

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA ARQUITECTURA Y DISEÑO

INGENIERÍA EN SOFTWARE Y TECNOLOGÍAS EMERGENTES




ORGANIZACIÓN DE COMPUTADORAS


Taller 7

ADRIAN BALDERAS ROSAS

Jonatan Crespo Ragland

1. De acuerdo al código de prueba 1, responde y desarrolla lo siguiente: a. Agrega comentarios en el código explicando su funcionamiento.



42vmc3j46 

```
1- section .data
2   num1 db 5           ; Define el primer número (5) en la sección de datos
3   num2 db 11          ; Define el segundo número (11) en la sección de datos
4   result db 0         ; Define una variable para almacenar el resultado
5   msg db "Resultado: ", 0 ; Mensaje que será impreso, terminado con un null byte
6
7- section .bss
8   buffer resb 4       ; Reserva un espacio de 4 bytes para almacenar el resultado en ASCII
9
10 section .text
11 global _start
12
13 _start:
14     ; Cargar los valores de num1 y num2 en AL y sumar
15     mov al, [num1]     ; Mueve el valor de num1 al registro AL
16     add al, [num2]     ; Suma el valor de num2 al registro AL
17     mov [result], al   ; Almacena el resultado de la suma en la variable result
18
19     ; Convertir el resultado numérico a ASCII
20     movzx eax, byte [result] ; Mueve el resultado a EAX con extensión cero
21     add eax, 48         ; Convierte el valor a su representación ASCII sumando 48 (ASCII de '0')
22
23     mov [buffer], al    ; Almacena el carácter ASCII en el buffer
24
25     ; Imprimir "Resultado: "
26     mov eax, 4         ; syscall para escribir (sys_write)
27     mov ebx, 1         ; file descriptor para stdout
28     mov ecx, msg       ; apuntar al mensaje que queremos imprimir
29     mov edx, 11        ; longitud del mensaje
30     int 0x80           ; llamada al sistema (syscall)
31
32     ; Imprimir el valor almacenado en buffer
33     mov eax, 4         ; syscall para escribir (sys_write)
34     mov ebx, 1         ; file descriptor para stdout
35     mov ecx, buffer     ; apunta al buffer donde está almacenado el carácter ASCII
36     mov edx, 1         ; longitud del buffer (1 byte)
37     int 0x80           ; llamada al sistema (syscall)
38
39     ; Salir del programa
40     mov eax, 1         ; syscall para salir (sys_exit)
41     xor ebx, ebx       ; código de salida 0
42     int 0x80           ; llamada al sistema (syscall)
43
```

Modificación del código:

```

1 section .data
2     characters db 65, 92, 36, 38, 49 ; Valores ASCII de A, l, $, 8, 1
3     msg db 'Resultado: ', 0
4
5 section .bss
6     buffer resb 1
7
8 section .text
9     global _start
10
11 _start:
12     mov esi, characters ; Apuntar al inicio del arreglo de caracteres
13
14     ; Imprimir cada carácter en orden
15 print_char:
16     mov al, [esi] ; Cargar el valor ASCII en al
17     mov [buffer], al ; Guardar el valor en el buffer
18
19     ; Imprimir el carácter almacenado en buffer
20     mov eax, 4 ; syscall para escribir
21     mov ebx, 1 ; file descriptor para stdout
22     mov ecx, buffer ; buffer con el carácter
23     mov edx, 1 ; longitud del carácter
24     int 0x80 ; llamada al sistema
25
26     add esi, 1 ; Avanzar al siguiente carácter
27     cmp byte [esi], 0 ; Verificar si llegamos al final
28     jne print_char ; Si no es el final, imprimir el siguiente carácter
29
30     ; Salir del programa
31     mov eax, 1 ; syscall para salir
32     xor ebx, ebx ; código de salida 0
33     int 0x80 ; llamada al sistema
34

```

Direccionamiento inmediato:

```

1 section .data
2     msg db '@', 0
3
4 section .text
5     global _start
6
7 _start:
8     ; Imprimir '@' directamente
9     mov eax, 4 ; syscall para escribir
10    mov ebx, 1 ; file descriptor para stdout
11    mov ecx, msg ; mensaje que contiene '@'
12    mov edx, 1 ; longitud de 1 byte
13    int 0x80 ; llamada al sistema
14
15    ; Salir del programa
16    mov eax, 1 ; syscall para salir
17    xor ebx, ebx ; código de salida 0
18    int 0x80 ; llamada al sistema
19

```

Direccionamiento indirecto:

```

1 section .data
2     character db '@'
3
4 section .bss
5     buffer resb 1
6
7 section .text
8 global _start
9
10 _start:
11     ; Usar direccionamiento indirecto para obtener '@'
12     mov al, [character] ; Cargar el valor de '@'
13     mov [buffer], al    ; Guardar en buffer
14
15     ; Imprimir el carácter
16     mov eax, 4          ; syscall para escribir
17     mov ebx, 1          ; file descriptor para stdout
18     mov ecx, buffer     ; buffer que contiene '@'
19     mov edx, 1          ; longitud de 1 byte
20     int 0x80            ; llamado al sistema
21
22     ; Salir del programa
23     mov eax, 1          ; syscall para salir
24     xor ebx, ebx        ; código de salida 0
25     int 0x80            ; llamado al sistema
26

```