Facultad de Informática
Universidad Complutense de Madrid

# Examen final - Fundamentos de Computadores II
## 19 de Mayo de 2023

Una UART (*Universal Asynchronous Receiver/Transmitter*) es un circuito utilizado para transmitir y recibir datos serie. Su funcionamiento es bien sencillo: Para enviar un *Byte* completo, el emisor envía un 0, seguido de los 8 bits de datos (de menos a más significativo) y por último envía un 1. El receptor, por su parte, captura esta información esperando a recibir un 0, luego los 8 bits, y finalmente el 1 que indica el fin.

Se dispone de un microcontrolador RISC-V que se quiere utilizar como emisor UART. A fin de enviar los datos, se ha conectado la UART a la dirección de memoria `DIR_BUS`. Los bits se escriben **de uno en uno** en esa dirección, y la UART se encarga de transmitirlos. Se proporciona un pseudocódigo en RISC-V con una función `enviar_byte` que implementa la funcionalidad de transmisión a la UART.

```
.text
enviar_byte:
    sw zero, 0(a1)          #escribo el primer 0 en la dirección indicada
    addi t1, zero, 8        #inicializo iterador
for_bits:
    beq t1, zero, fin_for
    andi t2, a0, 1          #copio el bit menos significativo en t2
    sw t2, 0(a1)            #escribo el bit menos significativo
    srli a0, a0, 1          #desplazo para tener el siguiente bit
    addi t1, t1, -1         #actualizo iterador
    j for_bits
fin_for:
    addi t1, zero, 1
    sw t1, 0(a1)            #escribo el 1 de terminación
    ret                     #termino la función
```

Responda a las siguientes preguntas:

1. **ASM [0.5pt]** La función proporcionada. ¿Está escrita correctamente? ¿Por qué? En caso contrario, ¿Qué código debemos añadir?

2. **ASM [0.25pt]** Supongamos que tengo en el registro s1 el byte que quiero enviar, y en s4 la dirección de la UART a la cual lo envío. Escriba el código necesario para llamar a la función `enviar_byte` respetando el estándar de llamadas RISC-V.

3. **ASM [0.25pt]** El bucle realiza la comprobación de terminación en la primera instrucción. Reescríbalo haciendo que la comprobación se realice al final, consiguiendo en el proceso que el bucle pase de 6 a 5 instrucciones.

4. **CPU [1pt]** Supongamos que disponemos de dos procesadores multiciclo con las siguientes características:

| Ciclos | A | B |
|---|---|---|
| `sw` | 4 | 5 |
| `beq` | 3 | 3 |
| `j` | 3 | 4 |
| `ret` | 3 | 4 |
| Aritmético-lógicas | 4 | 3 |

   Calcule, para los procesadores $A$ y $B$, el CPI para la ejecución original de la función `enviar_byte`.

5. **CPU [1pt]** Suponiendo que $A$ tiene una frecuencia de $100MHz$ y $B$ de $95MHz$, ¿Cuánto tiempo tarda cada uno en ejecutar la función? ¿Cuál es más rápido?

6. **CPU [1pt]** Manteniendo el mismo tiempo de ciclo, se puede mejorar el procesador $B$ con dos alternativas: Bajar los ciclos de `sw` de 5 a 3, o bajar los ciclos de aritmético-lógicas de 3 a 2. ¿Qué opción obtendría más mejora? Argumenta tu respuesta.

7. **CPU [2pt]** Supongamos que se ejecuta una iteración del bucle original (6 instrucciones) en el procesador segmentado RISC-V. El procesador tiene gestión de conflictos de datos por anticipación, predicción de salto no tomado, y escritura en banco de registros a mitad de ciclo, como el visto en clase. Realice el diagrama de ejecución señalando las paradas y anticipaciones si las hubiera. ¿Cuántos ciclos tarda en ejecutarse cada iteración del bucle?

8. **CPU [1pt]** Si nuestro procesador segmentado no dispusiera de la lógica necesaria para anticipar operandos (si bien escribe en el banco de registros en mitad de ciclo), y parara el pipeline ante un salto hasta conocer la dirección destino, ¿cuántos ciclos de penalización tendríamos en el diagrama de ejecución anterior? Calcule el CPI de la función completa en ambos casos.

9. **MEM [0.75pt]** Asumamos que para nuestro procesador tenemos una memoria principal de 4KB, y una caché de 64B de emplazamiento directo con bloques de 8B. Indique el formato de las direcciones para memoria caché.

10. **MEM [1.5pt]** Supongamos que la sección `.text` comienza en la dirección `0x200`. Realizamos una llamada a la función `enviar_byte` con `a1=0x438` (para mandar un byte a la UART conectada en esa dirección). ¿Cuántos fallos de caché se producen al ejecutar el código en el procesador multiciclo? Calcule cuál es la tasa de fallos. Téngase en cuenta que la caché es unificada para datos e instrucciones, y está vacía al comenzar.

11. **MEM [0.75pt]** Repetimos la llamada, pero esta vez con `a1=0x408`. Comente y razone en qué afecta este detalle a la tasa de fallos.