Ejercicio 1

Escribe un skip, **UNA** instrucción que no hace nada, con cada nemónico. **Si no se puede, poner raya**.

ADDI		LDURH	
ADDS		LSL	
ANDI		MOVK	
ANDS		ORR	
B.cond		STUR	
BR		STURH	
CBZ		SUB	
EORI		SUBIS	

Ejercicio 2

a) Se aplica una técnica de *loop unrolling* en un programa para que duplique los elementos de un arreglo, enteros de 64 bits.

```
    ADD X0,XZR,XZR
    ADDI X1,XZR,#0x32
    ADD X2,XZR,XZR
L1: LDUR X8,[X0,#0x50]
    ADD X8,X8,X8
    STUR X8,[X0,#0x50]
    ADDI X0,X0,#0x8
    SUBI X1,X1,#0x1
    CBNZ X1,L1
    RET
```

```
    ADD X0,XZR,XZR
    ADDI X1,XZR,#0x32
    ADD X2,XZR,XZR
L1: LDUR X8,[X0,#0x50]
    ADD X8,X8,X8
    STUR X8,[X0,#0x50]
    LDUR X8,[X0,#0x58]
    ADD X8,X8,X8
    STUR X8,[X0,#0x58]
    ADDI X0,X0,0x10
    SUBI X1,X1,#0x2
    CBNZ X1,L1
    RET
```

	Ocupación de memoria	**Instrucciones ejecutadas**
ProgIzq		
ProgDer		

b) Este es un fragmento de *ProgIzq*. Escribe 2 permutaciones equivalentes y 2 no equivalentes. Hay 5!=125 permutaciones posibles pero no todas hacen lo mismo que el código original.

	Equivalente 1	**Equivalente 2**	**No equivalente 1**	**No equivalente 2**

`LDUR X8,[X0,#0x50]`
`ADD X8,X8,X8`
`STUR X8,[X0,#0x50]`
`ADDI X0,X0,#0x8`
`SUBI X1,X1,#0x1`				

Ejercicio 3

Dado el siguiente programa en Assembler LEGv8 y el estado inicial de la memoria:

```
      MOVZ X0,#0x100,LSL#0
      LSL X1,X0,#1
      ORRI X2,X1,#0x100
loop: LDUR X3,[X0,#0]
      ADDI X3,X3,#1
      CBZ X3,end
      SUBI X3,X3,#1
      LSL X3,X3,#3
      ADD X3,X1,X3
      LDUR X3,[X3,#0]
      STUR X3,[X2,#0]
      ADDI X0,X0,#8
      ADDI X2,X2,#8
      B loop
end:
```

```
Dirección: contenido
...
0x100: 0x1
0x108: 0x2
0x110: 0x0
0x118: 0xFFFFFFFFFFFFFFFF
...
0x200: 0xCAFECAFE
0x208: 0xCOCACO1A
0x210: 0xDEADBEEF
...
0x300: 0x0
0x308: 0x0
0x310: 0x0
...
```

a) Mostrar el **valor final de la memoria** escribiendo al costado solo las posiciones que cambian.
b) Explicar **en una línea** que hace el código.

Ejercicio 4

Desensamblar el programa que se volcó directamente desde la memoria RAM, byte a byte.

```
08 00 45 F8 08 01 08 8B 08 00 05 F8 00 20 00 91 21 04 00 D1
```


Ejercicio 5

Ensamblar estos dos programas:

	Ensamblador	Código máquina

Delay loop 1:	`.org 0x2000`
	`L0: SUBI X0,X0,#1` __
	`CBNZ X0,L0` __
		`# Ensambla en 0x2000`
		`0x __ __ __ __ __ __ __ __`
		`0x __ __ __ __ __ __ __ __`

Delay loop 2:	`.org 0x4000`
	`L1: SUBI X0,X0,#1` __
	`CBNZ X0,L1` __
		`# Ensambla en 0x4000`
		`0x __ __ __ __ __ __ __ __`
		`0x __ __ __ __ __ __ __ __`

Ejercicio 6

Dada la compilación con la siguiente relación entre variables y registros:

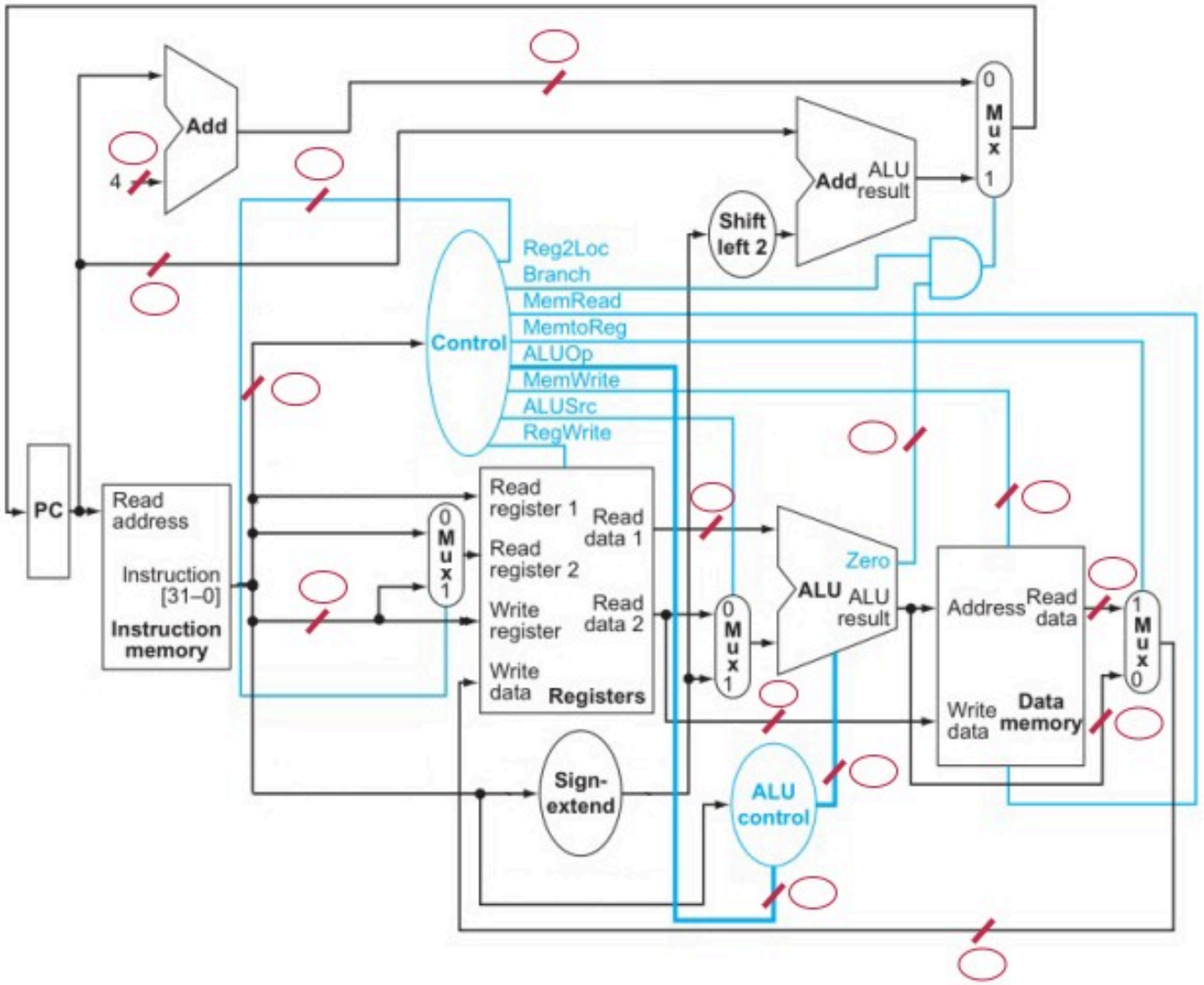
- `x ↔ X0`
- `score ↔ X1`
- `speedx ↔ X2`

`if (x==0)`
`  score++;`
`else`
`  speedx = -speedx`
	`  CBZ X0,L1`
	`  SUBI X8,XZR,#1`
	`  ADD X2,X2,X8`
	`  ADDI X2,X2,#1`
	`L1: ADDI X1,X1,#1`

Explicar en una línea si la compilación es correcta o no y por qué.

Ejercicio 7

Marcar el ancho en bits de las 15 líneas de datos marcadas.



Ejercicio 8

Un opcode genera una **instrucción no documentada** que hace que Control tome estos valores

Reg2Loc	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALU
0	0	1	1	1	0	0	ADD

Describir qué operación realiza: _____

Invente su nemónico: _____