Simulizer Test Report

Charlie Street Kelsey McKenna Matthew Broadway Michael Oultram
Theo Styles

March 11, 2016

1 Introduction

This document is aimed at displaying the testing procedures carried out for the Simulizer software, created by team A4 for the team project at the University of Birmingham.

1.1 Test plan

The plan for the testing of this software is to split it up into split it into different sections such that we can fully and comprehensively test our system with a high level of acceptance at the end of the project. The area of tests carried out will be as follows:

- Unit testing
- Integration testing
- Comparison testing (Comparing our software's execution with that of existing software)
- Functional testing
- End-to-end testing
- User testing/evaluation

The unit testing will be carried out as we write the software, with many JUnit tests written for each component of the software (in isolation). The integration tests will concern us integrating the different components of our software together, for example, combining the assembler and simulation together and checking that they, together, can be used to correctly execute an assembly language program. Due to the fact that our aim in this project is to make a better, more usable alternative to existing software, we feel it is also a sensible idea to concern various outputs from our software with that of the existing software (Spimulator in this case). This will tell us whether we have at least reached the same level of quality/accuracy as the current software and, hopefully, tell us where we have made an improvement of some sort. The final area of our testing is user testing/evaluation. We are very fortunate to have a well defined set of users for this project and hence we intend to utilise this by getting opinions on our software and feedback from both a set of students taking the Computer Systems & Architecture module, as well as the module lecturer, Ian Batten. Due to the agile nature of this project, we will able to do this continuously throughout the project, to give the best possible end product to our users.

2 Unit testing

All separate components of our software (bar maybe our main UI) will be tested through the use of JUnit tests. All of these tests will be visible in the src/test directory in svn, but the following will lay out our tests on the separate components of the software along with their expected and recorded results:

Test ID	Method	Instruction	Word 1	Word 2	Expected	Actual	Pass (y/n)	Date tested
ALU1	execute	abs	-10	empty	10	10	y	16/2/16
ALU2	execute	abs	27	empty	27	27	y	16/2/16
ALU3	execute	abs	0	empty	0	0	y	16/2/16
ALU4	execute	and	0	17	0	0	y	16/2/16
ALU5	execute	and	17	24	16	16	y	16/2/16
ALU6	execute	and	-5	-2	-6	-6	y	16/2/16
ALU7	execute	add	4	7	11	11	y	16/2/16
ALU8	execute	add	-4	-10	-14	-14	y	16/2/16
ALU9	execute	add	-7	6	-1	-1	y	16/2/16
ALU10	execute	addu	2^{31}	$2^{31} - 1$	$2^{32} - 1$	$2^{32} - 1$	y	16/2/16
ALU11	execute	addu	2^{30}	2^{30}	2^{31}	2^{31}	y	16/2/16
ALU12	execute	addu	4	0	4	4	y	16/2/16
ALU13	execute	addi	4	7	11	11	y	16/2/16
ALU14	execute	addi	-4	-10	-14	-14	y	16/2/16
ALU15	execute	addi	-7	6	-1	-1	y	16/2/16
ALU16	execute	addiu	2^{31}	$2^{31} - 1$	$2^{32} - 1$	$2^{32} - 1$	y	16/2/16
ALU17	execute	addiu	2^{30}	2^{30}	2^{31}	2^{31}	y	16/2/16
ALU18	execute	addiu	4	0	4	4	y	16/2/16
ALU19	execute	sub	4	7	-3	-3	y	16/2/16
ALU20	execute	sub	7	4	3	3	y	16/2/16
ALU21	execute	sub	-4	-10	6	6	y	16/2/16
ALU22	execute	subu	2^{31}	2^{31}	0	0	y	16/2/16

Test ID	Method	Instruction	Word 1	Word 2	Expected	Actual	Pass (y/n)	Date tested
ALU23	execute	subu	2^{31}	0	2ˆ{31}	2ˆ{31}	y	16/2/16
ALU24	execute	subu	$2^{32} - 1$	1	$2^{31} - 2$	$2^{31} - 2$	y	16/2/16
ALU25	execute	subi	4	7	-3	-3	y	16/2/16
ALU26	execute	subi	7	4	3	3	y	16/2/16
ALU27	execute	subi	-4	-10	6	6	y	16/2/16
ALU28	execute	subiu	2^{31}	2^{31}	0	0	y	16/2/16
ALU29	execute	subiu	2^{31}	0	2^{31}	2^{31}	y	16/2/16
ALU30	execute	subiu	$2^{32} - 1$	1	$2^{31} - 2$	$2^{31} - 2$	y	16/2/16
ALU31	execute	mul	2^{15}	$2^{15} - 1$	1083709056	1083709056	y	16/2/16
ALU32	execute	mul	0	2^{15}	0	0	y	16/2/16
ALU33	execute	mul	-4	-3	12	12	y	16/2/16
ALU34	execute	mulo	2^{15}	$2^{15} - 1$	1083709056	1083709056	y	16/2/16
ALU35	execute	mulo	0	2^{15}	0	0	y	16/2/16
ALU36	execute	mulo	-4	-3	12	12	y	16/2/16
ALU37	execute	mulou	2^16	$2^16 - 1$	4294901760	4294901760	y	16/2/16
ALU38	execute	mulou	0	2^16	0	0	y	16/2/16
ALU39	execute	mulou	4	3	12	12	y	16/2/16
ALU40	execute	div	0	4	0	0	y	16/2/16
ALU41	execute	div	4	2	2	2	y	16/2/16
ALU42	execute	div	4	-2	-2	-2	y	16/2/16
ALU43	execute	divu	0	4	0	0	y	16/2/16
ALU44	execute	divu	$2^{32} - 1$	$2^{32} - 1$	1	1	y	16/2/16
ALU45	execute	divu	4	2	2	2	y	16/2/16
ALU46	execute	neg	0	empty	0	0	y	16/2/16
ALU47	execute	neg	1	empty	-1	-1	y	16/2/16
ALU48	execute	neg	-1	empty	1	1	y	16/2/16

Test ID	Method	Instruction	Word 1	Word 2	Expected	Actual	Pass (y/n)	Date tested
ALU49	execute	nor	0	0	0	0	y	16/2/16
ALU50	execute	nor	0	1	2^{30}	2^{30}	y	16/2/16
ALU51	execute	nor	-1	1	1	1	y	16/2/16
ALU52	execute	not	0	empty	-1	-1	y	16/2/16
ALU53	execute	not	-1	empty	0	0	y	16/2/16
ALU54	execute	not	1	empty	-2	-2	y	16/2/16
ALU55	execute	or	0	1	1	1	y	16/2/16
ALU56	execute	or	1	4	5	5	y	16/2/16
ALU57	execute	or	16	4	20	20	y	16/2/16
ALU58	execute	ori	0	1	1	1	y	16/2/16
ALU59	execute	ori	1	4	5	5	y	16/2/16
ALU60	execute	ori	16	4	20	20	y	16/2/16
ALU61	execute	xor	4	14	10	10	y	16/2/16
ALU62	execute	xor	3	1	2	2	y	16/2/16
ALU63	execute	xor	3	0	3	3	y	16/2/16
ALU64	execute	xori	4	14	10	10	y	16/2/16
ALU65	execute	xori	3	1	2	2	y	16/2/16
ALU66	execute	xori	3	0	3	3	y	16/2/16
ALU67	execute	b	0	empty	branchTrue	branchTrue	y	16/2/16
ALU68	execute	b	-20	empty	branchTrue	branchTrue	y	16/2/16
ALU69	execute	b	20	empty	branchTrue	branchTrue	y	16/2/16
ALU70	execute	beq	0	1	branchFalse	branchFalse	y	16/2/16
ALU71	execute	beq	0	0	branchTrue	branchTrue	y	16/2/16
ALU72	execute	beq	0	-1	branchFalse	branchFalse	y	16/2/16
ALU73	execute	bne	0	1	branchTrue	branchTrue	y	16/2/16
ALU74	execute	bne	0	0	branchFalse	branchFalse	y	16/2/16

Test ID	Method	Instruction	Word 1	Word 2	Expected	Actual	Pass (y/n)	Date tested
ALU75	execute	bne	0	-1	branchTrue	branchTrue	y	16/2/16
ALU76	execute	bgez	0	1	branchTrue	branchTrue	y	16/2/16