RISC-V SUMMIT

NORTH AMERICA

RISC-V SUMMIT
NORTH AMERICA

RISC-V and Software

RISC-V 101

Nathan Egge

Software Readiness

RISC-V Software Success Today

- RISC-V has good adoption in microcontrollers
  - Single purpose application
  - Limited set of standard extensions needed, custom instructions
  - RTOS or Bare Metal
  - Control often a driving factor

- Examples
  - Seagate custom SOCs in HDD
  - Meta custom RISC-V video transcoding
  - Nvidia using RISC-V in GPUs for 9 years!
  - Billions and billions of cores shipped!

Nvidia, Google to Speak About RISC-V Use at Annual Summit

By Doug Eadline

October 19, 2024

Nvidia will discuss how it uses the RISC-V architecture at the RISC-V Summit from October 22 to 24. The GPU maker has used the RISC-V CPU architecture in its GPU microcontrollers for nine years. A 20-minute keynote from Frans Sijstermans, vice president at Nvidia, will be held on October 22 and will reveal additional details.

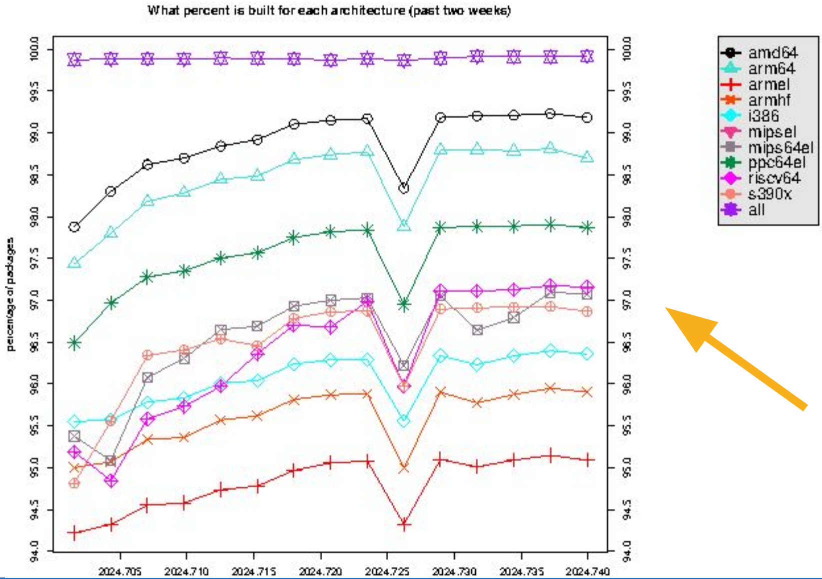
[1] https://www.hpcwire.com/2024/10/19/nvidia-google-to-speak-about-risc-v-use-at-annual-summit/

Operating Systems

- Most systems software in C/C++ without significant specialization
  - libc + syscalls good enough for POSIX support
- Examples
  - Linux: Debian, Fedora, Gentoo, etc.
  - Embedded: Yocto
  - RTOS: Zephyr, FreeRTOS
- What percent of Linux packages are enabled?

Operating Systems

- Most systems software in C/C++ without significant specialization
  - libc + syscalls good enough for POSIX support
- Examples
  - Linux: Debian, Fedora, Gentoo, etc.
  - Embedded: Yocto
  - RTOS: Zephyr, FreeRTOS
- What percent of Linux packages are enabled?
  - As of July 2024, 97% in Debian! [1]



[1] https://wiki.debian.org/RISC-V

Linux Kernel

- Active work to enable RISC-V in Linux
  - Early HWCAP feature detection, but limited to 32 long bit-vector
  - RISC-V Vector 1.0 support in 6.5
  - hwprobe() syscall added in 6.6
  - PMU support, pointer masking, bitmanip, and others on-going

- SOC support still a challenge
  - Most developer boards come with a heavily modified vendor kernel
  - Requires “bring-up” to get suitable environment for development
  - No generic RISC-V kernel in Debian, can still replace rootfs

Application Software

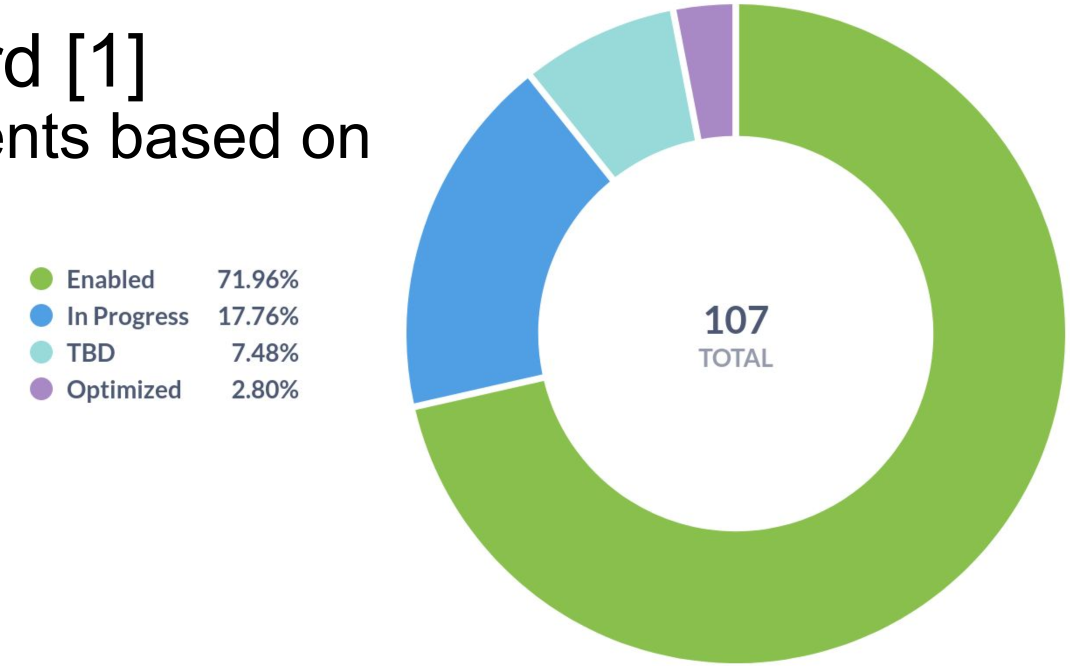
- Must "just work" on range of heterogeneous hardware
  - Scale from IOT device, to laptop, to HEDT, to server class
  - Multimedia: IOT camera, watch streaming video, multi-channel transcoding
    - Potentially all use the same libraries

- Written in managed or interpreted languages
  - Most runtimes work, but performance limited
    - No or partial JIT, native .so not compiled for RISC-V, e.g. Python
  - Java and Go getting performance optimizations through RISE

- Variable set of workloads, performance critical execution
  - Really only one mechanism for performance, SIMD aka RISC-V Vector 1.0
  - May not contain the same extensions, or same vector length

Software Readiness

- Very much application or "domain" dependent
  - Do not need everything to be perfect, just enough to get work done
- RISC-V Software Ecosystem Dashboard [1]
  - Attempts to catalog key software components based on
    - Enabled: RISC-V base support established
    - In Progress: Active development underway
    - Optimized: Software performant on RISC-V
    - TBD: No commitment to RISC-V enablement



[1] https://tech.riscv.org/software-ecosystem

Software Readiness

- In practice this is hard to measure, readiness can also mean
  - Does it build (configure and compile)

Software Readiness

- In practice this is hard to measure, readiness can also mean
  - Does it build (configure and compile)
  - Does it run

Software Readiness

- In practice this is hard to measure, readiness can also mean
  - Does it build (configure and compile)
  - Does it run
  - Does it run correctly (unit and integration tests)

Software Readiness

- In practice this is hard to measure, readiness can also mean
  - Does it build (configure and compile)
  - Does it run
  - Does it run correctly (unit and integration tests)
  - Does it run correctly on my hardware

Software Readiness

- In practice this is hard to measure, readiness can also mean
  - Does it build (configure and compile)
  - Does it run
  - Does it run correctly (unit and integration tests)
  - Does it run correctly on my hardware
  - Does it run correctly on my hardware with enough performance

Software Readiness

- In practice this is hard to measure, readiness can also mean
  - Does it build (configure and compile)
  - Does it run
  - Does it run correctly (unit and integration tests)
  - Does it run correctly on my hardware
  - Does it run correctly on my hardware with enough performance
- Optimized software is also nebulous
  - Performance often achieved over time through incremental improvements
  - Unclear what the lower bound is on compute
    - dav1d-1.5.0 still improving 6 years later

Toolchains

- GCC
  - C/C++
    - RVV intrinsics
    - Inline assembly
  - Many more
- LLVM
  - C/C++
    - RVV intrinsics
    - Inline assembly
  - Rust
- Cranelift
  - WebAssembly and more
- Golang
- v8
- OpenJDK
  - Java

Languages and Runtimes

Language	Implementation	Status	Notes
C/C++	GCC, Clang	Good	RVV Intrinsics, tunings per target, autovectorization
Javascript	v8, Spidermonkey	Works	Upstreamed, v8 wiki, spidermonkey initial support Plenty of performance work ongoing
WebAssembly	v8, Cranelift	Works	Upstreamed, available, plenty of work ongoing still
Go	golang	Good	Since Go 1.16 Supports also cgo.
Rust	rustc (LLVM, Cranelift)	Works	But no RVV intrinsics yet, no cpu features runtime detection
Python	CPython, pypy	Good	You can run pytorch just fine, jit backend for pypy
Java	OpenJDK	Good	Tracker, Apertus Distributes LTS for Java 11, 17, 21 and 22
Haskell	GHC	Works	Tracker, both LLVM and NCG backends are supported
Erlang	otp	Works	No JIT yet

Additional Tools

Tool	Type	Status	Notes
GDB	Debugger	Works	Does not print RVV registers yet
LLDB	Debugger	Works	Less available by default
linux-perf	Profiler	Sort-of	On some platforms only custom events are available
rr-project	Debugger	Missing	Tracker, Could work for cpu with **Zacas** support
mold	Linker	Good	Works

Ways to Improve Performance

Auto-vectorization

- Pros: Compiler does all the work
  - Performance can get better with newer compilers
- Cons: Language and code have to give hints
  - Scalar code often does not map to efficient vector operations
  - Compiler support may not always be present

Intrinsics

- Pros: Code uses primitives present in the instruction set
  - Same language as the rest of the code, easy to reason about and debug
  - The instruction scheduling should be optimal and tuned for the target
- Cons: Compiler support may not always be present
  - Intrinsic version changes force code updates 0.11 -> 0.12

Pure Assembly

- Pros: Full control, no chance of mis-compilation
  - Overcome ABI limitations, not everything representable with intrinsics
- Cons: Must account for everything: scheduling, register allocation, etc…
  - Difficult to write, difficult to debug, difficult to modify

Code Size Considerations

- Can trade binary size for more specialization
  - Multiple implementations selected at runtime, even with same extensions
  - e.g., Intrinsics + Function Multi-Versioning for micro-architecture tuning
- Some deployments sensitive to binary size, no universal solution
  - Desktop application on DVD may be fine
  - Mobile applications highly sensitive to download time
  - Middleware vendors differentiate on binary size
  - Server can and often rebuild everything bespoke for hardware
- Reasonable, domain-specific tradeoffs should be made

Conclusions

- Most software "ready" in that it will build and run on Linux
- Good performance is domain specific, need to test on target HW
- Toolchain support is good and RISC-V parity steadily improving
- Many software workloads will run fine as-is and unmodified
- Top priority for RISC-V enablement is more optimizations, e.g., for V (vector), Zb{a,b,c,s} (bit manip) and Zvk (vector crypto)
- You can start today!

Getting Started Guide

Ways to Develop for RISC-V

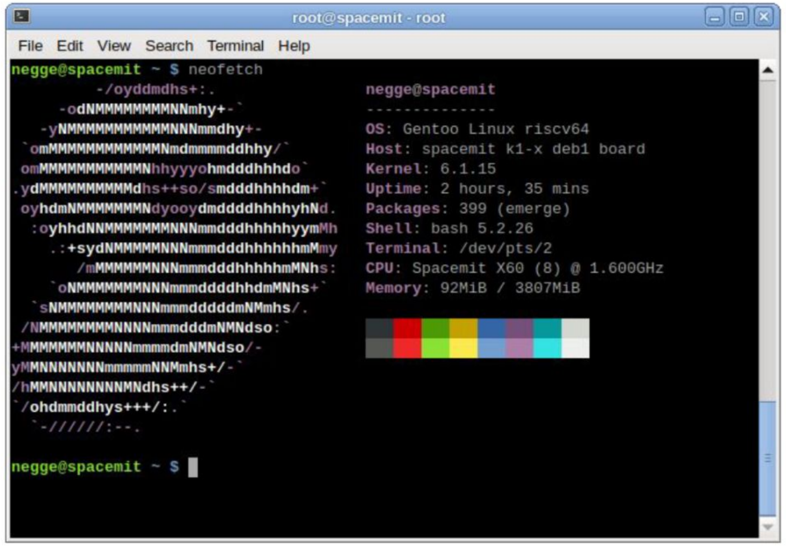
- FPGA
  - Pros: Cycle accurate model of hardware
  - Cons: Whole system booting very slow, 10's of MHz
- Emulation
  - Pros: First to get RVI extensions, flexible and configurable
  - Cons: Essentially impossible to measure performance
- Hardware
  - Pros: Performance will match what ships exactly,
  - Cons: Long lead time, may overfit microarchitecture, limited vector length

RISC-V Summit EU 2024

- Presented single slide on manually prebuilt developer images
  - Since then work has focused on build automation

Prebuilt Developer Images

- Facilitate development by providing up-to-date toolchains for building and testing
  - Latest toolchain package versions
    - clang-18.1.5
    - gcc-13.2.1_p20240503
    - rust-1.77.1
    - binutils-2.42
    - cmake-3.29.3
    - python-3.12.3
    - perl-5.38.2
    - git-2.45.1
    - subversion-1.14.3
  - Kendryte K230 and Banana Pi BPI-F3



[1] https://people.videolan.org/~negge/canaan-3G-2024-04-08.img.xz
[2] https://people.videolan.org/~negge/spacemit-4G-2024-05-15.img.xz

ROMA II image coming soon!!

RISE

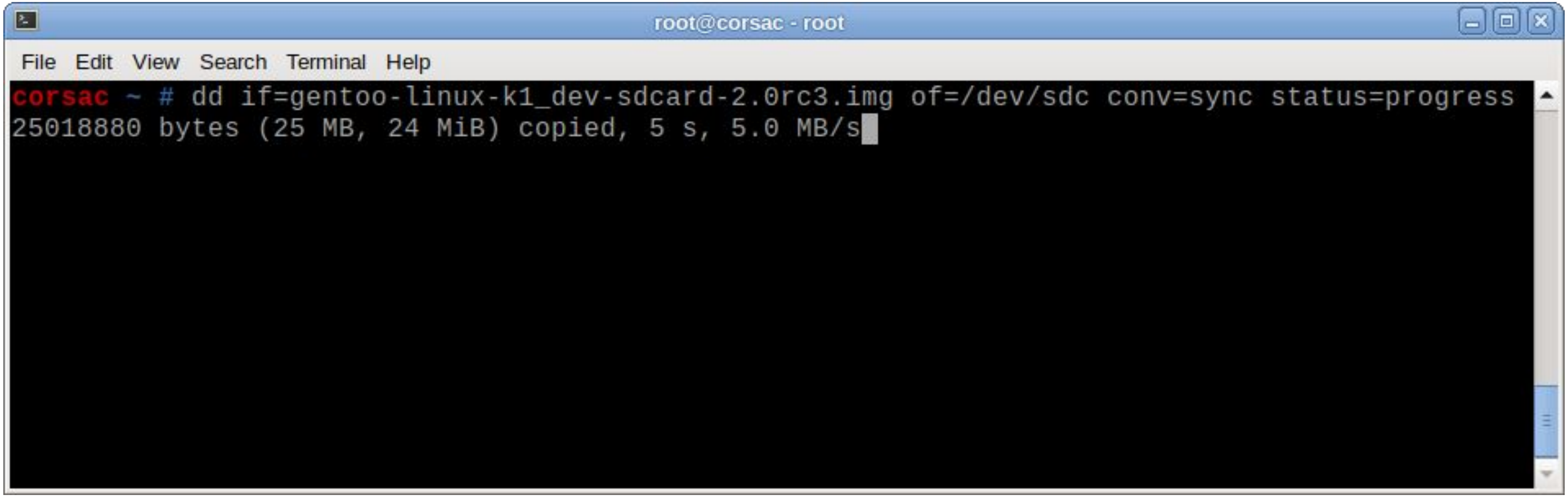
Gentoo Developer Images

- Project Goal
  - Fastest way to create bootable images with **up-to-date** toolchains!
- *Key Idea*: Automate developer image building
  - Now takes only ~300 minutes (!) to cross compile bootable image
  - U-Boot + Kernel + ~330 software packages
  - Bespoke CFLAGS possible, testing surfaced several gcc autovector issues [1]
- Partnered with Luca Barbato, RISC-V Gentoo developer
  - Fixed multiple issues unblocking full cross compilation **<-- no other distro has this**
  - Right now BPI-F3 and potentially other boards based on SpacemiT K1
  - Joint blog post in-progress to post on RISE website

[1] GCC Bug 116242 - [meta-bug] Tracker for zvl issues in RISC-V

Installing the Image [1] (from Aug-15)

```
$ dd if=gentoo-linux-k1_dev-sdcard-2.0rc3.img of=/dev/sdc conv=sync status=progress
```



[1] https://dev.gentoo.org/~lu_zero/gentoo-linux-k1_dev-sdcard-2.0rc3.img.xz

Boot Process

```
U-Boot SPL 2022.10spacemit (Aug 14 2024 - 20:15:22 -0000)
DDR type LPDDR4X
lpddr4_silicon_init consume 11ms
Change DDR data rate to 2400MT/s
Boot from fit configuration k1-x_deb1
## Checking hash(es) for config conf_2 ... OK
## Checking hash(es) for Image uboot ... crc32+ OK
## Checking hash(es) for Image fdt_2 ... crc32+ OK
## Checking hash(es) for config config_1 ... OK
## Checking hash(es) for Image opensbi ... crc32+ OK


U-Boot 2022.10spacemit (Aug 14 2024 - 20:15:22 -0000)

CPU:   rv64imafdcv
Model: spacemit k1-x deb1 board
DRAM:  DDR size = 4096 MB
DDR size = 4096 MB
DDR size = 4096 MB
```

```
## Loading kernel from FIT Image at 11000000 ...
   Using 'conf-default' configuration
   Verifying Hash Integrity ... OK
   Trying 'kernel' kernel subimage
     Description:  Linux 6.6.36+
     Type:         Kernel Image
     Compression:  gzip compressed
     Data Start:   0x110000bc
     Data Size:    14255955 Bytes = 13.6 MiB
     Architecture: RISC-V
     OS:           Linux
     Load Address: 0x00200000
     Entry Point:  0x00200000
     Hash algo:    crc32
     Hash value:   7c3065e0
   Verifying Hash Integrity ... crc32+ OK
## Flattened Device Tree blob at 31000000
   Booting using the fdt blob at 0x31000000
   Uncompressing Kernel Image
```

Boot Process (Con't)

```
   OpenRC 0.54.2 is starting up Gentoo Linux (riscv64)

 * Mounting /proc ...
 [ ok ]
 * Mounting /run ...
 [ ok ]
 * /run/openrc: creating directory
 * /run/lock: creating directory
 * /run/lock: correcting owner
 * Caching service dependencies ...
[    5.445256] usb 2-1.5: new high-speec
 [ ok ]
 * Mounting /sys ...
 [ ok ]
 * Mounting debug filesystem ...
 [ ok ]
 * Mounting config filesystem ...
 [ ok ]
 * Mounting fuse control filesystem ...
```

```
 * Create Volatile Files and Directories ...
 [ ok ]
INIT: Entering runlevel: 3
 * Starting metalog ...
 [ ok ]
 * Starting DHCP Client Daemon ...
dhcp_vendor: No such process
 [ ok ]
 * Mounting network filesystems ...
 [ ok ]
 * Starting sshd ...
 [ ok ]
 * Starting local ...
 [ ok ]


This is localhost (Linux riscv64 6.6.36+) 21:56:52

localhost login:
```

Full Gentoo Linux System

```
localhost ~ # neofetch
         -/oyddmdhs+:.                root@localhost
     -odNMMMMMMMMNNmhy+-`             --------------
   -yNMMMMMMMMMMMNNNmmdhy+-           OS: Gentoo Linux riscv64
 `omMMMMMMMMMMMMNmdmmmmddhhy/`        Host: spacemit k1-x deb1 board
 omMMMMMMMMMMMNhhyyyohmdddhhhdo`      Kernel: 6.6.36+
.ydMMMMMMMMMMdhs++so/smdddhhhhdm+`    Uptime: 23 mins
 oyhdmNMMMMMMMNdyooydmddddhhhhyhNd.   Packages: 330 (emerge)
  :oyhhdNNMMMMMMMNNNmmdddhhhhhyymMh   Shell: bash 5.2.32
    .:+sydNMMMMMNNNmmmdddhhhhhhmMmy   Terminal: /dev/console
       /mMMMMMMNNNmmmdddhhhhhmMNhs:   CPU: Spacemit X60 (8) @ 1.600GHz
    `oNMMMMMMMNNNmmmddddhhdmMNhs+`    Memory: 207MiB / 3808MiB
  `sNMMMMMMMMNNNmmmdddddmNMmhs/.
 /NMMMMMMMMNNNNmmmdddmNMNdso:`
+MMMMMMMNNNNNmmmmdmNMNdso/-
yMMNNNNNNNmmmmmNNMmhs+/-`
/hMMNNNNNNNNMNdhs++/-`
`/ohdmmddhys+++/:.`
  `-//////:--.
```

Up-to-date Toolchains!

```
localhost ~ # clang --version
clang version 18.1.8
Target: riscv64-unknown-linux-gnu
Thread model: posix
InstalledDir: /usr/lib/llvm/18/bin
Configuration file: /etc/clang/riscv64-unknown-linux-gnu-clang.cfg
localhost ~ # gcc --version
gcc (Gentoo 14.2.0 p4) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

localhost ~ # /usr/lib/ld-linux-riscv64-lp64d.so.1 --version
ld.so (Gentoo 2.40 (patchset 1)) stable release version 2.40.
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
localhost ~ #
```

2024 Jun 20

2024 Aug 1

2024 Jul 22

Wifi and ethernet just work out of the box

```
localhost ~ # modprobe 8852bs
localhost ~ # /etc/init.d/wpa_supplicant start
 * Starting WPA Supplicant Daemon ...
Successfully initialized wpa_supplicant                                       [ ok ]
localhost ~ # ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.9.202  netmask 255.255.255.0  broadcast 192.168.9.255
        inet6 fd1a:637d:f215:0:d7e5:b531:88fc:b12e  prefixlen 64  scopeid 0x0<gl
obal>
        inet6 fd95:b4c7:7c8b:0:300c:64b3:c757:b96f  prefixlen 64  scopeid 0x0<gl
obal>
        inet6 fd8d:88cb:94f4:0:b5d0:53e4:4cf:2073  prefixlen 64  scopeid 0x0<glo
bal>
        inet6 fe80::b7a2:5296:db95:fe64  prefixlen 64  scopeid 0x20<link>
        inet6 fd1a:637d:f215::93c  prefixlen 128  scopeid 0x0<global>
        ether c0:4b:24:36:6b:af  txqueuelen 1000  (Ethernet)
        RX packets 12326  bytes 51588926 (49.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12397  bytes 696222 (679.9 KiB)
```

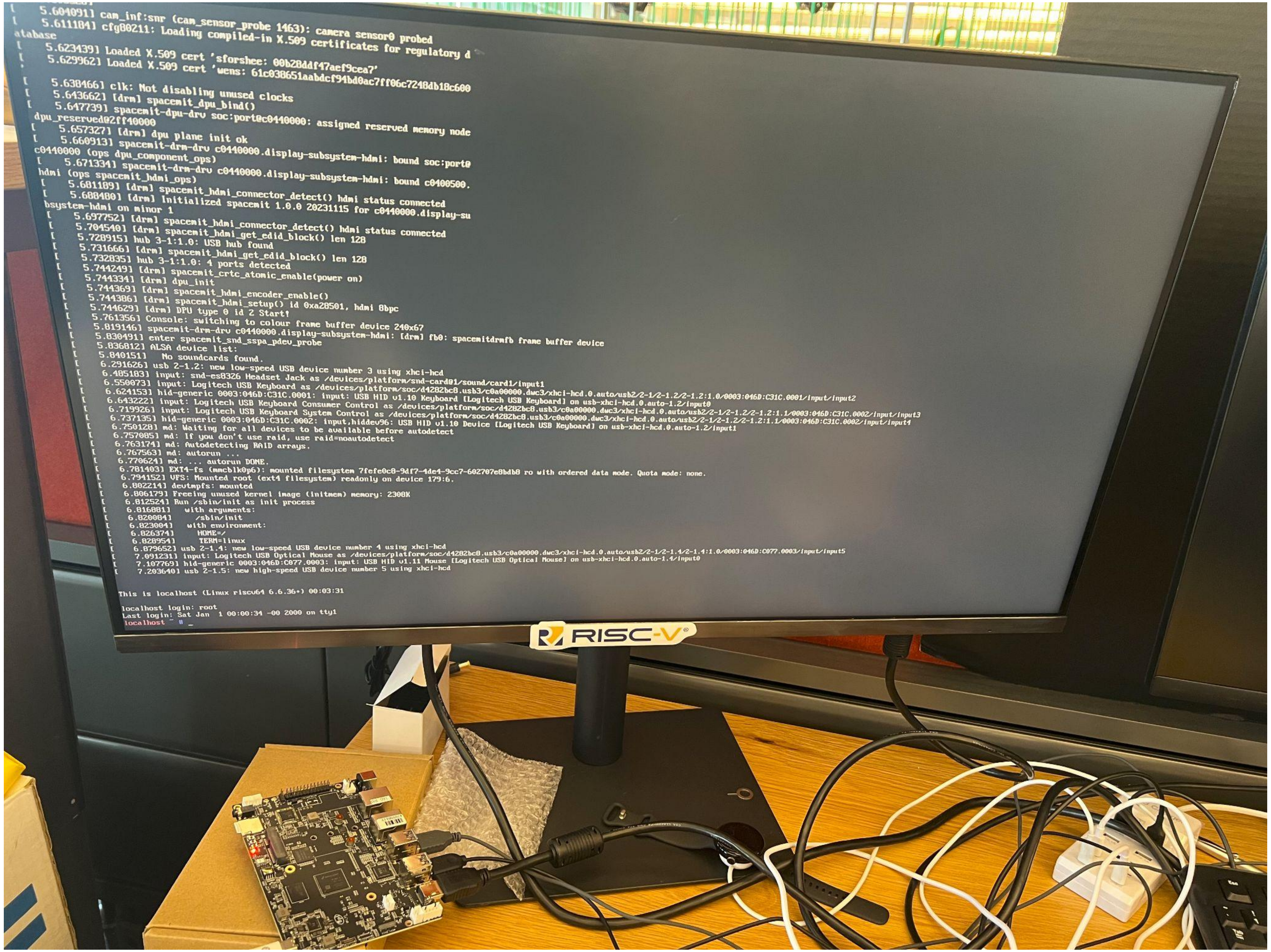
Emerge (install) new packages

```
localhost ~ # ldconfig
localhost ~ # emerge-webrsync -q
 * Latest snapshot date: 20240814
 *
 * Approximate snapshot timestamp: 1723682700
 *        Current local timestamp: 1723682400
 *
 * The current local timestamp is possibly identical to the
 * timestamp of the latest snapshot. In order to force sync, use
 * the --revert option or remove the timestamp file located at
 * '/var/db/repos/gentoo/metadata/timestamp.x'.
localhost ~ # getuto && emerge -g neofetch

 * IMPORTANT: 18 news items need reading for repository 'gentoo'.
 * Use eselect news read to view new items.


Local copy of remote index is up-to-date and will be used.
Calculating dependencies... done!
```

Demo in RISE Lounge



Future Work

- Experiment with alternate whole system build configs
  - Crossdev already supports `riscv64-unknown-linux-musl` as target
  - Paves the way to build the whole system using clang
- Build everything with **-O3 -march=rv64gcv_zvl256b**
  - Blocked on gcc bugs, but may work with clang
- Improve the overall cross-building experience
  - This project already found many bugs
    - Few packages (e.g. perl) already got some fixes
    - Crossdev has a pending patch to make it profile-aware

Latest: https://dev.gentoo.org/~lu_zero/riscv/gentoo-linux-k1_dev-sdcard-2.0rc7.img.xz

Example RVV 1.0 Code #1 - Get vector length

```
.global _start

_start:
  csrr a0, vlenb
  addi a7, x0, 93
  ecall
```

nathan@corsac - nathan

File Edit View Search Terminal Help

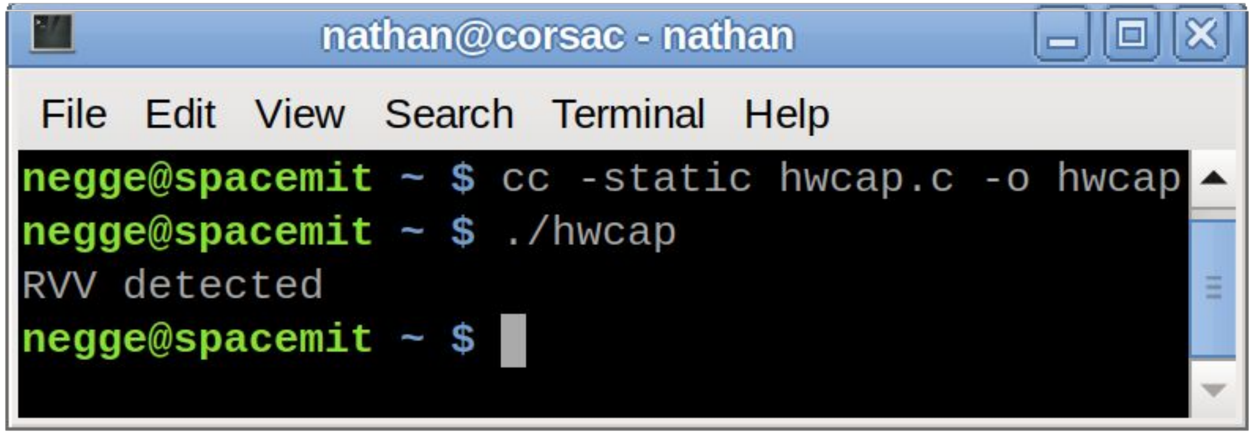
```
negge@spacemit ~ $ as getvlenb.as -march=rv64gcv -o getvlenb.o
negge@spacemit ~ $ ld getvlenb.o -o getvlenb
negge@spacemit ~ $ ./getvlenb
negge@spacemit ~ $ echo $?
32
negge@spacemit ~ $
```

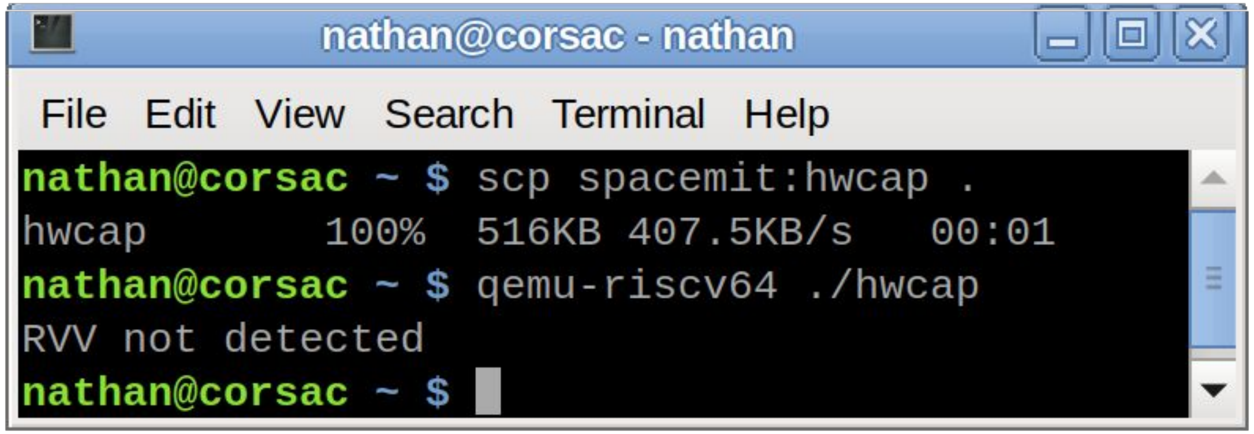
Example RVV 1.0 Code #2 - Run-time detect

```c
#include <sys/auxv.h>
#include <stdio.h>

#define ISA_V_HWCAP (1 << ('v' - 'a'))

void main() {
  unsigned long hw_cap = getauxval(AT_HWCAP);
  printf("RVV %s\n", hw_cap & ISA_V_HWCAP ? "detected" : "not found");
}
```





Example RVV 1.0 #3 - Application Profiling

```
$ perf record -e u_mode_cycle ./dav1d -i Bosphorus_1080p_8bit.ivf -o /dev/null
dav1d 1.5.0-3-g55fb943 - by VideoLAN
Decoded 600/600 frames (100.0%) - 10.04/30.00 fps (0.33x)
[ perf record: Woken up 84 times to write data ]
[ perf record: Captured and wrote 22.223 MB perf.data (582464 samples) ]
$ perf report
```

```
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 578K of event 'u_mode_cycle:u'
# Event count (approx.): 218918769644
#
# Overhead  Command       Shared Object              Symbol
# ........  ............  .........................  ..............................................................
#
    67.59%  dav1d-worker  libdav1d.so.7.0.0          [.] prep_8tap_c
     3.68%  dav1d-worker  libdav1d.so.7.0.0          [.] put_8tap_c
     3.58%  dav1d-worker  libdav1d.so.7.0.0          [.] $xrv64i2p1_m2p0_a2p1_f2p2_d2p2_c2p0_zicsr2p0_zifencei2p0_zmmul1p0
     2.45%  dav1d-worker  libdav1d.so.7.0.0          [.] wiener_c
     2.22%  dav1d-worker  libdav1d.so.7.0.0          [.] $xrv64i2p1_m2p0_a2p1_f2p2_d2p2_c2p0_zicsr2p0_zifencei2p0_zmmul1p0
     2.11%  dav1d-worker  libdav1d.so.7.0.0          [.] prep_8tap_smooth_sharp_c
     1.44%  dav1d-worker  libdav1d.so.7.0.0          [.] prep_8tap_smooth_regular_c
     1.29%  dav1d-worker  libdav1d.so.7.0.0          [.] dav1d_mask_8bpc_rvv
     0.87%  dav1d-worker  libdav1d.so.7.0.0          [.] $xrv64i2p1_m2p0_a2p1_f2p2_d2p2_c2p0_zicsr2p0_zifencei2p0_zmmul1p0
     0.80%  dav1d-worker  libdav1d.so.7.0.0          [.] load_tmvs_c
     0.70%  dav1d-worker  libdav1d.so.7.0.0          [.] prep_8tap_sharp_c
     0.69%  dav1d-worker  libdav1d.so.7.0.0          [.] prep_8tap_smooth_c
     0.68%  dav1d-worker  libdav1d.so.7.0.0          [.] decode_b
     0.62%  dav1d-worker  libdav1d.so.7.0.0          [.] dav1d_create_lf_mask_inter
     0.58%  dav1d-worker  libdav1d.so.7.0.0          [.] put_8tap_scaled_c
     0.54%  dav1d-worker  libc.so.6                  [.] __strxfrm_l
```

RISC-V SUMMIT NORTH AMERICA

Questions?