

SISTEMAS BASADOS EN MICROPROCESADORES

Grado en Ingeniería Informática

Examen Final Extraordinario – Curso 2010/11

NOMBRE : _____ DNI : _____
 APELLIDOS : _____

P1. Suponiendo que **CS=2000h**, **DS=424Eh**, **ES=424Eh**, **SS=424Eh**, **BX=0004h**, **BP=000Ah** y **DI=000Ah**, 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:0008 FF A0 23 56 45 3A 30 2E

<code>mov AX, [BX]</code>	<code>AX = ????h</code>
<code>mov AX, DS:[BX][DI]</code>	<code>AX = 2E30h</code>
<code>mov AL, [BP + 1]</code>	<code>AX = ??56h</code>
<code>mov AX, ES:5[BP]</code>	<code>AX = ??2Eh</code>
<code>mov AH, [DI]</code>	<code>AX = 23??h</code>

P2. Al inicio de la ejecución de una función invocada desde lenguaje C, se tiene que **SP=0** 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 (long p, int n);`

Indicar el valor de los tres 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 = 00320025h n = E9A2h`

Caso FAR: `p = E9A20032h n = C100h`

P3. 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. (1,5 puntos)

```

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
set_cursor ENDP

```

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

P4. 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. (2 puntos)

```

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

```

P5. Escribir en ensamblador de 80x86 utilizando instrucciones básicas (sin instrucciones de manipulación de cadenas ni de bucles) la función `maximo` de C cuyo código se reproduce a continuación. Esta función determina el valor máximo de una tabla de enteros con signo que recibe como argumento. Se supone que el programa en C está compilado en **modelo pequeño** (*small*). Se valorará la eficiencia del código. (3 puntos)

```

int maximo( int n, int *tabla )
{
    int i, max;

    max = tabla[0];

    for (i=1; i<n; i++)
        if (tabla[i] > max)
            max = tabla[i];

    return max;
}

```

```

_maximo PROC NEAR

    push bp
    mov bp, sp
    push bx cx

    mov bx, [bp+6]    ; bx == &tabla[i] (inicialmente i = 0)
    mov ax, [bx]     ; ax == max
    mov cx, 1        ; cx == i

bucle:    cmp cx, [bp+4];    ; ¿i == n?
          je final

          add bx, 2
          cmp [bx], ax     ; ¿tabla[i] > max?
          jle no_maximo    ; tabla[i] <= max

          mov ax, [bx]     ; max = tabla[i]

no_maximo: inc cx          ; i++
           jmp bucle

final:    pop cx bx
          ret

_maximo ENDP

```