

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.

Se ha escrito un programa en ensamblador para la CPU teórica. El programa realiza una encriptación **XOR** de una cadena de caracteres. Esta codificación es ampliamente utilizada por los virus para encriptarse y mutar, dificultando así la tarea de los programas de detección. Además de la rapidez y sencillez de la implementación se trata de una codificación **reversible**.

- El programa parte de una cadena de caracteres y de una contraseña, también de caracteres, y genera otra cadena de caracteres ya codificada.

- Se recorre la contraseña tantas veces como sea necesario para completar la longitud de la cadena a encriptar. Se realiza la operación OR-exclusiva entre el carácter a codificar y el carácter de la contraseña que ocupa su misma posición.

Este es un fragmento del programa, las líneas que comienzan por **punto y coma** son comentarios:

```
...
;R0 direccion de la cadena a encriptar
MOVH R0, 10h
MOVL R0, 0h
;R1 n° caracteres a encriptar
MOVH R1, 0
MOVL R1, 18
;R2 direccion de la clave a utilizar
MOVH R2, 10h
MOVL R2, 12h
```

```
;R3 n° caracteres de la clave
MOVH R3, 0
MOVL R3, 10

;R7 direccion cadena encriptada
MOVH R7, 10h
MOVL R7, 1Ch

;Inicio Bucle
;R5 caracter a codificar
MOV R5, [R0]
;R6 caracter clave usado para codificar
MOV R6, [R2]
;R4 almacenaremos el resultado del XOR
XOR R4, R5, R6
;Escribimos el caracter encriptado
MOV [R7], R4 ③

;Avanzamos un caracter
①

;¿hemos terminado la clave?
;afirmativo: reiniciamos la clave
;negativo: comprobamos si hemos terminado
;la cadena
DEC R3
②

;reiniciamos la clave
;R2 direccion de la clave a utilizar
MOVH R2, 10h
MOVL R2, 12h
;R3 n° caracteres de la clave
MOVH R3, 0
MOVL R3, 10

;comprobamos si hemos terminado la cadena

DEC R1
BRNZ -15 ;repetimos el bucle

JMP -1
...
```

— ¿Qué instrucción o instrucciones falta/n en ①?

INC R0 ;Avanzamos un caracter
INC R2 ;también en la clave
INC R7; también en la cadena resulta

— ¿Qué instrucción o instrucciones falta/n en ②?

BRNZ +4

— ¿Qué dirección de memoria será modificada la última vez que se ejecute la instrucción③?
Contestar en hexadecimal.

Dirección: 102Dh

— ¿Cuál es la codificación en **base 8** de la instrucción③?

17600

— Sabiendo que la palabra usada para codificar ha sido "**contraseña**", que el resultado de la encriptación del decimotercer carácter ha sido **0h** y que el código ASCII de '**c**' es **0063h**. ¿cuál es el decimotercer carácter de la frase original?

'n'

— Completa los caracteres que faltan en el interior de las casillas, teniendo en cuenta los caracteres que ya hay y sus códigos ASCII.

45h 65h 69h 6Dh 30h 35h

E	e	i	m	0	5
---	---	---	---	---	---

— Se pretende realizar una función lógica que detecte la presencia de dos o más unos en una secuencia de tres bits. Expresa la función como suma de minterms. Ejemplo: $S = m_0 + m_3 + m_7$
NOTA: Los subíndices de "m" indican el valor decimal de las entradas.

$S = m_3 + m_5 + m_6 + m_7$

— Se tiene un formato de coma flotante con 1 bit para el signo de la mantisa, n para la magnitud de la mantisa y 4 bits para el exponente. La mantisa se representa en signo-magnitud, todo fracción y se representa el primer bit, que será siempre 1. El exponente se representa en exceso a 8. Se sabe que el mayor número representable en este formato es 112. ¿Cuánto vale n ?

3

— Cuáles de las siguientes afirmaciones son **CIERTAS**?

- A. En exceso a Z central pueden codificarse 2^n números reales.
- B. La CPU elemental puede construirse utilizando sólo puertas lógicas.
- C. El registro de instrucción de una CPU forma parte de su camino de datos
- D. En una RAM dinámica se puede almacenar la misma cantidad de información que en una RAM estática pero por menos dinero.

D

— ¿Cuáles de las siguientes afirmaciones son **CIERTAS**?

- A. El código ASCII estándar permite imprimir 128 caracteres.
- B. Un decodificador con 2^n entradas tiene n entradas de selección.
- C. El uso de PLAs facilita la construcción de Unidades de Control cableadas.
- D. El número de bits de una palabra de control depende del paso de instrucción que se esté ejecutando en un instante dado

C

— Por una de las entradas de una ALU de siete bits se introduce el código ASCII de la letra "B", y por la otra, el código ASCII de la letra "b". Sabiendo que el ASCII de la "A" es 65d y que el código ASCII de la "c" es 99d, ¿qué número **binario** se obtiene a la salida de la ALU al realizar una operación **XOR**?

0100000

— En una ALU de 6 bits se introducen el número -2 en complemento a 2 y el número 4 en signo-magnitud. ¿Qué número en base diez se obtiene al realizar una **SUMA** si el resultado binario obtenido se interpreta como un número codificado en exceso a $Z=8$?

-6

— ¿Cuántas entradas y cuántas salidas tiene una ALU de cuatro bits idéntica a la construida en las prácticas de la asignatura?

Entradas: 12 Salidas: 8

— Un multiplexador con 2^n canales de entrada maneja más señales digitales (entradas y salidas) que un decodificador con n entradas. ¿Cuántas más?

1

— Si una CPU con un registro de instrucciones de 16 bits permite utilizar 256 instrucciones diferentes, ¿cuántos bits como máximo se pueden dedicar a la codificación del direccionamiento inmediato?

8

— Se crea una nueva instrucción MOV [R1], [R2], que mueve a la posición de memoria apuntada por R1 el contenido de la dirección de memoria apuntada por R2. Escribir a continuación las señales de ejecución que deben activarse a partir del paso 4 de esta nueva instrucción

Paso	Señales activas
4	R2-IB, IB-MAR, READ
5	
6	R1-IB, IB-MAR, WRITE
7	FIN
8	
9	
10	

— ¿Cuál será el menor número de bits que se podría utilizar para representar los números $-2_{(10)}$ y $-30_{(10)}$ en complemento a 2 de forma que su suma no produzca overflow? ¿Y para que no se produzca acarreo?

NOTA: Si no es posible evitar el overflow o el acarreo al hacer la suma, responder 0 bits.

Overflow: 6 Acarreo: 0

