Apellidos: Nombre:

DNI:

1. Enumera los tres componentes fundamentales de un computador, indicando muy brevemente la función de cada uno de ellos.

   El procesador: ejecuta las instrucciones y genera las señales de control necesarias para el funcionamiento del computador.

   La memoria principal: almacena los datos e instrucciones del programa en ejecución.

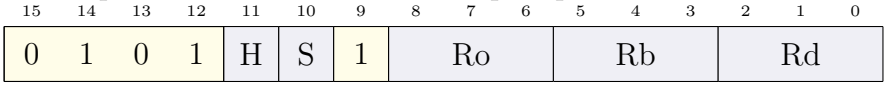
   La entrada/salida: comunica el ordenador con el mundo exterior.

2. Indica en qué fases de la ejecución de la instrucción «`str r4, [r0]`», el procesador interactúa con la memoria y describe brevemente tales interacciones.

   Primero, en la fase de lectura o adquisición de la instrucción, el procesador envía a la memoria el contenido del contador de programa como dirección, indicando que se va a realizar una lectura de información. La memoria responde enviando el contenido de tal dirección, que el procesador recibe y almacena.

   Segundo, en la fase de ejecución de la instrucción, en concreto en la subfase de escritura de resultados, el procesador envía los valores que previamente había leído del registro «`r4`» y del registro «`r0`», y genera las señales de control para indicar a la memoria que escriba el primero (dato) en la dirección dada mediante el segundo. La memoria almacena el dato en la dirección y concluye la ejecución de la instrucción.

3. Codifica en binario y en hexadecimal la instrucción «**`ldrsh`** `r0, [r1, r3]`» sabiendo que el formato de instrucción utilizado para dicha instrucción es el que aparece a continuación.



H y S Identifican la instrucción:

H	S	Instrucción
0	0	«**`strh`** `rd, [rb, ro]`»
0	1	«**`ldrsb`** `rd, [rb, ro]`»
1	0	«**`ldrh`** `rd, [rb, ro]`»
1	1	«**`ldrsh`** `rd, [rb, ro]`»

Ro Registro desplazamiento.

Rb Registro base.

Rd Registro fuente/destino.

El contenido en binario de cada uno de los campos, de izquierda a derecha, será:

- `0101`, que es fijo y nos indica de qué tipo de instrucción se trata.
- `11` en el campo `HS`, pues se trata de una instrucción «**`ldrsh`**».
- `1`, que es fijo y forma parte del código de operación.
- `011` en el campo `Ro`, que codifica el registro desplazamiento, «`r3`».
- `001` en el campo `Rb`, que codifica el registro base, «`r1`».
- `000` en el campo `Rd`, que codifica el registro destino, «`r0`».

Juntando los campos se tiene:

Que en hexadecimal es 0x5EC8.

4. Describe el modo de direccionamiento de cada uno de los operandos de la instrucción «**`lsl`** `r4, r0, #4`».

   La instrucción propuesta tiene tres operandos, dos fuente y uno destino, especificado cada uno de ellos con su modo de direccionamiento.

   Los operandos fuente son «`r0`» y «`#4`». El primero utiliza el modo directo a registro, donde el valor se encuentra en el registro «`r0`». El segundo usa el modo inmediato, en que el valor constante 4 se codifica en la propia instrucción.

   El operando destino, «`r4`», también utiliza el modo directo a registro. En este caso se indica que el resultado de desplazar el contenido de «`r0`» 4 veces a la izquierda se va a almacenar en el registro «`r4`».

5. Dado el siguiente programa en ensamblador Thumb de ARM, describe su funcionamiento comentando cada una de sus líneas y anota a continuación el contenido final de los registros «`r0`» y «`r2`» y los contenidos de las direcciones de memoria `0x2007 0004` (etiquetada como «`vec`») y `0x2007 0018`.

```
1         .data
2  tam:   .word 6
3  vec:   .word 1, 2, 3, 4, 5, 6
4
5         .text
6         ldr r7, =tam            @ r7 apunta al comienzo del área de datos
7         add r0, r7, #4          @ y r0 al vector (la dirección de tam + 4)
8         ldr r1, [r7]            @ + Se guarda en r1 el número de elementos - 1 y
9         sub r1, r1, #1          @ | si el resultado es <= 0, termina
10        ble fin                 @ + pues el vector tiene a lo sumo 1 elemento.
11        lsl r1, r1, #2          @ Multiplica número elementos - 1 por 4 (enteros)
12        add r1, r0, r1          @ Dirección de la última posición del vector.
13        ldr r3, [r0]            @ Preserva el primer elemento en r3
14 bucle: ldr r2, [r0, #4]        @ + Lee un elemento de r0 + 4 y lo escribe en
15        str r2, [r0]            @ +  la dirección anterior, dada por r0.
16        add r0, r0, #4          @ Incrementa r0 para pasar al siguiente elemento
17        cmp r0, r1              @ + Si r0 es igual a r1, se está en la última
18        bne bucle               @ + posición y termina el bucle. Si no, continúa.
19        str r3, [r0]            @ Escribe el primer elemento del vector original,
20                                @   en la última posición del vector.
21 fin:   wfi
22
23        .end
```

El programa realiza una rotación de los elementos del vector, de tal manera que cada valor pasa a ocupar la posición del anterior y el primero va a ocupar la última.

En los registros y direcciones de memoria indicadas habrán los siguientes valores una vez se haya ejecutado el programa.

`r0`	0x20070018
`r2`	6

`0x2007 0004`	2
`0x2007 0018`	1