Port luajit to RISC-V
Motivation, first steps and perspectives

Anton Kuzmin

2020-02-01
Who am I...

- not really a software developer
  ...but write code sometimes
- developing embedded systems for 25 years
- VME, CompactPCI, AdvancedTCA, SoM
- FPGA and SoC-FPGA (Altera/Intel, Microsemi/Microchip)
- VHDL (RTL-code, no, it is not a software)

My usual problem with the software is how to make it run on a hardware which is known not to be working yet and how to bring-up and test this hardware. With a soft-core CPU it is getting even worse.
Intro

Motivation
  Why RISC-V
  Why Lua
  Why luajit

luajit

Status

What’s next...

Contact info
Why RISC-V?

RISC-V ("risk-five") is an open source Instruction Set Architecture (ISA) specification

- open-source, royalty-free
- scalable from 32-bit MCU with reduced number of registers (rv32e) to multi-core 64-bit OoO CPU (and even 128-bit)
- proven on FPGAs and silicon tape-outs
- modular design with both standard extensions and reserve encoding space for custom extensions
- already used in products as a soft-core MCU on FPGAs, SoC-FPGA with hard-core RISC-V CPU is coming later this year
Why Lua?

Lua is a powerful, efficient, lightweight, embeddable scripting language. It supports procedural programming, object-oriented programming, functional programming, data-driven programming, and data description.

- simple
- interactive
- very compact and portable
- simple interface to C and, through it, to an underlying hardware
- fast (if it really matters)

Contemporary replacement for Forth. First tried on NIOS II for optical link transceiver testing and training in 2012.
Why luajit?

LuaJIT is a Just-In-Time Compiler (JIT) for the Lua programming language.

• **dynamic assembler**
  • for experiments with CPU extensions and hardware accelerators
  • gcc/binutils/llvm are well beyond my comprehension and are not suitable for rapid experiments directly on hardware

• used in various interesting projects
  • LuaRadio (https://luaradio.io/)
  • Tarantool DB (https://www.tarantool.io/en/)
  • luapower (https://luapower.com/)

• **why not**
  • the project seems to be not actively developed and maintained since 2017
  • several [non synchronized] forks
  • stick to Lua 5.1
luajit components

- dynasm
- Lua VM (hand-optimized assembler)  
  the best [hardest] way to learn RISC-V ISA
- tracing JIT
- Garbage Collector
- extensions: FFI, bit-ops, hard & soft Floating Point support
Current status

- running Lua 5.1 on spike and soft-core RISC-V CPU
- git repository (fork of luajit v2.1)
- Makefile changes
- rv32i base instruction encoding
- registers/immediates/labels – WIP
- everything else – TODO
What to do

• write [all] the code:
  • Lua VM on RISC-V assembler
  • GC
  • soft and hard FP support
  • tracing JIT

• tests

• benchmarks
Perspectives

Interesting stuff to do, when everything above is running

• RISC-V J extension – what language will be the first to shape it?
  • Java
  • JavaScript
  • C#
  • what else?...

• hardware-assisted hot trace detection

• hardware-assisted memory management/garbage collection

• custom ISA extension (accelerator) – development and testing

• to write yet another Forth on dynasm
  (then to write yet another assembler on Forth)
Thank you!

Anton Kuzmin
anton.kuzmin@cs.fau.de
https://github.com/ak-fau/

Questions?..