SISTEMAS BASADOS EN MICROPROCESADORES

Grado en Ingeniería Informática
Escuela Politécnica Superior – UAM

COLECCIÓN DE PROBLEMAS DE LOS TEMAS 1.1 A 2.6

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.

`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=424Eh**, **ES=4240h, SS=424Dh, BX=0, BP=3, DI=3, SP=30** y **AX=1234h**, indicar el **valor hexadecimal de los 16 primeros bytes del segmento DS** una vez ejecutado el siguiente programa.

```
mov 2[BX][DI], AH
mov DS:[BP][DI], AX
mov 22[BP], AX
push AX
```

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

P3. Suponiendo que **CS=2000h**, **DS=424Eh**, **ES=424Dh, SS=424Eh, BX=0004h, BP=0003h** y **DI=0002h**, 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 '?'.**

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

`mov AX, 1[BX]`	AX = **7365h**
`mov AX, [0005h]`	AX = **7365h**
`mov AL, ES: [BP + 16]`	AX = **??61h**
`mov AH, [BP + 3]`	AX = **73??h**
`mov AX, ES: 14[BX][DI]`	AX = **6520h**

P4. Suponiendo que **CS=2000h**, **DS=424Eh**, **ES=424Dh, SS=424Eh, BX=0004h, BP=0003h** y **DI=FFFFh**, 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 '?'.**

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

`mov AX, [BX][DI]`	`AX = 2061h`
`mov AX, [0000h]`	`AX = 424Eh`
`mov AL, DS:[BP]`	`AX = ??61h`
`mov AH, [BP]13`	`AX = ????h`
`mov AX, 2[DI]`	`AX = 7442h`

P5. Suponiendo que **CS=2000h**, **DS=1000h**, **ES=1234h, SS=4321h** y **BX=5432h**, 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.

`mov AH, [BX]`	`@ = 15432h`
`mov AX, 3[BX]`	`@ = 15435h`
`mov AX, ES: [BX + 3]`	`@ = 17775h`
`mov AL, [1000h]`	`@ = 11000h`

P6. Escribir una secuencia de instrucciones de ensamblador para leer sobre el **registro AX** una **palabra de 16 bits** almacenada en la **dirección física E256Ah**.

```
mov ax, 0E256h
mov ds, ax
mov ax, [000Ah]
```

P7. Suponiendo que **CS=2000h**, **DS=193Fh**, **ES=193Eh, SS=2222h** y **BX=0001h**, Indicar el valor del **registro AX** tras ejecutar cada una de las instrucciones siguientes **(independientes entre si)**, dado el volcado de memoria adjunto. Expresar los **dígitos hexadecimales desconocidos de AX con un '?'.**

```
193F:0000 CD 20 FF 9F 00 9A F0 FE
```

`mov AX, [BX]`	`AX = FF20h`
`mov AH, 3[BX]`	`AX = 00??h`
`mov AL, ES: [BX + 20]`	`AX = ??9Ah`
`mov AX, ES: [10h]`	`AX = 20CDh`

P8. Indicar el valor de la constante TMP dado el siguiente fragmento de código *(1 punto):*

```
Valores DW  4, 5*9, 10h+2*34, 23h, 'A'
TMP EQU ($ - Valores)
```

TMP = 10

P9. Suponiendo que **CS=1000h**, **DS=2000h**, **ES=4321h, SS=1111h**, **BX=2222h**, **BP=3333h** 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.

`mov AH, 4[BX][SI]`	@ = 22228h
`mov AH, SS:[BP][SI]`	@ = 14445h
`mov AL, [BP + 4]`	@ = 14447h
`mov AL, CS:[1000h]`	@ = 11000h

P10. 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.

`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

P11. 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 '?'.**

```
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

P12. Suponiendo que **CS=2000h**, **DS=424Eh**, **ES=4240h, SS=424Eh, BX=0, BP=3, DI=3, SP=8** y **AX=1234h**, indicar el **valor hexadecimal de los 16 primeros bytes del segmento DS** una vez ejecutado el siguiente programa.

```
mov SS:[BX][DI], AH
mov DS:[9], AX
mov [BP+11], AX
push ES
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			12h			40h	42h		34h	12h				34h	12h

P13. 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.

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

P14. 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 '?'.**

```
204F:0000 73 65 67 20 00 68 61 6E
```

`mov AX, [SI]`	AX = **2067h**
`mov AH, 3[DI]`	AX = **68??h**
`mov AL, ES:[BX + 5]`	AX = **??68h**
`mov AX, ES:[20h]`	AX = **6573h**
`mov AX, [SI][DI]`	AX = **6800h**

P15. Suponiendo que **CS=1234h**, **DS=2222h**, **ES=F000h, SS=3333h**, **BX=1111h**, **BP=0003h** y **DI=0004h**, 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.

`mov AH, CS:[DI]`	@ = 12344h
`mov AX, 4[DI]`	@ = 22228h
`mov AL, [BX + 8]`	@ = 23339h
`mov AX, DS:[BP][DI]`	@ = 22227h
`mov AL, [BP]`	@ = 33333h

P16. Suponiendo que **CS=2000h**, **DS=204Fh**, **ES=204Fh, SS=2000h, BX=0004h, BP=04F0h**, **SI=000Ah 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 '?'.**

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

`mov AH, ES:[BX][SI]`	AX = 33??h
`mov AL, 3[SI]`	AX = ??00h
`mov AX, [BP + 4]`	AX = 6800h
`mov AL, ES:[BX + 11]`	AX = ??11h
`mov AX, SS:[BP][SI]`	AX = 004Eh

P17. Declarar mediante directivas de ensamblador de 8086 las variables que se describen a continuación. El nombre de la variable se indica ente paréntesis.

```
; (tabla1) Tabla de 12 palabras de 16 bits inicializadas a cero.

tabla1 dw 12 dup (0)
; (contador) Entero de 4 bytes sin inicializar.

contador dd ?


; (tabla2) Tabla de 255 elementos, donde cada elemento es el
           carácter 'A' seguido de un entero de 2 bytes
           inicializado a FFFFh.

tabla2 db 255 dup ( 'A', 0FFh, 0FFh )

; (mensaje) Cadena "Fichero inexistente" seguida de los valores
           10 y 13.

mensaje db "Fichero inexistente", 10, 13
```

```
; (scontador) Entero de 2 bytes inicializado con el segmento de
               la variable "contador".

scontador dw SEG contador
```

P18. Teniendo en cuenta la sección de código de la izquierda, implementar cada una de las operaciones solicitadas en el cuadro de la derecha mediante una **única** instrucción de ensamblador de 8086, suponiendo que las instrucciones se ejecutan en la zona de puntos del procedimiento `inicio`. **Se debe indicar si la instrucción solicitada no es posible.**

```
datos segment
   cadena     db  "Adios",13,10
   longitud   db   $-cadena
datos ends

res segment
   resultado  db  200 dup (?)
   contador   dw  ?
res ends

codigo segment
     assume cs:codigo, ds:datos
     inicio proc far

          mov ax, codigo
          mov ds, ax
          mov ax, datos
          mov es, ax

          ...

          mov ax, 4C00h
          int 21h
     inicio endp
codigo ends
end inicio
```

```
; Leer en AX la variable "longitud".

     mov ax, es: WORD PTR longitud

; Leer en BX la variable "contador".

     No es posible

; Escribir en la tabla "resultado" la cadena
; "Error fatal." en la posición indicada
; por DI.

     No es posible

; Escribir en la tabla "resultado" el valor
; 1024.

     No es posible

; Leer en AX la posición de la tabla "cadena"
; indicada por SI.

     mov ax, es: WORD PTR cadena[ si ]
```

P19. Suponiendo que **CS=1234h**, **DS=1000h**, **ES=F000h, SS=2000h**, **BX=2222h**, **BP=0000h** y **DI=0001h**, 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.

`mov AH, ES:16[DI]`	@ = F0011h
`mov AX, 16[DI]`	@ = 10011h
`mov AL, [BX + 8]`	@ = 1222Ah
`mov AX, CS:[BP][DI]`	@ = 12341h
`mov AL, [BP - 1]`	@ = 2FFFFh

P20. Suponiendo que **CS=2000h**, **DS=204Fh**, **ES=204Fh, SS=2040h, BX=0004h, BP=00F0h**, **DI=000Ah 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 '?'.**

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

`mov AH, ES:[BX][DI]`	`AX = 33??h`
`mov AL, 3[DI]`	`AX = ??00h`
`mov AX, [BP + 7]`	`AX = 126Eh`
`mov AX, ES:[BX + 11]`	`AX = ??11h`
`mov AX, SS:[BP][DI]`	`AX = 004Eh`

P21. Declarar mediante directivas de ensamblador de 8086 las variables que se describen a continuación. El nombre de la variable se indica ente paréntesis.

```
; (tabla1) Tabla de 256 bytes no inicializada

tabla1 db 256 dup (?)
; (contador) Entero de 2 bytes inicializado a 65535.

contador dw 65535

; (tabla2) Tabla de 25 elementos no inicializados, donde cada
           elemento es un entero de 2 bytes seguido de un
           entero de 4 bytes.

tabla2 dw 25 dup ( ?, ?, ? )

; (mensaje) Cadena "Parámetro incorrecto" seguida del valor 0.

mensaje db "Parámetro incorrecto", 0


; (pcontador) Entero de 2 bytes inicializado con la dirección de
              la variable "contador".

pcontador dw contador
```

P22. Teniendo en cuenta la sección de código que se reproduce a la izquierda, escribir las instrucciones de ensamblador de 8086 que se solicitan en el cuadro de la derecha suponiendo que se ejecutan en la zona de puntos suspensivos del procedimiento `inicio`. **Se deberá indicar si la instrucción solicitada no es posible.**

```
datos segment
    cadena     dw  "Hola"
    longitud   db   ?
datos ends

resultados segment
    resultado  db  200 dup (?)
    contador   dw  0
resultados ends

codigo segment
     assume cs:codigo, ds:datos
     inicio proc far

          mov ax, resultados
          mov ds, ax
          mov ax, datos
          mov es, ax

          ■■■

          mov ax, 4C00h
          int 21h
     inicio endp
codigo ends
end inicio
```

```
; Leer en AL la variable “longitud”.

      mov al, es: longitud

; Leer en BX la variable “contador”.

      mov bx, ds: contador

; Escribir en la tabla “resultado” el código
; ASCII de la letra X en la posición indicada
; por DI.

      mov ds: resultado[ di ], ‘X’

; Escribir en la tabla “resultado” el valor
; 65535.

      mov ds: WORD PTR resultado, 65535

; Leer en DX la posición de la tabla “cadena”
; indicada por BX.

      mov dx, es: cadena[ bx ]
```

P23. 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 ‘?’.**

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

`mov AX, [BX]`	`AX = ????h`
`mov AX, DS:[0000h]`	`AX = ????h`
`mov AH, [BP + 3]`	`AX = 61??h`
`mov AL, ES:[BP]15`	`AX = ????h`
`mov AX, 2[DI]`	`AX = 6174h`

P24. 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 ‘?’.**

```
424E:0008 FF A0 23 56 45 3A 30 2E
```

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

P25. Suponiendo que **CS=2222h**, **DS=1234h**, **ES=F000h, SS=3333h**, **BX=1111h**, **BP=0004h** y **SI=0004h**, 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. Responder con "**Incorrecto**" en caso de que el modo de direccionamiento indicado no sea posible.

`mov AH, [SI]`	`@ = 12344h`
`mov AX, CS:[SI]4`	`@ = 22228h`
`mov AL, [BX][BP]`	`@ = Incorrecto`
`mov AX, CS:[SI][BP]`	`@ = 22228h`
`mov AL, ES:[BP]`	`@ = F0004h`

P26. Suponiendo que **CS=4200h**, **DS=424Eh**, **ES=424Dh, SS=424Eh, BX=0, BP=3, DI=3, SI=04ECh** y **AX=1234h**, indicar el **valor hexadecimal de los 16 primeros bytes del segmento DS** una vez ejecutado el siguiente programa.

```
mov CS:04E0h[BX], AX
mov SS:[DI]1, AH
mov DS:[BP][DI]2, AX
mov CS:[SI], AL
mov ES:28[BX][DI], AX
```

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

P27. Declarar mediante directivas de ensamblador de 80x86 las variables que se describen a continuación. El nombre de la variable se indica ente paréntesis.

```
; (tabla1) Tabla de 128 bytes sin inicializar.

tabla1 db 128 dup (?)

; (contador) Entero de 2 bytes inicializado a -1.

contador dw -1
```

```
; (tabla2) Tabla de 100 elementos, donde cada elemento es una
           tabla de 50 palabras de 16 bits inicializadas a 0.

tabla2 dw 100 dup ( 50 dup (0) )

; (mensaje) Cadena "Fichero inexistente" seguida del carácter
            '$'.

mensaje db "Fichero inexistente", '$'

; (pcontador) Dirección larga de la variable "contador".

pcontador dd contador
```

P28. Teniendo en cuenta la sección de código de la izquierda, implementar cada una de las operaciones solicitadas en el cuadro de la derecha mediante una **única** instrucción de ensamblador de 80x86, suponiendo que las instrucciones se ejecutan en la zona de puntos del procedimiento `inicio`. **Se debe indicar si la instrucción solicitada no es posible.**

```
datos segment
   tabla      db  1,2,3,4,5
   v          dw  ?
datos ends

res segment
   resultado db 100
   w         dw ?
res ends

codigo segment
     assume cs:datos, es:res

     inicio proc far

          mov ax, datos
          mov ds, ax
          mov ax, res
          mov es, ax

          ...

          mov ax, 4C00h
          int 21h
     inicio endp
codigo ends
end inicio
```

```
; Leer en AX los dos primeros valores de
; "tabla".

     mov ax, WORD PTR ds:tabla


; Escribir en "v" el entero contenido a partir
; de la posición de memoria con offset FFFEh.

     No es posible

; Escribir en "resultado" el valor almacenado
; en la posición de memoria indicada por SI.

     No es posible

; Escribir en "resultado" el valor almacenado
; en BX.

     mov WORD PTR resultado, BX

; Escribir en "w" el valor almacenado en "v".

     No es posible
```

P29. Suponiendo que **CS=2222h**, **DS=1234h**, **ES=F000h, SS=3333h**, **BX=1111h**, **BP=0006h**, **SI=0004h** y **DI=0003h**, 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. Responder con "**Incorrecto**" en caso de que el modo de direccionamiento indicado no sea posible.

`mov AH, [DI]`	@ = 12343h
`mov AX, CS:7[DI]`	@ = 2222Ah

`mov AL, [SI][DI]`	`@ = Incorrecto`
`mov AX, CS:[BP][DI]`	`@ = 22229h`
`mov AL, DS:[BP]`	`@ = 12346h`

P30. Declarar mediante directivas de ensamblador de 80x86 las variables que se describen a continuación. El nombre de la variable se indica ente paréntesis.

```
; (tabla1) Tabla de 256 enteros de 16 bits sin inicializar.

tabla1 dw 256 dup (?)

; (contador) Entero de 4 bytes inicializado a -1.

contador dd -1

; (tabla2) Tabla de 50 elementos, donde cada elemento es una
           tabla de 50 bytes inicializados a 0.

tabla2 db 50 dup (50 dup (0))

; (mensaje) Cadena “Fichero inexistente” seguida del carácter
           ‘$’.

mensaje db “Fichero inexistente”,’$’

; (pcontador) Dirección larga de la variable “contador”.

pcontador dd contador
```