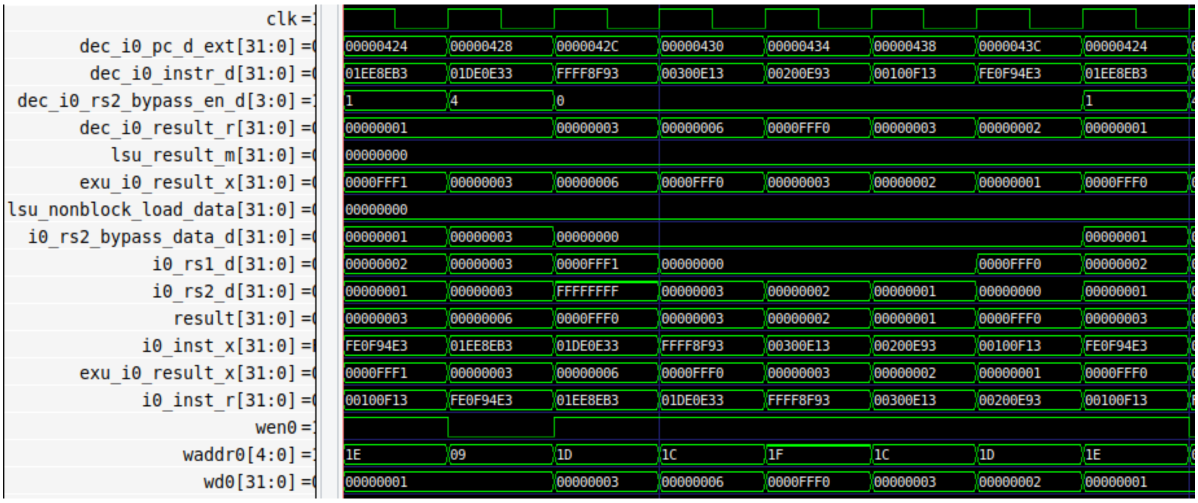
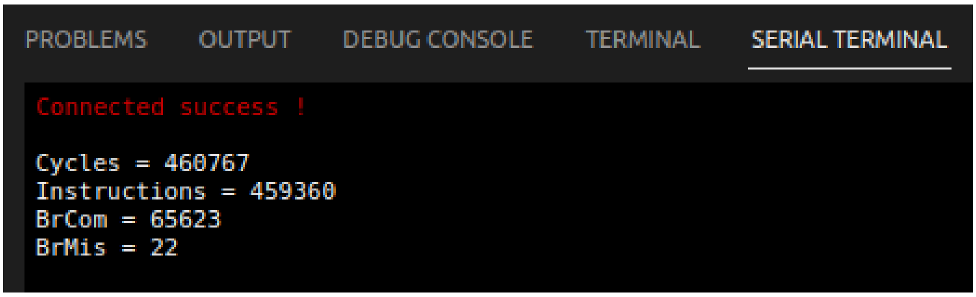
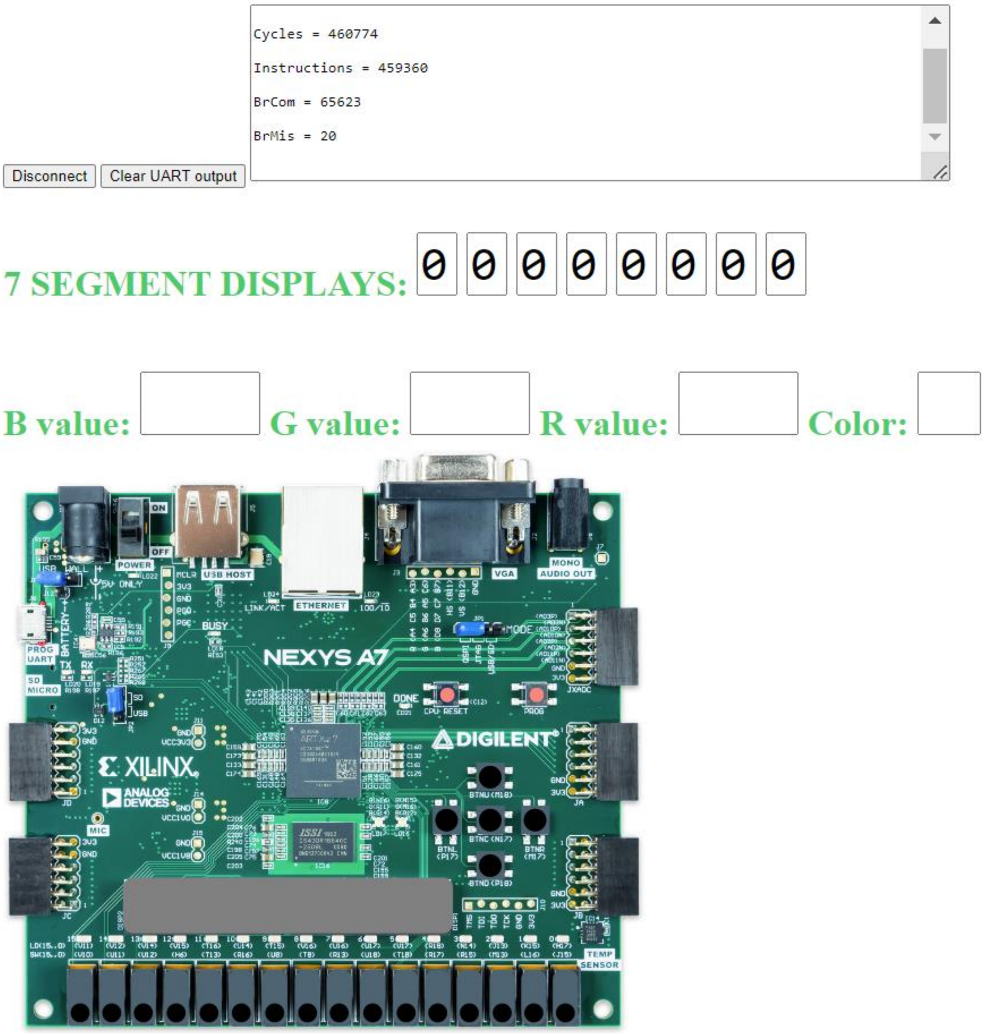
TASK: Remove all `nop` instructions in the example from **Error! Reference source not found.**. Generate the trace with the RVfpga-Trace simulator, analyse the simulation on RVfpga-Pipeline, and then compute the IPC by using the Performance Counters while executing the program on the board (remember that you must uncomment all instructions in the main function, in file *Test.c*).







- The IPC = 1, thanks to the forwarding logic that allows the dependent instruction to not stall.
- The number of cycles is as expected 0xffff * 7 = 458745

<u>Exercise 1</u>: In the example from **Error! Reference source not found.**, analyse and explain similar situations where you replace the dependent `add` instruction for other dependent instructions, such as:

-
```
add t4,t4,t5
mul t3,t3,t4
```

-
```
add t4,t4,t5
div t3,t3,t4
```

-
```
add t4,t4,t5
lw  t3, 0(t4)
```

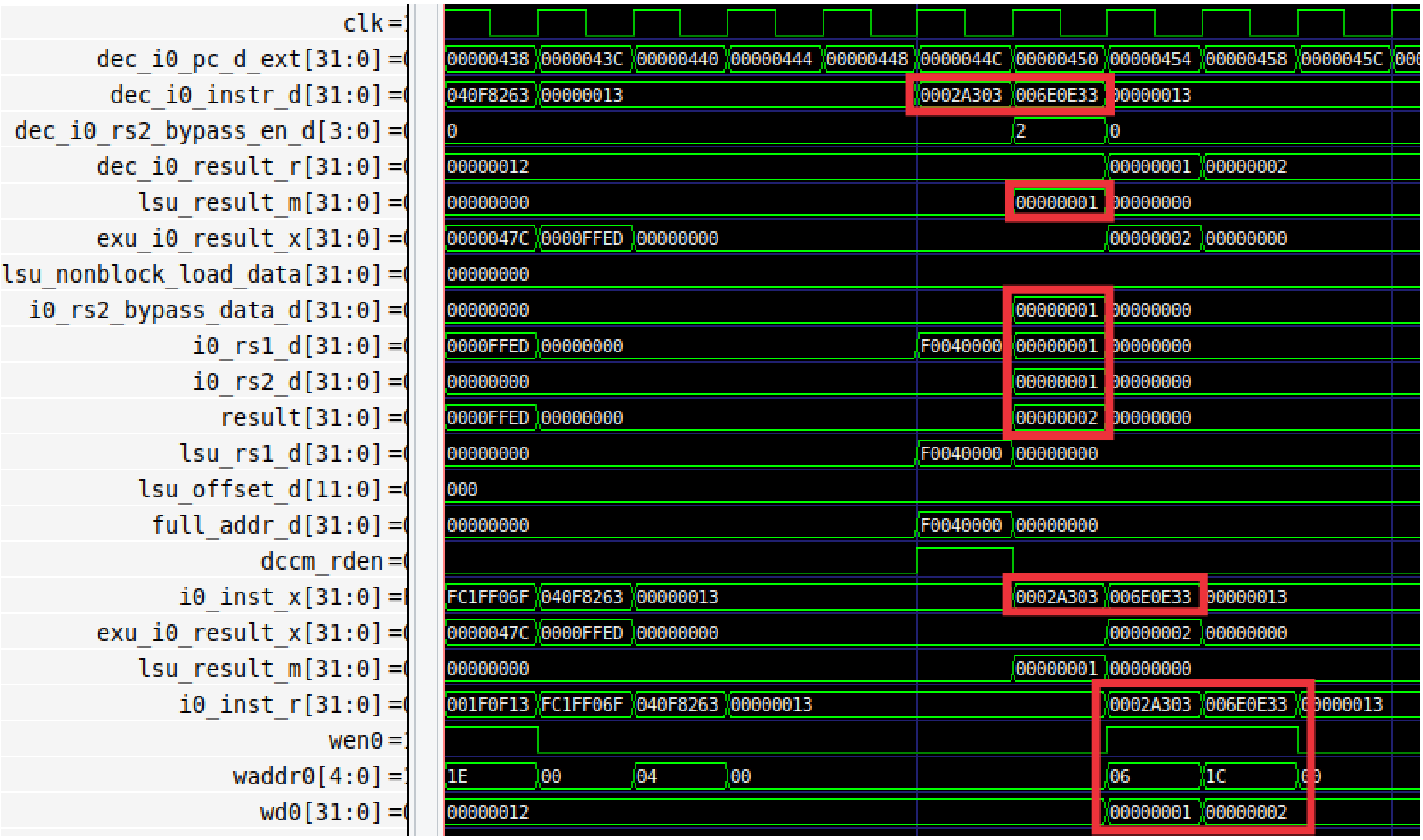
Solution not provided for this exercise.

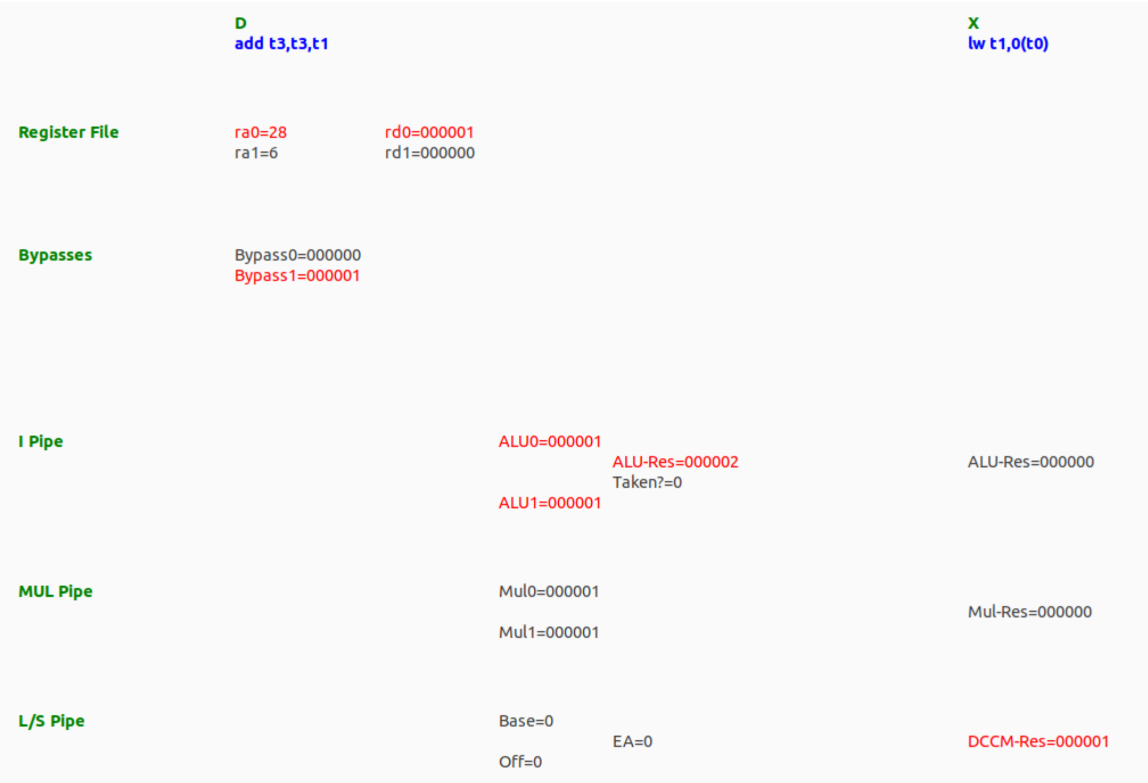
<u>Exercise 2</u>: Use the project called *DataHazards_LW-AL* to analyse a hazard between a `load` and an `add` instruction. Analyse the following two situations:

- **DCCM**: The load reads from the DCCM in a single cycle. You can use the tcl script called *test_DCCM.tcl*. For mapping the data to the DCCM, uncomment, in file Test_Assembly.S, line: `.section .midccm`, and comment line: `.section .ram`

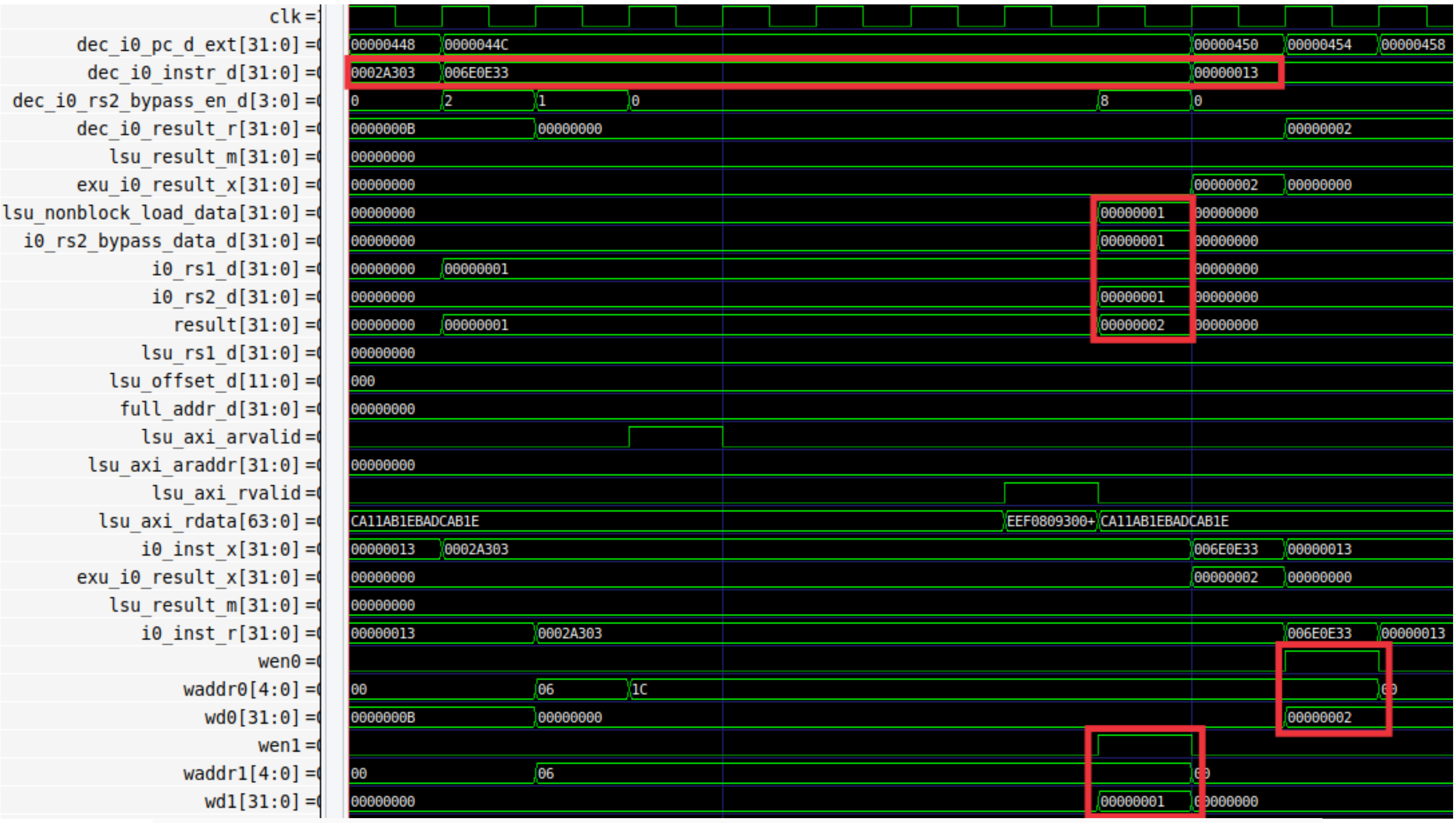
- **Main Memory**: The load reads from Main Memory in several cycles. You can use the tcl script called *test_MainMemory.tcl*. For mapping the data to Main Memory, uncomment, in file Test_Assembly.S, line: `.section .ram`, and comment line: `.section .midccm`

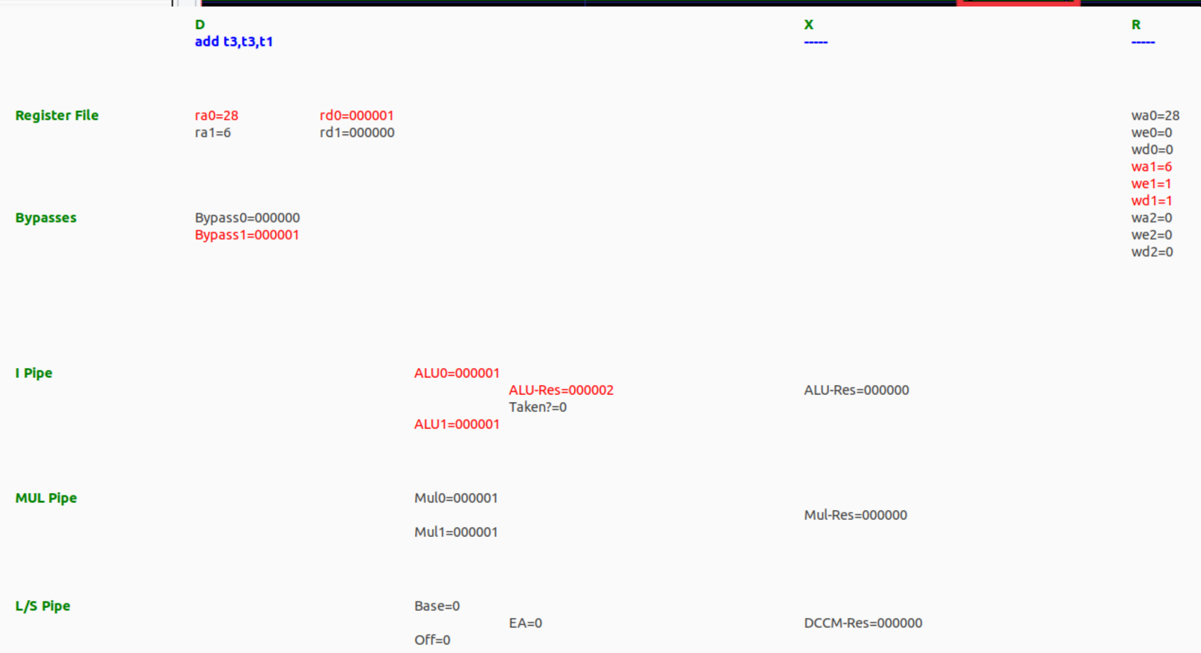
Simulate the program both in RVfpga-Trace and RVfpga-Pipeline.





- The DCCM provides the data in one cycle.
- That data (0x1) is forwarded to the add instruction, which uses it as its second operand instead of the data read from the Register File, which is incorrect.
- There are no stalls thank sto the DCCM low-latency.





- The Main Memory needs several cycles for providing the result, due to the latency of the memory itself and the AXI bus protocol.
- When data is available, it is written to the RF and provided to the add instruction through forwarding, so that it can start right away.