

SISTEMAS BASADOS EN MICROPROCESADORES

Grado en Ingeniería Informática

Examen Final Ordinario – Curso 2010/11

NOMBRE : _____ DNI : _____
 APELLIDOS : _____

P1. Suponiendo que **CS=2000h**, **DS=4000h**, **ES=424Dh**, **SS=424Eh**, **BX=0004h**, **BP=0000h** y **DI=24E0h**, Indicar el valor del registro **AX** tras ejecutar cada una de las instrucciones siguientes (independientes entre ellas), dado el volcado de memoria adjunto. Expresar los dígitos hexadecimales desconocidos de **AX** con un '?'. (1,5 puntos)

424E:0000 4E 42 74 61 20 65 73 20

| | |
|---------------------------------|-------------------------|
| <code>mov AX, [BX]</code> | <code>AX = ????h</code> |
| <code>mov AX, DS:[0000h]</code> | <code>AX = ????h</code> |
| <code>mov AH, [BP + 3]</code> | <code>AX = 61??h</code> |
| <code>mov AL, ES:[BP]15</code> | <code>AX = ????h</code> |
| <code>mov AX, 2[DI]</code> | <code>AX = 6174h</code> |

P2. Al inicio de la ejecución de una función invocada desde lenguaje C, se tiene que **SP=8** y que las 16 primeras posiciones de la pila contienen los siguientes valores:

| | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 11h | A0h | 25h | 00h | 32h | 00h | A2h | E9h | 00h | C1h | 24h | 00h | 63h | 00h | 41h | 12h |

La signatura de dicha función es: `int fun (char p, int n);`

Indicar el valor de los dos parámetros con que esa función fue invocada desde C, tanto cuando todas las direcciones son cercanas (**NEAR**), como cuando son lejanas (**FAR**). (2 puntos)

Caso NEAR: `p = 24h` `n = 0063h`

Caso FAR: `p = 63h` `n = 1241h`

P3. Escribir en ensamblador de 80x86 el código necesario para que el puerto paralelo LPT2 genere interrupciones. Se valorará la eficiencia del código. (1,5 puntos)

```
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.

```

P4. 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. (2 puntos)

```

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

```

P5. Escribir en ensamblador de 80x86 utilizando instrucciones básicas (sin instrucciones de manipulación de cadenas ni de bucles) la función `sumatorio` de C cuyo código se reproduce a continuación. Esta función calcula de forma recursiva el sumatorio de los n primeros números naturales, con n siendo el entero de 32 bits que recibe como argumento. Se supone que la función no detecta desbordamiento del resultado y que el programa en C está compilado en **modelo largo**. Se valorará la eficiencia del código. (3 puntos)

```

long sumatorio( long n )
{
    if (n == 1) return 1;
    else return n + sumatorio( n-1 );
}

```

_sumatorio PROC FAR

```

push bp
mov bp, sp

; Accede a parámetro de entrada de 32 bits (n)
mov ax, [bp+6] ; AX <= Parte baja de n
mov dx, [bp+8] ; DX <= Parte alta de n

cmp dx, 0
jne noes1 ; n != 1
cmp ax, 1
je final ; n == 1 => Retorna 1 en DX:AX

; n != 1

noes1: ; Decrementa n

dec ax ; Decrementa parte baja
sbb dx, 0 ; Resta acarreo (borrow) a parte alta

; Apila n-1 (parte alta primero) y llama recursivamente

push dx ax
call _sumatorio ; Llamada recursiva
add sp, 4 ; Reequilibra la pila

; sumatorio( n-1 ) retornado en DX:AX

; DX:AX := DX:AX + n

add ax, [bp+6] ; Suma parte baja
adc dx, [bp+8] ; Suma parte alta y acarreo

final: pop bp
ret

_sumatorio ENDP

```