ARQUITECTURA DE COMPUTADORAS I

Guía 2 de Ejercicios: Assembler del ARMv8

Ejercicio 1.

Para el siguiente enunciado en C, escriba el código ensamblador LEGv8 correspondiente. Suponga que las variables f, g y h ya se han colocado en los registros X0, X1 y X2, respectivamente. Utilice un número mínimo de instrucciones ensambladoras LEGv8.

```
f = g + (h - 5);
```

Ejercicio 2.

Para el siguiente enunciado en C, escriba el código ensamblador LEGv8 correspondiente.
Variables: a=X1, b=X2, c=X3, f=X0

```
int f = a + b + c;
```

Ejercicio 3.

Para el siguiente enunciado en C, escriba el código ensamblador LEGv8 correspondiente.
Variables: x=X0, y=X1

```
if (x == 0) {
        y = y + 1;
        }
else {
        y = y - 1;
     }
```

Ejercicio 4.

Para el siguiente enunciado en C, escriba el código ensamblador LEGv8 correspondiente.
Variables: n=X0, sum=X1, val=X2

```
while (n > 0) {
                  sum = sum + val;
                  n--;
               }
```

Ejercicio 5.

Para el siguiente enunciado en C, escriba el código ensamblador LEGv8 correspondiente. Variables: i=X0, N=X1, a=X2, b=X3

```
for (i = N; i > 0; i--)
    {
        a = a + b;
    }
```

Ejercicio 6.

¿Es posible escribir una única instrucción en C equivalente a las dos instrucciones de ensamblador LEGv8 que se muestran a continuación? Suponiendo que las asignaciones son las siguientes: $X0 \rightarrow f$, $X1 \rightarrow g$, $X2 \rightarrow h$, $X3 \rightarrow i$.

```
ADD X0, X1, X2
ADD X0, X3, X0
```

Ejercicio 7.

Para la siguiente instrucción en C, escriba el código ensamblador LEGv8 correspondiente. Suponga que las variables i,j están asignadas a los registros X3 y X4, respectivamente. Suponga que las direcciones base de los arreglos A y B están en los registros X6 y X7, respectivamente.

```
B[8] = A[i-j];
```

Ejercicio 8.

Interprete y traduzca este código LEGv8 a C. Variables: a=X0, b=X1, c=X2, d=X3, e=X4

```
    ADD X5, X0, X1
    SUB X6, X3, X4
    MUL X7, X5, X6
    STUR X7, [X2, #0]
```

Ejercicio 9.

Dado el siguiente código LEGv8, expréselo en C con las variables f=X0, g=X1.

```
    CBZ X0, End
    ADD X1, X1, X0
    End:
```

Ejercicio 10.

Interprete el código LEGv8 y traduzca a C, variables x=X0, y=X1, n=X2

```
    MOVZ X3, #0
    Loop:
    CBZ X2, End
    ADD X3, X3, X1
    SUBI X2, X2, #1
    B Loop
    End:
    STUR X3, [X0, #0]
```

Ejercicio 11.

Explique linea por linea lo que hace cada instrucción en el siguiente fragmento de codigo ensamblador:

```
.section .data
numeros: .word 2, 5, 8, 7, 4

.section .text
.global _start
_start:


LDR R0, =numeros
MOV R1, #0
MOV R2, #0
bucle:
CMP R2, #5
BEQ fin
LDR R3, [R0, R2, LSL #2]
ANDS R4, R3, #1
BNE siguiente
ADD R1, R1, R3
siguiente:
ADD R2, R2, #1
B bucle
fin:
    STR R1, [R0, #20]
```

Ejercicio 12.

Explique linea por linea lo que hace cada instrucción en el siguiente fragmento de codigo ensamblador:

```
.section .data
valores: .word 4, 10, 3, 8, 2, 15
.section .text
.global _start

_start:
LDR R0, =valores
LDR R1, [R0]
MOV R2, #1

bucle:
CMP R2, #6
BEQ fin
LDR R3, [R0, R2, LSL #2]
CMP R3, R1
BLE siguiente
MOV R1, R3
siguiente:
ADD R2, R2, #1
B bucle
fin:
STR R1, [R0, #24]
```

Ejercicio 13.

En un sistema embebido que monitorea sensores, se reciben lecturas que se almacenan en un arreglo. Para filtrar valores anómalos, se necesita contar cuántos valores superan un umbral determinado y almacenar ese conteo para actuar en consecuencia (p.ej. activar una alarma). Proponer una lógica realizar esta tarea y codificarla en ensamblador ARMv8. Comprobar la solución propuesta usando QEMU y gdb.

Ejercicio 14.

En aplicaciones críticas, a menudo es necesario crear copias de datos para respaldo o procesamiento paralelo. Implementar un código que copie todos los valores de un arreglo origen a otro destino. Codificar en ensamblador ARMv8 y comprobar la solución propuesta usando QEMU y gdb.

Ejercicio 15.

En comunicaciones digitales, los datos recibidos pueden tener errores. Para un sencillo chequeo, se necesita contar cuántos valores "0"hay en un tramo de datos, para luego evaluar la calidad de transmisión. Implementar una lógica realizar esta tarea y codificarla en ensamblador ARMv8. Comprobar la solución propuesta usando QEMU y gdb.