## Ejercicio 1

; Implementar en assembly un código que permita sacar por los leds un valor

;de 8 bits. Comprobarlo en la placa.

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT all\_clear , 00 ; Ceros

;=============================================================

; Codigo assembly principal

;=============================================================

main: LOAD s0, all\_clear ; Cargo ceros al registro s0

LOAD s1, FF ; Cargo 1111 1111 al registro s1

OUTPUT s1, leds\_out ; Encienden LEDs segun registro s1

JUMP main ; Vuelvo a main

## Ejercicio 2

; Implementar en assembly un código que permita sacar por 4 leds el estado

;de los switches. Comprobarlo en la placa.

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT sw\_in , 00 ; Switch read port

CONSTANT all\_clear , 00 ; Ceros

;=============================================================

; Codigo assembly principal

;=============================================================

main: LOAD s0, all\_clear ; Cargo ceros al registro s0

INPUT s2, sw\_in ; Leo estado de los switchs

LOAD s1, s2 ; Cargo registro s2 al registro s1

OUTPUT s1, leds\_out ; Encienden LEDs segun registro s1

JUMP main ; Vuelvo a main

## Ejercicio 3

; Implementar en assembly la suma de dos números de 8 bits.

;Comprobarlo en la placa.

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT all\_clear , 00 ; Ceros

;=============================================================

; Codigo assembly principal

;=============================================================

main: LOAD s2, 03 ; Cargo 0010 0000 al registro s2

LOAD s3, 39 ; Cargo 0000 0100 al registro s3

ADD s2, s3 ; Sumo registros s2 y s3, cargo en s2

OUTPUT s2, leds\_out ; Encienden LEDs segun registro s2

JUMP main ; Vuelvo a main

## Ejercicio 4

…

## Ejercicio 5

; Implementar en assembly un desplazador a izquierda en función del

;estado de los switches

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT sw\_in , 00 ; Switchs read port

CONSTANT clear\_all , 00 ; Limpieza

;=============================================================

; Codigo assembly principal

;=============================================================

LOAD s3, clear\_all ; Limpio registro s3

LOAD s3, 01 ; Cargo 01 a s3

main: INPUT s4, sw\_in ; Leo estados de los switchs

AND s4, 0F ; Aplico mascara para quedarme con los bits - significativos

COMPARE s4, 00 ; Comparo. Si s4 == 00, Z = 1

JUMP Z, main ; Si Z = 1, vuelvo a main

RL s3 ; Muevo bits hacia la izquierda

CALL delay\_1s ; Rutina de demora

OUTPUT s3, leds\_out ; Veo registro s3 en leds

JUMP main ; Vuelvo a main

delay\_1s: LOAD s2, 7D

LOAD s1, E2

LOAD s0, 90

delay\_loop: SUB s0, 1'd

SUBCY s1, 0'd

SUBCY s2, 0'd

JUMP NZ, delay\_loop

return

; FUNCIONA :D

## Ejercicio 6

; Implementar en assembly un multiplicador con operandos de 8 bits sin utilizar instrucciones de

;salto (tener en cuenta que el resultado debe ser tal que no supere los 8 bits). Comprobarlo en la

;placa.

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT sw\_in , 00 ; Switchs read port

CONSTANT clear\_all , 00 ; Limpieza

;=============================================================

; Codigo assembly principal

;=============================================================

main: LOAD s1, clear\_all ; Limpio registro s1

LOAD s0, 03 ; Cargo 3 en s0

ADD s1, s0

ADD s1, s0

OUTPUT s1, leds\_out

## Ejercicio 7

; Implementar en assembly la suma de dos números de 8 bits.

;Comprobarlo en la placa.

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT all\_clear , 00 ; Ceros

;=============================================================

; Codigo assembly principal

;=============================================================

main: LOAD s2, 03 ; Cargo 0010 0000 al registro s2

LOAD s3, 39 ; Cargo 0000 0100 al registro s3

ADD s2, s3 ; Sumo registros s2 y s3, cargo en s2

OUTPUT s2, leds\_out ; Encienden LEDs segun registro s2

JUMP main ; Vuelvo a main

## Ejercicio 8

…

## Ejercicio 9

; Implementar en assembly un contador de los 1’s presentes en un registro.

;Comprobarlo en la placa.

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT sw\_in , 00 ; Switchs read port

CONSTANT clear\_all , 00 ; Limpieza

;=============================================================

; Codigo assembly principal

;=============================================================

main: LOAD s0, clear\_all ; Limpio registro s0

LOAD s1, clear\_all ; Limpio registro s1

LOAD s2, clear\_all ; Limpio registro s2

; Cargo valores para contar 1's

LOAD s0, 99 ; Cargo 0x99 a s0 (1001 1001)

LOAD s1, 08 ; Cargo 0x08 a s1 para poder iterar

; Entro a un ciclo for para contar los 1's

for: RL s0 ; Roto el registro s1, bit 7 se copia en Carry

JUMP NC, nocount ; Si C == 0, salto a 'nocount'

ADD s2, 01 ; Adiciono 0x01 a s2

nocount: SUB s1, 01 ; Si C != 0, resto 0x01 a s1

JUMP Z, salgo ; Si s1 == 0, salto a 'salgo'

JUMP for ; Si s1 != 0, salto a 'for'

; Muestro resultado en LEDs

salgo: OUTPUT s2, leds\_out

## Ejercicio 10

; Implementar en assembly la suma de dos números de 8 bits.

;Comprobarlo en la placa.

;=============================================================

; Constantes

;=============================================================

CONSTANT leds\_out , 01 ; LED write port

CONSTANT all\_clear , 00 ; Ceros

;=============================================================

; Codigo assembly principal

;=============================================================

main: LOAD s2, 03 ; Cargo 0010 0000 al registro s2

LOAD s3, 39 ; Cargo 0000 0100 al registro s3

ADD s2, s3 ; Sumo registros s2 y s3, cargo en s2

OUTPUT s2, leds\_out ; Encienden LEDs segun registro s2

JUMP main ; Vuelvo a main