

Todas las preguntas tienen la misma puntuacin, salvo indicacin expresa de lo contrario. Las respuestas errneas no restan puntos.

- ☐ Se ha escrito un programa que tiene por cometido copiar un texto que está almacenado en una dirección de la memoria a otra dirección de memoria, pero “dándole la vuelta” durante la copia. Es decir, si el texto original era por ejemplo “HOLA”, el resultado una vez copiado será “ALOH”. La cadena origen llevará un caracter nulo (ASCII 0) al final para indicar dónde termina.

La “copia hacia atrás” la realiza un procedimiento llamado AL\_REVES. Este procedimiento recibe dos parámetros que el procedimiento principal debe apilar antes de llamarle y que son:

- La dirección de memoria donde está la cadena a copiar
- La dirección de memoria donde debe dejarse la cadena copiada “al revés”.

El procedimiento simplemente prepara dos registros, uno de ellos apuntando al final de la cadena que hay que copiar y el otro apuntando al principio de la dirección destino, y va haciendo la copia mientras decreuenta el primer puntero e incrementa el segundo.

Para saber en qué dirección termina la cadena origen, hace uso de otro procedimiento llamado CUANTAS\_LETRAS. Este procedimiento simplemente recibe la dirección de una cadena y retorna en el registro R0 cuántas letras tiene esa cadena. Por ejemplo, ante la cadena "HOLA", 0 retornaría 4 (ya que el terminador no lo cuenta). El código fuente de este procedimiento, aunque es trivial, no se muestra. Una vez conocido el número de letras que tiene el texto origen, se suma este número menos 1 a la dirección en que empieza el texto, y así se tiene la dirección de la última letra del mismo.

Seguidamente se muestra parte del código de dicho programa, en el que se han tapado algunas líneas por cuyo contenido se preguntará más adelante.

```

1 ORIGEN [ ]
2 INICIO start
3 .PILA [ ]
4 .DATOS
5 texto VALOR "Prueba",0
6 result VALOR " ",0
7 .CODIGO
8 PROCEDIMIENTO CUANTAS_LETRAS
9 ; El código de este procedimiento se omite
10 ; para mayor brevedad. Véase el enunciado.
11 FINP
12
```

```

13 PROCEDIMIENTO AL_REVES
14 PUSH R6
15 MOV R6, R7
16 PUSH R0
17 PUSH R1
18 PUSH R2
19 PUSH R3
20
21 INC R6 ; Recuperar parámetros
22 INC R6
23 MOV [ ], [R6]
24 INC R6
25 MOV [ ], [R6]
26 ; Calcular longitud de cadena origen
27 PUSH R2
28 CALL CUANTAS_LETRAS
29 INC R7
30 ; En R0 tenemos el número de letras de la cadena
31 ; Hacer apuntar R2 a la última letra
32 [ ]
33
34
35 ; Hacer la copia
36 ; (repetir tantas veces como indique R0)
37 bucle:
38 MOV R3, [R2]
39 MOV [R1], R3
40 [ ]
41 [ ]
42 DEC R0
43 BRNZ bucle
44
45 POP R3
46 POP R2
47 POP R1
48 POP R0
49 POP R6
50 RET
51 FINP
52
53 start:
54 ; Probamos el procedimiento con la cadena texto
55 MOVL R0, BYTEALTO DIRECCION texto
56 MOVL R0, BYTEBAJO DIRECCION texto
57 PUSH R0
58 MOVL R0, BYTEALTO DIRECCION result
59 MOVL R0, BYTEBAJO DIRECCION result
60 PUSH R0
61 CALL AL_REVES
62 [ ]
63 [ ]
64 FIN
```

Cuando está a punto de ejecutarse la primera instrucción de la rutina AL\_REVES, el simulador de la CPU nos permite averiguar los siguientes datos. Algunos de estos datos son necesarios para responder a las preguntas que siguen.

- R7 vale 0292h
- En la cima de la pila hay un 0273h
- Justo debajo de la cima de la pila hay un 0237h

— ¿Qué valor se utilizó en la directiva ORIGEN del programa?

— ¿Qué valor se utilizó en la directiva PILA del programa?

— Si la rutina CUANTAS\_LETRAS, que no se muestra, empieza en la forma estándar, esto es, con PUSH R6 seguido de MOV R6, R7 ¿qué valor tomará el registro R6 tras el MOV R6, R7?

— ¿Qué falta en las líneas 23 y 25 en las que se recuperan los parámetros?

Línea 23:

Línea 25:

— Al retorno de la función CUANTAS\_LETRAS, el registro R0 contiene el número de letras de la cadena origen. Esta información se usa para actualizar el registro que apunta a esta cadena, de modo que pase a apuntar a la última letra de la misma. Esto lo hacen las instrucciones que faltan en la línea 32 y la siguiente ¿qué instrucciones serían éstas?

— La línea 40 y la siguiente actualizan los punteros a las cadenas. Escribe qué instrucciones se han ocultado en estas líneas.

— Tras el CALL AL\_REVES (línea 61) aparecen dos instrucciones ocultas ¿cuáles serían?

- En un instante la CPU está ejecutando el MOV de la línea 39 y en ese momento R3 vale 0072h ¿cuál es el valor en ese momento de R0?

- ☐ Se diseña un formato de coma flotante análogo al del estándar IEEE-754, pero que usa sólo 16 bits en lugar de 32. Estos bits se distribuyen en la siguiente forma: 1 para el signo, 6 para el exponente y 9 para la mantisa. Al igual que en IEEE-764 se reservan los casos “todo ceros” y “todo unos” del exponente para números “denormales” y “especiales”. En los restantes casos, el exponente se codifica en exceso a Z central menos uno.

- ¿Qué código tendría el número -1,5 en el formato anterior? Responde en hexadecimal.

- Si en los números denormales de este formato el exponente implícito es -30 ¿Cuál es el número más pequeño representable en este formato? (es decir, el positivo más cercano a cero) ¿Qué código (hexadecimal) tiene?

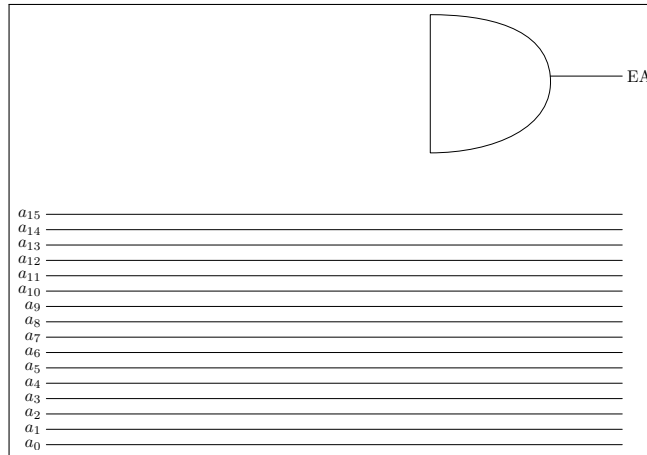
Valor:	Código:
--------	---------

- ☐ Al enviar el dato 3 al dispositivo *luces* ¿cuántas bombillas se encenderán?

- ☐ Se tiene un periférico *pantalla* para la CPU elemental, pero que no es idéntico al estudiado en teoría. Este nuevo periférico tiene una pantalla en la que caben más letras, en concreto tiene 10 filas y en cada fila caben 20 caracteres. Además, tiene un registro de control que funciona como en el periférico pantalla explicado en teoría.

- ¿Cuántas líneas de dirección requiere el interfaz de este periférico?

- Si queremos que la primera posición de esta pantalla aparezca “mapeada” en la dirección B800 del espacio de direcciones, dibuja cómo tendría que ser el circuito de activación y su conexión al bus de direcciones.



- El siguiente fragmento de código imprime en esa pantalla una letra:

```

1  MOVH  R0, 0B8h
2  MOVL  R0, 2Ah
3  MOVH  R1, 0Fh
4  MOVL  R1, 78h
5  MOV   [R0], R1

```

- ¿En qué fila y columna aparecerá la letra? Recuerda que comienzan a numerarse desde cero.

FILA:	COLUMNA:
-------	----------

- ¿Qué letra y en qué colores aparecerá?

LETRA:
COLORES:

- ☐ ¿Cuál o cuáles de las siguientes afirmaciones acerca de la ALU de la CPU elemental son **falsas**?

- A) Siempre que se necesite incrementar en 1 un registro de la CPU, es la ALU quien debe realizar la operación.  
 B) La ALU puede operar indistintamente con números naturales o enteros en complemento a 2.  
 C) El bit de préstamo en una resta coincide con el acarreo generado por la última alu de 1 bit.  
 D) La ALU, además del resultado, genera cinco bits de estado (ZF, OF, CF, SF e IF).

- ☐ Explica en qué consiste la “instalación” de una rutina de servicio para una interrupción.

- ❑ Se pretende confeccionar una nueva instrucción para la CPU teórica, que permite efectuar la operación lógica NOR sobre dos registros de la CPU, dejando el resultado en un una posición de memoria indicada por otro registro. La operación NOR consiste en la negación (NOT) del resultado de OR. El mnemónico de la nueva instrucción es:

NOR [Ri], Rd<sub>1</sub>, Rd<sub>2</sub> ; [Ri] ← Rd<sub>1</sub> NOR Rd<sub>2</sub> a

Para implementar esta operación, es necesario realizar primero la operación OR entre los dos registros de datos, dejando el resultado en un registro temporal, para después negar (NOT) el contenido de ese registro. Se ha decidido utilizar como registro temporal MDR y como “truco” para negarlo, el hacer una operación XOR entre ese registro y el dato FFFh.

La siguiente tabla muestra las señales de control necesarias en cada paso para implementar esta instrucción, tomando como ejemplo la instrucción NOR [R0], R1, R2:

Paso	Señales
4	R1-IB, IB-TMPE
5	R2-IB, OR, ALU-TMPS
6	
7	MDR-IB, TMPE-SET, XOR, ALU-SR, ALU-TMPS
8	TMPS-IB, IB-MDR
9	
10	FIN

— ¿Qué señales se activan en el paso 6?

— ¿Qué señales faltan en el paso 9?

— Si el reloj de la CPU opera a 50MHz ¿cuánto tiempo tardaría en ejecutarse la instrucción NOR [R0], R1, R2? Puedes dejar indicada la fórmula, en lugar del resultado numérico, pero debes indicar las unidades de tiempo de la respuesta.

- ❑ La CPU está ejecutando el siguiente código:

```

1  atras:
2      MOVH R0, 0
3      MOVL R0, 'A'
4      MOVH R1, 0
5      MOVL R1, 18h
6  bucle:
7      MOV [R3], R0
8      INC R0
9      DEC R1
10     BRNZ bucle
11     JMP  atras

```

En un instante dado, está a punto de ejecutar el Paso 4 de una de las instrucciones anteriores. A priori no sabemos cuál de ellas. Tampoco sabemos cuántas veces ha repetido ya el bucle. Sin embargo, tenemos el valor de algunos registros de la CPU en el instante mencionado, que se recogen en la tabla siguiente.

Registro	Valor	Registro	Valor
R0		R4	0000
R1	0015	R5	0000
R2	FFFF	R6	0304
R3	B800	R7	0302
PC	0246	IR	F5FC

— ¿Cuál sería el valor de R0 que falta en la tabla anterior?

— ¿Cuál será el próximo valor que tendrá el registro PC cuando la instrucción en curso llegue a su último paso (es decir, a la señal FIN)?

— Codifica la última instrucción del listado anterior. Expresa el código en hexadecimal.

— En el instante del que se habla en el enunciado, se produce una interrupción de vector 3. Asumiendo que el bit IF está a 1, ¿cuál será el próximo valor (hexadecimal) que aparecerá en el bus de direcciones de sistema?

— Si el código antes mostrado tiene por objetivo sacar las letras de la 'A' a la 'Z' en la esquina superior izquierda de la pantalla ¿en qué dirección estaría mapeado el interfaz de vídeo?