DNI:	Apellidos:	Nombre:

Cuadrícula de respuestas:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
D✓	A✓	C✓	B✓	B✓	C✓	B✓	B✓	D✓	B✓	B✓	B✓	A✓	A✓	B✓	B✓	B✓	B✓	B✓	C✓

Bien	Mal	NC

Normas

1. La duración del examen será de 1 hora y media.
2. No está permitido:
   a) Abandonar el aula sin entregar el examen.
   b) Utilizar cualquier tipo de documentación.
   c) Utilizar calculadora.
3. El examen se calificará teniendo en cuenta únicamente las respuestas anotadas en la cuadrícula de respuestas. Cada respuesta correcta sumará 0,5 puntos. Cada respuesta errónea restará 0,5/3 puntos. Las preguntas no contestadas no se tendrán en cuenta.

1. Dada la instrucción «`cmp r0, r1`» en Thumb ARM, ¿qué flag o flags del registro de estado (CPSR) se ven afectados y cómo?

   a) Los flags N, Z, C (carry) y V (overflow) se actualizan en base a la suma de `r0` y `r1`.

   b) Solo el flag Z (cero) si `r0` y `r1` son iguales.

   c) Solo el flag N (negativo) si el resultado es negativo.

   d) Los flags N, Z, C (carry) y V (overflow) se actualizan en base a la resta de `r0` y `r1`.

2. Implementa una función en ensamblador Thumb ARM que reciba dos números de 32 bits sin signo en `r0` y `r1`, y retorne el mayor de ellos en `r0`.

   a)

```
max:    cmp r0, r1
        bhi g
        mov r0, r1
g:      mov pc, lr
```

   b)

```
max:    sub r2, r0, r1
        bmi l
        mov r0, r1
l:      mov pc, lr
```

   c)

```
max:    cmp r0, r1
        bls l
        mov r1, r0
l:      mov pc, lr
```

   d)

```
max:    cmp r0, r1
        bgt g
        mov r0, r1
g:      mov pc, lr
```

3. Si se activa una interrupción en un sistema ARM Cortex-M, ¿qué ocurre automáticamente con los registros del procesador antes de que se ejecute la rutina de tratamiento de la interrupción?

   a) El procesador guarda todos los registros de propósito general y el registro de estado (CPSR) en una memoria especial de acceso rápido.

   b) El procesador guarda automáticamente todo el contexto del procesador en la pila principal.

   c) El procesador solo guarda los registros `r0` a `r3`, `r12`, `lr` y `pc` y el registro de estado (CPSR) en la pila.

d) El programador debe guardar manualmente todos los registros al inicio de la rutina de tratamiento de la interrupción.

4. Un procesador sin caché accede a memoria de datos con una latencia de 10 ciclos. Si un programa ejecuta 100 instrucciones, de las cuales 20 son accesos a memoria (lectura o escritura de datos) y el resto son operaciones ALU o de control de flujo que no acceden a memoria en su fase de ejecución, ¿cuántos ciclos de reloj se requieren para ejecutar el programa completo, asumiendo 1 ciclo por instrucción ALU/control y 10 ciclos por acceso a memoria? (No tengas en cuenta el acceso a las propias instrucciones, ya que se realiza a través de otra memoria.)

   a) 100 ciclos.

   b) 280 ciclos.

   c) 200 ciclos.

   d) 300 ciclos.

5. Un dispositivo de entrada/salida opera en modo DMA (Direct Memory Access). ¿Qué implicación tiene esto para la CPU en términos de comportamiento del interlocutor?

   a) El dispositivo DMA solo puede transferir datos desde/hacia registros del procesador.

   b) La CPU solo inicia y finaliza la transferencia, delegando el movimiento de datos a la memoria al controlador DMA.

   c) La CPU debe gestionar cada byte de la transferencia, lo que aumenta su carga.

   d) El DMA reduce la productividad al introducir un cuello de botella.

6. Después de la ejecución de la instrucción «`lsr r3, r0, #1`» del programa mostrado en el Anexo A, ¿qué valor contendrá el registro `r3`?

   a) 5.

   b) 7.

   c) 6.

   d) 13.

7. ¿Por qué es fundamental que la instrucción «`push {lr}`» se realice al inicio de una subrutina que, a su vez, realiza una llamada a otra subrutina?

   a) Para cumplir con las convenciones de llamada del sistema operativo.

   b) Para guardar la dirección de retorno de la subrutina actual antes de que la siguiente llamada sobrescriba `lr`.

   c) Para liberar el registro `lr` y poder usarlo para otros cálculos dentro de la subrutina.

   d) Para optimizar el uso de la caché de instrucciones.

8. Un procesador ARM puede operar en modo *Little-endian* o *Big-endian*. Si un programa intenta acceder a una media palabra (16 bits) desde la dirección `0x0000 2000` en un sistema configurado como *Big-endian*, y la memoria contiene los bytes `0xAA`, `0xBB`, `0xCC` y `0xDD` a partir de la dirección `0x0000 2000`, ¿cuál es el valor que se escribirá en el registro de destino?

   a) `0xCCDD`.

   b) `0xAABB`.

   c) `0xDDCC`.

   d) `0xBBAA`.

9. Considera un microcontrolador Cortex-M que dispone de un registro de control de GPIO, «`GPIOC_MODER`», mapeado en la dirección `0x40020800 ,` donde los bits $[1:0]$ controlan el modo del pin C_0 (00: entrada digital, 01: salida digital, 10: función alternativa, 11: salida analógica). Si se ejecuta el siguiente código:

```
1        ldr r0, =0x40020800   @ Dirección de GPIOC_MODER
2        ldr r1, [r0]
3        mov r2, #0x01
4        orr r1, r1, r2
5        str r1, [r0]
```

   ¿Cuál será el efecto final en el pin C_0 del puerto GPIO después de la ejecución de este código?

   a) Se configura el pin C_0 como función alternativa o salida analógica, en función del valor original del registro «`GPIOC_MODER`».

   b) Se configura el pin C_0 como salida analógica, independientemente del valor original del registro «`GPIOC_MODER`».

   c) Se configura el pin C_0 como salida digital, independientemente del valor original del registro «`GPIOC_MODER`».

   d) Se configura el pin C_0 como salida digital o analógica, en función del valor original del registro «`GPIOC_MODER`».

10. ¿Cuál es la principal ventaja de utilizar interrupciones sobre la consulta de estado (*polling*) para sincronizar la E/S en un sistema donde los eventos de E/S son esporádicos y de baja frecuencia?

    a) Simplificación del código de manejo de dispositivos, ya que no se necesita un bucle de espera.

    b) Reducción significativa del consumo de ciclos de CPU, permitiendo que la CPU realice otras tareas.

    c) Mayor determinismo en la respuesta a eventos de E/S.

    d) Mayor productividad de la E/S, ya que los datos se transfieren más rápidamente.

11. ¿Cuál es la principal razón para la existencia de la jerarquía de memoria en un sistema computacional moderno?

    a) Simplificar la programación al proporcionar una única interfaz de memoria.

    b) Maximizar el rendimiento del acceso a datos, compensando la brecha entre la velocidad del procesador y la memoria principal.

    c) Minimizar el coste total del sistema sin comprometer la capacidad de almacenamiento.

    d) Reducir el consumo de energía general del sistema.

12. Si un procesador ARM añade una nueva instrucción como «`rbit`» (reverse bits) a su juego de instrucciones, ¿cuál de las siguientes características de la arquitectura del juego de instrucciones (ISA) es la que se ve directamente modificada?

    a) Los tipos de datos soportados por defecto.

    b) El conjunto de operaciones (opcodes) disponibles.

    c) El conjunto de registros visibles.

    d) Los modos de direccionamiento de los operandos.

13. En un sistema con múltiples dispositivos que pueden generar interrupciones, ¿qué mecanismo se utiliza comúnmente para determinar qué dispositivo ha generado la interrupción y cómo se priorizan si ocurren simultáneamente?

    a) Un controlador de interrupciones externo (el NVIC en el ARM Cortex-M) mapea las fuentes de interrupción a vectores de interrupción y gestiona su priorización.

    b) El procesador consulta el estado de todos los dispositivos en un bucle después de cada interrupción.

    c) Cada dispositivo tiene un pin de interrupción dedicado conectado directamente a la CPU.

    d) La CPU determina el origen de la interrupción basándose en el valor del Program Counter en el momento de la interrupción.

14. Se ha diseñado un nuevo microprocesador ARM que implementa la ISA Thumb-2 utilizando una tubería (pipeline) de 5 etapas en lugar de 3. Este cambio afecta principalmente a:

    a) La organización, pues se refiere a los detalles internos de implementación y rendimiento del procesador.

    b) Ambos, arquitectura y organización, de forma equitativa.

    c) La arquitectura (ISA), ya que se modifica el comportamiento visible del procesador.

    d) Ninguno, ya que la tubería es una característica de la microarquitectura, no de la organización.

15. Considere la instrucción «`ldr r0, [r1, r2]`». ¿Qué modo de direccionamiento se utiliza para calcular la dirección efectiva del operando en memoria y cómo se forma esta dirección?

    a) Direccionamiento indirecto con registro de desplazamiento escalado, donde la dirección es $r1 + (r2 \times 4)$.

    b) Direccionamiento indirecto con registro de desplazamiento, donde la dirección es $r1 + r2$.

    c) Direccionamiento inmediato con desplazamiento, donde la dirección vendrá dada por $r1 +$ desplazamiento.

    d) Direccionamiento indirecto, donde `r0` contiene la dirección.

16. ¿Qué valor final tendrá el registro r4 cuando el programa mostrado en el Anexo A alcance la instrucción «`wfi`»?

    a) -1.

    b) 1.

    c) 0.

    d) Ninguno, el programa se detiene antes de asignar valor a `r4`.

17. Un sistema empotrado, cuyo reloj funciona a 10 MHz, requiere que un bit específico del puerto de salida (bit 2 de `0x4001080C`) se ponga a '1' durante exactamente 50 microsegundos y luego vuelva a '0', con una precisión de microsegundos. ¿Cuál es la estrategia más eficiente y precisa para lograr esto en ensamblador Thumb ARM?

    a) Utilizar un bucle de retardo («`nop`»s o «`sub r0, #1`») calibrado manualmente.

    b) Configurar un temporizador para generar una interrupción después de 50 μs para apagar el bit, y encenderlo justo antes de iniciar el temporizador.

    c) Usar un bucle de retardo («`nop`»s) y medir el tiempo en el que la salida está a '1' con un osciloscopio para ajustar el número de «`nop`»s.

    d) Depender del sistema operativo para programar un retardo preciso.

18. La arquitectura ARM soporta enteros de 8, 16 y 32 bits. Si se realiza una operación «`add r0, r1, r2`» donde `r1` y `r2` contienen valores de 32 bits, y el resultado excede $2^{31} - 1$ (para un entero con signo) o $2^{32} - 1$ (para un entero sin signo), ¿cómo maneja la arquitectura ARM esta situación?

    a) El procesador genera automáticamente una excepción de *overflow*.

    b) El resultado se trunca a 32 bits y los flags C (*carry*) y V (*overflow*) del registro de estado (CPSR) se actualizan para indicar si hubo desbordamiento sin signo o de signo.

    c) El resultado se satura al valor máximo representable de 32 bits.

d) La operación se recalcula utilizando registros de 64 bits de forma transparente.

19. Durante la ejecución de una instrucción de salto condicional como «`bne`» (*Branch if Not Equal*), ¿cómo se establece la condición para el salto y qué componente del procesador es crucial para ello?

a) El contador de programa (PC) se actualiza directamente con la dirección de destino.

b) Durante la ejecución de una instrucción previa, la ALU actualiza los flags del registro de estado (CPSR), que serán luego evaluados por la unidad de control.

c) El decodificador de instrucciones lee el opcode (código de operación) y fuerza el salto.

d) La memoria de datos proporciona un valor booleano que determina el salto.

20. Un sistema de red tiene una latencia de 10 ms para enviar un paquete y una capacidad de enlace de 1 Gbps. Si se envía un archivo de 10 MB, ¿cuál es el tiempo mínimo teórico para transferir el archivo, considerando solo el tiempo de transmisión y la latencia inicial de un paquete?

a) 80 ms.

b) 10 ms.

c) 90 ms.

d) 100 ms.

A. Programa ARM Thumb M0

```
1         .data
2  x:     .word 10
3  y:     .word 3
4
5         .text
6  main:
7         ldr r0, =x
8         ldr r0, [r0]
9
10        ldr r1, =y
11        ldr r1, [r1]
12
13        add r0, r0, r1
14        sub r2, r0, #5
15        lsr r3, r0, #1
16        cmp r2, r3
17        bge dst
18        mov r4, #0
19        b   fin
20 dst:   mov r4, #1
21 fin:   wfi
```