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
- Rendering on server CPUs
- Utilizes run-time specialization
- Reduces binary size
- Designed as GPU fallback
- Inner workings of GPUs inaccessible

Considered alternatives:
- LLVMpipe[3] tied to large Mesa stack
- Potential use cases:
  - Port of SwiftShader for RISC-V CPUs
  - Vulkan driver targeting the CPU [1]
  - SwiftShader’s internal representation
  - 3D Graphics on open-source SoC
  - Maintained by Google
  - 3D Graphics on RISC-V without GPU
  - Kazan [2] incomplete implementation
  - Low level graphics API

Swiftshader on RISC-V
- 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

MARI (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

Evaluation

Headless Rendering [7]

Framebuffer Rendering

Gears scene rendered at 1080p resolution

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

<table>
<thead>
<tr>
<th>Target</th>
<th>Frame rate</th>
<th>RAM</th>
<th>Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>QEMU*</td>
<td>1-4fps</td>
<td>120MB</td>
<td>80%-90%</td>
</tr>
<tr>
<td>Nezha[5]</td>
<td>2fps</td>
<td>77MB</td>
<td>80%</td>
</tr>
<tr>
<td>X86[6]</td>
<td>180fps</td>
<td>33MB</td>
<td>48%</td>
</tr>
<tr>
<td>GPU[5]</td>
<td>2600fps</td>
<td>60MB</td>
<td>56%</td>
</tr>
</tbody>
</table>

* AMD Ryzen 7 PRO 4750U with 8x1.7-4.1Ghz
\[5\] Alex Fan. Mesa merge requests: Add RISC-V support to LLVMpipe.

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

<table>
<thead>
<tr>
<th>Cores[5]</th>
<th>Frame rate</th>
<th>RAM</th>
<th>Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1fps</td>
<td>120MB</td>
<td>90%</td>
</tr>
<tr>
<td>2</td>
<td>2fps</td>
<td>120MB</td>
<td>90%</td>
</tr>
<tr>
<td>3</td>
<td>3fps</td>
<td>120MB</td>
<td>85%</td>
</tr>
<tr>
<td>4</td>
<td>4fps</td>
<td>120MB</td>
<td>83%</td>
</tr>
<tr>
<td>5</td>
<td>4fps</td>
<td>120MB</td>
<td>80%</td>
</tr>
</tbody>
</table>

* AMD Ryzen 7 PRO 4750U with 8x1.7-4.1Ghz

3D Graphics on RISC-V [5]

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