Universidad Autónoma de Baja California
Facultad de Ingeniería Arquitectura y Diseño

Organización de Computadoras

Docente:

Crespo Regland Jonatan

Alumno:

Garcia Ceseña Edgar Rafael

GRUPO 932

Ingeniería en Software y Tecnologías Emergentes

TALLER 7

1.

```
section .data
num1 db 5 ; Declaración de la variable num1 con valor 5
num2 db 11 ; Declaración de la variable num2 con valor 11
result db 0 ; Variable result inicializada en 0 para almacenar el resultado de la
suma
msg db 'Resultado: ', 0 ; Mensaje que se mostrará antes del resultado
section .bss
buffer resb 4 ; Reserva un espacio de 4 bytes en el buffer para almacenar el
carácter ASCII del resultado
section .text
global _start
_start:
; Realizar la suma de num1 y num2
mov al, [num1] ; Cargar el valor de num1 en el registro AL
add al, [num2] ; Sumar el valor de num2 al valor en AL
mov [result], al ; Guardar el resultado de la suma en la variable result
; Convertir el resultado numérico en su representación ASCII
movzx eax, byte [result] ; Cargar el valor de result en EAX y expandirlo a 32 bits
add eax, 48 ; Sumar 48 para convertir el número en su carácter ASCII
correspondiente ('0' = 48)
mov [buffer], al ; Almacenar el carácter ASCII en el buffer
; Mostrar el mensaje "Resultado: "
mov eax, 4 ; Llamada al sistema para escribir (sys_write)
mov ebx, 1 ; File descriptor 1 (salida estándar)
mov ecx, msg ; Dirección del mensaje
mov edx, 11 ; Longitud del mensaje
int 0x80 ; Interrupción para ejecutar la llamada al sistema
```

```
; Mostrar el resultado en ASCII

mov eax, 4 ; Llamada al sistema para escribir (sys_write)
mov ebx, 1 ; File descriptor 1 (salida estándar)
mov ecx, buffer ; Dirección del buffer que contiene el carácter ASCII del resultado
mov edx, 1 ; Longitud de 1 byte (un carácter)
int 0x80 ; Interrupción para ejecutar la llamada al sistema
; Salir del programa
mov eax, 1 ; Llamada al sistema para salir (sys_exit)
xor ebx, ebx ; Código de salida 0
int 0x80 ; Interrupción para ejecutar la llamada al sistema
```

2.

```
section .data
    num1 db 17              ; Valor que producirá A (65) al sumar con num2
    num2 db 48              ; Ajuste de num2 para obtener los valores ASCII
    msg db 'Resultado: ', 0

section .bss
    buffer resb 4

section .text
global _start
_start:
    ; Imprimir cada carácter modificando num1 para obtener el ASCII deseado

    ; Obtener 'A'
    mov al, [num1]          ; num1 = 17
    add al, [num2]          ; num2 = 48, 17 + 48 = 65 ('A')
    mov [buffer], al
    call print_buffer

    ; Obtener '\'
    mov al, 44              ; num1 ajustado a 44 para '\'
    add al, [num2]          ; 44 + 48 = 92 ('\')
    mov [buffer], al
    call print_buffer

    ; Obtener '$'
    mov al, -12             ; num1 ajustado a -12 para '$'
    add al, [num2]          ; -12 + 48 = 36 ('$')
    mov [buffer], al
    call print_buffer

    ; Obtener '&'
    mov al, -10             ; num1 ajustado a -10 para '&'
    add al, [num2]          ; -10 + 48 = 38 ('&')
    mov [buffer], al
    call print_buffer

     ; Obtener '1'
     mov al, 1                  ; num1 ajustado a 1 para '1'
```

```
    ; Obtener '1'
    mov al, 1               ; num1 ajustado a 1 para '1'
    add al, [num2]          ; 1 + 48 = 49 ('1')
    mov [buffer], al
    call print_buffer

    ; Salir del programa
    mov eax, 1              ; sys_exit
    xor ebx, ebx
    int 0x80

print_buffer:
    mov eax, 4              ; sys_write
    mov ebx, 1              ; File descriptor (stdout)
    mov ecx, buffer         ; Dirección del buffer con el carácter ASCII
    mov edx, 1              ; Longitud de 1 byte
    int 0x80
    ret                     ; Retorno para reutilizar la rutina

```

3.

```
section .data
    msg db "Resultado: ", 0

section .bss
    buffer resb 4

section .text
global _start
_start:
    ; Cargar valor ASCII de '@' directamente
    mov eax, 64             ; Carga inmediata del ASCII '@' (64)
    mov [buffer], al        ; Almacenar '@' en buffer

    ; Imprimir el mensaje "Resultado: "
    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, 11
    int 0x80

    ; Imprimir '@'
    mov eax, 4
    mov ebx, 1
    mov ecx, buffer
    mov edx, 1
    int 0x80

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

4.

```
section .data
    symbol db 64            ; ASCII '@'
    msg db 'Resultado: ', 0

section .bss
    buffer resb 4

section .text
global _start
_start:
    ; Cargar el valor de '@' indirectamente
    mov eax, [symbol]       ; Acceso indirecto al valor ASCII '@' desde la dirección de symbol
    mov [buffer], al        ; Almacenar '@' en buffer

    ; Imprimir el mensaje "Resultado: "
    mov eax, 4
    mov ebx, 1
    mov ecx, msg
    mov edx, 11
    int 0x80

    ; Imprimir '@'
    mov eax, 4
    mov ebx, 1
    mov ecx, buffer
    mov edx, 1
    int 0x80

    ; Salir del programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```