

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) En la CPU elemental, la instrucción **POP R1** tarda tanto en ejecutarse como la instrucción **PUSH R1**.
- B) A partir del código ASCII extendido de la letra ‘t’, se puede deducir el código ASCII extendido de todas las letras mayúsculas y minúsculas
- C) Las instrucciones de desplazamiento de Intel utilizan el bit CF del registro de estado.
- D) La directiva del ensamblador de Intel **DUP** equivale a la directiva **VALOR** del ensamblador de la CPU elemental

C

Se sabe que las entradas de la ALU de 16 bits vista en clase tienen los siguientes valores:

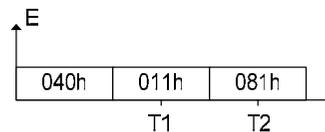
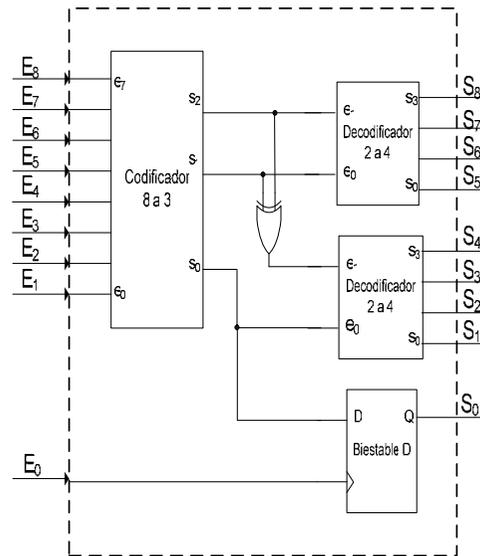
- En el **operando A** se encuentra la combinación de bits que representa el cero en formato exceso a Z con 16 bits y exceso central.
- En el **operando B** se encuentran los 16 bits más altos de la codificación del número $2^{-127}(1+2^{-1}+2^{-2})$ en formato IEEE-754.
- **CarryIn** = 0, **Resta** = 1, **Op1** = 1 y **Op0** = 1.

Se recuerda que las ALUs elementales que componen la ALU de 16 bits tienen conectadas las salidas de una puerta AND, de una puerta OR, de una puerta XOR y el resultado de un sumador elemental, a las entradas e_0 , e_1 , e_2 y e_3 del multiplexador, respectivamente.

– ¿Cuál es el resultado obtenido en la salida? Expresar el resultado con **cuatro dígitos hexadecimales**.

7F 90h

– El circuito de la figura implementa un sistema secuencial de nueve entradas y nueve salidas. Se sabe que el instante inicial $t=0$, la salida S_0 se encuentra a cero. A partir de dicho instante las entradas evolucionan tal como se indica en el cronograma. ¿Cuál será la salida en el instante T_1 y T_2 ? (Expresar el resultado en hexadecimal)



S_{T1} : 051h S_{T2} : 103h

Se quiere fabricar un circuito digital que sea capaz de contar el número de dobles ceros (dos ceros contiguos) que aparecen en un número binario de 4 bits. El circuito tiene 4 entradas por donde se recibe el número y 2 salidas en las que se obtiene el número de dobles ceros codificado en binario. Por ejemplo, para la entrada 0100 se obtiene la salida 01 (1 en binario) y para el número 0000 se obtiene la salida 10 (2 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: 8 OR: 2

En un computador basado en la CPU elemental se tiene el siguiente mapa de memoria:

- Dispositivo de memoria RAM a partir de la posición de memoria 0000h y tamaño 48K
- Dispositivo de memoria ROM a partir de la posición de memoria C800h y tamaño 14K

– Se desea añadir al computador un dispositivo de entrada cuyo interfaz requiere 2K dentro del espacio de direcciones. ¿A partir de qué posición de memoria mapearías dicho interfaz? Contestar en hexadecimal.

C000h

– Se pretende diseñar un dispositivo de organización 16Kx8 a partir de chips de organización 2Kx4. ¿Qué tipo de decodificador sería necesario dentro del dispositivo de memoria? Ejemplo de respuesta: 1/2

3/8



Se pretende crear un sistema de monitorización del flujo de vehículos de una carretera concreta. El sistema se desarrolla para un computador basado en la CPU elemental al que se le conecta un dispositivo periférico. Este periférico posee dos displays de visualización y varios sensores encargados de detectar el paso de vehículos.

El primer prototipo del sistema se va a instalar en una carretera de dos carriles y dos sentidos de circulación (sentido A y sentido B). El periférico genera una interrupción asociada al vector 5 cada vez que detecta un vehículo circulando en algún sentido. La CPU atiende las peticiones de interrupción por medio de la rutina de servicio Rut.

El interfaz del periférico consta de tres registros, uno de control y dos de datos (RegA y RegB), mapeados en este orden a partir de la dirección 7000h, que realizan las siguientes funciones:

- Registro de Control: El bit más significativo de este registro indica el sentido de circulación del vehículo detectado: 1 = sentido A, 0 = sentido B.
- RegA: Almacena el número de vehículos detectados circulando en sentido A.
- RegB: Almacena el número de vehículos detectados circulando en sentido B.

Cuando se produce una interrupción, la rutina de tratamiento comprueba, en primer lugar, el sentido de circulación del vehículo detectado y, a continuación, actualiza el registro de datos necesario (regA o regB) en función de dicho sentido de circulación.

NOTA: Al inicio de la ejecución del programa, el valor de RegA y de RegB es igual a 0.

```

ORIGEN    500h
.INICIO   ini
.PILA     ???h

.CODIGO
ini:
; Instalar rutina de tratamiento de
; la interrupción
    MOVL    R0, BYTEBAJO DIRECCION Rut
    MOVH    R0, BYTEALTO DIRECCION Rut
    ----- HUECO 1 -----
; Activar las interrupciones
    STI
ejecuta:
    JMP     ejecuta
  
```

```

PROCEDIMIENTO Rut
    PUSH    R0
    PUSH    R1
    PUSH    R2
; Leer el registro de control
    ----- HUECO 2 -----
    MOV     R1, [R0]

; Comprobar sentido de circulación
    ----- HUECO 3 -----
    AND     R2, R2, R1

; Si no circula en sentido A, ir a
sentidoB
    ----- HUECO 4 -----

sentidoA:
    INC     R0
; Actualizar número de vehículos
    ----- HUECO 5 -----
    JMP     procesado

sentidoB:
    INC     R0
    INC     R0
; Actualizar número de vehículos
    ----- HUECO 6 -----

procesado:
    ----- HUECO 7 -----

FINP
FIN
  
```

- ¿Cuál es el valor **mínimo** que debe sustituir a ??? en la directiva .PILA para que el programa se ejecute correctamente?

```
X = 5h
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 1?

```

    MOVL    R1, 05h
    MOVH    R1, 00h
    MOV     [R1], R0
  
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 2?

```

    MOVL    R0, 00h
    MOVH    R0, 70h
  
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 3?

```

    MOVL    R2, 00h
    MOVH    R2, 80h
  
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 4?

```
BRZ     sentidoB
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 5?

```

    MOV     R2, [R0]
    INC     R2
    MOV     [R0], R2
  
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 6?

```

    MOV     R2, [R0]
    INC     R2
    MOV     [R0], R2
  
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 7?

```

    POP     R2
    POP     R1
    POP     R0
    IRET
  
```

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

- A) Antes de la ejecución de la primera instrucción de una rutina de interrupción se puede aceptar una nueva interrupción.
- B) Cuando se acepta una interrupción se salva en la pila el **PC** y el **SR** (en este orden).
- C) Los periféricos hacen la petición de interrupción a través de la línea **INTA**.
- D) El bit **IF** a cero permite aceptar la petición de interrupción.

Ninguna

La siguiente secuencia de señales de control corresponde a una instrucción de la CPU elemental. Los pasos están desordenados.

Pasos	Acciones de Control
A	MDR_IB, IB_IR
B	TMPS_IB, IB_R5, FIN
C	TMPS_IB, IB_PC
D	PC_IB, IB_MAR, TMPE_CLR, CARRY_IN, ADD, ALU_TMPS, READ
E	R5-IB, TMPE_CLR, CARRY_IN, ADD, ALU_SR, ALU_TMPS

– ¿Cuál es la codificación de la instrucción que se acaba de ejecutar (Expresar el resultado en hexadecimal)?
¿Cuál es el orden adecuado de los pasos?

Codificación: 8D00h

Orden: D,C,A,E,B

Canal Plus codifica su señal de vídeo desordenando las líneas que forman la imagen. Por lo tanto, el decodificador lo único que tiene que hacer es restaurar el orden original de las líneas a partir de la secuencia de codificación que se utilizó sobre la imagen original. Por ejemplo, si se tiene una imagen codificada de 4

líneas (L0,L1,L2,L3) y la secuencia de decodificación es 3,1,2,0 entonces la imagen decodificada es L3,L1,L2,L0. En la realidad, las imágenes constan de 576 líneas, cada una de ellas con 480 puntos de tamaño doble palabra. Por lo tanto, la secuencia de decodificación es una lista de 576 números de 4 bytes cada uno.

A continuación se muestran algunos fragmentos del programa que permite decodificar la señal de vídeo de Canal Plus. El procedimiento *Decodifica* se utiliza para decodificar una imagen. A este procedimiento se le pasan tres parámetros por la pila (en el orden indicado):

- Dirección de la imagen codificada
- Dirección de la imagen decodificada
- Dirección de la secuencia de decodificación

El procedimiento recorre la imagen codificada copiando cada una de sus líneas a la imagen decodificada en la posición indicada por la secuencia de decodificación. Para ello se utilizan dos procedimientos auxiliares:

- *BuscaLinea* calcula la dirección de memoria en la que se encuentra la línea N de una imagen. Recibe como parámetros el número de línea (N) y la dirección de la imagen, devolviendo en EAX la dirección de la línea buscada.
- *CopiaLinea* copia los puntos de una línea a otra línea. Se le pasan como parámetros la dirección de la línea destino y la dirección de la línea original (en el orden indicado).

Procedimiento Decodifica:

```
Decodifica PROC
```

```
PUSH EBP
MOV EBP, ESP
```

```
;Guardar el valor original de los
;registros
PUSH ESI ;Dir. imagen codificada
PUSH EDI ;Dir. imagen decodificada
PUSH ECX ;Numero de líneas
PUSH EBX ;Dir. secuencia codificación
```

```
;Recuperar parámetros de la pila
--- HUECO 1 ---
```

```
;Cargar nº de líneas en la imagen
MOV ECX, 576
```

```
bucle_1:
```

```
;Obtener la dirección de la línea N en la
;imagen decodificada
PUSH DWORD PTR [EBX]
PUSH EDI
CALL BuscaLinea
```

```
;Copiar la línea actual de la imagen
;original a la imagen decodificada
```

```
--- HUECO 2 ---
CALL CopiaLinea
```

```
;Pasar al siguiente numero de la
;secuencia de decodificación
ADD EBX, 4
```

```
;Pasar a la siguiente línea de
;la pantalla
ADD ESI, 1920; 480 líneas x 4 bytes
```

```
???? bucle_1 <--- 5a
```

```
POP EBX
POP ECX
POP EDI
POP ESI
POP EBP
RET 12
```

```
Decodifica ENDP
```

Procedimiento CopiaLinea:

```
CopiaLinea PROC
```

```
PUSH EBP
MOV EBP, ESP
```

```
;Guardar el valor original de los registros
PUSH EAX ;Almacenamiento temporal
PUSH EBX ;Punto de la línea a copiar
```

```
--- HUECO 3 ---
```

```

;Recuperar parámetros de la pila
MOV ESI, [EBP+8] ;Dir. línea origen
MOV EDI, [EBP+12] ;Dir. línea destino
XOR EBX, EBX

bucle_3:
;Copiar punto de línea original a destino
MOV EAX, ???? <----- 4a
MOV ???? , EAX <----- 4b

;Pasar al siguiente punto
INC EBX

;Comprobar si se ha llegado al final
CMP EBX, 480 ;480 puntos en la línea
???? bucle_3 <---- 5b

POP EDI
POP ESI
POP EBX
POP EAX
POP EBP
--- HUECO 6 ---
CopiaLinea ENDP
    
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 1?

```

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

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 2?

```

PUSH EAX
PUSH ESI
    
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 3?

```

PUSH ESI
PUSH EDI
    
```

- Completar las instrucciones marcadas como 4a y 4b

```

4a: [ESI+4*EBX]
4b: [EDI+4*EBX]
    
```

- Completar las instrucciones marcadas como 5a y 5b

```

5a: LOOP          5b: JB, JNZ, JNAE, JL, JNGE, JNE
    
```

- ¿Qué instrucción o instrucciones faltan en el hueco etiquetado como HUECO 6?

```

RET 8
    
```

La figura muestra un momento de la ejecución del procedimiento Decodifica utilizando el Turbo Debugger (TD32.EXE)

- ¿En qué dirección de memoria se encuentra la imagen codificada? Contestar en hexadecimal

```

00402024h
    
```

- ¿Qué línea de la imagen codificada se está procesando en el instante mostrado en la figura? NOTA: Las líneas se numeran desde cero

```

4
    
```

The screenshot displays the Turbo Debugger interface. On the left, the assembly window shows the following instructions:

```

Hexamen#Decodifica
:00401000 55          * PUSH EBP
:00401001 8BEC       * MOV EBP, ESP
:00401003 56          * PUSH ESI ;Direccion imagen codi
:00401004 57          * PUSH EDI ;Direccion imagen deco
:00401005 51          * PUSH ECX ;Numero de lineas
:00401006 53          * PUSH EBX ;Direccion secuencia d
:00401007 8B7510     *
:00401009 8B7D0C     *
:0040100B 8B5D00     *
:00401010 B9402000   * MOV ECX, 576
:00401015 7405 JNZ EBX, 0040101A
:00401017 57          * PUSH EDI
:00401018 E81A000000 * CALL Buscalinea
:0040101D 50          *
:0040101E 56          *
:0040101F E832000000 * CALL Copialinea
:00401024 83C304     * ADD EBX, 4
:00401027 81C680070000 * ADD ESI, 1920
:0040102D E2E6     * bucle_1
:0040102F 5B          * POP EBX
    
```

On the right, the register window shows the following values:

```

eax 00512424 c=0
ebx 00402010 z=0
ecx 0000023C s=0
edx 7C91EB94 o=0
esi 00403E24 p=1
edi 00510024 a=0
ebp 0012FFB0 i=1
esp 0012FFA0 d=0
ds 0023
es 0023
fs 003B
gs 0000
ss 0023
cs 001B
eip 00401015
    
```

At the bottom, the memory window shows the address 00402024 with the value 00 00 00 00.