

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)?

- A) 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.
- B) El valor de un vector de interrupción indica la prioridad de la interrupción asociada a ese vector.
- C) La línea **INTA** nunca se puede activar si no se ha activado la línea **INT** previamente.
- D) La rutina de servicio de una interrupción puede recibir parámetros a través de la pila.

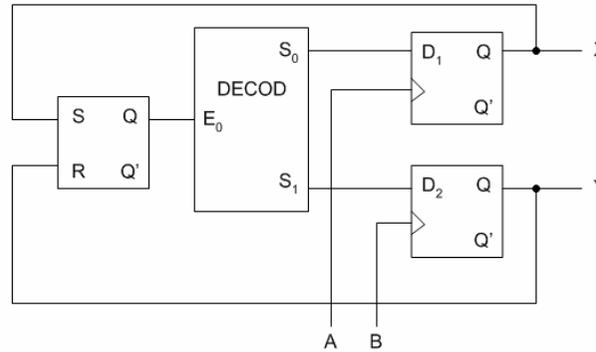
C

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).

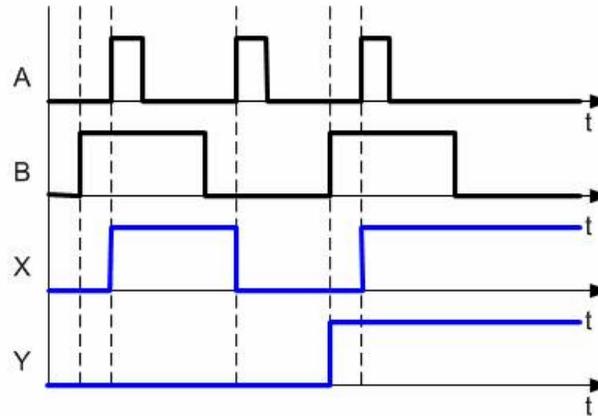
– Si se construye el circuito utilizando un PLA, ¿cuántas puertas AND y puertas OR serán necesarias? (Suponer que tanto las puertas AND como las OR pueden tener cualquier número de entradas).

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)?

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

A, C



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

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

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

```



Apellidos _____

Nombre _____

DNI _____



Examen de Fundamentos de los Computadores. Área de Arquitectura y Tecnología de Computadores

Final de Junio: 24-06-2005

```

POP     EDI
POP     ESI
POP     EDX
POP     ECX
POP     EBX
POP     EAX
----- HUECO 7 -----
ChequeaSenyal     ENDP
END inicio

```

¿Qué instrucción/es falta/n en el HUECO 1a?

```

PUSH     OFFSET ResultTmp
PUSH     OFFSET Temperatura
PUSH     [TmpSuperior]
PUSH     [TmpInferior]

```

¿Qué instrucción/es falta/n en el HUECO 2?

```

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

```

¿Qué instrucción/es falta/n en el HUECO 3?

```

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

```

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

```

MOV      EAX, [ESI+ECX*4]

```

¿Qué instrucción falta en el HUECO 5a y qué instrucción falta en el HUECO 5b?

```

HUECO 5a      JB      limite_inferior
HUECO 5b      JA      limite_superior

```

¿Qué instrucción falta en el HUECO 6a y qué instrucción falta en el HUECO 6b?

```

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

```

¿Qué instrucción/es falta/n en el HUECO 7?

```

POP     EBP
RET     16

```

La figura muestra un instante de la ejecución del programa de validación de soldaduras utilizando el depurador Turbo Debugger (TD32.EXE).

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  ◆ MOV     EDI, OFFSET ResultTmp          eax 00000000  c=0
:00401005 6800204000  ◆ MOV     EDI, OFFSET Temperatura       ebx 7FFDF000  z=0
:0040100A FF352C204000 ◆ MOV     [TmpSuperior]                 ecx 0012FFB0  s=0
:00401010 FF3528204000 ◆ MOV     [TmpInferior]                 edx 7FFE0304  o=0
:00401016 E839000000  ◆ CALL   ChequeaSenyal                  esi 00001000  p=0
:0040101B 6839204000  ◆ MOV     [SenyalCrr]                   edi 00000005  a=0
:00401020 6814204000  ◆ MOV     [SenyalCrr]                   edi 00000005  i=1
:00401025 FF3534204000 ◆ MOV     [SenyalCrr]                   edi 00000005  d=0
:0040102B FF3530204000 ◆ MOV     [SenyalCrr]                   edi 00000005
:00401031 E81E000000  ◆ CALL   ChequeaSenyal                  ds 0023
:00401036 0FBEE0538204000 ◆ MOV     [SenyalCrr]                   es 0023
:0040103D 0FBED139204000 ◆ MOV     [SenyalCrr]                   fs 0038
:00401044 23C3        ◆ MOV     EDI, 0                         ss 0023
:00401046 BF3A204000  ◆ MOV     EDI, OFFSET ResultSoldadura   cs 001B
:0040104B 8907        ◆ MOV     [EDI], EAX
:0040104D 6A00        ◆ PUSH   0
:0040104F E84C000000  ◆ CALL   ExitProcess                    :0012FFCC 00001000
:00402000 13 03 00 00 B6 03 00 00 !!♥  â♥      :0012FFC8 00000005
:00402008 CF 03 00 00 84 03 00 00 x♥  ä♥      :0012FFC4 77E614C7
:00402010 CB 02 00 00 A5 39 00 00 7♥  ñ9      :0012FFC0 00402039
:00402018 81 3B 00 00 BB 3C 00 00 ü;  π<      :0012FFBC 00402014
:00402020 8B 3A 00 00 63 38 00 00 i:  c8      :0012FFB8 00003E80
:00402022 8B 3A 00 00 63 38 00 00 i:  c8      :0012FFB4 000038A4

```