Technische Universität München

3D Graphics on RISC-V

First Vulkan Graphics Driver for RISC-V CPUs

Martin Troiber - m.troiber@tum.de

Introduction

Summary
- Port of SwiftShader for RISC-V CPUs
- 3D Graphics on RISC-V without GPU
- Potential use cases:
  - Rendering on server CPUs
  - Rendering on low cost SBCs

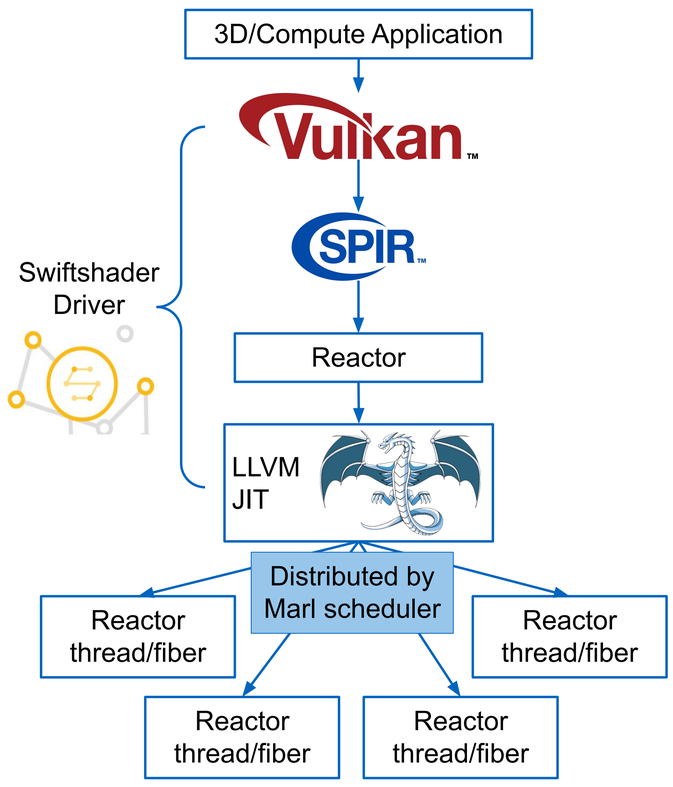
Motivation
- Inner workings of GPUs inaccessible
- 3D Graphics on open-source SoC

Vulkan
- Low level graphics API
- Graphics application split into shaders
- Shaders pre-compiled to **SPIR-V**

Swiftshader
- Vulkan driver targeting the CPU [1]
- Designed as GPU fallback
- Maintained by Google
- Considered alternatives:
  - Kazan [2] incomplete implementation
  - LLVMpipe[3] tied to large Mesa stack

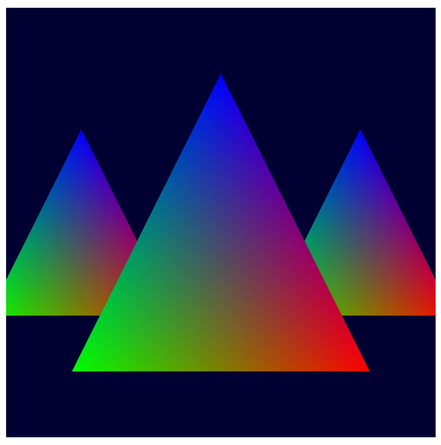
Reactor
- SwiftShader's internal representation
- Utilizes run-time specialization
- Reduces binary size

SwiftShader Architecture

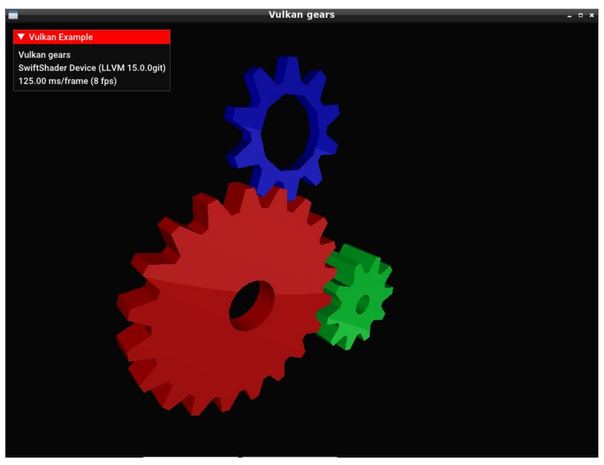


Evaluation

Headless Rendering [7]



Framebuffer Rendering



Gears scene rendered at 1080p resolution

Performance by target platform
Avg. values after a minute of execution

Target	Frame rate	RAM	Compute
QEMU[a]	1-4fps	120MB	80%-90%
Nezha[b]	2fps	77MB	80%
X86[a]	180fps	33MB	48%
GPU[c]	2600fps	60MB	56%

[a] AMD Ryzen 7 PRO 4750U with 8x1.7-4.1Ghz
[b] Allwinner D1 with 1x1.0Ghz
[c] AMD Radeon RX Vega 7 with 1.6Ghz

Scaling by number of QEMU cores
Avg. values after a minute of execution

Cores[a]	Frame rate	RAM	Compute
1	1fps	120MB	90%
2	2fps	120MB	90%
4	3fps	120MB	85%
6	4fps	120MB	83%
8	4fps	120MB	80%

[a] AMD Ryzen 7 PRO 4750U with 8x1.7-4.1Ghz

Swiftshader on RISC-V [5]

No changes to Vulkan, SPIR-V, Reactor

LLVM JIT (Just-in-time compiler) [4]
- Compiles Reactor programs
- Places binaries in memory
- Experimental RISC-V support

Changes to LLVM JIT
- Updated LLVM to version 16
- Better support for RISC-V relocations
- Decoupled SwiftShader - LLVM version
- Switched linking layer for JIT
- Modern RTDyldObjectLinkingLayer
- Support for RISC-V linking

Marl (Scheduler)
- Distributes binaries from JIT to cores
- Added RISC-V register layout
- Required for context switching

SwiftShader cross-compilation
- Identified compatible configuration
- Reduced time from hours to minutes

Target Platforms

Debian with X11 on both platforms

QEMU (Emulation) [6]
- 1-8 x RV64GC, 8GB RAM
- Host with AMD Ryzen PRO 4750U featuring 8 cores @ 1.7Ghz - 4.1Ghz

Sipeed Nezha (SBC)
- 1 x RV64GC @ 1.0Ghz, 1GB DDR3
- Allwinner D1 SoC (XuanTie C906)
- Features DSP for 2D graphics
- Lacks 3D graphics acceleration

Future Work

- Undergoing upstreaming with Google
- Test scaling on server CPU
- Utilize RISC-V vector extension
- Investigate RISC-V GPU architectures

Technische Universität München
TUM School of Computation, Information and Technology
Department of Computer Engineering

I would like to thank:
Prof. Michael Taylor (University of Washington), Prof. Martin Schreiber (Grenoble Alpes University),
Nicolas Capens (SwiftShader), Lang Hames (LLVM),
Dr. Thomas Wild (TUM), Prof. Andreas Herkersdorf (TUM)

[1] Nicolas Capens. SwiftShader. https://swiftshader.googlesource.com/SwiftShader. 2022.
[2] Jacob Lifshay. Kazan. https://salsa.debian.org/Kazan-team/kazan. 2020.
[3] Alex Fan. Mesa merge requests: Add RISC-V support to LLVMpipe. https://gitlab.freedesktop.org/mesa/mesa/-/merge_requests/17801. 2022.
[4] Chris Lattner and Vikram Adve. "LLVM: A compilation framework for lifelong program analysis & transformation". In: International Symposium on Code Generation and Optimization, 2004. CGO 2004. IEEE. 2004, pp. 75–86.
[5] Martin Troiber. Swiftshader issue tracker: Port Swiftshader to RISC-V. https://issuetracker.google.com/issues/217573066. 2022.
[6] Fabrice Bellard. "QEMU, a fast and portable dynamic translator." In: USENIX annual technical conference, FREENIX Track. Vol. 41. California, USA. 2005, p. 46.
[7] Sascha Willems. Vulkan C++ examples and demos. https://github.com/SaschaWillems/Vulkan. 2015.