

Apellidos:.....**SOLUCIÓN**.....

Nombre:.....

P1		P2		P3	

Duración 2:00 horas

1.- [3 Puntos] Preguntas cortas:

a) Analice este código e indique qué hace:

```

xor.w  r12, r13
xor.w  r13, r12
xor.w  r12, r13

```

SOLUCIÓN

Hace el intercambio de R12 y R13 en 3 ciclos sin usar un registro extra. Supongamos R12=A y R13=B. Desglose de operaciones:

```

xor.w  r12, r13      ;R13=A XOR B
xor.w  r13, r12      ;R12=A XOR (A XOR B)=B
xor.w  r12, r13      ;R13=(A XOR B) XOR B=A

```

b) Desensamble el siguiente trozo de programa e indique qué hace, cuánto ocupa y cuánto tarda en ejecutarse suponiendo $f_{MCLK}=1\text{MHz}$:

```

$B33C
$7C0C

```

SOLUCIÓN

;	Instrucción	Cód.máq. (hexa)	Código máquina (binario) OpCd -Rf- DdBDf -Rd- (tipo I)
;	-----	-----	-----
	bit.w #-1, r12	;\$B33C	1011 0011 0011 1100 (tipo I)
	subc.w r12, r12	;\$7C0C	0111 1100 0000 1100 (tipo I)

Análisis:

- La primera instrucción comprueba todos los bits de R12. El bit Z se pondrá a 1 si TODOS valen 0. 1 en caso contrario. Es decir, $Z = \text{NOR}(\text{bits de R12})$. $C = Z = \text{OR}(\text{bits de R12})$.
- La segunda calcula $R12 = R12 - R12 - C$, lo que deja en R12 un 0 si $C=0$ y un -1 (0xFFFF) si $C=1$. Es decir, si algún bit de R12 era 1, poner TODOS los bits de R12 a 1. 0 en caso contrario.

Ocupación:

- 2 palabras (4 bytes)

Tiempo de ejecución:

- Ambas instrucciones tardan un ciclo en ejecutarse, ya que la primera usa R3 para generar la constante -1. Con $f_{MCLK} = 1\text{MHz}$, cada ciclo es $1\mu\text{s}$. Por tanto, $2\mu\text{s}$.

c) Describa qué es la TVI (Tabla de Vectores de Interrupción) y cómo se configura.

CRITERIO DE CORRECCIÓN

- a) 30%
- b) 40%
- c) 30%

2.- [3 Puntos] Se solicita programar en ensamblador del MSP430 la subrutina `strncpy` que sigue el convenio de llamada de C:

```
unsigned int strncpy(char *dst, const char *src, unsigned int size);
```

donde `dst` es el búfer de destino donde se copiará la cadena, `src` es la cadena de origen y `size` es el tamaño total del búfer de destino (incluyendo el terminador de cadena).

Devuelve la longitud total de la cadena que intentó crear (es decir, la longitud de `src`). Si el valor devuelto es mayor o igual que `size`, significa que la cadena fue truncada debido a la falta de espacio.

SOLUCIÓN

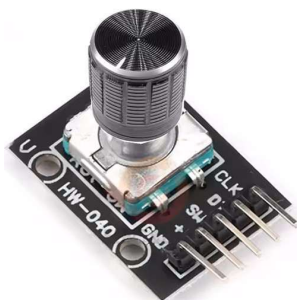
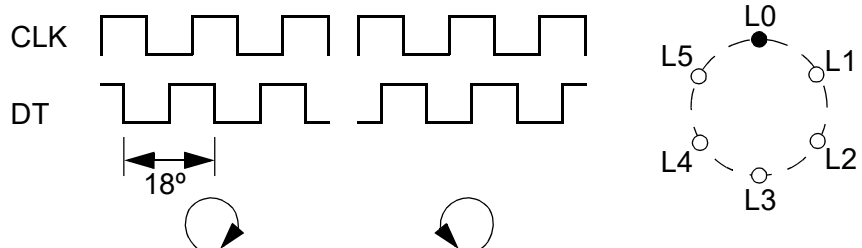
Mapeo de registros:

- `dst` en R12 (Puntero, 16 bits). También se usa para el valor de retorno.
- `src` en R13 (Puntero, 16 bits).
- `size` en R14 (Contador entero sin signo, 16 bits)
- R15, copia de R13 para calcular la longitud de `src`
- R11, variable auxiliar para leer los caracteres a copiar

```
-----  
;unsigned int strcpy(char *dst, const char *src, unsigned int size);          v1.0  
-----  
strcpy      mov.w  r13, r15          ;R15=src. Se usará para calcular longitud  
            dec.w  r14              ;Sólo se pueden copiar size-1 caracteres  
            jlo    strlcpyLen        ;Era 0? Sólo medir cadena src  
  
            ;Copiar src en dst mientras haya sitio  
strlcpyBuc  mov.b  @r13+, r11        ;Leer un carácter  
            mov.b  r11, 0(r12)      ;Escribir en destino  
            tst.b  r11              ;Era final de cadena?  
            jeq    strlcpyFin        ;...sí. Hemos terminado todo  
            inc.w  r12              ;...no. Avanzar puntero destino  
            dec.w  r14              ;Un hueco menos en destino. Queda hueco?  
            jnz    strlcpyBuc        ;...sí. Seguir mientras haya sitio  
            clr.b  0(r12)          ;...no. Marcar final de cadena en destino  
  
            ;Se ha acabado el buffer fuente. Recorrer (resto de) src y medir longitud  
strlcpyLen  mov.b  @r13+, r11        ;Leer un carácter  
            tst.b  r11              ;Era final de cadena?  
            jne    strlcpyLen        ;...no. Avanzar hasta encontrar terminador  
  
            ;Calcular la longitud de src, devolver por R12 y salir  
strlcpyFin  sub.w  r15, r13          ;...sí. Calcular longitud de src  
            dec.w  r13              ;No contabilizar terminador  
            mov.w  r13, r12         ;Salida por R12  
            ret
```

CRITERIO DE CORRECCIÓN

- 3.- [4 Puntos] El HW-040 es un encoder diferencial de 20 pulsos por vuelta. Se trata de un dispositivo con un eje accionable que proporciona 2 salidas indicando la dirección y velocidad de giro (CLK y DT). Además, dispone de un pulsador asociado al eje Z (salida SW) y dos terminales de alimentación (+ y GND).



Es fácil determinar sentido de giro (vea la figura). Así, se gira sentido horario si, al detectar un flanco en CLK, ambas salidas son iguales (CLK=DT), mientras que si son complementarias, el sentido es el contrario al de las agujas del reloj.

Se ha conectado un HW-040 al Launchpad del MSP430: CLK en P1.0 y DT en P1.1 (SW no se usa). En P1.2 a P1.7 hay 6 leds (L0 a L5 respectivamente) conectados en forma de círculo en configuración de lógica positiva (1 enciende el led y 0 lo apaga).

Realice un programa en ensamblador del MSP430 usando programación por interrupciones y con el mínimo consumo posible que inicialmente encienda L0 y cambie el led encendido en función de los pulsos que se reciban en el encoder. Cuando se gire en sentido horario, se encenderá el siguiente (siguiente de L5 es L0). Si se gira en sentido antihorario, se encenderá el anterior (anterior de L0 es L5). Cada pulso recibido supondrá un cambio de led. Considere pila inicializada, perro guardián desactivado, puertos desbloqueados, ACLK a 32768Hz.

SOLUCIÓN

Inicialmente se configurarán P1.0 (CLK) y P1.1 (DT) como entradas. El primero sensible a flanco de bajada y con las interrupciones habilitadas. El resto de bits del puerto P1 como salidas para los leds. Se enciende el L0. Hecho esto se entra en modo de bajo consumo (LPM4).

En la ISR se borra el flag de interrupción y se guarda el estado de los leds en R4 para agilizar los accesos. Se lee DT. Si es 0, se ha girado en sentido horario. Si es 1, en sentido antihorario. En el primer caso se rotan circularmente los bits 2 a 7 de R4 hacia la izquierda. Para ello, se desplaza a la izquierda R4 y, si el bit de salida, que se guarda en el C, es 1, se enciende el L0. En el segundo caso, se hace el desplazamiento a la derecha. Si el bit 1 se enciende, se apaga y se enciende el L5. Una vez preparado el siguiente led activo, se manda al puerto de salida.

Se podría haber detectado el flanco de subida de CLK en lugar del de bajada, con ningún otro cambio en el código. También se podría haber configurado la interrupción de DT en lugar de CLK, pero aquí la lógica de CLK hubiera sido la contraria: sentido antihorario si las señales son iguales y horario en caso contrario.

```
;Constantes de configuración
LEDS_B0      .equ    BIT2          ;Led0, bit
LEDS_BITS    .equ    BIT2|BIT3|BIT4|BIT5|BIT6|BIT7
LEDS_P       .equ    P1IN         ;Leds, puerto
CLK_B        .equ    BIT0         ;Salida CLK del encoder. Puerto y bit
CLK_P        .equ    P1IN
DT_B         .equ    BIT1         ;Salida DT del encoder. Puerto y bit
DT_P         .equ    P1IN

;-----
;main
;-----
main          ;Leds salida
              mov.b   &LEDS_B0, &LEDS_P+POUT      ;Encender el L0 al principio
              bis.b   #LED_BITS, &LEDS_P+PDIR      ;Leds salida

              ;CLK y DT entrada
              ;CLK IRQ sensible en flanco de bajada
              bic.b   #CLK_B, &CLK_P+PDIR         ;CLK entrada
              bic.b   #DT_B, &DT_P+PDIR         ;DT entrada
              bis.b   #CLK_B, &CLK_P+PIES        ;CLK sensible en flanco de bajada
              bic.b   #CLK_B, &CLK_P+PIFG        ;CLK borrar flag de interrupción
              bis.b   #CLK_B, &CLK_P+PIE         ;CLK activar IRQ

              bis.w   #LPM4|GIE, sr;Ea, a dormir

;-----
; P1ISR
;-----
; ISR del P1 (CLK)
;-----
P1ISR         bic.b   #CLK_B, &CLK_P+PIFG        ;Borrar IRQ
              push.w  r4                        ;No modificar registros
              mov.b   &LEDS_P+POUT, r4          ;R4=Estado de leds
              bit.b   #DT_B, &DT_P+PIN          ;Sentido horario?
              jz      P1ISRHor                  ;...sí. Incrementar led
P1ISRNoHor    ;Sentido antihorario              ...no. decrementar led
```

```

        clrc
        rrc.b r4
        bit.b #LEDS_B0>>1, r4
        jz    P1ISRLedOut
        xor.b #LEDS_B5|LEDS_B0>>1, r4
        jmp   P1ISRLedOut
P1ISRHor ;Sentido horario
        rla.b r4
        jnc   P1ISRLedOut
        bis.b #LEDS_B0, r4
P1ISRLedOutmov.b r4, &LEDS_P+POUT
        pop.w r4
        reti

.intvecRESET_VECTOR, main
.intvecPORT1_VECTOR, P1ISR

```

```

;C=0, para hacer un LSR
;Desplazar a la derecha un bit
;Se ha salido?
;...no. Actualizar leds
;...sí. Encender led del otro extremo
;Actualizar leds
    Decrementar led
;Desplazar a la izquierda un bit
;...no. Actualizar leds
;...sí. Encender led del otro extremo
;Actualizar led
;No modificar registros

```

CRITERIO DE CORRECCIÓN