

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES II (2º)

Ingeniería Informática – Escuela Politécnica Superior – UAM

Parcial - Curso 09-10

NOMBRE : _____ DNI : _____
APELLIDOS : _____

P1. Suponiendo que **CS=0000h**, **DS=1000h**, **ES=FFFFh**, **SS=2000h**, **BX=2222h**, **BP=0000h** y **SI=0002h**, indicar la **dirección física** de memoria (@) a la que se está accediendo en cada una de las siguientes instrucciones, considerando los registros de segmento por defecto. (1,5 puntos)

mov AH, ES:16[SI]	@ = 00002h
mov AH, 16[SI]	@ = 10012h
mov AL, [BP - 2]	@ = 2FFFEh
mov AL, CS:[FFFFh]	@ = 0FFFFh
mov AL, DS:[BP - 1]	@ = 1FFFFh

P2. Suponiendo que **CS=2000h**, **DS=204Fh**, **ES=204Fh**, **SS=2000h**, **BX=0001h**, **BP=04F8h**, **DI=0007h** y **SP=04F8h**, Indicar el valor del **registro AX** tras ejecutar cada una de las instrucciones siguientes (**independientes entre sí**), dado el volcado de memoria adjunto. Expresar los **dígitos hexadecimales desconocidos de AX con un '?'**. (1,5 puntos)

```
204F:0000 73 65 67 20 00 68 61 6E
204F:0008 12 34 4E 00 FF 00 33 11
```

mov AH, [BX][DI]	AX = 12??h
mov AL, 3[DI]	AX = ??4Eh
mov AX, [BP - 6]	AX = 2067h
pop AX	AX = 3412h
mov AX, 16[BX]	AX = ????h

P3. Suponiendo que **SS=424Dh**, **SP=14**, **AX=3412h** y **BX=5678h**, indicar el **valor hexadecimal de los 16 primeros bytes del segmento SS** una vez ejecutado el siguiente programa. (1 punto)

```
push AX
pop BX
push SS
push BX
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
										12h	34h	4Dh	42h		

P4. Declarar mediante directivas de ensamblador de 8086 las mismas variables que aparecen en el siguiente extracto en lenguaje C, teniendo en cuenta que las cadenas de caracteres en C acaban con el byte 0. (1,5 puntos)

```

char nombre[20];          // Cadena de caracteres de 20 bytes
short edad = 17;         // Entero de 2 bytes inicializado
char tabla2D[10][2];     // Tabla de bytes de 10 filas por 2 columnas
short valores[5] = { 1, 2, 3, 4, 5 };
char despedida[20] = "Hasta luego";

    nombre db 20 dup (?)
    edad dw 17
    tabla2D db 10*2 dup (?)
    valores dw 1, 2, 3, 4, 5
    despedida db "Hasta luego", 0, 20-($-despedida) dup (?)

```

P5. El siguiente programa en lenguaje ensamblador de 8086, que debe **invertir el orden de los caracteres de una cadena dada de 512 bytes como máximo**, tiene varios errores. Proponer una versión correcta del mismo programa haciendo el **menor número de cambios**. Sólo es necesario reescribir las líneas erróneas. (3 puntos)

```

datos segment
    cadena    dw "Hola"
    longitud  db  cadena-$
datos ends

resultados segment
    resultado db 200 dup (?)
resultados ends

codigo segment
    assume cs:codigo, ds:datos
    invertir proc far
        mov ax, datos
        mov ds, ax
        mov ax, resultado
        mov es, ax
        mov si, longitud
        mov di, 0
seguir:    mov al, cadena[si-1]
        mov resultado[di], al
        dec si
        inc di
        jz seguir
        mov ax, 4C00h
        int 21h
    invertir endp
codigo ends
end codiqo

```

```

datos segment
    cadena    db "Hola"
    longitud  dw  $-cadena
datos ends

resultados segment
    resultado db 200h dup (?)
resultados ends

codigo segment
    assume cs:codigo, ds:datos, es:resultados
    invertir proc far
        mov ax, datos
        mov ds, ax
        mov ax, resultados
        mov es, ax
        mov si, longitud
        mov di, 0
seguir:    mov al, cadena[si-1]
        mov resultado[di], al
        inc di
        dec si
        jnz seguir
        mov ax, 4C00h
        int 21h
    invertir endp
codigo ends
end invertir

```

P6. Escribir en ensamblador un procedimiento lejano (descontar2_32) que **decremente en dos unidades** el valor de la variable de 32 bits cuya dirección recibe mediante los registros AX y BX tal como se indica en el código adjunto. Tras su ejecución, este procedimiento no deberá alterar los valores previos de ningún registro del banco general ni de segmento. Se valorará la eficiencia del código. (1,5 puntos)

```
datos segment
    contador          dd  0FFFFFFFFh
datos ends

...

mov ax, OFFSET contador
mov bx, SEG contador

call descontar2_32

descontar2_32 PROC far

    push bx es

    mov es, bx
    mov bx, ax

    sub WORD PTR es:[bx], 2
    sbb WORD PTR es:[bx+2], 0

    pop es bx

    ret

descontar2_32 ENDP
```