

SISTEMAS BASADOS EN MICROPROCESADORES

Grado en Ingeniería Informática Escuela Politécnica Superior – UAM

COLECCIÓN DE PROBLEMAS DE LOS TEMAS 5.5 A 7.2

P1. Escribir una rutina de ensamblador que se quede **bloqueada en un bucle hasta que el bit de petición de interrupción (IRQF) del RTC se ponga a uno**. Se valorará la eficiencia del código.

Esperar_IRQF PROC

```
push ax

itera: mov al, 0Ch
        out 70h, al      ; Pide leer registro C del RTC
        in al, 71h       ; Lee el registro C del RTC
        test al, 10000000b ; IRQF es bit de más peso
        jz itera         ; Itera mientras IRQF sea cero

pop ax
ret
```

Esperar_IRQF ENDP

P2. Escribir una rutina de ensamblador que **lea en el registro DX la dirección base del puerto paralelo LPT2**. Se valorará la eficiencia del código.

Leer_LPT2 PROC

```
push es

xor dx, dx
mov es, dx
mov dx, es:[040Ah] ; Lee la dirección base de LPT2 en la BIOS

pop es
ret
```

Leer_LPT2 ENDP

P3. Usando la BIOS (se adjunta un extracto de la documentación de Ralph Brown), escribir en ensamblador el código de la función de C **char Keystroke()**, que **retorna un 1 si se ha pulsado alguna tecla o un 0 si no se ha pulsado**. Se valorará la eficiencia del código.

_Keystroke PROC

```
mov ah, 1
int 16h

jnz retorna1
```

```

; No hay tecla
mov ax, 0
ret

retorna1: ; Hay tecla
mov ax, 1
ret

_Keystroke ENDP

```

KEYBOARD - CHECK FOR KEYSTROKE

AH = 01h

Return:

ZF set if no keystroke available

ZF clear if keystroke available

AH = BIOS scan code

AL = ASCII character

Category: [Bios](#) - [Int 16h](#) - [K](#)

P4. Escribir en ensamblador el código que sea necesario para ejecutar un EOI en los manejadores de las siguientes interrupciones:

76h (Disco Dur o)

02h (NMI)

0Ch (COM1)

<pre> mov al, 20h out 20h, al out A0h, al </pre>	EOI innecesario	<pre> mov al, 20h out 20h, al </pre>
--	------------------------	--------------------------------------

P5. Escribir una subrutina de ensamblador que **programe la oscilación del RTC a 512 interrupciones por segundo**. Se valorará la eficiencia del código.

RTC_512 PROC

```

push ax
mov al, 0Ah
out 70h, al
mov al, 00100111b ; DV = 010 (32768 Hz), RS = 0111 (512 Hz)
out 71h, al
pop ax
ret

```

RTC_512 ENDP

P6. Usando la BIOS (se adjunta un extracto de la documentación de Ralph Brown), escribir en ensamblador el código de la función de C **void Imprimir_Letra(char ascii)**, que **imprime en la posición actual del cursor la letra de código ASCII indicado, usando color rojo intenso sobre fondo verde parpadeante**. La controladora de vídeo está configurada en modo CGA de texto. Todas las direcciones son cercanas. Se valorará la eficiencia del código.

_Imprimir_Letra PROC NEAR

```

push bp
mov bp, sp
push ax
push bx
push cx

mov al, 4[bp]
mov bl, 10101100b
xor bh, bh
mov cx, 1

```

VIDEO - WRITE CHARACTER AND ATTRIBUTE AT CURSOR POSITION

AH = 09h

AL = character to display

BH = page number

BL = attribute (text mode)

CX = number of times to write character

Return:

Nothing

Category: [Video](#) - [Int 10h](#) - [V](#)

```
mov ah, 09h
int 10h
```

```
pop cx
pop bx
pop ax
pop bp
ret
```

`_Imprimir_Letra ENDP`

P7. Implementar en ensamblador de 8086 la subrutina lejana `imprimir_ASCIIIZ`, que ha de imprimir por el puerto paralelo LPT1 una cadena ASCIIIZ cuya dirección recibe en el registro BX. La impresión debe realizarse siguiendo el protocolo BUSY. Se valorará la eficiencia del código.

`imprimir_ASCIIIZ PROC FAR`

```
; Imprime por el LPT1 con protocolo BUSY la
; cadena ASCIIIZ cuya dirección recibe en BX.
```

```
push ax bx dx es
```

```
mov ax, 0
mov es, ax
mov dx, es:[0408h] ; Lee en dx la dirección base del LPT1
```

`continuar:`

```
mov al, [bx]
cmp al, 0
jz final ; Acaba por haber llegado a final de cadena
```

```
call espera_libre ; Espera a que impresora no esté ocupada
call imprimir_caracter ; Imprime carácter contenido en al
```

```
inc bx
jmp continuar
```

`final:`

```
pop es dx bx ax
ret
```

`imprimir_ASCIIIZ ENDP`

`espera_libre PROC FAR`

```
; Espera activa mientras señal #BUSY = 0. Recibe
; en DX la dirección base del puerto paralelo
```

```
push ax dx
```

```
inc dx ; Obtiene dirección del registro de estado
```

`ocupada:`

```
in al, dx ; Lee registro de estado
test al, 10000000b ; Comprueba bit 7 (#BUSY)
```

```

    jz ocupada          ; Impresora ocupada si #BUSY = 0

    ; Impresora ya está libre

    pop dx ax
    ret

espera_libre ENDP

espera_ocupada PROC FAR
    ; Espera activa mientras señal #BUSY = 1. Recibe
    ; en DX la dirección base del puerto paralelo

    push ax dx

    inc dx              ; Obtiene dirección del registro de estado

libre:
    in al, dx           ; Lee registro de estado
    test al, 10000000b  ; Comprueba bit 7 (#BUSY)
    jnz libre           ; Impresora libre si #BUSY = 1

    ; Impresora ya está ocupada

    pop dx ax
    ret

espera_ocupada ENDP

imprimir_caracter PROC FAR
    ; Recibe en AL el código ASCII del carácter que
    ; se debe enviar y en DX la dirección base del
    ; puerto paralelo. Debe generar un
    ; pulso positivo de STROBE.

    push ax dx

    out dx, al          ; Envía carácter a registro de datos

    ; Genera pulso positivo de STROBE

    add dx, 2           ; Obtiene dirección de registro de control
    in al, dx           ; Lee registro de control

    or al, 00000001b
    out dx, al          ; STROBE = 1

    sub dx, 2           ; Obtiene dirección base
    call espera_ocupada ; Espera activa hasta que la impresora pasa
                        ; a estar ocupada.
    add dx, 2           ; Obtiene dirección de registro de control

    and al, 11111110b
    out dx, al          ; STROBE = 0

```

```
pop dx ax
ret
```

```
imprimir_caracter ENDP
```

P8. Implementar en ensamblador de 8086 la subrutina lejana enviar_ASCIIIZ, que ha de enviar por el puerto serie COM1 una cadena ASCIIIZ cuya dirección recibe en el registro BX. La transmisión ha de realizarse a 9600 bps, 8 bits, 1 bit de parada y paridad par. Se valorará la eficiencia del código.

```
enviar_ASCIIIZ PROC FAR
```

```
; Envía por el COM1 a 9600 bps, 8 bits, 1 bit de parada y
; paridad par la cadena ASCIIIZ cuya dirección recibe en BX.
```

```
push ax bx dx es
```

```
mov ax, 0
mov es, ax
mov dx, es:[0400h] ; Lee en dx la dirección base del COM1
```

```
; Configura puerto serie a 9600 bps, 8 bits, 1 bit de parada
; y paridad par.
```

```
add dx, 3 ; Obtiene dirección de reg. de control de línea (LCR)
mov al, 10011011b ; D7=1 (DLAB=1), D6=D5=0, D4=1 (paridad par),
; D3=1 (paridad habilitada)
; D2=0 (1 bit de parada), D1=D0=1 (8 bits)
out dx, al ; Escribe registro de control de línea (LCR)
```

```
sub dx, 3 ; Obtiene dirección de reg. divisor de frecuencia (DLL)
; 9600 bps = 0.009600 Mbps => DLL = 12, DLH = 0
```

```
mov al, 12
out dx, al ; Escribe registro DLL
```

```
inc dx ; Obtiene dirección de registro DLH
mov al, 0
out dx, al ; Escribe registro DLH
```

```
add dx, 2 ; Obtiene dirección de reg. de control de línea (LCR)
in al, dx ; Lee registro de control de línea (LCR)
and al, 01111111b ; D7=0 (DLAB=0)
out dx, al ; Escribe registro de control de línea (LCR)
```

```
sub dx, 3 ; Recupera en dx la dirección base de COM1
; Puerto configurado
```

```
continuar:
```

```
mov al, [bx]
cmp al, 0
jz final ; Acaba por haber llegado a final de cadena
```

```

    call espera_enviar    ; Espera a que reg. de salida (THR) esté vacío.
    call enviar_caracter ; Envía carácter contenido en AL a THR.

    inc bx
    jmp continuar

final:
    pop es dx bx ax
    ret

enviar_ASCII ENDP

espera_enviar PROC FAR
    ; Espera activa mientras registro de salida (THR) no esté vacío.
    ; Recibe en DX la dirección base del puerto serie.

    push ax dx

    add dx, 5      ; Obtiene dirección de reg. de estado de línea (LSR)

llenado:
    in al, dx      ; Lee registro de estado de línea (LSR)
    test al, 00100000b ; Comprueba bit 5 (THR vacío)
    jz llenado     ; THR no está vacío

    pop dx ax
    ret

espera_enviar ENDP

enviar_caracter PROC FAR
    ; Recibe en AL el código ASCII del carácter que
    ; se debe enviar y en DX la dirección base del
    ; puerto serie.

    out dx, al     ; Envía carácter a registro de salida (THR)

    ret

enviar_caracter ENDP

```

P9. Escribir en ensamblador de 80x86 el código necesario para que el puerto paralelo LPT2 genere interrupciones. Se valorará la eficiencia del código.

```

    mov ax, 0
    mov es, ax
    mov dx, es:[040Ah] ; Lee dirección base de LPT2 desde BIOS
    add dx, 2          ; Calcula dirección de registro de control
    in al, dx          ; Lee registro de control
    or al, 00010000b   ; Activa bit 4 (IRQEN)
    out dx, al         ; Modifica registro de control

    ; A partir de aquí, la señal #ACK del LPT2 activa IRQ5 de
    ; PIC maestro.
    ; Para que estas interrupciones lleguen a la CPU es necesario

```

```

; que el bit 5 del registro de máscara IMR del PIC maestro
; esté a cero, que es su valor por defecto tras la inicialización
; del PIC.

```

P10. Escribir en ensamblador de 80x86 una subrutina que inicie la emisión a través del altavoz del PC de un sonido de frecuencia lo más próxima posible a **440 Hz usando el temporizador (timer)**. Se valorará la eficiencia del código.

```

tocar_440Hz PROC FAR
    push ax

    ; Genera palabra de control
    ; SC = 10 (contador 2), RW = 11 (byte bajo + byte alto)
    ; M = 011 (modo 3: onda cuadrada); BCD = 0 (binario)

    mov al, 10110110b    ; Palabra de control SC|RW|M|BCD
    out 43h, al          ; Configura timer

    ; Valor inicial = 1193182 / 440 = 2711,77 ≈ 2712
    ; Frecuencia real = 1193182 / 2712 = 439,96 Hz
    ; 2712 = 10 * 256 + 152

    mov ax, 2712
    out 42h, al          ; Envía byte bajo de valor inicial (152)
    mov al, ah
    out 42h, al          ; Envía byte alto de valor inicial (10)

    ; Activa salida del altavoz y puerta del contador 2

    in al, 61h           ; Lee registro de control
    or al, 00000011b     ; Activa bit 0 (puerta) y bit 1 (salida)
    out 61h, al

    ; Altavoz empieza a sonar

    pop ax
    ret

tocar_440Hz ENDP

```

P11. Usando la BIOS (se adjunta un extracto de la documentación de Ralph Brown), escribir en ensamblador de 80x86 una subrutina que posicione el cursor de modo texto en la fila y columna de la página cero especificadas en el registro AX (**AH = columna, AL = fila**). Se valorará la eficiencia del código.

```

set_cursor PROC FAR
    push ax bx dx

    mov dl, ah    ; Define columna
    mov dh, al    ; Define fila
    mov bh, 0     ; Define página 0
    mov ah, 2     ; Código de operación
    int 10h       ; BIOS vídeo

    pop dx bx ax
    ret

```

VIDEO - SET CURSOR POSITION

AH = 02h
 BH = page number
 0-3 in modes 2&3
 0-7 in modes 0&1
 0 in graphics modes
 DH = row (00h is top)
 DL = column (00h is left)

Return: Nothing

set_cursor ENDP

P12. Escribir en ensamblador de 80x86 una **rutina de servicio a la interrupción** del reloj de tiempo real (RTC), que llame a la subrutina _Actualizar cada vez que se reciba una **interrupción de actualización de la hora/fecha**. Se supone que el RTC tiene habilitadas todas sus interrupciones. Se valorará la eficiencia del código.

```
    rsi_RTC PROC FAR

        sti                ; Activa interrupciones
        push ax

        ; Lee registro C del RTC (banderas de interrupción)

        mov al, 0Ch
        out 70h, al
        in al, 71h

        test al, 00010000b    ; Comprueba bandera UF (Update Flag)
        jz final              ; UF = 0 <==> no hay actualización

        call _Actualizar

final:    mov al, 20h
        out 20h, al          ; EOI al maestro
        out 0A0h, al         ; EOI al esclavo

        pop ax
        iret

    rsi_RTC ENDP
```