Enero 2020

Ejercicio 1 | Análisis de programa en lenguaje de ensamble | **1.8 puntos**

La memoria RAM de un microcontrolador basado en un procesador ARM *Cortex-M0* (*little-endian*) contiene los datos dados en la siguiente tabla:

Dirección / y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x1000 000y	0E	3E	2D	40	7F	7F	7F	00	01	7F	7E	7F	7F	7F	0D	A6

81 > 7F

Los valores de algunos registros del procesador son:

Registro	Contenido
R0	0x1000 0000
R1	0xDEAD C0DE
R2	0xFEE1 BAAD
R3	0xA5A5 5A5A

Stack Pointer

y SP apunta a una zona de memoria RAM. En estas circunstancias se llama a una función cuyo prototipo es `void fun(uint8_t *)` y cuyo cuerpo es:

```
fun
        PUSH    {R2,R3}     ; guarda el contenido de R2 y R3 en el stack
        LDRB    R1, [R0]
        MOVS    R2, #0x7F
fun_loop
        LDRB    R3, [R0, R1]
        CMP     R3, R2
        BLE     fun_tail
        STRB    R2, [R0, R1]
fun_tail
        SUBS    R1, #1
        BNE     fun_loop
        POP     {R2,R3}     ; recupera los contenidos guardados en el stack
        BX      LR          ; return
```

if

$R_3 \leq R_2$

$\rightarrow R_2 = 0x\ 0000\ 007F$

$R_1 = 0x0000\ 000E$

$\rightarrow R_3 = 0x\ 0000\ 000D$

Apartado A. Rellene la siguiente tabla indicando cómo quedarán los registros del procesador tras la ejecución de `fun()` (es decir, cuando se alcance la instrucción `BX LR`, que retorna de la función).

Registro	Contenido (hexadecimal con 32 bits)
R0	0x 0000 0001
R1	0x0000 0000
R2	0x FEE1 BAAD
R3	0x A5A5 5A5A

Apartado B. Rellene en la siguiente tabla los valores de las posiciones de memoria que se modifiquen tras la ejecución de `fun()`. No rellene todas las posiciones, solo complete las posiciones que se modifiquen como resultado de la ejecución de la función.

Dirección / y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x1000 000y																

Apartado C. Indique la *utilidad* de la función `fun()`, señalando claramente la relación entre los datos inicialmente existentes en RAM y los datos finales en la misma.

Todos aquellos bytes de la memoria que sean mayores a 0x7F se sobreescriben con dicho valor.

Apartado D. Repita el apartado A suponiendo ahora que se modifica `fun()` eliminando las instrucciones `PUSH` y `POP`.

Registro	Contenido (hexadecimal con 32 bits)
R0	11
R1	11
R2	0x 0000 007F
R3	0x 0000 003E

Ejercicio 5 | Periféricos | **1.6 puntos**

En una placa *mbed* LPC1768 (o Nucleo-l432kc) está corriendo el siguiente código:

```
#include "mbed.h"
#include "pinout.h"      // para TRG_PIN, IN_PIN y OUT_PIN

InterruptIn trigger(TRG_PIN);
AnalogIn    in(IN_PIN);
AnalogOut   out(OUT_PIN);
Serial      pc(USBTX, USBRX);

bool        trigger_rise_evnt;
bool        trigger_fall_evnt = true;

void trigger_isr(void) {
  if(trigger) {
    trigger_rise_evnt = true;
  } else {
    trigger_fall_evnt = true;
  }
}

void main(void) {
  uint16_t  sample = 0x4CCE;

  trigger.rise(trigger_isr);
  trigger.fall(trigger_isr);

  pc.baud(10000);
  pc.format(8, pc.Even, 2);

  while(true) {
    if(trigger_rise_evnt) {
      trigger_rise_evnt = false;
      sample = in.read_u16();
    }
    if(trigger_fall_evnt) {
      trigger_fall_evnt = false;
      sample = 0xC000 - sample/2;
      out.write_u16(sample);
      if(sample > 0xB000) {
        pc.putc(0x87);
      }
    }

    __disable_irq();
    if(!trigger_rise_evnt && !trigger_fall_evnt) {
      __WFI();
    }
    __enable_irq();
  }
}
```

PAR

8bits Even (par) stop=2

t=0

45874

6553

19662

58981

49152 − 58981/2

49056

65535

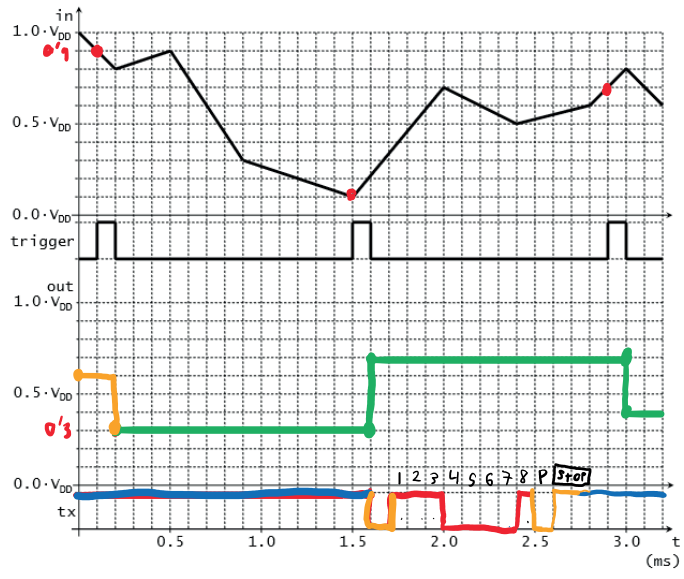
1000 0111

= 0'3

0'7

0'4

En el siguiente cronograma se muestra la evolución de las señales `in` y `trigger` asociadas a los pines `IN_PIN` y `TRG_PIN` respectivamente. **Complete el cronograma** para las señales `out` y `tx` asociadas a los pines `OUT_PIN` y `USBTX`. Suponga que el tiempo necesario para la ejecución de cada una de las líneas de código del programa es despreciable en la escala de tiempos del cronograma y que $t = 0$ coincide con el instante en el que, por primera vez, se alcanza la línea `while(true)`. V_{DD} es la tensión de alimentación del microcontrolador y coincide con la tensión de referencia de ADC y DAC.



Sample$_1$ = 0'9 · 65535 = 58981

Sample$_2$ = 0'1 · $(2^{16}-1)$ = 6553

Sample$_3$ = 0'7 · 11 = 45874