

Compliance Task Group Call – Minutes

Thur, 05Nov2020 8am Pacific → Daylight ← Time

See slide 6 for agenda

Antitrust Policy Notice



RISC-V International meetings involve participation by industry competitors, and it is the intention of RISC-V International to conduct all its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

Examples of types of actions that are prohibited at RISC-V International meetings and in connection with RISC-V International activities are described in the RISC-V International Regulations Article 7 available here: <https://riscv.org/regulations/>

If you have questions about these matters, please contact your company counsel.

RISC-V International Code of Conduct



RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

We are a transparent, collaborative community where all are welcomed, and all members are encouraged to participate.

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone.

<https://riscv.org/risc-v-international-community-code-of-conduct/>

Charter

The Compliance Task Group will

- Develop compliance tests for RISC-V implementations, taking into account approved specifications for:
 - Architectural versions (e.g. RV32I, RV32E, RV64I, RV128I)
 - Standard Extensions (H,S,U,A,B,C,D,F,J,K,M,N,P,Q,T,V,N), Priv Mode ← change to only ratified spec as of this date
 - All spec'ed implementation options
 - (incl. MHSU modes, optional CSRs, optional CSR bits)
- Develop a method for selecting and configuring appropriate tests for a RISC-V implementation, taking into account:
 - Platform profile and Execution Environment (EE)
 - Implemented architecture, extensions, and options
- Develop a framework to apply the appropriate tests to an implementation and verify that it meets the standard
 - test result signature stored in memory will be compared to a golden model result signature

Administrative Pointers

- Chair – Allen Baum allen.baum@esperantotech.com
- Co-chair – Bill McSpadden bill.mcspadden@seagate.com
- TG Email tech-compliance@lists.riscv.org
 - Notetakers: please send emails to allen.baum@esperantotech.com
- Meetings -Bi-monthly at 8am Pacific time on 2nd/4th Wednesdays.
 - See https://docs.google.com/spreadsheets/d/1L15_gHI5b2ApkcHVtpZyl4s_A7sgSrNN zoom link
- Documents, calendar, roster, etc. in
 - <https://lists.riscv.org/tech-compliance/> see /documents & /calendars subdirectories
 - <https://riscv.readthedocs.io/en/latest/> riscv
 - <https://riscv-config.readthedocs.io/en/latest/> config: YAML and WARL spec
 - <https://drive.google.com/drive/folders/1DemKMAD3D0Ka1MeESRoVCJipSrwiUIEs> (lifecycle in "policies/supporting docs" folder, gaps in "planning" folder, compliance specific in "compliance folder")
- Git repositories
 - <https://github.com/riscv/riscv-compliance/>
 - <https://riscv.readthedocs.io/en/latest/index.html> (riscv docs)
 - <https://gitlab.com/incoresemi/riscv> (riscv framework)
 - https://gitlab.com/incoresemi/riscv-compliance/riscv_ctg (test generator)
 - <https://github.com/riscv/riscv-config/> (config YAML, WARL handling)
 - <https://github.com/rem-s-project/sail-riscv/> (Sail formal model)
- JIRA: <https://jira.riscv.org/projects/CSC/issues/CSC-1?filter=allopenissues>

Meeting Agenda

0. **Looking for more admins, maintainers for riscv-compliance git repo !!**
- I. Updates, Status, Progress
- II. Continued Discussion:
 1. Data propagation report
 2. Merging new Base ISA : remaining issues (Nov 15 review deadline)
- III. Next steps and Ongoing maintenance
 1. Migration to Framework v.2. video: <https://youtu.be/VIW1or1Oubo>, slides: <https://lists.riscv.org/g/tech-compliance/files/Presentations/TestFormatSpec.pdf>
 - What steps do we need to complete to cut over to V.2 (see slide 12)
 - (e.g. ~~Sail model updates~~, pipecleaning, N people have run it, testing all the “fixed in riscov” issues
 - Review Pipecleaner tests: What do we need to do to exercise capabilities for Priv Mode tests
 2. Moving all model proposal to remove all model specific support code out of riscv-compliance repo
 3. Specific Compliance Policy/Process Gaps:
 1. Develop SAIL maintenance plan
 2. Certifying passing architecture tests: what needs to be in the report? Where does report get sent? (e.g. vendorID/archID)
 3. Can we certify actual HW if only its core RTL has passed architecture tests?
 4. How do we enable configurable & licensed core IP
 5. Identify Tool providers, e.g. coverage model, test generation for new features/extensions
 6. Flesh out test development order & identify resources (e.g. Priv,FDD or F,Priv,D..., JIRA CSC-3,5
 7. Provide a reference RTL test fixture (as opposed to SW functional model). See. JIRA CSC-6

Discussion

Updates:

RFQ repo

- Makefile has updated target names: build, run, verify, clear
- RISC_DEVICE docs updated to clarify that it is a single device extension, not a list
- TestFormatSpec updated with macro definition and label fixes, where they must be used, example code fixed to match updated spec, signature now always starts at XLEN bit boundary
- redundant/temporary links/folders removed
- trap handler bugs fixed (not yet used for base ISA tests)
- regdump disabled, removing false mismatches
- SL{TE}[U] and Bxx coverpoints augmented to remove coverage hole
- redundant .align removed that caused mismatch in some cases

Users:

Inspire: can get tests to compile and run in his (Inspire) environment. Had problems with wrong number of zeroes at the beginning of tests.

Incore knows about this; had to do with Spike required alignment. Now fixed

Co-chair: were you able to compile and run JAL instruction?

Inspire: yes, compiled, haven't tried to run

Data propagation report: **Incore** presenting.

(presentation added to <https://lists.riscv.org/g/tech-compliance/files/Presentations>)

slide 2: Definition: What is a data propagation report? (DPR)

slide 3: What a DPR is not?

slide 4: Statistics the DPR includes

#instructions hitting unique coverpoints that update signature
#instructions hitting non-unique coverpoints that update signature (redundant)
#instructions hitting unique coverpoints that don't update signature (bad)
#multiple signature updates

(e.g. trap operands, ops that have non-register dependencies like FP rounding modes)

times a signature was overwritten (bad)

slide 5: How is report generated?

Risc-V ISAC tool parse execution of test running on golden model

Trace is filtered by code inside the test region

Trace is triggered by coverpoint hit

trace from that point until stores to signature area

check if coverpoint has been previously found or not & dump trace & coverpoint

also check if coverpoint hit without the store

slide 6/7: examples

Inspire: restricted to Spike and Sail models?

No; front and back ends. Traces from other models can be substitutes using custom pre-processor

Enhancement requests:

Inspire Can we correlate stat3 report to line of code in .S file?
(discussion of various methods)

1. **Inspire**: wants a "printf" debug macro, will submit a pull request. Consists of a new macro. Prints pass/fail, details and keeps going (or stops)
 - (my) preference: augment the assertion macro with a parameter that selects:
 - disabling assertions
 - print matches & mismatches, and continue past mismatches,
 - print matches & mismatches, & halt upon finding mismatch,
 - print only the mismatches, and continue past mismatches,
 - print only the mismatches, & halt upon finding mismatch

discussion of how to populate the "correct" values in assertions, since they are not available until the Sail model generates them – how to back annotate?

(e.g. post-process Sail reference file, converting each output value to a unique macroname associated with the test case & #include that file)

2. **Co-chair**: wants a way so object dumps can be more easily correlated with .S file: insert unique label at each test case, turn on object dump so that assembly line shows up (GCC with -E flag?). (option needs to be documented where to set)

3. **Marc**: want dummy linker script / model inclusion template in repo-> AR: **Inspire** will provide one.

4. **Co-Chair**: wants to prevent upstreaming of custom model collateral from happening? Discussion was to use .gitignore, which would prevent that unless deliberately overridden. Repo maintainers could check this isn't done if necessary

5. **Co-chair**: wants script to stop at first compile error (in process)

Chair: In <2 weeks, we have a hard stop for accepting new changes to the RFQ tests repo to get them fixed before the Dec. summit when they will be merged:

<https://github.com/incoresemi/riscv-compliance/tree/dev>

If you find issues, github issues must be filed before Nov 15.

Linker script to separate data from text sections

Below is a snippet from this file with explanation as to what it does.

```
MEMORY {
sram (rwx) : ORIGIN = 0x0EE90000, LENGTH = 0x4000
maskrom_mem (rx) : ORIGIN = 0x2EFC0000, LENGTH = 0x20000. }

SECTIONS {
.text.init ALIGN((ORIGIN(maskrom_mem) + 0x0), 64) :
    AT(ALIGN((ORIGIN(maskrom_mem) + 0x0), 64)) {
    PROVIDE(_ftext =.);
    *(.text.init)
    PROVIDE(_etext =.); }

.text ALIGN((ADDR(.text.init) + SIZEOF(.text.init)), 64) :
    AT( ALIGN((LOADADDR(.text.init) + SIZEOF(.text.init)), 64)) {
    *(.text)
    }

.tohost ALIGN((ORIGIN(sram)), 64) :
    AT( ALIGN((LOADADDR(.text) + SIZEOF(.text)), 64)) {
    *(.tohost)
    }

.data ALIGN((ADDR(.tohost) + SIZEOF(.tohost)), 64) :
    AT(ALIGN((LOADADDR(.tohost) + SIZEOF(.tohost)), 64)) {
    *(.data)
    }

PROVIDE(_data = ADDR(.data));
PROVIDE(_data_lma = LOADADDR(.data));
PROVIDE(_edata = ADDR(.data) + SIZEOF(.data));
```

Each section has is defined either to start at a region in the MEMORY area via the ORIGIN(). This is the area the code / data expects to be at runtime. So for the .text.init section this address is 0x2EFC0000 and for the .tohost section it is 0x0EE90000.

The section that dictates how the elf and binary derived from the eld is packaged is the AT() keyword. This tells the linker where the load address for a section is. For the elf file this would be where the elf loader would load this section to. It also dictates where the section is physically in the elf file. This is used to build up the ROM image.

In the above .text.init is the first section and expects to be loaded at 0x2EFC0000. The next section .text follows this in the elf image at 0x2EFC0000 + sizeof(.text.init). In the same way each of the next sections follows the previous section.

In the above you can see the next section .tohost has a different runtime 0x0EE90000 address than its load address of LOADADDR(.text) + SIZEOF(.text).

The linker script file is used to setup a runtime address and a load address for each section. Sometimes, in the case of .text and .text.init these would be the same address. In other cases .tohost, .data and .data.strings they are different.

The linker script file can also export variables back into the code to provide the starting and ending address of a given section. Above you see the .data section is start and end execute addresses and its load address are exported as variables _data, _edata and _data_lma.

In the code you can then use these addresses to copy from the load address to the execute address.

```
la t0, _data_lma; \
la t1, _data; \
la t2, _edata; \
1: \
lw t3, 0(t0); \
sw t3, 0(t1); \
addi t0, t0, 4; \
addi t1, t1, 4; \
bltu t1, t2, 1b;
```

You can specify that all sections get loaded and executed from the same address space using this framework as well. In this case all the load addresses would equal the execute addresses. You would not need the copy code in this case.

Decisions & Action Items

Decisions

Outstanding Action Item

NEW

Co-Chair: Put dummy link script / model inclusion script into repo <new>

Old

Chair : get contact info for all model maintainers and inform them removal of model specific support code from the rep <not done>

Chair/Incore: update test format spec with macro, entry, exit changes <done>

Co-Chair: submit ticket issue RE: doc updates for splitting compile/run <done>

QC: extract bits of FAQ as guidelines for test writing <?>

Incore: add text to current github Readme pointing to riscv <done>

Incore: Try YAML version of SAIL to see if it works <not done>

[everybody]: comment on base ISA cover points:

<https://github.com/incoresemi/riscv-compliance/tree/dev/coverage>

(this is needed to complete the TG's responsibilities for the RFQ)

Imperas: make pull request for updated assertion macro

SH: write up coverage taxonomy

Everybody: read policy docs, send gaps in compliance (e.g. formal model support, possible mismatch between config TG and riscv-config) and priority to cto@riscv.org

Architectural Test Rationale – Intent and Limits

RISC-V Architectural Tests are an evolving set of tests that are created to help ensure that SW written for a given RISC-V Profile will run on all implementations that comply with that profile.

These tests also help ensure that the implementer has both understood and implemented the specification.

The RISC-V Architectural Tests test suite is a minimal filter. Passing the tests and having the results approved by RISC-V International is a prerequisite to licensing the RISC-V trademarks in connection with the design.

Passing the RISC-V Architectural Tests does **not** mean that the design complies with the RISC-V Architecture. These are only a basic set of tests.

The RISC-V Architectural Tests are **not** a substitute for rigorous design verification; it is the responsibility of the implementer to deploy extensive testing.

To be added to the `riscv/riscv-compliance/doc/` directory as “RISC-V Architectural Test Rationale”

Test Acceptance Criteria – second cut

Tests must:

- conform to current standard of test spec (macros, labels)
- run in framework
- run in SAIL and not fail any tests
- Report that test results propagate to signature
- generate a valid signature using SAIL (that can be saved & compared with another DUT/sim)
- has a clear configuration - i.e. which ISA extension it can be used with
- improves coverage
- use only standard instructions (fixed size per architecture LI, LA allowed)
- use only files that are part of the defined support files in the repository
- must be commented test_case header

Framework Requirements – first cut

The framework must:

- Use the TestFormat spec and macros described therein
 - (which must work - including assertions)
- Choose test cases according to equations that reference the YAML configuration
- Define macro variables that can be used inside tests based on the YAML configuration
- Include the compliance trap handler, & handle its (separate) signature area
- Load, initialize, and run selected tests between two selected models, extract the signatures, compare results, and write out a report file
- Exist in a riscv github repo, with a few than one maintainer.
- Be easy to get running, e.g.:
 - run under a variety of OSES with the minimum number of distro specific tools.
 - Not require sudo privileges
- Maybe: have the ability to measure and report coverage
 - Coverage specification is a separate file
 - Could be a separate app

Pull/Issue Status

Issue#	Date	submitter	title	status	comments
#04	3-Jul-18	kasanovic	Section 2.3 Target Environment	Fixed in RISCOF	
#22	24-Nov-18	brouhaha	I-MISALIGN_LDST-01 assumes misaligned data access will trap		
#40	4-Feb-19	debs-sifive	Usage of tohost/fromhost should be removed		
#45	12-Feb-19	debs-sifive	Reorganization of test suites for code maintainability		
#63	13-Aug-19	jeremybennett	Global linker script is not appropriate		
#78	26-Jan-20	bobbl	RV_COMPLIANCE_HALT must contain SWSIG		
#90	11-Feb-20	towoe	Report target execution error		HW misalign support not configurable
#137	12-oct-20	Ode-jtz	The signature.outputs aren't identical w /references.		
#72	26-Oct-19	vogelpi	Allow for non-word aligned `mtvec`	deferred	needs v.2
#105	22-Apr-20	jeremybennett	Non-standard assembler usage	under discussion	Simple fix
#106	22-Apr-20	jeremybennett	Use of pseudo instructions in compliance tests	under discussion	
#107	22-Apr-20	jeremybennett	Clang/LLVM doesn't support all CSRs used in compliance test suite	under discussion	
#108	22-Apr-20	bluewww	RISCV's `compliance_io.h` fails to compile with clang	under discussion	
#109	06-May-20	Olofk	Swerv fails because parallel make	under discussion	
#115	06-jun-20	adchd	How to support on-board execution?	under discussion	
#116	06-jun-20	simon5656	loss of 64bit test infrastucture	under discussion	
#119	17-jun-20	allenjbaum	Missing RV32i/RV64i test: Fence	Test has been written	Close when test is merged
#125	15-jul-20	ShashankVM	Request to stop hosting closed source code on riscv repo	under discussion	
pull#128	29-jul-20	nmeum	grift: update for new directory structure		Who can review this?
pull#129	31-jul-20	nmeum	sail-riscv-ocaml: Disable RVC extension on all devices not using it		Who can review this?
#132	15-aug-20	davidmlw	Why not just use mepc for mret?	answered	Should be resolved
#135	04-sep-20	MikeOpenHWGroup	Request for a Tag on this Repo	assigned	
#137	27-Oct-20	MarcKarasek	xori on 64 bit compiler needs neg number for 32 bit tests	Fixed	Should be resolved

JIRA Status

Issue#	Date	submitter	title	status	comments
IT-1	27Aug/20	Allen Baum	Need to modify the description of compliance in https://riscv.org/technical/specifications/	done	
IT-4	01/Sep/20	Allen Baum	Add Jira link to TG home pages	In prog	
CSC-1	20/Aug/20	Ken Dockser	Come up with names for the tests suites that we are creating		1 st step done
CSC-2	20/Aug/20	Ken Dockser	Produce concise text to explain the Architecture Tests intent and Limits		Written, needs pull req
CSC-3	20/Aug/20	Ken Dockser	Come up with an internal goal for what we wish to accomplish with the Architectural Tests		Not written
CSC-4	20/Aug/20	Ken Dockser	Develop a roadmap for all the different categories of test suites that will need to be created		Not written
CSC-5	20/Aug/20	Ken Dockser	Develop a roadmap for releases of single-instruction Architecture Tests		Not written
CSC-6	20/Aug/20	Ken Dockser	Develop a reference RTL test fixture that can stimulate and check the CPU under test		Needs more discussion