

Universidad Autónoma de Baja California



Facultad de Ingeniería, Arquitectura y Diseño


Ingeniería en Software

Organización de Computadoras

Taller 12 GIT HUB


Brayan Arturo Rocha Meneses

Ensenada Baja California Noviembre de 2024

 Assembly ▼



```
1 section .data
2 msg_input db "Introduce un número: ", 0 ; Mensaje para entrada
3 msg_sum db "La suma es: ", 0 ; Mensaje para el resultado
4 num1 db 0 ; Almacenará el primer número
5 num2 db 0 ; Almacenará el segundo número
6 result db 0 ; Almacenará la suma
7 buffer db 10 ; Buffer para entrada de datos
8 section .bss
9 temp resb 1 ; Variable temporal para cálculos
10 section .text
11 global _start
12 _start:
13 ; === Entrada del primer número ===
14 mov edx, len msg_input ; Longitud del mensaje
15 mov ecx, msg_input ; Dirección del mensaje
16 mov ebx, 1 ; STDOUT
17 mov eax, 4 ; Llamada al sistema write
18 int 0x80 ; Llamada al kernel
19 ; Leer número desde STDIN
20 mov eax, 3 ; Llamada al sistema read
21 mov ebx, 0 ; STDIN
22 mov ecx, buffer ; Buffer para almacenar entrada
23 mov edx, 2 ; Leer hasta 2 bytes (número + \n)
24 int 0x80 ; Llamada al kernel
```

 Assembly ▼



```
25 ; Convertir entrada de ASCII a entero
26 sub byte [buffer], '0' ; Convertir de ASCII a valor numérico
27 mov [num1], byte [buffer] ; Guardar el número en num1
28 ; === Entrada del segundo número ===
29 mov edx, len msg_input ; Longitud del mensaje
30 mov ecx, msg_input ; Dirección del mensaje
31 mov ebx, 1 ; STDOUT
32 mov eax, 4 ; Llamada al sistema write
33 int 0x80 ; Llamada al kernel
34 ; Leer segundo número desde STDIN
35 mov eax, 3 ; Llamada al sistema read
36 mov ebx, 0 ; STDIN
37 mov ecx, buffer ; Buffer para almacenar entrada
38 mov edx, 2 ; Leer hasta 2 bytes (número + \n)
39 int 0x80 ; Llamada al kernel
40 ; Convertir entrada de ASCII a entero
41 sub byte [buffer], '0' ; Convertir de ASCII a valor numérico
42 mov [num2], byte [buffer] ; Guardar el número en num2
43 ; === Sumar los dos números ===
44 mov al, [num1] ; Cargar el primer número
45 add al, [num2] ; Sumar el segundo número
46 mov [result], al ; Guardar el resultado
47 ; === Mostrar el resultado ===
48 mov edx, len msg_sum ; Longitud del mensaje
```



Assembly ▾



```
41 sub byte [buffer], '0' ; Convertir de ASCII a valor numérico
42 mov [num2], byte [buffer] ; Guardar el número en num2
43 ; === Sumar los dos números ===
44 mov al, [num1] ; Cargar el primer número
45 add al, [num2] ; Sumar el segundo número
46 mov [result], al ; Guardar el resultado
47 ; === Mostrar el resultado ===
48 mov edx, len msg_sum ; Longitud del mensaje
49 mov ecx, msg_sum ; Dirección del mensaje
50 mov ebx, 1 ; STDOUT
51 mov eax, 4 ; Llamada al sistema write
52 int 0x80 ; Llamada al kernel
53 ; Convertir resultado de entero a ASCII
54 add byte [result], '0' ; Convertir a ASCII
55 mov eax, 4 ; Llamada al sistema write
56 mov ebx, 1 ; STDOUT
57 mov ecx, result ; Dirección del resultado
58 mov edx, 1 ; Longitud del resultado
59 int 0x80 ; Llamada al kernel
60 ; === Salir del programa ===
61 mov eax, 1 ; Llamada al sistema exit
62 xor ebx, ebx ; Código de salida 0
63 int 0x80 ; Llamada al kernel
64 len equ $ - msg_input ; Macro para calcular longitud
```