COMPU RASPBERRY

1. Write a program that turns one red led, one yellow led and one green led of the expansion board. At startup, we know that the LEDs are all off. Denote Exer1.s this program. Remember that due to the lack of an operating system, your program has to finish by an infinite loop.

```
@Programa que enciende una led roja, otra amarilla y otra verde

    ldr r0, =GPBASE
    ldr r1,=0xE000D00       @roja,amarilla,verde uno de cada
    str r1,[r0,#GPSET0]     @turn on

end : b end
```

2. Write a program that turns on one of the yellow leds while you are pressing one of the buttons, and it turns it off if you release the button.

```
@Programa que enciende una led amarilla mientras se presione uno de los
@dos botones

    ldr r0, =GPBASE
x:
    ldr r1,=0x800           @amarillo
    ldr r2,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r2,#0b00100         @force z=1 or 0
    streq r1,[r0,#GPSET0]   @si esta presionado se enciende
    strne r1,[r0,#GPCLR0]   @si no se apaga
    b x

end: b end
```

3. Write a program that turn on the two green leds after pressing one button (edge triggered). If the other button is pressed, both leds will turn off.

```
@Programa que tras pulsar un boton encienda dos amarillas y si el otro boton se
@presiona, ambos se apagan

    ldr r0, =GPBASE
x:
    ldr r1,=0x08400000      @dos amarillas
    ldr r2,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r2,#0b00100         @force z=1 or 0 (#0b00100 boton1)
    streq r1,[r0,#GPSET0]   @si esta presinado lo enciende
    ldr r3,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r3,#0b01000         @force z=1 or 0 (#0b00100 boton2)
    streq r1,[r0,#GPCLR0]   @si esta presinado lo enciende
    b x

end: b end
```

4. Write a program that after pressing the button 1, one of the yellow leds turns on permanently, and if you press the button 2 a green led will turn on permanently.

```
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que tras presionar el boton1 enciende una amarilla y tras presionar
@boton2 enciende el verde

    ldr r0, =GPBASE
x:
    ldr r1,=0x800           @amarilla
    ldr r4,=0xE000000       @verde

    ldr r2,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r2,#0b00100         @force z=1 or 0 (#0b00100 boton1)
    streq r1,[r0,#GPSET0]   @si esta presionado se enciende

    ldr r3,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r3,#0b01000         @force z=1 or 0 (#0b01000 boton2)
    streq r4,[r0,#GPSET0]   @si esta presionado se enciende
    b x

end: b end
```

5. Write a program that turns on the two red LEDs. After this, the program polls the state of the push buttons. Once a push button is pressed, the corresponding LED will keep on, whereas the other LED will turn off.

```
@Programa que enciende las dos rojas, comprueba el estado de los botones
@y enciende la led del boton presionado mientras que del otro lo apaga

    ldr r0, =GPBASE
    ldr r1,=0x600           @las dos rojas
    str r1,[r0,#GPSET0]     @las enciende

x:
    ldr r1,=0x200           @una roja
    ldr r3,=0x400           @segunda roja

    ldr r2,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r2,#0b00100         @force z=1 or 0 (#0b00100 boton1)
    streq r1,[r0,#GPCLR0]   @si esta presionado se apaga

    ldr r4,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r4,#0b01000         @force z=1 or 0 (#0b01000 boton2)
    streq r3,[r0,#GPCLR0]   @si esta presionado se apaga
    b x

end: b end
```

6.Modify the program Ejer1.s to create a new one which flashes the LEDs at the rate of 1 s. ON-OFF. Insert the corresponding delay using the timer.

```
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que va encendiendo y apagando la led roja en intervalos de 1s

    ldr r0, =GPBASE
    ldr r1,=0xE00D00            @roja,amarilla,verde uno de cada
    str r1,[r0,#GPSET0]         @turn on

loop:   str r1,[r0,#GPSET0]     @la enciende
        BL wait                 @entra en el bucle wait
        str r1,[r0,#GPCLR0]     @la apaga
        BL wait                 @vuelve a esperar
        B loop                  @vuelve a comenzar

wait:   ldr r7,=STBASE          @r7=0x3F003004 (address of counter CLO)
        ldr r3,[r7,#STCLO]      @read the value of the counter
        ldr r4,=1000000         @r4=1000000 mu s
        add r4,r4,r3            @adding to the current count to get the final count
        ret1: ldr r3,[r7,#STCLO]@read the current count
            cmp r3,r4           @comparing current count with the final count
            blt ret1
            bx lr

end: b end
```

7. Flash the LEDs at a rate of 1 s. ON, 0.25 s. OFF by using the timer.

```
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Bucle que enciende las leds durante 1s y las apaga 0,25s

    ldr r0, =GPBASE
    ldr r1,=0xE000D00           @roja,amarilla,verde uno de cada

loop:   str r1,[r0,#GPSET0]     @la enciende
        BL wait1                @entra en el bucle wait de 1s
        str r1,[r0,#GPCLR0]     @la apaga
        BL wait2                @entra en el bucle wait de 0,25s
        B loop                  @vuelve a empezar

wait1:  ldr r7,=STBASE          @r7=0x3F003004 (address of counter CLO)
        ldr r3,[r7,#STCLO]      @read the value of the counter
        ldr r4,=1000000         @r4=1000000 mu s
        add r4,r4,r3            @adding to the current count to get the final count
        ret1: ldr r3,[r7,#STCLO]@read the current count
            cmp r3,r4           @comparing current count with the final count
            blt ret1
            bx lr


wait2:  ldr r7,=STBASE          @r7=0x3F003004 (address of counter CLO)
        ldr r3,[r7,#STCLO]      @read the value of the counter
        ldr r4,=25000           @r4=25000 mu s
        add r4,r4,r3            @adding to the current count to get the final count
        ret2: ldr r3,[r7,#STCLO]@read the current count
            cmp r3,r4           @comparing current count with the final count
            blt ret2
            bx lr


end: b end
```

8. Modify the program of the previous exercise such that instead of acts on the LED, it generates a 440Hz tone (note LA) on the speaker.

```
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Bucle que hace la nota LA (440Hz) durante 1s y la apaga 0,25s

    ldr r0, =GPBASE
    ldr r1,=0x010               @sonido

loop:   str r1,[r0,#GPSET0]     @suena
        BL wait
        str r1,[r0,#GPCLR0]     @se apaga
        BL wait
        B loop

wait:   ldr r7,=STBASE          @r7=0x3F003004 (address of counter CLO)
        ldr r3,[r7,#STCLO]      @read the value of the counter
        ldr r4,=2272            @LA
        add r4,r4,r3            @adding to the current count to get the final count
        ret: ldr r3,[r7,#STCLO] @read the current count
            cmp r3,r4           @comparing current count with the final count
            blt ret
            bx lr


end: b end
```

9. Write a program that the green LEDs flashes at a rate of 1 second ON-OFF, then at a rate of 500 ms. ON-OFF and finally 250 ms. ON-OFF in an infinite loop.

```
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que enciende las luces verdes 1s, 500ms, 250ms

    ldr r0, =GPBASE
    ldr r1,=0xE000D00

loop: str r1,[r0,#GPSET0]       @enciende
    BL wait1                    @espera 1s
    str r1,[r0,#GPCLR0]         @apaga
    BL wait1                    @espera 1s
    str r1,[r0,#GPSET0]
    BL wait2                    @espera 0,5
    str r1,[r0,#GPCLR0]
    BL wait2                    @espera 0,5
    str r1,[r0,#GPSET0]
    BL wait3                    @espera 0,25
    str r1,[r0,#GPCLR0]
    BL wait3                    @espera 0,25
    B loop
```

```
wait1:  ldr r7,=STBASE          @r7=0x3F003004 (address of counter CLO)
        ldr r3,[r7,#STCLO]      @read the value of the counter
        ldr r4,=1000000         @r4=1000000 mu s
        add r4,r4,r3            @adding to the current count to get the final count
        ret1: ldr r3,[r7,#STCLO]@read the current count
            cmp r3,r4           @comparing current count with the final count
            blt ret1
            bx lr

wait2:  ldr r7,=STBASE
        ldr r3,[r7,#STCLO]
        ldr r4,=500000
        add r4,r4,r3
        ret2: ldr r3,[r7,#STCLO]
            cmp r3,r4
            blt ret2
            bx lr

wait3:  ldr r7,=STBASE
        ldr r3,[r7,#STCLO]
        ldr r4,=25000
        add r4,r4,r3
        ret3: ldr r3,[r7,#STCLO]
            cmp r3,r4
            blt ret3
            bx lr

end: b end
```

10. Write a new program which polls both the buttons 1 (GPIO 2) and 2 (GPIO 3). If the first pressed button is the button 1, a tone of 262Hz (note DO) has to be generated. Otherwise, if the first pressed button is the botton 2, a tone of 391Hz (note SOL) is generated.

```
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que si se presiona el boton1 (GPIO2) suena 252Hz(DO),
@boton2 (GPIO3) 391Hz (SOL)

    ldr r0, =GPBASE
    ldr r1,= 0x010          @sonido

x:  ldr r2,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r2,#0b00100         @force z=1 or 0 (#0b00100 boton1)
    beq loop1

    ldr r4,[r0,#GPLEV0]     @read button (1 no pressed, 0 pressed)
    tst r4,#0b01000         @force z=1 or 0 (#0b01000 boton2)
    beq loop2

    b x

loop1:  str r1,[r0,#GPSET0] @suena
        BL wait1
        str r1,[r0,#GPCLR0] @para
        BL wait1
        B loop1
```

```
loop2:  str r1,[r0,#GPSET0]
        BL wait2
        str r1,[r0,#GPCLR0]
        BL wait2
        B loop2

wait1:  ldr r7,=STBASE      @r7=0x3F003004 (address of counter CLO)
        ldr r3,[r7,#STCLO]  @read the value of the counter
        ldr r4,=3816        @DO
        add r4,r4,r3        @adding to the current count to get the final count

        ret1:   ldr r3,[r7,#STCLO]  @read the current count
                cmp r3,r4           @comparing current count with the final count
                blt ret1
                bx lr

wait2:  ldr r7,=STBASE
        ldr r3,[r7,#STCLO]
        ldr r4,=2557
        add r4,r4,r3

        ret2:   ldr r3,[r7,#STCLO]
                cmp r3,r4
                blt ret2
                bx lr


end:    b end
```

11. Write a code (Exer11.s) which configure the timer C1 comparator such that after 2 seconds a IRQ is produced, and the corresponding handler routine turns on the YELLOW LEDs

```
/* Vector Table inicialization */
    mov r0,#0
    ADDEXC 0x18, regular_interrupt

/* Stack init for IRQ mode */
    mov     r0, #0b11010010
    msr     cpsr_c, r0
    mov     sp, #0x8000
/* Stack init for FIQ mode */
    mov     r0, #0b11010001
    msr     cpsr_c, r0
    mov     sp, #0x4000
        mov r8,#0
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que utiliza C1 y dsps de 2 segundos produce un IQR y enciende las
@leds amarillas

    @Preparing C1 to send an interrupt after "y" microseconds
    ldr r0, =STBASE
    ldr r1, [r0, #STCLO]
    add r1, #0x200000       @y microseconds
    str r1, [r0, #STC1]     @#STC3 for C3


    @Enable local interrupt IRQ?
    ldr r0,=INTBASE
    mov r1, #0b0010         @0b1000 si fuera C3
    str r1,[r0,#INTENIRQ1]

    @Enable global interrupt IRQ (for SVC mode)
    mov r1, #0b01010011
    msr cpsr_c, r1

end:    b end
```

```
regular_interrupt:

    push {r0,r1}

    ldr r0, = GPBASE
    ldr r1, =0x30800        @led amarillo
    str r1, [r0, #GPSET0]   @enciende

    ldr r0, =STBASE         @boton 1
    ldr r2, [r0, #STCS]
    tst r2,#0b0010

    @aquí preparamos el programa para que pueda haber otro interrupt
    @Reset (end) of interrupt IRQ2 (GPIO2&3) to allow a new interrupt via GPIO2&3
    ldr r0, =GPBASE
    mov r1, #0b0010
    str r1,[r0,#GPEDS0]

    pop {r0,r1}
    subs pc, lr, #4         @Return
```

12. Write a code (Exer12.s) to flash one of the green LEDs by interrupt. The handler routine has to re-program the comparator to provoke a new interrupt. The led has to be turned ON-OFF depending on the previous ON-OFF state. The cadence is 0.20 seconds (0.20 seconds ON, 0.20 seconds OFF).

```
/* Vector Table inicialization */
    mov r0,#0
    ADDEXC 0x1C, fast_interrupt      @only if used

/* Stack init for IRQ mode */
    mov     r0, #0b11010010
    msr     cpsr_c, r0
    mov     sp, #0x8000
/* Stack init for FIQ mode */
    mov     r0, #0b11010001
    msr     cpsr_c, r0
    mov     sp, #0x4000
        mov r8,#0
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que enciende una led verde. 0,20 encendido 0,20 apagado

    @Preparing C1 to send an interrupt after "y" microseconds
    ldr r0,=STBASE
    ldr r1,[r0,#STCLO]
    add r1,#0x20000
    str r1,[r0,#STC3]

    @Enable FIQ
    ldr r0, =INTBASE
    ldr r1,=0x083           @fast interrupt
    str r1,[r0,#INTFIQCON]

    @Enable FIQ (SVC mode)
    mov r1,#0b10010011
    msr cpsr_c, r1

    mov r7,#0
```

```
end:    nop
        nop
        b end

/* Fast interrupt (only if used) */
fast_interrupt:
    push {r0,r1}

    @encendemos si r7=1
    @apagamos si r7=0

    ldr r0,=GPBASE
    ldr r1,=0x400000        @verde
    eors r7,#1
    streq r1,[r0,#GPSET0]
    strne r1,[r0,#GPCLR0]

    @aqui queremos preparar el programa para que pueda haber otro interrupt

    ldr r0,=STBASE
    mov r1,#0b01000         @#0b0010 si estuvieramos en c1
    str r1,[r0,#STCS]

    @tenemos que poner la espera de nuevo porque volvemos a end,
    @no al principio del programa

    ldr r0,=STBASE
    mov r1,#0b01000         @#0b0010 si estuvieramos en c1
    str r1,[r0,#STCS]

    ldr r0,=STBASE
    ldr r1,[r0,#STCLO]
    add r1,#0x20000
    str r1,[r0,#STC3]

    pop {r0,r1,r2}
    subs  pc, lr, #4
```

13.The same as previous exercise, but for the 6 LEDS at the same time

```
/* Vector Table inicialization */
    mov r0,#0
    ADDEXC 0x1C, fast_interrupt      @only if used

/* Stack init for IRQ mode */
    mov     r0, #0b11010010
    msr     cpsr_c, r0
    mov     sp, #0x8000
/* Stack init for FIQ mode */
    mov     r0, #0b11010001
    msr     cpsr_c, r0
    mov     sp, #0x4000
        mov r8,#0
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que enciende una led verde. 0,20 encendido 0,20 apagado

    @Preparing C1 to send an interrupt after "y" microseconds
    ldr r0,=STBASE
    ldr r1,[r0,#STCLO]
    add r1,#0x20000
    str r1,[r0,#STC3]

    @Enable FIQ
    ldr r0, =INTBASE
    ldr r1,=0x083           @fast interrupt
    str r1,[r0,#INTFIQCON]

    @Enable FIQ (SVC mode)
    mov r1,#0b10010011
    msr cpsr_c, r1

    mov r7,#0

end:    nop
        nop
        b end

/* Fast interrupt (only if used) */
fast_interrupt:
    push {r0,r1}

    @encendemos si r7 =1
    @apagamos si r7=0

    ldr r0,=GPBASE
    ldr r1,=0b00001000010000100000111000000000 @todos
    eors r7,#1
    streq r1,[r0,#GPSET0]
    strne r1,[r0,#GPCLR0]

    @aqui queremos preparar el programa para que pueda haber otro interrupt
    ldr r0,=STBASE
    mov r1,#0b01000         @#0b0010 si estuvieramos en c1
    str r1,[r0,#STCS]

    @tenemos que poner la espera de nuevo porque volvemos a end,
    @no al principio del programa
    ldr r0,=STBASE
    mov r1,#0b01000         @#0b0010 si estuvieramos en c1
    str r1,[r0,#STCS]

    ldr r0,=STBASE
    ldr r1,[r0,#STCLO]
    add r1,#0x20000
    str r1,[r0,#STC3]

    pop {r0,r1,r2}
    subs  pc, lr, #4
```

14. Write a code (Exer14.s) that flashes the LEDs in turns with a cadence of 0.25 seconds (each led will be ON for

```
/* Vector Table inicialization */
mov r0,#0
ADDEXC 0x1C, fast_interrupt      @only if used

/* Stack init for FIQ mode */
mov     r0, #0b11010001
msr     cpsr_c, r0
mov     sp, #0x4000
mov r8,#0
/* Stack init for SVC mode */
mov     r0, #0b11010011
msr     cpsr_c, r0
mov     sp, #0x8000000

/* Continue my program here */

@Programa que enciende en cadena (0,25s) todos los leds en un bucle infinito

    @Preparing C3 to send an interrupt after "y" microseconds
    ldr r0, =STBASE
    ldr r1,[r0,#STCLO]
    ldr r7,=250000
    add r1,r1,r7
    str r1,[r0,#STC3]

    @Enable FIQ
    ldr r0, =INTBASE
    ldr r1,=0x83
    str r1,[r0,#INTFIQCON]

    @Enable FIQ (SVC mode)
    mov r1,#0b10010011
    msr cpsr_c,r1

    mov r7, #6

end:    b end
```

```
/* Fast interrupt (only if used) */
fast_interrupt:
push {r0-r6}

    ldr r0,=GPBASE
    ldr r6, =0x08000000
    ldr r5,=0x00400000
    ldr r4,=0x20000
    ldr r3,=0x800
    ldr r2,=0x400
    ldr r1,=0x200

      cmp r7,#6                 @si contador=6 enciende r1 y apaga r6
      streq r1,[r0,#GPSET0]
      streq r6,[r0,#GPCLR0]

      cmp r7,#5
      streq r2,[r0,#GPSET0]
      streq r1,[r0,#GPCLR0]

      cmp r7,#4
      streq r3,[r0,#GPSET0]
      streq r2,[r0,#GPCLR0]

      cmp r7,#3
      streq r4,[r0,#GPSET0]
      streq r3,[r0,#GPCLR0]

      cmp r7,#2
      streq r5,[r0,#GPSET0]
      streq r4,[r0,#GPCLR0]

      cmp r7,#1
      streq r6,[r0,#GPSET0]
      streq r5,[r0,#GPCLR0]

      sub r7,r7,#1              @va disminuyendo el contador

      cmp r7,#0                 @si es 0 vuelve a ponerlo en 6 para empezar de nuevo
      moveq r7,#6
```

```
    @Reset (end) of the timer interrupt by comparator C3
    ldr r0, =STBASE
    mov r1,#0b01000
    str r1,[r0,#STCS]

    @Preparing C3 to send an interrupt after "y" microseconds
    ldr r0, =STBASE
    ldr r1,[r0,#STCLO]
    ldr r8,=250000
    add r1,r1,r8
    str r1,[r0,#STC3]

    pop { r0-r6}
    subs  pc, lr, #4
```

15. Write a code (Exer15.s) that turns on the two red LEDs. After pressing any button, an IQR is generated. The handler routine has to determine what is the pressed button and keep ON only the LED of the same side (this

```
/* Vector Table inicialization */
    mov r0,#0
    ADDEXC 0x18, regular_interrupt @only if used

/* Stack init for IRQ mode */
    mov     r0, #0b11010010
    msr     cpsr_c, r0
    mov     sp, #0x8000
/* Stack init for FIQ mode */
    mov     r0, #0b11010001
    msr     cpsr_c, r0
    mov     sp, #0x4000
        mov r8,#0
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que apaga el led rojo contrario al boton pulsado

    ldr r0,=GPBASE
    ldr r1,=0x0200
    ldr r2,=0x0400
    str r1,[r0,#GPSET0]         @los enciende
    str r2,[r0,#GPSET0]

    @GPED0 Trigged by falling edge (for both GPIO2&3)
    ldr r0,=GPBASE
    mov r1,#0b01100
    str r1,[r0,#GPFEN0]

    @Enable local interrupt IRQ
    ldr r0,=INTBASE
    ldr r1,=0x00100000
    str r1,[r0,#INTENIRQ2]

    @Enable global interrupt IRQ (for SVC mode)
    mov r1,#0b01010011
    msr cpsr_c, r1

end:    b end
```

```
/* Regular interrupt (only if used) */
regular_interrupt:
    push {r0,r1}

    @Check GPIO2 (button) (for GPIO3 use #0b01000)
    ldr r0,=GPBASE
    ldr r1,[r0,#GPEDS0]
    tst r1,#0b00100

    ldr r2,=0x0200
    ldr r3,=0x0400

    strne r3,[r0,#GPCLR0]   @si no esta pulsado apaga

    tst r1,#0b01000
    strne r2,[r0,#GPCLR0]

    @Reset (end) of interrupt IRQ2 (GPIO2&3) (to allow a new interrupt via GPIO2&3)
    ldr r0,=GPBASE
    mov r1, #0b01100
    str r1,[r0,#GPEDS0]

    pop {r0,r1}
    subs  pc, lr, #4
```

16. In this exercise we work with the comparators C1 and C3 and the IRQ simultaneously. With C1 we control the ON state of the LEDs with a cadence similar to the exercise 14, but with a time of 2s instead of 0.25 s. With C3 we control the speaker to produce a continuous sound of 440Hz

```
/* Vector Table inicialization */
    mov r0,#0
    ADDEXC 0x18, regular_interrupt @only if used

/* Stack init for IRQ mode */
    mov     r0, #0b11010010
    msr     cpsr_c, r0
    mov     sp, #0x8000
/* Stack init for FIQ mode */
    mov     r0, #0b11010001
    msr     cpsr_c, r0
    mov     sp, #0x4000
    mov r8,#0
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa q enciende en cadena (2s) los leds (C1) y mientra suena 440Hz (C3)

    mov r7,#0
    mov r6,#0
    mov r5,#0

    @enable timer irq c1
    ldr r0,=INTBASE
    mov r1,#0b0010
    str r1,[r0,#INTENIRQ1]

    @enable timer irq c3
    mov r1,#0b1000
    str r1,[r0,#INTENIRQ1]

    @get time
    ldr r0,=STBASE
    ldr r1,[r0,#STCLO]
```

```
    @prepare c1
    ldr r2,=2000000  @2 seconds
    add r2,r1,r2
    str r2,[r0,#STC1]

    @prepare c3
    add r2,r1,#1136  @440Hz
    str r2,[r0,#STC3]

    mov r1,#0b01010011
    msr cpsr_c,r1

end: b end

/* Regular interrupt (only if used) */
regular_interrupt:
    push {r0,r1,r3}

    ldr r0,=STBASE
    ldr r2,[r0,#STCS]
    tst r2,#0b0010
    bne C1

    @move the membrane of speaker
    ldr r0,=GPBASE
    ldr r1,=0x10
    eors r5,r5,#1
    streq r1,[r0,#GPSET0]
    strne r1,[r0,#GPCLR0]

    @reset timer
    ldr r0,=STBASE
    mov r1,#0b1000
    str r1,[r0,#STCS]

    @prepare c3
    ldr r1,[r0,#STCLO]
    add r2,r1,#1136
    str r2,[r0,#STC3]

    b end_ri
```

```
C1:
    ldr r0,=GPBASE
    cmp r6,#1
    beq led1
    cmp r6,#2
    beq led2
    cmp r6,#3
    beq led3
    cmp r6,#4
    beq led4
    cmp r6,#5
    beq led5

    ldr r1,=0x200
    b cont

led1:
    ldr r1,=0x400
    b cont

led2:
    ldr r1,=0x800
    b cont

led3:
    ldr r1,=0x20000
    b cont

led4:
    ldr r1,=0x400000
    b cont

led5:
    ldr r1,=0x8000000
    b cont

cont: eors r7,r7,#1
    strne r1,[r0,#GPSET0]
    streq r1,[r0,#GPCLR0]
    addeq r6,r6,#1
    cmp r6,#6
    movge r6,#0

    ldr r0,=STBASE
    mov r1,#0b0010
    str r1,[r0,#STCS]

    ldr r1,[r0,#STCLO]
    ldr r2,=2000000
    add r2,r1,r2
    str r2,[r0,#STC1]

end_ri:
    pop {r0,r1,r3}
    subs  pc, lr, #4
```

17. Similar to the previous exercise, but every LED will be associated to a different sound. In this case, each LED (and its associate tone) will be ON for 0.5 s.. To do that, apart from the regular IRQ, we are going to use a fast interrupt (FIQ) in such a way that we have two independent handler routines. C1, which controls the sequence of lighting of the LEDs, will interrupt with a IRQ, whereas C3, which controls the speaker, will work with a FIQ. The interrupt associated to C3 has the highest priority since it will take place more frequently. Next table shows the

```
/* 1 Vector Table inicialization */
    mov r0,#0
    ADDEXC 0x18, regular_interrupt @only if used
    ADDEXC 0x1C, fast_interrupt      @only if used

/* 2 Stack init for IRQ mode */
    mov     r0, #0b11010010
    msr     cpsr_c, r0
    mov     sp, #0x8000
/* 2 Stack init for FIQ mode */
    mov     r0, #0b11010001
    msr     cpsr_c, r0
    mov     sp, #0x4000
        mov r8,#0
/* 3 Stack init for SVC mode    */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

@Programa que enciende en cadena los leds (2s) (C1)(IRQ) y mientras suena un sonido
@distinto por cada led (0,5s)(C3)(FIQ)

    /* 5 The peripheral interrupt is the timer*/
    ldr r0, =STBASE
    ldr r1, [r0, #STCLO]
    ldr r2, =500000
    add r1, r2, r1    @y microseconds
    str r1, [r0, #STC3]

    ldr r0, =STBASE
    ldr r1, [r0, #STCLO]
    ldr r2, =500000
    add r1, r2, r1    @y microseconds
    str r1, [r0, #STC1]
```

```
    /* 6 Enable local FIQ */
    ldr r0, =INTBASE
    ldr r1, =0x083
    str r1, [r0, #INTFIQCON]

    /* 6 Enable local IRQ */
    ldr r0,=INTBASE
    mov r1, #0b0010 @(#0b1000 for C3)
    str r1,[r0,#INTENIRQ1]

    /* 7 Enable global FIQ and IRQ in svc mode */
    mov r1, #0b00010011
    msr cpsr_c, r1


/* 8 infinite loop*/
    mov r7, #0
    mov r6, #0
end:  b end

/* Regular interrupt (only if used)*/
regular_interrupt:
    push {r0,r1}
    /* 3 programita */
    ldr r0, =GPBASE
    ldr r1, =0x8420E00 @ all led
    str r1,[r0, #GPCLR0] @ los apagamos todos

    cmp r7, #0
    ldreq r1, =0x200
    addeq r7, #1
    beq encender

    cmp r7, #1
    ldreq r1, =0x400
    addeq r7, #1
    beq encender

    cmp r7, #2
    ldreq r1, =0x800
    addeq r7, #1
    beq encender
```

```
    cmp r7, #5
    ldreq r1, =0x8000000
    ldreq r7, =0

encender: str r1,[r0, #GPSET0]

    /* 4 reset the timer */
    ldr r0, =STBASE
    mov r1, #0b0010 @for c3 #0b0010 for C1!!!
    str r1,[r0,#STCS]

    ldr r1, [r0, #STCLO]
    ldr r2, =500000
    add r1, r2, r1    @y microseconds
    str r1, [r0, #STC1]

    pop {r0,r1}
    subs  pc, lr, #4


/* Fast interrupt (only if used) */
fast_interrupt:
    push { r0,r1}

    /* 3 programita */
    ldr r0, =GPBASE

    cmp r7, #0
    ldreq r4, =1984
    beq sonar

    cmp r7, #1
    ldreq r4, =1706
    beq sonar

    cmp r7, #2
    ldreq r4, =1515
    beq sonar
```

```
    cmp r7, #3
    ldreq r4, =1432
    beq sonar

    cmp r7, #4
    ldreq r4, =1275
    beq sonar

    cmp r7, #5
    ldreq r4, =1136


sonar: ldr r0, =GPBASE
    ldr r1, =0x010 @( =0b010000)
    eors r6, #1
    streq r1,[r0, #GPSET0]
    strne r1,[r0, #GPCLR0]

    /* 4 reset the timer */
    ldr r0, =STBASE
    mov r1, #0b01000 @for c3 #0b0010 for C1!!!
    str r1,[r0,#STCS]

    ldr r1, [r0, #STCLO]
    add r1, r4, r1    @y microseconds
    str r1, [r0, #STC3]

    pop { r0,r1}
    subs  pc, lr, #4
```

EJERCICIOS PARCIALES

PARCIAL 2018

DIRECTIONS FOR THE EXAM

User: EC001, psw: labec

First of all, **check that your Raspberry works** by using the Full_test.s program.

Then, download all the files that you want to use from your pendrive to the hard disc (after 5 minutes, no any further download will be permitted). Mobile phone is also prohibited for the exam.

After this, you can start with the exam. Create your programs in the folder d:/EC/RBPi

You have to make two independent programs. The name of the first one is your initials followed by 1 (first program) and followed by 2 for the second program. For example, if your name is Juan Lopez Gomez, the name will be jlg1.s and jlg2.s. At the end of the exam, upload both programs to the virtual campus.

Once you finish the first exercise, please call me to assess your program (the same for the second program).

EXAM:

1) (First program) Write a program that performs the following tasks (up to 4 points):
    a. Initially, turn on all leds (1 pt)
    b. After pressing button 1, the leds will turn off in turns starting by the led on GPIO 9 (red led) every 200 ms. (+2 pts).
    c. Then (when all leds turn off), emit a sound of 1 KHz (+1 pts)

```
/* Basic skeleton for programs using ports (without interruptions) */

.include "configuration.inc"
.include "symbolic.inc"

/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

    ldr r0, =GPBASE
    ldr r2, =0x8020400 @All leds
    str r2,[r0,#GPSET0]

    ldr r5, =0x200              @Red led 1
    ldr r6, =0x400              @Red led 2
    ldr r8, =0x800              @Yellow led 1
    ldr r9, =0x20000            @Yellow led 2
    ldr r10, =0x00400000        @Green led 1
    ldr r11, =0x08000000        @Green led 2

    ldr r1,[r0,#GPLEV0]
    tst r1,#0b00100             @pulsado
    ldr r4,=200000
    bl wait1
    streq r5,[r0,#GPCLR0]
    ldr r4,=200000
    bl wait1
    streq r6,[r0,#GPCLR0]
    ldr r4,=200000
    bl wait1
    streq r8,[r0,#GPCLR0]
    ldr r4,=200000
    bl wait1
    streq r9,[r0,#GPCLR0]
    ldr r4,=200000
    bl wait1
    srteq r10,[r0,#GPCLR0]
    ldr r4,=200000
    bl wait1
    streq r11,[r0,#GPCLR0]

    ldr r12,=0x010
    str r12,[r0,#GPSET0]
    bl wait2
    str r12,[r0,#GPCLR0]
    bl wait2

wait1:  ldr r7,=STBASE
        ldr r3,[r7,#STCLO]
        add r4,r3,r4
        ret1:   ldr r3,[r7,#STCLO]
                cmp r3,r4
                blt ret1
                bx lr

wait2:  ldr r7,=STBASE
        ldr r3,[r7,#STCLO]
        ldr r4,=500
        add r4,r3,r4

end:    b end
```

2) (Second program) Write a program that performs the following tasks (up to 4 points):
   a. A sound of 5 kHz will be emitted for just one second (1 pt)
   b. After this, a sound of 2500 Hz will be emitted for 2 seconds and then a sound of 500 Hz will be emitted for three seconds, and repeat this sequence forever (5KHz-1s., 2500Hz-2s.,500Hz-3s.) (+1 pts)

      Do one of these two options:
   c. Implement one of the next two options (different points for both)
      i. If any button is pressed AT ANY TIME, then all leds will turn on just at that time (+1) and the sound keeps as usual (+ 1 pt)
      ii. If the left button 1 is pressed at AT ANY TIME, then the three most left leds will turn on just at that time, whereas if the right button is pressed, the three most right leds will turn on just at that time. In both cases, the sequence of sounds will keep unchanged (+2 pts).

When you finish, your programs to the virtual campus. The name of each program is your initials1.s and your initial2.s. For example, if you name is Juan Lopez Gomez, the name will be jlg1.s and jlg2.s

```
/* Basic skeleton for programs using interrupts */

.include "configuration.inc"
.include "symbolic.inc"

/* Vector Table inicialization */
    mov r0,#0
    ADDEXC 0x18, regular_interrupt @only if used
    ADDEXC 0x1C, fast_interrupt      @only if used


/* Stack init for IRQ mode */
    mov     r0, #0b11010010
    msr     cpsr_c, r0
    mov     sp, #0x8000
/* Stack init for FIQ mode */
    mov     r0, #0b11010001
    msr     cpsr_c, r0
    mov     sp, #0x4000
        mov r8,#0
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

        ldr r5,=1000000
        ldr r6,=0
        ldr r7,=0

        @STC1
        ldr r0,=GPBASE
        ldr r1,[r0,#STCLO]
        add r1,r1,#1000000
        str r1,[r0,#STC1]

        @STC3
        ldr r0,=STBASE
        ldr r1,[r0,#STCLO]
        add r1,r1,#500
        str r1,[r0,#STC3]

        @Enable C1 IRQ
        ldr r0,=INTBASE
        mov r1,#0b1010
        str r1,[r0,#INTENIRQ1]

        @Enable C3 FIQ
        ldr r0,=INTBASE
        ldr r1,=0x083
        str r1,[r0,#INTFIQCON]

        @IRQ and FIQ
        mov r1,#0b00010011
        msr cpsr_c, r1

end:    b end

/* Regular interrupt (only if used) */
regular_interrupt:
    push {r0,r1,r2}
        push {LR}
            ldr r0,=STBASE
            ldr r2,[r0,#STCS]
            tst r2,#0b0010
            blne waiting
        pop {LR}
    pop {r0,r1,r2}
    subs  pc, lr, #4
```

```
waiting: push{r0-r4}
            push{LR}
                add r7,#1

                cmp r7,#0
                moveq r5,#1000000

                cmp r7,#1
                addeq r5,#1000000

                cmp r7,#2
                addeq r5,#1000000
                moveq r7,#0

                @STC1
                ldr r0,=STBASE
                mov r1,#0b0010
                str r1,[r0,#STCS]

                ldr r0,=STBASE
                ldr r1,[r0,#STCLO]
                add r1,r1,r5
                str r1,[r0,#STC1]

            pop{LR}
        push{r0-r4}
        bx lr

/* Fast interrupt (only if used) */
fast_interrupt:
    push {r0,r1,r2}
        push {LR}
            ldr r0,=STBASE
            ldr r2,[r0,#STCS]
            tst r2,#0b1000
            blne speaker
        pop {LR}

    pop {r0,r1,r2}
    subs  pc, lr, #4
```

```
speaker: push {r0-r5}
            ldr r0,=GPBASE
            eors r6,#1
            streq r1,[r0,#GPSET0]
            strne r1,[r0,#GPCLR0]

            @STC3
            ldr r0,=STBASE
            mov r1,#0b1000
            str r1,[r0,#STCS]

            cmp r7,#0
            moveq r4,#100

            cmp r7,#1
            moveq r4,#200

            cmp r7,#2
            moveq r4,#1000

            ldr r0,=GPBASE
            ldr r1,[r0,#STCLO]
            add r1,r1,r4
            str r1,[r0,#STC3]

            pop {r0-r5}
            bx lr
```

PARCIAL 2017

1) Write a program that carries out the following tasks (up to 5 points):
   a. Initially, the first yellow have to be ON (1 pt)
   b. Turn ON the second red LED while you are pressing the first push button (+2 pt)
   c. After pressing the second push button, a sound of 2 KHz has to be produced permanently, and one of the green leds will turn on (+2 pts)

```
/* Stack init for SVC mode  */
    mov     r0, #0b11010011
    msr     cpsr_c, r0
    mov     sp, #0x8000000

/* Continue my program here */

    ldr r0, =GPBASE
    ldr r2, =0x800      @Yellow led 1
    ldr r5, =0x600      @Red led 2
    ldr r6, =0x         @Green led
    ldr r8, =0x010      @Speaker
    str r2,[r0,#GPSET0]

loop:   ldr r1,[r0,#GPLEV0]
    tst r1,#0b00100
    streq r5,[r0,#GPSET0]

    ldr r1,[r0,#GPLEV0]
    tst r1,#0b01000
    streq r8,[r0,#GPSET0]
    bl wait
    streq r6,[r0,#GPSET0]
    b loop


wait:   ldr r7,=STBASE
        ldr r3,[r7,#STCLO]
        ldr r4,=250
        add r4,r3,r4

        ret1:   ldr r3,[r7,#STCLO]
                cmp r3,r4
                blt ret1
                bx lr

end:   b end
```