

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

Junio 2010-prog-CPU

❑ Se ha desarrollado un programa para la CPU teórica que cuenta el número de veces que se pulsa la tecla "a" y muestra por pantalla ese número en binario. Para ahorrar espacio de pantalla, sólo se muestran los cinco bits más bajos del número. Cuando el número de pulsaciones excede las que pueden ser representadas correctamente, los cinco bits pasan a representarse sobre fondo rojo en lugar de fondo negro para indicar que hay un error.

Se ha conectado a la CPU un teclado y una pantalla. El programa principal está en un bucle infinito. Una rutina de interrupción se encarga de leer las teclas pulsadas. Para cada tecla, comprueba si es una "a" y en ese caso incrementa la variable global contador, que indica el número de veces que se ha pulsado esa tecla, e imprime por pantalla el nuevo valor del contador.

El programa utiliza el procedimiento `imprime_num_bin`, que recibe por la pila un número y un color (codificado en la parte alta de la palabra) e imprime los cinco bits más bajos de ese número con el color indicado. Para ello, va aplicando en un bucle una máscara que permite obtener el valor individual de cada dígito.

El código del programa se muestra a continuación:

```

1  origen 500h
2  inicio main
3  .pila 100h
4
5  .datos
6  contador VALOR 0
7
8  .codigo
9
10 procedimiento rut_teclado
11  push r0
12  push r1
13  push r2
14  push r3
15  push r4
16
17  movl r0, 00h ; r0: reg. datos teclado
18  movh r0, 26h
19
20  mov r1, r0 ; r1: reg. control teclado
21  inc r1
22
23  movl r2, 0 ; r2: máscara para detectar teclas
24  movh r2, 1
25
```

```

26  movl r5, 'a' ; r5: tecla buscada
27  movh r5, 0
28
29  tratar_tecla:
30  mov r3, [r1]
31  and r3, r3, r2 ; mirar si quedan pulsaciones
32  brz no_quedan_teclas
33
34  mov r3, [r0]
35  movh r3, 0
36  comp r3, r5
37  brnz tratar_tecla
38
39  ; Es la tecla buscada, incrementar contador
40  movl r3, bytebajo direccion contador
41  movh r3, bytealto direccion contador
42  mov r4, [r3] ; r4: contador
43  inc r4
44  mov [r3], r4
45
46  ; Poner el color adecuado según el valor
47  ; del contador
48  movl r3, 31
49  movh r3, 0
50  comp r3, r4 ; comprobar si es representable
51  [redacted]
52  movh r3, 7h ; poner fondo normal
53  jmp imprimir
54  error:
55  movh r3, 27h ; poner fondo de error
56
57  imprimir:
58  movl r3, 0
59  push r4
60  push r3
61  call imprime_numero_bin
62  inc r7
63  inc r7
64  jmp tratar_tecla
65
66  no_quedan_teclas:
67  pop r4
68  pop r3
69  pop r2
70  pop r1
71  pop r0
72  iret
73  finp
74
75  procedimiento imprime_numero_bin
76  push r6
77  mov r6, r7
78
79  push r0
80  push r1
81  push r2
82  push r3
83  push r4
84  push r5
85
86  ; Sacar parámetros de la pila:
87  ; r3: color
88  ; r0: número a imprimir
89
```

```

90  inc r6
91  inc r6
92  [redacted]
93  [redacted]
94  [redacted]
95
96  ; r1: base para imprimir (una posición antes
97  ; del comienzo de la interfaz de vídeo)
98  movh r1, 24h
99  movl r1, 0FFh
100
101  ; r2: máscara para obtener bits
102  movh r2, 0
103  [redacted]
104
105  ; r5: número de bits a imprimir
106  movh r5, 0
107  movl r5, 5
108
109  bucle:
110  ; Mirar si el bit es 1 ó 0 y poner en la
111  ; parte baja de r3 el ASCII del 1 o del 0
112  and r4, r2, r0
113  brz es_cero
114  movl r3, '1'
115  jmp escribir
116  es_cero:
117  movl r3, '0'
118  escribir:
119  add r6, r5, r1
120  mov [r6], r3
121
122  add r2, r2, r2
123  dec r5
124  brnz bucle
125
126  pop r5
127  pop r4
128  pop r3
129  pop r2
130  pop r1
131  pop r0
132  pop r6
133  ret
134  finp
135
136  main:
137  movl r0, bytebajo direccion rut_teclado
138  movh r0, bytealto direccion rut_teclado
139  movl r1, 7
140  movh r1, 0
141  mov [r1], r0
142  sti
143
144  jmp -1
145  fin
```

— ¿Qué instrucción falta en el hueco de la línea 51?

`brc error`

— ¿Qué instrucciones faltan en el hueco de la línea 92?

```
mov r3, [r6]
inc r6
mov r0, [r6]
```

— ¿Qué instrucción falta en el hueco de la línea 103?

```
movl r2, 1
```

— ¿Qué valor debe poner la interfaz del teclado en el bus de datos del sistema (SDB) después de recibir la señal INTA? Contestar en hexadecimal.

7

— Se sabe que la primera vez que la CPU ejecutó una instrucción del programa con código F417h, tardó 2 ns. ¿Cuál es la frecuencia de la CPU? Incluir las unidades en la respuesta.

2 GHz

— Sabiendo que en el segundo paso de ejecución de una instrucción el registro TMPS vale 0503h, ¿qué valor aparecerá en el tercer paso en el registro MDR? Contestar en hexadecimal.

3100h

— Sabiendo que tras la ejecución de la primera instrucción push que ejecuta el programa R7 vale 0658h, ¿cuál era el valor inicial de este registro? Contestar en hexadecimal.

065Bh

— En una configuración en la que el programa está funcionando correctamente, ¿cuántos dispositivos de memoria de 16K podría tener conectados la CPU como máximo?

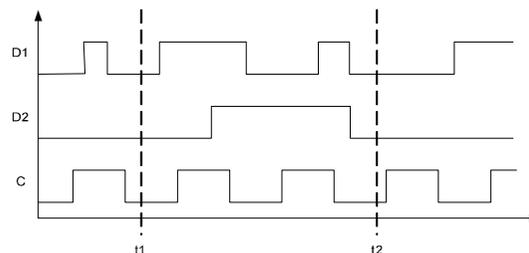
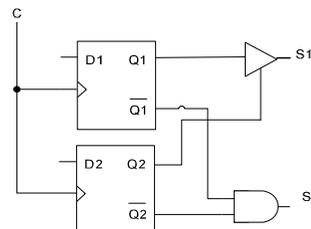
3

□ ¿Cuántos minterms aparecerán en la función canónica de la salida S_3 de un decodificador 3:8?

1

□ Dado el circuito de la figura

Junio2010-secuenciales



¿Qué valores (0, 1 o Z) tomarán las salidas S1 y S2 en los instantes t1 y t2?

	S1	S2
t1	Z	1
t2	0	0

□ Una unidad de control microprogramada para la CPU teórica genera señales de control que se interpretan como se muestra en la figura adjunta (sólo se muestran los 12 bits inferiores de la palabra de control).

Junio2010-UC-microprogramada

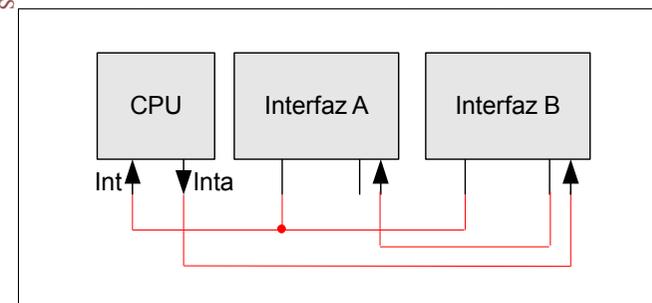
	SUB	ADD	ALU-TMPS	PC-IB	WRITE	READ	JUMP	TMPS-IB	IB-PC	IB-TMPE	IB-MAR	FIN
..												

Se sabe que en el paso 7 de una instrucción generó la palabra de control 620h. ¿Qué palabra de control se generará en el paso 8 de esa instrucción? Responder en hexadecimal.

019h

□ En la siguiente figura se representa la conexión de dos interfaces a la CPU teórica. Complétala el dibujo con las líneas necesarias para que las dos interfaces funcionen con interrupciones de tal manera que la interfaz B tenga más prioridad que la interfaz A.

Junio2010-ES



□ El código hexadecimal EB4 representa un número codificado usando un formato de representación de números reales en coma fija que utiliza 8 bits para la parte entera y 4 para la fraccionaria y que representa el signo utilizando el criterio del complemento a 2. Codificar ese valor utilizando el convenio IEEE-754 de representación de números reales. **Representar el resultado en hexadecimal.**

Junio2010-numerosreales

C1A60000

□ Codificar usando UTF-8 la cadena: «Caña:2». Datos: ASCII(A)= 41h, ASCII(Z)-ASCII(z)= 32, ASCII(:)= 3Ah, ASCII(0)= 30h, Unicode(ñ)= U+00F1. **Representar los bytes del resultado en hexadecimal.**

Junio2010-caracteres

43 61 C3 B1 61 3A 32

□ Se dispone de un sumador que opera con cantidades de 6 bits. Por una de las entradas se introduce el número -15 codificado en exceso a Z central para $n = 6$. Por la otra entrada se introduce el número 25 codificado en signo-magnitud.

Junio2010-ZCOS-Sumador

- ¿Cuál será el resultado de la suma? Interpretalo en complemento a 2 y responde en decimal.
- ¿Cuál es el valor de los flags de estado?

Resultado: -22 Z=0 C=0 O=1 S=1

□ El sistema ASCII de codificación de caracteres se diseñó utilizando 7 bits para representar cada carácter con el objeto de usar el octavo bit para controlar errores en la transmisión de datos. Cuando se enviaba un dato través de una línea de comunicaciones, se contaba el número de bits que ese dato tenía a 1 y, dependiendo de ese valor, el octavo bit se ponía a 1 o a 0. Este sistema de control de errores de transmisión se denomina **Paridad** y puede ser de dos tipos:

Par Cuando el número total de bits a 1 del dato que se transmite es par. En este caso, el octavo bit se pone a 1 si el número de bits del dato es impar o se pone a 0 cuando el dato ya tiene un número par de bits a 1. Por ejemplo, si se quiere transmitir el carácter «5» cuyo código ASCII es 0110101 el octavo bit sería un 0. En cambio, si el dato a transmitir fuera el carácter «4» cuyo ASCII es 0110100, el octavo bit se debería de poner a 1.

Impar Cuando el número total de bits a 1 del dato que se transmite es impar. En este caso el octavo bit se pone a 1 para conseguir que el número total de bits que se transmiten sea impar.

En el lado del receptor del mensaje se van contando los bits que se reciben a 1 y si la paridad no coincide con la esperada, se marca el dato como recibido con error.

El programa que se muestra a continuación va colocar el bit de paridad (el bit más significativo) al valor adecuado para tener paridad par en una lista de números de 15 bits. La diferencia más importante de nuestro programa con el ejemplo explicado del ASCII es que **nuestros datos son de 15 bits** (7 en el caso del ASCII) y **el bit de paridad es el bit dieciseis** (el octavo, en el caso ASCII).

Nuestro programa utiliza dos procedimientos para realizar su trabajo:

ContBit1 Este procedimiento cuenta el número de bits que están a 1 en un **dato de tipo palabra** que recibe por la pila. El resultado lo devuelve en el registro EAX. Para realizar su trabajo, ContBit1, va desplazando los bits del dato, comprobando si están a 1 e incrementando EAX en ese caso.

Paridad Este procedimiento añade el bit de paridad a una lista de números que recibe como parámetro. El procedimiento recibe como parámetros de entrada por la pila, y en este orden:

- La dirección de la memoria a partir de la cual se encuentra una lista de datos de tamaño palabra.

- El número de elementos que tiene esa lista. **Este parámetro será un dato de tamaño doble palabra.**

Teniendo todo esto en cuenta, contestar a las siguientes preguntas:

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

```

1  .386
2  .Model FLAT, stdcall
3  ExitProcess PROTO, :DWORD
4  .DATA
5  ListNum DW 3, 2, 60h, 1110h, 15
6  TamList DW 5
7
8  .CODE
9  ContBit1 PROC
10     [redacted]; Prólogo
11     [redacted]
12
13     push ebx
14     push ecx
15     xor eax, eax
16     ;Cargamos en DX el parámetro
17     mov bx, [ebp+8]
18     ;Tenemos que mirar los 16 bits
19     mov ecx, 16
20     bucle:
21     ;Desplazamos los bits y miramos si están a 1
22     s [redacted]
23     [redacted]
24     ;Si es 1 incrementamos el contador
25     inc eax
26     ;Si el bit<>1 pasamos al siguiente
27     otro:
28     loop bucle
29     ;restauramos registros
30     pop ecx
31     pop ebx
32     pop ebp
33     ;salimos del procedimiento
34     [redacted]
35     ContBit1 ENDP
36
37     Paridad PROC
38     [redacted]; Prólogo
39     [redacted]
40     ;Salvamos los registros
41     push ebx
42     push ecx
43     push edx
44     push esi
45
46     xor esi, esi
47     ;Accedemos al 1er parámetro: Dirección lista
48     mov ebx, [redacted]
49     ;Accedemos al 2º parámetro:          Tamaño lista
50     mov ecx, [redacted]
51     ProcList:
52     ;leemos el primer elemento de la lista
53     mov dx, [redacted]

```

```

54 ;contamos los bits a 1 que tiene
55     push dx
56     call ContBit1
57 ;comprobamos si el número de 1s es par
58     and eax, 1
59     jz otro
60 ;Si no es par, activamos el bit de paridad
61     or dx, 8000h
62 ;y salvamos el resultado en la lista
63     mov [redacted], dx
64     otro:
65     ;Si es par pasamos al siguiente elemento
66     inc esi
67     loop ProcList
68 ;restauramos registros y salimos
69     pop esi
70     pop edx
71     pop ecx
72     pop ebx
73
74     pop ebp
75     [redacted]
76     Paridad ENDP
77
78
79
80     inicio:
81     ;Apilamos la dirección de la lista de núm.
82     push OFFSET ListNum
83     ;Convertimos TamList en una cantidad de 32 bits
84     [redacted]
85     ;Apilamos
86     push eax
87     call Paridad
88
89     push 0
90     call ExitProcess
91     END inicio

```

— ¿Qué falta en el hueco de la línea 34?

RET 2

— ¿Qué debería haber en los huecos de las líneas 22 y 23 para que el programa funcione correctamente?

SHL/SHR/SAL BX, 1
JNC otro

— Completar los huecos de las líneas 10-11 y 38-39.

PUSH EBP
MOV EBP, ESP

— ¿Qué instrucción falta en el hueco de la línea 84?

MOVZX EAX, [TamList]

— Teniendo en cuenta los tipos de datos de la lista de números, ¿qué falta en las instrucciones de las líneas 53 y 63?

[ebx+esi*2]

— Completa las instrucciones de las líneas 48 y 50?

MOV EBX, [EBP + 12]
MOV ECX, [EBP + 8]

- Junio2010-memoria
- El circuito de activación de un dispositivo de memoria conectado a la CPU teórica implementa la función: $CA = a_{15}\overline{a_{14}}a_{13}$. Si sabemos que para construir el dispositivo se han usado 16 chips y un decodificador 2 : 4, ¿cuál es la distribución MxN de cada chip usado? ¿Y la del dispositivo?

MxN chip= 2Kx4 MxN dispositivo= 8Kx16

- Junio2010-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. Si al número de líneas del SAB de un sistema le restamos el número de líneas que van al circuito de activación de un dispositivo de memoria cualquiera instalado en ese sistema, obtenemos el mismo número que si sumamos las líneas de dirección que van a cualquiera de los chips que componen ese dispositivo con las líneas que van al decodificador, si lo tiene, de ese mismo dispositivo.
2. Una memoria flash es un ejemplo de circuito combinatorial puesto que al quitarse la alimentación, la memoria no pierde los datos que tiene almacenados.
3. Aunque la memoria A tiene un tiempo de lectura doble que el de la memoria B y un tiempo de escritura de la mitad, ambas memorias tienen el mismo tiempo de acceso.
4. La extrema volatilidad de las memorias SRAM es la característica que las hace más adecuadas para su uso en la memoria caché del sistema.

1

Junio2010-chicle1

- 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 único factor que influye en el rendimiento de los procesadores es la velocidad del reloj que controla los cambios de estado internos: a más frecuencia de reloj, más velocidad.
2. El registro IP de la arquitectura x86-64 contiene la dirección de la memoria en la que se encuentra el código de la siguiente instrucción a ejecutar.
3. Un microprograma de una instrucción es un conjunto de palabras de control que, sacadas de la memoria de control en el orden adecuado, permiten a una Unidad de Control ejecutar las instrucciones del juego básico de la CPU.
4. La arquitectura de 64 bits diseñada por Intel (IA-64) en sus procesadores Itanium I y II ha fracasado comercialmente debido a que, como tenían que mantener la compatibilidad del juego de instrucciones con los procesadores de 16 bits, el rendimiento disminuía gravemente.

3