

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

❑ El programa en ensamblador de la CPU teórica que se muestra a continuación procesa matrices de números naturales. A través del procedimiento `CalcularMSFMatriz`, se calcula la suma de cada uno de los elementos de una fila. De este conjunto de sumas, calcula su valor máximo. Una matriz de dimensiones $M \times N$ (filas \times columnas) se organiza en memoria como $M \times N$ posiciones consecutivas; primero los N elementos de la primera fila, luego los N de la segunda, etc.

El procedimiento `CalcularMSFMatriz` recibe por la pila tres parámetros, en el siguiente orden:

- Número de columnas.
- Número de filas.
- Dirección de memoria de comienzo de la matriz.

El procedimiento devuelve el máximo buscado en el registro `R0`.

```

1  ORIGIN 2000h
2  INICIO ini
3  .PILA 10h
4  .DATOS
5  Matriz VALOR 1,2,3,4,0,1,2,3
6  MSFMatriz VALOR 0
7  .CODIGO
8  PROCEDIMIENTO CalcularMSFMatriz
9    PUSH R6
10   MOV R6,R7
11   PUSH R1
12   PUSH R2
13   PUSH R3
14   PUSH R4
15   PUSH R5
16 ; Inicializar máximo
17 XOR R0,R0,R0
18 ; Acceso a parámetros - Dirección de la Matriz
19
20
21
22 ; Acceso a parámetros - Número de Filas
23 INC R6
24 MOV R2,[R6]
25 INC R6
26 BucleFilas:
27 ; Acceso a parámetros - Número de Columnas
28 MOV R3,[R6]
29 XOR R4,R4,R4 ; Acumula cada Fila
30 BucleColumnas:
31 MOV R5,[R1]
```

```

32 ADD R4,R4,R5
33 INC R1 ; Siguiente posición de memoria Matriz
34 DEC R3 ; Decrementar número de columnas
35 BRNZ BucleColumnas
36 COMP R4,R0
37 BRC NoHayNuevoMaximo
38 BRZ NoHayNuevoMaximo
39 MOV R0,R4
40 NoHayNuevoMaximo:
41 DEC R2
42 BRNZ BucleFilas
43 POP R5
44 POP R4
45 POP R3
46 POP R2
47 POP R1
48 POP R6
49 RET
50 FINP
51
52 ini:
53 MOVL R0,04h
54 PUSH R0
55 MOVL R0,02h
56 PUSH R0
57 MOVL R0,BYTEBAJO DIRECCION Matriz
58 MOVH R0,BYTEALTO DIRECCION Matriz
59 PUSH R0
60 CALL CalcularMSFMatriz
61 INC R7
62 INC R7
63 INC R7
64 ; Guardamos el resultado en MSFMatriz
65
66
67
68
69 FIN
```

— ¿Qué instrucción o instrucciones faltan en el hueco de las líneas 65 a 67?

```

MOVL R1, BYTEBAJO DIRECCION MSFMatriz
MOVH R1, BYTEALTO DIRECCION MSFMatriz
MOV [R1], R0
```

— ¿Qué instrucción o instrucciones faltan en el hueco de las líneas 19 a 21?

```

INC R6
INC R6
MOV R1,[R6]
```

— En la llamada al procedimiento y justo después de ejecutar la instrucción `RET`, el valor de `R7` es `2047h`. ¿Cuál es el valor inicial de `R7`? Expresar el resultado en **hexadecimal**.

204Ah

— Durante la ejecución de una instrucción del programa, en el paso 3 el valor que contiene `IB` es `3100h`. ¿De qué instrucción se trata? Contestar con el mnemónico.

PUSH R1

— La CPU dónde se ejecuta el programa anterior tiene una frecuencia de trabajo de 2 Mhz. ¿Cuánto tiempo tarda en ejecutarse completamente el bucle `BucleColumnas`? Incluye las unidades en la respuesta.

55 us

❑ Se ha diseñado una representación de números enteros con 8 bits en exceso a Z con la particularidad de que el 75 % de los códigos se asignan a los números negativos y el 25 % a los números positivos (incluido el cero). ¿Cómo se representaría el número cero en este formato? **Dar el resultado en hexadecimal**.

C0h

❑ A un sumador de 8 bits se le introduce en una de sus entradas el código ASCII de la letra E. En su salida aparece el código ASCII de la letra A ¿Cuál es el valor que tiene en su otra entrada? **Dar el resultado en hexadecimal**.

FC

❑ 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

❑ 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 pasado 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 EBX, ;Control de bisiestos
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 ;Direc. lista días/mes
94 ;Apila mes de la fecha
95 PUSH [Dia] ;Apila día de la fecha
96 CALL DiasAnioActual
97 ADD EAX, EBX ; Actualizamos contador días
98
99
100 ;El programa continua
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
```

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

EDI*4

- ❑ Se ha diseñado un formato de coma flotante, similar al IEEE-754 pero con sólo 12 bits. De ellos, 5 se utilizan para el exponente codificado en exceso a Z central y los restantes para la mantisa en signo-magnitud normalizada como en IEEE-754. La figura siguiente muestra cómo se organizan estos 12 bits.

S	Exponente	Mantisa
---	-----------	---------

En el campo de exponente, las cantidades permitidas van desde 00001 hasta 11110, puesto que los casos 00000 y 11111 se reservan para codificar cantidades especiales como el cero o el infinito. Representa en este formato la cantidad 8.75 y expresa el resultado en **hexadecimal**.

4C6

- ❑ Se dispone de un sumador que opera con cantidades de 6 bits. Por una de las entradas se introduce el número negativo más grande, en valor absoluto, que se puede representar en complemento a 2. En la otra entrada se introduce el número positivo más grande que se puede representar en signo-magnitud.

- ¿Cuál será el resultado de la suma? Expresarlo en decimal, interpretado en complemento a 2.
- ¿Cuál es el valor de los flags de estado?

Resultado:-1
Z=0 C=0 O=0 S=1

- ❑ 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 principal característica de los Sistemas Digitales Secuenciales es que son síncronos.
2. El carácter U+05D1 tiene una codificación en UTF-8 de D7 91 h.
3. Las instrucciones lógicas sólo modifican el bit ZF del registro de estado.
4. Un procedimiento sólo puede tener una instrucción RET.

2

final2007-08-prog-entrada-salidaFI-A

- ❑ 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  .DATOS
5  posBola VALOR 8069h
6  numAciertos VALOR 0
7
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   [REDACTED]
18   [REDACTED]
19   MOV R0, [R0]
20   [REDACTED]
21   [REDACTED]
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
34   COMP R0, R1
35   BRNZ salirRutina
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
49   ; Código omitido
50   ; ...
51
52   PROCEDIMIENTO calcularNuevaPos
53   PUSH R0
54   PUSH R1
55   PUSH R2
56
57   MOVL R0, BYTEBAJO DIRECCION posBola
58   MOVH R0, BYTEALTO DIRECCION posBola
59   MOV R1, [R0]
60   INC R1
61   MOVH R2, 80h
62   MOVL R2, 78h
63   COMP R1, R2
64   BRNZ salir
65
66   ; Reinicialiar la posición
67   MOVL R1, 69h
68   MOVH R1, 80h
69
70   salir:
71   ; Actualizar posición
72   MOV [R0], R1
73
74   POP R0
75   POP R1
76   POP R2
77   RET
78   FINP
79
80   PROCEDIMIENTO pintarBola
81   PUSH R0
82   PUSH R1
83   PUSH R2
84
85   MOVL R0, BYTEBAJO DIRECCION posBola
86   MOVH R0, BYTEALTO DIRECCION posBola
87   MOV R1, [R0]
88

```

```

89 ; Borrar la posición vieja
90 PUSH R1
91 CALL calcularPosVieja
92 INC R7
93 ; Pintar en negro sobre negro en
94 ; la posición vieja
95 MOVL R2, ' '
96
97
98
99 ; Mirar si es una posición que se deba
100 ; ocultar
101 MOVL R2, 72h ; R2 = primera pos a ocultar
102 MOVH R2, 80h
103 COMP R1, R2
104 BRC pintar
105
106 MOVL R2, 77h ; R2 = última pos a ocultar
107 COMP R1, R2
108 BRC noPintar
109
110 ; Escribir la posición nueva
111 pintar:
112 MOVL R0, 'O'
113 MOVH R0, 7h ; Blanco sobre negro
114 MOV [R1], R0
115
116 noPintar:
117 POP R0
118 POP R1
119 POP R2
120 RET
121 FINP
122
123 main:
124 CALL instalarRutina
125
126 buclePintado:
127 CALL calcularNuevaPos
128 CALL pintarBola
129 JMP buclePintado
130
131 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 96 y 97?

```

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

```

— 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)

❑ Para simplificar la programación de algunos problemas se define una nueva instrucción en la CPU teórica que permite almacenar en la pila un parámetro por referencia:
PUSH [Rs]
Completar la siguiente tabla con las señales de control necesarias en cada paso para implementar dicha instrucción.

Paso	Señales
4	Rs-IB, IB-MAR, Leer
5	R7-IB, TMPE-SET, Suma, ALU-TMPS
6	TMPS-IB, IB-R7, IB-MAR, ESCRIBIR
7	FIN

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

1PFinalJunio2008-ALU final2007-08-rango-banco

❑ Una ALU realiza la operación de SUMAR. El primer operando, en binario natural, es el 3. El segundo, en complemento a 2 es el 6. Tras realizar la operación, se activa el bit de Overflow. ¿De cuantos bits es la ALU?

4

❑ 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