Digital Design and Computer Architecture LU

Lab Protocol

Exercise III

Group 14

Anna Balla, Matr. Nr. 11709469

e11709469@student.tuwien.ac.at

Philipp Geisler, Matr. Nr. 11775812

e11775812 @ student.tuwien.ac. at

Vienna, July 1, 2020

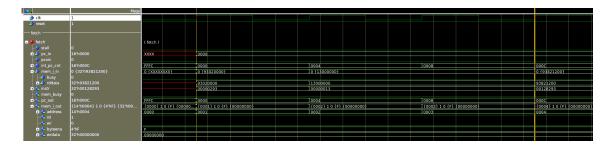


Figure 1: Simulation screenshot for Listing 1. 0.015 us to 0.045 us.

Make sure the following signals are visible in Figure 1 and the signal values are readable: the program counter in the fetch stage, the instruction being fetched, the content of register x5 and the fields address, rd, wr, byteena, and wrdata in the mem_out signal coming out of the pipeline in the memory stage.

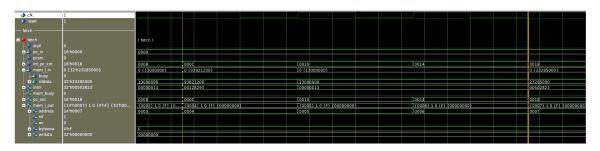


Figure 2: Simulation screenshot from 0.045 us after start-up until 0.075 us.

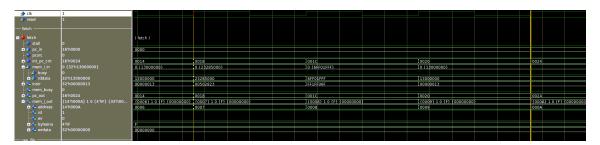


Figure 3: Simulation screenshot from 0.075 us after start-up until 0.105 us.

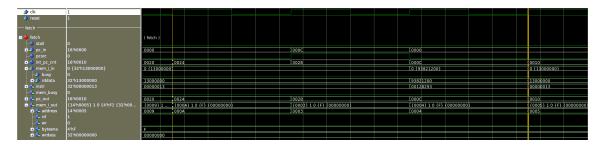


Figure 4: Simulation screenshot from 0.105 us after start-up until 0.135 us.

The figures 1, 2, 3 and 4 show one iteration of the whole code presented in submission.S and the behavior the code triggers in the fetch stage.

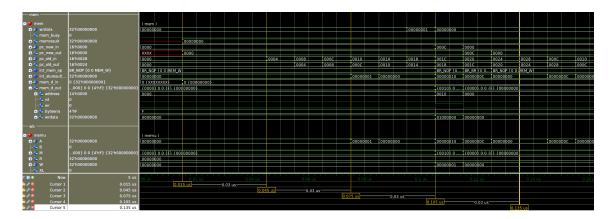


Figure 5: Simulation screenshot showing the behavior of the mem stage during one iteration.

→ mem		(mem)												
📭 🥠 wrdata	32'h00000000	00000000				00000001	00000000							
mem_busy	0													
🖙 📤 memresult	32'h00000000	00000000												
🚅 🥠 pc_new_in		0000					000C		0000					
	16'h0000	0000							000C		0000			
📭 🧆 pc_old_in	16'h0028	000C		0010	10014	[0018	001C		0020		0024		0028	
	16'h0024	8000		000C	10010	[0014	0018		001C		0020		0024	
🖪 🧇 int mem op	BR_NOP {0 0 MEM_W}	BR_NOP {0 I	MEM_W}				BR_NOP {0.1	MEM_W}	BR_BR {0 0 M	EM_W}	BR_NOP {0 0	MEM_W}		
🖪 🧇 int_aluresult	32'h00000000	00000000		00000001	100000000		00000010		0000000C		00000000			
		0 {0000000												
🚍 🧆 mem d out	000} 0 0 {4'hF} {32'h00000000}	{0000} 0 0	{F} {0000000	0}			(0010) 0 1 {	F} {010000	[{0000} 0 0 {F	F} {000000000	4			
🗖 🐴 address	14'h0000	0000					0010		0000					
	0													
- wr	0													
🛱 💠 byteena	4'hF	F												
🗖 🔷 wrdata	32'h00000000	00000000					01000000		00000000					

Figure 6: Simulation screenshot from 0.075 us to 0.135 us of the mem stage.

The figures $\,5$ and $\,6$ show whats going on in the mem stage.

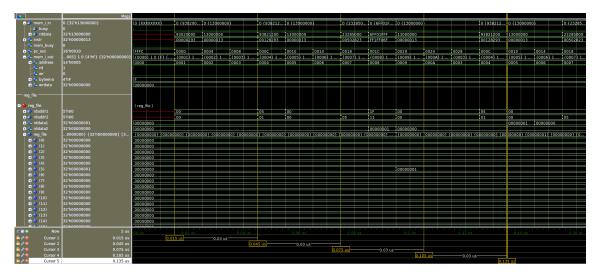


Figure 7: Simulation screenshot from 0 us after start-up until 0.135 us.

Figure 7 shows what's going on in the regfile during the iteration. One can clearly see that the output of the addi instruction is written back to the regfile. If you simulate longer than the first iteration one can see that the assembly-code essential implements a counter that is being incremented.

```
\label{eq:Listing 1: Assembler example without forwarding} $$ addi \ x5 \ , \ x0 \ , \ 0 $$ nop $$ nop $$ so \ x5 \ , \ 15 \ nop $$ nop $$ sw \ x5 \ , \ 16(\ x0) $$ jal \ x0 \ , \ loop $$ nop $$ n
```