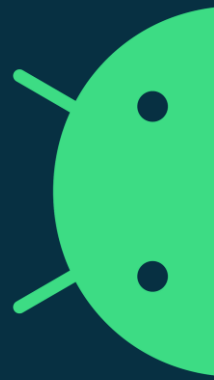
Android on RISC-V Progress & Updates

Lars Bergstrom, PhD
Director of Engineering, Android

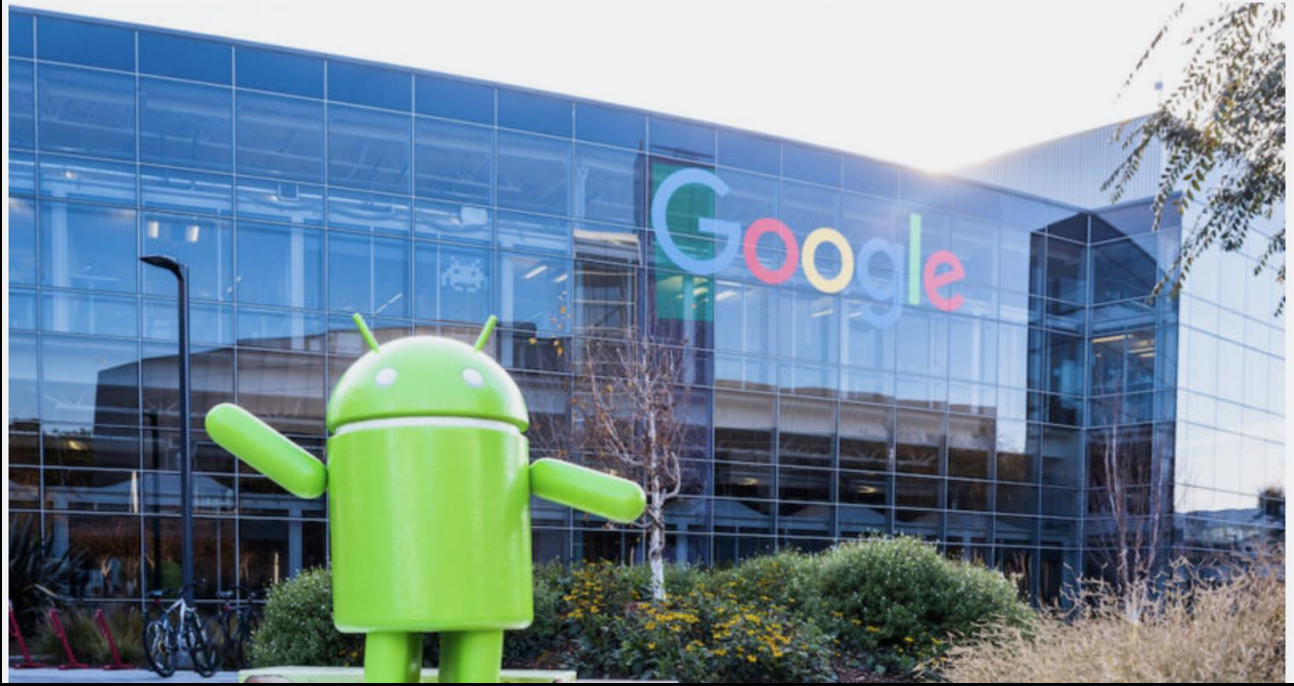


android

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

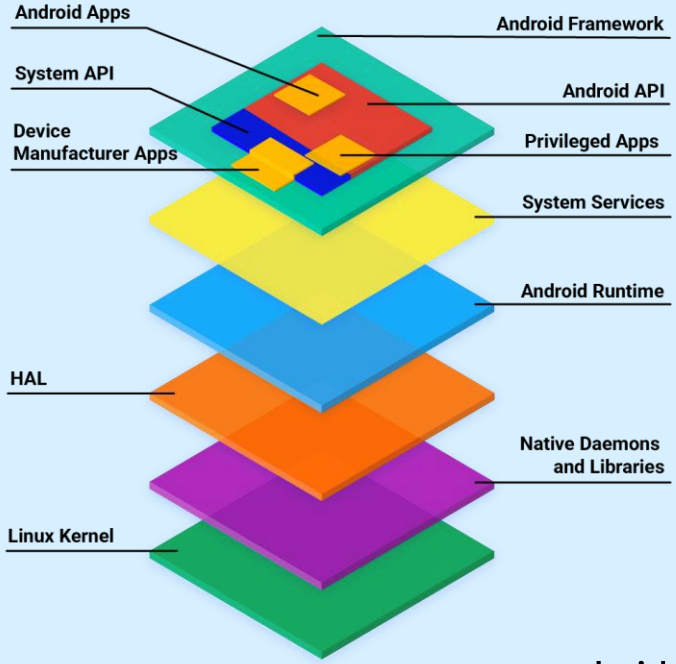


Android is an open source operating system

AOSP is the corresponding open source project led by Google

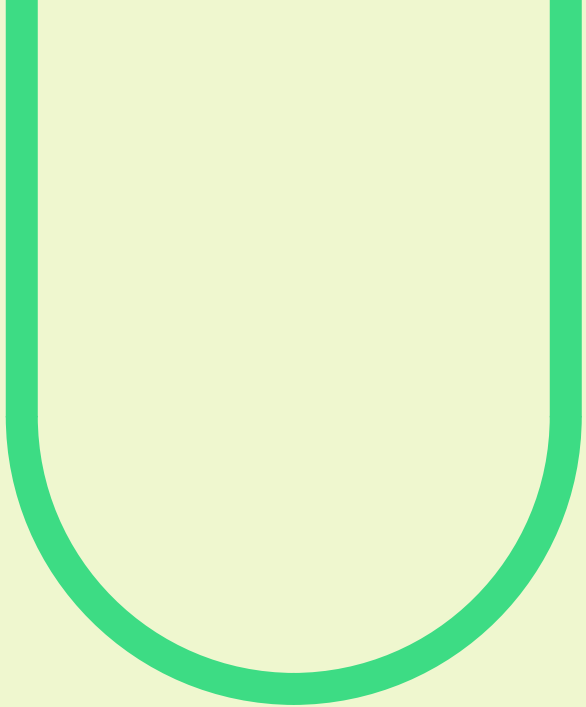
Documentation and source code needed to build, customize, port to new hardware, and meet compatibility requirements are available at:

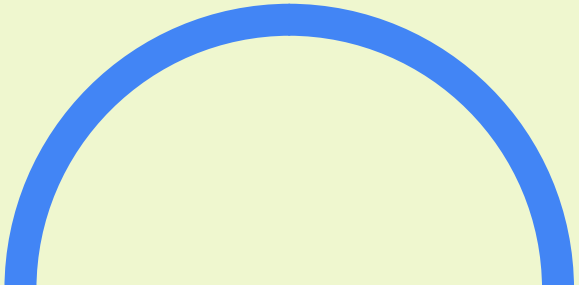
https://source.android.com
https://android.googlesource.com



android

Status of Android on RISC-V





android

Still no product announcements...

But we’re far more ready for *your* products!

android

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, prebuilts coming soon)

Open Source Projects

Contributions and work with ecosystem partners and upstream maintainers on

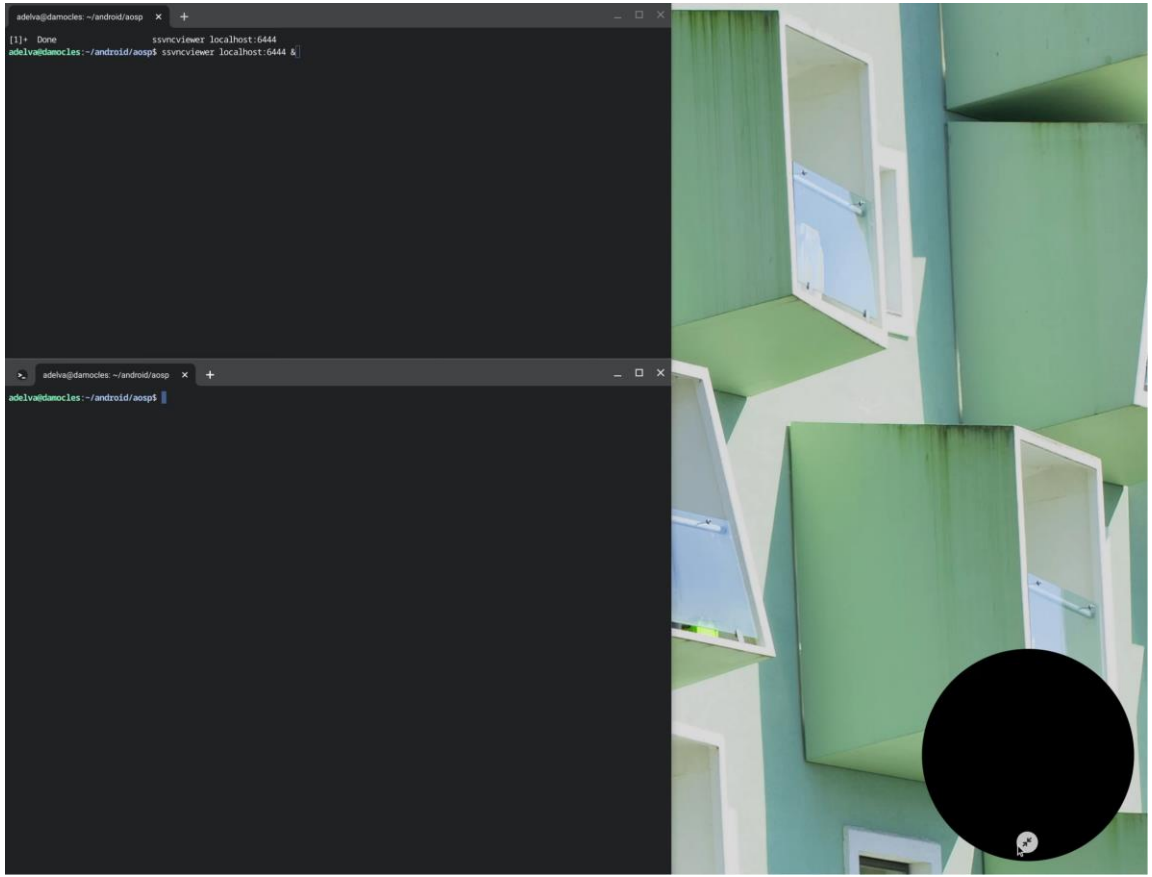
- LLVM
- Kernel
- QEMU
- Graphics libraries
- Crypto libraries
- Codecs

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
```

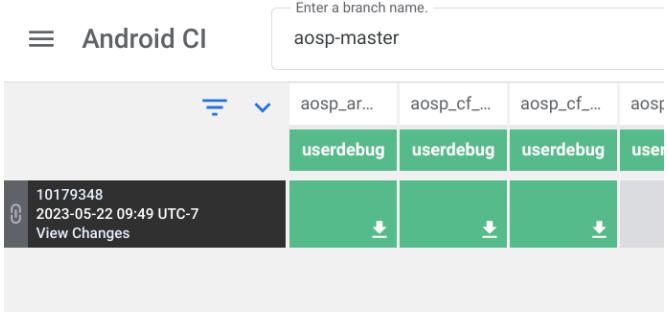
Then, use `vncviewer` to connect

android

Emulation - Cuttlefish for RISC-V Roadmap

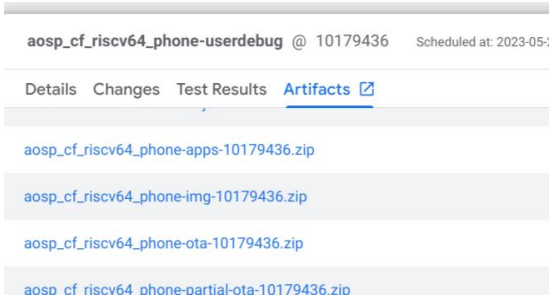
Today

- Cuttlefish prebuilt bootloader and kernels added to AOSP
- Lunch targets for phone, slim & minidroid added to AOSP
- Builds available from ci.android.com
- 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

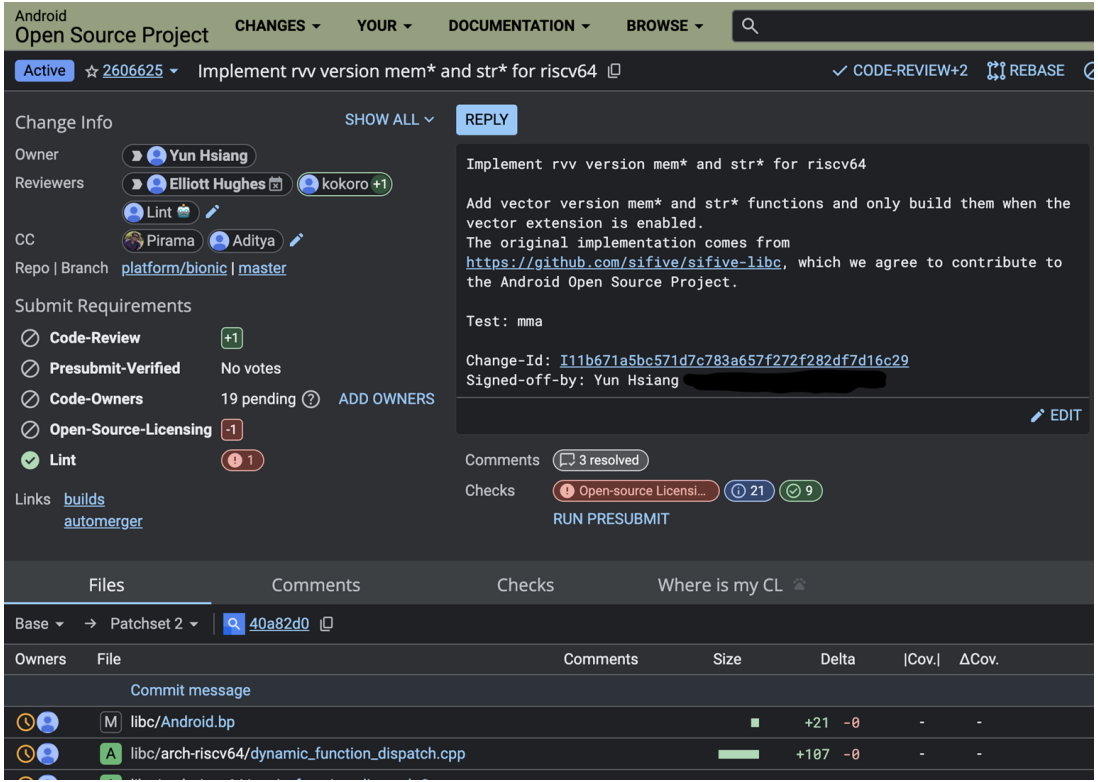
Platform - Bionic optimization

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.



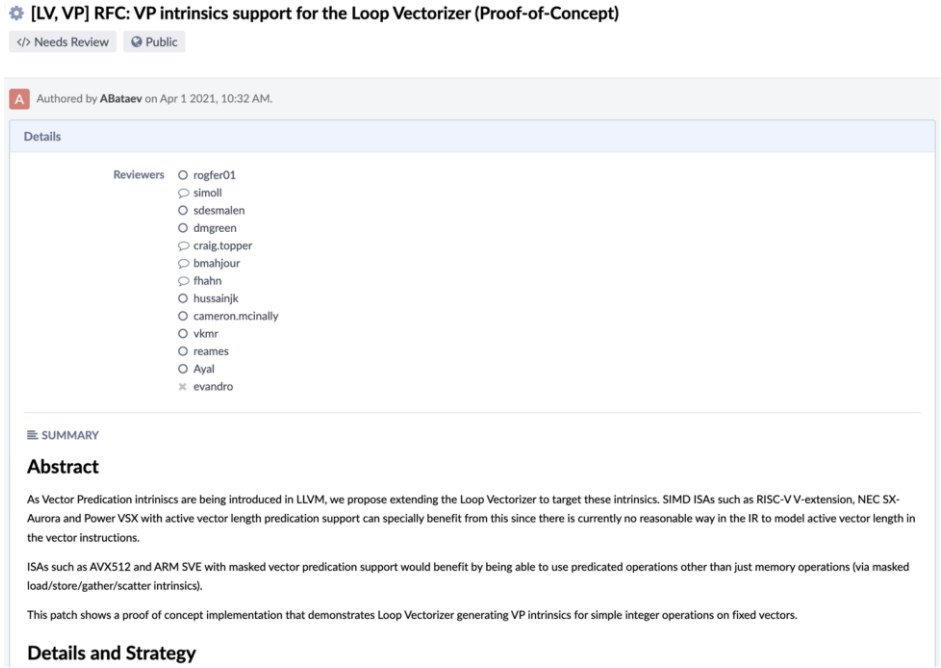
Platform - Toolchain & Compilation

We're mostly focused on ABI changes at the moment:

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



[LV, VP] RFC: VP intrinsics support for the Loop Vectorizer (Proof-of-Concept)

Needs Review · Public

Authored by ABataev on Apr 1 2021, 10:32 AM.

Details

Reviewers
- rogfer01
- simoll
- sdesmalen
- dmgreen
- craig.topper
- bmahjour
- fhahn
- hussainjk
- cameron.mcinally
- vkmr
- reames
- Ayal
- evandro

SUMMARY

Abstract

As Vector Predication intriniscs are being introduced in LLVM, we propose extending the Loop Vectorizer to target these intrinsics. SIMD ISAs such as RISC-V V-extension, NEC SX-Aurora and Power VSX with active vector length predication support can specially benefit from this since there is currently no reasonable way in the IR to model active vector length in the vector instructions.

ISAs such as AVX512 and ARM SVE with masked vector predication support would benefit by being able to use predicated operations other than just memory operations (via masked load/store/gather/scatter intrinsics).

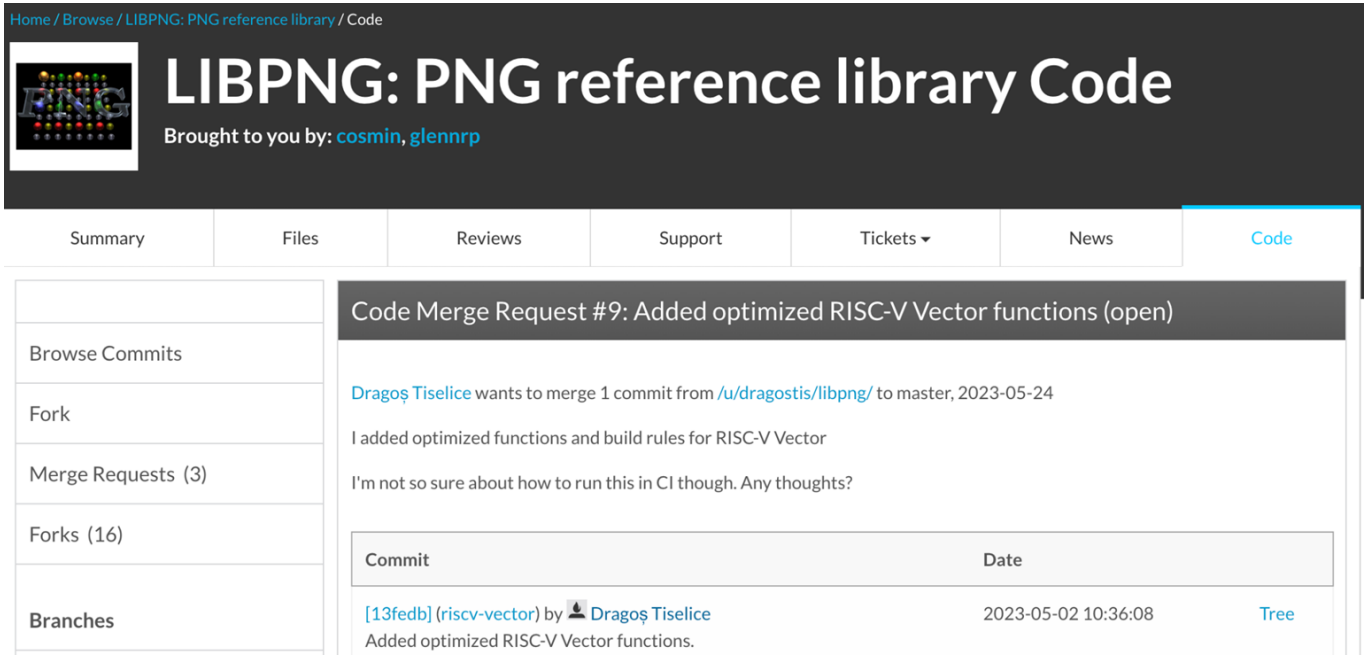
This patch shows a proof of concept implementation that demonstrates Loop Vectorizer generating VP intrinsics for simple integer operations on fixed vectors.

Details and Strategy

Libraries - libpng

Updated image processing libraries!

But, work remains around helping with upstream CI/testing especially for important optimizations.

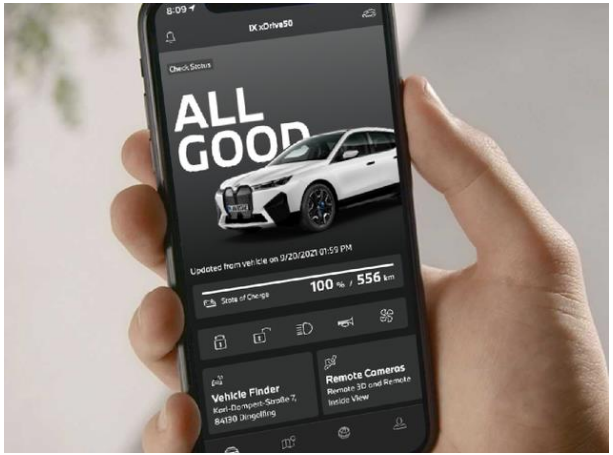


Languages - Dart on RISC-V

***Dart**: fast apps on any platform*

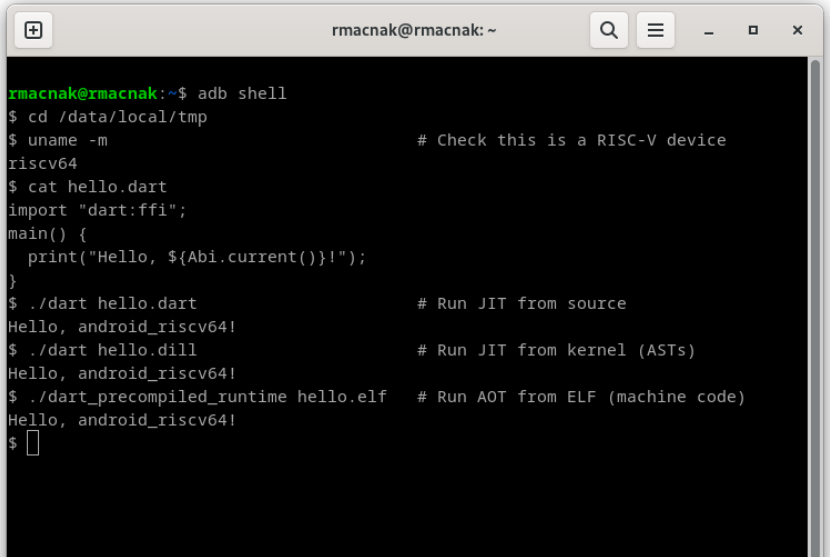
Dart, with Flutter, powers more than 1M apps in Google Play, e.g.:

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

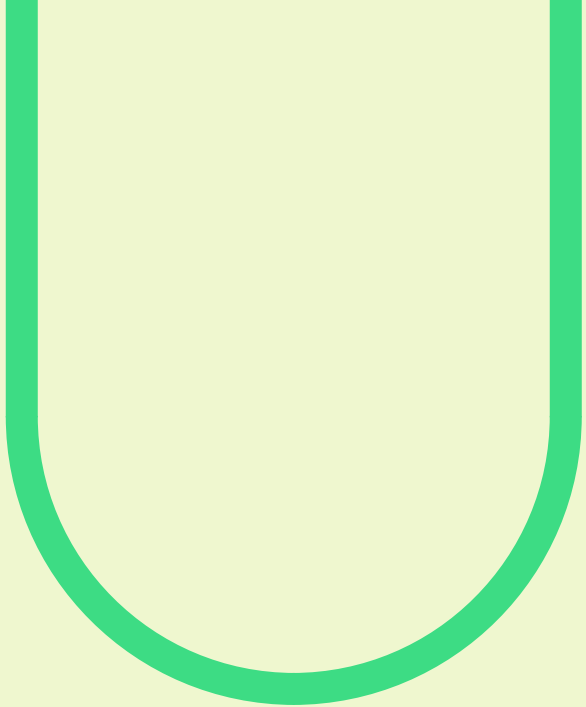


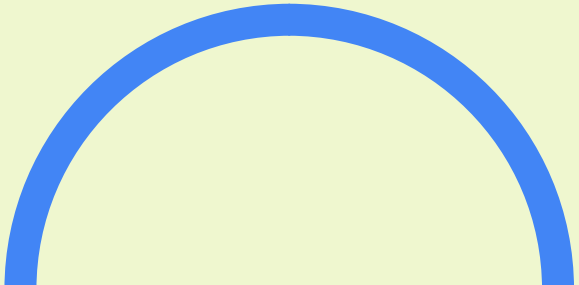
Dart is

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



The Android ABI for RISC-V





android

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

ATS	• Required for Automotive partners to verify compliance.
BTS	• Security scans on preloaded system apps and system image.
GTS	• Google Mobile Services & look / feel validation.
ITS	• Image Test Suite
MTS	• Mainline test suite.
STS	• Security test suite.
TVTS	• Required for Android TV partners.
WTS	• Required for Wearable partners.
VTS	• Required for hardware / chipset validation.

Android Profiles

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

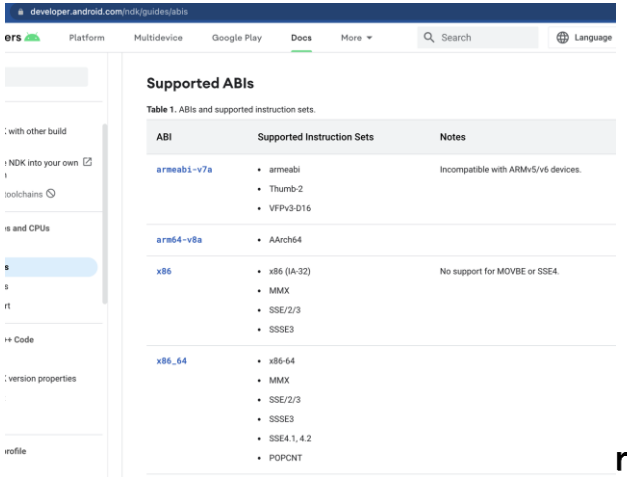
- All “supported instruction sets” will be a **combination** of
  - A RISC-V profile (probably RVA22)
  - Ratified extensions (probably vector + vector crypto)
  - Intentional omissions: SIMD, Scalar Crypto

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

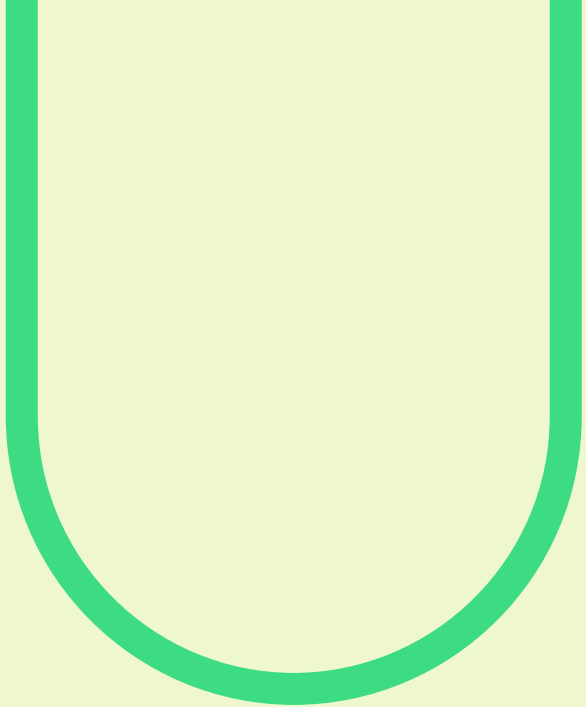
Platforms (but not applications!) can take advantage of RISC-V features in the reserved vendor space

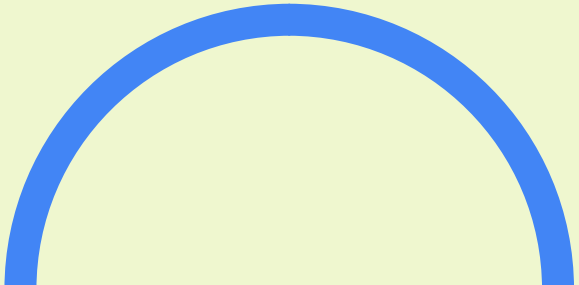
- [C-0-3] MUST be source-compatible (i.e. header-compatible) and binary-compatible (for the ABI) with each required library in the list below.
- [C-0-5] MUST accurately report the native Application Binary Interface (ABI) supported by the device, via the `android.os.Build.SUPPORTED_ABIS`, `android.os.Build.SUPPORTED_32_BIT_ABIS`, and `android.os.Build.SUPPORTED_64_BIT_ABIS` parameters, each a comma separated list of ABIs ordered from the most to the least preferred one.
- [C-0-6] MUST report, via the above parameters, a subset of the following list of ABIs and MUST NOT report any ABI not on the list.
  - `armeabi` (no longer supported as a target by the NDK)
  - `armeabi-v7a`
  - `arm64-v8a`
  - `x86`
  - `x86-64`
- [C-0-7] MUST make all the following libraries, providing native APIs, available to apps that include native code:



roid

Looking to the future





android

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
- Zisslpcfi 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

Q3 2023

Virtual devices with accelerated graphics

Android Runtime (ART) optimizations for both the fast interpreter and precompiled code

Optimizations landing for QEMU, kernel, and all in-tree libraries (including use of bitmask & vector operations)

Q4 2023

NDK ABI finalized & canary builds available on Android's public CI shortly thereafter: https://ci.android.com/builds/branches/aosp-master-ndk/grid

RISC-V on x86-64 & ARM64 available for easier testing of riscv64 Android applications on a host machine

2024

Emulators available publicly, with full feature set to test applications for various device formfactors

Released NDK contains RISC-V support

Upstream at RISC-V International

Collaborating on innovation

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

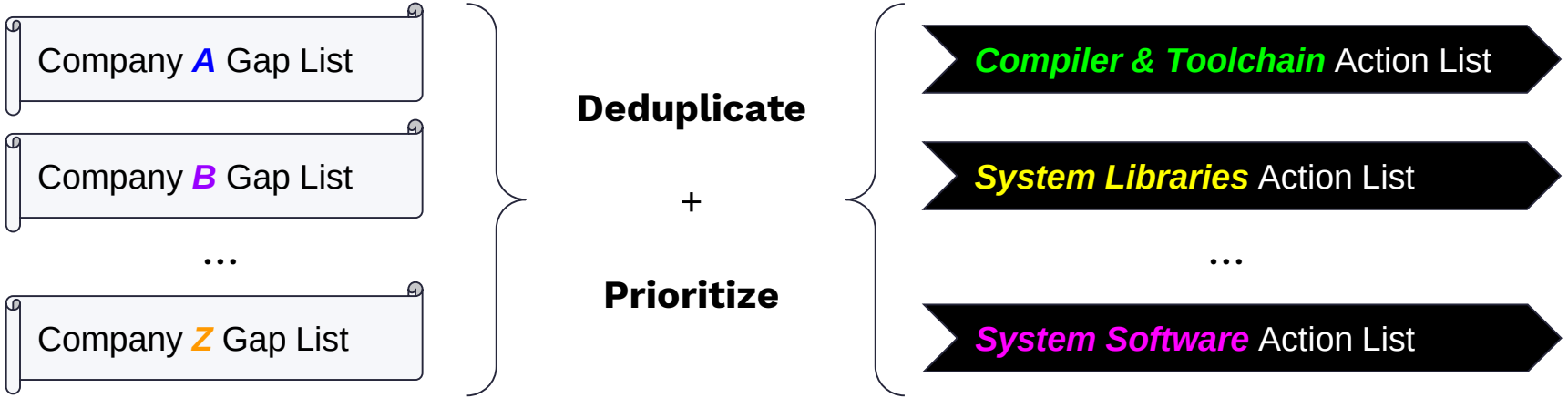
Focus Areas

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.

Compilers & Toolchains	LLVM, GCC
System Libraries	Glibc, OpenSSL, OpenBLAS, LAPACK, OneDAL, Jemalloc
Kernel & Virtualization	Linux, Android
Language Runtimes	Python, OpenJDK/Java, V8
Linux Distro Integration	Ubuntu, Debian, RHEL, Fedora, Alpine
Debug & Profiling Tools	Performance profiles, DynamoRIO, Valgrind
Simulator/Emulators	QEMU, SPIKE
System Software	UEFI, ACPI

Working Model

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



For each *Action*, complete work in responsible upstream project (e.g., LLVM)

Examples of RISE Efforts

Simulator/ Emulator

QEMU for helping test features & prove out ahead of hardware support

- AIA support
- AIA support enhancements - IRQ filtering
- Vector Cryptography support
- WorldGuard support

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/