Código 7: Ordenar números de un arreglo (Método Burbuja)

```
section .data
   vector dd 5, 3, 8, 1, 4, 2
   len    equ 6
   salto  db 0x0A       ; salto de línea

section .bss
   buffer resb 12       ; espacio para número convertido a cadena

section .text
   global _start

_start:
   call burbuja

   ; Imprimir vector ordenado
   mov ecx, 0
imprimir_loop:
   mov eax, [vector + ecx*4]
   push ecx
   push eax
   call print_int
   add esp, 4
   ; imprimir salto de línea
   mov eax, 4
   mov ebx, 1
   mov ecx, salto
   mov edx, 1
   int 0x80
   pop ecx
   inc ecx
   cmp ecx, len
   jl imprimir_loop

   ; Salir
   mov eax, 1
   xor ebx, ebx
   int 0x80

;-------------------------------
; Función: burbuja
;-------------------------------
burbuja:
   mov     ecx, len
   dec     ecx
   jle     fin_burbuja
outer:
```

```
    push    ecx
    mov     esi, 0
inner:
    mov     eax, [vector + esi*4]
    mov     ebx, [vector + esi*4 + 4]
    cmp     eax, ebx
    jle     no_swap
    mov     [vector + esi*4], ebx
    mov     [vector + esi*4 + 4], eax
no_swap:
    inc     esi
    cmp     esi, [esp]
    jl      inner
    pop     ecx
    dec     ecx
    jg      outer
fin_burbuja:
    ret

;---------------------------------
; Función: print_int
;---------------------------------
print_int:
    mov edi, buffer + 11
    mov byte [edi], 0
    mov ebx, 10
print_loop:
    xor edx, edx
    div ebx
    add dl, '0'
    dec edi
    mov [edi], dl
    test eax, eax
    jnz print_loop

    mov eax, 4
    mov ebx, 1
    mov ecx, edi
    mov edx, buffer + 11
    sub edx, edi
    int 0x80
    ret
```

Este programa en ensamblador NASM para Linux ordena un arreglo de seis enteros utilizando el método de la burbuja y luego imprime los elementos ordenados, uno por línea, en la consola. En la sección .data se define el vector con los valores a ordenar, usando la directiva dd para declarar palabras dobles (enteros de 32 bits). También se define una constante de salto de línea (0x0A), que será usada para mejorar la legibilidad al imprimir los resultados. En la sección .bss, que se usa para variables no inicializadas, se reserva un espacio llamado buffer, con 12 bytes, para convertir enteros a texto antes de imprimirlos.

El programa comienza su ejecución en la etiqueta _start, la cual es reconocida como el punto de entrada por el enlazador. Lo primero que hace es llamar a la subrutina burbuja, encargada de realizar el ordenamiento. Esta subrutina implementa el clásico algoritmo de la burbuja, donde se realizan múltiples pasadas por el vector, comparando elementos adyacentes y haciendo intercambios si están en el orden incorrecto. El número total de pasadas necesarias es igual al número de elementos menos uno. Para lograr esto, se utilizan registros como ecx (contador de pasadas externas), esi (índice interno) y eax/ebx (valores a comparar e intercambiar). Las comparaciones y los intercambios se hacen directamente sobre la memoria del vector usando direccionamiento con desplazamiento y escalamiento ([vector + esi*4]).

Una vez que el vector ha sido ordenado, el control regresa a _start, que procede a imprimir cada uno de los elementos del vector ya ordenado. Para ello, se inicia un bucle controlado por el registro ecx, que recorre los índices del vector. En cada iteración, el valor del elemento actual se carga en el registro eax, se guarda temporalmente en la pila y luego se llama a la subrutina print_int.

La subrutina print_int se encarga de convertir el número entero en formato decimal ASCII para poder imprimirlo. Esto se logra utilizando divisiones sucesivas por 10, almacenando los restos (dígitos) desde el final del búfer hacia el inicio. Una vez que se completa la conversión, se realiza una llamada al sistema sys_write (mediante int 0x80) para enviar los caracteres correspondientes al descriptor de archivo 1 (stdout), mostrando así el número en pantalla.

Después de imprimir cada número, el programa también imprime un salto de línea para separar visualmente los valores. Esto se repite para cada elemento del vector. Finalmente, al terminar el bucle de impresión, el programa invoca sys_exit con un código de retorno de 0, terminando su ejecución de forma limpia.