Android on RISC-V
Progress & Updates

Lars Bergstrom, PhD
Director of Engineering, Android
Google wants RISC-V to be a “tier-1” Android architecture

Google's keynote at the RISC-V Summit promises official, polished support.

RON AMADEO · 1/3/2023, 3:14 PM
Android is an open source operating system

AOSP is the corresponding open source project led by Google

Documentation and source code needed to build, customize, port to new hardware, and meet compatibility requirements are available at:

https://source.android.com
https://android.googlesource.com
Status of Android on RISC-V
Still no product announcements...

But we're far more ready for your products!
Key areas of progress in 2023

Thanks to the entire ecosystem!

AOSP
- Android Runtime (ART) available
- Cuttlefish emulator available
- Prebuilt tools - compilers & system root libraries available
- Initial support landing soon for extensions beyond RV64GC to optimize the platform
  - Vector
  - Bit-manipulation extension optimizations (Zba/b/s)
- Profiling works (again, prebuilds coming soon)

Open Source Projects
- Contributions and work with ecosystem partners and upstream maintainers on
  - LLVM
  - Kernel
  - QEMU
  - Graphics libraries
  - Crypto libraries
  - Codecs

Upstream at RISC-V International
- Work to ensure the ABI is forwards-compatible with new potential atomics additions
- Support for many RISC-V members on the Android SIG looking into standing up AOSP on a variety of emulation and physical devices
Emulation - Getting Started with Cuttlefish for RISC-V

https://github.com/google/android-riscv64

$ lunch aosp_cf_riscv64_phone-userdebug
$ m -j
$ launch_cvd -cpus=8 -memory_mb=8192

Then, use vncviewer to connect
Emulation - Cuttlefish for RISC-V Roadmap

Today
- Cuttlefish prebuilt bootloader and kernels added to AOSP
- Lunch targets for phone, slim & minidroid added to AOSP
- Builds available from [ci.android.com](http://ci.android.com)
- Phone target can reach boot complete in 8 minutes with QEMU TCG on a fast PC

Soon
- Scalable testing with accelerated GPU + QEMU on server platforms
- Reduced boot time and reduced flakiness

Later
- Scalable testing with host-side SwiftShader and QEMU on GCP
- Hardware virtualization, crosvm support
As on other platforms, either SIMD or vector optimizations are required for efficient string and memory copy, zeroing and permutation options

- memchr, memcmp, memcpy...
- strcat, strcmp, strcpy,...

Optimizations for libm are also available

- fabs, ceil, floor, fmax/fmin, round, etc.
We’re mostly focused on ABI changes at the moment:

- Emulated TLS ([https://reviews.llvm.org/D147834](https://reviews.llvm.org/D147834))
- TLSDESC (change coming soon)
- Future-compatible atomics ([https://reviews.llvm.org/D149486](https://reviews.llvm.org/D149486))

Autovectorization is a top priority

- Required for many libraries such as Skia and benchmark suites such as Geekbench
- Increased complexity with multiple potential implementations and many different cross-ISA considerations
Libraries - libpng

Updated image processing libraries!

But, work remains around helping with upstream CI/testing especially for important optimizations.
Languages - Dart on RISC-V

**Dart: fast apps on any platform**

Dart, with Flutter, powers more than 1M apps in Google Play, e.g.:

- Alibaba, BMW, ByteDance, eBay, Google, Tencent, ...

Dart is

- Memory-safe, garbage-collected
- JIT for dev, AOT for prod: ARM, x86
- Experimental RISC-V on android/riscv64

```
rmacnak@rmacnak:~$ adb shell
$ cd /data/local/tmp
$ uname -m
riscv64
$ cat hello.dart
import 'dart:ffi';
main() {
  print("Hello, ${Abi.current()}!");
}
$ ./dart hello.dart # Check this is a RISC-V device
Hello, android_riscv64!
$ ./dart hello.dll # Run JIT from source
Hello, android_riscv64!
$ ./dart_precompiled_runtime hello.elf # Run AOT from ELF (machine code)
Hello, android_riscv64!
```
The Android ABI for RISC-V
What is Android Compatibility?

Establish an open platform for developers to build innovative applications

Provide a consistent application and hardware environment to application developers.

Enable a consistent application experience for consumers.

Enable device manufacturers to differentiate while being compatible.

Minimize costs and overhead associated with compatibility.

Key point: the Android Open Source Project is free to use and build products and even ecosystems without being “Android-compatible”!
Ensuring Application Compatibility

CDD - Compatibility Definition Document
- Ensures a compatible API surface for application developers

CTS - Compatibility Test Suite
- Validate Android compatibility requirements (CDD)
  - CTS: the primary, automated test suite
  - CTS Verifier: for manual tests which cannot be automated (minimize wherever possible)
- Open sourced; develop and release per API level
- Essential tools Google uses to approve partner device launch
- **Must pass this** to be considered “Android-compatible”
### Additional test suites

<table>
<thead>
<tr>
<th>Test Suite</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATS</td>
<td>Required for Automotive partners to verify compliance.</td>
</tr>
<tr>
<td>BTS</td>
<td>Security scans on preloaded system apps and system image.</td>
</tr>
<tr>
<td>GTS</td>
<td>Google Mobile Services &amp; look / feel validation.</td>
</tr>
<tr>
<td>ITS</td>
<td>Image Test Suite</td>
</tr>
<tr>
<td>MTS</td>
<td>Mainline test suite.</td>
</tr>
<tr>
<td>STS</td>
<td>Security test suite.</td>
</tr>
<tr>
<td>TVTS</td>
<td>Required for Android TV partners.</td>
</tr>
<tr>
<td>WTS</td>
<td>Required for Wearable partners.</td>
</tr>
<tr>
<td>VTS</td>
<td>Required for hardware / chipset validation.</td>
</tr>
</tbody>
</table>
Supported ABI will be added to the CDD list per top-right ("riscv64", with no 32-bit equivalent)

Will be linked to the descriptive text in the NDK Supported ABIs

- All “supported instruction sets” will be a combination of
  - A RISC-V profile (probably RVA22)
  - Ratified extensions (probably vector + vector crypto)
  - Intentional omissions: SIMD, Scalar Crypto

Will require Android-compatible devices to be conforming hardware

- Must correctly implement the RISC-V ISA
- Must not misuse elements of the encoding space reserved for future extensions

Platforms (but not applications!) can take advantage of RISC-V features in the reserved vendor space
Looking to the future
RISC-V Android ABI Progress and Wishlist

See our current progress here: https://github.com/google/android-riscv64
Known issues here: https://github.com/google/android-riscv64/issues
Join the Android SIG mailing list and come to the monthly meetings for more: https://lists.riscv.org/g/sig-android

What’s next after “rva22 + vector + vector crypto”?

First: need to make sure to land vector crypto!
  ● Still haven't voted on ratification at time of writing (https://github.com/riscv/riscv-crypto/releases)

Very excited for platform support for the following extensions, but unclear if it’s required for Android applications as well...
  ● Zjid instruction/data consistency for JIT
  ● Zisslpcri for security
  ● Zjpm pointer masking for hwasan
  ● Hans Boehm’s proposed new atomics
  ● bfloat16 vector support
The road ahead for AOSP and RISC-V

Continuing to build out features & performance

<table>
<thead>
<tr>
<th>Q3 2023</th>
<th>Q4 2023</th>
<th>2024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual devices with accelerated graphics</td>
<td>NDK ABI finalized &amp; canary builds available on Android’s public CI shortly thereafter: <a href="https://ci.android.com/builds/branches/aosp-master-ndk/grid">https://ci.android.com/builds/branches/aosp-master-ndk/grid</a></td>
<td>Emulators available publicly, with full feature set to test applications for various device formfactors</td>
</tr>
<tr>
<td>Android Runtime (ART) optimizations for both the fast interpreter and precompiled code</td>
<td>RISC-V on x86-64 &amp; ARM64 available for easier testing of riscv64 Android applications on a host machine</td>
<td>Released NDK contains RISC-V support</td>
</tr>
<tr>
<td>Optimizations landing for QEMU, kernel, and all in-tree libraries (including use of bitmask &amp; vector operations)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Security is a key area where we are looking to collaborate more

- How do we help secure & isolate the tens of components on the SoC from each other and other workloads?
- Memory safety issues and side channels heavily affect code, especially native - how can we isolate it?

Several technologies we are very interested in

- TEE
  - How do we protect the execution of privacy and security-sensitive operations?
- WorldGuard
  - Can we isolate some of the hardware components from each other more rigorously?
- CHERI
  - Software compartmentalization via processes is one of the highest memory and latency costs on Android!
  - Are there hardware mechanisms for providing better spatial isolation of memory?
Goal:  Accelerate open source SW for RISC-V architecture

How:  Align on highest priorities & avoid (accidental) duplication of work
Focus Areas

Coordination and collaboration among the RISE members is across an array of software areas to deliver high quality and high performance implementations for RISC-V software.

<table>
<thead>
<tr>
<th>Focus Areas</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compilers &amp; Toolchains</td>
<td>LLVM, GCC</td>
</tr>
<tr>
<td>System Libraries</td>
<td>Glibc, OpenSSL, OpenBLAS, LAPACK, OneDAL, Jemalloc</td>
</tr>
<tr>
<td>Kernel &amp; Virtualization</td>
<td>Linux, Android</td>
</tr>
<tr>
<td>Language Runtimes</td>
<td>Python, OpenJDK/Java, V8</td>
</tr>
<tr>
<td>Linux Distro Integration</td>
<td>Ubuntu, Debian, RHEL, Fedora, Alpine</td>
</tr>
<tr>
<td>Debug &amp; Profiling Tools</td>
<td>Performance profiles, DynamoRIO, Valgrind</td>
</tr>
<tr>
<td>Simulator/Emulators</td>
<td>QEMU, SPIKE</td>
</tr>
<tr>
<td>System Software</td>
<td>UEFI, ACPI</td>
</tr>
</tbody>
</table>
Working Model

RISE is a tool to prioritize and bring more resources to help address gaps

Company A Gap List

Company B Gap List

... 

Company Z Gap List

{Deduplicate + Prioritize}

Compiler & Toolchain Action List

System Libraries Action List

System Software Action List

For each Action, complete work in responsible upstream project (e.g., LLVM)
Examples of RISE Efforts

QEMU for helping test features & prove out ahead of hardware support
- AIA support
- AIA support enhancements - IRQ filtering
- Vector Cryptography support
- WorldGuard support

Simulator/Emulator

Compilers & Tool Chain

- A.7 compatible atomics mappings
Join RISE

RISE is focused on positive and transparent collaborations with upstream projects to deliver commercial-ready software for various use cases.

RISE Membership requires Linux Foundation Europe membership & RISC-V International membership.

We are excited for your team to join this journey!
riseproject.dev
Learn more & contribute

Many ways to participate in Android on RISC-V!

Source Contributions
Visit https://source.android.com/docs/setup/contribute

RISC-V Collaborations
Participate in the Android SIG here at RISC-V International sig-android@lists.riscv.org
https://lists.riscv.org/g/sig-android

Consider Joining RISE
Ensure the software ecosystem is prepared for the products you are bringing to market
https://riseproject.dev/