1. De acuerdo al código de prueba 1, responde y desarrolla lo siguiente:

a. Agrega comentarios en el código explicando su funcionamiento.

```
    ; Declaracion de datos constantes
    num1 db 5              ; Primer número (byte)
    num2 db 11             ; Segundo número (byte)
    result db 0            ; Variable para almacenar el resultado de la suma
    msg db 'Resultado: ', 0 ; Mensaje a mostrar antes del resultado

section .bss
    ; Sección para variables no inicializadas
    buffer resb 4          ; Reserva 4 bytes para almacenar caracteres

section .text
global _start

_start:
    ; Cargar num1 en el registro AL
    mov al, [num1]

    ; Sumar el valor de num2 a AL
    add al, [num2]

    ; Almacenar el resultado de la suma en la variable result
    mov [result], al

    ; Convertir el número resultante a su representación ASCII
    movzx eax, byte [result] ; Mueve el valor a EAX con extensión cero
    add eax, 48              ; Suma 48 para obtener el código ASCII del número ('0' = 48)

    ; Guardar el carácter ASCII resultante en el buffer
    mov [buffer], al

    ; Llamada al sistema para imprimir el mensaje "Resultado: "
    mov eax, 4               ; syscall número 4: sys_write
    mov ebx, 1               ; descriptor de archivo 1: salida estándar
    mov ecx, msg             ; dirección del mensaje
    mov edx, 11              ; longitud del mensaje ("Resultado: ")
    int 0x80                 ; interrupción para invocar syscall

    ; Llamada al sistema para imprimir el carácter del resultado
    mov eax, 4               ; syscall número 4: sys_write
    mov ebx, 1               ; salida estándar
    mov ecx, buffer          ; dirección del buffer con el resultado ASCII
    mov edx, 1               ; longitud: 1 carácter
    int 0x80                 ; invoca syscall

    ; Terminar el programa correctamente
    mov eax, 1               ; syscall número 1: sys_exit
    xor ebx, ebx             ; código de salida 0
    int 0x80                 ; invoca syscall
```

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

- Para A:

```
section .data
    ; Declaración de datos con
    num1 db 5              ; P
    num2 db 12             ; S
    result db 0            ; V
    msg db 'Resultado: ', 0 ;
```

- Para \

```
section .data
    ; Declaración de datos con
    num1 db 8               ; P
    num2 db 12              ; S
    result db 0             ; V
    msg db 'Resultado: ', 0 ;

section .bss
    ; Sección para variables n
    buffer resb 4           ; R

section .text
global _start

_start:
    ; Cargar num1 en el regist
    mov al, [num1]

    ; Sumar el valor de num2 a
    add al, [num2]

    ; Almacenar el resultado d
    mov [result], al

    ; Convertir el número resu
    movzx eax, byte [result] ;
    add eax, 72              ;
```

- Para $

```
    ; Convertir el numero resu
    movzx eax, byte [result]
    add eax, 16             ;
```

- Para &

```
movzx eax, byte [result] ; M
add eax, 18              ; Su
```

- Para 1

```
section .data
    ; Declaración de datc
    num1 db 0
    num2 db 1
    result db 0
    msg db 'Resultado: ',

section .bss
    ; Sección para variak
    buffer resb 4

section .text
global _start

_start:
    ; Cargar num1 en el r
    mov al, [num1]

    ; Sumar el valor de r
    add al, [num2]

    ; Almacenar el result
    mov [result], al

    ; Convertir el númerc
    movzx eax, byte [resu
    add eax, 48
```

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

section .bss
    buffer resb 1

section .text
    global _start

_start:
    ; Dirección inmediata: carga directamente 64 en AL
    mov al, 64
    mov [buffer], al

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

    ; Imprimir el carácter '@'
    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
```