

.data: Esta sección del código se utiliza para definir variables globales estáticas y sus valores iniciales.

oper1: Es una palabra (4 bytes) que contiene el valor 5.

oper2: Es otra palabra que contiene el valor 4.

res: Es una palabra inicializada con 0, donde se almacenará el resultado de la suma.

.text: Esta sección contiene el código ejecutable.

ldr r1, =oper1: Carga en el registro r1 la dirección de memoria de oper1.

ldr r2, =oper2: Carga en el registro r2 la dirección de memoria de oper2.

ldr r3, =res: Carga en el registro r3 la dirección de memoria de res.

push {r1, r2, r3}: Guarda los valores de los registros r1, r2 y r3 en la pila. Estos registros contienen las direcciones de memoria de oper1, oper2 y res, respectivamente.

bl suma2: Llama a la función suma2.

suma2: Esta es una subrutina que realiza la suma.

pop {r1, r2}: Recupera los valores de los registros r1 y r2 de la pila.

ldr r1, [r1]: Carga el valor almacenado en la dirección de memoria apuntada por r1 (valor de oper1) en r1.

ldr r2, [r2]: Carga el valor almacenado en la dirección de memoria apuntada por r2 (valor de oper2) en r2.

add r4, r1, r2: Suma los valores de r1 y r2 y guarda el resultado en r4.

pop {r3}: Recupera el valor de r3 de la pila.

str r4, [r3]: Almacena el resultado de la suma (en r4) en la dirección de memoria apuntada por r3 (en res).

mov pc, lr: Devuelve el control a la instrucción que llamó a esta subrutina.

mov r5, r3: Mueve el contenido de r3 (la dirección de memoria de res) a r5.

bl fin: Llama a la función fin.

fin: Termina el programa.

Un ejemplo simple de cómo funciona este programa sería:

Supongamos que las direcciones de memoria de oper1, oper2 y res son 0x100, 0x104 y 0x108 respectivamente. Y en esas direcciones, los valores son 5, 4 y 0 inicialmente.

La suma de 5 y 4 da como resultado 9.

Entonces, después de que se ejecute este programa, el valor en la dirección de memoria 0x108 (res) será 9.