RTEMS on RISC-V

Sebastian Huber

embedded brains GmbH & Co. KG

2022-12-14

rtems@embedded-brains.de

Overview

Topics of this Presentation

- What is RTEMS?
- RTEMS on RISC-V: History and Status
- RTEMS on RISC-V vs. ARM, PowerPC, SPARC
- RISC-V Shortcomings
- Summary

What is RTEMS?

RTEMS - Real-Time Executive for Multiprocessor Systems

- No user-space/kernel-space separation
- Single address space, multiple threads
- Programming languages: C, C++, OpenMP, Ada
- Architectures: ARM, MicroBlaze, Nios II, RISC-V, PowerPC, SPARC, ... (32 and 64 bit)
- Support for Symmetric Multiprocessing (SMP)
- Scalable to Non-Uniform Memory Access (NUMA) systems in principle (clustered scheduling)
- Open-source with permissive license
- Qualification Data Packages (QDP) for projects using ECSS software development standards available from embedded brains

RTEMS on RISC-V

History

- 2010 RISC-V project start at the University of California, Berkeley
- 2015 RISC-V Foundation
- 2017 RTEMS port by Hesham Almatary (University of York)
- 2018 SMP support for RTEMS by embedded brains
- 2022 Qualification Data Packages (QDP) available from embedded brains for RISC-V PLIC/CLINT systems (for example Gaisler NOEL-V, Microchip PolarFire)

Status

RTEMS is ready for flight on RISC-V. Just ask for a QDP.

RTEMS on RISC-V vs. ARM, PowerPC, SPARC (1)

Interrupt Entry/Exit

The interrupt entry/exit is a straight forward save/restore of volatile registers with a switch to an interrupt stack similar to ARM and PowerPC:

- Save 18 general purpose registers
- Save 20 floating-point registers and the `fcsr`
- Switch to interrupt stack, do some house keeping, call the high level handlers
- Restore 20 floating-point registers and the `fcsr`
- Restore 18 general purpose registers
- Return from interrupt

There is no fancy stuff such as processor modes and register windows. Needs about 132 instructions due to a lack of load/store multiple register instructions. On ARMv7-AR we have 57 instructions (including VFP support).

RTEMS on RISC-V vs. ARM, PowerPC, SPARC (2)

Context Switching from one Task to Another

The context switching is a straight forward save/restore of non-volatile registers similar to ARM and PowerPC:

- Save 14 general purpose registers
- Save 12 floating-point registers and the `fcsr`
- In SMP configurations, do some extra work to hand over the context
- Restore 12 floating-point registers and the `fcsr`
- Restore 15 general purpose registers (including `tp`)

Needs about 65 instructions and no branches (excluding SMP support). No privileged instructions are used (interrupt enable/disable status is not thread-specific). The code is simpler compared to SPARC since there are no register windows (50 + used windows $\times$ 17 instructions; all floating-point registers are volatile). On ARMv7-AR we have 15 instructions (including VFP support).

RTEMS on RISC-V vs. ARM, PowerPC, SPARC (3)

Thread-Local Storage

The thread-local storage support is quite efficient through the use of a dedicated general-purpose register: `tp`. A load/store of a thread-local object needs just one instruction more compared to accessing a global object.

Differences to SPARC

- RISC-V has no register windows: more predictable timing
- RISC-V systems are usually little endian, the Gaisler SPARC processors are big endian: could be an application porting issue

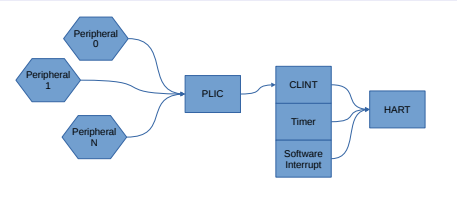
RISC-V vs. Interrupt Priorities

Use Case

Allow interrupt nesting, high priority interrupts can interrupt lower priority interrupts.

RISC-V Shortcoming

The machine timer and software interrupts bypass the PLIC priority handling. These interrupts would usually have a low priority.



Solutions on other architectures

- ARM: Global Interrupt Controller (GIC)
- PowerPC: NXP MPIC
- ...

RISC-V vs. High Priority Interrupts

Use Case

Direct service of high priority interrupts which may **interrupt critical sections of the operating system**. Associated lower priority work may be done by software interrupt under control of the operating system.

RISC-V Shortcoming

There is only one exception handling register set (for M-mode `mcause` and `mpec`). NMI are not recoverable since they overwrite `mcause`.

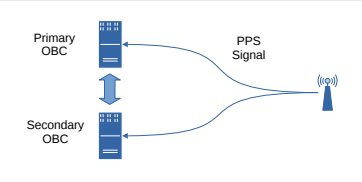
Solutions on other architectures

- ARM: Fast Interrupt Exception (FIQ)
- PowerPC Book E: Critical Input Interrupt
- Nios II: Shadow register sets
- ...

RISC-V vs. Time Synchronization

Use Case

Synchronize internal time with an external reference clock, for example two reduntant OBCs with hot standby.



RISC-V Shortcoming

The machine timer of the CLINT has no counter capture mechanism triggered by an external signal, for example a pulse per second (PPS) signal.

What is needed?

There should be platform-specific support to capture the machine timer value at externally triggered events.

Solutions on other architectures

- Gaisler IRQ(A)MP interrupt timestamping
- ...

RISC-V vs. Double Word Atomic Operations

Use Case

Want to use atomic counters which never overflow: 64-bit integers. Example: gcov code coverage control flow edge counters; transaction identifiers; ring buffer indices.

RISC-V Shortcoming

The 32-bit RISC-V ISA (RV32IMA) does not support 64-bit atomic operations.

Solutions on other architectures

- ARM: `ldrexd` and `strexd` instructions which use two 32-bit destination/source registers
- ...

Finally

RISC-V Summary

- RISC-V has solid foundations
- There is an aspiring and growing ecosystem
- Some rough edges need to be polished, for example the interrupt handling and time synchronization could be improved

What can we at **embedded brains** do for you?

- RTEMS training and workshops
- RTEMS development to support new architectures, chip families, system on chips, boards
- Device driver development
- Porting of third-party libraries and applications to RTEMS
- Consulting to select the right hardware for your project
- Delivery of Qualification Data Packages (QDP) for RTEMS on your hardware
- Consulting and support to do ECSS conforming software development

Questions?

...