RA FS 21 Series 4

Alp Sari Eren, Sepehr Sameni, Abdelhak Lemkhenter

The fourth series has to be solved by Tuesday, 3. May 2021 at 3 p.m. and has to be uploaded to ILIAS. If questions arise, you can use the ILIAS forum at any time. Any problems should be communicated as soon as possible. We will be happy to help.
Have fun!

Theoretical Questions

Total score: 13 points

1 Sign Extension (1 point)

Explain why the sign extension is especially needed for branch-, load- and store-instructions.

2 Logical and Bitwise Operations in C (1 point)

Determine the corresponding outputs:

Operation	Result
`0x0 && 0xEF`	
`0xD3 & 0x5B`	
`0x0 || 0xEF`	
`0xA3 | 0x3A`	
`!0xFE`	
`~0xFE`	

3 Infinite Loop (2 points)

The programmer intended the following MIPS program to execute `foo` in line 8 exactly nine times. However, the program does not terminate. Explain why and fix the problem while preserving the original intent (reordering, modifying or removing instructions is not allowed).

```
1  li $t0, 9
2
3  loop:
4  jal foo
5  addi $t0, $t0, -1
6  bne $t0, 0, loop
7
8  foo:
9  li $t1, 0xAFFFFF04
10 sw $t0, 0($t1)
11 jr $ra
```

4 `bne` instead of `beq` (2 points)

What changes are needed in the MIPS implementation presented in the lecture (see "Basic MIPS Architecture Review") to implement `bne` instead of `beq`?

(a) For the singlecycle implementation

(b) For the multicycle implementation

5 Pipeline Registers (1 point)

What are the registers (see slide 6, "Basic MIPS Pipelining Review") between the states needed for?

6 Pipelining Hazard (2 points)

Explain the difference between control, data and structural hazards.

7 Stall (2 points)

Explain why on slide 15 it is enough to wait two clock cycles but not on slide 19 (the slide numbers refer to the chapter "Basic MIPS Pipelining Review").

8 Data Hazard (2 points)

Show all data hazards in the following code. Which dependencies are data hazards that can be resolved by forwarding? Which dependencies are data hazards that will lead to a *stall*?

```
1 add $t0, $t5, $t4
2 lw $s2, 0($t0)
3 sub $s3, $t0 ,$s2
4 sw $t4, 4($s3)
```

Programming Part

The programming exercises can be solved in groups of max. two students. You and your group member work together such that both contribute equally. In order to ensure that both participants understand all code they turn in, we require each student to hand in a slightly different version of the exercise/code. As described below, these versions differ only in a very specific functionality for which each individual student is responsible.

Preparation

(a) Download the code skeleton, MARS MIPS tutorial and JAR file from ILIAS.

(b) Follow our tutorial and get familiar with the simulator.

(c) In this exercise we will use the *Digital Lab Sim* tool. Open it through the menu `Tools` → `Digital Lab Sim` and click on *Connect to MIPS*.

Countdown Clock

In this exercise you will implement a countdown clock in the MARS MIPS simulator as shown in Figure 1a. It will have the following functionality:

- The clock has two digits and can be set to a duration of max. 99 seconds.
- The clock is updating every one second as it is counting down.
- When reaching 0 seconds, the digits start blinking until the clock is turned off.

Tasks

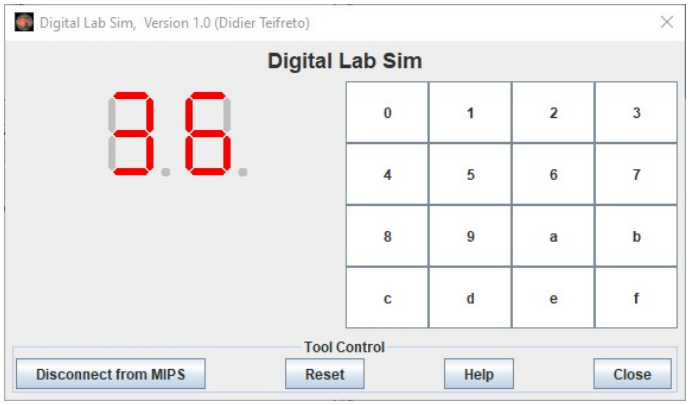
(a) Carefully study the skeleton code we provide. The relevant sections for each subtask is marked by comments in the code. Tutorials on how to use the MARS simulator are provided as supplementary material on ILIAS.

(b) Fill the missing information in the header of the source file (names and version).

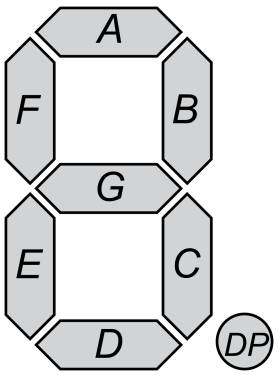
(c) Complete Table 1 with the correct codes to drive the 7-segment display and adjust the array values in section (c) of the source code.

(d) Implement the subroutine `write_digit`. Test your code by calling the subroutine in the `main` section with some test inputs before moving to the next task. Some hints are provided in the comments.

(e) Implement the main program loop of the countdown clock. The skeleton code provides hints in the comments and a subroutine for splitting the counter value into its two digits.



(a) The *Digital Lab Sim* GUI provided by MARS (open it with Tools → Digital Lab Sim).



(b) Use this legend as a reference to fill the table.

Figure 1: A countdown clock made of two 7-segment displays.

(f) Add a blinking animation at the end of the countdown when it reaches 0.

Student 1: You implement the version as described here.
Student 2: In your version of the code, the decimal points should be blinking instead of the digits.

(g) Study the `get_digits` subroutine. Use the MIPS reference sheet to learn about the instructions you don't know yet. Provide a comment for each line explaining what the instructions do.

(h) Upload your source code to ILIAS by Tuesday, 3. May 2021 at 3 p.m. If you are working in a group, you have to upload two versions, one for each student (you can upload them to the same group folder).

Digit	DP	G	F	E	D	C	B	A	Hex
0	0	0	1	1	1	1	1	1	3F
1	0								
2	0								
3	0								
4	0								
5	0								
6	0								
7	0								
8	0								
9	0								

Tabelle 1: Fill this table to determine the hex-codes for each digit. You will need these in section (c) of the code skeleton. Segment A corresponds to the least significant bit and segment DP (decimal point) is at the most significant bit.