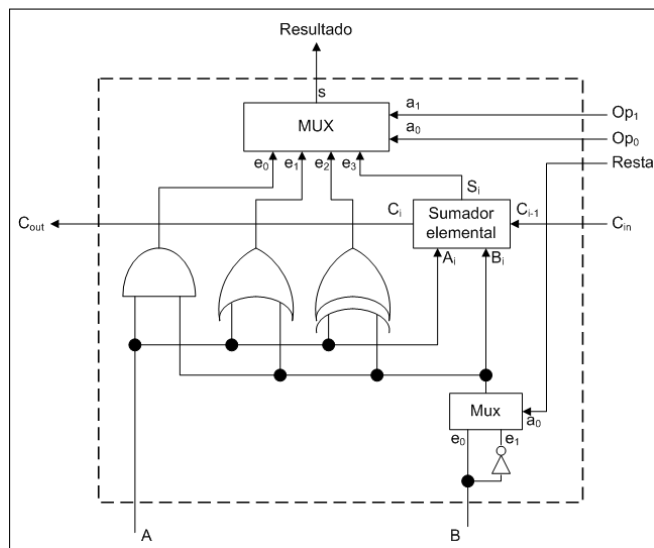


Todas las preguntas tienen la misma puntuación. Cada respuesta correcta suma un punto. Cada respuesta incorrecta, ilegible o vacía no suma ni resta. La nota del examen se obtiene multiplicando el número de preguntas correctas por 10 y dividiendo por el número total de preguntas del examen

— En la siguiente figura se muestra una ALU de 1 bit, capaz de realizar las operaciones AND, OR, XOR, Suma y Resta, a la que le faltan elementos y en la que algunos de los elementos mostrados no están identificados. Completa la ALU con los elementos que faltan, identifica todos los componentes, únelos correctamente y nombra todas las entradas y salidas de la ALU y de los elementos que la componen.



- ❑ Codifica en el formato IEEE-754 de representación de números reales la cantidad 129, 25 y expresa el resultado en **hexadecimal**.

43014000

- ❑ Si por una de las entradas de un sumador elemental de 5 bits se introduce el número más grande que se puede codificar en complemento a 2 y por la otra se mete el número 1 codificado en exceso a Z central, ¿cuál será el resultado que obtendremos a la salida del sumador si lo interpretamos codificado en exceso a Z central? ¿Qué valor tendrán los bits del registro de estado? **Dar el valor del resultado en decimal.**

Resultado= -16 Z= 1 C= 1 O= 0 S= 0

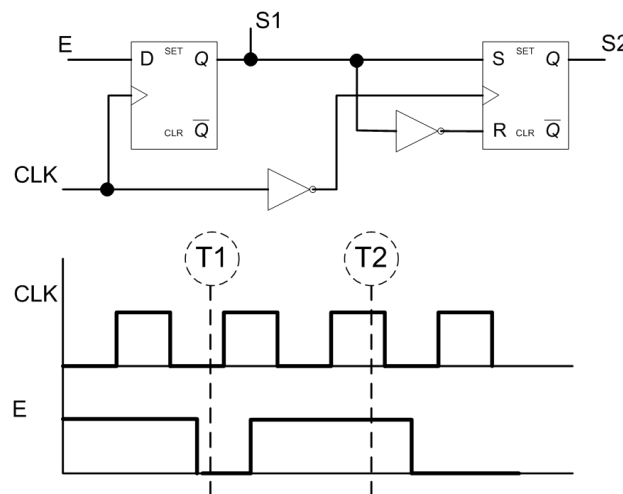
- ❑ La siguiente secuencia de bytes es la codificación UTF-8 de una serie de caracteres Unicode. ¿Qué caracteres Unicode son esos?

35 EA 9B AB 73

Contestar con el código Unicode en la forma U+XXXX.

U+0035 U+A6EB U+73

- ❑ Teniendo en cuenta el siguiente circuito y los cronogramas mostrados:



¿Cuál será el valor de S1 y S2 en T1 y T2?

T1: S1= 1 S2= 1 | T2: S1= 1 S2= 0

- ❑ 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.

- Una de las razones por las que la arquitectura x86-64 es más rápida que la x86-32 es porque la primera tiene más registros.
- La instrucción STI de la CPU teórica genera una interrupción.
- En programas Java, el enlazador es el encargado de traducir bytecodes de la máquina virtual de Java a instrucciones de la arquitectura que ejecute el programa.
- En ASCII no se pueden representar a la vez caracteres latinos, griegos y japoneses, mientras que en ISO-Latin sí.

1

- ❑ En una CPU teórica recién inicializada con su memoria a cero se carga un programa que utiliza una interrupción para comunicarse con un periférico. El programa contiene un error: activa las interrupciones antes de haber inicializado la tabla de vectores de interrupción. Si se produce una interrupción del periférico antes de que se haya ejecutado el código de inicialización la tabla de vectores, ¿qué ejecución se ejecutará? Contestar con el mnemónico.

NOP

- ❑ Se desea añadir una nueva instrucción para la CPU teórica con el mnemónico SWAP R0, R1. Esta instrucción debe intercambiar los valores de R0 y R1 sin modificar ningún otro registro que pueda afectar a los programas. Indicar la secuencia de señales de control necesarias para su ejecución en el menor número de pasos posibles.

Paso	Señales
4	R0-IB, IB-TMPE, TMPE-CLR, ADD, ALU-TMPS
5	R1-IB, IB-R0
6	TMPS-IB, IB-R1, FIN
7	-

- ❑ 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).

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

Se sabe que en el paso 4 de una instrucción se generó la palabra de control 104h y en el paso 6, la palabra 019h. ¿Qué palabra de control se generó en el paso 5 de esa instrucción? Responder en **hexadecimal**.

620h

- ❑ En el programa que se muestra a continuación se realiza la asignación automatizada de escaños a partidos políticos usando el método D'Hont. El método, para asignar  $N$  escaños, divide los votos obtenidos por cada partido,  $V_i$ , entre 1, 2, 3... hasta  $N$  y asigna un escaño a cada uno de los  $N$  cocientes más grandes obtenidos. El algoritmo va calculando cocientes y asignando escaños de uno en uno hasta asignarlos todos. Para ello usa una lista con los votos de cada partido (valores  $V_i$ ), otra lista con el valor de los coeficientes por el que hay que dividir esos votos (varían entre 1 y  $N$ ) y una tercera dónde se almacenan los cocientes obtenidos. Todas las listas contienen números naturales.

Por ejemplo, supongamos tres partidos A, B y C que recibieron, respectivamente, 450000, 375000 y 200000 votos, y 5 escaños para repartir. Inicialmente tendríamos que la lista `VotosPar` tendría los valores 450000 375000 200000, la lista de coeficientes `CoefPar` contendría: 1 1 1 y la lista de cocientes `CociPar` estaría inicializada a cero. En la primera iteración calculamos los cocientes de dividir los votos recibidos por el coeficiente de cada partido (1 en este caso). Obtenemos:

	A	B	C
VotosPar	450000	375000	200000
CoefPar	1	1	1
CociPar	450000	375000	200000

Buscamos el cociente más grande (450000) y asignamos el primer escaño al partido A. Ahora tenemos que incrementar su coeficiente en una unidad. La lista de coeficientes sería: 2 1 1. Con estos valores, volvemos a calcular los cocientes y llegaríamos a la siguiente situación:

	A	B	C
VotosPar	450000	375000	200000
CoefPar	2	1	1
CociPar	225000	375000	200000

El segundo escaño se asigna al partido B ya que es el que tiene el mayor cociente. Como en el caso anterior, una vez asignado el escaño, tenemos que incrementar su coeficiente en una unidad. La lista de coeficientes sería: 2 2 1. Volvemos a calcular los cocientes y tendríamos la siguiente situación:

	A	B	C
VotosPar	450000	375000	200000
CoefPar	2	2	1
CociPar	225000	187500	200000

Con estos valores, el escaño se asignaría a A. Una vez asignado, tendríamos que incrementar su coeficiente antes de volver a iniciar el ciclo de cálculo. El algoritmo terminará cuando se hayan asignado todos los escaños.

En nuestro programa, el procedimiento `CalCoci` recibe como parámetros de entrada, y en este orden:

- La dirección de la lista que contiene los votos recibidos por cada partido.
- La dirección de la lista que contiene los coeficientes por los que hay que dividir esos votos.
- La dirección de la lista en la que hay que dejar el cociente de cada una de las divisiones.
- El número de elementos (partidos) que contiene cada una de las listas anteriores.

El procedimiento calcula el cociente de dividir los votos por el coeficiente y lo almacena en la lista de cocientes. La división la realiza simplemente contando el número de veces que puede restar al total de votos el coeficiente.

El procedimiento `BuscaMax` recibe como parámetros la dirección de una lista de números naturales y el número de elementos de esa lista y devuelve en `EAX` el índice, empezando por cero, que ocupa en esa lista el valor más grande. Ante la entrada 2250 3750 2000 devolvería en `EAX` el valor 1. No se muestra el código de este procedimiento.

```
1 ;. . . (Cabecera eliminada) ...
2 .DATA
3 VotosPar DD 340, 280, 160, 60, 15
4 CociPar DD 0, 0, 0, 0, 0
5 CoefPar DD 1,1,1,1,1
6 EscanPar DB 0, 0, 0, 0, 0
7 TotEscan DB 7
8 NumPart DD 5
9 .CODE
10 CalCoci PROC
11     push ebp
12     mov ebp, esp
13 ;Salvaguada de los registros
14     push eax
15     push ebx
16     push ecx
17     push edx
18     push edi
19     push esi
20 ;acceso a los parámetros
21 ;Carga en ECX el nº de partidos
22     mov ecx, [ebp + 8]
23 ;Direccion lista cocientes
24     mov edi, [ebp + 12]
25 ;Direccion lista coeficientes
26     mov esi, [ebp + 16]
```

```
27 ;Direccion lista votos totales
28     mov ebx, [ebp + 20]
29
30 bucle:
31     xor eax, eax
32 ;Carga en EDX del nº de votos
33     mov edx, [ebx]
34 ;Aquí se divide el nº de votos (V) por el
35 ;coeficiente (c) correspondiente
36 ;y se calcula el cociente (Coc) con el algoritmo:
37 ;     Si V > c entonces
38 ;         V = V-c
39 ;         Coc++
40 divide:
41     cmp edx, [esi]
42     [esi]
43     sub edx, [esi]
44     inc eax
45     jmp divide
46 ;Se guarda el cociente y se pasa a otro partido
47 otropar:
48     mov [edi], eax
49     add ebx, 4
50     add esi, 4
51     add edi, 4
52     loop bucle
53
54 fin:
55     pop esi
56     pop edi
57     pop edx
58     pop ecx
59     pop ebx
60     pop eax
61     [esi]
62
63 CalCoci ENDP
64
65 inicio:
66     push OFFSET VotosPar
67     push OFFSET CoefPar
68     push OFFSET CociPar
69     push [NumPart]
70     call CalCoci
71
72 Asignacion:
73 ;Se busca el cociente máximo
74     push OFFSET CociPar
75     push [NumPart]
76     call BuscaMax
77
78 ;incrementamos escaños del partido
79     inc [esi]
80 ;incrementamos el coeficiente del partido
81     inc DWORD PTR [CoefPar + eax*4]
82 ;decrementamos escaños
83     dec BYTE PTR [TotEscan]
84 ;Si quedan por asignar escaños, volvemos al principio
85     [esi]
86     jne inicio
87     push 0
88     call ExitProcess
89 END inicio
```

— ¿Qué instrucciones faltan en los huecos de las líneas 61 y 62?

```
pop ebp
ret 16
```

— ¿Qué instrucción falta en el hueco de la línea 42?

```
jb otropar
```

— ¿Qué instrucción falta en la línea 85?

```
CMP [TotEscan], BYTE PTR 0
```

— Completa la instrucción de la línea 79.

```
inc BYTE PTR [EscanPar + eax]
```

— Si las instrucciones que van de la línea 66 a la 70 hubiesen sido generadas por un compilador de C++ usando la convención de llamadas cdecl, ¿cuál sería la línea del programa fuente que originó ese código? *Nota1: Suponemos que en el programa C++ existen las variables VotosPar, CoefPar, CociPar y NumPart.*

*Nota2: En C++, cuando se quiere pasar la dirección de una variable, se usa la expresión &variable.*

```
CalCoci( NumPart, &CociPart,
        &CoefPart, &VotosPar);
```

— Se sabe que durante la ejecución de la instrucción que se encuentra en la línea 48 el registro EBP contiene el valor 0012FFACh y que las posiciones de memoria que van de la 0012FFA4h a la 0012FFBBh contienen, de forma consecutiva, los siguiente bytes:

```
00 E0 F0 7F 00 00 00 00 F0 FF 12 00
95 10 40 00 05 00 00 00 14 40 40 00
```

¿A partir de qué posición de memoria se encuentra la instrucción de la línea 74?

```
00401095
```

❑ Se tiene un periférico de seguridad para la CPU teórica con interfaz mapeada en la dirección D000h y vector de interrupción 6. La interfaz sólo contiene un registro de estado que indica con un 1 en el bit 0 si se ha producido un escape de agua, en el bit 1 si se ha producido uno de gas y en el bit 2 si ha saltado la alarma anti-intrusos. Escribir el fragmento de código necesario para detectar si se ha producido un escape de gas y, en ese caso, que se salte al procedimiento avisa\_escape\_gas. Utilizar sólo los registros R0 y R1.

```
MOVL R0, 00h ; Reg. estado
MOVH R0, 0D0h
MOV [R0], R0
MOVL R1, 2 ; Máscara
MOVH R1, 0
AND R1, R1, R0
BRZ seguir
CALL avisa_escape_gas
seguir:
```

❑ 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 la arquitectura Von Neumann, los periféricos y la memoria están comunicados con la CPU sólo a través de los buses de datos y de direcciones.
2. Si dos CPUs tienen el mismo juego de instrucciones, siempre será más rápida la que tenga un reloj de mayor frecuencia.
3. Si se desean almacenar números enteros en 32 bits, es mejor utilizar el formato de coma flotante IEEE-754 que complemento a 2 porque, al tener IEEE-754 un rango mayor que complemento a 2, se pueden representar más números enteros.
4. En alguno de los pasos de aceptación de una interrupción, la dirección de la rutina de interrupción aparece en el bus de datos.

4

❑ Se sabe que para cubrir el 100 % de la memoria de un computador que tiene un SAB de 20 líneas y un SDB de 16 líneas son necesarios 8 dispositivos de memoria. Sabemos que cada dispositivo está implementado con un decodificador de 3 entradas y una serie de chips de memoria organizados en filas y columnas. A cada columna de chips se distribuyen 4 líneas de datos. ¿Cuál es la distribución MxN de cada dispositivo? ¿Cuál es la distribución MxN de cada chip? **Dar los valores de M y N en decimal, ej 512Mx32.**

MxN dispositivo= **128Kx16**      MxN chip= **16Kx4**

❑ Se ha desarrollado un programa para la CPU teórica que incluye dos procedimientos:

- Multiplica: recibe por la pila dos factores y devuelve en R0 su multiplicación interpretándolos como números naturales.
- PotenciaMem: recibe por la pila (en este orden) una número que interpreta como base, otro que interpreta como potencia y una dirección de memoria. Calcula el resultado de elevar la base a la potencia y lo guarda en la dirección de memoria. El procedimiento supone todos los números naturales.

El código del programa se muestra a continuación:

```
1  ORIGIN 2000h
2  INICIO main
3  .PILA 20h
4
5  .DATOS
6  resultado1 VALOR 0
7  resultado2 VALOR 0
8
9  .CODIGO
10
11 PROCEDIMIENTO Multiplica
12   PUSH R6
13   MOV R6, R7
14
15   PUSH R1
16   PUSH R2
17
18   INC R6
19   INC R6
20   MOV R1, [R6]
21   INC R6
22   MOV R2, [R6]
23
24   ; Si algún factor es 0, devolver 0
25   XOR R0, R0, R0
26   COMP R2, R0
27   BRZ salirMultiplica
28   COMP R1, R0
```

```
29 BRZ salirMultiplica
30
31 bucleMultiplica:
32 ADD R0, R0, R1
33 DEC R2
34 BRNZ bucleMultiplica
35
36 salirMultiplica:
37 POP R2
38 POP R1
39
40 POP R6
41 RET
42 FINP
43
44 PROCEDIMIENTO PotenciaMem
45 PUSH R6
46 MOV R6, R7
47
48 PUSH R0
49 PUSH R1
50 PUSH R2
51 PUSH R3
52
53 ; Extracción de parámetros
54
55
56
57
58 MOV R1, [R6]
59 INC R6
60 MOV R2, [R6]
61
62 ; Si la base es 0, el resultado es 0
63 XOR R0, R0, R0
64 COMP R2, R0
65 BRZ salirPotencia
66
67 ; Si el exponente es 0, el resultado es 1
68 COMP R1, R0
69 BRNZ calcularPotencia
70 MOVL R0, 1
71 JMP salirPotencia
72
73 calcularPotencia:
74 MOV R0, R2 ; Incializar el resultado
75 DEC R1 ; Equivale a haber multiplicado una vez
76
77 buclePotencia:
78 PUSH R2
79 PUSH R0
80 CALL Multiplica
81 INC R7
82 INC R7
83
84
85
86 salirPotencia:
87 MOV [R3], R0
88 POP R3
89 POP R2
90 POP R1
91 POP R0
92
```

```
93 POP R6
94 RET
95 FINP
96
97 main:
98 MOVH R0, 0
99 MOVL R0, 9
100 PUSH R0
101 MOVL R0, 5
102 PUSH R0
103 MOVL R1, BYTEBAJO DIRECCION resultado1
104 MOVH R1, BYTEALTO DIRECCION resultado1
105 PUSH R1
106 CALL PotenciaMem
107 INC R7
108 INC R7
109 INC R7
110
111 MOVL R0, 2
112 PUSH R0
113 MOVL R0, 3
114 PUSH R0
115 MOVL R1, BYTEBAJO DIRECCION resultado2
116 MOVH R1, BYTEALTO DIRECCION resultado2
117 PUSH R1
118 CALL PotenciaMem
119 INC R7
120 INC R7
121 INC R7
122 FIN
```

— ¿Qué instrucción o instrucciones faltan en el hueco de la línea 54?

```
INC R6
INC R6
MOV R3, [R6]
INC R6
```

— ¿Qué instrucción o instrucciones faltan en el hueco de la línea 83?

```
DEC R1
BRNZ buclePotencia
```

— ¿Cuál sería el número más grande que se podría poner en la línea 113 sin que el programa calculase un resultado incorrecto? Responder en decimal.

15

— Se sabe que la primera vez que la CPU ejecutó una instrucción del programa con código F403h, tardó 1 ns. ¿Cuál es la frecuencia de la CPU? Incluir las unidades en la respuesta.

4 GHz

— ¿Qué valor hay en el bus interno en el paso 2 de la instrucción PUSH R6 del procedimiento Multiplica? Contestar en hexadecimal.

2003h

— Se sabe que la primera dirección de memoria que se modifica una vez que se empieza a ejecutar el programa es la 2071h. ¿En qué dirección está la instrucción de la línea 121? Contestar en hexadecimal.

2051h