Facultad de Informática
Universidad Complutense de Madrid

Introduction to Computers II - Exam
May 19th, 2023

A UART (Universal Asynchronous Receiver/Transmitter) is a circuit used to transmit and receive serial data. Its behavior is quite simple: In order to send a full byte, the transmitter sends a 0, followed by the 8 data bits (from least to most significant) and finally it sends a 1. The receiver gets this information, starting with a 0, then the 8 data bits and finally the last 1.

We have a RISC-V microcontroller that will be used as a UART transmitter. In order to send the data, the UART has been connected to the **DIR_BUS** memory address. Bits are written one by one in that address and the UART handles the transmission. A RISC-V pseudocode is provided, with a **send_byte** function that implements the transmission functionality.

```
.text
send_byte:
    sw zero, 0(a1)          # first 0 is written in the given address
    addi t1, zero, 8        # loop initialized
for_bits:
    beq t1, zero, end_for
    andi t2, a0, 1          #least significant bit copied in t2
    sw t2, 0(a1)            # least significant bit written
    srli a0, a0, 1          # shift to access the next bit
    addi t1, t1, -1         # loop update
    j for_bits
end_for:
    addi t1, zero, 1
    sw t1, 0(a1)            # final 1 is written
    ret                     # return
```

Answer the following questions:

1. **ASM [0.5pt]** Is the provided function written correctly? Why? If not, how should we modify the code?

2. **ASM [0.25pt]** Let us assume that that byte to transmit is in s1 and the UART address to which it is send in s4. Write the code to call the **`send_byte`** function, meeting the RISC-V function call convention.

3. **ASM [0.25pt]** The loop checks the end condition in the first instruction. Rewrite it so that the comparison is performed at the end and the number of instructions of the loop is reduced to 5.

4. **CPU [1pt]** Let us assume that we have two multicycle processors with the following features:

Cycles	A	B
sw	4	5
beq	3	3
j	3	4
ret	3	4
Arithmetic-logic	4	3

   Calculate, for the A and B microprocessors, the CPI for the original execution of the **`send_byte`** function.

5. **CPU [1pt]** Let us assume that A has a frequency of 100MHz and B of 95MHz. How much time is taken by each of the processors to execute the function? Which one is faster?

6. **CPU [1pt]** Keeping the same cycle time, the B processor could be improved with two alternatives: Decrease the cycles of sw from 5 to 3, or decrease the cycles of the arithmetic-logic instructions from 3 to 2. Which of these options would produce a bigger improvement? Explain why.

7. **CPU [2pt]** Let us assume that one iteration of the original loop (6 instructions) is executed in the RISC-V pipelined processor. The processor uses forwarding to handle data hazards, not-taken branch prediction is used and the register file is written at the middle of the cycle. Provide the execution diagram indicating the pipeline stalls and forwarding operations (if any). How many cycles are needed to execute each of the loop iterations?

8. **CPU [1pt]** If the pipelined processor did not have the forwarding mechanism and it stalled the pipeline until the branch destination address is known, how many penalty cycles would there be in the previous execution diagram? Calculate the CPI of the whole function in both cases.

9. **MEM [0.75pt]** Let us assume that there is a 4-KB main memory, and a 64B direct-mapped cache, with 8B blocks. Indicate the cache memory address format.

10. **MEM [1.5pt]** Let us assume that the **`.text`** section starts in the 0x200 address. We call the **`send_byte`** function with a1=0x438 (in order to send one byte to the UART connected in that address). How many cache misses would happen if the code is executed in the multicycle processor? Consider that there is a single cache for data and instructions, which is empty at the beginning.

11. **MEM [0.75pt]** We make another call, but this time with a1=0x408. Reason the effect of this on the number of misses.