[RISC-V] Can non-standard/custom RISC-V ISA extensions be supported?

2026-06-24 - Comments (0) - RISC-V

The RISC-V ISA specification defines “ISA extensions”, which can complement a base ISA with certain additional instructions. In addition to the standard ISA extensions, the ISA specification also standardizes a way that allows core designers to define own non-standard / custom RISC-V ISA extensions.

There are two different options how custom, non-standard RISC-V ISA extensions can be supported by our tool:

1. Direct integration into our tool:

In some circumstances, Lauterbach can directly add support for certain custom RISC-V ISA extensions to TRACE32. This mainly affects the disassembler and optionally the instruction set simulator and/or trace support. The support for the custom extensions can then be enabled by e.g. selecting the corresponding CPU entry via the **SYStem.CPU** command. If and under which circumstances such a direct integration into our software can be provided needs to be discussed with us individually.

2. Disassembler plugin via APU API:

Lauterbach provides an API that allows users to load their own custom "disassembler plugin" (via shared library such as *.dll or *.so), in order to extend the Lauterbach disassembler functionality by own non-standard RISC-V instructions of ISA extensions. This API is called "API for Auxiliary Processing Unit" (APU API). More information can be found in the document **API for Auxiliary Processing Unit**

The advantages of using such a plugin are:

- Users can test their ISA extension(s) already in very early stages of their chip and toolchain development and have full control over any changes.
- There is no license arrangement with Lauterbach required.
- It is not necessary to reveal details about the ISA extensions to Lauterbach or any other party (beneficial for confidential extensions).

In your TRACE32 installation directory you can also find generic example files for creating a disassembler extension for the APU API: *<T32-install-dir>/demo/apu/example_apudisass*

In order to support **instruction trace** decoding in combination with custom instructions, the APU API allows marking program flow control instructions with certain jump flags and allows defining jump target addresses. See restrictions mentioned below.

Please be aware of the restrictions of using a RISC-V APU API disassembler plugin:

- The APU API is not supported by the TRACE32 instruction set simulator. Lauterbach does provide an API for the simulator, however this API only allows to simulate periphery behavior (**API for TRACE32 Instruction Set Simulator**) and not behavior of a whole core.
- If there are custom instructions with more complex program flow control behavior (for example, if you add a new custom branch instruction), then it needs to be discussed individually if and how trace support can be covered by the generic interface of the APU API.

- Tags
- RISC-V