

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

❑ Se está desarrollando un teléfono móvil que usa una CPU basada en la CPU teórica. Dentro del proceso de desarrollo se están haciendo algunos programas de prueba de procedimientos básicos. Entre ellos, se encuentran estos dos:

- **numValido:** Recibe por la pila la dirección donde está almacenada una secuencia de nueve caracteres y comprueba que todos se corresponden con el código ASCII de un número. Si es así, devuelve uno en R0; en caso contrario, devuelve cero en R0.
- **listaValida:** Recibe por la pila la dirección de una lista de números y la cantidad de elementos de la lista. Comprueba que todos sean números válidos según los criterios de **numValido**. Si todos son válidos, devuelve uno en R0; en caso contrario, devuelve cero en R0.

Sabemos que, justo después de ejecutar la instrucción `PUSH R6` del procedimiento `numValido`, los registros de la CPU tenían los siguientes valores:

|         |         |         |
|---------|---------|---------|
| R0=0001 | R3=0001 | R6=0A9E |
| R1=0004 | R4=0009 | R7=0A94 |
| R2=0900 | R5=0000 | PC=0942 |




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

```

1  ORIGIN 0900h
2  INICIO main
3  .PILA 50h
4  .DATOS
5  lista1 VALOR "669126212", "6wa221233"
6          VALOR "9-8088990", "985943932"
7  valida1 VALOR 0
8  lista2 VALOR "688638848", "699188833"
9          VALOR "611123123"
10 valida2 VALOR 0
11 .CODIGO
12
13 PROCEDIMIENTO numValido
14     PUSH R6
15     MOV R6, R7
16
17     PUSH R1
18     PUSH R2
19     PUSH R3



```

```

20     PUSH R4
21     PUSH R5
22
23     INC R6
24     INC R6
25     MOV R2, [R6] ; R2: puntero al número
26
27     MOVL R1, 6 ; R1: contador
28     MOVH R1, 0
29
30     MOVL R4, '0' ; R4: '0'
31     MOVH R4, 0
32     MOVL R5, '9' ; R5: '9'
33     MOVH R5, 0
34
35     MOVL R0, 1 ; R0: valor de retorno
36     MOVH R0, 0
37
38     bucleNumValido:
39         MOV R3, [R2] ; R3: cifra actual
40         COMP R3, R4
41         
42         COMP R5, R3
43         
44         INC R2
45         DEC R1
46         BRNZ bucleNumValido
47         JMP salirNumValido
48
49     invalido:
50         XOR R0, R0, R0
51
52     salirNumValido:
53         POP R5
54         POP R4
55         POP R3
56         POP R2
57         POP R1
58
59         POP R6
60         RET
61     FINP
62
63     PROCEDIMIENTO listaValida
64         PUSH R6
65         MOV R6, R7
66
67         PUSH R1
68         PUSH R2
69         PUSH R3
70         PUSH R4
71
72         INC R6
73         INC R6
74         MOV R1, [R6] ; R1: núm. elementos
75
76         ; Poner en R2 la dirección de la lista
77         
78
79
80         MOVL R4, 9 ; R4: 9 para pasar al sgte.
81         MOVH R4, 0 ; número
82
83         MOVL R0, 1 ; R0: valor de retorno

```

```

84         MOVH R0, 0
85
86         XOR R3, R3, R3 ; cero para comparar
87         COMP R1, R3
88         BRZ salirListaValida
89
90         MOVL R3, 1 ; uno para comparar
91
92     bucleListaValida:
93         PUSH R2
94         CALL numValido
95         INC R7
96         COMP R3, R0
97         BRNZ invalida
98         ADD R2, R2, R4
99         DEC R1
100        BRNZ bucleListaValida
101        JMP salirListaValida
102
103     invalida:
104         XOR R0, R0, R0
105
106     salirListaValida:
107         POP R4
108         POP R3
109         POP R2
110         POP R1
111
112         POP R6
113         RET
114     FINP
115
116     main:
117         MOVL R0, BYTEBAJO DIRECCION lista1
118         MOVH R0, BYTEALTO DIRECCION lista1
119         PUSH R0
120         MOVL R0, 4
121         MOVH R0, 0
122         PUSH R0
123         CALL listaValida
124         
125
126         MOVL R1, BYTEBAJO DIRECCION valida1
127         MOVH R1, BYTEALTO DIRECCION valida1
128         MOV [R1], R0
129
130
131         MOVL R0, BYTEBAJO DIRECCION lista2
132         MOVH R0, BYTEALTO DIRECCION lista2
133         PUSH R0
134         MOVL R0, 3
135         MOVH R0, 0
136         PUSH R0
137         CALL listaValida
138         
139
140         MOVL R1, BYTEBAJO DIRECCION valida2
141         MOVH R1, BYTEALTO DIRECCION valida2
142         MOV [R1], R0
143
144
145     FIN

```

- ¿Qué valor hay en la posición de memoria 940h al final de la ejecución del programa? Responder en **hexadecimal**.

1

- ¿Qué valor hay en la parte alta del bus interno en el cuarto paso de ejecución de la segunda instrucción que se ejecuta? Responder en **hexadecimal**.

09h

- ¿Qué instrucción falta en el hueco de la línea 43?

BRC invalido

- ¿Qué instrucción/es falta/n en los huecos de la línea 124?

INC R7  
INC R7

- ¿Qué instrucción/es falta/n en los huecos de la línea 77?

INC R6  
MOV R2, [R6]

- Codifica la instrucción que se encuentra en la línea 100 del programa y expresa el resultado en **hexadecimal**.

F5F8h

- Reescribe la línea 3 usando el valor más pequeño que se pueda utilizar para este programa.

.PILA 16 (6 10h)

- ¿Cuál es el primer valor que se copia desde el bus interno al registro MDR? Responder en **hexadecimal**.

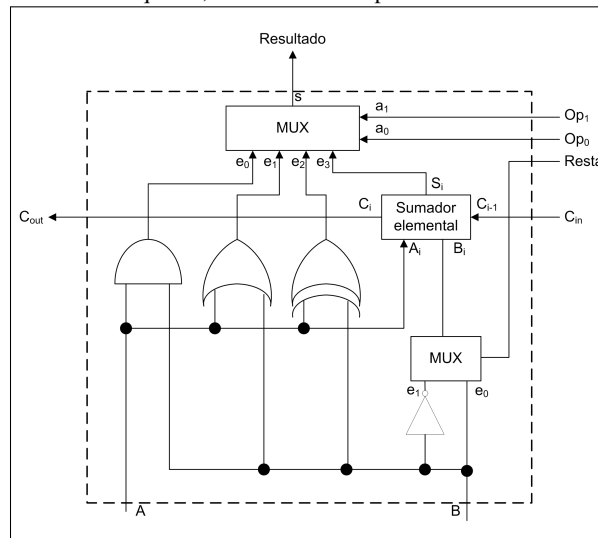
0900h

- ❑ Si cuando usamos la nomenclatura 'A' nos estamos refiriendo al código ASCII asignado a la letra A mayúscula, ¿Qué carácter ASCII obtendremos después de realizar la siguiente operación lógica?

$$1 \left( ('A' \text{ xor } 'a') \text{ xor } 'p' \right) + 1$$

'Q'

- ❑ En la figura se muestra el interior de una ALU de 1 bit. Completar las conexiones y elementos que faltan en el esquema. Rotular y dibujar de manera clara. En caso de duda en el esquema, se dará la respuesta como errónea.



- ❑ ¿Cuál es la codificación UTF-8 del carácter Unicode U+1A3C? Expresa el resultado en **hexadecimal**.

E1 A8 BC

- ❑ ¿Qué número representa el código hexadecimal 42 3D 80 00 si se interpreta utilizando el convenio IEEE-754 de números reales en coma flotante? Responder en **decimal**.

47,375

- ❑ Se quiere representar la cantidad  $-2^{-127} * (1 + 1/16)$  utilizando el estándar IEEE-754 de representación de números reales en coma flotante ¿Qué código habría que utilizar? Responder en hexadecimal.

80 44 00 00

- ❑ Para  $n=5$  bits llamamos  $\text{Max}(Z)$  al número más grande que se puede representar en exceso a Z central y  $\text{Max}(c2)$  al número más grande que se puede representar en complemento a 2. Si en un restador binario hacemos la operación  $\text{Max}(z) - \text{Max}(c2)$  ¿qué resultado obtendremos interpretado tanto en complemento a 2 como en exceso a Z central?

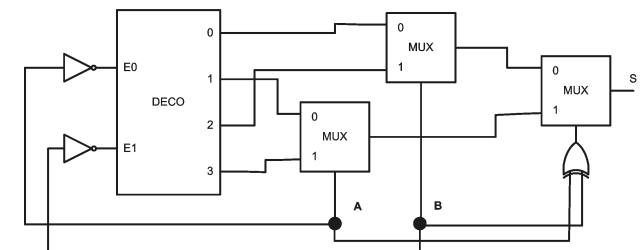
Compl. a 2 = -16

Exceso a Z = 0

- ❑ En una ALU de 5 bits similar a la de la CPU elemental se introduce en la entrada A el dato -2 codificado en signo-magnitud y en la entrada B el -2,5 usando una codificación de coma fija con 2 bits para la parte fraccionaria y codificando el signo en complemento a 2. Además, se selecciona la operación de suma. ¿Cuál es el resultado de la operación y el estado de los flags resultante? Dar el resultado en **decimal** interpretado en complemento a 2.

Resultado: 8 Z=0 C=1 O=1 S=0

- ❑ ¿Cuál es la expresión de la función lógica (S) entre las entradas A y B que implementa el circuito de la figura?

 $S = \bar{A} B$ 

- ❑ En una ALU de 5 bits similar a la de la CPU elemental se introduce en la entrada A el código 09h y en la entrada B el valor -12 codificado de complemento a 2. Después, 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**.

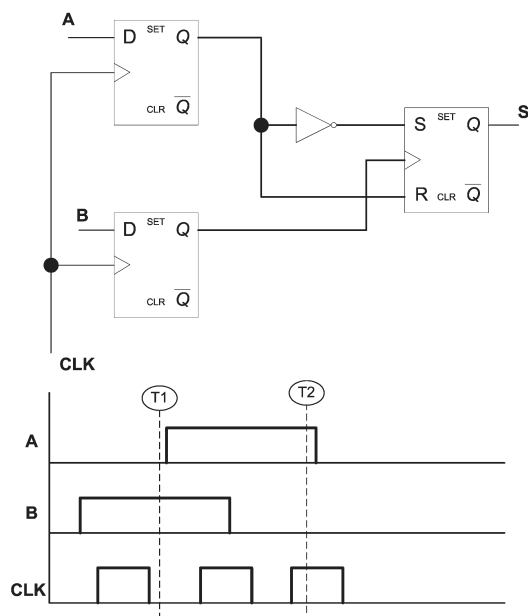
Resultado: 15h Z=0 C=1 O=1 S=1

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. Las memorias de RAM estática no necesitan refresco, por lo tanto, mantienen su información aunque se apague el ordenador.
2. Las memorias DRAM utilizan como celda básica un condensador que pierde la información almacenada si no se lee periódicamente.
3. Los decodificadores que se utilizan para construir los chips de memoria son secuenciales porque ante la misma combinación de entrada al chip (dirección) se pueden tener distintas salidas (datos) en función de lo que haya almacenado en cada momento.
4. En un chip de memoria 1024x8 el total de líneas de entrada o salida utilizadas para cualquier menester es de 18.

2

Si los biestables de la figura inicialmente tienen todos el valor 0 ¿Cuál será el valor de la salida S en los instantes T1 y T2?



T1=1 T2=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. Una instrucción es una operación básica que la CPU sabe realizar por diseño.
2. Para facilitar la ejecución de programas, las CPUs utilizan el registro contador de programa para almacenar el código de la siguiente instrucción a ejecutar.
3. El número de señales de control disponibles en una CPU es fijo. No es posible aumentarlo sin reconstruir la CPU.
4. Cuando se accede a un operando utilizando el modo de direccionamiento inmediato **NO** se activan las señales de LEER o ESCRIBIR.

1, 3 y 4

Se tiene el siguiente fragmento de código para la CPU teórica:

```
1 bucle:
2   DEC R1
3   COMP R1, R0
4   BRNS bucle
5   RET
```

Se sabe que antes de ejecutarlo R0 valía 5 y que la instrucción BRNS se ha ejecutado tres veces antes de que se ejecutase la instrucción RET. ¿Cuánto valía R1 antes de empezar a ejecutar el código? (Responder en **hexadecimal**)

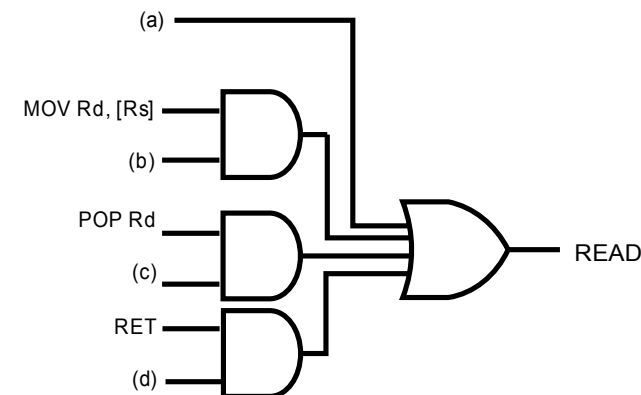
7

Se tiene una CPU análoga a la CPU teórica gobernada por un reloj que tiene una frecuencia de 1 GHz. Se sabe que, antes de ejecutar el fragmento de código mostrado a continuación, en la posición de memoria 267h se encuentra almacenado el valor 5. ¿Cuánto tarda en ejecutar el fragmento de código? Indicar las unidades de la respuesta.

```
1   MOVL R0, 3
2   MOVH R0, 0
3   MOVL R1, 67h
4   MOVH R1, 2h
5   MOV R1, [R1]
6   bucle:
7   DEC R1
8   COMP R0, R1
9   BRZ bucle
```

36 nanosegundos

La figura muestra una parte del circuito de una UC cableada que genera las mismas señales de control que las vistas en el caso de la CPU teórica. ¿Qué mnemónicos o pasos faltan en (a), (b), (c) y (d)?



a) = T1 b) = T4 c) = T4 d) = T4

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 16 bits inferiores de la palabra de control).



En un instante dado la palabra de control que se genera es 8500h. ¿Qué instrucción se está ejecutando? (Responder con el **mnemónico**)

JMP R1 o CALL R1