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 dispone de un sistema de control de acceso a un Centro de Proceso de Datos (CPD). El sistema consta de un dispositivo compacto, formado por un teclado y una pantalla de 5 filas por 25 columnas, colocado al lado de la puerta de acceso al CPD. El hardware del sistema se ha desarrollado basado en la CPU elemental y el software está programado en ensamblador.

Cuando una persona quiere acceder al CPD, teclea su código en el teclado, la interfaz del teclado encripta el código en 16 bits y se envía al registro de datos. A continuación, se muestra en pantalla si la persona tiene o no acceso autorizado y se abre la puerta en caso positivo.

El procedimiento acceso, del que se muestra una parte en listado anexo, pertenece al código desarrollado, que tiene estas características:

a.- Mediante muestreo periódico del registro de control/estado se detecta que se ha introducido un código y ha sido almacenado.

b.- En este caso se ejecutará el procedimiento acceso, que comprueba si el código está en la lista de códigos autorizados. Si es así, se muestra la información de aceptación o rechazo en pantalla y se abre la puerta en caso necesario.

c.- Los registros de datos y de control/estado del lector se mapean en ese orden y a partir de la dirección C000h.

d.- Un registro de control/estado, de 16 bits, que dispondrá de los siguientes bits:

- Bit 0: se pone a 1 cuando se desee abrir la puerta.
- Bit 2: se pone a 1 cuando el lector lee una tarjeta y su código de acceso es almacenado en el registro de datos. Se pone a 0 cuando la CPU lee el registro de datos.

```
PROCEDIMIENTO acceso

;Guardar el valor de los registros
utilizados por la rutina
PUSH R0
PUSH R1
PUSH R2
PUSH R3
PUSH R4
PUSH R5

; Cargar en R1 la dirección del registro de
estado/control de la pantalla
MOVH R1, 0FFh
MOVL R1, 0FDh

; Borrar el contenido de la pantalla
MOVL R2, 03h
MOVH R2, 00h
MOV [R1], R2

; Cargar en R0 el contenido del registro de
; datos del lector
--1--

; Cargar en R2 la dirección del primer
; código de la lista
MOVL R2, BYTEBAJO DIRECCION lista
MOVH R2, BYTEALTO DIRECCION lista
; Inicialización número de códigos de la
; lista
MOVL R3, 16h
MOVH R3, 00h

; Cargar en R5 la dir. de la primera
; posición de pantalla
MOVH R5, 0FFh
MOVL R5, 80h

; Bucle de aceptación o denegación de
; permiso
bucle:
MOV R4, [R2]
COMP R0, R4
BRNZ salto

;.............
FINP
```

1. ¿Qué instrucción/es falta/n en el hueco --1--?

MOVH Rx, 00h

MOVL Rx, 0C0h

MOV R0, [Rx]

2. ¿Cuál es la función del circuito de activación de la pantalla? Ejemplo de respuesta: $a_{15} \cdot \ldots \cdot a_{11} \cdot \overline{a}_{10}$

$a_{15} \cdot \ldots a_7$

3. Justo antes de ejecutarse la instrucción **MOV R4, [R2]**, parte del contenido de la pila era el siguiente (de posiciones de memoria mayores a menores): A001h, B010h, 1000h, 0001h, F0FFh, 8147h, con R7 apuntando a 8147h ¿Cuál era el valor del registro R4 en el programa principal justo antes de acceder al procedimiento?

F0FFh

4. Se ha modificado el teclado para que una vez introducido el código, genere de forma automática una interrupción. A la rutina de interrupción se le ha asignado el vector de interrupción 7. Dar el código necesario para instalar **acceso** como rutina de interrupción.

MOVL Rx, 07h

MOVH Rx, 00

MOVL Ry, BYTEBAJO DIRECCION acceso

MOVH Ry, BYTEALTO DIRECCION acceso

MOV [Rx], Ry

5. En la primera ejecución del nuevo programa se produce una interrupción y durante la fase de desencadenamiento (antes de ejecutar ninguna instrucción) de la misma aparecen en la pila los siguientes datos: F001h, 0001h, 0502h, con R7 apuntando a 0502h. ¿Cuál era la dirección de la instrucción que se estaba ejecutando en el programa principal cuando se produce la interrupción?

0501h

6. ¿Qué dato enviará el periférico a través del bus de datos en esta fase de aceptación de la interrupción? Responder en hexadecimal.

0007h

Se ha realizado un procedimiento, **FormateaFecha**, en ensamblador de Intel x86 de 32 bits para formatear fechas, pasando por ejemplo de **02/07/2007** a **02/Jun/2007**. El procedimiento recibe por la pila estos parámetros:

- La dirección de una fecha con formato **dd/mm/aaaa**, donde **dd** es el número de día, **mm** es el número de mes y **aaaa** es el número de año. Todos los números están representados como decimales en ASCII.

- Una dirección de memoria en donde copiará la fecha pero cambiará el número de mes por el conjunto de los tres caracteres iniciales del nombre del mes.

Para cambiar un número de mes por sus tres iniciales, se dispone del procedimiento **ObtenerMes**. Este procedimiento recibe por la pila dos parámetros:

- La dirección con la cadena que representa el número de mes.

- La dirección donde debe dejar las iniciales.

Para hacer la transformación de números a letras, el procedimiento **ObtenerMes** primero pasa el número de ASCII a decimal y a partir de ese número calcula el índice en la tabla **Meses** donde se encuentran las iniciales del mes buscado. Como las iniciales de cada mes tienen 3 caracteres, el mes 1 ocupará las posiciones 0 a 2; el mes 2, las posiciones 3 a 5, etc.

Para probar los procedimientos se ha realizado el programa mostrado en el listado siguiente:

```
.386
.model flat, stdcall
ExitProcess proto, ExitCode: dword

.data
  Meses DB "Ene", "Feb", "Mar", "Abr",
    "May", "Jun", "Jul", "Ago", "Sep",
    "Oct", "Nov", "Dic"
  FechaOrigen1 DB "02/06/2007"
  FechaOrigen2 DB "31/12/2007"
  FechaDestino1 DB 11 DUP (0)
  FechaDestino2 DB 11 DUP (0)

.code

FormateaFecha proc
  push ebp
  mov ebp, esp
  push eax
  push ebx
  push ecx
  push edx

  mov eax, [ebp+12] ; Dir. cad. origen
  mov ebx, [ebp+8]  ; Dir. cad. resultado

  xor ecx, ecx

; Copiar el día, sin cambiarlo, de la
; cadena de origen a la cadena de destino
CopiaDia:
  mov dl, [eax+ecx]
  -- 1 --
  inc ecx
  cmp ecx, 3
  jne CopiaDia

  ; Copiar mes transformado en texto
  mov edx, eax
  add edx, ecx
  push edx
  mov edx, ebx
  add edx, ecx
  push edx
  call ObtenerNombreMes

  add ecx, 2 ; Pasar al año
```

```
CopiaAgno:
  mov dl, [eax+ecx]
  mov [ebx+ecx+1], dl
  inc ecx
  cmp ecx, 10
  jne CopiaAgno

  pop edx
  pop ecx
  pop ebx
  pop eax
  pop ebp
  ret 8
FormateaFecha endp

ObtenerNombreMes proc
  push ebp
  mov ebp, esp
  push eax
  push ebx
  push ecx
  push edx
  push esi

  ; Sacar a eax la dir. cad. resultado
  ; y a ebx el número de mes en ASCII
  -- 2 --

  ; Pasar el número de mes de ASCII
  ; a número (en cl)

  xor ecx, ecx
  mov dl, [ebx]

  ; Sumar diez si empieza por '1'
  cmp dl, '1'
  -- 3 --
  mov cl, 10

noEsUno:
  mov dl, [ebx+1]
  sub dl, '0'
  add cl, dl

  ; Hacer que ecx apunte al nombre del mes
  ; El mes 1 tiene índice 0; el 2, 1; etc.
  dec ecx
  ; Multiplicar ecx por tres porque cada
  ; nombre tiene tres caracteres
  mov esi, ecx
  add ecx, esi
  add ecx, esi

  xor ebx, ebx ; índice del car. a copiar
CopiaCaracterMes:
  mov dl, [Meses + ecx + ebx]
  mov [eax + ebx], dl
  inc ebx
  cmp ebx, 3
  jne CopiaCaracterMes

  pop esi
  pop edx
  pop ecx
  pop ebx
  pop eax
  pop ebp
  ret 8
ObtenerNombreMes endp

Main:
  push offset FechaOrigen1
  push offset FechaDestino1
  call FormateaFecha

  push offset FechaOrigen2
  push offset FechaDestino2
  call FormateaFecha

  push 0
  call ExitProcess
end Main
```

7. ¿Qué falta en el hueco -- 1 --?

mov [ebx+ecx], dl

8. ¿Qué falta en el hueco -- 2 --?

mov eax, [ebp+8]

mov ebx, [ebp+12]

9. ¿Qué falta en el hueco -- 3 --?

jne noEsUno

10. Sabiendo que el primer **call FormateaFecha** se codifica como **E86CFFFFFFh** y que el **6Ch** de la anterior codificación se guarda en la dirección de memoria **004010ACh**, ¿qué byte hay en la cima de la pila justo antes de ejecutar el **push ebp** del procedimiento **FormateaFecha** la primera vez? Contestar en hexadecimal.

B0h

11. ¿Cuál debería ser el tamaño de la pila mínimo para que el programa funcionase correctamente? Responder en decimal e incluir las unidades.

68 bytes

12. Sabiendo que el valor inicial de **ebp** es **0012FFF0h** y el de **esp** es **0012FFC4h**, ¿cuánto vale **ebp** la segunda vez que se ejecuta el **push ebp** del procedimiento **ObtenerNombreMes?** Responder en hexadecimal.

0012FFB4h

13. ¿Cuánto vale el registro **ch** antes de ejecutar por primera vez la instrucción **add ecx, 2** del procedimiento **FormateaFecha**? Responder en hexadecimal.

0

14. Indica cuál o cuales de las siguientes afirmaciones son ciertas. Contesta "Ninguna" si crees que ninguna es cierta.

a) Windows y Linux utilizan el mismo tipo de fichero ejecutable.

b) El enlazador enlaza varios ficheros fuente para obtener un único fichero objeto.

c) Las convenciones de llamada de C especifican cómo se pasan los parámetros (en qué orden, en qué registros o en la pila) y quién se encarga de destruirlos.

d) Como es un lenguaje interpretado, cuando se ejecuta un programa de Java en un procesador Pentium no se ejecutan instrucciones del procesador.

c

15. Indica cuál o cuales de las siguientes afirmaciones son ciertas. Contesta "Ninguna" si crees que ninguna es cierta.

a) Cuando una interrupción es aceptada por la CPU teórica, el dispositivo comunica el vector de interrupción a través del bus de direcciones.

b) Si la codificación de una instrucción en Intel es C7 03 34 00 00 00, se almacenará el byte C7 en la posición de memoria más baja.

c) En la arquitectura x86-64 cada posición de memoria almacena 64 bits.

d) En la CPU teórica, la dirección de memoria 0100h se corresponde con un vector de la tabla de vectores de interrupción.

b

16. Se ha diseñado un dispositivo de memoria utilizando un decodificador de tres entradas y varios chips de organización 16Kx4. Se sabe además que el ancho de la palabra de datos del dispositivo es 32. Determinar la organización del dispositivo (M_dxN_d) y el número de chips utilizados en su diseño.

M_dxN_d = 128Kx32 Nº Chips = 64