UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño.

Ingeniería en Software y Tecnologías Emergentes

Organización de Computadoras

Taller 9

Alejandro Palomares Ceseña

Desarrollar lo siguiente en su cuaderno o computadora:

1. De acuerdo al código de prueba https://onecompiler.com/assembly/443pg5uuh responde y desarrolla lo siguiente:

a. Agrega comentarios en el código explicando su funcionamiento.

```
section .data
char db 'A'            ; Variable almacenada en memoria: contiene el caracter A

section .text
global _start

_start:
    movzx eax, byte [char] ; Carga el byte de memoria a EAX, extendiéndolo a 32 bits con ceros

    mov ebx, 1             ; Descriptor 1 = salida estándar (pantalla)
    mov ecx, char          ; Dirección de memoria del carácter a imprimir
    mov edx, 1             ; Cantidad de bytes a imprimir (1 byte)
    mov eax, 4             ; syscall 4 = write()
    int 0x80               ; Llamada al sistema → imprime el carácter

    mov eax, 1             ; syscall 1 = exit()
    mov ebx, 0             ; Código de salida 0
    int 0x80               ; Termina el programa
```

b. ¿Para qué sirve la instrucción 'movzx'?

movzx significa "Move with Zero-Extend"
Carga un valor pequeño (byte o word) en un registro más grande y rellena los bits superiores con ceros.

c. Está usando algún modo de direccionamiento. Si o no? Que modo de direccionamiento está utilizando.

Sí.
El programa usa:

Modo de direccionamiento directo a memoria.

d. Explica que imprime el programa y por qué.

El programa imprime:

A

Razones:

- La variable char contiene el byte 'A'
- Se usa la syscall write(1, char, 1)
- Eso envía 1 byte al terminal → el carácter ASCII 65 → 'A'

e. Modifica el programa para que imprima lo siguiente: A, \, $, & y 1. Documenta tu procedimiento.

```
section .data
chars db 'A', '\', '$', '&', '1'

section .text
global _start

_start:
    mov ebx, 1
    mov ecx, chars
    mov edx, 5
    mov eax, 4
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Output:

A\$&1

Ahora chars contiene 5 caracteres consecutivos en memoria.

“write(1, chars, 5)” imprime los 5 bytes en orden.

f. Después de terminar el punto anterior, contesta lo siguiente: ¿Fue la única forma de modificar el código para llegar a esos resultados? ¿Qué otra línea pudiste modificar para llegar a los mismos resultados?

No.
También se pudo modificar:

char db 'A'

Por:

char db 'A', '\', '$', '&', '1'

O de manera individual escribiendo:

mov byte [char], '\'

La tarea se puede resolver:

- modificando la definición de datos (.data)
- modificando el puntero ECX
- modificando el número de bytes EDX

g. Utilizando de nuevo el código de prueba original, modifica el código para que ahora utilice el modo de direccionamiento inmediato e indirecto (en programas separados) para que imprima el carácter '@'. Documenta tus resultados.

Modo de direccionamiento inmediato:

```
section .data

section .text
global _start

_start:

    mov al, '@'      ; inmediato: el valor literal se mete al registro
    mov [car], al    ; se pasa a memoria para imprimir

    mov edx, 1
    mov ecx, car
    mov ebx, 1
    mov eax, 4
    int 0x80

    mov eax, 1
    int 0x80

section .bss
car resb 1
```

Output:

@

Modo de direccionamiento indirecto:

```
section .data
    simbolo db '@'       ; carácter
    ptrSimbolo dd simbolo ; puntero al carácter (indirecto)

section .text
global _start

_start:

    mov ebx, [ptrSimbolo]   ; carga la dirección guardada en el puntero
    mov al, [ebx]           ; accede al carácter mediante direccionamiento indirecto
    mov [salida], al

    mov edx, 1
    mov ecx, salida
    mov ebx, 1
    mov eax, 4
    int 0x80

    mov eax, 1
    int 0x80

section .bss
    salida resb 1
```

Output:

@

2. Responde lo siguiente:

a. ¿Cómo afecta el modo de direccionamiento a la eficiencia de un programa en ensamblador?

- Modos **inmediatos** son los más rápidos (el dato está en la instrucción).
- Modos **de registro** también son rápidos.
- Modos **directos e indirectos** son más lentos porque requieren acceso a memoria.
- Modos **indexados o con desplazamiento** pueden requerir cálculos adicionales.

En ensamblador, elegir bien el modo puede ahorrar ciclos y mejorar el rendimiento.

b. ¿Qué papel juegan los modos de direccionamiento en la programación de

sistemas y controladores?

Son esenciales porque:

- permiten acceder a memoria, puertos y hardware
- permiten manipular buffers, tablas, estructuras y dispositivos
- permiten leer registros de control en direcciones fijas
- permiten manejar interrupciones y estructuras de memoria

Los drivers y sistemas operativos dependen completamente de los modos de direccionamiento.