UNSAM
UNIVERSIDAD NACIONAL DE SAN MARTÍN

ESCUELA DE CIENCIA Y TECNOLOGÍA

TRABAJO PRÁCTICO REGULARIZADOR

GUÍA 1.

ELECTRÓNICA DIGITAL 2

2do Cuatrimestre 2022

Profesores:
Ingeniero Nicolás Alvarez.
Ingeniero Miguel Angel Sagreras.

Alumna:
Oriana Farfán.

1. Implementar en assembly un código que permita mostrar por los leds un valor de 8 bits. Comprobarlo en el simulador.

```
address 0

input s0, 0
constant leds_port, 0
output s0, leds_port
```

2. Implementar en assembly la suma de dos números de 4 bits. Comprobarlo en el simulador.

```
address 0

input s0, 0 ; se ingresa un valor a s0.
input s1, 1 ; se ingresa un valor a s1.
load s2, 00001111'b ; se crea una máscara.
and s0, s2 ; se pasa al valor s0 por la máscara.
and s1, s2 ; se pasa al valor s1 por la máscara.
add s0, s1 ; se suman s0 y s1.
```

3. Implementar en assembly la suma de dos números de 8 bits. Comprobarlo en el simulador.

```
address 0

input s0, 0 ; se ingresa un valor a s0.
input s1, 1 ; se ingresa un valor a s1.
add s0, s1 ; se suman s0 y s1.

jump 0
```

4. Implementar en assembly un desplazador a izquierda en función del estado de las entradas simuladas.

```
address 0

input s0, 0 ; se ingresa un valor a s0.
input s1, 1 ; se ingresa el número de veces que se quiere mover al primer número.
sub s1, 1 ; se resta 1 al valor en s1.
jump c, end_loop ; si el número fue 0 -> no se realiza movimientos y se va al final del
programa.
add s1, 1 ; se suma el valor en s1 para restablecer su valor.
loop:
```

```
sl0 s0 ; se mueve un espacio a la izquierda al valor en s0.
sub s1, 1 ; se resta en 1 al valor ingresado en s1.
jump nz, loop ; si el resultado no fue 0 -> se vuelve a la línea con el loop.
end_loop:
```

5. Implementar en assembly un multiplicador con operandos de 4 bits sin utilizar instrucciones de salto. Comprobarlo en el simulador.

```
address 0

input s0, 0
input s1, 1
load s2, 00001111'b ; se crea una máscara
and s0, s2 ; se pasa a s0 por la máscara y se obtiene un número de 4 bits.
and s1, s2 ; se pasa a s1 por la máscara y se obtiene un número de 4 bits.

load s3, s0
load s5, s0
load s7, s0
load s9, s0 ; se copia el valor que se encuentra en s0 en otras direcciones..

load s4, s1
load s6, s1
load s8, s1
load s10, s1 ; se copia el valor que se encuentra en s1 en otras direcciones.

slx s4
slx s4
slx s4
slx s4 ; se obtiene una extensión de lo que está en el primer bit del segundo número.
and s4, s2 ; se pasa a s4 por la máscara.
and s3, s4 ; se obtiene el primer número si el bit menos significativo del segundo es 1 o se obtiene 0 si el bit menos significativo del segundo número es un 0.

rr s6 ; se desplaza a la derecha un espacio al segundo número.
slx s6
slx s6
slx s6
slx s6 ;se obtiene una serie de repeticiones del segundo bit del segundo número.
and s6, s2 ; se pasa a s6 por la máscara.
and s5, s6 ; se obtiene el valor de s1 si el segundo bit del segundo número es un 1 o se obtiene 0 si el segundo bit del segundo número es un 0.
sl0 s5 ; se desplaza el número a la izquierda para poder sumarlo.
add s3, s5 ; se suman los resultados en s3.
```

```
rr s8
rr s8 ; se desplaza hacia la derecha dos espacios al segundo número.
slx s8
slx s8
slx s8
slx s8 ; se obtiene una serie de repeticiones del tercer bit del segundo número.
and s8, s2 ; se pasa a s8 por la máscara.
and s7, s8 ; se obtiene el valor de s1 si el tercer bit del segundo número es un 1 o se
obtiene 0 si el tercer bit del segundo número es un 0.
sl0 s7
sl0 s7 ; se desplaza el número hacia la izquierda dos veces para sumarlo después.
add s3, s7 ; se suman los resultados en s3.

rr s10
rr s10
rr s10 ;se desplaza hacia la derecha tres espacios al segundo número.
slx s10
slx s10
slx s10
slx s10 ; se obtiene una serie de repeticiones del cuarto bit del segundo número.
and s10, s2 ; se pasa a s10 por la máscara
and s9, s10 ; se obtiene el valor de s1 si el cuarto bit del segundo número es un 1 o se
obtiene 0 si el cuarto bit del segundo número es un 0.
sl0 s9
sl0 s9
sl0 s9 ; se desplaza el número a la izquierda dos veces para sumarlo después.
add s3, s9 ; se suma el resultado en s3 -> se obtiene el resultado de la multiplicación.
```

6. Implementar en assembly un multiplicador con operandos de 4 bits utilizando instrucciones de salto. Comprobarlo en el simulador.

```
address 0

input s0, 0
input s1, 1
load s2, 00001111'b
and s0, s2
and s1, s2
load s3, s0 ; se hace una copia en s3 de s0.
load s4, s1 ; se hace una copia en s4 de s1.
jump Z, if_zero ; en caso de que el segundo número es 0 se salta la línea con if_zero.

sub s1, 1 ; se resta 1 al segundo valor.
```

```
jump Z, end ; en caso de que el segundo número es 1 se salta al fin de programa y se
queda con el valor de s0.

loop:
add s0, s3 ; se suma
sub s1, 1 ; se resta 1 al segundo número ingresado.

jump NZ, loop ; si el segundo número no llega a 0 se vuelve a sumar.
jump Z, end ; si se llega a 0 se va al final de programa y se queda con el valor en s0.

if_zero:
sub s0, s0 ; si se multiplico por 0 -> s0 es 0.

end:
```

7. Implementar en assembly un contador de los 1’s presentes en un registro. Comprobarlo en el simulador.

```
address 0

input s0, 0
load s1, 00000000'b ; se inicializa un contador en s1.
load s2, 00000001'b ; máscara que solo deja pasar al bit menos significativo.

load s3, s0
load s4, s0
load s5, s0
load s6, s0
load s7, s0
load s8, s0
load s9, s0
load sa, s0 ; se copia el valor de s0 en varias direcciones

and s3, s2 ; aplico máscara a s3.
add s1, s3 ; se suma el valor al contador.

sr0 s4 ; se mueve los bits 1 espacio a la derecha.
and s4, s2 ; aplico máscara a s4.
add s1, s4 ; se suma el valor al contador.

sr0 s5
sr0 s5 ; se mueve los bits 2 espacios a la derecha.
and s5, s2 ; aplico máscara a s5.
add s1, s5 ; se suma el valor al contador.
```

```
sr0 s6
sr0 s6
sr0 s6 ; se mueve los bits 3 espacios a la derecha.
and s6, s2 ; aplico máscara a s6.
add s1, s6 ; se suma el valor al contador.

sr0 s7
sr0 s7
sr0 s7
sr0 s7 ; se mueve los bits 4 espacios a la derecha.
and s7, s2 ; aplico máscara a s7.
add s1, s7 ; se suma el valor al contador.

sr0 s8
sr0 s8
sr0 s8
sr0 s8
sr0 s8 ; se mueve los bits 5 espacios a la derecha.
and s8, s2 ; aplico máscara a s8.
add s1, s8 ; se suma el valor al contador.

sr0 s9
sr0 s9
sr0 s9
sr0 s9
sr0 s9
sr0 s9 ; se mueve los bits 6 espacios a la derecha.
and s9, s2 ; aplico máscara a s9.
add s1, s9 ; se suma el valor al contador.

sr0 sa
sr0 sa
sr0 sa
sr0 sa
sr0 sa
sr0 sa
sr0 sa ; se mueve los bits 7 espacios a la derecha.
and sa, s2 ; aplico máscara a sa.
add s1, sa ; se suma el valor al contador. se queda con el número de 1s del número ingresado.
```

8. Implementar en assembly un intercambiador entre la posición baja y alta de un registro. Comprobarlo en el simulador.

```
address 0

input s0, 0
rr s0
rr s0
rr s0
rr s0 ; se mueve los valores 4 espacios a la derecha.
```