Basic RISC CPU (BRC) Verification

By Spencer Hernandez

Introduction

In this report, we aim to verify that the instruction set of the BRC we have been building this semester is fully functional. This will be done by simulating four assembly programs in a Java-based simulator, as well as in Modelsim, and verifying that the final state of each machine matches. Once we do this, we will know that the instructions perform as we intend them to, so the CPU will be operational.

Add Tests (add)

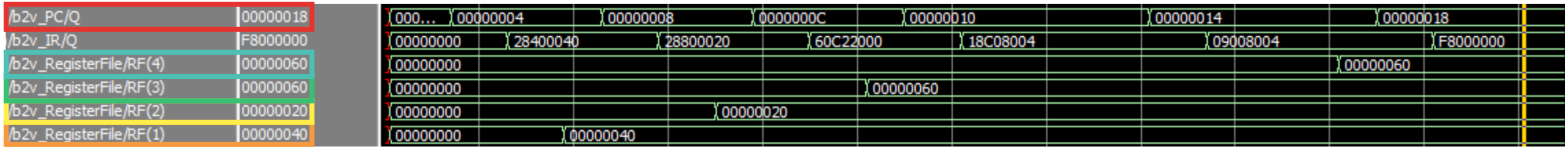
This program verifies the la, add, st, ld, and stop instructions by loading 2 different numbers in different registers, adding them, storing them, and then loading them into a new register

Add Tests (add)

Java-based simulator results

PC =	00000018						
r0 =	00000000	r1 =	00000040	r2 =	00000020	r3 =	00000060
r4 =	00000060	r5 =	00000000	r6 =	00000000	r7 =	00000000
r8 =	00000000	r9 =	00000000	r10 =	00000000	r11 =	00000000
r12 =	00000000	r13 =	00000000	r14 =	00000000	r15 =	00000000
r16 =	00000000	r17 =	00000000	r18 =	00000000	r19 =	00000000
r20 =	00000000	r21 =	00000000	r22 =	00000000	r23 =	00000000
r24 =	00000000	r25 =	00000000	r26 =	00000000	r27 =	00000000
r28 =	00000000	r29 =	00000000	r30 =	00000000	r31 =	00000000

Modelsim results



ALU Tests (alu_tests)

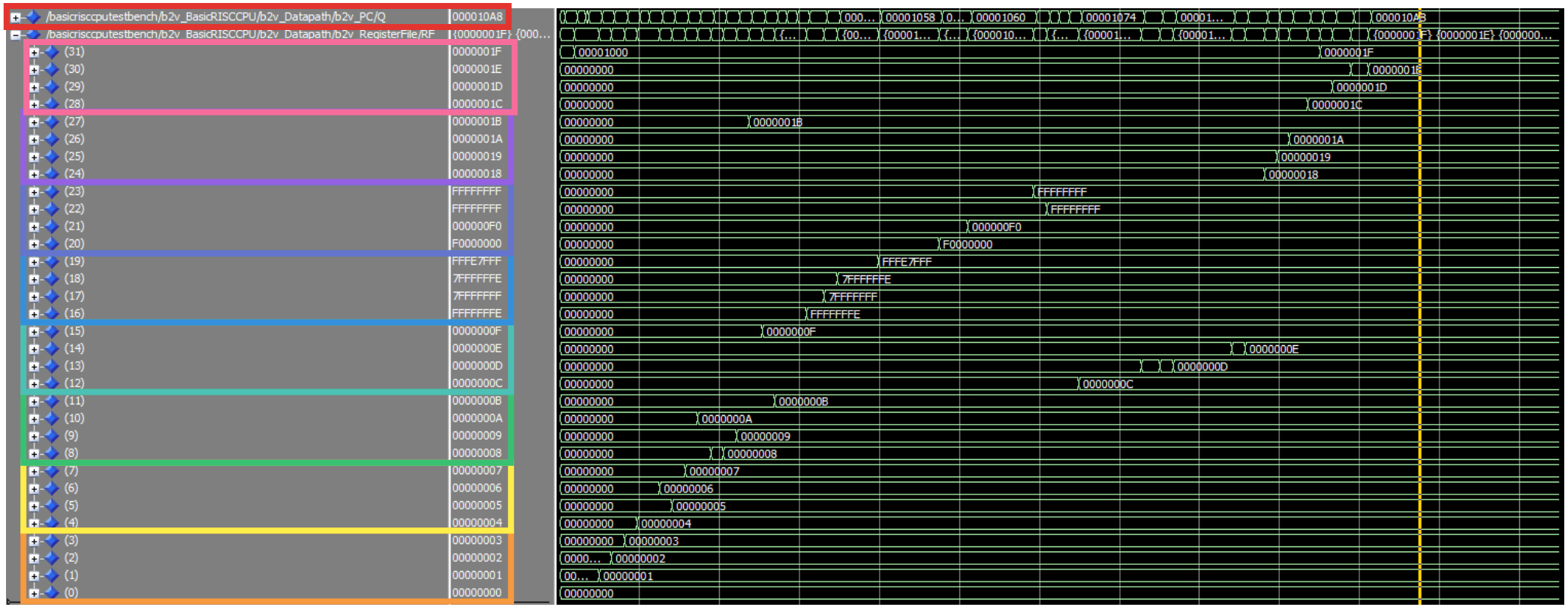
- This program verifies the br, addi, and, or, sub, ori, nop, not, shr, shc, shl, shra, and andi instructions.
- Registers r1 through r15 are filled with their corresponding values (ie: r6 gets loaded with 6)
- Registers r16 through r23 are filled with strings that mostly contain F to see that the bits are being properly changed
- Registers r24 throughr31 are filled with their corresponding values (ie: r29 gets loaded with 29)

ALU Tests (alu_tests)

PC =	000010a8						
r0 =	00000000	r1 =	00000001	r2 =	00000002	r3 =	00000003
r4 =	00000004	r5 =	00000005	r6 =	00000006	r7 =	00000007
r8 =	00000008	r9 =	00000009	r10 =	0000000a	r11 =	0000000b
r12 =	0000000c	r13 =	0000000d	r14 =	0000000e	r15 =	0000000f
r16 =	fffffffe	r17 =	7fffffff	r18 =	7ffffffe	r19 =	fffe7fff
r20 =	f0000000	r21 =	000000f0	r22 =	ffffffff	r23 =	ffffffff
r24 =	00000018	r25 =	00000019	r26 =	0000001a	r27 =	0000001b
r28 =	0000001c	r29 =	0000001d	r30 =	0000001e	r31 =	0000001f

Java-based simulator results

Modelsim results



Branch Tests (branch_tests)

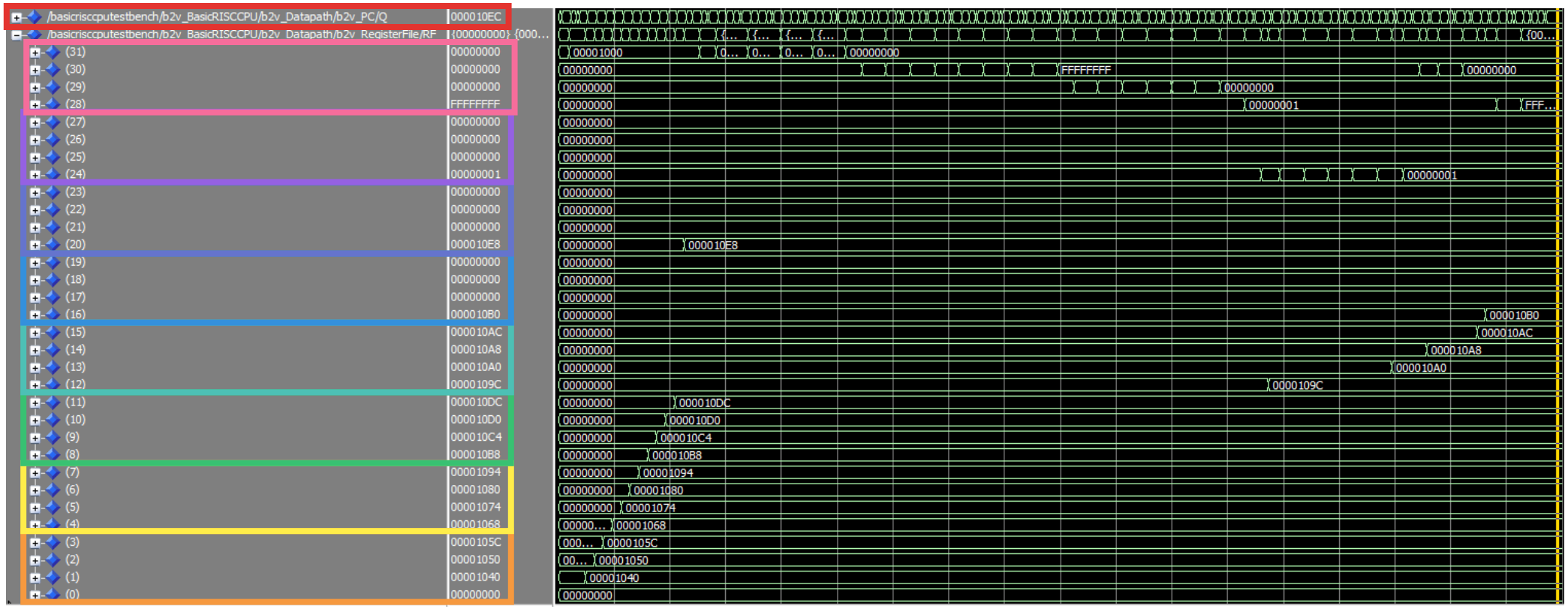
- This program verifies the brl instruction.
- Several addresses are loaded into registers r1 through r11, and then branching instructions are used to act as loops and increment values in registers.
- Branch and link addresses are loaded into registers r12 through r16. The tester must monitor the statements; executed to confirm that control is properly passed based on the register values used.
- r20 is loaded with the address of stop

Branch Tests (branch_tests)

PC =	000010ec						
r0 =	00000000	r1 =	00001040	r2 =	00001050	r3 =	0000105c
r4 =	00001068	r5 =	00001074	r6 =	00001080	r7 =	00001094
r8 =	000010b8	r9 =	000010c4	r10 =	000010d0	r11 =	000010dc
r12 =	0000109c	r13 =	000010a0	r14 =	000010a8	r15 =	000010ac
r16 =	000010b0	r17 =	00000000	r18 =	00000000	r19 =	00000000
r20 =	000010e8	r21 =	00000000	r22 =	00000000	r23 =	00000000
r24 =	00000001	r25 =	00000000	r26 =	00000000	r27 =	00000000
r28 =	ffffffff	r29 =	00000000	r30 =	00000000	r31 =	00000000

Java-based simulator results

Modelsim results



Load Store Tests (load_store_tests)

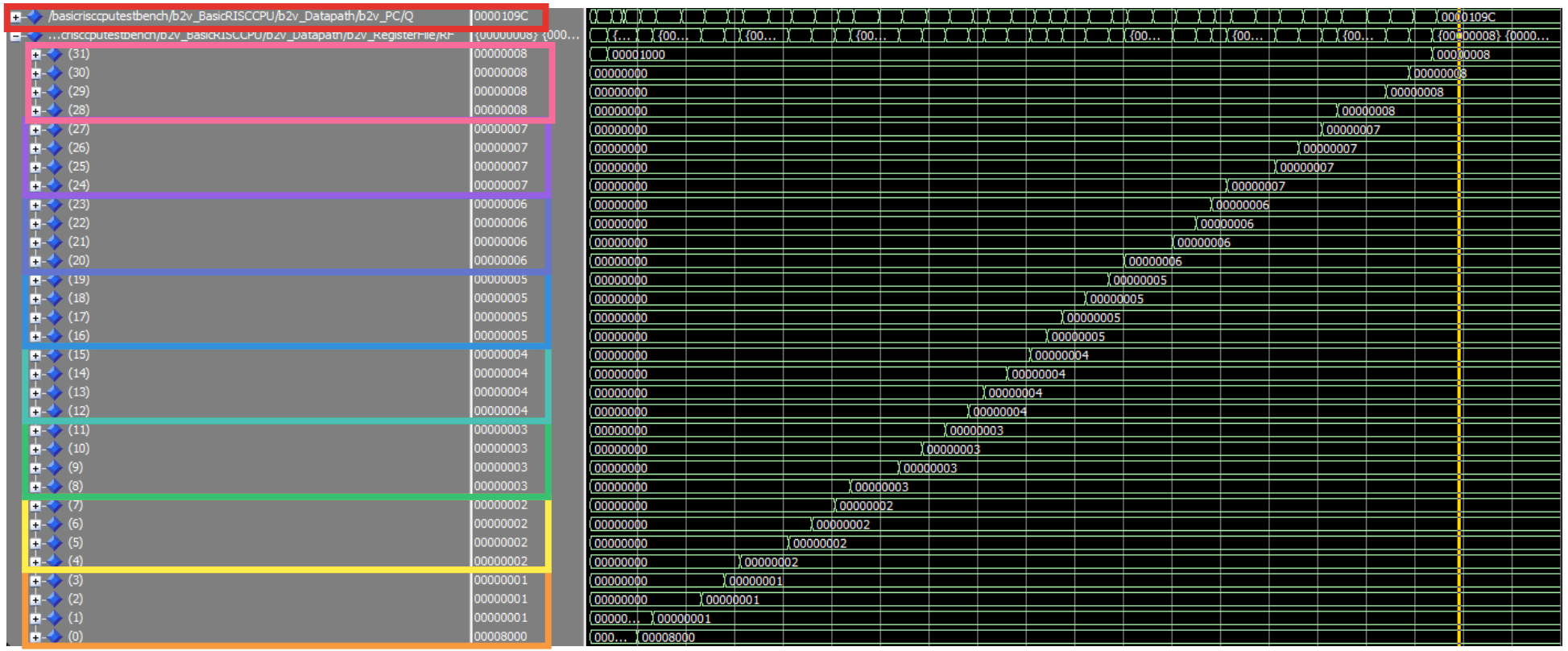
- This program verifies the str, ldr, and lar instructions.
- r0 loaded with 8000 (hex)
- r1 through r3 are loaded with 1s, r4 through r7 loaded with 2s, r8 through r11 filled with 3s, r12 through r15 loaded with 4s, r16 through r19 are loaded with 5s, r20 through r23 loaded with 6s, r24 through r27 filled with 7s, r28 through r31 loaded with 8s

Load Store Tests (load_store_tests)

PC =	0000109c						
r0 =	00008000	r1 =	00000001	r2 =	00000001	r3 =	00000001
r4 =	00000002	r5 =	00000002	r6 =	00000002	r7 =	00000002
r8 =	00000003	r9 =	00000003	r10 =	00000003	r11 =	00000003
r12 =	00000004	r13 =	00000004	r14 =	00000004	r15 =	00000004
r16 =	00000005	r17 =	00000005	r18 =	00000005	r19 =	00000005
r20 =	00000006	r21 =	00000006	r22 =	00000006	r23 =	00000006
r24 =	00000007	r25 =	00000007	r26 =	00000007	r27 =	00000007
r28 =	00000008	r29 =	00000008	r30 =	00000008	r31 =	00000008

Java-based simulator results

Modelsim results



Conclusion

After successfully simulating the tests in the Java-based simulator and running the tests in Modelsim, we can see that for each test, the final state (the values held in all the registers) of each simulation matches. Therefore, the entirety of the instruction set of the CPU is functioning properly, and we can conclude that the functionality of this Basic RISC CPU is verified.