Team 3 - ECE 586 Project - Test Plan

Braden Harwood, Michael Weston, Stephen Short, Drew Seidel

https://github.com/Nast33Narwhal/ECE586_Team3

March 15, 2022 - V1.0

Test Plan

The test plan for this simulator involved using the RISCV specifications document (can be found at: https://riscv.org/technical/specifications/), going through each implemented function, and making sure that each function followed characteristics and behavior described in the specification. Other tests were written as well to check for other errors (writing to registers illegally, etc.). The main method of creating the tests was by creating assembly (.s) files, compiling them into object (.o) files using **rvas -ahld file.s -o file.o**. These created object files were used with **rvobjdump -d file.o** to dump the commands along with their hex values. These hex values were then formatted into .mem files which are the type of files this simulator runs. Expected values for each function were placed in the .s files as comments, and these expected values were compared against the results of the simulator. In some cases, a simulator created by Cornell University was also used to compare against (found at https://www.cs.cornell.edu/courses/cs3410/2019sp/riscv/interpreter/). The following table illustrates all the implemented functions of this simulator along with the test cases they passed.

Instruction Type	Operation Type	Filename	Test Notes
R	Register Arithmetic	add.mem	Largest positive number Largest negative number Negative number result Positive number result Negative overflow Positive overflow
	Logical operations	logic.mem	Tests register OR - 0 OR 0 = 0 - 0 OR 1 = 1 - 1 OR 1 = 1 Tests register AND - 0 AND 0 = 0 - 0 AND 1 = 0 - 1 AND 1 = 1 Tests register XOR - 0 XOR 0 = 0 - 0 XOR 1 = 1 - 1 XOR 1 = 0
	Multiply operations	mul.mem	All RV32M extension tests MULH/MULHU DIV DIVU REM REMU

	Logical shifts	shifts.mem	Tests logical shifts SLL SRL SRA (sign extension = MSBit)
	SLT / SLTU	slt_u.mem	Test SLT - correct sign extensions - correct positive # comparison - correct negative # comparison - rs1 == rs2 test Test SLTU - compare registers w/ 1 in MSBit (opposite result to SLT w/ same register values because unsigned) - compare registers w/ 0 in MSBit (same result to SLT w/ same register values) - SNEZ special case test. Rd = 1 if rs2 != 0
	Register subtract	sub.mem	Subtraction test cases Positive - Negative Negative - Negative (-(-) == +) Negative overflow test Positive overflow test
I	Immediate Arithmetic	addi.mem	ADDI - Positive overflow test - Negative overflow test - negative/positive result test - MV pseudo instruction test
	ECALL	ecall.mem	Ecall test cases Reading from STDIN Printing to STDOUT Exiting program
	Immediates in C	immediate.mem	C program created test case
	Load	load.mem	LW, LH, LB - tested loading from byte-aligned addresses - tested sign extension correctness for LH and LB MSBit LHU, LBU - tested sign extension of 0s for MSBit
	Logic with immediates	logic_I.mem	ANDI, ORI, XORI - Same tests as ‘logic.mem’ but w/ rs1 and immediate
	Shifting with immediates	shifts_I.mem	SLLI, SLRI, SRAI - Same tests as ‘shifts.mem’ but w/ rs1 and immediate

	SLT/SLTU with immediate	slti_u.mem	SLTI, SLTIU - tested correct sign extension of immediate for SLTI and SLTIU - pseudo instruction (SEQZ) test (rd is set to 1 if rs1 == 0 and imm == 0x1)
U	LUI/AUIPIC tests	lui_auipc.mem	LUI implicitly tested in all other tests because of its need to be used to set up other tests. AUIPC tested
S	Store tests	store.mem	SW, SB, SH - tested for byte-wise memory storage w/ no sign extension for word aligned memory
B	Conditional branch basic	branch.mem	BEQ, BLT, BGE, BLTU, BGEU - Test all types and conditions working properly
	Conditional branch complex	branch2.mem	C made: For loops, if, else branching
	Conditional branch bounds	branchBound.mem	Testing minimum and maximum branch bounds (+4092 bytes or - 4096 bytes)
J	Unconditional branches	jal.mem	Tests JAL function, and unconditional branching in the simulator