

Todas las preguntas tienen la misma puntuación. Cada respuesta correcta suma un punto. Cada respuesta incorrecta, ilegible o vacía no suma ni resta. La nota del examen se obtiene multiplicando el número de preguntas correctas por 10 y dividiendo por el número total de preguntas del examen

❑ La CPU elemental se está utilizando en un dispositivo para reconocimientos médicos. El dispositivo dispone de una pantalla similar a la del computador elemental. Además tiene un nuevo periférico: un mando con cuatro flechas y un botón. Cuando se pulsa cualquiera de las flechas o el botón, se genera una interrupción. El interfaz de este periférico sólo tiene un registro de estado que indica entre los bits 0 y 3 si se ha pulsado alguna de las flechas, y en el bit 4 si se ha pulsado el botón. Este registro está mapeado en la dirección 9000h y se ha asignado el vector de interrupción 2 a la interfaz.

En una de las pruebas de reconocimiento, se va pintando, en la línea de abajo de la pantalla, una bola (una "o" mayúscula) desde la izquierda a la derecha. A mitad de camino, la bola desaparece, para volver a aparecer en la última posición de la pantalla después del tiempo necesario para que llegue allí según la velocidad a la que se estaba mostrando. En esta prueba sólo es necesario detectar cuándo se pulsa el botón. El usuario debe pulsarlo justo cuando cree que la bola va a aparecer en el extremo de la pantalla.

El código adjunto muestra parte del programa que muestra la bola y cuenta los aciertos del usuario en la variable numAciertos. Entre el código omitido se encuentran:

- El procedimiento instalarRutina, que prepara una rutina de interrupción para el periférico.
- El procedimiento calcularPosVieja, que recibe como parámetro la posición de la bola y devuelve en R0 en qué posición se pintó la vez anterior.

```

1 ORIGIN 0720h
2 INICIO main
3 .PILA 100h
4
5 .DATOS
6 posBola VALOR 8069h
7 numAciertos VALOR 0
8
9 .CODIGO
10
11 PROCEDIMIENTO rutina
12     PUSH R0
13     PUSH R1
14
15     ; Comprobar que se ha pulsado el botón y no

```

```

16     ; una flecha
17
18
19     MOV R0, [R0]
20
21
22     R1, R1, R0
23     BRZ salirRutina
24
25     ; Posición donde debe estar la bola si
26     ; el usuario acertó
27     MOVL R0, 77h
28     MOVH R0, 80h
29
30     MOVL R1, BYTEBAJO DIRECCION posBola
31     MOVH R1, BYTEALTO DIRECCION posBola
32     MOV R1, [R1]
33     COMP R0, R1
34     BRNZ salirRutina
35
36     ; Hubo acierto
37     MOVL R1, BYTEBAJO DIRECCION numAciertos
38     MOVH R1, BYTEALTO DIRECCION numAciertos
39     MOV R0, [R1]
40     INC R0
41     MOV [R1], R0
42
43 salirRutina:
44     POP R1
45     POP R0
46     IRET
47 FINP
48 ; Código omitido
49 ; ...
50
51 PROCEDIMIENTO calcularNuevaPos
52     PUSH R0
53     PUSH R1
54     PUSH R2
55     MOVL R0, BYTEBAJO DIRECCION posBola
56     MOVH R0, BYTEALTO DIRECCION posBola
57     MOV R1, [R0]
58     INC R1
59     MOVH R2, 80h
60     MOVL R2, 78h
61     COMP R1, R2
62     BRNZ salir
63
64     ; Reinicialiar la posición
65     MOVL R1, 69h
66     MOVH R1, 80h
67
68 salir:
69     ; Actualizar posición
70     MOV [R0], R1
71     POP R0
72     POP R1
73     POP R2
74     RET
75 FINP
76
77 PROCEDIMIENTO pintarBola
78     PUSH R0
79     PUSH R1

```

```

80     PUSH R2
81     MOVL R0, BYTEBAJO DIRECCION posBola
82     MOVH R0, BYTEALTO DIRECCION posBola
83     MOV R1, [R0]
84
85     ; Borrar la posición vieja
86     PUSH R1
87     CALL calcularPosVieja
88     INC R7
89     ; Pintar en negro sobre negro en
90     ; la posición vieja
91     MOVL R2, ' '
92
93
94
95     ; Mirar si es una posición que se deba
96     ; ocultar
97     MOVL R2, 72h ; R2 = primera pos a ocultar
98     MOVH R2, 80h
99     COMP R1, R2
100    BRC pintar
101
102    MOVL R2, 77h ; R2 = última pos a ocultar
103    COMP R1, R2
104    BRC noPintar
105
106    ; Escribir la posición nueva
107    pintar:
108        MOVL R0, 'O'
109        MOVH R0, 7h ; Blanco sobre negro
110        MOV [R1], R0
111
112    noPintar:
113        POP R0
114        POP R1
115        POP R2
116        RET
117    FINP
118
119    main:
120        CALL instalarRutina
121
122    buclePintado:
123        CALL calcularNuevaPos
124        CALL pintarBola
125        JMP buclePintado
126    FIN

```

— ¿Qué instrucción o instrucciones faltan en el hueco de las líneas 17 y 18?

```

MOVL R0, 00h
MOVH R0, 90h

```

— ¿Qué instrucción o instrucciones faltan en el hueco de las líneas 20 y 21?

```
MOVL R1, 10h
MOVH R1, 0
```

— ¿Qué mnemónico falta en el hueco de la línea 22?

```
AND
```

— ¿Qué instrucción o instrucciones faltan en el hueco de las líneas 92 y 93?

```
MOVH R2, 0
MOV [R0], R2
```

— ¿Escribir el código del procedimiento instalarRutina? Nota: No salvar guardar registros.

```
MOVL R0, 2
MOVH R0, 0
MOVL R1, BYTEBAJO DIRECCION rutina
MOVH R1, BYTEALTO DIRECCION rutina
MOV [R0], R1
STI
RET
```

— Sabiendo que R7 comienza en 088Eh, ¿cuál es el valor más pequeño que puede alcanzar dentro del procedimiento calcularNuevaPos? Responder en hexadecimal.

```
886h
```

— En el penúltimo paso de ejecución de la instrucción POP R0, que está situada en la posición de memoria 75Bh, el registro de estado vale Bh. Si se produce justo en ese momento una interrupción, ¿qué valores aparecerán en el bus de datos durante la fase de aceptación de la interrupción? Nota: El orden no se evaluará.

```
000Bh (SR), 075Ch (PC), 0002h (vector int.), 0722h (dir. rutina)
```

— ¿Cuál será la función del circuito de activación de la interfaz del periférico de entrada? Ejemplo de respuesta:

$$a_3\bar{a}_5 \dots \bar{a}_9 a_{28}$$

```
a15a14a13a12a11...a0
```

2P2007-08-progC-C ☐ Se ha desarrollado el programa en C para una arquitectura x86-32 que se muestra en el siguiente listado:

```
1 int suma(int a, int b, int c)
2 {
3     int temp=0;
4
5     temp= a + b + c;
6     return(temp);
7 }
8
9 int main(int argc, char * argv)
10 {
11     int result;
12
13     result= suma(7, 9, 23);
14 }
```

— Dentro del procedimiento suma ¿cuál es la sentencia en ensamblador que generará el compilador de C para crear el espacio necesario para la variable temp? Nota: Los enteros son de 32 bits de tamaño.

```
sub esp, 4
```

— Si el compilador ha utilizado la convención de llamadas fastcall al generar el código del procedimiento ¿Cuál será la última instrucción de la CPU que se ejecuta en el procedimiento suma?

```
ret 4
```

— Escribe las instrucciones que generará el compilador de C para realizar el paso de parámetros al procedimiento suma si la convención de llamadas es cdecl (la estándar de C).

```
PUSH 23
PUSH 9
PUSH 7
```

2P2007-08-chicle-2 ☐ Indica cuál o cuáles de las siguientes afirmaciones son ciertas. Puedes contestar *todas* o *ninguna* en el caso de que todas o ninguna de ellas sean ciertas.

- Definimos el mapa de memoria de un computador como la especificación de las posiciones de memoria que ocupan la RAM y la ROM instaladas en el sistema.
- Cuando Intel pasó de la arquitectura de 16 bits a la de 32 bits, una dirección de memoria pasó de acceder a un byte a acceder a una palabra de 16 bits.
- La jerarquía de memoria se organiza para que la CPU vea un sistema de memoria de más capacidad del que realmente existe.
- El incremento de los registros de propósito general que hay en la arquitectura de 64 bits de AMD mejora las prestaciones de los programas porque reducen el número de accesos a memoria.

```
4
```

final2007-08-memorias ☐ Se tienen dos dispositivos de memoria, A y B, para la CPU teórica, ambos de 1Kx16. El dispositivo A está mapeado cubriendo la zona más baja del espacio de direcciones de la CPU y el dispositivo B está a continuación. El dispositivo A se construye a partir de chips de memoria de organización 512x8 y el dispositivo B a partir de chips de 256x2.

— Si R0 vale 0404h, ¿cuántos chips de memoria se activan en el quinto paso de ejecución de la instrucción MOV R1, [R0]?

```
8
```

— ¿Cuál será el tipo del decodificador con más líneas de entrada de los dos dispositivos? Ejemplo de respuesta: 32:64.

```
2:4
```

final2007-08-rango-banco ☐ Se han utilizado ocho chips de memoria de 256x4 para diseñar un dispositivo de memoria. Este dispositivo se ha conectado a una CPU con un bus de direcciones de 10 líneas. El dispositivo cubre el rango de direcciones 000h-1FFh. ¿De cuántas líneas es el bus de datos de la CPU?

```
16
```

□ Se ha construido un programa que permite saber qué día de la semana es cualquier fecha del siglo XXI. El algoritmo se limita a calcular los días que han pasado desde el 1-1-2001 (lunes) hasta la fecha a calcular y luego, al dividir ese número por 7, el resto obtenido nos dice el día de la semana que es (0=lunes, 1=martes, 2=miércoles...). Para calcular los días pasados se sigue el siguiente procedimiento:

- Para una determinada fecha, p. e. 12-9-2008, se calculan primero los días pasados en años completos. En el ejemplo, han pasado 7 años completos (2001 a 2007) a 365 días por año más un día extra pues ha habido un año bisiesto (2004).
- A continuación se calculan los días que han pasado en el año de la fecha (2008). Para ello se suman los días de los meses pasados (enero a agosto, en el ejemplo). El número de días que tiene cada mes se obtiene de una tabla. Si el año es bisiesto y la fecha es posterior a febrero, hay que sumar un día extra. Finalmente se añaden los días del mes de la fecha (12 en el ejemplo).

El programa está construido en base a una serie de procedimientos que se detallan:

EsBisiesto Es un procedimiento que recibe como parámetro el año y nos devuelve en EAX un 1 si es bisiesto y un 0 si no lo es. No se muestra el código.

DiasAniosPasados Este procedimiento calcula los días correspondientes a los años completos desde 1-1-2001 hasta la fecha deseada. El procedimiento recibe como parámetro el año de la fecha a calcular. Por cada año completo desde 2001 hasta el pasado como parámetro suma 365 días. Cada cuatro años, suma un día más. Retorna el resultado en EAX.

DiasAnioActual Calcula los días pasados en el año de la fecha. Recibe como parámetros, y en este orden, el año, la dirección de memoria dónde se encuentra una lista con la duración en días de cada mes del año (posición 0= enero, posición 1= febrero, ...), el mes y el día. Retorna en EAX el número de días.

Nota: El programa presenta algunos huecos sobre los que no se hacen preguntas.

```
1 .Data
2 Dia DD 5
3 Mes DD 4
4 Anio DD 2005
5 Meses DD 31, 28, 31, 30, 31, 30
```

```
6 DD 31, 31, 30, 31, 30, 31
7
8 .Code
9
10 DiasAniosPasados PROC
11
12
13 ;Salvaguarda de registros
14 PUSH ECX
15 PUSH EBX
16 ;Acceso a los parámetros
17
18 ;Cálculo años completos
19 SUB ECX, 2001
20 JECXZ salir ;No hay años completos
21 XOR EAX, EAX
22 XOR EBX, EBX
23 suma:
24 ADD EAX, 365 ;Suma 365 días por año
25 INC EBX ;Incrementa controlador de bisiestos
26
27
28 INC EAX ;Cada 4 años hay uno bisiesto
29 XOR EBX, EBX
30 nobisi:
31 LOOP suma
32 salir:
33 POP EBX
34 POP ECX
35 POP EBP
36
37 DiasAniosPasados ENDP
38
39 DiasAnioActual PROC
40
41
42 ;Salvaguarda de los registros
43 PUSH EBX
44 PUSH ECX
45 PUSH ESI
46 PUSH EDI
47 PUSH EDX
48 ;Acceso a los parámetros
49 MOV EBX, [EBP + 8];Desap. día de la fecha
50 MOV ECX, [EBP + 12];Desap. mes de la fecha
51 MOV ESI, [EBP + 16];Desap. lista de dias/mes
52 MOV EDX, [EBP + 20];Desap. año de la fecha
53
54 XOR EAX, EAX
55 ;Calculamos los meses completados (mes fecha-1)
56 DEC ECX
57 JECXZ dias ; Ningún mes completado
58 XOR EDI, EDI ;Inicializamos índice de mes
59 sumdi:
60 ;Suma, mes a mes, los días de los completados
61 ADD EAX, [ESI + ]
62 INC EDI ; Pasamos al siguiente mes
63 LOOP sumdi
64 ;Si mes > febrero
65 CMP ECX, 1
66 JBE dias
67 ;Salvaguardamos temporalmente EAX
68 MOV EDI, EAX
69 ; Comprobamos si el año es bisiesto
```

```
70 PUSH EDX
71 CALL EsBisiesto
72 ADD EAX, EDI ;y lo añadimos al acumulador
73 ;Sumamos finalmente los días de la fecha
74 dias:
75 ADD EAX, EBX
76
77 POP EDX
78 POP EDI
79 POP ESI
80 POP ECX
81 POP EBX
82 POP EBP
83
84 DiasAnioActual ENDP
85
86 Main:
87 PUSH [Anio] ; Se apila año de la fecha
88 CALL DiasAniosPasados
89
90 MOV EBX, EAX ;Guardamos temporalmente EAX
91
92 PUSH [Anio] ;Apila el año de la fecha
93
94 ;Direc. lista días/mes
95 ;Apila mes de la fecha
96 PUSH [Dia] ;Apila día de la fecha
97 CALL DiasAnioActual
98 ADD EAX, EBX ; Actualizamos contador días
99
100
101
102
103 END Main
```

— ¿Qué falta en los huecos de las líneas 11 y 12?

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

— Completar el hueco de la línea 17.

```
MOV ECX, [EBP + 8]
```

— ¿Qué debería haber en los huecos de las líneas 26 y 27 para que el programa funcione correctamente?

```
CMP EBX, 4
JNE nobisi
```

— ¿Qué instrucción falta en el hueco de la línea 83?

```
RET 16
```

— Completar las líneas del programa 93 y 94.

```
PUSH OFFSET Meses  
PUSH [Mes]
```

— ¿Qué falta en el hueco de la instrucción de la línea 61?

```
EDI*4
```

— ¿Cuántos bytes ocupa la pila en el momento de máxima ocupación? Contesta en decimal.

```
52
```

- ☐ La puerta AND de activación de la interfaz de un periférico conectado a la CPU teórica tiene tres entradas. Si se reserva para mapeo de interfaces de este tipo la mitad del espacio de direcciones, ¿cuántas interfaces podrían estar mapeadas?

```
4
```

- ☐ Indica cuál o cuáles de las siguientes afirmaciones son ciertas. Puedes contestar *todas* o *ninguna* en el caso de que todas o ninguna de ellas sean ciertas.

1. La arquitectura de 64 bits diseñada por Intel (IA-64) es compatible con el juego de instrucciones del 80386.
2. El encadenamiento de la línea `INTA` permite a los periféricos, cuando la activan, interrumpir a la CPU de manera ordenada para reclamar operaciones de E/S.
3. Un programa escrito en ensamblador del 8086 se puede ejecutar sin cambios en un procesador Intel Core Duo.
4. En una jerarquía de memoria de 4 niveles tendríamos, desde la CPU, la siguiente cadena de tecnologías de memoria: SRAM -> SRAM -> DRAM -> Disco duro.

```
3 y 4
```