Codigo 1: Multiplicación de vectores con conversión ASCII correcta

```
section .data
   vec1    dd 1, 2, 3, 4, 5      ; Vector 1
   vec2    dd 2, 3, 4, 5, 6      ; Vector 2
   len     equ ($ - vec2) / 4     ; Longitud (5 elementos)
   msg     db "Resultado: ", 0     ; Mensaje
   nl      db 0xA                ; Nueva línea

section .bss
   buffer  resb 10               ; Búfer para la conversión ASCII

section .text
   global _start

_start:
   ; --- Calcular producto punto ---
   xor eax, eax              ; Acumulador (resultado)
   xor ebx, ebx              ; Índice

loop:
   mov ecx, [vec1 + ebx*4]     ; Carga vec1[i]
   imul ecx, [vec2 + ebx*4]    ; Multiplica por vec2[i]
   add eax, ecx              ; Suma al acumulador
   inc ebx                  ; Siguiente elemento
   cmp ebx, len
   jl loop                  ; Repetir si ebx < len

   ; --- Convertir resultado a ASCII (para múltiples dígitos) ---
   mov r8d, eax              ; Guardar resultado
   lea rsi, [buffer + 9]      ; Apuntar al final del búfer
   mov byte [rsi], 0xA        ; Añadir nueva línea al final
   mov ecx, 10               ; Divisor (base 10)

convert_loop:
   xor edx, edx              ; Limpiar EDX para la división
   div ecx                  ; Dividir EAX/10 (EAX = cociente, EDX = resto)
   add dl, '0'               ; Convertir resto a ASCII
   dec rsi                  ; Mover puntero hacia atrás
   mov [rsi], dl             ; Almacenar dígito
   test eax, eax             ; ¿Terminamos?
   jnz convert_loop          ; Si EAX != 0, repetir

   ; --- Imprimir mensaje ---
   mov rax, 1                ; sys_write
   mov rdi, 1                ; stdout
   mov rdx, 11               ; Longitud de "Resultado: "
   mov rsi, msg
```

```
    syscall

    ; --- Imprimir resultado ---
    mov rdx, buffer + 10       ; Calcular longitud del número
    sub rdx, rsi               ; RDX = longitud (buffer+10 - RSI)
    mov rax, 1
    mov rdi, 1
    syscall

    ; --- Imprimir nueva línea ---
    mov rax, 1
    mov rdi, 1
    mov rsi, nl
    mov rdx, 1
    syscall

    ; --- Salir ---
    mov rax, 60                ; sys_exit
    xor rdi, rdi               ; código 0
    syscall
```

El programa calcula el producto punto entre dos vectores definidos en la sección .data. Los vectores vec1 y vec2 contienen valores enteros de 32 bits, y su longitud se calcula automáticamente mediante una operación aritmética con las direcciones de memoria.

En la sección .bss se reserva espacio para almacenar el resultado después de convertirlo a formato ASCII. El búfer tiene 10 bytes de capacidad para garantizar que pueda contener números grandes.

El cálculo principal ocurre en un bucle que recorre ambos vectores simultáneamente. En cada iteración, multiplica los elementos correspondientes de ambos vectores y acumula la suma en el registro EAX. Este enfoque eficiente evita accesos innecesarios a memoria.

Para mostrar el resultado, el programa realiza una conversión numérica a ASCII mediante divisiones sucesivas por 10. Cada dígito se convierte individualmente y se almacena en orden inverso en el búfer reservado, asegurando la correcta representación del número.

Finalmente, el programa utiliza llamadas al sistema para imprimir primero el mensaje descriptivo, seguido del resultado numérico convertido y un salto de línea. La ejecución termina limpiamente con una llamada a sys_exit, devolviendo código 0 para indicar éxito.

El diseño garantiza que funcione para cualquier resultado válido de 32 bits, mostrando correctamente números de uno o múltiples dígitos. La conversión a ASCII evita los problemas de representación que ocurren al intentar mostrar números directamente.