

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

- 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. El enlazador es un programa que combina distintos ficheros con código máquina en un programa ejecutable.
2. Utilizar instrucciones de tamaño variable sólo tiene ventajas frente a utilizar instrucciones de tamaño fijo.
3. La arquitectura x86-64 permite direccionar exactamente el doble de posiciones que la arquitectura x86-32.
4. La convención de llamadas *fastcall* utiliza sólo la pila porque es más rápido que utilizar también algunos registros.

1

- Se tiene un dispositivo que cubre todo el rango de direcciones de la CPU teórica. Se sabe que está formado por chips de memoria con organización 16Kx1. ¿Qué tipo de decodificador está utilizando (ejemplo de respuesta: 8:256) y cuántos chips contiene el dispositivo?

Decodificador = 2:4 Número de chips = 64

- El MBR (*Master Boot Record*) es una forma de organizar el primer sector de un disco duro. Entre otras cosas, incluye una tabla que sirve para dividir el disco duro en distintas particiones. Esta tabla tiene cuatro entradas, una para cada posible partición. Cada entrada ocupa 16 bytes y contiene diversa información relativa a la partición, entre la que se encuentra la siguiente:

- En el byte 0 se guarda el indicador de arranque. Si vale 80h la partición se usa para arrancar; en otro caso debe valer 00h.
- En los bytes 6 y 7 se guarda el sector y cilindro donde acaba la partición (los sectores y los cilindros son divisiones de los discos). Dentro de estos dos bytes, los 6 bits de más peso indican el sector final y los otros 10, el cilindro final.
- En los bytes 12 a 15 se guarda el número de sectores que contiene la partición.

Se están desarrollando algunas funciones para modificar el MBR y se está utilizando un programa de prueba que tiene una tabla de particiones de ejemplo en la variable `part`. Las funciones desarrolladas son:

- CambiarActiva:** recibe por la pila tres parámetros: la dirección de comienzo de la tabla de particiones, el número de partición a cambiar (entre 0 y 3) y un 1 si se desea activar la partición o un 0 si se desea desactivarla. Para activar, el procedimiento escribe un 80h en el byte 0 de la entrada correspondiente a la partición y para desactivar escribe en esa posición un 0.
- CambiarCilindroFinal:** recibe por la pila tres parámetros: la dirección de comienzo de la tabla de particiones, el número de partición a cambiar (entre 0 y 3) y el nuevo valor para el número de cilindro final de esa partición. Como se explica en la tabla anterior, ese número se almacena dentro de los bytes 6 y 7 correspondientes a esa partición, en concreto dentro de los 10 bits de menos peso.

A continuación se presenta el listado correspondiente a este programa con algunos huecos.

```

1  .386
2  .model flat, stdcall
3  ExitProcess proto, ExitCode:dword
4
5  .data
6  part DB 80h, 1h, 01h, 0h, 7h, 0FEh, 0FFh, 6Dh
7       DB 3Fh, 0h, 0h, 0h, 0AFh, 39h, 0D7h, 0h
8       DB 0h, 0h, 0C1h, 6Eh, 0Ch, 0FEh, 0FFh, 0FFh
9       DB 0EEh, 39h, 0D7h, 0h, 0BDh, 86h, 0BBh, 0h
10      DB 0h, 0h, 0h, 0h, 0h, 0h, 0h, 0h
11      DB 0h, 0h, 0h, 0h, 40h, 1h, 0h, 0h
12      DB 0h, 0h, 0h, 0h, 0h, 0h, 0h, 0h
13      DB 0h, 0h, 0h, 0h, 0h, 0h, 0h, 0h
14
15  .code
16  CambiarActiva proc
17      push ebp
18      mov ebp, esp
19      push eax
20      push ebx
21      push ecx
22
23      mov eax, [ebp + 8] ; activar o desactivar
24      mov ebx, [ebp + 12] ; índice de partición
25      mov ecx, [ebp + 16] ; dir. base de tabla
26
27      ; Activar o desactivar en función de eax
28      cmp eax, 0
29      [ ]
30
31      desactivar:
32      mov al, 0
33  
```

```

34
35  escribir:
36      ; Multiplicar ebx por 16
37      shl ebx, [ ]
38      mov [ecx + ebx], al
39
40      pop ecx
41      pop ebx
42      pop eax
43      pop ebp
44      [ ]
45  CambiarActiva endp
46
47  CambiarCilindroFinal proc
48      push ebp
49      mov ebp, esp
50      push eax
51      push ebx
52      push ecx
53      push edx
54
55      mov eax, [ebp + 8] ; nuevo cilindro
56      mov ebx, [ebp + 12] ; núm. partición
57      mov ecx, [ebp + 16] ; dir. base de tabla
58
59      ; Multiplicar ebx por 16
60      shl ebx, [ ]
61
62      ; Escribir nuevo cilindro sin perder los bits
63      ; relativos al sector
64      mov dx, [ecx + ebx + 6]
65      and dx, 0FC00h
66      [ ]
67      mov [ecx + ebx + 6], dx
68
69      pop edx
70      pop ecx
71      pop ebx
72      pop eax
73      pop ebp
74      [ ]
75  CambiarCilindroFinal endp
76
77  Inicio:
78      push offset part
79      push 0 ; partición 0
80      push 0 ; desactivar
81      call CambiarActiva
82
83      push offset part
84      push 1 ; partición 1
85      push 1 ; activar
86      call CambiarActiva
87
88      push offset part
89      push 1 ; partición 1
90      push 1ABh ; nuevo cilindro
91      call CambiarCilindroFinal
92
93      push 0
94      call ExitProcess
95  end Inicio

```

— ¿Qué falta en el hueco de las líneas 44 y 74? (Es la misma instrucción.)

```
ret 12
```

— ¿Qué falta en el hueco de las líneas 37 y 60? (Es el mismo valor.)

4

— ¿Qué instrucciones faltan en el hueco que comienza en la línea 29?

```
jz desactivar
mov al, 80h
jmp escribir
```

— ¿Qué instrucción falta en el hueco de la línea 66?

or dx, ax

— ¿Cuántos sectores contiene la partición 2 (tercera partición) de la tabla `part`? Ten en cuenta que este número se almacena en memoria como un dato de tamaño doble palabra. **Contestar en decimal.**

320

— Se sabe que la instrucción de la línea 91 está situada a partir de la dirección de memoria 401098h y durante se ejecución modifica las posiciones de memoria de la 12FFB4h a la 12FFB7h (ambas incluidas). ¿Cuánto vale ebp en la línea 55? **Contestar en hexadecimal.**

12FFB0h

❑ Se tiene una ALU de 5 bits similar a la ALU para la CPU teórica. Por la entrada A se introduce el número -12 codificado en exceso 16 y por la entrada B se introduce el número más grande que se puede codificar en complemento a 2. Si se realiza la operación suma, ¿cuál es el resultado calculado y cuánto valen los bits de carry y overflow? **Interpretar el resultado en complemento a 2 y dar la solución en decimal.**

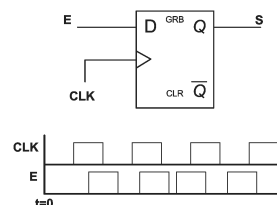
Resultado= -13 C= 0 O= 1

Julio2010-caracter&Julio2010-circuito-combinacional

❑ ¿Cuántos caracteres hay en la secuencia de bits expresada en hexadecimal 31 E2 82 AC C3 B1 teniendo en cuenta que está codificada en UTF-8?

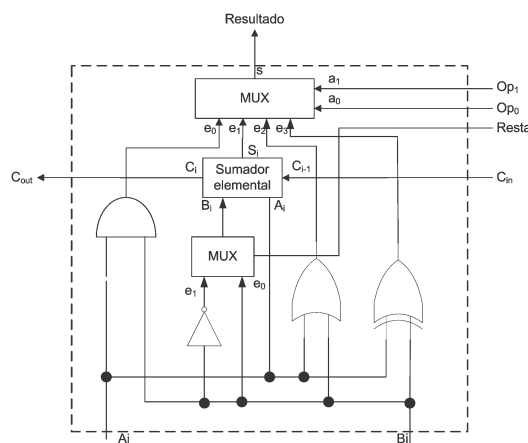
3

❑ En el circuito de la figura, en el instante 0 el biestable almacena el valor 0. Si la frecuencia de la señal CLK es de 5 Hz, ¿cuántos segundos tarda la salida del biestable en valer 1?



t=0,5 seg

❑ Se ha construido una ALU de 16 bits utilizando ALUs de 1 bit modificadas tal y como se muestra en la figura. Si A y B son dos números de 16 bits que se quieren restar en esa ALU, ¿cuáles deben de ser los valores de todas las líneas de control de esta ALU de 16 bits para poder realizar dicha operación?



Op0=1 Op1=0 Resta=1 (Carry_in=1 ó 0)

Julio2010-chicle2

❑ 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. En un formato de coma fija, al aumentar el número de bits para la parte entera se aumenta la precisión disponible.
2. En una unidad de control cableada la memoria de microprograma almacena las señales de control que se deben generar para cada instrucción.
3. El uso de memoria caché amplía la cantidad de memoria que se puede direccionar.
4. La memoria SRAM es más rápida que la DRAM, por lo que se utiliza para la memoria caché.

4

Julio2010-UC-Seniales

❑ Se quiere definir una nueva instrucción aritmética en la CPU teórica que sume un dato inmediato de 8 bits a un registro:
ADD Rx, Inm_8
El dato inmediato, que puede ser un número entero con signo, irá codificado en el byte de menor peso de la instrucción. Indicar la secuencia de señales de control necesarias para su ejecución.

Paso	Señales
1	PC-IB, IB-MAR, Leer, TMPE-CLR, Carry-in, ADD, ALU-TMPS
2	TMPS-IB, IB-PC
3	MDR-IB, IB-IR
4	Rx-IB, IB-TMPE
5	JUMP, ADD, ALU-SR, ALU-TMPS
6	TMPS-IB, IB-Rx, FIN
7	

Julio2010-numerosenteros

- A continuación se muestra el código de un programa que realiza la división entera entre los números naturales de dos listas dadas. El programa va dividiendo los números de la lista de dividendos entre los correspondientes de la lista de divisores. El cociente obtenido lo va almacenando en la lista de cocientes y el resto de la división lo guarda en la lista de restos.

En la posición n de la lista de cocientes se encuentra el cociente entero de dividir la posición n de la lista de dividendos entre la posición n de la lista de divisores. En la posición n de la lista de restos se encontraría el resto de esa misma división.

Para realizar su trabajo, el programa utiliza el procedimiento divide. Este procedimiento recibe 4 parámetros:

- El valor del dividendo.
- El valor del divisor.
- La dirección en la que hay que dejar el cociente.
- La dirección en la que se guardará el resto.

Para implementar el algoritmo, al dividendo se le va restando el divisor mientras sea más pequeño. Cada vez que se puede restar el divisor se incrementa en uno el cociente. Cuando ya no se le pueda restar al dividendo el divisor, lo que nos quede será el resto de la división entera.

Sabemos que, al empezar la ejecución del programa, los registros de la CPU tenían los siguientes valores:

R0=0000	R3=0000	R6=0000
R1=0000	R4=0000	R7=036E
R2=0000	R5=0000	PC=0339

```

1  ORIGIN 300h
2  INICIO empieza
3  .PILA 20
4  .DATOS
5  Nelem VALOR 5
6  Dividen VALOR 10, 28, 36, 43, 55
7  Divisor VALOR 10, 11, 12, 13, 14
8  Cocient VALOR 5 VECES 0
9  Resto VALOR 5 VECES 0
10 .CODIGO
11 PROCEDIMIENTO Divide
12
13
14 PUSH R0
15 PUSH R1
16 PUSH R2
17 PUSH R3
18 PUSH R4
19 PUSH R5
20 ;Se desapila la dirección del resto
21
```

```

22
23
24 ;Se desapila la dirección del cociente
25 INC R6
26 MOV R4, [R6]
27 ;Se desapila el divisor
28 INC R6
29 MOV R1, [R6]
30 ;Se desapila el dividendo
31 INC R6
32 MOV R2, [R6]
33
34 XOR R3, R3, R3
35 XOR R0, R0, R0
36 ;Si el divisor es cero salimos
37 COMP R0, R1
38 BRZ Salir
39
40 repite:
41 ;Se compara dividendo con divisor
42 COMP R2, R1
43 ;Si es menor, salimos
44 BRC Salir
45 ;Si es mayor, restamos
46 SUB R2, R2, R1
47 ;Incrementamos el cociente
48 INC R3
49 JMP repite
50 Salir:
51 ;Guardamos los valores del cociente y del resto
52 MOV [R4], R3
53 MOV [R5], R2
54 POP R5
55 POP R4
56 POP R3
57 POP R2
58 POP R1
59 POP R0
60 POP R6
61 RET
62 FINP
63
64 empieza:
65 ;Se accede a las variables del programa
66 MOVL R0, BYTEBAJO DIRECCION Nelem
67 MOVH R0, BYTEALTO DIRECCION Nelem
68 MOV R1, [R0]
69 MOVL R2, BYTEBAJO DIRECCION Dividen
70 MOVH R2, BYTEALTO DIRECCION Dividen
71 MOVL R3, BYTEBAJO DIRECCION Divisor
72 MOVH R3, BYTEALTO DIRECCION Divisor
73 MOVL R4, BYTEBAJO DIRECCION Cocient
74 MOVH R4, BYTEALTO DIRECCION Cocient
75 MOVL R5, BYTEBAJO DIRECCION Resto
76 MOVH R5, BYTEALTO DIRECCION Resto
77 ;Si no hay elementos en las listas, acabamos
78 XOR R6, R6, R6
79 COMP R6, R1
80 BRZ Sacabo
81 bucle:
82
83
84
85
```

```

86 ;Se apila la dirección del cociente
87 PUSH R4
88 ;Se apila la dirección del resto
89 PUSH R5
90 CALL Divide
91 ;Se limpia la pila
92
93
94
95
96 ;Se pasa al siguiente elemento de cada lista
97 INC R2
98 INC R3
99 INC R4
100 INC R5
101 ;Se decrementa el número de elementos
102 DEC R1
103 ;Si nos quedan elementos, seguimos
104 COMP R1, R6
105 BRNZ bucle
106
107 Sacabo:
108 JMP -1
109 FIN
```

— ¿Qué instrucción/es faltan en el hueco de la línea 12?

```
PUSH R6
MOV R6, R7
```

— ¿Qué instrucción/es faltan en el hueco de la línea 21?

```
INC R6
INC R6
MOV R5, [R6]
```

— ¿Cuál es la primera posición de la memoria a la que accede el programa y qué valor hay en ella? Constar en **hexadecimal**.

Dirección: 0339 Contenido: 2000

— ¿Cuáles son los valores máximo y mínimo que alcanza el registro R7 durante la ejecución del programa? Constar en **hexadecimal**.

Máximo: 036E Mínimo: 0362

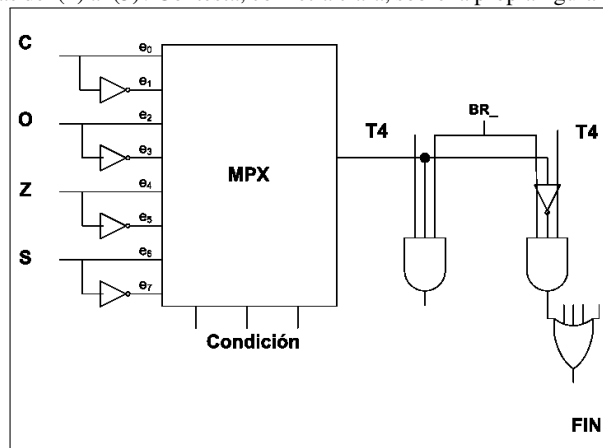
— ¿Qué instrucción/es falta/n en los huecos de las líneas 82 a la 85? Si es necesario, usar R0 como registro auxiliar.

```
MOV R0, [R2]
PUSH R0
MOV R0, [R3]
PUSH R0
```

- Sustituye las 4 líneas de código que van de las líneas 92 a 95 por otras tres que hagan exactamente lo mismo.

```
MOVL R0, 4
MOVL R0, 0
ADD R7, R7, R0
```

□ Se ha implementado una Unidad de Control cableada para la CPU teórica. En la figura se muestra cómo se generan algunas señales en esa unidad de control. Si por las entradas del multiplexador marcadas como `condición` se introducen los 3 bits que codifican la condición en una instrucción de salto condicional (en el mismo orden que aparecen en la instrucción), ¿qué se debería de poner en las entradas marcadas del (1) al (5)? Contesta, con letra clara, sobre la propia figura



- ❑ Para controlar una máquina de café se utiliza una CPU basada en la CPU teórica a la que se le conecta una interfaz específica. La interfaz consta de dos registros, uno de datos y otro de control, que se mapean en memoria en ese orden y a partir de la dirección FF00h. En el registro de datos el usuario puede especificar cómo quiere su café/infusión y en el registro de control la máquina informa de posibles incidencias. Como ejemplo, el bit 0 se usa para indicar que la máquina no tiene agua y el bit 4 se usa para indicar que el café/infusión está listo. La interfaz también es capaz de generar interrupciones. Teniendo esto en cuenta contestar:

- Si en el registro R0 tenemos cargado el valor FF00h y en R1 el valor 0, escribe el trozo de código que mediante muestreo periódico nos permita saber si nuestro café está listo.

```
INC R0
MOVL R1, 10h
lee: MOV R2, [R0]
AND R2, R2, R1
BRZ lee (-3)
```

— Si el programa de control dispone de una rutina denominada `CntrlCafetera`, ¿qué instrucciones harán que se ejecute cuándo llegue una interrupción identificada por el número 20? Supón que los registros `R0` y `R1` están inicializados con los mismos valores que en la pregunta anterior.

```
MOVL R1, 20
MOVL Rx, BYTEBAJO DIRECCION CntrlCafetera
MOVH Rx, BYTEALTO DIRECCION CntrlCafetera
MOV [R1], Rx
```

❑ ¿Cuál es la codificación en el formato IEEE-754 del número 7,25? **Representar el resultado en hexadecimal.**

40E80000