Ejercicio 1:

Determinar cuáles de las siguientes instrucciones pueden ser ensambladas en LegV8. Justificar las respuestas negativas.

Instrucción	Si/No	Justificación
B 0	Si	Es valido el 0
LDUR X1, [XZR, X3]	No	En [.] admite un registro y un inmediato normal
MOVK X15, #0xCAFE, LSL#8	No	LSL solo puede ser 00 16 32 48
ORRI XZR, X1, #-16	No	No admite negativos
LSL X30, X30, #33	Si	X30 existe y LSL [00-63]

Ejercicio 2:

Usando el siguiente ejemplo, escriba un programa en LegV8 que divida X0 por X1 y salte dependiendo si el resultado es mayor o igual a 0.33. Se pueden usar todas las instrucciones de LegV8 excepto las de punto flotante. Debe resolverse en 5 instrucciones o menos.

$\frac{X0}{X1} \geq \frac{1}{3} \iff X0*3 \geq X1$

```
MOVZ X0, #1, LSL#0        //X0 = 1
MOVZ X1, #3, LSL#0        //X1 = 3
LSL X2, X0, #1            // X0*2
Add X0, X0, X2            // X0*3
CMP X0, X1                //
B.GE end                  // Saltar si ~~X0/X1 ≥ 0,33~~   X0*3 ≥ X1
```

Ejercicio 3:

Dada la siguiente sección de un programa en assembler LegV8, el registro X1 contiene la dirección base de un arreglo A, mientras que X0 contiene el tamaño de dicho arreglo. Asuma que los registros y la memoria contienen los valores mostrados en la tabla al inicio de la ejecución de dicha sección.

```
        ADDI X10, XZR, #0      // X10 = 0
        SUBI X0, X0, #1        // X0 = 0*3
Loop:   CMP X10, X0
        B.GE LOOP_END // GE: mayor o igual   // X10 ≥ X0 -> LOOP_END   (Eso es un error)
Proc:   LSL X11, X10, #3       // X11 = X10*8
        ADD X11, X1, X11       // X11 = X1 + X11 (A+offset)
        LDUR X12, [X11, #0]    // X12 = A+offset
        LDUR X13[X11, #8]      // X13 = Elemento siguiente
        CPM X13, X12
        B.GT NO_XCHG //    GT: Mayor   // X13 > X12 => NO_XCHG
        STUR X13 [X11, #0]
        STUR X12, [X11, #0]    (posible error)
NO_XCHG: ADDI X10, X10, #1     // X10 ++
        B LOOP                 // Repite
LOOP_END …
```

Memoria		
Dirección	Contenido (antes)	Contenido (después)
0x10000100	122	-30
0x10000108	-30	-30
0x10000110	70	10
0x10000118	10	10
0x10000120	45	45
0x10000128	200	200

Registros	
X0	0x00000004
X1	0x10000100

Responder:

a) ¿Cómo queda el contenido de todas las posiciones de memoria mostradas en la tabla al finalizar la sección del programa? Responde completando las columnas en blanco de la tabla de memoria.

b) La instrucción contenida en la línea con el label PROC se ejecuta ___3___ veces.

-Posiblemente sea 4 si es GT en vez de GE

Ejercicio 4:

Considere el segmento de memoria que se muestra en la primera columna de la forma *dirección: contenido* que contiene codificado un programa en ISA LegV8. Parte del programa desensamblado se presenta en la 2da columna.

(label:)	Dirección: contenido	Desensamblado
	0x00000100: 0x910013E0	ADDI X0, XZR, #4
(loop:)	0x00000104: 0xB89103E9	LDURSW X9 [XZR, #0x110]
	0x00000108: 91001529	ADDI X9, X9, #5
	0x0000010C: B81103E9	STURW X9, [XZR, 0x110]
	0x00000110: 13FFFFFD	B ~~loop~~ +2
	0x00000114: 8B1F03E0	ADD X0, XZR, XZR
	0x00000118: D3600400	LSL X0, X0, #1

a) Completar las instrucciones restantes.

b) ¿Cuántas veces se ejecuta la instrucción con label: loop? 1

c) ¿Cuál es el valor de X0 luego de la ejecución del segmento? 4*2 = 8

a) 0xB89103E9

S C 4 DT Address OP Rn Rt

= 1011 1000 1001 0001 0000 0011 1110 1001

OPcode: 0x5C4 LDURSW tipo D (11 bits)

Rt = X9

Rn = X31 = XZR

DT_Addres = 1 0001 0000 = 0x110

6 9 B Rm Shamt Rn Rd

0xD3600400 = 110 0011 0110 0000 0000 0100 0000 0000

OPcode 0x69B LSL tipo R (11 bits)

Rd = X0

Rn = X0 => LSL X0, X0, #1

Shamt = 1

X9 = 0x 13 FFFFFD +5

0x 5

0x 14000002 → desensamblar

opcode

0001 0100 0000 0000 0000 0000 0000 0010

Br address

Opcode Branch tipo B

BR Adress = 2

Ejercicio 5:

Considerar que el procesador está ejecutando la instrucción de LegV8: 0xF801816A y el contenido de los registros es:

X10 = 0x20, X11 = 0x23, X12 = 0x54. X13 = 0x00, PC = 0x104

1111 1000 0000 0001 1000 0001 0110 1010

opcode 0x7C0 = STUR

Rt = X10

Rn = X11

DT = 24 = 0x18

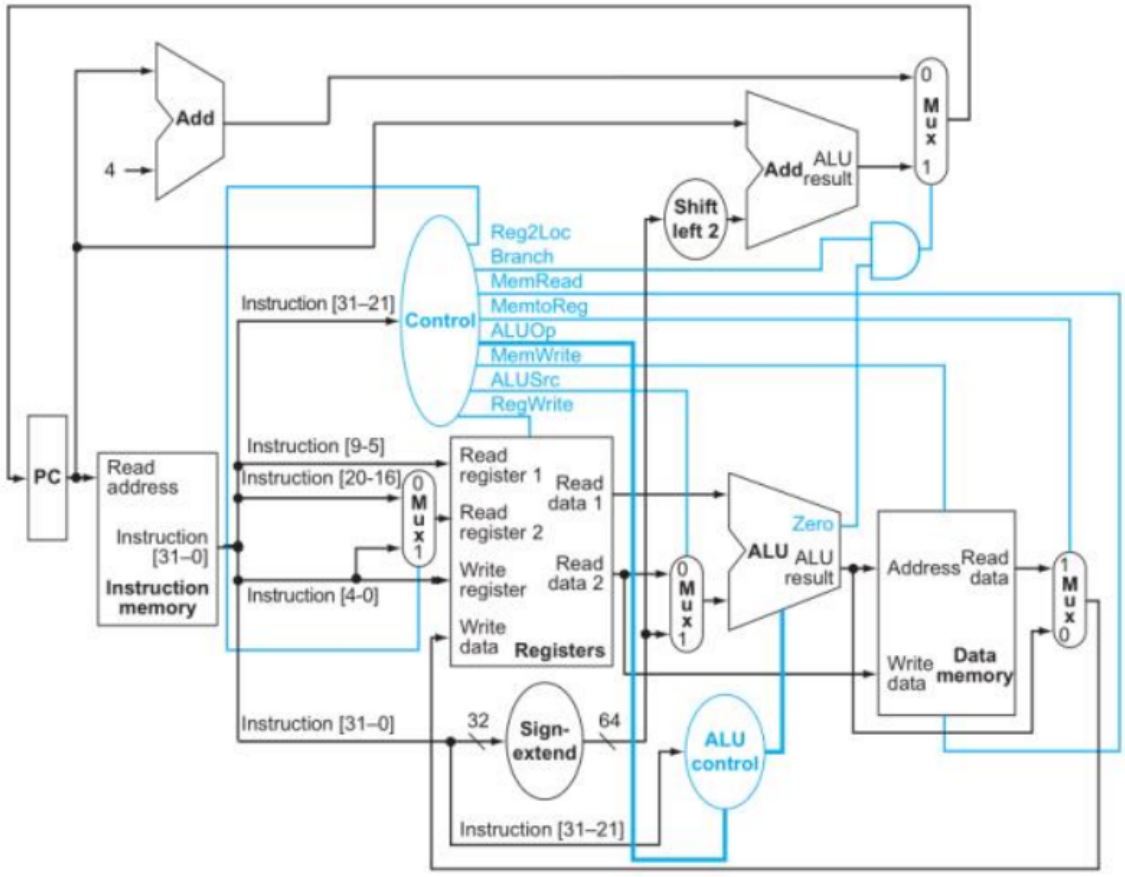
STUR X10, [X11, 0x18]

a) Desensamblar la instrucción.

b) ¿Qué operación realiza la ALU? suma: Reg[X11]+0x18

c) Completar el estado de las señales de control.

Reg2Log	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch
1	1	X	0	0	1	0



Completar:

Señal	E/S	N° de bits	Valor
Register, Read data 1	S	5	0x23
Register, Read register 2	E	5	Reg[x10]
Data Memory, address	E	64	0x23 + 0x18 = 0x3B
Register, write register	E	64	X Desconozco
Entrada del pc	E	64	0x108
Add, ALUResult	S	64	0x164 = 0x104 + 0x18 * 0x2

0x18 * 2 = 0001 1000 * 10 = 0110 0000 = 0x60

```
  0x104
+ 0x 60
--------
  0x164
```