Universidad Autónoma de Baja California

Facultad de Ingeniería,Arquitectura y Diseño

Ingeniería en Software

Organización de Computadoras

Taller 7 GIT HUB

Brayan Arturo Rocha Meneses

Ensenada Baja California Noviembre de 2024

```
section .data
 num1 db 5 ; Definimos el primer número con valor 5
 num2 db 11 ; Definimos el segundo número con valor 11
 result db 0 ; Definimos una variable para almacenar el resultado
 msg db 'Resultado: ', 0 ; Cadena de texto para el mensaje "Resultado: "
section .bss
 buffer resb 4 ; Espacio reservado para almacenar el resultado en formato ASCII
section .text
 global _start ; Punto de entrada principal del programa
_start:
 ; Cargar los valores de num1 y num2, sumarlos y almacenarlos en 'result'
 mov al, [num1] ; Mueve el valor de num1 al registro AL
 add al, [num2] ; Suma el valor de num2 al registro AL
 mov [result], al ; Almacena el resultado de la suma en 'result'
 ; Convertir el resultado a su equivalente en ASCII
 movzx eax, byte [result] ; Carga el valor de 'result' en EAX, extendiendo con ceros
 add eax, 48 ; Convierte el valor numérico en su equivalente ASCII ('0' = 48)
 mov [buffer], al ; Almacena el carácter ASCII en el buffer
 ; Mostrar el mensaje "Resultado: "
 mov eax, 4 ; Llamada al sistema 'sys_write'
 mov ebx, 1 ; Descriptor del archivo (1 para stdout)
 mov ecx, msg ; Dirección del mensaje
 mov edx, 11 ; Longitud del mensaje
 int 0x80 ; Interrupción de Linux para escribir
```

```
 mov al, [num1] ; Mueve el valor de num1 al registro AL
 add al, [num2] ; Suma el valor de num2 al registro AL
 mov [result], al ; Almacena el resultado de la suma en 'result'
 ; Convertir el resultado a su equivalente en ASCII
 movzx eax, byte [result] ; Carga el valor de 'result' en EAX, extendiendo con ceros
 add eax, 48 ; Convierte el valor numérico en su equivalente ASCII ('0' = 48)
 mov [buffer], al ; Almacena el carácter ASCII en el buffer
 ; Mostrar el mensaje "Resultado: "
 mov eax, 4 ; Llamada al sistema 'sys_write'
 mov ebx, 1 ; Descriptor del archivo (1 para stdout)
 mov ecx, msg ; Dirección del mensaje
 mov edx, 11 ; Longitud del mensaje
 int 0x80 ; Interrupción de Linux para escribir
 ; Mostrar el resultado en formato ASCII
 mov eax, 4 ; Llamada al sistema 'sys_write'
 mov ebx, 1 ; Descriptor del archivo (1 para stdout)
 mov ecx, buffer ; Dirección del buffer que contiene el resultado ASCII
 mov edx, 1 ; Longitud de 1 byte (solo un carácter)
 int 0x80 ; Interrupción de Linux para escribir
 ; Terminar el programa
 mov eax, 1 ; Llamada al sistema 'sys_exit'
 xor ebx, ebx ; Código de salida 0
 int 0x80 ; Interrupción de Linux para salir

```

Introduce un título...

Assembly

```
section .data
 num1 db 'A' ; Se cambia num1 al carácter 'A'
 num2 db '\' ; Se cambia num2 al carácter '\'
 result db '$' ; Se cambia result al carácter '$'
 msg db 'Resultado: ', 0
section .bss
 buffer resb 4
section .text
 global _start
_start:
 ; Almacenamos los caracteres deseados en el buffer
 mov byte [buffer], 'A'
 mov byte [buffer + 1], '\'
 mov byte [buffer + 2], '$'
 mov byte [buffer + 3], '&'
 ; Mostrar el mensaje "Resultado: "
 mov eax, 4
 mov ebx, 1
 mov ecx, msg
 mov edx, 11
 int 0x80
 ; Mostrar cada carácter del buffer
 mov eax, 4
 mov ebx, 1
```

Introduce un título...

Assembly

```
 mov byte [buffer + 2], '$'
 mov byte [buffer + 3], '&'
 ; Mostrar el mensaje "Resultado: "
 mov eax, 4
 mov ebx, 1
 mov ecx, msg
 mov edx, 11
 int 0x80
 ; Mostrar cada carácter del buffer
 mov eax, 4
 mov ebx, 1
 mov ecx, buffer
 mov edx, 4 ; Mostramos 4 caracteres ('A', '\', '$', '&')
 int 0x80
 ; Mostrar el número '1'
 mov eax, 4
 mov ebx, 1
 mov ecx, '1'
 mov edx, 1 ; Mostramos solo el carácter '1'
 int 0x80
 ; Terminar el programa
 mov eax, 1
 xor ebx, ebx
 int 0x80
```