Basic RISC CPU Verification

Kenny Zhang

ECE 4435: Computer Architecture and Design

April 10, 2023

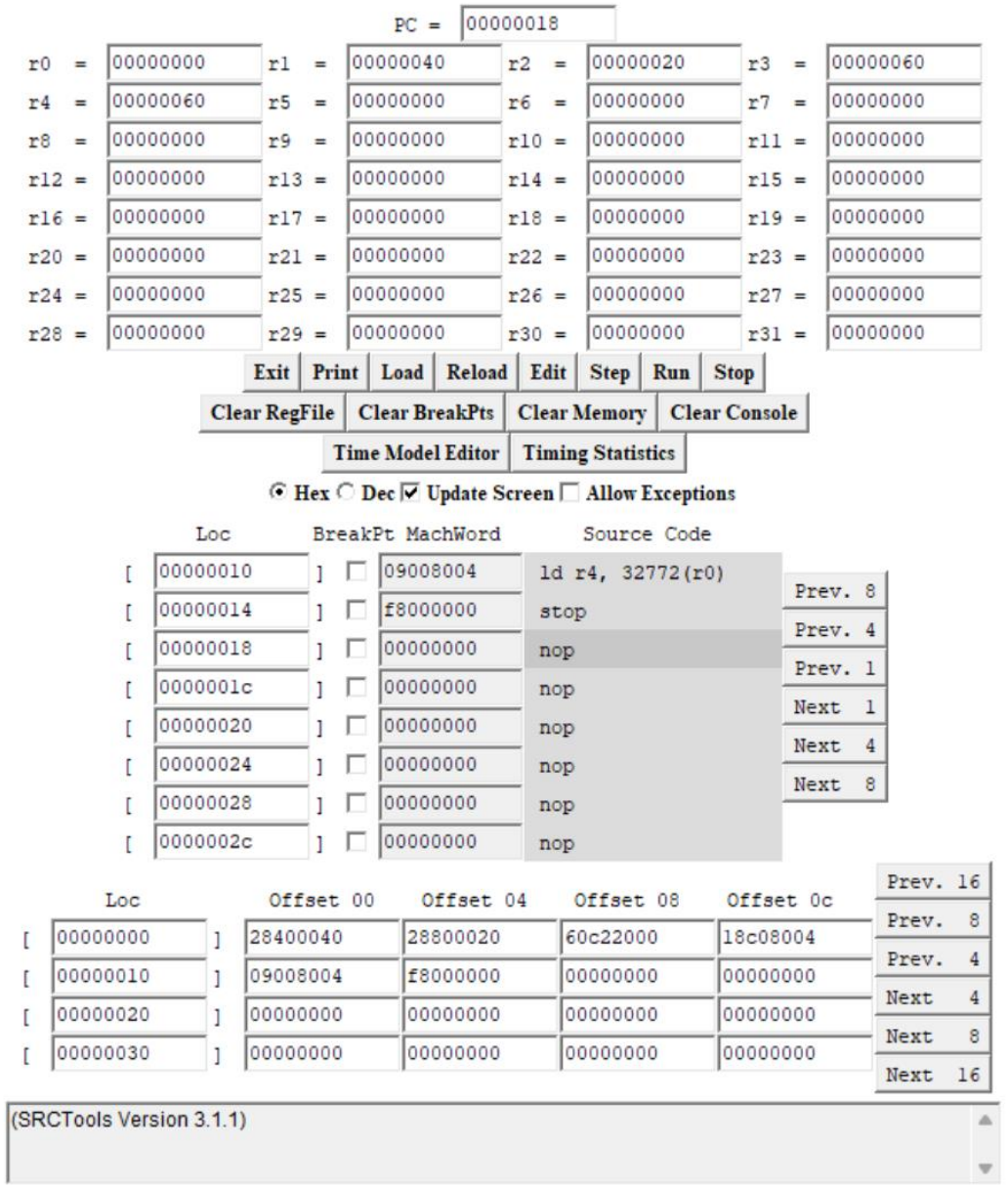
Add Test

1. Program loads 0x00000040 into R1 using la instruction
2. Program loads 0x00000020 into R2 using la instruction
3. Program adds the contents of R1 and R2 and stores the result in R3
4. Program stores the R3 contents into memory and loads it back into R4
5. At the end of the program, PC = 0x00000018 and IR = 0xF8000000

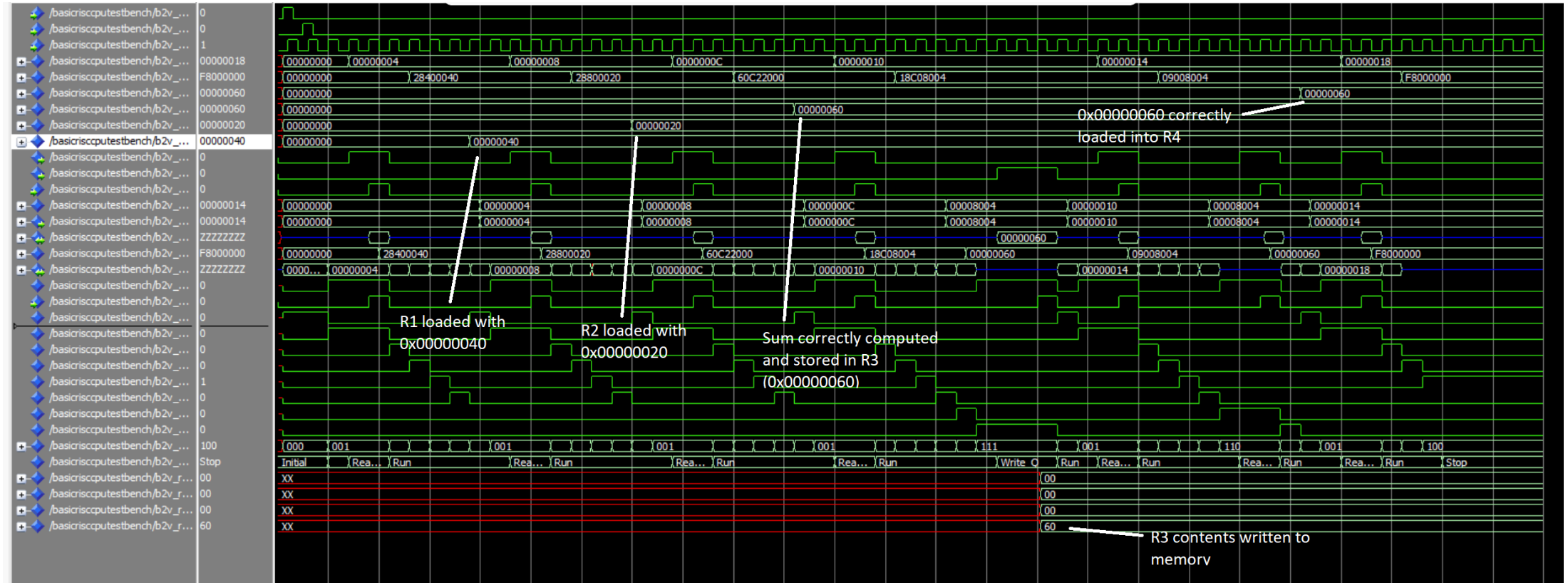
Add Test: Simulator

Register Contents

- PC = 0x00000018
- IR = 0xF80000000
- R1 = 0x00000040
- R2 = 0x00000020
- R3 = 0x00000060
- R4 = 0x00000060
- R5-R31 = 0x00000000



Add Test: Modelsim



Register Contents

PC = 0x00000018 R1 = 0x00000040 R2 = 0x00000020 R3 = 0x00000060 R4 = 0x00000060

IR = 0xF8000000 R5-R31 = 0x0000000-0

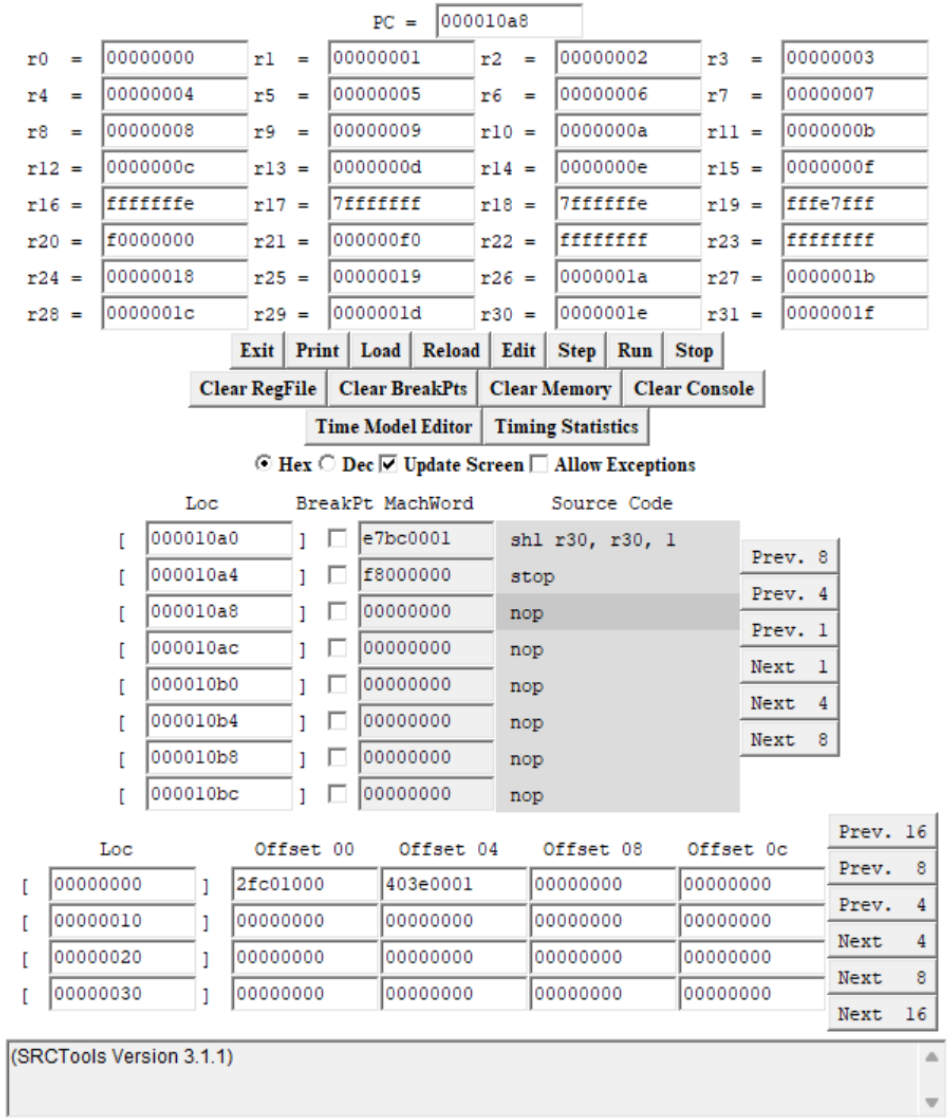
ALU Tests

1. Program uses a variety of arithmetic ALU operations (add, addi, or, ori, sub, neg, and) to populate R1-R11, R15, and R27 (using their register number)
2. Registers R16 through R23 were populated with strings to test the shift and bit manipulation operations (not, shr, shra, shl, shc)
3. Registers R24 through R31 were filled with their register number
4. At the end of the program, PC = 0x000010A8 and IR = 0xF8000000

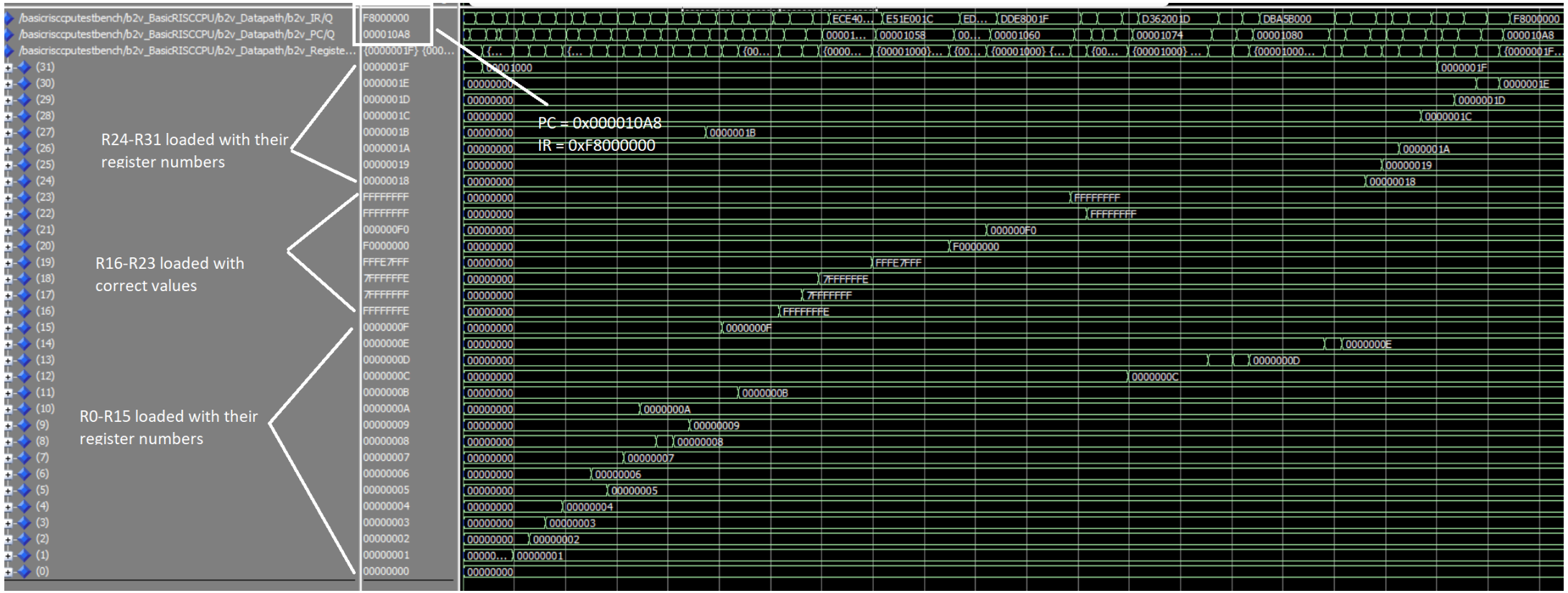
ALU Tests: Simulator

Register contents

- PC = 0x000010A8
- IR = 0xF8000000
- R0-R15 = 0x00000000-0x0000000F (0 through 15)
- R16 = 0xFFFFFFFE
- R17 = 0x7FFFFFFF
- R18 = 0x7FFFFFFE
- R19 = 0xFFFE7FFF
- R20 = 0xF0000000
- R21 = 0x000000F0
- R22 = 0xFFFFFFFF
- R23 = 0xFFFFFFFF
- R24-R31 = 0x00000018-0x0000001F (24 through 31)



ALU Tests: Modelsim



Register Contents

PC = 0x000010A8 IR = 0xF8000000 R0-R15 = 0x00000000-0x0000000F R16 = 0xFFFFFFFE

R17 = 0x7FFFFFFF R18 = 0x7FFFFFFE R19 = 0xFFFE7FFF R20 = 0xF0000000 R21 = 0x000000F0

R22 = 0xFFFFFFFF R23 = 0xFFFFFFFF R24-R31 = 0x00000018-0x0000001F

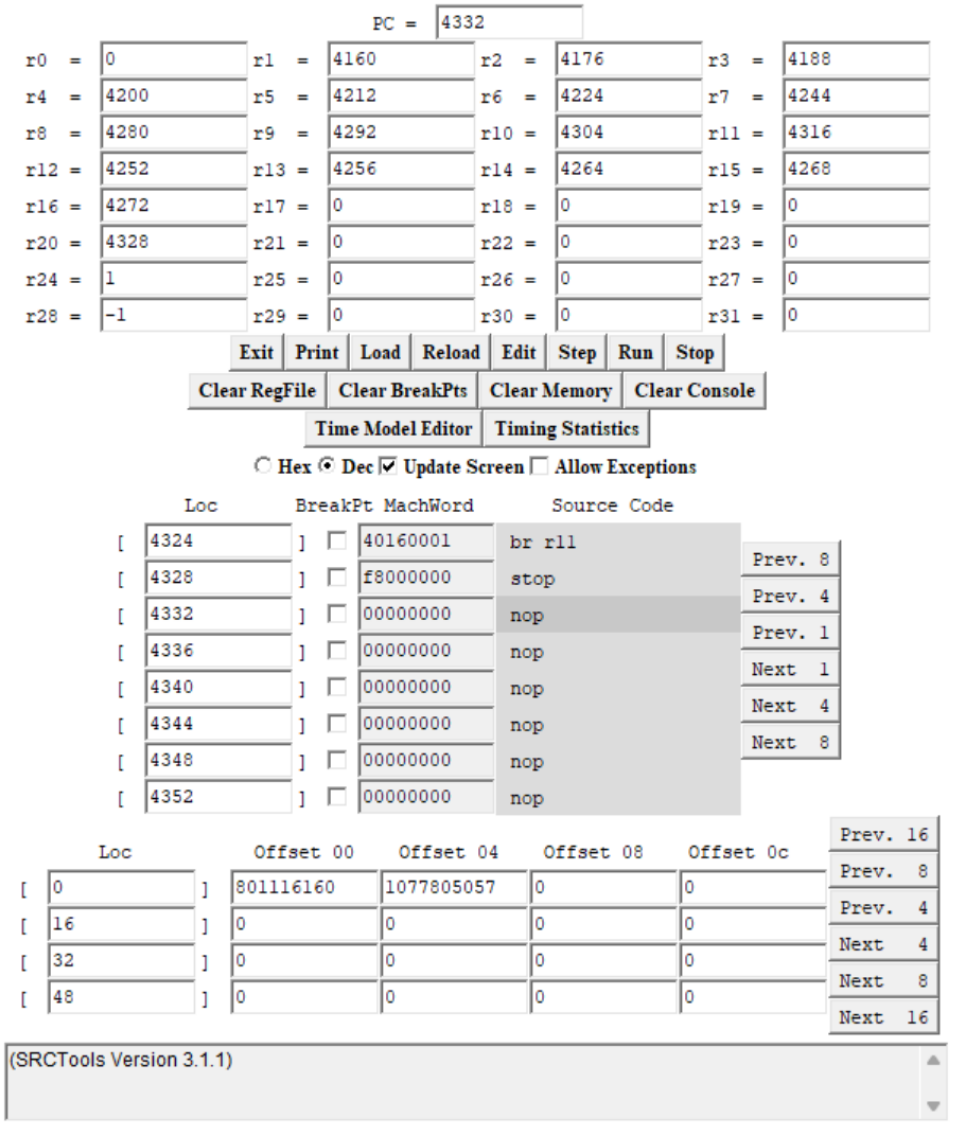
Branch Tests

1. First, 4096 is loaded into R31 and 11 various addresses are loaded into R1-R11 using la to act as jump locations
2. R20 is loaded with the address of “Stop”
3. Various loops are used to test the different branch instructions (br, brzr, brnv, brmi, brpl, brlzr, brl, brlnv, brlnz, brlmi)
    1. R31 loop counts from 5-0 then breaks
    2. R30 loop counts from 7 to -1
    3. R29 loop counts from -6 to 0
    4. R24 loop counts from 5-0 then up to 1
    5. R28 loop counts from 1 to 0 to -1 then breaks

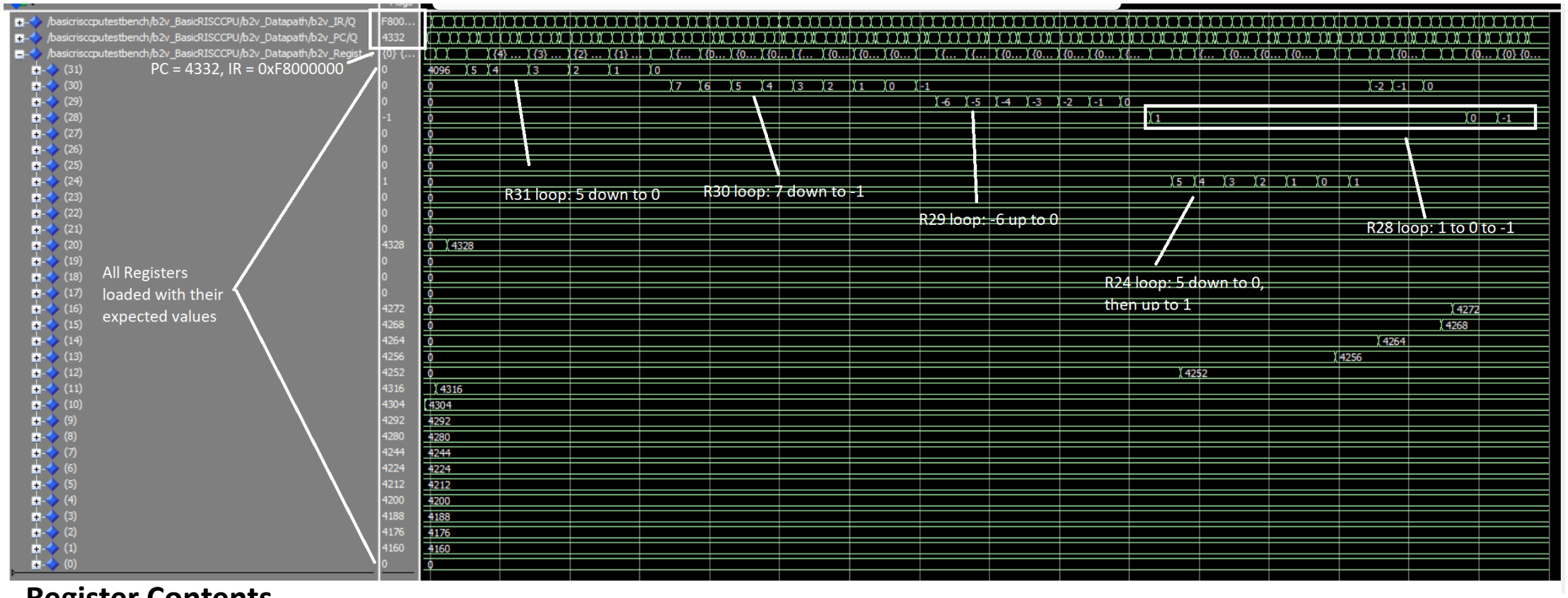
Branch Tests: Simulator

Register Contents

- PC = 4332
- IR = 0xF8000000
- R0 = 0, R1 = 4160, R2 = 4176, R3 = 4188
- R4 = 4200, R5 = 4212, R6 = 4224, R7 = 4244
- R8 = 4280, R9 = 4292, R10 = 4304, R11 = 4316
- R12 = 4252, R13 = 4256, R14 = 4264, R15 = 4268
- R16 = 4272, R17-R19 = 0, R20 = 4328, R21-R23 = 0
- R24 = 1, R25-R27 = 0, R28 = -1, R29-R31 = 0



Branch Tests: Modelsim



Register Contents

- PC = 4332, IR = 0xF8000000
- R0 = 0, R1 = 4160, R2 = 4176, R3 = 4188, R4 = 4200, R5 = 4212, R6 = 4224, R7 = 4244
- R8 = 4280, R9 = 4292, R10 = 4304, R11 = 4316, R12 = 4252, R13 = 4256, R14 = 4264, R15 = 4268
- R16 = 4272, R17-R19 = 0, R20 = 4328, R21-R23 = 0, R24 = 1, R25-R27 = 0, R28 = -1, R29-R31 = 0

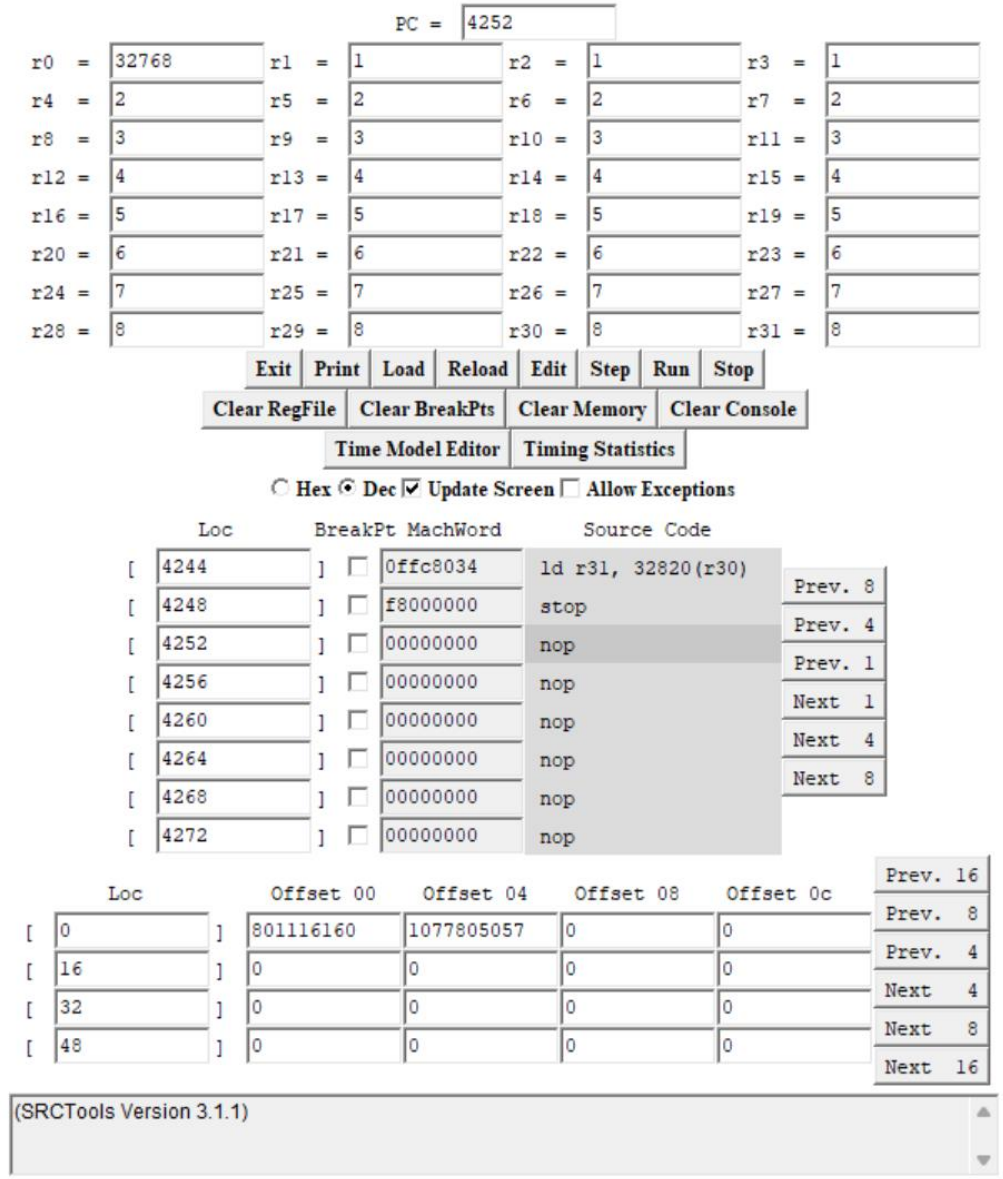
Load-Store Tests

- First, R0 is loaded with the constant 0x8000 (32768)
- Using the lar, str, ldr, st, ld, and la operations, each of the registers R1 through R31 are loaded with values representing their row numbers
- Ex: Registers R1-R3 are in row 1, R4-R7 are in row 2, R8-R11 are in row 3, etc
- At the end of the program, PC = 4252 and IR = 0xF8000000

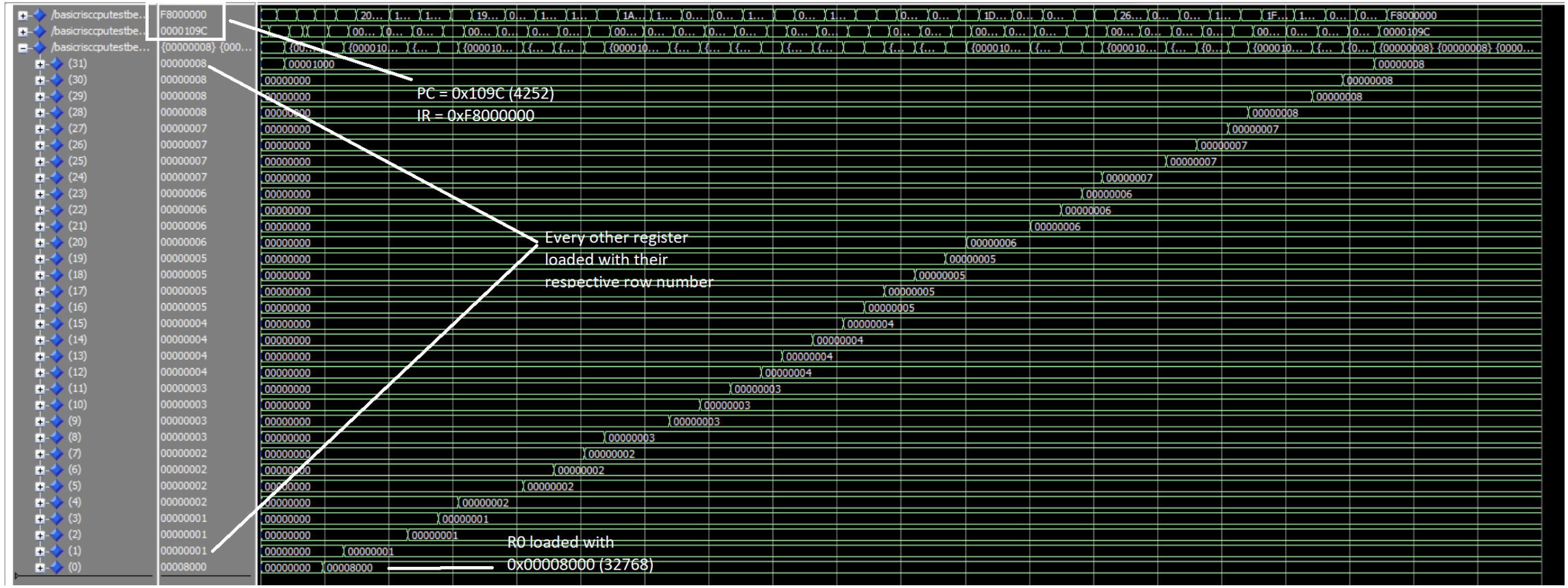
Load-Store Tests: Simulator

Register Contents

- PC = 4252
- IR = 0xF8000000
- R0 = 32768
- R1-R3 = 1
- R4-R7 = 2
- R8-R11 = 3
- R12-R15 = 4
- R16-R19 = 5
- R20-R23 = 6
- R24-R27 = 7
- R28-R31 = 8



Load-Store Tests: Modelsim



Register Contents

- PC = 4252 (0x0000109C), IR = 0xF8000000, R0 = 32768 (0x00008000), R1-R3 = 1
- R4-R7 = 2, R8-R11 = 3, R12-R15 = 4, R16-R19 = 5, R20-R23 = 6, R24-R27 = 7, R28-R31 = 8

Conclusion

- Collectively, all 4 of the assembly programs test all of the CPU instructions
- Every Modelsim simulation matches the results produced by the assembly simulator
- Thus, we can confidently say that the Basic RISC CPU that we have designed works as intended