



Apellidos, Nombre: \_\_\_\_\_

El examen constará de dos partes:

- 1ª parte: Test, con una puntuación de 4 puntos y 30 minutos de tiempo.
- 2ª parte: Ejercicios prácticos, con una puntuación de 6 puntos y 2 horas de tiempo.

**No se considerarán como válidas las respuestas en las que no se justifiquen los cálculos realizados**

**No se permite el uso de ningún tipo de documentación, ni de calculadora**

**Sólo existe una única opción correcta por pregunta de test**

**Puntuación:** Respuesta correcta  $\Rightarrow$  **0,4** Respuesta incorrecta  $\Rightarrow$  **-0,15** Pregunta no contestada  $\Rightarrow$  **0**

En la arquitectura de Von Neumann:

- a) Se interconectan varias unidades funcionales a través de los buses de datos, direcciones y control.
- b) La memoria principal del computador almacena un programa escrito en lenguaje máquina.
- c) Tanto la memoria principal, como el banco de registros del computador, pueden almacenar los datos de un programa representados en binario.
- d) **Todas las anteriores.**

Cuando se está ejecutando un programa en un computador con arquitectura Von Neumann, éste debe estar almacenado en:

- a) El disco duro.
- b) **Memoria principal.**
- c) En el CD-ROM.
- d) Todas las anteriores.

Señala cuál de las siguientes afirmaciones es correcta:

- a) En el lenguaje máquina cada instrucción está representada con cadenas de ceros y unos.
- b) El lenguaje ensamblador es diferente para cada computador, salvo que exista compatibilidad en una familia de computadores.
- c) Una instrucción del lenguaje máquina posee información sobre la operación que va a realizarse y la ubicación de los operandos.
- d) **Todas las anteriores.**

Indicar los valores que toma el número 1000 0011 b, en cada uno de los siguientes sistemas de representación:

(NOTA: En ningún caso esta pregunta tendrá una puntuación negativa)

Signo-magnitud: \_\_\_\_\_-3\_\_\_\_\_.

Binario puro: \_\_\_\_\_131\_\_\_\_\_.

Exceso  $2^{n-1}$ : \_\_\_\_\_3\_\_\_\_\_.

Complemento a 2: \_\_\_\_\_-125\_\_\_\_\_.

Complemento a 1: \_\_\_\_\_-124\_\_\_\_\_.

**S-M: signo (-)  $\Rightarrow$  -3**

**Binario Puro: 128+2+1=131**

**Exceso: 131-128 = 3**

**C2: Negativo (0111 1101) = -125**

**C1: Negativo (0111 1100) = -124**

La resta de A: 0111 1111 b menos B: 0101 1101 b, ambos representados en C1, es:

- a) **22 h.**
- b) DC h.
- c) -7 d.
- d) Otro valor: \_\_\_\_\_.

**Realizamos operación A + (-B):**

```

0111 1111
1010 0010 +
-----
10010 0001
          1 + Se recircula el acarreo
-----
0010 0010

```



Apellidos, Nombre: \_\_\_\_\_

Normaliza el siguiente número fraccionario:

- Exponente: 0 000 0010, en signo-magnitud.
  - Mantisa: 0 001 1110, en signo-magnitud, con coma a la izquierda del MSB y sin bit implícito.
- a) Exponente: 0 000 0001  
Mantisa: 0 011 1100.
- b) Exponente: 0 000 0010  
Mantisa: 0 111 1000.
- c) Exponente: 0 000 0000  
Mantisa: 0 111 1000.**
- d) Exponente: \_\_\_\_\_  
Mantisa: \_\_\_\_\_.

La instrucción SHR BX, CL, desplaza lógicamente a la derecha el registro BX tantas veces como indique el registro CL. Si inicialmente BX es 8000 h y CL es 3. ¿Cuál es el contenido de BX tras ejecutar la instrucción?

- a) 0008 h.
- b) 0000 h.
- c) 1000 h.**
- d) Otro valor: \_\_\_\_\_.

Desplaza tres veces introduciendo 0 el valor  
1000 0000 0000 0000  
Luego:  
0001 0000 0000 0000

¿Qué modo de direccionamiento emplea el operando **destino** en cada una de las siguientes instrucciones?

MOV AH, 02h	ADD Numero, CL	SUB SI, DL
<input type="checkbox"/> a) Inmediato.	<input type="checkbox"/> a) A registro.	<input type="checkbox"/> a) Inmediato.
<input type="checkbox"/> b) Implícito.	<input type="checkbox"/> b) Relativo a índice.	<input type="checkbox"/> b) Relativo a base.
<input type="checkbox"/> <b>c) A registro.</b>	<input type="checkbox"/> c) Inmediato.	<input type="checkbox"/> <b>c) A registro.</b>
<input type="checkbox"/> d) Directo a memoria.	<input type="checkbox"/> <b>d) Directo a memoria.</b>	<input type="checkbox"/> d) Relativo a índice.

Un computador dispone de una memoria de 4 Kbytes. Suponiendo que en cada posición de memoria se almacena una palabra de 32 bits. ¿Cuántas líneas de dirección serían necesarias para referenciar cada palabra?

- a) 12 líneas.
- b) 10 líneas.**
- c) 7 líneas.
- d) Otro valor: \_\_\_\_\_.

4 Kbytes =  $4 \cdot 2^{10} \cdot 2^3 = 2^{15}$  bits  
 32 bits de palabra =  $2^5$  bits de palabra  
 $2^{15} / 2^5 = 2^{10}$

Indicar la opción en la que los dispositivos de memoria están ordenados de menor a mayor tiempo de acceso:

- a) Registros del procesador, memoria principal, disco duro, memoria caché, disquetes y cinta.
- b) Memoria caché, registros del procesador, memoria principal, disco duro, disquetes y cinta.
- c) Registros del procesador, memoria caché, memoria principal, disco duro, disquetes y cinta.**
- d) Otra opción: \_\_\_\_\_



La resolución de cada uno de los ejercicios debe realizarse en hojas separadas

No se considerarán como válidas las respuestas en las que no se justifiquen los cálculos realizados

No se permite el uso de ningún tipo de documentación, ni de calculadora

**EJERCICIO 1:****(2 puntos)**

Se desea diseñar un computador que permita representar números en coma flotante, contenidos en el rango siguiente:

$$\text{Rango: } [-2^{-1} \cdot 2^{127}; -(2^{-2}+2^{-25}) \cdot 2^{-128}; 2^{-2} \cdot 2^{-128}, (2^{-1}-2^{-25}) \cdot 2^{127}]$$

Se sabe que el exponente está representado en exceso  $2^{8-1}$  y que la mantisa está representada en complemento a 2, es fraccionaria y emplea la técnica del bit implícito (la coma se sitúa a la izquierda del bit implícito).

- a. Calcular el número de bits necesarios para el exponente y para la mantisa de tal forma que se obtenga el rango anterior. **(0,5 puntos)**

$$\text{Rango del exponente: } [2^{q-1} - 1; -2^{q-1}] = [127; -128]$$

**R. representación con bit implícito:**

$$[(2^{-1}-2^{-(p+1)}) \cdot 2^{127}; 2^{-2} \cdot 2^{-128}; -(2^{-2}+2^{-(p+1)}) \cdot 2^{-128}; -2^{-1} \cdot 2^{127}]$$

Por tanto  $n^\circ$  de bits para el exponente = **8**

$n^\circ$  de bits para la mantisa  $\Rightarrow 25 = P+1 \Rightarrow P = \mathbf{24}$

- b. Representar el número  $-14,125$  en dicho formato. **(0,5 puntos)**

$$14,125_{10} = 1110,001_b \Rightarrow \text{Normalizando: } 0,01110001 \cdot 2^5; \text{ Exponente} = 5$$

Como es un  $n^\circ$  negativo realizamos el complemento a 2 de la mantisa (01110001) = 10001111

La mantisa con bit implícito: 0001 111

**Sol: 1000 0101 0001 1110 0000 0000 0000 0000**

Exponente                      Mantisa (24 bits)

Se desea aumentar la precisión de la representación, añadiendo un byte al formato anteriormente calculado.

- c. ¿Dónde se deberá añadir dicho byte: al exponente o a la mantisa? Justificar la respuesta. **(0,5 puntos)**

Para mayor precisión se añaden **a la mantisa**. (ver en el rango los terminos  $2^{-(p+1)}$ )

Si se deseara ampliar el rango sería mejor añadirlo al exponente.

Calcular con el nuevo formato de representación el valor del número:

Exponente: 10 ... 01, Mantisa: 001 ... 000

**(0,5 puntos)**

Exponente = 1

Mantisa = 1001 0....0 Es un  $n^\circ$  negativo luego descomplementamos: 0111 0....0

Por tanto:  $-(0,0111 0....0) \cdot 2^1 = -(0,111)$

**Sol: - 0,875**

**EJERCICIO 2:****(2 puntos)**

A continuación se muestra un fragmento de un programa escrito en lenguaje ensamblador del i80x86.

La rutina **Teclear\_notas** (que no aparece reflejada en el anterior fragmento de programa), realiza la lectura de un conjunto de notas de alumnos desde el teclado, hasta que el usuario introduce el carácter de **Fin\_lista**.

...

**.DATA**

```
Aprobado      EQU 5h
Fin_lista     EQU 7Fh
Fin_programa  EQU 4Ch
Nro_alumnos   DB 0
Nro_aprobados DB 0
Nota_maxima   DB 0
Lista_notas   DB 100 DUP (0)      ; Reserva de 100 posiciones de memoria inicializadas a 0
```

**.CODE**

```
P_principal:
    mov AX, @DATA
    mov DS, AX
    call Teclear_notas
```

**Recuento:**

```
    push SI                ; Salva el valor de los registros
    push BX
    xor SI, SI
```

**Inicio\_bucle:**

```
    cmp Lista_notas[SI], Fin_lista ; ¿Se ha llegado al final de la lista?
    jz Fin_recuento                ; Sí, finalizar procedimiento
    inc Nro_alumnos                 ; No, incrementar el contador de alumnos

    cmp Lista_notas[SI], Aprobado   ; Nota ≤ 5?
    jle Comprobar_nota_maxima       ; Sí, comprobar si es la mayor de la lista
    inc Nro_aprobados               ; No, incrementar el nro. de aprobados
```

**Comprobar\_nota\_maxima:**

```
    mov BL, Lista_notas[SI]
    cmp BL, Nota_maxima             ; ¿Nota leída ≤ Nota_maxima?
    jle Siguiente_nota              ; Sí, leer la siguiente nota de la lista
    mov Nota_maxima, BL             ; No, es la nota máxima de la lista
```

**Siguiente\_nota:**

```
    inc SI
    jmp Inicio_bucle
```

**Fin\_recuento:**

```
    pop BX                       ; Restaura el valor de los registros
    pop SI
```

**Fin P\_principal:**

```
    mov AH, Fin_programa
    int 21h
```



- a. Explicar brevemente la funcionalidad del anterior fragmento de programa. **(0,2 puntos)**

De Lista\_notas va cogiendo una a una las notas y las compara con 5, si son > de 5 incrementa Nro\_aprobados. Además calcula el N° de alumnos incrementando Nro\_alumnos por cada nota y calcula la nota máxima de todas. Termina al llegar a la última nota de lista\_notas “7Fh”

- b. ¿A partir de qué nota se considera que un alumno está aprobado? ¿Habría que realizar alguna modificación en el programa para que el aprobado se contabilizara a partir de 5 puntos? Justificar ambas respuestas. **(0,2 puntos)**

Estará aprobado si su nota es mayor de 5, no si es igual a 5.

Para que el aprobado fuera a partir de 5 bastaría con cambiar la instrucción *jle Comprobar\_nota\_maxima por jl Comprobar\_nota\_maxima* para que tanto si es > como si es = se incrementara Nro\_aprobados

Sabiendo que cuando finaliza la ejecución de la rutina **Teclear\_notas** el contenido de la variable **Lista\_notas** es: 00 h, 05 h, 06 h, 07 h, 03 h, 7F h, 00 h, ... 00 h y los valores del banco de registros son los siguientes:

Banco de registros					
AX	4F93 h	SI	0000 h	DS	4F93 h
BX	0000 h	DI	0000 h	ES	4F7A h
CX	0000 h	SP	00FE h	SS	4F95 h
DX	0000 h	BP	0000 h	CS	4F8A h
		IP	001F h		

- c. Indicar la dirección física de la instrucción **push SI** en la memoria principal del computador. Justificar la respuesta. **(0,2 puntos)**

$$DF = CS * 10h + IP = 4F8A0 h + 001F h = \mathbf{4F8BF h}$$

El valor de IP es justamente el dado en el Banco de registros pues push SI es la 1ª instrucción tras finalizar la ejecución de la rutina **Teclear\_notas**.

- d. Suponiendo que detenemos la ejecución del programa cuando el contador de programa apunta a la instrucción **pop BX**, indicar qué registros se han modificado y cuál es su contenido en dicho instante. Justificar la respuesta. **(0,2 puntos)**

**SI = 05** Se ha ido incrementando (partiendo de 0) hasta que lista\_notas[SI] apunte a 7F h

**BX = 03** Es la última nota hasta antes del caracter de fin “7F”

**IP** se habrá incrementado para apuntar a la dirección de la instrucción pop BX

El resto de registros queda como estaba.

- e. Suponiendo que el programa se ejecuta completamente, indicar para cada una de las variables del programa: **(0,8 puntos)**

- El nombre de dicha variable.
- La dirección que ocupa con respecto al inicio del segmento de datos.



- La dirección física que ocupa en la memoria principal del computador.
- El contenido de la variable.

**Sol:**

Nombre variable	Dir. Efectiva	Dir. Física	Contenido
Nro_alumnos	0 h	4F930 h	05 h
Nro_aprobados	1 h	4F931 h	02 h
Nota_maxima	2 h	4F932 h	07 h
Lista_notas	3 h	4F933 h	00 h, 05 h, 06 h, 07 h, 03 h, 7F h, 00 h, ... 00 h

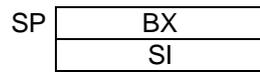
- f. Explicar cada uno de los cambios que se producen en la pila, desde el inicio del programa hasta su finalización. Justificar la respuesta. **(0,4 puntos)**

Con la instrucción *call Teclear\_notas* se guarda en la pila la dirección de la siguiente instrucción a ejecutar.

$SP \leftarrow SP - 2; [SP] \leftarrow IP$

Con la instrucción de retorno *ret* que se encontrará en la subrutina *call Teclear\_notas* se devuelve el valor de retorno a IP:  $[SP] \leftarrow IP; SP \leftarrow SP + 2$

Con las instrucciones *push SI* y *push BX* Se guarda el valor de los registros SI y BX en la pila:



Con las instrucciones *pop BX* y *pop SI* Se recupera el valor de los registros SI y BX.

**EJERCICIO 3:****(2 puntos)**

Se quiere diseñar una memoria, para un ordenador que tiene una arquitectura basada en palabras de 16 bits y un bus de direcciones con 20 líneas, con las siguientes características:

- 640 Kpalabras de memoria RAM
- 320 Kpalabras de memoria ROM

a partir de los siguientes módulos de memoria disponibles para realizar el circuito:

Memoria RAM	Memoria ROM
256 K x 8	128 K x 8
256 K x 16	128 K x 16
512 K x 8	256 K x 8

- a. Indicar, de manera justificada, si es posible realizar el diseño que se solicita.

**(0,2 puntos)**

Es posible realizar el diseño pues con 20 bits de bus de direcciones se pueden direccionar 1024 kpalabras y tenemos que direccionar 640 Kpalabras de memoria RAM y 320 Kpalabras de memoria ROM en total 960 Kpalabras.

- b. Calcular el menor número de módulos necesarios para realizar el circuito, con las especificaciones requeridas.

**(0,2 puntos)**

De Ram:

$640 \text{ K} / 256 \text{ K} \times 16 / 16 = 3 \times 1 = 3$  **pastillas de 256 K x 16** (De las cuales una no estará completamente utilizada).

De Rom:

$320 \text{ K} / 128 \text{ K} \times 16 / 16 = 3 \times 1 = 3$  **pastillas de 128 K x 16** (De las cuales una no estará completamente utilizada).



- c. Representar el mapa de memoria, teniendo en cuenta que la memoria RAM ocupa las posiciones más bajas del mapa de memoria y la memoria ROM las posiciones más altas. **(0,6 puntos)**

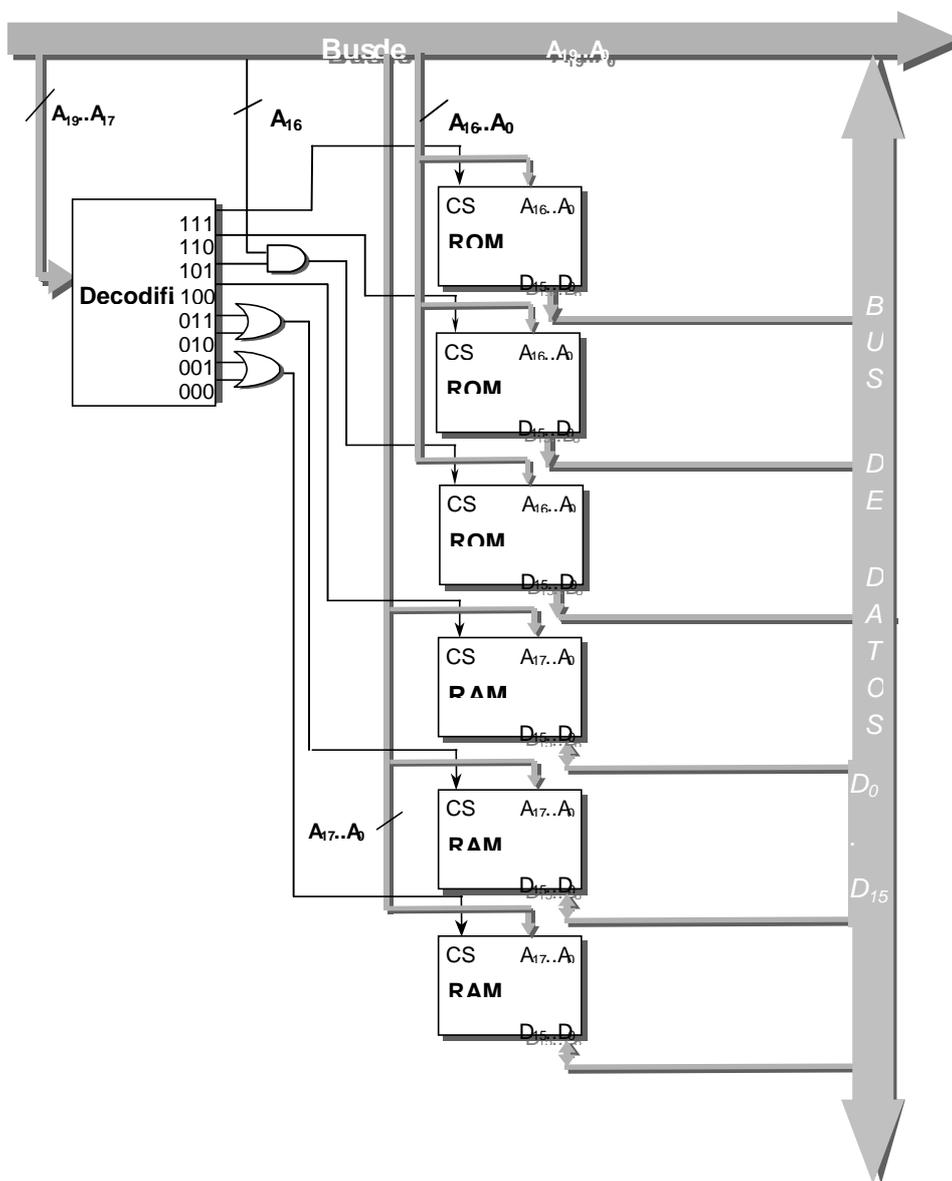
	$A_{19}$	$A_{18}$	$A_{17}$	$A_{16}$	$A_{15}$	...	$A_0$		
FFFFF h E0000 h	1	1	1	1	1	...	1	1024 K – 1 896 K	3ª pastilla ROM
DFFFF h C0000 h	1	1	0	1	1	...	1	896 K – 1 768 K	2ª pastilla ROM
BFFFF h B0000 h	1	0	1	1	1	...	1	768 K – 1 704 K	1ª pastilla ROM
AFFFF h A0000 h	1	0	1	0	1	...	1	704 K – 1 640 K	LIBRE
9FFFF h 80000 h	1	0	0	1	1	...	1	640 K – 1 512 K	3ª pastilla RAM
7FFFF h 40000 h 3FFFF h	0	1	1	1	1	...	1	512 K – 1 256 K	2ª pastilla RAM
00000 h	0	0	1	1	1	...	1	256 K – 1 0	1ª pastilla RAM

De la 1ª pastilla de Rom de 128k x 16 usamos solo 64k x 16.  
De la 3ª pastilla de Ram de 256k x 16 usamos solo 128k x 16.



d. Dibujar el circuito correspondiente.

(0,6 puntos)



e. Indicar, justificadamente, si con el diseño anterior se pueden realizar operaciones de acceso a todos los bytes de manera independiente. (0,4 puntos)

Si, De forma inequívoca se selecciona cada una de las direcciones correspondiente en las pastillas de Rom y Ram.