```
.stack 100h

.data
operand1 db ?
operand2 db ?
result db ?
operator db ?
msg_divide_by_zero db "Infinity", 0Dh, 0Ah, "$"
msg_invalid_operator db "Invalid operator", 0Dh, 0Ah, "$"
negative_sign db "-"
ascii db 2 DUP(?)

.code
main proc
    mov ax, @data
    mov ds, ax

    ; Read the first operand
    mov ah, 01h
    int 21h
    sub al, '0'  ; Convert ASCII to binary
    mov operand1, al

    ; Read the operator
    mov ah, 01h
    int 21h
    mov operator, al

    ; Read the second operand
```

```
    mov ah, 01h
    int 21h
    sub al, '0'  ; Convert ASCII to binary
    mov operand2, al

    ;Print Equal sign
    mov dl, '='
    mov ah, 02h
    int 21h

    ;Perform the operation based on the operator
    cmp operator, '+'
    je addition
    cmp operator, '-'
    je subtraction
    cmp operator, '*'
    je multiplication
    cmp operator, '/'
    je division
    jmp invalid_operator

addition:
    mov al, operand1
    add al, operand2
    mov result, al
    jmp print_result

subtraction:
    mov al, operand1
```

```
    sub al, operand2
    mov result, al
    jmp print_result

multiplication:
    mov al, operand1
    mul operand2
    mov result, al
    jmp print_result

division:
    cmp operand2, 0
    je divide_by_zero
    mov al, operand1
    mov bl, operand2
    mov ah, 0 ; Clear AH for DIV operation

    div bl
    mov result, al
    jmp print_result

divide_by_zero:

    mov ax, 0
    mov es, ax

    mov al, 75h
    mov bl, 4h
    mul bl
```

```
    mov bx, ax

    mov si, offset [infinity_msg]
    mov es:[bx], si
    add bx, 2
    mov ax, cs
    mov es:[bx], ax

    int 75h
    jmp quit_program

print_result:
    ; Check if result is negative
    cmp result, 0
    jns print_result_positive
    ; If negative, print negative sign
    mov dl, negative_sign
    mov ah, 02h
    int 21h
    ; Convert result to positive for ASCII conversion
    neg result

print_result_positive:
    MOV AL, result
    MOV AH, 0
    MOV BH, 0
    MOV BL, 10

    DIV BL          ; Divide AX by BL, quotient in AL (tens digit), remainder in AH (ones digit)
```

```
    ADD AL, '0'
    MOV ascii[0], AL ; Store tens digit in ASCII representation
    ADD AH, '0'     ;
    MOV ascii[1], AH ; Store ones digit in ASCII representation

    ; Terminate the message string
    MOV BYTE PTR [ascii+2], 0Dh ; Carriage return
    MOV BYTE PTR [ascii+3], 0Ah ; Line feed
    MOV BYTE PTR [ascii+4], '$' ; End of string ('$')

    ; Display ASCII
    MOV AH, 09h
    LEA DX, ascii
    INT 21h
    jmp quit_program

invalid_operator:

    mov ah, 09h
    lea dx, msg_invalid_operator
    int 21h

quit_program:
    mov ah, 4Ch
    int 21h

main endp

infinity_msg PROC
```

```
    mov ah, 09h
    lea dx, msg_divide_by_zero
    int 21h
    IRET

infinity_msg ENDP

end main
```