UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTADA DE INGENIERÍA ARQUITECTURA Y DISEÑO

INGENIERÍA EN SOFTWARE Y TECNOLOGÍAS EMERGENTES

ORGANIZACIÓN DE COMPUTADORAS

Taller 11

ADRIAN BALDERAS ROSAS

Jonatan Crespo Ragland

932

Código 1

```
section .data
 num1 db 5 ; Define el primer número, 5, almacenado en una variable de 1 byte num2 db
11 ; Define el segundo número, 11, almacenado en una variable de 1 byte result db 0 ;
Variable para almacenar el resultado de la suma, inicialmente en 0
 message db "Resultado: ", 0 ; Mensaje a mostrar antes del resultado, seguido de un
terminador null

section .bss
 buffer resb 4 ; Reserva un buffer de 4 bytes en la sección .bss para almacenar
datos temporales

section .text
 global _start ; Define la etiqueta _start como el punto de inicio del programa
; Macro para imprimir una cadena
%macro PRINT_STRING 1
 mov eax, 4 ; Llamada al sistema para escribir (syscall número 4 en Linux) mov ebx, 1 ;
Descriptor de archivo 1 (stdout) para la salida estándar mov ecx, %1 ; La dirección de la
cadena que se pasará como argumento a la macro mov edx, 13 ; Longitud de la cadena
que se va a imprimir
 int 0x80 ; Llama a la interrupción 0x80 para ejecutar la llamada al sistema
%endmacro

; Macro para imprimir un número
%macro PRINT_NUMBER 1
 mov eax, %1 ; Carga el número que se desea imprimir en el registro EAX add
eax, '0' ; Convierte el valor numérico a su equivalente en ASCII
 mov [buffer], eax ; Almacena el valor ASCII en el buffer
```

mov eax, 4 **; Llamada al sistema para escribir**

mov ebx, 1 **; Descriptor de archivo 1 (stdout) para la salida estándar** mov ecx, buffer

; Dirección del buffer que contiene el número en formato ASCII mov edx, 1 **;**

Longitud del dato a imprimir (1 byte)

int 0x80 **; Llama a la interrupción 0x80 para ejecutar la llamada al sistema**

%endmacro

_start:

; Realiza la suma de los valores en num1 y num2

mov al, [num1] **; Carga el valor de num1 en el registro AL**

add al, [num2] **; Suma el valor de num2 al valor en AL**

mov [result], al **; Almacena el resultado de la suma en la variable result**

; Imprime el mensaje de texto "Resultado: "

PRINT_STRING message ; Llama a la macro PRINT_STRING para imprimir el mensaje
; Imprime el resultado de la suma

PRINT_NUMBER [result] ; Llama a la macro PRINT_NUMBER para imprimir el valor almacenado en result

; Salir del programa

mov eax, 1 **; Llamada al sistema para salir del programa (syscall número 1 en Linux)** mov

ebx, 0 **; Código de salida 0 (sin errores)**

int 0x80 **; Llama a la interrupción 0x80 para ejecutar la salida del programa Macros:** Se definen dos macros para imprimir mensajes y números, que ayudan a mantener el código limpio y evitan repetición.

Operación Principal: La suma de num1 y num2 se almacena en result y luego se imprime usando las macros.

Salida: El programa usa llamadas al sistema de Linux (int 0x80) para manejar la impresión y la salida del programa.

Código 2

```
section .data
 message db "La suma de los valores es: ", 0 ; Mensaje inicial para
mostrar newline db 10, 0 ; Nueva línea para la salida

section .bss
 buffer resb 4 ; Buffer para convertir números a caracteres

section .text
 global _start

%macro DEFINE_VALUES 3
 ; Define una "estructura" con tres valores
 val1 db %1 ; Primer valor
 val2 db %2 ; Segundo valor
 val3 db %3 ; Tercer valor
%endmacro

%macro PRINT_STRING 1
 ; Macro para imprimir una cadena de caracteres
 mov eax, 4 ; Syscall número para 'write'
 mov ebx, 1 ; File descriptor para stdout
 mov ecx, %1 ; Dirección del mensaje
 mov edx, 25 ; Longitud del mensaje
 int 0x80 ; Ejecuta la syscall
%endmacro

%macro PRINT_NUMBER 1
```

; Convierte un número en eax a caracteres ASCII y lo imprime

mov eax, %1 **; Carga el número a imprimir en eax** mov ecx,

buffer + 3 **; Apunta al final del buffer**

mov ebx, 10 **; Divisor para obtener dígitos decimales**

.next_digit:

xor edx, edx **; Limpia edx para la división**

div ebx **; Divide eax entre 10, cociente en eax, residuo en edx** add dl, '0' **;**

Convierte el dígito a ASCII

dec ecx **; Mueve hacia atrás en el buffer**

mov [ecx], dl **; Almacena el dígito en el buffer**

test eax, eax **; Verifica si quedan dígitos**

jnz .next_digit **; Si quedan dígitos, continúa**

; Calcula la longitud del número en el buffer

mov edx, buffer + 4 ; Posición final del buffer
sub edx, ecx ; Calcula la longitud real del número

; Imprime el número

mov eax, 4 ; Syscall para write
mov ebx, 1 **; Salida estándar**

mov ecx, ecx **; Dirección inicial en el buffer**

int 0x80 **; Ejecuta la syscall**

%endmacro

%macro PRINT_SUM 0

; Realiza la suma de tres valores y la imprime

mov al, [val1] **; Carga el primer valor en AL**

```
add al, [val2] ; Suma el segundo valor
add al, [val3] ; Suma el tercer valor
movzx eax, al ; Expande AL a EAX para asegurar un valor de 32 bits

; Imprime el resultado de la suma
PRINT_NUMBER eax
PRINT_STRING newline
%endmacro

; Definimos los tres valores con la macro DEFINE_VALUES
DEFINE_VALUES 3, 5, 7

_start:
; Imprime el mensaje inicial
PRINT_STRING message

; Imprime la suma de los valores
PRINT_SUM
; Salir del programa
mov eax, 1 ; Syscall para 'exit'
mov ebx, 0 ; Código de salida
int 0x80 ; Ejecuta la syscall para salir del programa
```