1. De acuerdo al código de prueba 1, responde y

desarrolla lo siguiente:

a. Agrega comentarios en el código explicando su

funcionamiento.

HelloWorld.asm 43h9zs4wh

```asm
section .data
    num1 db 5          ; Define un byte con valor 5
    num2 db 11         ; Define un byte con valor 11
    result db 0        ; Variable para almacenar el resultado
    msg db 'Resultado: ', 0 ; Cadena de texto a imprimir

section .bss
    buffer resb 4      ; Reservar 4 bytes para almacenar caracteres (sobra en este caso)

section .text
    global _start

_start:

    mov al, [num1]     ; Cargar el valor de num1 (5) en AL
    add al, [num2]     ; Sumar el valor de num2 (11) al contenido de AL → AL = 16
    mov [result], al   ; Guardar el resultado (16) en la variable result

    ; Convertir el resultado a su código ASCII para poder imprimirlo como carácter
    movzx eax, byte [result] ; Expandir sin signo el valor de result a 32 bits
    add eax, 48        ; Sumar 48 para convertir a código ASCII (16 + 48 = 64 → '@')
    mov [buffer], al   ; Guardar ese carácter en buffer

    ; Imprimir la cadena "Resultado: "
    mov eax, 4         ; syscall número 4 = sys_write
    mov ebx, 1         ; descriptor de archivo 1 = salida estándar
    mov ecx, msg       ; apuntador al mensaje
    mov edx, 11        ; longitud del mensaje
    int 0x80

    ; Imprimir el carácter convertido (el resultado de la suma)
    mov eax, 4
    mov ebx, 1
    mov ecx, buffer    ; apuntador al carácter
    mov edx, 1         ; longitud = 1 byte
    int 0x80

    ; Terminar el programa
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

b. ¿Para qué sirve la instrucción ‘movzx’?
Sirve para copiar un valor pequeño (como un byte) a un registro más grande (como EAX de 32 bits).

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

Si [result] = 16 (en binario: 00010000), al hacer movzx eax, byte [result], EAX se vuelve 00000000 00000000 00000000 00010000.}

d. Explica que imprime el programa y por qué.

Imprime una arroba.

5 + 11 = 16, se convierte el resultado (16) en su carácter ASCII sumándole 48: 16 + 48 = 64 El carácter ASCII con valor decimal 64 es '@'

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

Los códigos ASCII de esos caracteres son: 'A' = 65 ‘' = 92 ‘$' = 36 '&' = 38 '1' = 49.

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 fue la única forma. Otra opción es modificar directamente el valor que se le suma para formar el carácter ASCII.

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.

“

```
section .data
    caracter db '@'

section .text
    ; ...
    lea esi, [caracter] ; Cargar dirección (modo indirecto)
    mov al, [esi]
    mov [buffer], al
```

”