Ejercicio 1

Muchos ISAs utilizan un campo como origen en algunas instrucciones y como destino en otras, **forzando el agregar hardware extra en sitios potencialmente críticos** (en tiempo) para seleccionar el campo correcto.

El problema con esto es que deberíamos tener dos formas de decodificar las mismas operaciones.

Una para cada posición distinta de los registros, forzando agregar hardware extra en el decoder.

Ejercicio 2

```
#este es un comentario
 .text #comenzamos con esta directiva

#completar

.globl main

.globl desplazar

main:   add x11, x0, x0

        add x11, x10, x0    #acumulador sumas desplazamientos

desplazar:   srli x10, x10, 1

             add x11, x11, x10

             bne x0, x10, -8 #el registro x0 siempre tiene
                                                    el valor 0

#finalizamos con estas líneas
 fin : addi x17 , x0 , 0x11
 ecall
```

posiciones de memoria de las instrucciones:

add 0
add 4
srli 8
add 12
bne 16

registros utilizados:

x10, x11 son únicamente los mencionados en el enunciado.

decodificación de las instrucciones:

sw x10, 20(x10) **0000000 01010 01010 010 10100 0100011**

lw x11, 20(x10) **000000010100 01010 010 01011 0000011**

srli x10, x10, 1 **0000000 00001 01010 101 01010 0010011**

add x11, x11, x10 **0000000 01010 01011 000 01011 0110011**

bne x0, x10, -8 **1111111 01010 00000 001 10001 1100011**

Ejercicio 3

¿Cuáles son los objetivos no funcionales que guían el diseño del set de instrucciones de RISC-V? ¿Cómo impactan en su diseño?

Ejercicio 4

¿Qué problema puede introducir habilitar interrupciones cuando se atiende una interrupción en OrgaSmallI?

¿Qué pasa cuando interrumpen al CPU?

En el caso de un procesador OrgaSmallI, si el dispositivo de E/S activa la señal de interrupción y el flag I vale 1, termina de ejecutar la instrucción en curso y realiza atómicamente los siguientes pasos:

Coloca `[0xFF] = PC`

Coloca `I=0` para evitar que el procesador vuelva a interrumpirse

Coloca `PC = [0x00]`

Activa la señal INTA para indicarle al dispositivo que atenderá su pedido

Luego, comienza a ejecutarse la rutina de atención de la interrupción propiamente dicha.

Guardando el PC de la instrucción siguiente a ejecutar luego de la rai en la posición de memoria 0xFF y lo que hay en la posición de memoria 0x00 en el PC para que se ejecute la rai.

Ahora suponiendo que no se coloca `I=0`, en el fetch sólo se ejecutarán los pasos de:

```
Coloca [0xFF] = PC   # guarda la siguiente instrucción a ejecutar
Coloca PC = [0x00]   # ejecuta la rai
```

Activa la señal INTA para indicarle al dispositivo que atenderá su pedido

Luego, comienza a ejecutarse la rutina de atención de la interrupción int2 propiamente dicha.

Si mientras se ejecuta la rai se recibe una interrupción int2, se ejecutarán los siguientes pasos:

```
Coloca [0xFF] = PC   # el PC está en 0x00 porque se está ejecutando una rai
Coloca PC = [0x00]   # ejecuta la rai
```

Activa la señal INTA para indicarle al dispositivo que atenderá su pedido

Luego, comienza a ejecutarse la rutina de atención de la interrupción int2 propiamente dicha.

Por lo tanto, la rutina de atención a la interrupción anterior no terminaría nunca de ejecutarse porque en la dirección en donde se guarda la siguiente instrucción a ejecutar se pisó a 0x00 y no se podría salir de la rai.

Ejercicio 5

Escriba el microprograma que permita sumar el contenido del valor que se encuentra en la dirección referida por el registro indicado en el **operando X del decoder** consigo mismo.

#código de operación de la microinstrucción

```
xxxxx: ; SUMAR R, op

     reset_microOp
```