Table Of Contents

Basic Information

Name: Ninad Sunil Jangle

Major: Electronics Engineering

University: Veermata Jijabai Technological Institute, Mumbai

Github: @ninja3011

Gitter: Ninad S. Jangle, @ninja3011

Email: *ninadjangle3011@gmail.com*

Phone: (+91) *8879017402*

Website: https://ninadjangle.tech/

Timezone: Indian Standard Time (UTC +5:30)

Project Porting Spidermonkey to RISC-V

MIDTERM REPORT

Mentor:

Wei Wu

Mentees:

1. **Ninad Jangle**
2. **Dhairya Shah**

Abstract

Porting Spidermonkey to RISC-V takes the first spot on the PLCT Lab RISCV-Open-wishlist. RISC-V is an open-source ISA that rose to prominence in recent years. Using the Instruction Set Architecture a lot of enthusiasts and organisations have developed their implementations of processors. However, RISC-V is yet to see a major commercial boost.

With the Porting Spidermonkey to RV64GC RISC-V processor project, its value to the general audience will experience that boost. To understand it simply, the genius of intuitive design captivates the active RISC-V community. However, for the general public, it is not the design that's the keeper, but the facilities it avails them.

While there will be applications written in the future to specifically target the RISC-V Processors, in the meantime, there is a lot of software that the public is attached to, which they expect to run on any Computer they purchase. One such beloved application is Firefox. By porting Firefox to RISC-V architecture, future users can make use of the advanced browser. This new addition to the RISCV ecosystem will further elevate its standing in the marketplace.

Project Goals

Spidermonkey is the JavaScript Engine inside Firefox. It has JIT compilers for generating native binary codes on the fly. This project aims to port Spidermonkey to the RV64GC platform.

The main goals of the project include Cross-compile Spidermonkey to RV64GC Linux (Fedora) platform. Patches that let Spidermonkey running on RV64GC Linux under interpreter mode. Porting baseline compilers so that Spidermonkey can enable at least one JIT compiler on the RV64GC platform. Submit all patches to upstream for code review.

Proposed Deliverables (during LFX)

1. Cross-compile Spidermonkey to RV64GC Linux (Fedora) platform.
2. Patches that let Spidermonkey run on RV64GC Linux under interpreter mode.
3. Porting the baseline compilers so that Spidermonkey can enable at least one JIT compiler on the RV64GC platform
4. Submit all patches to upstream for code review (merging into upstream is encouraged but not required)

Important Documents

- LFXProposal_NinadJangle
- Main progress Doc
- JIT_documentation
- JIT_Theory_and_logs

PROJECT PROGRESS TRACKER

WEEK	TASKS	Comment
PRE-SELECTION (WEEK 0.1 - 0.2)	Find the source code repo of Spidermonkey. Build it on your machines. Run Octane/Kraken/SunSpider benchmarks using the js shell you just built.	COMPLETED *Ran all the benchmarks and logged the results in the proposal.*
	Try to set up the cross-toolchain for RISC-V (RV64GC) platform. You can either install it using apt/yum to clone the riscv-gnu-toolchain repo and build it yourself. Then try to cross-compile the spidermonkey to RV64GC. The bonus is to prepare a QEMU emulator which can execute RV64GC binaries.	COMPLETED with BONUS TASK *Set up the cross-compilation toolchain and an emulator capable of running rv64gc binaries. Tried cross-compiling mozilla-unified directory and logged the results in the proposal.*
	Make a Detailed Proposal. Show my understandings and knowledge background. - This proposal, done.	COMPLETED *(Proposal)*
INITIAL WORK ASSIGNMENTS (Week 1)	Native build spidermonkey on riscv64 and run regression tests. Log the results. (one)	IN PROGRESS *Qemu was unable to handle such a heavy binary and we faced issues in transferring the files onto the emulator system.*
	Cross-build spidermonkey (host=x86 target=riscv64) and run regression tests on riscv64. Log the results. (one)	IN PROGRESS *Facing a lot of errors with this task. Recently came across makotokato san's work on the*

		riscv64 firefox version. Currently, trying to cross-compile for it on our systems and VM.
	Say hello and introduce yourself in Mozilla's chat room. (all)	COMPLETED
WEEK 2-4	Go through the codebase of Spidermonkey. Write documents describing the internals. Blog your doc. submit the internal doc as patchset to Mozilla codebase.	IN PROGRESS *Regularly go through the entire spidermonkey codebase, it was too huge to understand as a whole so I narrowed my understanding to the Spidermonkey folders taking a more general approach to the rest of the files*
	Search the codebase and try to duplicate the arm64 or mips64 backend of spidermonkey. Copy it to the riscv64 folder/file/functions without real modifications. Try your best to pass the build and regression testing.	ATTEMPTED *Attempted to run the replicate the cross-compilation by copying the files into a riscv64X folder and adding its options to the configure.in config.sub and other related files.*
	Paste your questions and findings at https://discourse.mozilla.org/c/spidermonkey/551	IN PROGRESS *We continue to ask doubts on the chat.mozilla channels*
WEEK 4-6	Working on virtual machines and Instances Set up required toolchains and folders on the remote device.	COMPLETED *Communicated with Wei and got access to Virtual machines with greater processing power and memory capacity to counter certain errors which may have been hardware-based.*

	Work on build errors and work to build riscv for cross-compilation on VM	IN PROGRESS *Errors being dealt with: (i) Use of deleted functions, (ii) Segmentation fault 11 , (iii) usr/bin/ld cannot find -lz.*
	Solve Linking issues	IN PROGRESS *In cross compilations, the major challenge is not about how to change the binary of individual files but to transfer those links to a new system with a new architecture. The -lz error is a way of saying even though the zlib devel files exist the link to them was lost in cross-compilation*

IMPORTANT MILESTONES

Toolchains Setup

- Cross Compilers
  - riscv-linux-gnu-gcc + riscv-linux-gnu-g++ (install)
  - riscv-unknown-linux-gnu-gcc + riscv-unknown-linux-gnu-g++ (install)
- Firefox Versions:
  - Mozilla-unified (download)
  - Gecko-dev (clone)
- RISC-V Emulator:
  - Qemu (setup)
- Remote access on PLCT Computer Boards

Baseline Compiler Study

Completed a comprehensive files study of the BaselineCompiler in Spidermonkey. The documentation can be found at JIT_Documentation

The Baseline Compiler uses the same Inline Caches mechanism from the Baseline Interpreter but additionally translates the entire bytecode to native machine code. This removes dispatch overhead and does minor local optimizations. This machine code still calls back into C++ for complex operations. The translation is very fast but the BaselineScript uses memory and requires mprotect and flushing CPU caches.

Focusing on these files:

- Baseline Compiler
  - BaselineBailouts.cpp
  - BaselineCacheIRCompiler.cpp BaselineCacheIRCompiler.h
  - BaselineCodeGen.h BaselineCodeGen.cpp
  - BaselineDebugModeOSR.cpp BaselineDebugModeOSR.h
  - BaselineFrame.cpp BaselineFrame.h
  - BaselineFrameInfo.cpp BaselineFrameInfo.h
  - BaselineFrameInfo-inl.h
  - BaselineFrame-inl.h

- BaselineIC.cpp BaselineIC.h
- BaselineICList.h
- BaselineJIT.cpp BaselineJIT.h

Along with Wei Wu, we set an aim of porting the Baseline Compiler to the riscv64 target, first, without optimisation. To achieve this in the future, we have dug deep into the existing codebase and developed an understanding of the flow, structure and logic behind the Baseline Compiler.

Building on Makoto Kato san's patches

Around the end of September 2021, makotokato successfully cross-compiled firefox for riscv64. He built the browser without porting the JIT's. This was a breakthrough in the project, though it came from outside. His work and our goals fit in perfectly with each other. We are working on porting those JIT's he left out and his work got us a leap in our chances of achieving that.

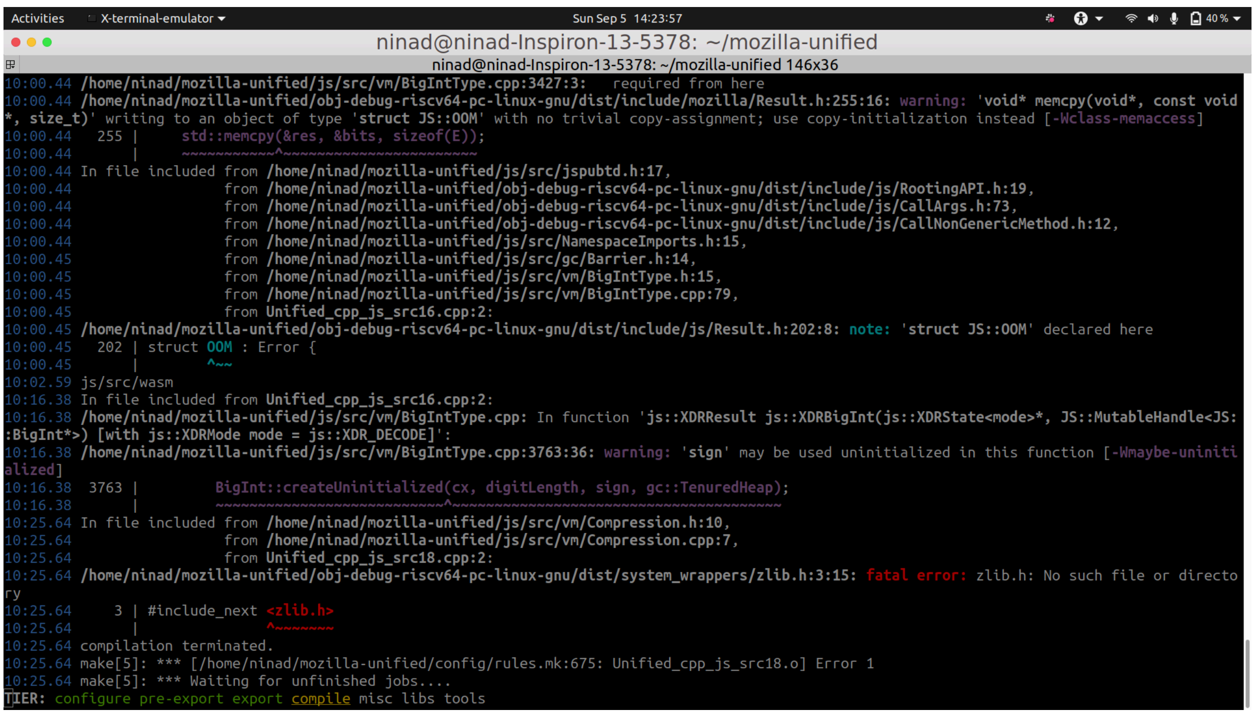
Our journey in cross-compiling has been a lot like learning to ride a bicycle, we continuously face challenges every run of every day. New errors have a way of teaching as well, with the various errors I will discuss next, came a deeper understanding of the codebase which was not possible with a general study.

The build of the gecko-dev repo riscv64 branch is yet incomplete by me, I keep running into issues, the solutions to which aren't always clear. The only two people I know who have built it successfully are makotokato and felixonmars, I will try reaching out to them to get their help on it. So I can start the work on JITs sooner.

PESKY ERRORS & FIXES

There are a few errors anyone who is trying to cross-compile jsshell of firefox is bound to run into. Let's see some of the few :

1) Zlib.h not found



First check if the files exist, if not try to install the zlib devel files:

```
sudo apt install zlib1g-dev

sudo apt-get install libz-dev
```

More common, the files already do exist, but during cross-compilation, the necessary links are not transferred. So the build system is not able to find the files even though they exist.

Afterwards, check the cross compiler used. This issue comes with using riscv64-unknown-linux-gnu but not with riscv64--linux-gnu.

2) Error: Can't find header fontconfig/fcfreetype.h.

Like above, Check the cross compiler used. This issue comes with using riscv64-unknown-linux-gnu but not with riscv64--linux-gnu.

3) ld terminated with signal 11 [Segmentation fault], core dumped

Terminal dump

The error arises due to our rust version being 2.

FIX :

```
export RUSTC_OPT_LEVEL=1
```

Otherwise due to enable-optimise: rustc would segfault 11 when compiling neqo-transport.

4) No `_HIDIOCGRDESC` in `platform::hidwrapper` .For more information about this error, try `rustc --explain E0432`. Error: Could not compile `authenticator`

This error is occurring because our rust version is older now.

In Rust 2015, paths in `use` statements are relative to the crate root. To import items relative to the current and parent modules, use the `self::` and `super::` prefixes, respectively.

In Rust 2018, paths in `use` statements are relative to the current module unless they begin with the name of a crate or a literal `crate::`, in which

case they start from the crate root. As in Rust 2015 code, the `self::` and `super::` prefixes refer to the current and parent modules respectively.

FIX, rustup update; But this is resulting in still new errors.

5) Attempted to duplicate mips64 files in a riscv64X folder in JIT and run from there

```
ninad@ninad-Inspiron-13-5378:~/mozilla-unified$ ./mach build
checking for vcs source checkout...
hg
Adding configure options from /home/ninad/mozconfigs/debug/MOZCONFIG
  --enable-application=js
  --enable-debug
  --enable-optimize
  --target=riscv64X
checking for host system type...
Executing: `sh /home/ninad/mozilla-unified/build/moz.configure/../autoconf/config.guess`
x86_64-pc-linux-gnu
checking for target system type...
Executing: `sh /home/ninad/mozilla-unified/build/moz.configure/../autoconf/config.sub riscv64X-pc-linux-gnu`
Unknown CPU type: riscv64X

ninad@ninad-Inspiron-13-5378:~/mozilla-unified$ ./mach build
checking for vcs source checkout...
hg
Adding configure options from /home/ninad/mozconfigs/debug/MOZCONFIG
  --enable-application=js
  --enable-debug
  --enable-optimize
  --target=riscv64X
checking for host system type...
Executing: `sh /home/ninad/mozilla-unified/build/moz.configure/../autoconf/config.guess`
x86_64-pc-linux-gnu
checking for target system type...
Executing: `sh /home/ninad/mozilla-unified/build/moz.configure/../autoconf/config.sub riscv64X-pc-linux-gnu`
'riscv64X' is not a valid value for EnumStringSubclass
```

One by one I added the riscv64X option in CPU and Fedors as OS in the Moz.configure, configure.in, config.sub files and so forth

Observing the errors and compensating for them through riscv64X options. The process was riddled with errors. And we stopped midway to replicate Makoto Kato's build. However, got to learn a lot about how the option of CPU and archs and OS is added to firefox which will be helpful down the road.

6) When looking at the .rustup being used;
    - Actually using '/home/ninad/.rustup/toolchains/stable-x86_64-unknown-linux-gnu/bin/rustc'
    - Actually using '/home/ninad/.rustup/toolchains/stable-x86_64-unknown-linux-gnu/bin/cargo'

On rustup show we get,

```
ninad@ninad-Inspiron-13-5378:~/gecko-dev-riscv-master$ rustup
show
Default host: x86_64-unknown-linux-gnu
rustup home:  /home/ninad/.rustup

installed targets for active toolchain
--------------------------------------

riscv64gc-unknown-linux-gnu
x86_64-unknown-linux-gnu

active toolchain
----------------

stable-x86_64-unknown-linux-gnu (default)
rustc 1.55.0 (c8dfcfe04 2021-09-06)
```

So I installed a toolchain for riscv64gc as well,

```
ninad@ninad-Inspiron-13-5378:~/gecko-dev-riscv-master$ rustup
toolchain install stable-riscv64gc-unknown-linux-gnu
error: DEPRECATED: future versions of rustup will require
--force-non-host to install a non-host toolchain as the default.
warning: toolchain 'stable-riscv64gc-unknown-linux-gnu' may not
be able to run on this system.
warning: If you meant to build software to target that platform,
```

```
perhaps try `rustup target add riscv64gc-unknown-linux-gnu`
instead?
info: syncing channel updates for
'stable-riscv64gc-unknown-linux-gnu'

  stable-riscv64gc-unknown-linux-gnu unchanged - (error reading
rustc version)

info: checking for self-updates
```

Looks like the system does not like me running the toolchain on its x86 system. So I am currently trying to use the rust build system cargo and specifying the release to riscv64gc-unknown-linux-gnu

```
cargo build --release --target=riscv64gc-unknown-linux-gnu
```


```
error: environment variable `MOZ_TOPOBJDIR` not defined
  --> modules/libpref/init/static_prefs/src/lib.rs:12:5
   |
```

```
12 |     env!("MOZ_TOPOBJDIR"),
```

```
export MOZ_TOPOBJDIR=~/gecko-dev-riscv-master
```

```
error: couldn't read
/home/ninad/gecko-dev-riscv-master/modules/libpref/static_prefs.
rs: No such file or directory (os error 2)
  --> modules/libpref/init/static_prefs/src/lib.rs:11:1
   |
11 | / include!(concat!(
12 | |     env!("MOZ_TOPOBJDIR"),
13 | |     "/modules/libpref/static_prefs.rs"
14 | | ));
   | |___^
   |
   = note: this error originates in the macro `include` (in
Nightly builds, run with -Z macro-backtrace for more info)

error: could not compile `static_prefs` due to previous error
```

MOZCONFIGS USED

- MakotoKato's MOZCONFIG taken from blog
- Felixonmars Inspired MOZCONFIG made from PR and patch
- Bootstrap Enabled MOZCONFIG got from cross compile firefox
- Without ZLIB MOZCONFIG

PLAN AHEAD

Initially, the challenges coming in the process, especially the build boggled me a bit. But as I continued to work on the project I realised that it is an infuriating process with a lot of variables which will make sure 95% of my effort will go to understanding these specific issues

I look forward to continuing to work on this now that I have a stronger foothold on the problems and finding new solutions.

NEXT STEPS

The next step is to get the **build cleared** asap. There are very few people in the field of cross-compilation and the resources are sparse, jumping on the makoto's code and trying to build it has been a challenge as there are no supporting docs. We have posted on the Mozilla IRC for solutions and tried their approaches but they haven't yielded the required results either. An opportunity to **talk to Makoto once** if possible would be a great push and I will reach out to him on this

Once the build is cleared and out of the way, I will get to writing the files for the **BaselineCompiler** from the **riscv64** perspective. Inside of **js/src/jit/riscv64** folder. Get the **interpreter** working and **one** of the **compilers working**.

I am confident that once the build is cleared the further steps will follow as well as we have had time to prepare for them and they come with more clear instructions. I look forward to **porting** this **by** the **end of 2022.**

Working Environment And Schedule

I have a Dell Inspiron i7 processor, 20GB ram, 400 internal 1TB external space, Ubuntu/Windows dual booted laptop with an additional full-size monitor with Bluetooth keyboard, mouse and speakers.

I have my room, with stable 20-50Mbps WiFi connectivity to work in comfortably.

For the majority duration of the timeline, the college will run on online mode. This will give me great flexibility over my timings. I'll be available for calls and meetings pretty much any time of the day as I am always glued to my PC anyway. For the work portion of it, I will mostly do LFX daily in two sessions day and night. The day between 11 am-5 pm and Night between 11 pm-2 am

I have started working on the remote computer boards provided to use by Wei as well. I had a misconception that the segmentation fault was due to memory space when it was due to the rustc version. However, it builds the software much faster and has overall increased my productivity.

Communication

We have weekly meets **every Monday on zoom at 1 pm, Indian Standard Time.**

Meet link:
https://us02web.zoom.us/j/88640309076?pwd=RS9QTjBvaTFvZUZ1TVVCSGdFWnV5QT09

iCalendar Invite:
https://us02web.zoom.us/meeting/tZwrc-mrrjIqHdI31HSBKUm_K378SADbZlLs/ics?icsToken=98tyKuGgqD8qGdScsB6BRpw-BI-gc-nziFhfgqd0z07NMAJ4Vw7JHbEVaoFeNNrq

We have also joined the RISCV International Slack channel and are a part of the Mozilla IRC channel on chat.mozilla.org

Post-LFX Plans

I plan to harvest the experience and knowledge I will gain from LFX and move forward to making regular contributions on open source projects related to RISC-V.

I would like to show people my work and ask them to join in on the RISC-V revolution. And also contribute to other projects to gain experience and pass it on.

I plan to help my juniors prepare for next years LFX, introduce them to open source and help them get a head start on projects my club mates and I have worked on in LFX.

References

---

1. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43216.pdf
2. Links:
    a. *BASICS and OVERVIEW:*
3. Required skills and possible Firefox Contributions: https://firefox-source-docs.mozilla.org/contributing/contributing_to_mozilla.html
4. Mozilla Documentation: https://firefox-source-docs.mozilla.org/js/index.html
5. Work on v8 engine: https://www.youtube.com/watch?v=j-KvrPhXA5Q&t=204s
6. Building Firefox: https://wiki.mozilla.org/Building_Firefox/SURF
    a.
    b. *SPIDERMONKEY:*
7. **Source code repo for Spidermonkey**: https://hg.mozilla.org/mozilla-central
8. **Spidermonkey Documentation**: https://spidermonkey.dev/docs/
    a. https://searchfox.org/mozilla-central/source/js/src/jit/
9. **Spidermonkey Garbage Collector**:
    a. https://firefox-source-docs.mozilla.org/js/gc.html
10. Building reference: https://wiki.mozilla.org/JavaScript:New_to_SpiderMonkey#Build_the_js_shell
11. Benchmarks: https://wiki.mozilla.org/JavaScript:New_to_SpiderMonkey#Benchmark_your_changes
12. Octane: https://svn.webkit.org/repository/webkit/trunk/PerformanceTests/Octane/
13. Kraken: https://wiki.mozilla.org/Kraken
14. SunSpider: https://webkit.org/perf/sunspider/sunspider.html
15. https://svn.webkit.org/repository/webkit/trunk/PerformanceTests/SunSpider/
    a. *Regressions: (mozregression)*
16. **Mozregression:** https://mozilla.github.io/mozregression/
    a. *RISC-V:*
17. **RISC-V**: https://twilco.github.io/riscv-from-scratch/2019/03/10/riscv-from-scratch-1.html
    a. https://saveriomiroddi.github.io/Quick-riscv-cross-compilation-and-emulation/
    b. https://wiki.debian.org/RISC-V
18. **RISC-V SiFive**: https://www.sifive.com/boards
    a. *Baseline JIT:*
19. **A crash course on JIT compilers: Series of articles** https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/
20. **Mozilla Developer Guides:**
    a. https://youtube.com/playlist?list=PLo3w8EB99pqJVPhmYbYdInBvAGarDavh-
    b.
21. BaselineJIT.h: https://searchfox.org/mozilla-central/source/js/src/jit/BaselineJIT.h

22. BaselineJIT.cpp: https://searchfox.org/mozilla-central/source/js/src/jit/BaselineJIT.cpp
    a. *QEMU:*
23. **QEMU** : https://www.qemu.org/download/
    a. https://colatkinson.site/linux/riscv/2021/01/27/riscv-qemu/
        i. https://youtu.be/AAfFewePE7c
24. QEMU and RISC-V toolchain setup: https://twilco.github.io/riscv-from-scratch/2019/03/10/riscv-from-scratch-1.html
    a. https://saveriomiroddi.github.io/Quick-riscv-cross-compilation-and-emulation/
    b. https://risc-v-getting-started-guide.readthedocs.io/en/latest/linux-qemu.html
    c. *MOZCONFIGS:*
25. https://gist.github.com/EricRahm/23d8f486f7d8bc6369a7d240df7ce572
26. https://wiki.ubuntu.com/CompileFirefoxNewVersion
27. **https://wontfix.blogspot.com/2021/07/firefox-on-linuxriscv64.html**
    a. *Cross compiled Spidermonkey on riscv64:*
28. [1] https://github.com/makotokato
29. [2] https://wontfix.blogspot.com/2021/07/firefox-on-linuxriscv64.html
30. [3] https://github.com/felixonmars/archriscv-packages/pull/139
31. **[4] https://twitter.com/felixonmars/status/1442140779955191812**
    a. *Zlib:*
32. http://www.zlib.net/
33. http://webcache.googleusercontent.com/search?q=cache:S8HLazolv7kJ:mozilla.6506.n7.nabble.com/NSS-and-NSPR-compilation-error-ssl3con-c-36-18-fatal-error-zlib-h-No-such-file-td359855.html+&cd=20&hl=en&ct=clnk&gl=in&client=ubuntu
34. https://bugzilla.mozilla.org/show_bug.cgi?id=763651
35. https://bugzilla.mozilla.org/attachment.cgi?id=673561&action=diff
36. **https://doc.dpdk.org/guides/compressdevs/zlib.html**
    a. *Baseline Compiler and Interpreter:*
37. **Baseline Interpreter:** https://hacks.mozilla.org/2019/08/the-baseline-interpreter-a-faster-js-interpreter-in-firefox-70/
    a. *WebAssembly:*
38. Web Assembly, Implementation: https://webassembly.org/
    a. https://webassembly.org/docs/faq/#will-webassembly-support-view-source-on-the-web
39. ECMAScript: https://tc39.es/ecma262/
40. Web Assembly(WASM) vs asm.js: (Web Assembly is generic n faster than asm.js) https://hacks.mozilla.org/2017/03/why-webassembly-is-faster-than-asm-js/
41. **https://github.com/makotokato**
42. https://github.com/makotokato/gecko-dev/tree/riscv64
43. https://wontfix.blogspot.com/2021/07/firefox-on-linuxriscv64.html
44. **https://github.com/felixonmars/archriscv-packages/pull/139**
45. **https://twitter.com/felixonmars/status/1442140779955191812**

cross-compile