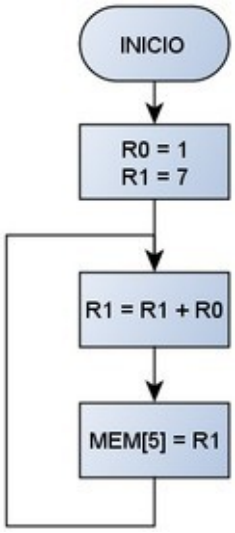
Programas usando el lenguaje ensamblador.

Ejemplo1. Vamos a realizar un programa que genere un contador a partir del numero 7 y se incremente de uno en uno. Observemos el diagrama de flujo de la ilustración 1.

Diagrama de flujo



Programa

```
LI R0, #1
LI R1, #7

CICLO:
ADD R1, R1, R0

SWI R1, 5

B CICLO
```

Ilustración 1 Algoritmo de un contador

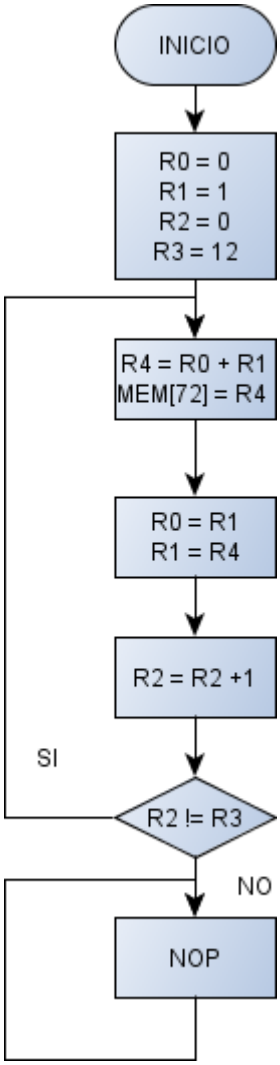
El programa que implementa el diagrama de flujo de la ilustración 1 se muestra en la tabla 1.

Instrucciones	Significado	Dirección	24…20	19…16	15...12	11...8	7...4	3...0	T
LI R0, #1	R0 = 1	0	00001	0000 (R0)	0000	0000	0000	0001	I
LI R1, #7	R1 = 7	1	00001	0001 (R1)	0000	0000	0000	0111	I
CICLO: ADD R1, R1, R0	R1 = R1 + R0	2	00000	0001 (R1)	0001 (R1)	0000 (R0)	xxxx	0000	R
SWI R1, 5	Mem[5] = R1	3	00011	0001 (R1)	0000	0000	0000	0101	I
B CICLO	goto CICLO (PCx = 2)	4	10011 (19)	xxxx	0000	0000	0000	0010	J

Tabla 1 Programa del contador

Ejemplo 2. Vamos a realizar un programa que obtenga los primeros **12 términos** de la serie de Fibonacci. Los valores iniciales de la serie son 0 y 1 y se colocan R0 = 0 y R1 = 1. Cada término de la serie se obtiene sumando los 2 números anteriores. Además, el término de la serie calculado debe colocarse en la dirección de memora 72. Observemos la solución mostrada en el diagrama de flujo de la ilustración 2.

Diagrama de flujo



Programa

```
LI R0, #0
LI R1, #1
LI R2, #0
LI R3, #12

CICLO:

ADD R4, R0, R1
SWI R4, 72

ADDI R0, R1, #0  ; R0 = R1 + 0
ADDI R1, R4, #0  ; R0 = R4 + 0

ADDI R2, R2, #1

BNEI R2, R3, CICLO

FIN:

NOP

B FIN
```

Ilustración 2 Algoritmo de la serie de Fibonacci

El programa que implementa el diagrama de flujo de la ilustración 2 se muestra en la tabla 2.

Instrucciones	Significado	Dir	24…20	19…16	15...12	11...8	7...4	3...0	T
LI R0, #0	R0 = 0	0	00001	0000 (R0)	0000	0000	0000	0000	I
LI R1, #1	R1 = 1	1	00001	0001 (R1)	0000	0000	0000	0001	I
LI R2, #0	R2 = 0	2	00001	0010 (R2)	0000	0000	0000	0000	I
LI R3, #12	R3 = 12	3	00001	0011 (R3)	0000	0000	0000	1100	I
CICLO: ADD R4, R0, R1	R4 = R0 + R1	4	00000	0100 (R4)	0000 (R0)	0001 (R1)	xxxx	0000	R
SWI R4, 72	MEM[72] = R4	5	00011	0100 (R4)	0000	0000	0100	1000	I
ADDI R0, R1,#0	R0 = R1 + 0	6	00101	0000 (R0)	0001 (R1)	0000	0000	0000	I
ADDI R1, R4,#0	R1 = R4 + 0	7	00101	0001 (R1)	0100 (R4)	0000	0000	0000	I
ADDI R2, R2,#1	R2 = R2 + 1	8	00101	0010 (R2)	0010 (R2)	0000	0000	0001	I
BNEI R2, R3, CICLO	If(R2 != R3) goto CICLO PCx = PCx + (-5) = 9 + (-5) = 4	9	01110	0010 (R2)	0011 (R3)	1111	1111	1011	I
CICLO : NOP		A	10110	xxxx	xxxx	xxxx	xxxx	xxxx	
B CICLO	goto CICLO (PCx = A)	B	10011	xxxx	0000	0000	0000	1010	J

Tabla 2 Programa de la serie de Fibonacci.

En este ejemplo, las instrucciones de salto condicional utilizan SALTOS RELATIVOS. En este tipo de saltos la dirección se obtiene sumando un número al valor actual del contador de programa. Esto se hace porque en el formato de instrucción se tienen solo 12 bits (bit 11 al bit 0) para poner el valor de la dirección de memoria. Con estos 12 bits solo se pueden direccionar 2^{12} = 4192 = 4k, pero el mapa de memoria es de 2^{16} = 65535 = 64k, así que usamos el **principio de localidad.** Este principio me dice que la dirección de la siguiente instrucción a ejecutar esta muy cerca de la dirección actual del Contador de Programa (PC). Entonces en la instrucción de salto condicional podemos saltar en una ventana de 4K alrededor del valor actual del PC.

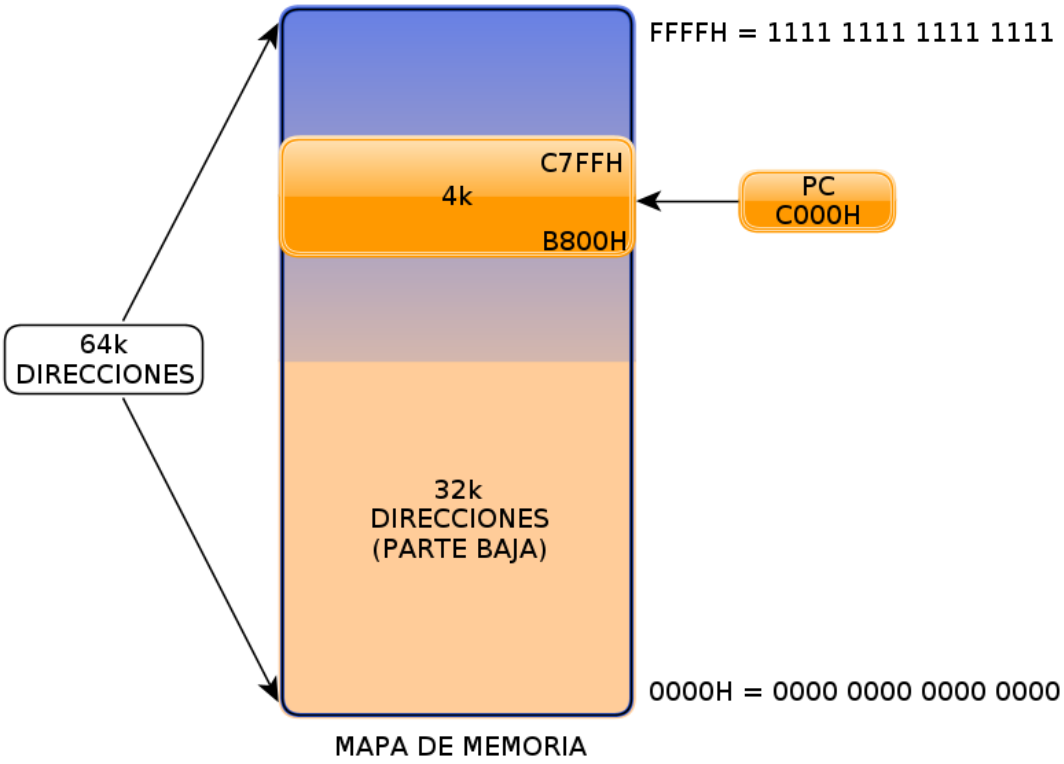


Figura 1: Ejemplo de salto relativo