Código 6: Números Múltiplos de 3

```
section .data
   arr     dd 3, 5, 6, 9, 10, 12
   len     equ 6
   newline db 10

section .bss
   buffer  resb 12

section .text
   global _start

_start:
   xor ecx, ecx       ; índice
   xor edi, edi       ; contador

.loop:
   mov eax, [arr + ecx*4]
   xor edx, edx
   mov ebx, 3
   div ebx
   test edx, edx
   jne .skip
   inc edi

.skip:
   inc ecx
   cmp ecx, len
   jl .loop

   ; convertir edi a cadena ASCII
   mov eax, edi
   mov ebx, 10
   mov esi, buffer + 11  ; apuntamos al final
.conv:
   xor edx, edx
   div ebx
   dec esi
   add dl, '0'
   mov [esi], dl
   test eax, eax
   jnz .conv

   ; imprimir resultado
   mov eax, 4
   mov ebx, 1
   mov ecx, esi
```

```
mov edx, buffer + 12
sub edx, ecx
int 0x80

; imprimir salto de línea
mov eax, 4
mov ebx, 1
mov ecx, newline
mov edx, 1
int 0x80

; salir
mov eax, 1
xor ebx, ebx
int 0x80
```

Este programa en ensamblador NASM para Linux cuenta cuántos números en un arreglo de enteros son divisibles entre 3. Para ello, primero define un arreglo de seis enteros en la sección .data, además de un salto de línea para mostrar el resultado correctamente. En la sección .bss, se reserva espacio con buffer para convertir el número final a una cadena de texto (ASCII) antes de imprimirlo.

El programa inicia con la etiqueta _start. Primero se inicializan dos registros: ecx para llevar el índice del arreglo, y edi como contador de cuántos elementos cumplen la condición (ser divisibles entre 3). Luego, entra en un bucle .loop que recorre cada elemento del arreglo. En cada iteración, carga el valor actual en eax y lo divide entre 3 usando div. El resultado de la división no importa directamente, sino el residuo (edx). Si edx es cero, eso significa que el número es divisible entre 3, así que incrementa el contador edi.

Después de recorrer todos los elementos, el valor final del contador edi se convierte a texto ASCII para que se pueda imprimir. Esto se hace con otro pequeño bucle .conv, donde se toman los dígitos de derecha a izquierda y se guardan en el búfer. Una vez convertido, se imprime el resultado con sys_write usando la interrupción int 0x80.

Finalmente, se imprime un salto de línea y se termina el programa limpiamente con sys_exit. Todo el código es de 32 bits y está diseñado para ejecutarse en Linux con llamadas al sistema (int 0x80), lo cual lo hace directo y funcional para propósitos educativos.