1. suma_simple.asm

Por qué se escogió:

Este ejercicio fue elegido porque permite practicar operaciones aritméticas básicas, como la suma, en lenguaje ensamblador. Es un punto de partida para entender cómo se manejan los registros y la memoria.

Simplicidad:

Muy simple. No hay bucles ni entrada del usuario. Solo muestra cómo sumar dos números fijos.

2. resta_simple.asm

Por qué se escogió:

Complementa el ejercicio anterior al incluir una operación diferente (resta). Ayuda a reforzar el uso de instrucciones aritméticas y el almacenamiento en registros.

Simplicidad:

Fácil. Opera directamente con constantes definidas.

3. saludo_usuario.asm

Por qué se escogió:

Muestra cómo trabajar con cadenas de texto. Aunque es muy similar a imprimir un mensaje, este ejemplo sirve para entender cómo se define una cadena y se usa con write.

Simplicidad:

Fácil. No incluye lógica adicional, solo salida de texto.

4. suma_usuario.asm

Por qué se escogió:

Este programa simula la suma de dos números sin entrada real. Fue seleccionado como puente entre los programas básicos y los que sí usan lógica más compleja.

Simplicidad:

Fácil. Opera con variables simuladas como si fueran datos del usuario.

◈ Programa 5: saludo_usuario.asm

Descripción: Imprime un saludo personalizado al usuario.
Razón de elección:
Este ejercicio permite practicar la impresión de cadenas en la salida estándar. Se eligió por su **simplicidad** y por ser un paso inicial para comprender cómo trabajar con texto en ensamblador usando **syscalls de Linux**. Además, sienta la base para ejercicios más complejos que implican interacción con el usuario.

6: ciclo_for.asm

Descripción: Simula un ciclo for que imprime números del 1 al 5.
Razón de elección:
Este ejercicio se escogió porque **introduce estructuras de control repetitivas**, en particular bucles. Es un punto intermedio entre los programas más básicos y aquellos más complejos. Ayuda a desarrollar lógica de iteración sin introducir estructuras avanzadas.

7: comparacion.asm

Descripción: Compara dos números y muestra cuál es mayor.
Razón de elección:
Este programa permite practicar instrucciones de **comparación (cmp) y salto condicional (jg, jl, etc.)**, lo cual es fundamental en cualquier lenguaje de programación. Se eligió por ser sencillo, pero muy útil para comenzar a controlar el flujo del programa.

8: entrada_usuario.asm

Descripción: El usuario introduce un carácter, y este se muestra en pantalla.
Razón de elección:
Este ejercicio **introduce la lectura de datos desde teclado**, utilizando llamadas al

sistema de Linux. Fue elegido porque incrementa la dificultad de los programas anteriores al incluir entrada y salida de datos, lo cual es esencial para crear aplicaciones interactivas.

9: multiplicacion.asm

Descripción: Multiplica dos números definidos en el código.
Razón de elección:
Se seleccionó porque permite trabajar con operaciones **aritméticas básicas** usando la instrucción mul. Aunque es sencillo, va un paso más allá de la suma/resta básica, introduciendo el manejo del resultado de multiplicación y registros especiales como ax.

Programa 10: factorial.asm

Descripción: Calcula el factorial de un número usando un bucle.
Razón de elección:
Este es el programa más **complejo de la primera mitad**. Se eligió porque combina conceptos aprendidos previamente: bucles, aritmética, comparaciones y lógica de control. El cálculo del factorial es un excelente ejemplo para probar la comprensión de estructuras iterativas y operaciones en ensamblador