Programas con arreglos usando el lenguaje ensamblador.

Ejemplo 1. Vamos a realizar un programa inicialice con ceros un arreglo de 8 elementos declarado en memoria a partir de la dirección 10. Observemos el diagrama de flujo de la ilustración 1.

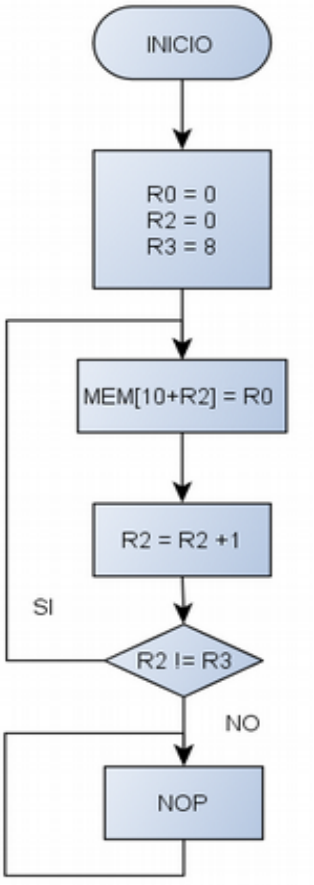

Ilustración 1 Algoritmo de inicialización de arreglo.

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

Instrucciones	Significado	Dir	24…20	19…16	15...12	11...8	7...4	3...0	T
LI R0, #0	R0 = 0	0	00001	0000	0000	0000	0000	0000	I
LI R2, #0	R2 = 0	1	00001	0010	0000	0000	0000	0000	I
LI R3, #8	R3 = 8	2	00001	0011	0000	0000	0000	1000	I
INI: SW R0, 10(R2)	MEM[10+R2] = R0	3	00100	0000	0010	0000	0000	1010	I
ADDI R2, R2, #1	R2 = R2 + 1	4	00101	0010	0010	0000	0000	0001	I
BNEI R2, R3, INI	If(R2 != R3) goto INI (PCx = PCx + -2, = 5 + -2 = 3)	5	01110	0010	0011	1111	1111	1110	J
CICLO: NOP		6	10110	xxxx	xxxx	xxxx	xxxx	xxxx	
B CICLO	goto CICLO PCx = 6	7	10011	xxxx	0000	000	0000	0110	J

Tabla 1 Programa de inicialización de arreglo.

Ejemplo 2. Vamos a realizar un programa que calcule el promedio de un arreglo de 4 elementos declarado en memoria a partir de la dirección 10, el promedio se guardará en la dirección de memoria 20. Observemos el diagrama de flujo de la ilustración 2.

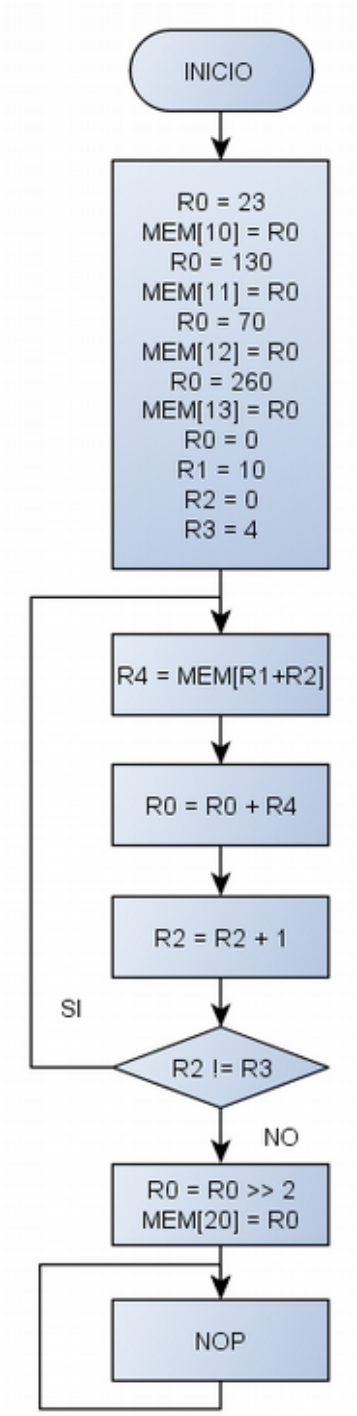

Ilustración 2 Algoritmo para obtener el promedio de los datos de un arreglo.

El promedio se realiza utilizando una operación de corrimiento a la derecha de 2 bits sobre el registro $R0$. Cuando se realiza el corrimiento a la derecha de “n” bits en el contenido de algún registro R, equivale a una división dada por:

$$R = \frac{R}{2^n}$$

Para un corrimiento de 2 bits, esto equivale a dividir el valor contenido en $R0$ por 4. 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, #23	R0 = 23	0	00001	0000	0000	0000	0001	0111	I
SWI R0, 10	MEM[10] = R0	1	00011	0000	0000	0000	0000	1010	I
LI R0, #130	R0 = 130	2	00001	0000	0000	0000	1000	0010	I
SWI R0, 11	MEM[11] = R0	3	00011	0000	0000	0000	0000	1011	I
LI R0, #70	R0 = 70	4	00001	0000	0000	0000	0100	0110	I
SWI R0, 12	MEM[12] = R0	5	00011	0000	0000	0000	0000	1100	I
LI R0, #260	R0 = 260	6	00001	0000	0000	0001	0000	0100	I
SWI R0, 13	MEM[13] = R0	7	00011	0000	0000	0000	0000	1101	I
LI R0, #0	R0 = 0	8	00001	0000	0000	0000	0000	0000	I
LI R1, #10	R1 = 10	9	00001	0001	0000	0000	0000	1010	I
LI R2, #0	R2 = 0	A	00001	0010	0000	0000	0000	0000	I
LI R3, #4	R3 = 4	B	00001	0011	0000	0000	0000	0100	I
PROM: LW R4, R1, R2	R4 = MEM[R1+R2]	C	00000	0100	0001	0010	xxxx	1011	R
ADD R0, R0, R4	R0 = R0 + R4	D	00000	0000	0000	0100	xxxx	0000	R
ADDI R2, R2, #1	R2 = R2 + 1	E	00101	0010	0010	0000	0000	0001	I
BNEI R2, R3, PROM	If(R2 != R3) goto PROM (PCx = PCx + -3, = F + -3 = C)	F	01110	0010	0011	1111	1111	1101	J
SRL R0, R0, #2	R0 = R0 >> 2	10	00000	0000	0000	xxxx	0010	1010	R
SWI R0, 20	MEM[20] = R0	11	00011	0000	0000	0000	0001	0100	I
CICLO: NOP		12	10110	xxxx	xxxx	xxxx	xxxx	xxxx	
B CICLO	goto CICLO PCx = 12	13	10011	xxxx	0000	000	0001	0010	J

Tabla 2 Programa de cálculo de promedio.

Ejemplo 3. Vamos a realizar un programa que encuentre el número mayor en un arreglo de 4 elementos declarado en memoria a partir de la dirección 10, el número mayor se guardará en la dirección de memoria 20. Observemos el diagrama de flujo de la ilustración 3.

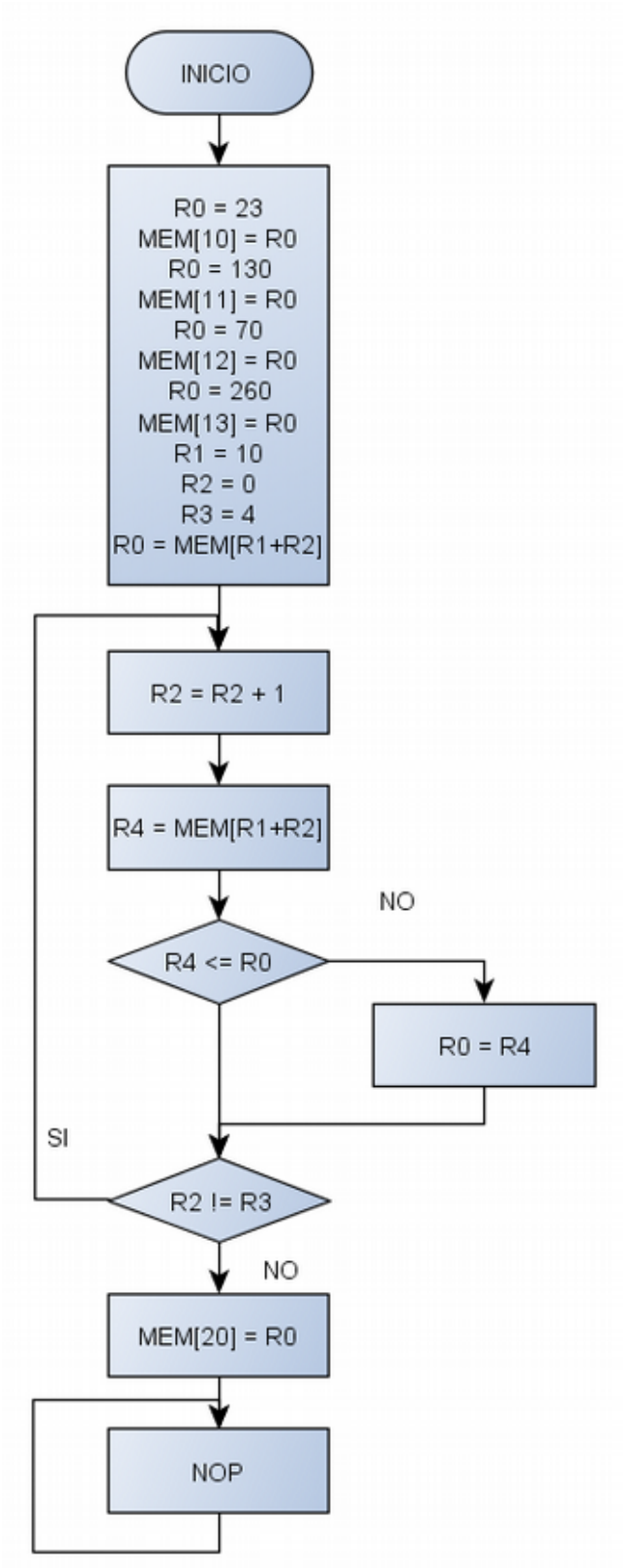

Ilustración 3 Algoritmo para obtener el número mayor de un arreglo.

Instrucciones	Significado	Dir	24…20	19…16	15...12	11...8	7...4	3...0	T
LI R0, #23	R0 = 23	0	00001	0000	0000	0000	0001	0111	I
SWI R0, 10	MEM[10] = R0	1	00011	0000	0000	0000	0000	1010	I
LI R0, #130	R0 = 130	2	00001	0000	0000	0000	1000	0010	I
SWI R0, 11	MEM[11] = R0	3	00011	0000	0000	0000	0000	1011	I
LI R0, #70	R0 = 70	4	00001	0000	0000	0000	0100	0110	I
SWI R0, 12	MEM[12] = R0	5	00011	0000	0000	0000	0000	1100	I
LI R0, #260	R0 = 260	6	00001	0000	0000	0001	0000	0100	I
SWI R0, 13	MEM[13] = R0	7	00011	0000	0000	0000	0000	1101	I
LI R1, #10	R1 = 10	8	00001	0001	0000	0000	0000	1010	I
LI R2, #0	R2 = 0	9	00001	0010	0000	0000	0000	0000	I
LI R3, #4	R3 = 4	A	00001	0011	0000	0000	0000	0100	I
LW R0, R1, R2	R0 = MEM[R1+R2]	B	00000	0000	0001	0010	xxxx	1011	R
MAYOR: ADDI R2, R2, #1	 R2 = R2 + 1	C	00101	0010	0010	0000	0000	0001	I
LW R4, R1, R2	R4 = MEM[R1+R2]	D	00000	0100	0001	0010	xxxx	1011	R
BLETI R4, R0, R0MAY	If(R4 <= R0) goto R0MAY (PCx = PCx + 2, = E + 2 = 10)	E	10000	0100	0000	0000	0000	0010	J
ADDI R0, R4, #0	R0 = R4 + 0	F	00101	0000	0100	0000	0000	0000	I
R0MAY: BNEI R2, R3, MAYOR	If(R2 != R3) goto MAYOR (PCx = PCx + -4, = 10 + -4 = C)	10	01110	0010	0011	1111	1111	1100	J
SWI R0, 20	MEM[20] = R0	11	00011	0000	0000	0000	0001	0100	I
CICLO: NOP		12	10110	xxxx	xxxx	xxxx	xxxx	xxxx	
B CICLO	goto CICLO PCx = 12	13	10011	xxxx	0000	000	0001	0010	J

Tabla 3 Programa para obtener el número mayor de un arreglo.