. Logical AND of two numbers", 10, 0
    len_5 equ $-option5
    option6 db "6. Logical OR of two numbers", 10, 0
```

```
    len_6 equ $-option6
    option7 db "7. Exit", 10, 0
    len_7 equ $-option7
    msg1 db "Enter first number: ", 10, 0
    len_msg1 equ $-msg1
    msg2 db "Enter second number: ", 10, 0
    len_msg2 equ $-msg2
    msg3 db "Result: ", 10, 0
    len_msg3 equ $-msg3

section .bss
    num1 resb 2
    num2 resb 2
    sum_result resb 2

section .text
global _start

_start:
    ; Print the menu options
    mov eax, 4
    mov ebx, 1
    mov ecx, menu
    mov edx, len_menu
    int 80h

    mov eax, 4
    mov ebx, 1
    mov ecx, option1
    mov edx, len_1
    int 80h

    mov eax, 4
    mov ebx, 1
    mov ecx, option2
    mov edx, len_2
    int 80h

    mov eax, 4
    mov ebx, 1
    mov ecx, option3
    mov edx, len_3
    int 80h
```

```
mov eax, 4
mov ebx, 1
mov ecx, option4
mov edx, len_4
int 80h

mov eax, 4
mov ebx, 1
mov ecx, option5
mov edx, len_5
int 80h

mov eax, 4
mov ebx, 1
mov ecx, option6
mov edx, len_6
int 80h

mov eax, 4
mov ebx, 1
mov ecx, option7
mov edx, len_7
int 80h

; Read the user's choice
mov eax, 3
mov ebx, 0
mov ecx, num1
mov edx, 2
int 80h

; Check the user's choice and perform the corresponding operation
cmp byte[num1], '1'
je add_numbers
; cmp byte[num1], '2'
; je subtract_numbers
; cmp byte[num1], '3'
; je multiply_numbers
; cmp byte[num1], '4'
; je divide_numbers
; cmp byte[num1], '5'
; je and_numbers
; cmp byte[num1], '6'
; je or_numbers
```

```
    cmp byte[num1], '7'
    je exit_program

add_numbers:
    ; Read the first number
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, len_msg1
    int 80h
    ; input first number
    mov eax, 3
    mov ebx, 0
    mov ecx, num1
    mov edx, 2
    int 80h

    ; Read the second number
    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, len_msg2
    int 80h

    ; input second number
    mov eax, 3
    mov ebx, 0
    mov ecx, num2
    mov edx, 2
    int 80h
    ; perform addition
    mov eax, [num1]
    sub eax, '0'
    mov ebx, [num2]
    sub eax, '0'
    add eax, ebx
    add eax, '0'
    ; move result ot variable
    mov [sum_result], eax
    ; displaying text
    mov eax, 4
    mov ebx, 1
    mov ecx, msg3
    mov edx, len_msg3
```

```
    int 80h
    ; displaying result
    mov eax, 4
    mov ebx, 1
    mov ecx, sum_result
    mov edx, 2
    int 80h
    jmp exit_program
exit_program:
    mov eax, 1
    mov ebx, 0
    int 80h
```
