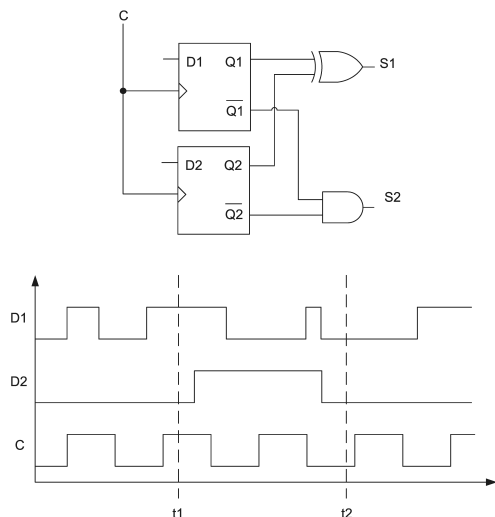


Todas las preguntas tienen la misma puntuación. Cada respuesta correcta suma un punto. Cada respuesta incorrecta, ilegible o vacía no suma ni resta. La nota del examen se obtiene multiplicando el número de preguntas correctas por 10 y dividiendo por el número total de preguntas del examen

1^{er} Parcial

- Dado el circuito de la figura



¿Qué valores tomarán las salidas S1 y S2 en los instantes t1 y t2?

	S1	S2
t1	1	0
t2	1	0

- Se tiene un sumador capaz de operar con cantidades de 6 bits. Por una de las entradas del sumador se introduce el número negativo con mayor 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 si se interpreta que está codificado en binario natural? Escribe el resultado en **decimal**.

100000 + 011111 = 111111 (63)

- ¿Qué número decimal representa la secuencia de bits 80100000h si es un número real codificado en IEEE 754? Contesta en decimal usando potencias de 2.

-2⁻¹²⁹

- Completa los caracteres que faltan en el interior de las casillas, teniendo en cuenta los caracteres que ya hay y sus códigos ASCII que se muestran encima. Nota: *oct* significa *codificado en octal*.

41h	141 oct	154 oct	6Ah	60 oct	34h
A	a	l	j	0	4

- ¿Cuántas patillas debe tener un chip SRAM de 256kx8? Nota: *Contar también las señales de control*.

29

- En la ALU de la CPU elemental se introduce en la entrada A el dato -10 codificado en signo-magnitud y en la entrada B el 250 codificado en binario natural. Además, se selecciona la operación de resta. ¿Cuál es el resultado de la operación y el estado de los flags resultante? Dar el resultado en **hexadecimal**.

Resultado: 7F10h Z=0 C=0 O=1 S=0

2º Parcial

- Se quiere añadir a la CPU teórica la posibilidad de que los procedimientos retiren los parámetros de la pila a la manera de Intel, para ello se ha de implementar la instrucción: RET N

El dato inmediato N irá codificado en el byte de menor peso del código de la instrucción. Indicar la secuencia de señales de control necesarias para su ejecución.

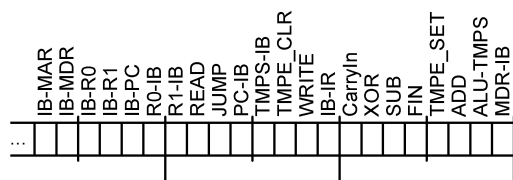
Paso	Señales
4	R7-IB, TMPE-CLR, CarryIn, ADD, ALU-TMPS, IB-MAR, Leer
5	TMPS-IB, IB-TMPE
6	MDR-IB, IB-PC
7	JUMP, ADD, ALU-TMPS
8	TMPS-IB, IB-R7, FIN

- 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.

- Todas las instrucciones de la CPU almacenan en algún momento en el Registro de Instrucción el contenido de la memoria cuya dirección se encuentra en el registro Contador de Programa.
- Como podemos comprobar en las hojas de codificación, el número total de instrucciones distintas que nuestra CPU puede ejecutar es de 36 (incluyendo a la instrucción NOP).
- En la memoria de microprograma, como se encuentra dentro de la propia CPU, se almacena información de manera temporal para mejorar las prestaciones de los accesos a memoria.
- Todas las instrucciones modifican al menos uno de los registros de propósito general.

a

- 2PFinalJunio2006-07-UC-microprogramada
- Una unidad de control microprogramada genera señales de control que se interpretan como se muestra en la figura adjunta (sólo se muestran los 22 bits inferiores de la palabra de control).



Se sabe que el reloj de esta CPU tiene una frecuencia de 2 MHz. Durante los 2 μ s que ha durado la ejecución de una determinada instrucción aparecieron únicamente los valores 0820 y 4A0B en los buses de datos y direcciones, respectivamente.

- Escribe el microprograma correspondiente a los pasos no comunes (4º y superiores) de la instrucción que se ha ejecutado. Expresa las palabras de control en **hexadecimal**

088010h (00 1000 1000 0000 0001 0000)

- Después de pasados esos 2 μ s ¿Cuál es el primer valor que aparecerá en el bus interno de la CPU? Responde en **hexadecimal**.

4A0C

- Si t_0 es el instante en el que se generó el flanco ascendente que inició la ejecución de la instrucción, ¿a partir de qué valor de tiempo podemos encontrar el dato 0820 en el bus de datos? Incluir las unidades de tiempo usadas. Ejemplo de respuesta: $t_0 + 2s$.

$t_0 + 1\mu s$

- 2PFinalJunio2006-07-prog-ens-A
- Se ha escrito un programa que cuenta los números de una lista de números enteros que son mayores o iguales que un valor determinado. El programa utiliza el procedimiento *analiza* para realizar el trabajo. El procedimiento recibe los siguientes parámetros y en este mismo orden:

- La dirección de la memoria en la que se encuentra la lista de los números a procesar.
- Número de elementos de la lista
- El valor de referencia con el que hay que comparar cada número.
- Dirección en la que se almacena el número de elementos de la lista mayores o iguales que el de referencia.

En el listado adjunto se muestra el código del programa y el del procedimiento *analiza*. En el programa principal se han eliminado algunas instrucciones repetitivas por lo que los huecos no coinciden exactamente con el número de instrucciones ausentes.

Se sabe que al inicio del programa los registros de la CPU tenían los siguientes valores:

R0=0000	R3=0000	R6=0000
R1=0000	R4=0000	R7=0357
R2=0000	R5=0000	PC=0308

Nota: Recordar que la condición $A > B$ para números enteros se cumple cuando $SF = OF$

```

1      ORIGIN 300h
2      INICIO empieza
3      .Pila 20
4      .DATOS
5      Lista VALOR 7, 10, 0ABCDh, 8000h, -6
6      Nelem VALOR 5
7      Ref VALOR 7
8      May_ig VALOR 0
9      .CODIGO
10
11     empieza:
12     ;Se apila el parámetro Lista
13     .....
14     .....
15     ;Se apila Nelem
16     .....
17     .....
18     ;Se apila Ref
19     .....
20     .....
21     ;Se apila May_ig
22     .....
23     .....
```

```

24      CALL Analiza
25     ;Se restaura la pila
26     .....
27     .....
28     JMP -1
29
30
31     PROCEDIMIENTO Analiza
32     PUSH R6
33     MOV R6, R7
34     PUSH R0
35     PUSH R1
36     PUSH R2
37     PUSH R3
38     PUSH R4
39     PUSH R5
40
41     INC R6
42     ;Direccion del resultado
43     INC R6
44     MOV R5, [R6]
45     ;Valor de la referencia
46     INC R6
47     MOV R4, [R6]
48     ;Elementos de la lista
49     INC R6
50     MOV R1, [R6]
51     ;Direccion lista de números
52     INC R6
53     MOV R2, [R6]
54
55     XOR R0, R0, R0
56     ;¿Hay números en la lista?
57     COMP R0, R1
58     BRZ Salir
59
60     repite: ;Accedemos a la lista de n°
61     MOV R3, [R2]
62     ; Comprobamos si es >= que la referencia
63     COMP R3, R4
64     ver_OF
65     menor
66     JMP may_ig
67     ver_OF: BRNO menor
68     may_ig: ;Es mayor: se incrementa contador
69     INC R0
70
71     menor:
72     ;Es menor. Pasamos al siguiente n° de la lista
73     INC R2
74     ;¿Quedan elementos?
75     DEC R1
76     BRNZ repite
77
78     Salir: ; Retornamos la cantidad de
79     ; mayores e iguales
80     MOV [R5], R0
81
82     POP R5
83     POP R4
84     POP R3
85     POP R2
86     POP R1
87     POP R0
```

```

88      POP R6
89      RET
90  FINP
91
92      FIN

```

— ¿Qué instrucciones de salto condicional faltan en los huecos de las líneas 64 y 65?

BRS
BRO

— Codifica la instrucción que se encuentra en la línea 58 del programa y expresa el resultado en **hexadecimal**.

F40A

— ¿Cuál es el valor del registro R6 después de ejecutarse la instrucción de la línea 86? Contestar en **hexadecimal**.

0356

— Escribe el código necesario para pasar el parámetro Ref al procedimiento.

```

MOVL Rx, BYTEBAJO DIRECCION Ref
MOVH Rx, BYTEALTO DIRECCION Ref
MOV Ry, [Rx]
PUSH Ry

```

— Después de ejecutarse la instrucción de la línea 80, ¿qué posición de la memoria se ha modificado y qué valor contiene? Contestar en **hexadecimal**.

Dirección: 0307h Contenido: 2

3^{er} Parcial

3PFinalJunio2006-07-rutina-interrupcion

- Durante la ejecución de la primera instrucción del código que se muestra a continuación, la CPU del computador elemental recibe una interrupción.

```

1  ADD R0, R1, R2
2  BRNZ no_cero
3  XOR R3, R3, R3
4  MOVL R3, 5
5  JMP sigue
6  no_cero:
7  XOR R3, R3, R3
8  MOVL R3, 7
9  sigue:

```

En la fase de aceptación de esa interrupción y antes de que se ejecutase ninguna instrucción de la rutina de tratamiento de la interrupción, aparecieron, no necesariamente en ese orden, en el bus de direcciones del sistema los siguientes valores: 3A68h, 000Ch, 3A67h. Durante la misma fase, en el bus de datos del sistema aparecieron, por orden, los siguientes datos: 0001h, 2F53h, 000Ch, 7BD0h. Con estos datos, contestar a las siguientes preguntas:

— ¿Qué valor tendrá el registro R3 al finalizar la ejecución de ese trozo de código?

7

— ¿Cuál será la dirección de la instrucción BRNZ no-cero?

2F53h

— ¿Cuál es el número de interrupción que se ha generado? Contestar en **decimal**.

12

3PFinalJunio2006-07-espacio-chips

- Mediante chips de 8Kx1 se desea construir un dispositivo de memoria que ocupa la mitad del espacio de direcciones de la CPU elemental, dejando la otra mitad para E/S. ¿Cuántos chips serán necesarios?

64

3PFinalJunio2006-07-prog-intel-A

- El siguiente programa trata cadenas de caracteres tipo pascal. Estas cadenas de caracteres se caracterizan por almacenar en la primera posición el número de caracteres que contienen, con lo cual se evita tener que incluir un carácter que indique donde termina la cadena (típicamente el 0).


En el programa se han realizado dos procedimientos:

CopyPStr: Permite copiar una cadena pascal en otra. Recibe como parámetros a través de la pila la dirección de la cadena destino y la dirección de la cadena origen (en este orden).

AddCarToPStr: Añade un carácter a una cadena pascal. Recibe como parámetros a través de pila, y en este orden, la dirección de la cadena destino y el carácter a añadir.

Durante la ejecución del programa, la sección de datos se almacena a partir de la dirección de memoria 00404000h.

```

1
2  .386
3  .model flat, stdcall
4  ExitProcess proto, ExitCode:dword
5
6  .data
7  Cadena1 DB 4, "Hola"
8  CadenaDestino DB 128 DUP(0)
9
10 .code
11
12 CopyPStr proc
13     push ebp
14     mov ebp, esp
15
16     push esi
17     push edi
18     push eax
19     push ecx
20
21     mov esi, [ebp + 8] ; cadena origen
22     mov edi, [ebp + 12] ; cadena destino
23
24     mov cl, [esi]
25     mov [edi], cl
26     movzx ecx, cl
27     inc edi
28     inc esi
29 bucle:
30     mov al, [esi]
31     
32
33     loop bucle
34
35
36     pop ecx
37     pop eax
38     pop edi
39     pop esi
40     pop ebp

```

```
41     ret 8
42 CopyPStr endp
43
44 AddCarToPStr proc
45     push ebp
46     mov ebp, esp
47
48     push edi
49     push eax
50     push ebx
51
52     mov eax, [ebp + 8] ; caracter
53     mov edi, [ebp + 12] ; cadena destino
54
55     mov bl, [edi]
56     movzx ebx, bl
57     mov [edi + ebx + 1], al
58 ; Actualiza longitud
59
60
61     pop ebx
62     pop eax
63     pop edi
64     pop ebp
65     ret 8
66 AddCarToPStr endp
67
68 inicio:
69     push offset CadenaDestino
70     push offset Cadenal
71     call CopyPStr
72
73     push offset CadenaDestino
74     push 's'
75     call AddCarToPStr
76
77     push 0
78     call ExitProcess
79     end inicio
```

— ¿Qué instrucción o instrucciones faltan en el hueco de la línea 31?

```
MOV [edi], al
inc edi
inc esi
```

— ¿Qué instrucción o instrucciones faltan en el hueco de la línea 59?

```
inc byte ptr [edi]
```

— ¿Qué valor de tipo byte hay almacenado en la posición de memoria [esp+4] justo antes de ejecutar la instrucción call CopyPStr? Responder en **hexadecimal**.

```
05h
```

— En el procedimiento CopyPStr, ¿cuántas instrucciones provocan una lectura de memoria?. Nota: *Las instrucciones del hueco de la línea 31 no provocan lecturas de memoria. Además, no se considera lectura de memoria la fase de búsqueda de la instrucción.*

```
10
```