Manual de Ejercicios: Macros y Procedimientos en Ensamblador (NASM)

Este manual se enfoca en la modularización del código ensamblador, utilizando **Macros** para la reutilización de bloques y **procedimientos** para el manejo de la pila y la estructura de interfaz.

1. Macros: La Automatización de la Interfaz

Las macros permiten definir bloques de código que el ensamblador expande antes de la compilación. Ideales para tareas repetitivas como imprimir cadenas.

Ejercicio 1: Macro para Impresión en Pantalla

```
None
%macro print 2
    mov eax, 4          ; sys_write
    mov ebx, 1          ; stdout
    mov ecx, %1         ; Dirección del mensaje
    mov edx, %2         ; Longitud
    int 0x80
%endmacro

section .data
    msg db 'Interfaz Cargada', 0xA
    len equ $-msg

section .text
    global _start
_start:
    print msg, len
```

```
    mov eax, 1
    int 0x80
```

2. Procedimientos: Lógica Estructurada

Los procedimientos utilizan la instrucción call y ret, empleando la pila (stack) para gestionar el flujo de datos.

Ejercicio 2: Procedimiento para Limpiar Pantalla

None

```
section .text
    global _start
_start:
    call limpiar_pantalla
    mov eax, 1
    int 0x80

limpiar_pantalla:
    ; Rutina técnica para limpiar interfaz (ANSI escape codes)
    push eax
    ; ... (instrucciones de sistema)
    pop eax
    ret
```

3. Desarrollo de Aplicaciones con Interfaz

Para aplicaciones de software con interfaz, los procedimientos permiten segmentar el código por funciones:

- **Menu_Principal:** Procedimiento para dibujar opciones.
- **Captura_Datos:** Macro para validación de entrada.
- **Logica_Procesamiento:** Procedimiento central de cálculo.

Tabla Comparativa: Macros vs. Procedimientos

Característica	Macros	Procedimientos
Tiempo de ejecución	Se expanden al compilar	Se ejecutan en tiempo real
Uso de Memoria	Aumentan el tamaño binario	Optimizan el uso de memoria
Flujo	Lineal/Sustitución	Salto (CALL/RET)