

A

Instrucciones generales para la realización de este examen

La respuesta debe escribirse en el hueco existente a continuación de cada pregunta **con letra clara**.

Cada respuesta correcta suma un punto. Cada respuesta incorrecta, ilegible o vacía no suma ni resta. El total de puntos se dividirá entre el total de preguntas y se multiplicará por 10 para obtener la nota del examen.

- ¿Cuál o cuáles de las siguientes afirmaciones son CIERTAS (puedes responder “ninguna” o “todas” si así lo consideras)?

- Al final de la rutina de servicio de una interrupción se tiene que ejecutar la instrucción **STI** para activar el flag de interrupción que se había puesto a cero durante la fase de aceptación.
- El valor de un vector de interrupción indica la prioridad de la interrupción asociada a ese vector.
- La línea **INTA** nunca se puede activar si no se ha activado la línea **INT** previamente.
- La rutina de servicio de una interrupción puede recibir parámetros a través de la pila.

C

Se quiere fabricar un circuito digital que sea capaz de contar el número de ceros que aparecen en un número binario de 3 bits. El circuito tiene 3 entradas por donde se recibe el número y 2 salidas en las que se obtiene el número de ceros codificado en binario. Por ejemplo, para la entrada 100 se obtiene la salida 10 (2 en binario) y para el número 101 se obtiene la salida 01 (1 en binario).

- Si se construye el circuito utilizando un PLA, ¿cuántas puertas AND y puertas OR serán necesarias? (Suponer que tanto las puertas AND como las OR pueden tener cualquier número de entradas).

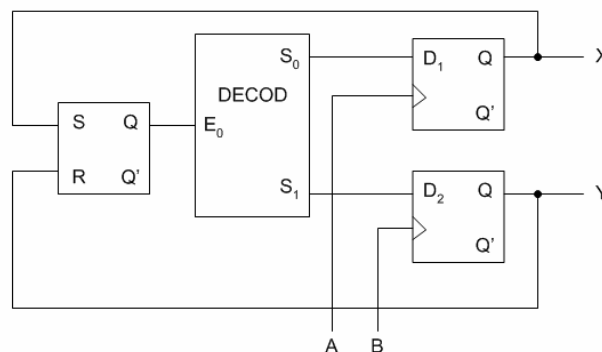
AND:

7

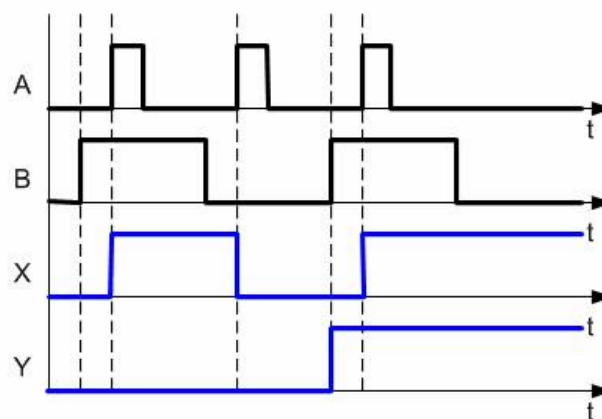
OR:

2

La siguiente figura muestra un circuito digital de dos entradas (A y B) y dos salidas (X e Y). Inicialmente la salida Q de todos los biestables está a cero.



- Completar el siguiente cronograma con los valores que tomarán las salidas X e Y a partir de las señales de entrada A y B:



El espacio de direcciones de un ordenador basado en la CPU elemental está organizado de la forma siguiente:

- La zona de direcciones más bajas está ocupada por un módulo de memoria de 32K
- Justo a continuación se encuentra un módulo de memoria de 16K
- El interfaz de teclado está mapeado a partir de la dirección E000h
- El interfaz de la pantalla se encuentra mapeado a partir de la dirección F000h

- En este ordenador se quiere cargar un programa que ocupa 1K en memoria. ¿Qué rango de valores se puede utilizar en la directiva ORIGIN de dicho programa? Ejemplo: 0A00h-A000h

0100h - BC00h

- Se dispone de chips de memoria de 4Kx2 para construir un módulo de memoria que cubra el espacio de direcciones completo de la CPU teórica. Si cada chip cuesta 0.50€, ¿cuánto costarán los chips necesarios para construir el módulo de memoria?

64€

En una entrada de una ALU de 8 bits se tiene el código ASCII de la letra 'J'. En la otra entrada se tiene el número 12d codificado en exceso a 8. Si la operación activa en la ALU es SUB,

- ¿qué número, interpretado en el formato signo-magnitud, se obtendrá en la salida de la ALU? Expresar el resultado en **decimal**. Nota: El código ASCII de la letra 'A' es 041h

54d

- ¿Cuál o cuáles de las siguientes afirmaciones son CIERTAS (puedes responder “ninguna” o “todas” si así lo consideras)?

- En una CPU del modelo Intel 286 no se puede ejecutar la instrucción **MOV EAX, 20h**
- La directiva **.MODEL** de Intel indica el modelo de CPU en el que se tiene que ejecutar el programa
- La instrucción **SAL** de Intel es equivalente a la instrucción **SHL**
- La instrucción **JB** de Intel es equivalente a la instrucción **JNGE**

A, C



Se pretende utilizar las capacidades multimedia y de control de periféricos que facilita la CPU teórica para desarrollar un sistema de control para un marcador/pantalla de un estadio deportivo. El marcador tiene las mismas características que el periférico **Pantalla** visto en la asignatura Las dimensiones físicas del dispositivo son las apropiadas para facilitar la visualización de los mensajes desde las gradas. El operario maneja el marcador a través un periférico de control similar al periférico **Luces** visto en la asignatura. El operario controlará con los interruptores del byte bajo la línea del marcador donde aparecerá el mensaje y con los del byte alto los atributos de color de texto y de fondo que presentará el mensaje.

Para comprobar el funcionamiento del dispositivo se ha escrito el siguiente código:

- **El programa principal:** Establece el vector de interrupción **0 (cero)** asociado al periférico **Luces** y su rutina.
- **rutina_luces:** Rutina que se ejecutará cuando el periférico **Luces** solicite una interrupción. Llamará a los tres procedimientos siguientes para escribir en la línea de pantalla indicada un texto concreto con unos atributos determinados.
- **PonAtributos:** Procedimiento para poner en todos los caracteres de una cadena los atributos de color de carácter y fondo. Se pasarán a través de la pila y en el orden indicado: dirección de la cadena, n° de caracteres de la cadena y los atributos en el byte más significativo de la palabra.
- **DireccionLineaPantalla:** Procedimiento que devolverá en el registro **R1** la dirección de comienzo de la línea de la pantalla pasada como parámetro a través de la pila. La línea cero será la primera, la uno la segunda, y así sucesivamente.
- **CopiaCadena:** Procedimiento que permite copiar una cadena de texto de una dirección de memoria a otra. Se pasarán como parámetros a través de la pila y en el orden indicado: dirección origen, dirección destino, n° de caracteres a copiar.

```
ORIGEN 300h
INICIO main
.PILA 30h
.DATOS
direcc_pantalla VALOR 0ff00h
direccRegCtrlPant VALOR XXXXh
direcc_luces VALOR 0ffd0h
cadena_texto VALOR "GOOOOOOOL!!!!"
ncaracteres VALOR 13
```

```
.CODIGO
```

```
PROCEDIMIENTO rutina_luces
; Salvar registros generales
PUSH R0
PUSH R1
PUSH R2
PUSH R3
PUSH R4
```

```
;;;;;;;;;;;;;;
; Leer el periférico luces
;
; R0 dirección variable direcc_luces
; R1 dirección periférico luces
; R4 contenido del periférico luces
```

```
MOVH R0,BYTEALTO DIRECCION direcc_luces
MOVL R0,BYTEBAJO DIRECCION direcc_luces
MOV R1,[R0]
MOV R4,[R1]
```

```
;;;;;;;;;;;;;;
; Preparar la llamada al procedimiento
; para poner atributos a toda la cadena
;
; R1 dirección de cadena_texto
; R2 número de caracteres
; R3 atributos
```

```
MOVH R1,BYTEALTO DIRECCION cadena_texto
MOVL R1,BYTEBAJO DIRECCION cadena_texto
MOVH R0,BYTEALTO DIRECCION ncaracteres
MOVL R0,BYTEBAJO DIRECCION ncaracteres
MOV R2,[R0]
```

```
; R0 mascara para leer atributo de R4
XOR R0,R0,R0
XXXXX R0,0FFh <----- 3a
```

```
; R3 atributo
AND R3,R0,R4
```

```
; Cargar los parámetros en la pila
----- HUECO 1a -----
```

```
; Llamada al procedimiento PonAtributos
CALL PonAtributos
```

```
; Limpiar la pila
----- HUECO 2a -----
```

```
; Preparar la llamada al procedimiento
; para averiguar la dirección de la línea
; de la pantalla
; R2 máscara para coger línea de R4
; R3 línea de pantalla
```

```
XOR R2,R2,R2
XXXXX R2,0FFh <----- 3b
AND R3,R2,R4
```

```
; Cargar los parámetros en la pila
----- HUECO 1b -----
```

```
; Lllamar al procedimiento
CALL DireccionLineaPantalla
; R1 contiene la dirección donde
; tiene que escribir en pantalla
```

```
; Limpiar la pila
----- HUECO 2b -----
```

```
; R2 dirección del registro de control de
; la pantalla
MOVH R0,BYTEALTO DIRECCION direccRegCtrlPant
MOVL R0,BYTEBAJO DIRECCION direccRegCtrlPant
MOV R2,[R0]
```

```
; Borrar la pantalla
XOR R0,R0,R0
----- HUECO 4 -----
```

```
;;;;;;;;;;;;;;
; Preparar la llamada al procedimiento
; para copiar la cadena a la pantalla
;
; R2 dirección de la cadena
; R1 ya contiene dirección línea pantalla
; R3 número de caracteres
MOVH R2,BYTEALTO DIRECCION cadena_texto
MOVL R2,BYTEBAJO DIRECCION cadena_texto
```

```
MOVH R0,BYTEALTO DIRECCION ncaracteres
MOVL R0,BYTEBAJO DIRECCION ncaracteres
MOV R3,[R0]
```

```
; Cargar los parámetros en la pila
----- HUECO 1c -----
```

```
; Lllamar al procedimiento CopiaCadena
CALL CopiaCadena
```

```
; Limpiar la pila
----- HUECO 2c -----
```

```
; Restaurar registros generales
```



```
POP R4
POP R3
POP R2
POP R1
POP R0
IRET

FINP

PROCEDIMIENTO CopiaCadena
PUSH R6
MOV R6,R7
PUSH R1
PUSH R2
PUSH R3
PUSH R4
; Recuperar de la pila los parámetros
; R3 Número de caracteres
; R2 dirección destino
; R1 dirección origen
----- HUECO 5 -----

bucle:
MOV R4,[R1]
MOV [R2],R4
; Incrementar dirección origen y destino
INC R1
INC R2
; Decrementar el número de caracteres
DEC R3
BRNZ bucle

POP R4
POP R3
POP R2
POP R1
POP R6
RET

FINP

PROCEDIMIENTO DireccionLineaPantalla
;;;;;;;;;;;;;;;;;;;;;;;;
; CÓDIGO OMITIDO INTENCIONADAMENTE
FINP

PROCEDIMIENTO PonAtributos
;;;;;;;;;;;;;;;;;;;;;;;;
; CÓDIGO OMITIDO INTENCIONADAMENTE
FINP

main:
XOR R0, R0, R0
MOVL R1, BYTEBAJO DIRECCION rutina_luces
MOVH R1, BYTEALTO DIRECCION rutina_luces
----- HUECO 6 -----
JMP -1

FIN
```

– ¿Qué instrucción o instrucciones faltan en **HUECO 1a** y **HUECO 1b** y **HUECO 1c**?

HUECO 1a	HUECO 1b	HUECO 1c
PUSH R1	PUSH R3	PUSH R2
PUSH R2		PUSH R1
PUSH R3		PUSH R3

– ¿Qué instrucción o instrucciones faltan en **HUECO 2a** y **HUECO 2b** y **HUECO 2c**?

HUECO 2a	HUECO 2b	HUECO 2c
INC R7	INC R7	INC R7
INC R7		INC R7
INC R7		INC R7

– Completa las instrucciones marcadas como **3a** y **3b**?

3a: MOVH

3b: MOVL

– ¿Qué instrucción/es falta/n en el **HUECO 4**?

```
MOVL R0,3
MOV [R2],R0
```

– ¿Qué instrucción/es falta/n en el **HUECO 5**?

```
INC R6
INC R6
MOV R3,[R6]
INC R6
MOV R2,[R6]
INC R6
MOV R1,[R6]
```

– ¿Qué instrucción/es falta/n en el **HUECO 6**?

```
MOV [R0],R1
STI
```

– ¿Cuál es el la dirección del registro de control de la pantalla y el rango de direcciones ocupadas por la pantalla? Contestar en **hexadecimal**.

REGISTRO CONTROL: FF78h

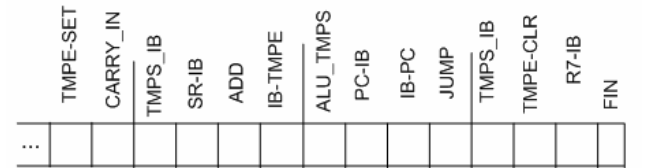
RANGO: FF00-FF7F

Se quiere añadir a las instalaciones multimedia del estadio un módulo de efectos sonoros, controlado también por el periférico **Luces**. Para activar el efecto sonoro se usarían los bits del byte menos significativo que no son necesarios para indicar la línea de pantalla. [Nota: La línea de pantalla se indica en los bits de menos peso de dicho byte].

– ¿Cuál sería entonces la máscara necesaria para obtener la línea de pantalla **Contestar en hexadecimal**.

Para indicar la última línea pondríamos un 7h, 3 bits son suficientes. La máscara entonces 0007h

En una CPU elemental microprogramada, el significado de los 12 bits inferiores de las palabras de control es el siguiente:



A continuación se muestra una tabla con los 12 bits inferiores de las palabras de control que se ejecutaron en los últimos tres pasos, junto con el valor que se encontraba en el bus interno (IB) en cada uno de estos pasos:

Paso	Palabra de control	Contenido de IB
4	0140h	1000h
5	0290h	00FAh
6	0029h	0FFBh

– ¿Cuál es la codificación de la instrucción que se acaba de ejecutar? Expresar el resultado en **hexadecimal**.

C0FAh

El grupo siderúrgico Arcelor, en colaboración con la Universidad de Oviedo, pretende desarrollar un módulo software capaz de validar soldaduras entre diferentes piezas de acero que se realizan en sus plantas de laminación. Las soldaduras que se pretenden validar se realizan utilizando un proceso de soldadura eléctrica. Una soldadura de estas características se puede validar a partir del estudio de dos señales analógicas participantes en la misma. Estas dos señales son:

- Temperatura que alcanza soldadura.
- Corriente aplicada al material a soldar.

En los laboratorios de la Universidad se propone un primer prototipo del módulo de validación de soldaduras desarrollado en lenguaje ensamblador para la arquitectura Intel. En este primer prototipo las señales se simplifican y se simulan con dos listas de números naturales:

- una lista de cinco elementos, **Temperatura**, que simula la señal de temperatura de un proceso de soldadura,
- una lista de cinco elementos, **Corriente**, que simula la señal de corriente de un proceso de soldadura.

Los límites mínimo y máximo para cada una de las señales se almacenan en las variables: **TmpInferior**, **TmpSuperior**, **CrrInferior** y **CrrSuperior**, que indican el límite inferior y el límite superior de la señal de temperatura y el límite inferior y el límite superior de la señal de corriente, respectivamente.

Se dispone de un procedimiento, **ChequeaSenyal**, que determina si los valores de una señal se encuentran dentro de unos límites máximo y mínimo. Para realizar este cálculo, el procedimiento debe recibir, en este orden, los siguientes parámetros:

- Dirección en la que almacenará el resultado del chequeo
- Dirección de la señal que va a chequear
- Límite superior para la validación
- Límite inferior para la validación

Este procedimiento recorre los elementos de la señal que se le indica como parámetro y comprueba si todos los valores de la misma están comprendidos entre los límites dados, es decir, mayores que el límite inferior y menores que el límite superior. Si es así, en la dirección en la que almacena el resultado del chequeo, escribirá un “1”. En caso contrario, en esa dirección escribirá un “0”.

El algoritmo para la validación de una soldadura invoca al procedimiento **ChequeaSenyal** una vez para cada una de las señales del proceso de soldadura que se debe validar.

Los resultados de estos chequeos estarán almacenados en **ResultTmp** y **ResultCrr**. Si el resultado de los chequeos de **ambas** señales es satisfactorio, la soldadura es correcta (se escribirá un “1” en la variable **ResultSoldadura**). En caso contrario, la soldadura es defectuosa (se escribirá un “0” en la variable **ResultSoldadura**).

```
.386
.MODEL FLAT
EXTERN ExitProcess:PROC

.DATA
Temperatura DD 787, 950, 975, 900, 715
Corriente   DD 14757, 15233, 15547, 14987, 14435

TmpInferior DD 750
TmpSuperior DD 950
CrrInferior DD 14500
CrrSuperior DD 16000
ResultTmp   DB 0
ResultCrr   DB 0
ResultSoldadura DD 0FFFFFFh

.CODE
inicio:
    ; Cargar parámetros en la pila
    ----- HUECO 1a -----

    ; Chequear la señal de Temperatura
    CALL  ChequeaSenyal

    ; Cargar parámetros en la pila
    ----- HUECO 1b -----

    ; Chequear la señal de Corriente
    CALL  ChequeaSenyal

    ; EAX <-- ResultTmp
    ; EBX <-- ResultCrr
    ; Calcular resultado de la soldadura en EAX
    ----- HUECO 2 -----

    ; Almacenar resultado del análisis
    MOV  EDI, OFFSET ResultSoldadura
    MOV  [EDI], EAX

    ; Fin del programa principal
    PUSH 0
    CALL  ExitProcess
    NOP
```

```
ChequeaSenyal PROC
    PUSH  EBP
    MOV   EBP, ESP

    ; Salvar registros generales
    PUSH  EAX
    PUSH  EBX
    PUSH  ECX
    PUSH  EDX
    PUSH  ESI
    PUSH  EDI

    ; Acceso a los parámetros
    ; EBX <-- Límite Inferior Señal
    ; EDX <-- Límite Superior Señal
    ; ESI <-- Dirección de Señal
    ; EDI <-- Dirección de resultado
    ----- HUECO 3 -----

    XOR   EAX, EAX
    XOR   ECX, ECX

bucle:
    ; EAX <-- Valor de la señal en
    ; la iteración actual
    ----- HUECO 4 -----

    CMP   EAX, EBX
    ; Si valor es menor que Límite
    ; Inferior, ir a limite_inferior
    ----- HUECO 5a -----

    CMP   EAX, EDX
    ; Si valor es mayor que Límite
    ; Superior, ir a limite_supeior
    ----- HUECO 5b -----

    ; Preparar siguiente iteración
    ; del bucle
    INC   ECX
    CMP   ECX, 5
    JNE   bucle
    JMP   senyal_correcta

limite_inferior:
limite_superior:
    ; Escribir resultado del chequeo de la señal
    ----- HUECO 6a -----
    JMP   resuelto

senyal_correcta:
    ; Escribir resultado del chequeo de la señal
    ----- HUECO 6b -----

resuelto:
    ; Restaurar registros generales
    ; y retornar al programa principal
```

ChequeaSenyal	ENDP
END	inicio

PUSH	OFFSET ResultTmp
PUSH	OFFSET Temperatura
PUSH	[TtmpSuperior]
PUSH	[TtmpInferior]

```
MOVZX    EAX, [ResultTmp]
MOVZX    EBX, [ResultCrr]
AND      EAX, EBX
```

```
MOV     EBX, [EBP+8]
MOV     EDX, [EBP+12]
MOV     ESI, [EBP+16]
MOV     EDI, [EBP+20]
```

```
MOV     EAX, [ESI+ECX*4]
```

HUECO 5a	JB	limite_inferior
HUECO 5b	JA	limite_superior

HUECO 6a	MOV	[EDI], BYTE PTR 0
HUECO 6b	MOV	[EDI], BYTE PTR 1

POP	EBP
RET	16

- A la vista de la misma, tras ejecutarse la instrucción CALL ChequeaSenyal, ¿en qué posición de memoria se almacenará el valor 00401036h?

0012FFB0h

```

:00401000 6838204000  ◆  CALL ChequeaSenyal          eax 00000000  c=0
:00401005 6800204000  ◆  CALL ChequeaSenyal          ebx 7FFDF000  z=0
:0040100A FF352C204000  ◆  CALL ChequeaSenyal          ecx 0012FFB0  s=0
:00401010 FF3528204000  ◆  CALL ChequeaSenyal          edx 7FFE0304  o=0
:00401016 E839000000  ◆  CALL ChequeaSenyal          esi 00001000  p=0
:0040101B 6839204000  ◆  CALL ChequeaSenyal          edi 00000005  a=0
:00401020 6814204000  ◆  CALL ChequeaSenyal          ebp 0012FFF0  i=1
:00401025 FF3534204000  ◆  CALL ChequeaSenyal          ds 0023
:0040102B FF3530204000  ◆  CALL ChequeaSenyal          es 0023
:00401031 E81E000000  ◆  CALL ChequeaSenyal          fs 0038
:00401036 0FBEE538204000  ◆  CALL ChequeaSenyal
:0040103D 0FBEE1D39204000  ◆  CALL ChequeaSenyal
:00401044 23C3  ◆  MOV EDI, OFFSET ResultSoldadura  ss 0023
:00401046 BF3A204000  ◆  MOV [EDI], EAX  cs 001B
:0040104B 8907  ◆  PUSH 0
:0040104D 6A00  ◆  CALL ExitProcess          :0012FFCC 00001000
:0040104F E84C000000  ◆  CALL ExitProcess          :0012FFC8 00000005

:00402000 13 03 00 00 B6 03 00 00 !!♥ â♥  ▲:0012FFC4 77E614C7
:00402008 CF 03 00 00 84 03 00 00 x♥ à♥  ■:0012FFC0 00402039
:00402010 CB 02 00 00 A5 39 00 00 u♥ Ñ9  ▒:0012FFBC 00402014
:00402018 81 3B 00 00 BB 3C 00 00 ü; ñ<  ▓:0012FFB8 00003E80
:00402020 8B 3A 00 00 63 38 00 00 i; c8  ▔:0012FFB4 000038A4

```