Ej13.pdf

GeXx_

Estructura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

```
      .include   "inter.inc"
.text
/*  Agrego  vector  interrupciÛn */
      ADDEXC   0x18,  irq_handler
/*  Inicializo  la pila en modos  IRQ y SVC */
      mov      r0, #0b11010010    @ Modo  IRQ,  FIQ&IRQ  desact
      msr      cpsr_c,  r0
      mov      sp, #0x8000
      mov      r0, #0b11010011    @ Modo  SVC,  FIQ&IRQ  desact
      msr      cpsr_c,  r0
      mov      sp, #0x8000000
/*  Configuro  GPIOs 4, 9, 10, 11, 17, 22 y 27 como  salida  */
      ldr      r0, =GPBASE
      ldr      r1, =0b00001000000000000001000000000000
      str      r1, [r0, #GPFSEL0]
/* guia  bits            xx99988777666555444333222111000 */
      ldr      r1, =0b00000000001000000000000000001001
      str      r1, [r0, #GPFSEL1]
      ldr      r1, =0b00000000001000000000000001000000
      str      r1, [r0, #GPFSEL2]

/*  Programo  C1 y C3 para  dentro  de 2 microsegundos  */
      ldr      r0, =STBASE
      ldr      r1, [r0, #STCLO]
      add      r1, #2
      str      r1, [r0, #STC1]
      str      r1, [r0, #STC3]

/*  Habilito  interrupciones,  local y globalmente  */
      ldr      r0, =INTBASE
      mov      r1, #0b1010
      str      r1, [r0, #INTENIRQ1]
      mov      r0, #0b01010011    @ Modo  SVC,  IRQ  activo
      msr      cpsr_c,  r0

/*  Repetir  para  siempre  */
      bucle:   b         bucle

/*  Rutina  de  tratamiento  de  interrupciÛn */
irq_handler:
      push     {r0,  r1, r2, r3}
/* Leo  origen  de la  interrupciÛn */
      ldr      r0, =STBASE
      ldr      r1, =GPBASE
      ldr      r2, [r0, #STCS]
      ands     r2, #0b0010
      beq      sonido
/* Si es C1,  ejecuto  secuencia  de LEDs */
      ldr      r2, =cuenta
      /* guia  bits    10987654321098765432109876543210 */
      ldr      r3, =0b00001000010000100000111000000000
      str      r3, [r1, #GPCLR0]      @ Apago  todos  los  LEDs
      ldr      r3, [r2]                     @ Leo  variable  cuenta
      subs     r3, #1                    @ Decremento
      moveq    r3, #6                  @ Si es 0,  volver a 6
      str      r3, [r2]                     @ Escribo  cuenta
      ldr      r3, [r2, +r3, LSL #2]  @ Leo  secuencia
      str      r3, [r1, #GPSET0]      @ Escribo  secuencia  en LEDs
```

```
/*  Reseteo  estado  interrupciÛn de C1 */
      mov      r3, #0b0010
      str      r3, [r0, #STCS]
/*  Programo  siguiente  interrupciÛn en 200ms */
      ldr      r3, [r0, #STCLO]
      ldr      r2, =200000         @ 5 Hz
      add      r3, r2
      str      r3, [r0, #STC1]
/* øHay  interrupciÛn pendiente  en C3? */
      ldr      r3, [r0, #STCS]
      ands     r3, #0b0100
      beq      final               @ Si no,  salgo

/* Si es C3,  hago  sonar el  altavoz  */
sonido:
      ldr      r2, =bitson
      ldr      r3, [r2]
      eors     r3, #1              @ Invierto  estado
      str      r3, [r2]
      mov      r3, #0b10000       @ GPIO 4 (altavoz)
      streq    r3, [r1, #GPSET0] @ Escribo  en  altavoz
      strne    r3, [r1, #GPCLR0] @ Escribo  en  altavoz
/*  Reseteo  estado  interrupciÛn de C3 */
      mov      r3, #0b1000
      str      r3, [r0, #STCS]
/*  Programo  interrupciÛn para  sonido  de 440 Hz */
      ldr      r3, [r0, #STCLO]
      ldr      r2, =1136           @ Contador  para  440 Hz
      add      r3, r2
      str      r3, [r0, #STC3]
/*  Recupero  registros y salgo  */
final:
      pop      {r0, r1, r2, r3}
      subs     pc, lr, #4
bitson: .word    0               @ Bit 0 = Estado  del  altavoz
cuenta: .word    1               @ Entre 1 y 6, LED a encender
/* guia  bits        76543210987654321098765432l0 */
secuen: .word    0b10000000000000000000000000000
.word    0b00000100000000000000000000000
.word    0b00000000001000000000000000000
.word    0b00000000000000010000000000
.word    0b00000000000000001000000000
.word    0b00000000000000000100000000
```