

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES II (2º)

Ingeniería Informática – Escuela Politécnica Superior – UAM

Final Septiembre Teoría - Curso 09-10

NOMBRE : _____ DNI : _____
APELLIDOS : _____

P1. Suponiendo que **CS=0001h**, **DS=1000h**, **ES=FFFFh**, **SS=2000h**, **BX=2222h**, **BP=0000h** y **DI=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)

<code>mov AH, ES:[DI]</code>	@ = FFFF2h
<code>mov AH, [DI]</code>	@ = 10002h
<code>mov AL, [BP + 2]</code>	@ = 20002h
<code>mov AL, CS:[000Fh]</code>	@ = 0001Fh
<code>mov AL, DS:[BP - 2]</code>	@ = 1FFFEh

P2. Suponiendo que **CS=2000h**, **DS=204Fh**, **ES=204Dh**, **SS=2222h**, **BX=0020h**, **SI=0002h** y **DI=0002h**, 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

<code>mov AX, [SI]</code>	AX = 2067h
<code>mov AH, 3[DI]</code>	AX = 68??h
<code>mov AL, ES:[BX + 5]</code>	AX = ??68h
<code>mov AX, ES:[20h]</code>	AX = 6573h
<code>mov AX, [SI][DI]</code>	AX = 6800h

P3. Si **SP=0004h** y **FLAGS=0210h** al inicio de la ejecución del código que se adjunta, indicar los valores contenidos en las **primeras seis posiciones de la pila** al ejecutar la primera instrucción del procedimiento `Leer_Datos`, tanto cuando todos los procedimientos del programa son cercanos (**NEAR**), como cuando son lejanos (**FAR**). La pila está inicializada a ceros. (1 punto)

```
20FF:4000 E8A8FD    call Leer_Datos
20FF:4003 89161000    mov Datos[0], dx
```

0	1	2	3	4	5
0	0	03h	40h	0	0

Caso NEAR

0	1	2	3	4	5
03h	40h	FFh	20h	0	0

Caso FAR

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 y que el tipo `short` ocupa 2 bytes. (1,5 puntos)

```
char caracter = 'A';
short edad;
short tabla[100];
char valores[5] = { 1, 2, 3, 4, 5 };
char despedida[12] = "Hasta luego";
```

```
caracter db 'A'
edad dw ?
tabla dw 100 dup (?)
valores db 1, 2, 3, 4, 5
despedida db "Hasta luego", 0
```

P5. El siguiente programa en lenguaje ensamblador de 8086, que debe **contar el número de 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 db "Hola", 0
datos ends

resultados segment
    resultado db 2 dup(?)
resultados ends

codigo segment
    assume cs:codigo, ds:datos, es:resultados

    contar proc far
        mov ax, cadena
        mov ds, ax
        mov ax, resultados
        mov es, ax
        mov si, 4
seguir: mov cadena[si], 0
        jz fin
        dec si
        jmp fin
fin:    mov resultado, si
        mov ax, 4C00h
        int 21h
    contar endp
codigo ends
end contar
```

```
datos segment
    cadena db "Hola", 0
datos ends

resultados segment
    resultado db 2 dup(?)
resultados ends

codigo segment
    assume cs:codigo, ds:datos, es:resultados

    contar proc far
        mov ax, datos
        mov ds, ax
        mov ax, resultados
        mov es, ax
        mov si, 0
seguir: cmp cadena[si], 0
        jz fin
        inc si
        jmp seguir
fin:    mov WORD PTR resultado, si
        mov ax, 4C00h
        int 21h
    contar endp
codigo ends
end contar
```

P6. Escribir en ensamblador un procedimiento lejano (contar4_48) que **incremente en cuatro unidades** el valor de la variable de 48 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 db 6 dup(0)
datos ends

...

mov ax, OFFSET contador
mov bx, SEG contador

call contar4_48

contar4_48 PROC far

    push bx es

    mov es, bx
    mov bx, ax

    add WORD PTR es:[bx], 4
    adc WORD PTR es:[bx+2], 0
    adc WORD PTR es:[bx+4], 0

    pop es bx

    ret

contar4_48 ENDP
```