SISTEMAS BASADOS EN MICROPROCESADORES

Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y Matemáticas
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 + 2]`	`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
```

P31. Suponiendo que **CS=2000h**, **DS=424Ch**, **ES=424Eh, SS=424Eh, BX=0002h, BP=0003h** y **SI=0003h**, 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 AL, 35[BX]                  AX = ??65h

mov AX, SS: [0007h]             AX = ??20h

mov AH, [BP + 35]               AX = ????h

mov AX, DS: [BP + 35]           AX = 2073h

mov AX, ES: [BX][SI]            AX = 7365h
```

P32. Suponiendo que **CS=2000h**, **DS=424Fh**, **ES=424Fh, SS=424Fh, BX=0007h, BP=FFFFh** y **DI=0005h**, 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 '?'.**

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

`mov AX, ES:[BX]`	`AX = ??20h`
`mov AX, [BP]`	`AX = ????h`
`mov AH, SS:[BP + 2]`	`AX = 42??h`
`mov AL, [DI]`	`AX = ??65h`
`mov AX, CS:[BP][DI]`	`AX = ????h`

P33. Suponiendo que **CS=1234h**, **DS=1234h**, **ES=FFF0h, SS=1111h**, **BX=1111h**, **BP=2222h, SI=0002h** 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. Responder con "**Incorrecto**" en caso de que el modo de direccionamiento indicado no sea posible.

`mov AH, [SI+16]`	`@ = 12352h`
`mov AX, ES:[SI+16]`	`@ = FFF12h`
`mov AL, [SI][DI]`	`@ = Incorrecto`
`mov AX, CS:[SI][BP]`	`@ = 14564h`
`mov AL, ES:[DI]`	`@ = FFF04h`

P34. Suponiendo que **CS=3000h**, **DS=324Ah**, **ES=324Bh, SS=324Ah, BP=0006h, SI=0003h** y **DI=024Ah**, 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 '?'**.

```
324A:0000 73 65 67 20 00 68 61 6E 12 BB A0 22 FF 00 98 7E
324A:0010 56 3D BB 35 DE C5 4F 24 02 FF 4D E5 11 AA 23 00
```

`mov AX, ES: [SI]`	`AX = DE35h`
`mov AH, CS: 3[DI]`	`AX = ????`
`mov AL, SS: [BP][SI]`	`AX = ??BBh`
`mov AX, ES: [000Ah]`	`AX = E54Dh`
`mov AH, CS: [024Bh]`	`AX = ????`

P35. 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 inicializados a -1.
tabla1 dw 256 dup (-1)
; (contador) Entero de 4 bytes sin inicializar.
```

```
contador dd ?

; (tabla2) Tabla de 100 elementos, donde cada elemento es una
           cadena de 200 caracteres '=' y dos enteros sin
           inicializar.

tabla2 db 100 dup (200 dup ('='), ?, ?, ?, ? )

; (mensaje) Cadena "Error en apertura del fichero".

mensaje db "Error en apertura del fichero"

; (pmensaje) Dirección corta de la variable "mensaje".

pmensaje dw mensaje
```

P36. 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  "HELLO"
    v          dw  ?
datos ends

res segment
    w          dw ?
res ends

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

           resultado dw 100

     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 BP los dos primeros caracteres de
; "tabla".

    mov bp, WORD PTR ds:tabla

; Escribir en "v" los dos primeros caracteres
; de "tabla".

    No es posible

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

    No es posible

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

    mov es:w, ax

; Escribir en el quinto byte de tabla el
; valor 0.

    mov ds:tabla[4], 0
```

P37. Suponiendo que **CS=0000h**, **DS=324Ah**, **ES=324Bh, SS=324Ah, BP=0006h, SI=0003h** y **DI=24A0h**, 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 '?'**.

```
324A:0000 73 65 67 20 00 68 61 6E 12 BB A0 22 FF 00 98 7E
324A:0010 56 3D BB 35 DE C5 4F 24 02 FF 4D E5 11 AA 23 00
```

`mov AX, ES: [SI]`	**AX = DE35h**
`mov AH, CS: 3[DI]`	**AX = 20??h**
`mov AL, SS: [BP][SI]`	**AX = ??BBh**
`mov AH, CS: [24B0h]`	**AX = 56??h**

P38. Suponiendo que **CS=3000h**, **DS=324Ah**, **ES=324Bh, SS=324Ah, BP=0003h, SI=0007h** y **DI=24A0h**, 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 '?'**.

```
324A:0000 56 3D BB 35 DE C5 4F 24 02 FF 4D E5 11 AA 23 00
324A:0010 73 65 67 20 00 68 61 6E 12 BB A0 22 FF 00 98 7E
```

`mov AX, SS: [SI][BP]`	**AX = E54Dh**
`mov AX, CS: [24AFh]`	**AX = 7300h**
`mov AX, ES: [SI]`	**AX = 126Eh**
`mov AL, CS: 15[DI]`	**AX = ??00h**

P39. Suponiendo que **CS=1234h**, **DS=1234h**, **ES=FFFFh, SS=2222h**, **BX=0010h**, **BP=0011h, SI=0002h** 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. Responder con "**Incorrecto**" en caso de que el modo de direccionamiento indicado no sea posible.

`mov AX, [BX]`	**@ = 12350h**
`mov AH, ES:[DI+10]`	**@ = FFFFEh**
`mov AX, [BP][DI]`	**@ = 22235h**
`mov AL, DS:[SI][BP]`	**@ = 12353h**
`mov AX, ES:[DI]`	**@ = FFFF4h**

P40. 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 1000 enteros de 32 bits inicializados a -1.
tabla1 dd 1000 dup (-1)

; (contador) Entero de 16 bits inicializado a 0.
contador dw 0
```

```
; (tabla2) Tabla de 200 elementos, donde cada elemento es un
           byte inicializado a 0 y una tabla de 20 enteros de
           16 bits inicializados a 00FFh.

tabla2 db 200 dup (0, 20 dup (0FFh, 0))

; (mensaje) Cadena "Error de comunicación" seguida del carácter
            '$'.

mensaje db "Error de comunicación$"

; (pmensaje) Dirección larga de la variable "mensaje".

pmensaje dd mensaje
```

P41. 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
   w          dw ?
datos ends

res segment
   resultado db 100 dup (?)
res ends

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

      tabla   db  "HELLO"
      v       dw  $-tabla

     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
```

```
; Escribir en AX el tamaño de "tabla"
; sin usar modo de direccionamiento inmediato.

          mov ax, ds:v

; Escribir en "w" el tamaño de "tabla"
; sin usar modo de direccionamiento inmediato.

          No es posible

; Inicializar "w" a 1234h.

          mov es:w, 1234h

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

          mov bp, es:w

; Escribir en el último byte de "resultado" el
; valor 0.

          No es posible
```

P42. Suponiendo que **CS=4000h**, **DS=424Ah**, **ES=424Bh, SS=424Ah, BP=0010h, SI=0006h** y **DI=24B0h**, 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 '?'**.

```
424A:0000 34 55 A2 23 81 99 AB 1F 12 AA 0A 22 FF FF 98 7E
```

```
424A:0010 65 D3 CC 44 DE C5 F4 42 37 FF D4 5E 00 0A 23 00
```

`mov AH, [SI]`	`AX = AB??h`
`mov AX, CS: 6[DI]`	`AX = 42F4h`
`mov AL, SS: [BP][SI]`	`AX = ??F4h`
`mov AX, CS: [24AFh]`	`AX = 657Eh`
`mov AX, ES: [SI]`	`AX = 42F4h`

P43. Suponiendo que **CS=4000h**, **DS=424Ah**, **ES=424Bh, SS=424Ah, BP=0010h, SI=0006h** y **DI=24B0h**, 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 '?'**.

```
424A:0000 65 D3 CC 44 DE C5 F4 42 37 FF D4 5E 00 0A 23 00
424A:0010 34 55 A2 23 81 99 AB 1F 12 AA 0A 22 FF FF 98 7E
```

`mov AX, [SI]`	`AX = 42F4h`
`mov AH, CS: 9[DI]`	`AX = AA??h`
`mov AL, SS: 4[BP][SI]`	`AX = ??0Ah`
`mov AX, CS: [24AEh]`	`AX = 0023h`
`mov AX, ES: [BP]`	`AX = ????h`

P44. Suponiendo que **CS=4321h**, **DS=4321h**, **ES=0000h, SS=2222h**, **BX=1000h**, **BP=0010h, SI=0004h** 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. Responder con "**Incorrecto**" en caso de que el modo de direccionamiento indicado no sea posible.

`mov AX, [SI][DI]`	`@ = Incorrecto`
`mov AH, CS:[BX][DI]`	`@ = 44212h`
`mov AX, DS:[BP][SI]`	`@ = 43224h`
`mov AL, ES:[SI+0FFFFh]`	`@ = 00003h`
`mov AX, SS:[DI]`	`@ = 22222h`

P45. Suponiendo que **CS=3000h**, **DS=324Ah**, **ES=324Eh, SS=324Ah, BP=0030h, SI=FFFEh** y **DI=24A0h**, 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 '?'**.

```
324A:0030 1B 22 01 00 1F C5 4F 24 02 FF 4D E5 11 AA 23 00
324A:0040 23 4E 21 AA FF DD 1A 6E 21 A0 01 33 12 00 98 7E
```

`mov AL, DS: 4[BP][SI]`	`AX = ??01h`
`mov AX, [BP]`	`AX = 221Bh`
`mov AH, CS: [BP][DI]`	`AX = 1B??h`
`mov AX, ES: [0000h]`	`AX = 4E23h`
`mov AL, DS: [SI+40h]`	`AX = ??23h`

P46. 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 1000 enteros de 8 bits inicializados a 100.

tabla1 db 1000 dup (100)


; (contador) Entero de 32 bits sin inicializar.

contador dd ?


; (tabla2) Tabla de 100 elementos, donde cada elemento es un
           entero de 32 bits inicializado a -1, la cadena de
           caracteres "12345" y un entero de 16 bits sin
           inicializar.

tabla2 db 100 dup (4 dup (-1), "12345", ?, ?)

; (mensaje) Cadena "ERROR" seguida de un byte inicializado a 0.

mensaje db "ERROR",0

; (ptabla2) Desplazamiento (offset) de la variable "tabla2".

pmensaje dw tabla2
```

P47. 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  "HELLO"
    x         db  $-tabla
    y         dw  255

datos ends

res segment
    resultado db 100 dup (0)
res ends

codigo segment
    assume cs:codigo

          w  dw ?

     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
```

```
;  Escribir  en  AX  el  tamaño  de  "tabla"
; sin usar modo de direccionamiento inmediato.

          No es posible

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

          No es posible

; Escribir en "y" el valor del registro AH.

          mov ds: BYTE PTR y, ah

; Escribir en BP los dos últimos bytes
; de "resultado".

          mov bp, es: WORD PTR resultado[98]

; Escribir en AX el valor de "w".

          mov ax, w
```

P48. Suponiendo que **CS=4200h**, **DS=424Ah**, **ES=424Bh, SS=424Ah, BP=FFFEh, SI=0006h** y **DI=04B0h**, 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 '?'**.

```
424A:0000 11 23 A2 3F 88 AA C0 F1 21 BA 0F 14 11 00 12 6D
424A:0010 34 3D A1 2E FF F5 F4 45 AA A9 01 00 A0 23 1F 1E
```

`mov AH, CS: [DI+4]`	AX = FF??h
`mov AX, 6[DI]`	AX = ????h
`mov AL, SS: 2[BP][SI]`	AX = ??C0h
`mov AX, CS: [04B2h]`	AX = 2EA1h
`mov AL, DS: [SI]`	AX = ??C0h

P49. Suponiendo que **CS=4200h**, **DS=424Bh**, **ES=424Ch, SS=424Bh, BP=FFFFh, DI=0000h** y **SI=04C0h**, 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 '?'**.

```
424A:0010 34 3D A1 2E FF F5 F4 45 AA A9 01 00 A0 23 1F 1E
424A:0020 11 23 A2 3F 88 AA C0 F1 21 BA 0F 14 11 00 12 6D
```

`mov AX, CS: [SI+14]`	`AX = 6D12h`
`mov AL, 6[DI]`	`AX = ??F4h`
`mov AX, SS: 2[BP][DI]`	`AX = A13Dh`
`mov AH, CS: [04B4h]`	`AX = FF??h`
`mov AX, DS: [SI]`	`AX = ????h`

P50. Suponiendo que **CS=AAAAh**, **DS=BBBBh**, **ES=CCCCh, SS=DDDDh**, **BX=1111h**, **BP=FFFFh**, **SI=**1111h 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. Responder con "**Incorrecto**" en caso de que el modo de direccionamiento indicado no sea posible.

`mov AH, [SI][DI]`	`@ = Incorrecto`
`mov AX, CS:[SI+10]`	`@ = ABBBBh`
`mov AL, DS:[BX]`	`@ = BCCC1h`
`mov AX, CS:[DI][BP]`	`@ = AAAA3h`
`mov AL, ES:[BP]`	`@ = DCCBFh`

P51. Suponiendo que **CS=4000h**, **DS=424Eh**, **ES=4200h, SS=424Eh, BX=24E2h, BP=5, DI=FFFEh, 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:[BX], AL
mov CS:[BX][DI], AX
mov ES:[SI+3], AH
mov SS:[BP][DI], AX
mov DS:[BP][DI]2, AL
```

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

P52. 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 100 enteros de 4 bytes sin inicializar
           seguida de tabla de 255 bytes inicializados a -1.

tabla1 dd 100 dup (?)
       db 255 dup (-1)

; (contador) Entero de 8 bytes inicializado a FFFFh.
```

```
contador dq 0FFFFh

; (tabla2) Tabla de 1000 elementos, donde cada elemento es un
           entero de 4 bytes inicializado a 0 y una tabla de 20
           enteros de 8 bytes inicializados a -2.

tabla2 dd 1000 dup ( 0, 20 dup (-2,-1) )

; (mensaje) Cadena "Undefined symbol" seguida de entero de 2
           bytes inicializado a 0A0Dh.

mensaje db "Undefined symbol", 0Dh, 0Ah

; (pmensaje) Dirección corta de la variable "mensaje".

pmensaje dw mensaje
```

P53. 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      dw  1,2,3,4,5
    v          dw  ?
datos ends

res segment
    resultado dw ?, ?
    w         dw ?
res ends

codigo segment
     assume ds:datos, es:res

     inicio proc far

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

          •••

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

```
; Leer en SI el tercer entero de "tabla".

     mov si, es:tabla[4]

; Escribir en "v" el tercer entero de "tabla".

     No es posible

; Escribir en "resultado" el entero de 4 bytes
  1234h.

     No es posible

; Leer en AL el byte alto de "w".

     mov al, BYTE PTR ds:w[1]

; Escribir en "w" la dirección corta de "v".

     mov ds:w, OFFSET v
```

P54. Suponiendo que **CS=1267h**, **DS=2235h**, **ES=1A00h, SS=F000h**, **BX=0001h**, **BP=0010h**, **SI=2222h** y **DI=3333h**, 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 BL, DS:[DI][BP]2`	`@ = 25695h`
`mov AX, CS:[SI-2]`	`@ = 14890h`
`mov AL, ES:[BX]`	`@ = 1A001h`
`mov AX, SS:[DI][BP]`	`@ = F3343h`
`mov AL, ES:[BP+0F00h]`	`@ = 1AF10h`

P55. Suponiendo que **CS=3000h**, **DS=3200h**, **ES=3201h, SS=3200h, BP=0006h, SI=000Fh** y **DI=2000h**, 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 '?'**.

```
3200:0000 73 65 67 20 00 68 61 6E 12 BB A0 22 FF 00 98 7E
3200:0010 56 3D BB 35 DE C5 4F 24 02 FF 4D E5 11 AA 23 00
```

`mov AL, SS: [BP][SI]`	`AX = ??C5h`
`mov AX, CS: [2010h]`	`AX = 3D56h`
`mov AX, ES: [SI-1]`	`AX = 0023h`
`mov AL, 3[DI]`	`AX = ????h`
`mov AH, ES: [SI]`	`AX = 00??h`

P56. 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 25 enteros de 2 bytes inicializados a 255


tabla1 dw 25 dup (255)

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

contador dd 1

; (tabla2) Tabla de 150 elementos, donde cada elemento es la
           cadena de caracteres "ROW" seguida de una tabla de
           10 enteros de 4 bytes inicializados a 12345678h.

tabla2 db 150 dup ( "ROW", 10 dup (78h, 56h, 34h, 12h) )

; (mensaje) Cadena "Error de protocolo" seguida del carácter
            '$'.

mensaje db "Error de protocolo$"

; (ptabla2) Dirección larga de la variable "tabla2".
```

```
ptabla2 dd tabla2
```

P57. 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 dw 1, 2, 3, 4, 5
    v     db ?
datos ends

res segment
    resultado dd 6 dup (?)
    w         dw ?
res ends

codigo segment
     assume ds:res, es:datos

     signature dw 0CAFEh

     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 AL el primer byte de "signature".

    mov al, BYTE PTR cs:signature

;   Escribir en "w" el offset del
  procedimiento "inicio".

    mov es:w, OFFSET inicio ; == 2

; Leer en AL el primer byte del procedimiento
  "inicio"

    mov al, cs:[2]

; Escribir en "resultado" el contenido de SI.

    mov WORD PTR es:resultado, si

; Leer en BP la quinta palabra de "tabla".

    mov bp, ds:tabla[8]
```

P58. Suponiendo que **CS=AAAAh**, **DS=BBBBh**, **ES=CCCCh**, **SS=DDDDh**, **BX=1000h**, **BP=0100h**, **SI=1111h** y **DI=FFFFh**, 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 CX, DS:[BX][CS]`	`@ = Incorrecto`
`mov AX, CS:[DI+1]`	`@ = AAAA0h`
`mov CH, [BX]`	`@ = BCBB0h`
`mov DL, SS:[DI]`	`@ = EDDCFh`
`mov DX, ES:[SI-1112h]`	`@ = DCCBFh`

P59. Suponiendo que **CS=1000h**, **DS=11FFh**, **ES=11F0h, SS=11FFh, BX=1, BP=3, DI=F0h, SI=1FF0h** y **AX=ABCDh**, indicar el **valor hexadecimal de los 16 primeros bytes del segmento DS** una vez ejecutado el siguiente programa.

```
mov CS:[BX][SI], AX
mov SS:[BP]5, AH
mov ES:[BP][DI], AX
mov [DI+0FF1Fh], AL
mov ES:251[BX], AX
```

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

P60. Declarar mediante directivas de ensamblador de 80x86 las variables que se describen a continuación. El nombre de la variable se indica ente paréntesis.

```
; (registro1) Cadena "*****" seguida de tabla de 1000 enteros de
;              4 bytes inicializados a FFFFFFFFh

 registro1  db "*****"
            dd 1000 dup (0FFFFFFFFh)


; (contador) Entero de un byte inicializado a -1

contador db -1

; (tabla2)  Tabla de 200 elementos, donde cada elemento es un
            entero de 2 bytes sin inicializar y un entero de 4
            bytes inicializado a ABCDEFh.

tabla2 dw 200 dup ( ?, 0CDEFh, 00ABh )

; (mensaje) Cadena "ERROR FATAL" seguida del byte 0.

mensaje db "ERROR FATAL", 0

; (pcontador) Offset de la variable "contador".

pcontador dw contador
```

P61. 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 dw 100 dup (0)
   w         dd ?
res ends

codigo segment
     assume ds:datos

     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 AH el primer byte de "v".

     mov ah, BYTE PTR v

; Escribir en "w" el offset de "resultado".

     mov WORD PTR es:w, OFFSET resultado

; Leer en AX el último byte de "tabla".

     No es posible

; Cambiar los dos primeros bytes de "tabla"
; por los valores 4 y 5 respectivamente.

     mov WORD PTR tabla, 0504h

; Leer en BP la última palabra de "resultado".

     mov bp, es:resultado[99*2]
```

P62. Suponiendo que **CS=A000h**, **DS=A201h**, **ES=A1FFh, SS=A200h, BX=0006h, SI=FFFEh** y **DI=201Ch**, 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 '?'**.

```
A200:0000 A6 23 1A 5F E5 21 9A 7A B3 D1 0A AA 1E 44 73 64
A200:0010 38 29 A2 CC 9B CA CA 61 00 BD 15 1A B8 E0 00 3D
```

`mov AX, [BX][SI]`	`AX = CA9Bh`
`mov AX, SS: [BX][SI]`	`AX = 21E5h`
`mov AH, [SI-4]`	`AX = ????h`
`mov AX, CS: 2[DI]`	`AX = 3D00h`

P63. Suponiendo que **CS=1111h**, **DS=2222h**, **ES=3333h, SS=4444h**, **BX=1000h**, **DX=1000h**, **BP=0100h**, **SI=0A0Ah** y **DI=FFFFh**, 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 AL, [BP-5]`	`@ = 4453Bh`
`mov CX, ES:[DX-1]`	`@ = Incorrecto`
`mov BL, CS:[BX][SI]`	`@ = 12B1Ah`

`mov CX, SS:[DI]`	`@ = 5443Fh`
`mov DH, ES:[1112h+DI]`	`@ = 34441h`

P64. Suponiendo que **CS=5300h**, **DS=5348h**, **ES=5347h, SS=5347h, BP=0010h**, **BX=FFFFh**, **SI=0006h** y **DI=047Fh**, 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 '?'**.

```
5347:0000 23 AF 21 AA FF 00 1A 6E 21 03 A1 4B CC 00 98 7E
5347:0010 1B 22 00 00 1F C5 4F 24 02 FF 4D E5 11 A8 23 00
```

`mov AH, [BX][SI]`	`AX = C5??h`
`mov AX, CS: [DI]`	`AX = 1B7Eh`
`mov AL, CS: [DI][BX]`	`AX = ??98h`
`mov AX, SS: [0005h]`	`AX = 1A00h`
`mov AX, ES: [BP+10]`	`AX = E54Dh`

P65. Declarar mediante directivas de ensamblador de 80x86 las variables que se describen a continuación. El nombre de la variable se indica ente paréntesis.

```
; (v) Entero de 4 bytes inicializado a ABCDEFh

v dd 0ABCDEFh
; (registro)  Tabla de 10 enteros de 8 bytes sin inicializar,
              seguida de tabla de 1000 enteros de un byte
              inicializados a FFh

 registro   dq 10 dup (?)
            db 1000 dup (0FFh)


; (tabla) Tabla de 100 elementos, donde cada elemento es: un
          entero de 4 bytes inicializado a 255, una tabla de
          100 enteros de 2 bytes sin inicializar, y un entero
          de 4 bytes inicializado a la dirección larga de "v".

tabla dw 100 dup (255, 0, 100 dup (?), OFFSET v, SEG v)

; (mensaje) Cadena "UNIVERSIDAD AUTÓNOMA" seguida del entero de
            2 bytes CDBAh.

mensaje db "UNIVERSIDAD AUTÓNOMA", BAh, CDh

; (ptabla) Segmento de la variable "tabla".

ptabla dw SEG tabla
```

P66. 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 10 dup (255)
    pres  dd 0
datos ends

datos2 segment
    res dw 100 dup (?)
    w   db ?, ?
datos2 ends

codigo segment
     assume ds:datos

     tabla2 db "HOLAS", ?

     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
```

```
; Escribir en la segunda palabra de "pres"
; el segmento de "res".

    mov WORD PTR pres[2], SEG res

; Escribir en el último byte de tabla2 el
; carácter '$'.

    mov cs:tabla2[5], '$'

; Escribir en "pres" el valor 123456h.

    No es posible

; Escribir en "w" el valor de BX.

    No es posible

; Cargar en AX la décima palabra de "res".

    No es posible
```

P67. Suponiendo que **CS=B000h**, **DS=BB01h**, **ES=BAFFh, SS=BB00h, BX=0006h, SI=FFFFh** y **DI=B01Fh**, 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 '?'**.

```
BB00:0000 38 29 A2 CC 9B CA CA 61 00 BD 15 1A B8 E0 00 3D
BB00:0010 A6 23 1A 5F E5 21 9A 7A B3 D1 0A AA 1E 44 73 64
```

`mov AX, ES:[BX][SI+16]`	`AX = CACAh`
`mov AH, [SI][BX]`	`AX = 21??h`
`mov AL, SS:[001Fh]`	`AX = ??64h`
`mov AX, CS:[DI]`	`AX = ??64h`

P68. Suponiendo que **CS=3500h**, **DS=3548h**, **ES=3547h, SS=3547h, BP=0010h**, **BX=FFFFh**, **SI=0006h** y **DI=047Fh**, 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 '?'**.

```
3547:0000 23 AF 21 AA FF 1A 00 6E 21 03 A1 4B CC 00 BF 12
3547:0010 AA 22 00 00 1F AA 4F 24 02 FF E5 4D 11 A8 23 00
```

`mov AL, [BX][SI]`	`AX = ??AAh`
`mov AH, CS: [DI][BX]`	`AX = BF??h`
`mov AX, ES: [BP+10]`	`AX = 4DE5h`
`mov AX, CS: [DI]`	`AX = AA12h`