

### PRÁCTICO 1 - Assembler de LEGv8

#### Ejercicio 1:

Dadas las siguientes sentencias en "C":

- a)  $f = g + h + i + j$ ;
- b)  $f = g + (h + 5)$ ;
- c)  $f = (g + h) - (g + h)$ ;

Escribir la secuencia **mínima** de código assembler LEGv8 asumiendo que f, g, h, i y j se asignan en los registros X0, X1, X2, X3 y X4 respectivamente.

#### Ejercicio 2:

Dadas las siguientes sentencias en assembler LEGv8:

- a) SUB X1, XZR, X1  
ADD X0, X1, X2
- b) ADDI X2, X0, #1  
SUB X0, X1, X2

Escribir la secuencia mínima de código "C" asumiendo que los registros X0, X1, y X2 contienen las variables f, g, y h respectivamente.

#### Ejercicio 3:

Dadas las siguientes sentencias en "C":

- a)  $f = -g - A[4]$ ;
- b)  $B[8] = A[i - j]$ ;

Escribir la secuencia **mínima** de código assembler LEGv8 asumiendo que f, g, i y j se asignan en los registros X0, X1, X2 y X3 respectivamente, y que la dirección base de los arreglos A y B se almacenan en los registros X6 y X7 respectivamente.

#### Ejercicio 4:

Dadas las siguientes sentencias en assembler LEGv8:

```
ADDI X9, X6, #8
ADD X10, X6, XZR
STUR X10, [X9, #0]
LDUR X9, [X9, #0]
ADD X0, X9, X10
```

4.1) Asumiendo que los registros X0, X6 contienen las variables f y A (dirección base del arreglo), escribir la secuencia mínima de código "C" que representa.

4.2) Asumiendo que los registros X0, X6 contienen los valores 0xA, 0x100, y que la memoria contiene los valores de la tabla, encuentre el valor del registro X0 al finalizar el código assembler.

Dirección	Valor
0x100	0x64
0x108	0xC8
0x110	0x12C

### Ejercicio 5:

Utilizar MOVZ, MOVK para cargar los registros:

5.1) {X0 = 0x1234000000000000}

5.2) {X1 = 0xBBB0000000000AAA}

5.3) {X2 = 0xA0A0B1B10000C2C2}

5.4) {X3 = 0x0123456789ABCDEF}

	Signed numbers		Unsigned numbers	
Comparison	Instruction	CC Test	Instruction	CC Test
=	B.EQ	Z=1	B.EQ	Z=1
≠	B.NE	Z=0	B.NE	Z=0
<	B.LT	N!=V	B.LO	C=0
≤	B.LE	~(Z=0 & N=V)	B.LS	~(Z=0 & C=1)
>	B.GT	(Z=0 & N=V)	B.HI	(Z=0 & C=1)
≥	B.GE	N=V	B.HS	C=1

Signed and Unsigned numbers	
Instruction	CC Test
Branch on minus (B.MI)	N= 1
Branch on plus (B.PL)	N= 0
Branch on overflow set (B.VS)	V= 1
Branch on overflow clear (B.VC)	V= 0

- Negative (N): the result that set the condition code had a 1 in the most significant bit.
- Zero (Z): the result that set the condition code was 0.
- Overflow (V): the result that set the condition code overflowed. Signed numbers.
- Carry (C): the result that set the condition code had a carry out of the most significant bit or a borrow into the most significant bit. Unsigned numbers.

Operation	Operand A	Operand B	Result indicating overflow
A + B	≥ 0	≥ 0	< 0
A + B	< 0	< 0	≥ 0
A - B	≥ 0	< 0	< 0
A - B	< 0	≥ 0	≥ 0

### Ejercicio 6:

Para estos dos programas con entrada y salida en X0, decir que función realizan.

<pre> SUBIS X0, X0, #0 B.LT else B done else: SUB X0, XZR, X0 done: </pre>	<pre> MOV X9, X0 MOV X0, XZR loop: ADD X0, X0, X9       SUBI X9, X9, #1       CBNZ X9, loop done: </pre>
--	--

**Ejercicio 7:**

Dado el siguiente programa LEGv8, dar el valor final de X10, cuyo valor inicial es: X10=0x0000000000000001.

```
        SUBIS XZR, X9, #0
        B.GE else
        B done
else: ORRI X10, XZR, #2
done:
```

**7.1)** Dado que inicialmente {X9=0x00000000000101000}.

**7.2)** Dado que inicialmente {X9=0x800000000001000}.

**Ejercicio 8:**

Dado el siguiente programa “C” y la asignación  $i, j, k, N \leftrightarrow X0, X1, X2, X9$ , escribir el programa LEGv8 que lo implementa. Notar que como se usa el operador `||` la [evaluación es por cortocircuito](#). **Opcional:** hacerlo con el operador `|` que no está cortocircuitado.

```
long i,j,k;
if (i==N || j==N) {
    ++k;
} else {
    ++i; ++j;
}
```

**Ejercicio 9:**

Dados los siguientes programas en LEGv8:

<pre>        ADD X10, XZR, XZR loop: LDUR X1, [X0,#0]         ADD X2, X2, X1         ADDI X0, X0, #8         ADDI X10, X10, #1         CMPI X10, #100         B.LT loop</pre>	<pre>        ADDI X10, XZR, #50 loop: LDUR X1, [X0,#0]         ADD X2, X2, X1         LDUR X1, [X0,#8]         ADD X2, X2, X1         ADDI X0, X0, #16         SUBI X10, X10, #1         CBNZ X10, loop</pre>
---	---

**9.1)** ¿Cuántas instrucciones LEGv8 ejecuta cada uno?

**9.2)** Reescribir en “C” dada la asignación  $X10, X1, X2, X0 \leftrightarrow i, a, result, MemArray$ .

**9.3) Opcional:** optimizar los códigos assembler para reducir el número de instrucciones LEGv8 ejecutadas.

### Ejercicio 10:

Traducir el siguiente programa "C" a LEGv8. La asignación de variables a registros X0, X1, X2, X3, X9  $\leftrightarrow$  A, s, i, j, N. Notar que en "C" los arreglos bidimensionales se representan en memoria usando un **orden por filas**, es decir  $\&A[i][j] = A + 8*(i*N+j)$ .

```
#define N (1<<10)
long A[N][N], s, i, j;
s=0;
for (i=0; i<N; ++i)
    for (j=0; j<N; ++j)
        s += A[i][j];
```

**Ejemplo de orden por filas:**

A[2][3] =

1	7	2
44	3	21

A ->

1	7	2	44	3	21
---	---	---	----	---	----