OSDT 2022

(Online)

IREE--MLIR-based end-to-end compiler & runtime for Machine Learning

Feng Li (李枫)
hkli2013@126.com
Dec 11, 2022

Agenda

I. Background
- Tech Stack
- Testbed

II. Architecture & Design
- Overview
- Runtime

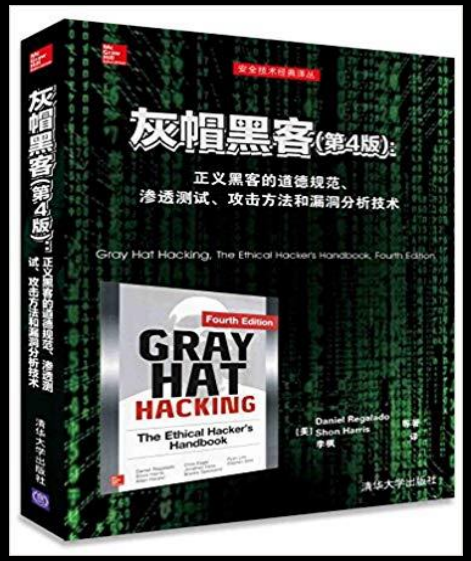
III. Practice on ARM
- RPi4
- …

IV. TinyIREE
- …

V. Wrap-up

Who Am I

- An indie developer from China
- The main translator of the book «Gray Hat Hacking The Ethical Hacker's Handbook, Fourth Edition» (ISBN：9787302428671) & «Linux Hardening in Hostile Networks, First Edition» (ISBN: 9787115544384)



- Pure software developmement for ~15 years(~11 years on Mobile dev)
- Actively participate in various activities of the open source community
- https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/Conf
  https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/LTS
- Recently, focus on infrastructure of Cloud/Edge Computing, AI, Virtualization, Program Runtimes, Network, 5G, RISC-V, EDA…

I. Background

1) Tech Stack

1.1 LLVM

1.1.1 MLIR

- https://mlir.llvm.org/

  Multi-Level Intermediate Representation

  The MLIR project is a novel approach to building reusable and extensible compiler infrastructure. MLIR aims to address software fragmentation, improve compilation for heterogeneous hardware, significantly reduce the cost of building domain specific compilers, and aid in connecting existing compilers together.

- **Motivation ("Meta IR" and Compiler Infrastructure for AI, EDA…)**

  MLIR is intended to be a hybrid IR which can support multiple different requirements in a unified infrastructure. For example, this includes:

  - The ability to represent dataflow graphs (such as in TensorFlow), including dynamic shapes, the user-extensible op ecosystem, TensorFlow variables, etc.
  - Optimizations and transformations typically done on such graphs (e.g. in Grappler).
  - Ability to host high-performance-computing-style loop optimizations across kernels (fusion, loop interchange, tiling, etc.), and to transform memory layouts of data.
  - Code generation “lowering” transformations such as DMA insertion, explicit cache management, memory tiling, and vectorization for 1D and 2D register architectures.
  - Ability to represent target-specific operations, e.g. accelerator-specific high-level operations.
  - Quantization and other graph transformations done on a Deep-Learning graph.
  - Polyhedral primitives.
  - Hardware Synthesis Tools / HLS.

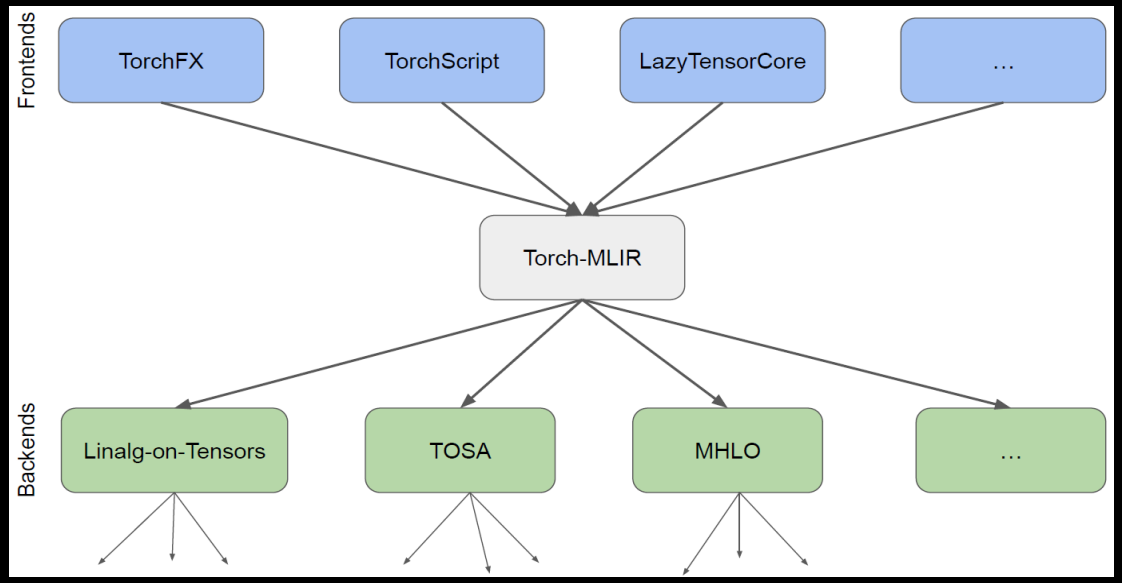
1.1.1.1 Torch-MLIR

- https://github.com/llvm/torch-mlir

  To provide first class support from the PyTorch ecosystem to the MLIR ecosystem.

Torch-MLIR Multiple Vendors use MLIR as the middle layer, mapping from platform frameworks like PyTorch, JAX, and TensorFlow into MLIR and then progressively lowering down to their target hardware. We have seen half a dozen custom lowerings from PyTorch to MLIR. Having canonical lowerings from the PyTorch ecosystem to the MLIR ecosystem would provide much needed relief to hardware vendors to focus on their unique value rather than implementing yet another PyTorch frontend for MLIR. The goal is to be similar to current hardware vendors adding LLVM target support instead of each one also implementing Clang / a C++ frontend.

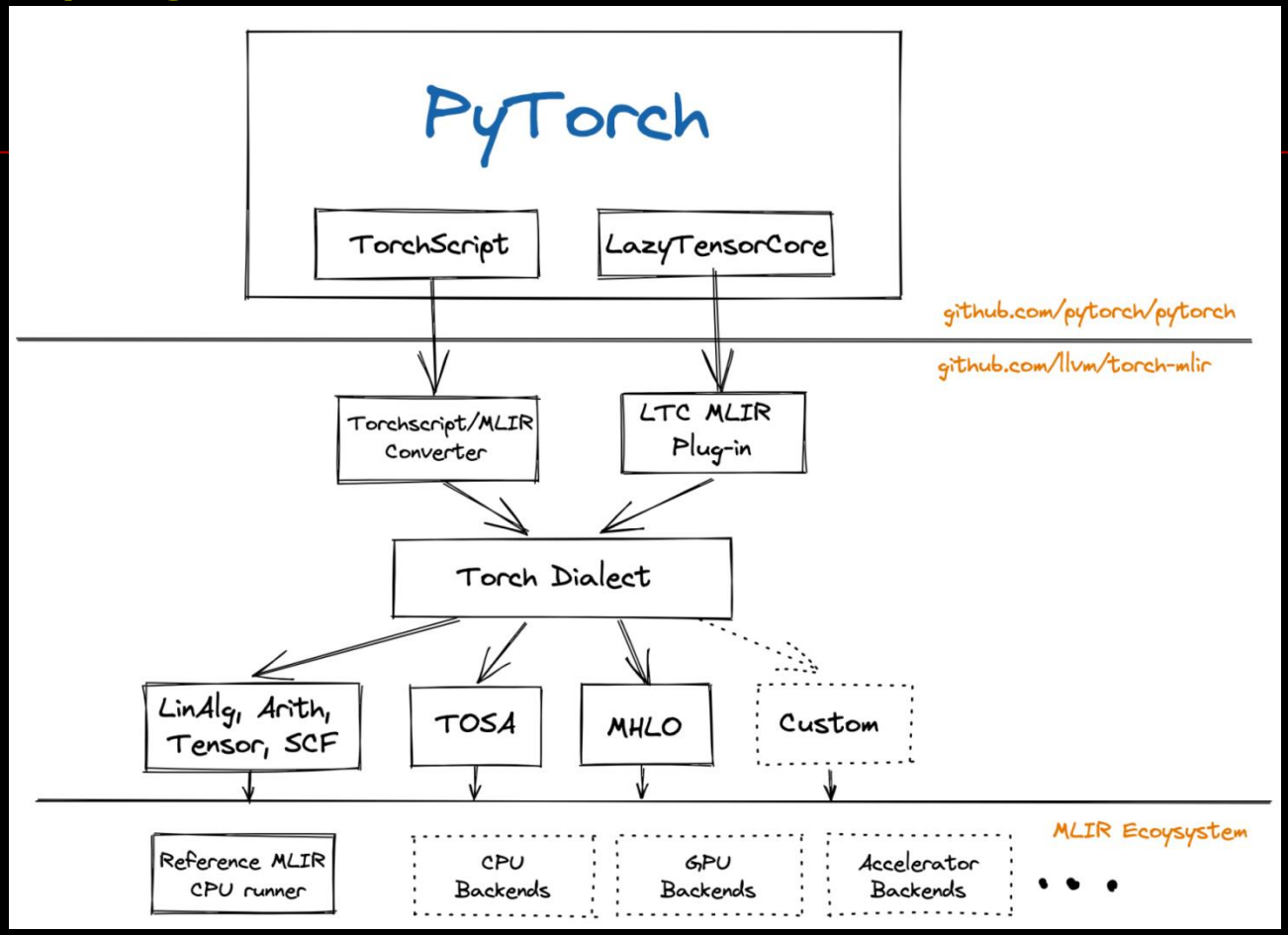
- **Frontends/Backends**



Source: https://mlir.llvm.org/OpenMeetings/2021-10-07-The-Torch-MLIR-project.pdf

Architecture & Design

- https://github.com/llvm/torch-mlir



TorchScript This is the most tested path down to Torch MLIR Dialect, and the PyTorch ecosystem is converging on using TorchScript IR as a lingua franca.

Lazy Tensor Core is a tracing system in PyTorch which is supported as an entry point to Torch-MLIR. After registering an LTC backend, all operations performed on lazy tensors are recorded and handed off to the backend implementation.

1.1.1.2 CIRCT

- https://circt.llvm.org/
  Circuit IR Compilers and Tools.
- Motivation

The EDA industry has well-known and widely used proprietary and open source tools. However, these tools are inconsistent, have usability concerns, and were not designed together into a common platform. Furthermore these tools are generally built with Verilog (also VHDL) as the IRs that they interchange. Verilog has well known design issues, and limitations, e.g. suffering from poor location tracking support.

The CIRCT project is an (experimental!) effort looking to apply MLIR and the LLVM development methodology to the domain of hardware design tools. Many of us dream of having reusable infrastructure that is modular, uses library-based design techniques, is more consistent, and builds on the best practices in compiler infrastructure and compiler design techniques.

By working together, we hope that we can build a new center of gravity to draw contributions from the small (but enthusiastic!) community of people who work on open hardware tooling. In turn we hope this will propel open tools forward, enables new higher-level abstractions for hardware design, and perhaps some pieces may even be adopted by proprietary tools in time.

- https://circt.llvm.org/docs/Charter/

LLHD

- http://www.llhd.io/

The Low Level Hardware Description language is an intermediate representation for digital circuit descriptions, together with an accompanying simulator and SystemVerilog/VHDL compiler.

LLHD separates input languages from EDA tools such as simulators, synthesizers, and placers/routers. This makes writing such tools easier, allows for more rich and complex HDLs, and does not require vendors to agree upon the implementation of a language.

Behavioural LLHD aims at capturing circuit descriptions in higher-level HDLs as easily as possible. It allows for simulation constructs and test benches to be fully represented, including assertions, file I/O, or formal verification information as intrinsics.
Structural LLHD limits the description to the parts that describe the input to output relations of a design. This covers essentially everything that can be represented by an entity (see § 4 for a more technical description).
Netlist LLHD further limits the description to just entities and instructions to instantiate and connect subcircuits. More specifically allowed are just the `entity` construct, as well as signal creation (`sig`), connection (`con`), delay (`del`), and sub-circuit instantiation (`inst`).

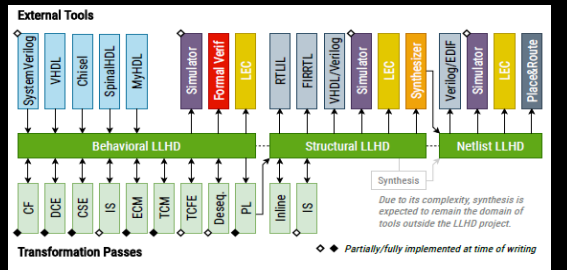


Figure 4. Optimization and transformation passes on the different IR levels of LLHD. See § 4 for a detailed description.

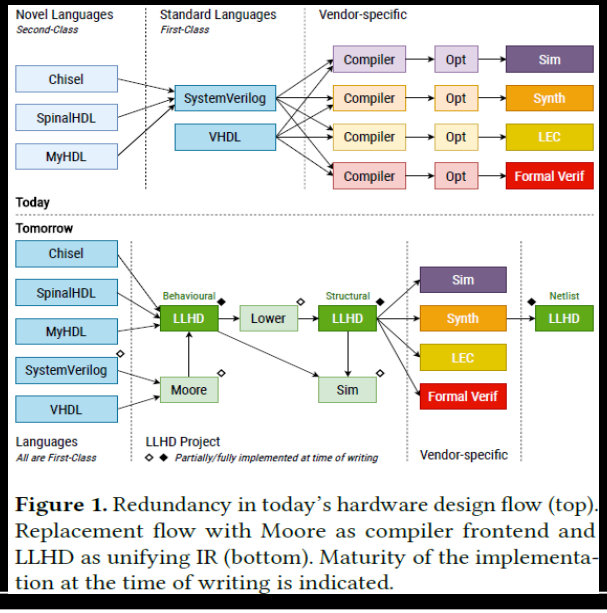
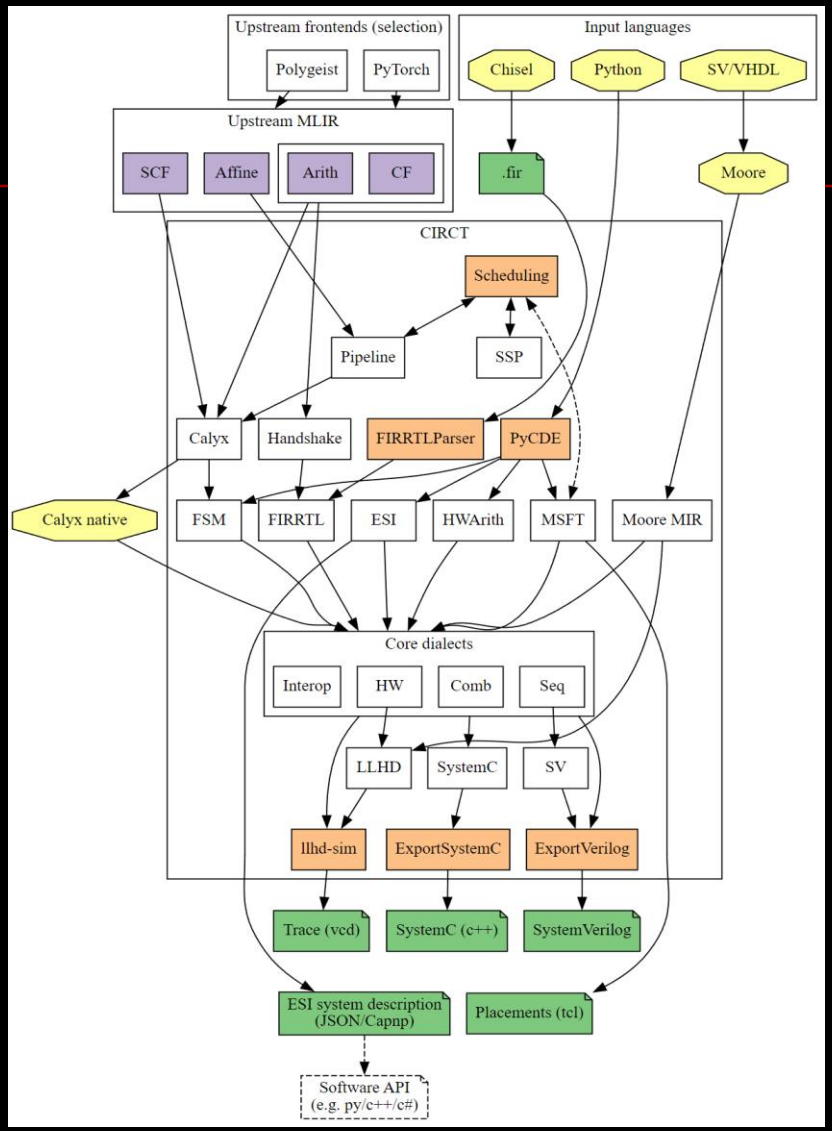


Figure 1. Redundancy in today's hardware design flow (top). Replacement flow with Moore as compiler frontend and LLHD as unifying IR (bottom). Maturity of the implementation at the time of writing is indicated.

- …

Source: https://arxiv.org/pdf/2004.03494

Dialects and how they interact

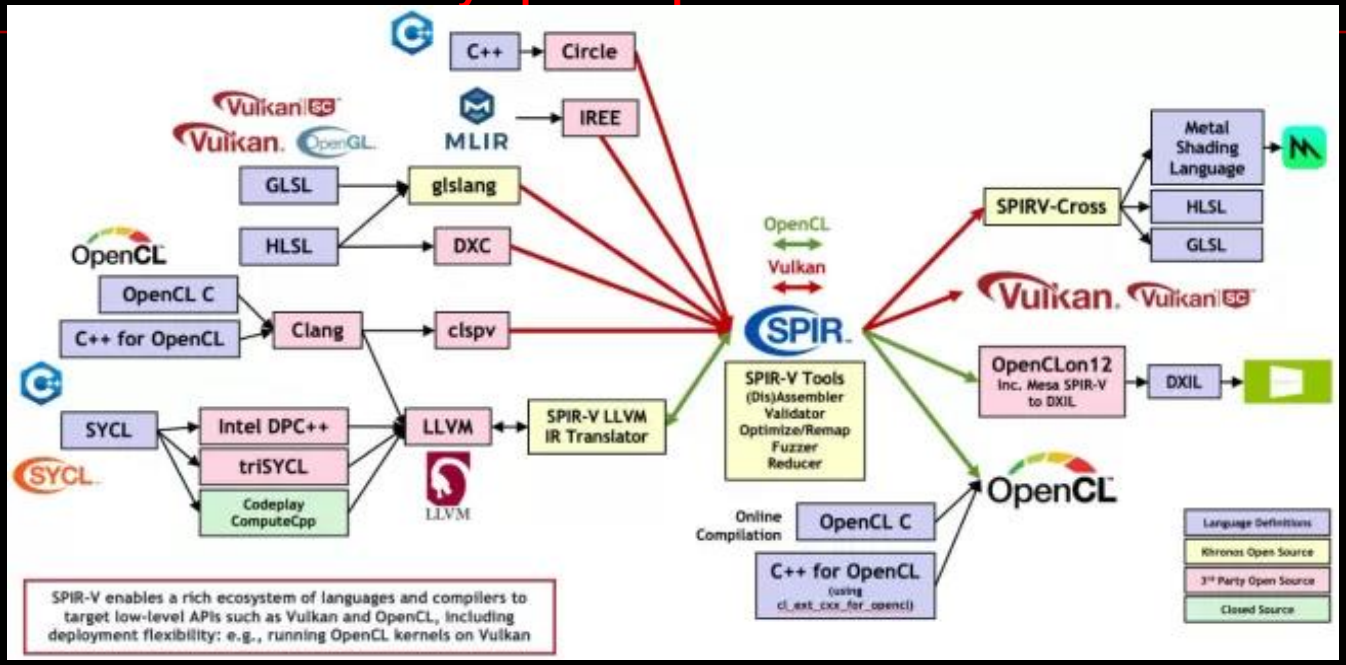


Source: https://circt.llvm.org/includes/img/dialects.svg

1.1.2 Initial SPIR-V Backend Code Lands in LLVM 15

- https://www.phoronix.com/news/LLVM-15-SPIR-V-Backend

  SPIR-V is at the heart of standards/software and with a mainline LLVM back-end can ultimately open it up to even more.

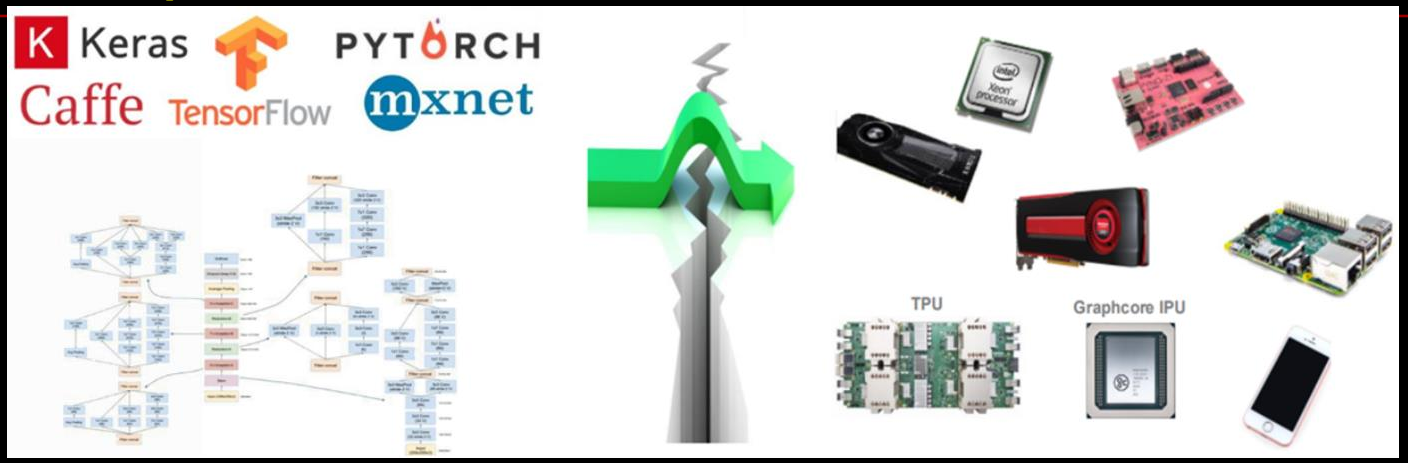


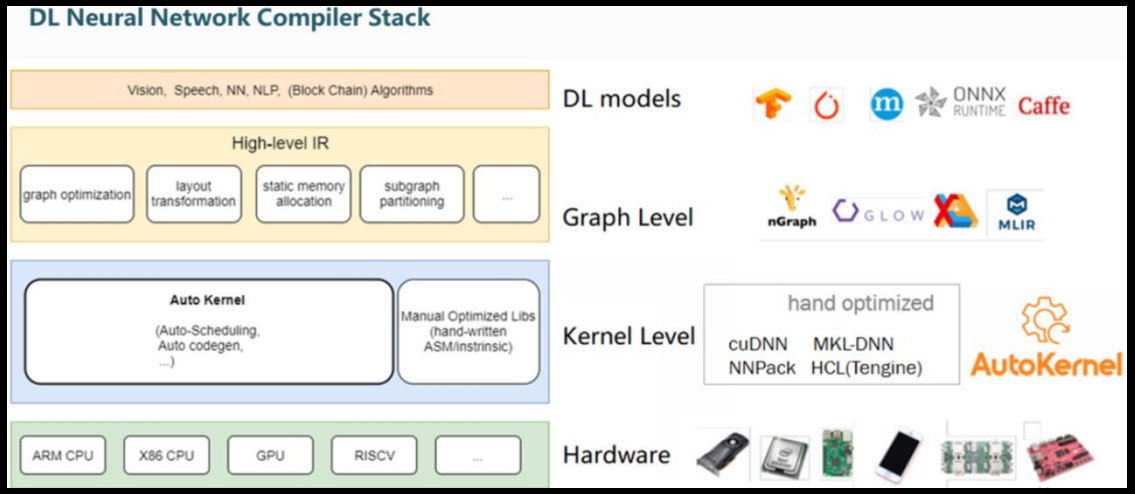
- https://github.com/llvm/llvm-project/commit/7fd4622d4801cc14823ecde678d55b6f3a106eb9
- https://github.com/KhronosGroup/LLVM-SPIRV-Backend
- https://github.com/KhronosGroup/SPIRV-LLVM-Translator
- …
- https://llvm.org/docs/SPIRVUsage.html

1.2 AI Compiler

Good Resources

- https://autokernel-docs-en.readthedocs.io/en/latest/blog/ai_compiler%20overview.html





- ...

1.3 Wasm

- https://en.wikipedia.org/wiki/WebAssembly

C source code and corresponding WebAssembly

C source code	WebAssembly . wat text format	WebAssembly . wasm binary format

`int factorial(int n) {`
`  if (n == 0)`
`  return 1;`
`  else`
`  return n * factorial(n-1);`
`}`
	`(func (param i64) (result i64)`
	`  local.get 0`
	`  i64.eqz`
	`  if (result i64)`
	`  i64.const 1`
	`  else`
	`  local.get 0`
	`  local.get 0`
	`  i64.const 1`
	`  i64.sub`
	`  call 0`
	`  i64.mul`
	`  end)`
		`00 61 73 6D 01 00 00 00`
		`01 00 01 60 01 73 01 73 06`
		`03 00 01 00 02`
		`0A 00 01`
		`00 00`
		`20 00`
		`50`
		`04 7E`
		`42 01`
		`05`
		`20 00`
		`20 00`
		`42 01`
		`7D`
		`10 00`
		`7E`
		`0B`
		`0B 15 17`

- https://webassembly.org/

WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.

WebAssembly 1.0 has shipped in 4 major browser engines.

- https://github.com/WebAssembly/design
- https://webassembly.org/specs/
- https://blog.scottlogic.com/2022/06/20/state-of-wasm-2022.html
- https://platform.uno/blog/the-state-of-webassembly-2021-and-2022/
- https://www.w3.org/groups/wg/wasm
- …

Implementations

- While WebAssembly was initially designed to enable near-native code execution speed in the web browser, it has been considered valuable outside of such, in more generalized contexts.[37][38] Since WebAssembly's runtime environments (RE) are low level virtual stack machines (akin to JVM or Flash VM) that can be embedded into host applications, some of them have found a way to standalone runtime environments like Wasmtime and Wasmer.[8][13]

  Web browsers [edit]

  In November 2017, Mozilla declared support "in all major browsers",[39] after WebAssembly was enabled by default in Edge 16.[40] The support includes mobile web browsers for iOS and Android. As of July 2021, 94% of installed browsers support WebAssembly.[41] But for older browsers, Wasm can be compiled into asm.js by a JavaScript polyfill.[42]

Compilers:

WebAssembly implementations usually use either ahead-of-time (AOT) or just-in-time (JIT) compilation, but may also use an interpreter. While the first implementations have landed in web browsers, there are also non-browser implementations for general-purpose use, including Wasmer,[10] Wasmtime[40] or WAMR,[16] wasm3, WAVM, and many others.[41]

Because WebAssembly executables are precompiled, it is possible to use a variety of programming languages to make them.[42] This is achieved either through direct compilation to Wasm, or through implementation of the corresponding virtual machines in Wasm. There have been around 40 programming languages reported to support Wasm as a compilation target.[43]

Emscripten compiles C and C++ to Wasm[26] using the Binaryen and LLVM as backend.[44] The Emscripten SDK can compile any LLVM-supported languages (such as C, C++ or Rust, among others) source code into a binary file which runs in the same sandbox as JavaScript code.[note 1] Emscripten provides bindings for several commonly used environment interfaces like WebGL.

As of version 8, a standalone Clang can compile C and C++ to Wasm.[49]

Its initial aim is to support compilation from C and C++,[50] though support for other source languages such as Rust, .NET languages[51][52][43] and AssemblyScript[53] (TypeScript-like) is also emerging. After the MVP release, there are plans to support multithreading and garbage collection[54][55] which would make WebAssembly a compilation target for garbage-collected programming languages like C# (supported via Blazor), F# (supported via Bolero[56] with help of Blazor), Python, and even JavaScript where the browser's just-in-time compilation speed is considered too slow. A number of other languages have some support including Python,[57] Julia,[58][59][60] and Ruby.[61]

- https://github.com/appcypher/awesome-wasm-langs
- https://github.com/appcypher/awesome-wasm-runtimes

Limitations

1. In general, WebAssembly does not allow direct interaction with the DOM. All interaction must flow through JavaScript interop.
2. Absence of garbage collection (although there are plans to address this.)
3. Security considerations (discussed below)

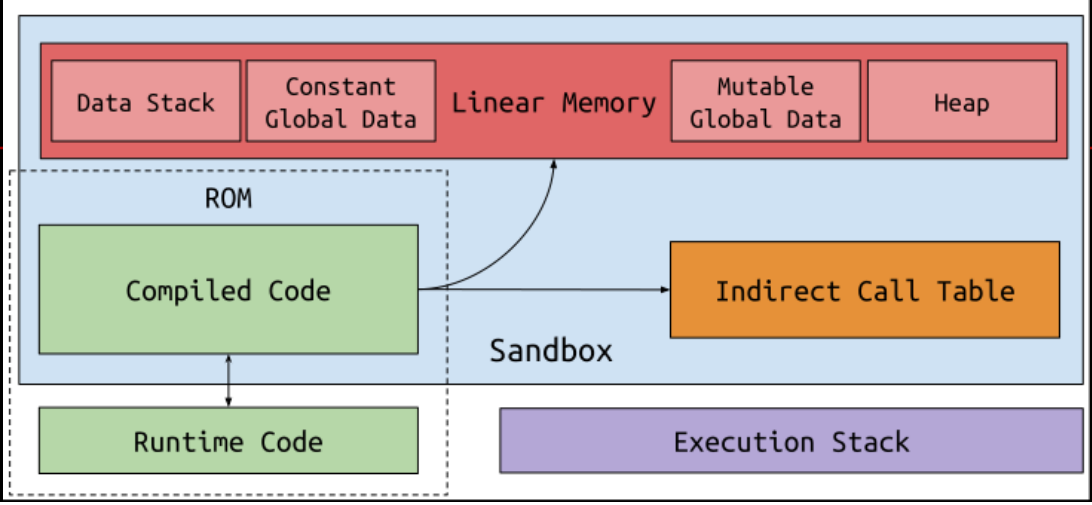
WebAssembly is supported on desktops, and mobile, but on the latter, in practice (for non-small memory allocations, such as with Unity game engine) there are "grave limitations that make many applications infeasible to be *reliably* deployed on mobile browsers [..] Currently allocating more than ~300MB of memory is not reliable on Chrome on Android without resorting to Chrome-specific workarounds, nor in Safari on iOS."[62]

There is no direct Document Object Model (DOM) access; however, it is possible to create proxy functions for this, for example through stdweb[63] or web_sys[64] when using the Rust language.

All major web browsers allow WebAssembly if Content-Security-Policy is not specified, or if "unsafe-eval" is used, but otherwise the major web browsers behave differently.[65] In practice WebAssembly can't be used on Chrome without "unsafe-eval",[66][67] while a worker thread workaround is available.[67]

1.3.1 Runtime

- Sandboxing



Wasm uses a co-design between the compiler, and the dynamic checks of the runtime system to provide the sandbox that isolates the surrounding system from the logic of the contained code. The figure depicts the main aspects of the sandbox. These include:

- *Linear memory* that holds all memory accessed by the sandbox. The compiler emits code that checks that all loads and stores remain within the linear memory, thus preventing errant accesses outside the sandbox. Linear memory is expandable much like a traditional heap.
- The *indirect function call table* that facilitates function pointer calls. To ensure that function pointer invocations are safe (to code generated by the compiler), function pointers reference an *offset* into the table. Each entry includes the type of the function, and ensures that function invocations are well-typed.
- The separation of the *execution stack* -- used to track function calls -- and the *data stack* -- used to contain stack-allocated data that can be referenced, thus must be in linear memory.

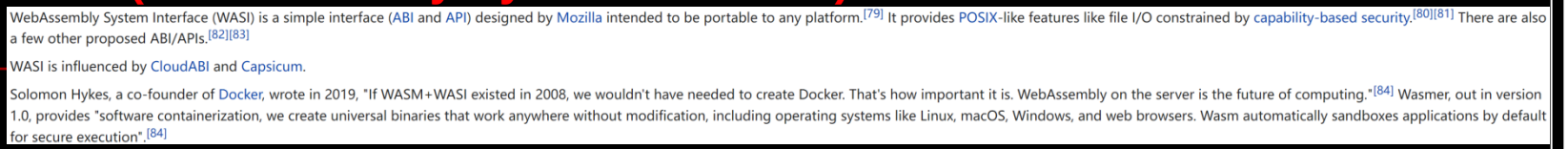
The first of these ensures the proper memory isolation of the sandbox, while the latter two provide control-flow integrity of the sandbox.

Source: https://github.com/gwsystems/aWsm/blob/master/doc/design.md

1.3.2 Beyond the browser

1.3.2.1 WASI

- WASI (WebAssembly System Interface)



WebAssembly System Interface (WASI) is a simple interface (ABI and API) designed by Mozilla intended to be portable to any platform.[79] It provides POSIX-like features like file I/O constrained by capability-based security.[80][81] There are also a few other proposed ABI/APIs.[82][83]

WASI is influenced by CloudABI and Capsicum.

Solomon Hykes, a co-founder of Docker, wrote in 2019, "If WASM+WASI existed in 2008, we wouldn't have needed to create Docker. That's how important it is. WebAssembly on the server is the future of computing."[84] Wasmer, out in version 1.0, provides "software containerization, we create universal binaries that work anywhere without modification, including operating systems like Linux, macOS, Windows, and web browsers. Wasm automatically sandboxes applications by default for secure execution".[84]

- https://wasi.dev/
- https://github.com/WebAssembly/WASI
- https://webassembly.org/docs/non-web/
- https://github.com/bytecodealliance/wasmtime/blob/main/docs/WASI-documents.md
- https://hacks.mozilla.org/2019/03/standardizing-wasi-a-webassembly-system-interface/
- https://training.linuxfoundation.org/announcements/wasi-bringing-webassembly-way-beyond-browsers
- https://www.zdnet.com/article/microsoft-google-back-bytecode-alliance-to-move-webassembly-beyond-the-browser/
- …

1.3.3 Wasm 2.0

- https://www.w3.org/blog/news/archives/9509

The WebAssembly Working Group has published three First Public Working Drafts:

- WebAssembly Core Specification – Version 2.0 describes version 2.0 of the core WebAssembly standard, a safe, portable, low-level code format designed for efficient execution and compact representation.
- WebAssembly JavaScript Interface – Version 2.0 provides an explicit JavaScript API for interacting with WebAssembly.
- WebAssembly Web API – Version 2.0 describes the integration of WebAssembly with the broader web platform.

The WebAssembly Working Group maintains a list of proposals finished since the last Recommendation and monitors the progress of "in-flight" proposals.

- https://github.com/WebAssembly/proposals
- …

Roadmap

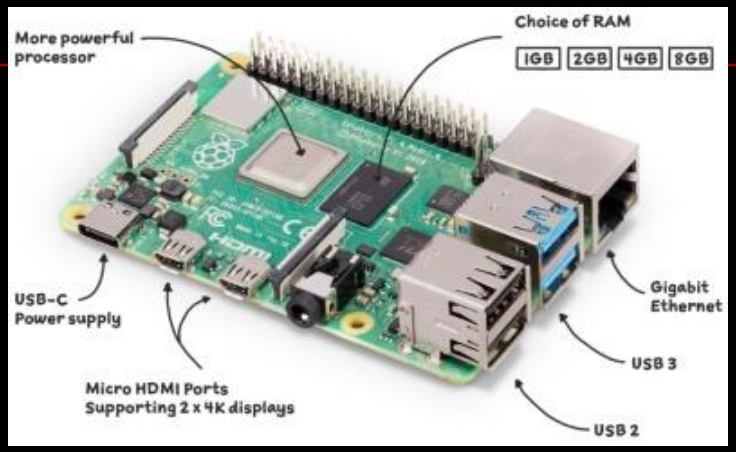
- https://webassembly.org/roadmap/

	Your browser	Chrome	Firefox	Safari	Wasmtime	Wasmer	Node.js	Deno	wasm2c
Standardized features									
JS BigInt to Wasm i64 integration	✓	85	78	14.1[f]	N/A	N/A	✓	✓	N/A
Bulk memory operations	✓	75	79	15	✓	✓	✓	✓	✓
Multi-value	✓	85	78	✓	✓	✓	✓	✓	✓
Import & export of mutable globals	✓	74	61	✓	✓	✓	✓	✓	✓
Reference types	✓	96	79	15	✓	✓	⚑[m]	✓	✓
Non-trapping float-to-int conversions	✓	75	64	15	✓	✓	✓	✓	✓
Sign-extension operations	✓	74	62	14.1[f]	✓	✓	✓	✓	✓
Fixed-width SIMD	✓	91	89	✗	✓	✓	✓	✓	✗
In-progress proposals									
Exception handling	✓	95	100	15.2	✗	✗	⚑[k]	✓	⚑[o]
Extended constant expressions	✗	⚑[a]	⚑[e]	✗	✗	✗	⚑[l]	✗	✗
Memory64	✗	⚑[b]	⚑[e]	✗	⚑[g]	✗	✗	✗	✗
Multiple memories	?	✗	✗	✗	⚑[h]	✗	✗	✗	⚑[h]
Module Linking	?	✗	✗	✗	⚑[i]	✗	✗	✗	✗
Relaxed SIMD	?	⚑[c]	⚑[e]	✗	✗	✗	✗	✗	✗
Tail calls	✗	⚑[d]	✗	✗	✗	✗	⚑[n]	✗	✗
Threads and atomics	✓	74	79	14.1[f]	✗	⚑[j]	✓	✓	✗
Type reflection	?	✗	⚑[e]	✗	✗	✗	✗	✗	✗

2) Testbed

2.1 Raspberry Pi 4(8GB LPDDR4) with Fedora 37

- HW env



SW env

```
[mydev@fedora /]$ uname -a
Linux fedora 6.0.11-300.fc37.aarch64 #1 SMP PREEMPT_DYNAMIC Fri Dec 2 20:14:38 UTC 2022 aarch64 aarch64 aarch64 GNU/Linux
[mydev@fedora /]$
[mydev@fedora /]$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7826         686        1277          78        5862        6667
Swap:           7825         442        7383
[mydev@fedora /]$
```

```
[mydev@fedora /]$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/aarch64-redhat-linux/12/lto-wrapper
Target: aarch64-redhat-linux
Configured with: ../configure --enable-bootstrap --enable-languages=c,c++,fortran,objc,obj-c++,ada,go,d,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-shared --enable-threads=posix --enable-checking=release --enable-multilib --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-libstdcxx-backtrace --with-linker-hash-style=gnu --enable-plugin --enable-initfini-array --with-isl=/builddir/build/BUILD/gcc-12.2.1-20221121/obj-aarch64-redhat-linux/isl-install --enable-gnu-indirect-function --build=aarch64-redhat-linux --with-build-config=bootstrap-lto --enable-link-serialization=1
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 12.2.1 20221121 (Red Hat 12.2.1-4) (GCC)
[mydev@fedora /]$
[mydev@fedora /]$ clang -v
clang version 15.0.4 (Fedora 15.0.4-1.fc37)
Target: aarch64-redhat-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
Found candidate GCC installation: /usr/bin/../lib/gcc/aarch64-redhat-linux/12
Selected GCC installation: /usr/bin/../lib/gcc/aarch64-redhat-linux/12
Candidate multilib: .;@m64
Selected multilib: .;@m64
[mydev@fedora /]$
```

```
[mydev@fedora /]$ python -V
Python 3.11.0
[mydev@fedora /]$
[mydev@fedora /]$ java --version
openjdk 19 2022-09-20
OpenJDK Runtime Environment GraalVM CE 23.0.0-dev (build 19+36-jvmci-23.0-b01)
OpenJDK 64-Bit Server VM GraalVM CE 23.0.0-dev (build 19+36-jvmci-23.0-b01, mixed mode, sharing)
[mydev@fedora /]$
[mydev@fedora /]$ gu list
ComponentId              Version             Component name                Stability                     Origin
------------------------------------------------------------------------------------------------------------------------
graalvm                  23.0.0-dev          GraalVM Core                  Experimental
js                       23.0.0-dev          Graal.js                      Experimental
llvm                     23.0.0-dev          LLVM Runtime Core             Experimental
llvm-toolchain           23.0.0-dev          LLVM.org toolchain            Experimental
native-image             23.0.0-dev          Native Image                  Experimental
native-image-llvm-backend23.0.0-dev          Native Image LLVM Backend     Experimental
nodejs                   23.0.0-dev          Graal.nodejs                  Experimental
python                   23.0.0-dev          GraalVM Python                Experimental
visualvm                 23.0.0-dev          VisualVM                      Experimental
wasm                     23.0.0-dev          GraalWasm                     Experimental
[mydev@fedora /]$
```

```
[mydev@fedora /]$ dotnet --version
7.0.100
[mydev@fedora /]$ mono --version
Mono JIT compiler version 6.12.0.182 (tarball Thu Jul 21 23:55:50 UTC 2022)
Copyright (C) 2002-2014 Novell, Inc, Xamarin Inc and Contributors. www.mono-project.com
        TLS:           __thread
        SIGSEGV:       normal
        Notifications: epoll
        Architecture:  arm64
        Disabled:      none
        Misc:          softdebug
        Interpreter:   yes
        LLVM:          supported, not enabled.
        Suspend:       preemptive
        GC:            sgen (concurrent by default)
[mydev@fedora /]$ _
```

```
[mydev@fedora /]$ rustup show
Default host: aarch64-unknown-linux-gnu
rustup home:  /home/mydev/.rustup

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

aarch64-unknown-linux-gnu
wasm32-unknown-unknown

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

stable-aarch64-unknown-linux-gnu (default)
rustc 1.65.0 (897e37553 2022-11-02)

[mydev@fedora /]$
```

...

2.1.1 Fedora

- A Linux distribution developed by the community-supported Fedora Project which is sponsored primarily by Red Hat.
- https://en.wikipedia.org/wiki/Fedora_(operating_system)
- https://getfedora.org/
- https://alt.fedoraproject.org/alt/
- https://spins.fedoraproject.org/
- https://fedoraproject.org/wiki/Architectures/ARM
- https://fedoramagazine.org/
- https://silverblue.fedoraproject.org/
- https://fedoraproject.org/wiki/Changes/RaspberryPi4
- Developer friendly!

2.1.1.1 EDA support

- https://fedoraproject.org/wiki/Electronic_Lab

Installation

- sudo dnf groupinstall "Electronic Lab“

```
[mydev@fedora /]$ dnf -v group info "Electronic Lab"
...
```

```
Group: Electronic Lab
 Group-Id: electronic-lab
 Description: Design and simulation tools for hardware engineers
 Default Packages:
   CUnit-2.1.3-25.fc36.aarch64
   LabPlot-2.8.1-6.fc36.aarch64
   acpica-tools-20220331-3.fc36.aarch64
   alliance-5.1.1-25.20160506gitd8c05cd.fc36.aarch64
   arduino
   avarice-2.13-15.fc36.aarch64
   avr-binutils-1:2.37-3.fc36.aarch64
   avr-gcc-1:11.2.0-1.fc36.aarch64
   avr-gcc-c++-1:11.2.0-1.fc36.aarch64
   avra-1.3.0-17.fc36.aarch64
   avrdude-6.4-3.fc36.aarch64
   cgnslib-4.2.0-6.fc36.aarch64
   dfu-programmer-0.7.2-10.fc36.aarch64
   dia-CMOS-0.1-20.fc36.noarch
   dia-Digital-0.1-20.fc36.noarch
   dia-electric2-0.1-20.fc36.noarch
   dia-electronic-0.1-20.fc36.noarch
   electric
   emacs-vregs-mode-1.470-28.fc36.noarch
   espresso-ab-1.0-24.fc36.aarch64
   flterm-1.2-19.fc36.aarch64
   freeDiameter-1.5.0-4.fc36.aarch64
   fritzing-0.9.10-1.20220514.fc36.aarch64
   gerbv-2.9.2-1.fc36.aarch64
   ghc-chalmers-lava2000-devel
   ghdl-0.38~dev-15.20201208git83dfd49.fc36.aarch64
   gnucap-0.35-35.fc36.aarch64
   gnuradio-3.10.1.0-2.fc36.aarch64
   gnusim8085-1.4.1-4.fc36.aarch64
   gpsim-0.31.0-6.fc36.aarch64
   gputils-1.5.0-9.fc36.aarch64
   gr-osmosdr-0.2.3-28.20210217gita100eb02.fc36.aarch64
   gsim85-0.3-28.fc36.aarch64
   gspiceui
   gtkterm-1.1.1-2.fc36.aarch64
   gtkwave-3.3.111-3.fc36.aarch64
   hct-0.7.60-33.fc36.noarch
   hiredis-1.0.2-2.fc36.aarch64
   icaro-2.0-10.fc36.noarch
   icestorm-0-0.22.20220705git2bc5417.fc36.aarch64
   irsim-9.7.104-7.fc36.aarch64
   iverilog-11.0-3.fc35.aarch64
   kicad-1:6.0.8-1.fc36.aarch64
   ktechlab-0.50.0-1.fc36.aarch64
   linsmith-0.99.31-9.fc36.aarch64
   magic-8.3.324-1.fc36.aarch64
   magic-doc-8.3.324-1.fc36.aarch64
   mcu8051ide
   mot-adms-2.3.7-4.fc36.aarch64
   netgen-1.3.7-43.fc36.aarch64
   nextpnr-1-12.20220912gitf1349e1.fc36.aarch64
   ngspice-37-1.fc36.aarch64
   openocd-0.11.0-2.fc36.1.aarch64
   pcb-4.2.0-9.fc36.aarch64
   perl-Hardware-Verilog-Parser-0.13-38.fc36.noarch
   perl-Hardware-Vhdl-Lexer-1.00-39.fc36.noarch
   perl-Hardware-Vhdl-Parser-0.12-39.fc36.noarch
   perl-Hardware-Vhdl-Tidy
   perl-ModelSim-List-0.06-37.fc36.noarch
   perl-Perlilog-1.0-18.fc36.noarch
   perl-SystemC-Vregs-1.470-28.fc36.noarch
   perl-Verilog-CodeGen
   perl-Verilog-Perl-3.478-3.fc36.aarch64
   perl-Verilog-Readmem-0.05-19.fc36.noarch
   picocom-3.1-11.fc36.aarch64
   pulseview-0.4.2-10.fc36.aarch64
   python3-migen-0.9.2-14.20220705git0fb9173.fc36.noarch
   python3-myhdl-0.11-7.fc36.noarch
   qucs-0.0.19-6.fc36.aarch64
   rtl-sdr-0.6.0-11.fc36.aarch64
   sdcc-4.1.0-3.fc36.aarch64
   sigrok-cli-0.7.2-3.fc36.aarch64
   sigrok-firmware-fx2lafw-0.1.7-6.fc36.noarch
   smartsim-1.4-18.fc36.aarch64
   srecord-1.64-22.fc36.aarch64
   tclspice-37-1.fc36.aarch64
   tkcvs-8.2.3-15.fc36.noarch
   tkgate-2.0-36.beta10.fc36.aarch64
   toped-0.9.81-31.svn2211.fc36.aarch64
   trellis-1.2.1-10.20220821git98e0ea3.fc36.aarch64
   uisp-20050207-31.fc36.aarch64
   verilator-4.108-3.fc35.aarch64
   vhd2vl-2.5-14.fc36.aarch64
   vrq
   xcircuit-3.10.30-4.fc36.aarch64
   xorg-x11-fonts-100dpi-7.5-33.fc36.noarch
   xorg-x11-fonts-ISO8859-1-100dpi-7.5-33.fc36.noarch
   xorg-x11-fonts-ISO8859-9-100dpi-7.5-33.fc36.noarch
   xorg-x11-fonts-Type1-7.5-33.fc36.noarch
   yosys-0.21-1.20220912gitd98738d.fc36.aarch64
 Optional Packages:
   kdesvn-2.1.0-5.fc36.aarch64
   minicom-2.8-1.fc36.aarch64
```

- sudo dnf install libevent libevent-devel boost boost-devel verilator yosys z3 z3-devel z3-libs java-z3 python-z3
- sudo dnf install python3-pytest dfu-util json-c json-c-devel
- sudo dnf install libftdi libftdi-devel python3-libftdi python3-pyftdi
- ...

2.1.1.2 Fedora 37

- https://fedoraproject.org/wiki/Releases/37/ChangeSet
- https://www.phoronix.com/news/Fedora-37-Released

Raspberry Pi 4

- **Fedora 37 To Offer Official Support On Raspberry Pi 4 Devices**
  https://www.phoronix.com/news/Raspberry-Pi-4-Fedora-37

With the open-source OpenGL and Vulkan support, these latest Raspberry Pi boards are suitable for Fedora Workstation usage. The latest milestones there are V3DV just having crossed Vulkan 1.2 and Raspberry Pi 4 V3D DRM/KMS driver support in Linux 6.0. However, there still are upstream issues surrounding WiFi support for the Raspberry Pi 400, the Raspberry Pi CM4 not necessarily being very suitable for desktop use but more edge/IoT/embedded, and some device support such as around audio may be problematic.

> ***The Raspberry Pi 4 is a widely available, reasonably priced device. It has worked well in Fedora for some time in IoT and Server use cases, and now with a fully accelerated graphics stack available it's a great device from a price-per-performance perspective, and it has a wide ecosystem, so fully supporting this in Fedora makes a compelling case.***

A month ago there was the Fedora 37 change proposal for Fedora to officially support the Raspberry Pi 4, including its accelerated Broadcom graphics and to better advertise Fedora for the Raspberry Pi. The Fedora Engineering and Steering Committee (FESCo) has now signed off on this "official" support for the Raspberry Pi 4.

The Raspberry Pi 4 to date hasn't been a significant focus for Fedora Workstation due to various patches not being upstreamed-- most notably, waiting on the open-source 3D graphics bits to be upstreamed in the kernel. Now though that those upstream bits are coming together, Fedora 37 will be focusing on advertising its support for the Raspberry Pi 4 Model B as well as the Raspberry Pi 400 and Raspberry Pi CM4 compute module.

2.2 Toolchain for RISC-V development
2.2.1 Install from distribution package
2.2.1.1 Fedora

```
[mydev@fedora /]$ dnf search riscv
============================================== Name & Summary Matched: riscv ===============
binutils-riscv64-linux-gnu.aarch64 : Cross-build binary utilities for riscv64-linux-gnu
gcc-c++-riscv64-linux-gnu.aarch64 : Cross-build binary utilities for riscv64-linux-gnu
gcc-riscv64-linux-gnu.aarch64 : Cross-build binary utilities for riscv64-linux-gnu
qemu-user-static-riscv.aarch64 : QEMU user mode emulation of riscv qemu targets static build
================================================== Name Matched: riscv ====================
qemu-system-riscv.aarch64 : QEMU system emulator for RISC-V
qemu-system-riscv-core.aarch64 : QEMU system emulator for RISC-V
[mydev@fedora /]$
```

```
[mydev@fedora GCC]$ sudo dnf --installroot=/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora --releasever=/ install gcc-riscv64-linux-gnu
[sudo] password for mydev:
Docker CE Stable - aarch64                                                    3.4 kB/s | 8.8 kB     00:02
Fedora 37 - aarch64                                                           7.6 MB/s |  61 MB     00:08
Fedora 37 openh264 (From Cisco) - aarch64                                     663  B/s | 2.5 kB     00:03
Fedora Modular 37 - aarch64                                                   173 kB/s | 2.9 MB     00:17
Fedora 37 - aarch64 - Updates                                                 1.6 MB/s |  16 MB     00:10
Fedora Modular 37 - aarch64 - Updates                                          11 kB/s | 844 kB     01:13
sbt-rpm                                                                       4.2 kB/s |  19 kB     00:04
Last metadata expiration check: 0:00:01 ago on Sat 10 Dec 2022 06:24:32 AM PST.
Dependencies resolved.
================================================================================================================================
 Package                                 Architecture                 Version                          Repository          Size
================================================================================================================================
Installing:
 gcc-riscv64-linux-gnu                   aarch64                      12.1.1-1.fc37.1                  fedora              24 M
Installing dependencies:
 basesystem                              noarch                       11-14.fc37                       fedora             7.0 k
 bash                                    aarch64                      5.2.9-3.fc37                     updates            1.8 M
 binutils-riscv64-linux-gnu              aarch64                      2.38-4.fc37                      fedora             2.7 M
 cross-binutils-common                   noarch                       2.38-4.fc37                      fedora             2.2 M
 cross-gcc-common                        noarch                       12.1.1-1.fc37.1                  fedora             2.3 M
 fedora-gpg-keys                         noarch                       37-1                             fedora             125 k
 fedora-release                          noarch                       37-15                            updates             11 k
 fedora-release-common                   noarch                       37-15                            updates             21 k
 fedora-release-identity-basic           noarch                       37-15                            updates             11 k
 fedora-repos                            noarch                       37-1                             fedora             9.6 k
 filesystem                              aarch64                      3.18-2.fc37                      fedora             1.1 M
 glibc                                   aarch64                      2.36-8.fc37                      updates            1.8 M
 glibc-common                            aarch64                      2.36-8.fc37                      updates            361 k
 glibc-minimal-langpack                  aarch64                      2.36-8.fc37                      updates             86 k
 gmp                                     aarch64                      1:6.2.1-3.fc37                   fedora             265 k
 isl                                     aarch64                      0.16.1-16.fc37                   fedora             835 k
 libgcc                                  aarch64                      12.2.1-4.fc37                    updates             94 k
 libmpc                                  aarch64                      1.2.1-5.fc37                     fedora              62 k
 libstdc++                               aarch64                      12.2.1-4.fc37                    updates            763 k
 libzstd                                 aarch64                      1.5.2-3.fc37                     fedora             267 k
 mpfr                                    aarch64                      4.1.0-10.fc37                    fedora             240 k
 ncurses-base                            noarch                       6.3-3.20220501.fc37              fedora              86 k
 ncurses-libs                            aarch64                      6.3-3.20220501.fc37              fedora             318 k
 setup                                   noarch                       2.14.1-2.fc37                    fedora             149 k
 tzdata                                  noarch                       2022f-1.fc37                     updates            715 k
 zlib                                    aarch64                      1.2.12-5.fc37                    fedora              93 k
Installing weak dependencies:
 glibc-gconv-extra                       aarch64                      2.36-8.fc37                      updates            1.7 M

Transaction Summary
================================================================================================================================
Install  28 Packages

Total download size: 42 M
Installed size: 175 M
```

```
[mydev@fedora /]$ /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora/usr/bin/riscv64-linux-gnu-gcc -v
Using built-in specs.
COLLECT_GCC=/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora/usr/bin/riscv64-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora/usr/bin/../libexec/gcc/riscv64-linux-gnu/12/lto-wrapper
Target: riscv64-linux-gnu
Configured with: ../gcc-12.1.1-20220507/configure --bindir=/usr/bin --build=aarch64-redhat-linux-gnu --datadir=/usr/share --disable-decimal-float --disable-dependency-tracking --disab
le-gold --disable-libgcj --disable-libgomp --disable-libmpx --disable-libquadmath --disable-libssp --disable-libunwind-exceptions --disable-shared --disable-silent-rules --disable-sjl
j-exceptions --disable-threads --with-ld=/usr/bin/riscv64-linux-gnu-ld --enable-__cxa_atexit --enable-checking=release --enable-gnu-unique-object --enable-initfini-array --enable-lang
uages=c,c++ --enable-linker-build-id --enable-lto --enable-nls --enable-obsolete --enable-plugin --enable-targets=all --exec-prefix=/usr --host=aarch64-redhat-linux-gnu --includedir=/
usr/include --infodir=/usr/share/info --libexecdir=/usr/libexec --localstatedir=/var --mandir=/usr/share/man --prefix=/usr --program-prefix=riscv64-linux-gnu- --sbindir=/usr/sbin --sh
aredstatedir=/var/lib --sysconfdir=/etc --target=riscv64-linux-gnu --with-bugurl=http://bugzilla.redhat.com/bugzilla/ --with-gcc-major-version-only --with-isl --with-newlib --with-plu
gin-ld=/usr/bin/riscv64-linux-gnu-ld --with-sysroot=/usr/riscv64-linux-gnu/sys-root --with-system-libunwind --with-system-zlib --without-headers --with-linker-hash-style=gnu
Thread model: single
Supported LTO compression algorithms: zlib zstd
gcc version 12.1.1 20220507 (Red Hat Cross 12.1.1-1) (GCC)
[mydev@fedora /]$
```

```
[mydev@fedora /]$ tree -L 1 /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora
/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora
├── afs
├── bin → usr/bin
├── boot
├── dev
├── etc
├── home
├── lib → usr/lib
├── lib64 → usr/lib64
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin → usr/sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

```
[mydev@fedora /]$ ll /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora/bin
lrwxrwxrwx. 1 root root 7 Aug  9 06:28 /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora/bin → usr/bin/
[mydev@fedora /]$
[mydev@fedora /]$ ll /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Fedora/usr/bin
total 47780
dr-xr-xr-x. 1 root root     1514 Dec 10 06:38 ./
drwxr-xr-x. 1 root root      134 Dec 10 06:38 ../
-rwxr-xr-x. 1 root root       33 Nov 18 06:27 alias*
-rwxr-xr-x. 1 root root  1434488 Nov 18 06:27 bash*
lrwxrwxrwx. 1 root root       10 Nov 18 06:27 bashbug → bashbug-64*
-rwxr-xr-x. 1 root root     7162 Nov 18 06:27 bashbug-64*
-rwxr-xr-x. 1 root root       30 Nov 18 06:27 bg*
-rwxr-xr-x. 1 root root       30 Nov 18 06:27 cd*
-rwxr-xr-x. 1 root root       35 Nov 18 06:27 command*
-rwxr-xr-x. 1 root root       30 Nov 18 06:27 fc*
-rwxr-xr-x. 1 root root       30 Nov 18 06:27 fg*
-rwxr-xr-x. 1 root root    70216 Nov 14 12:10 gencat*
-rwxr-xr-x. 1 root root    69416 Nov 14 12:10 getconf*
-rwxr-xr-x. 1 root root    70056 Nov 14 12:10 getent*
-rwxr-xr-x. 1 root root       35 Nov 18 06:27 getopts*
-rwxr-xr-x. 1 root root       32 Nov 18 06:27 hash*
-rwxr-xr-x. 1 root root    70880 Nov 14 12:10 iconv*
-rwxr-xr-x. 1 root root       32 Nov 18 06:27 jobs*
-rwxr-xr-x. 1 root root     5347 Nov 14 12:06 ldd*
lrwxrwxrwx. 1 root root       31 Nov 14 12:06 ld.so → ../../lib/ld-linux-aarch64.so.1*
-rwxr-xr-x. 1 root root    78008 Nov 14 12:10 locale*
-rwxr-xr-x. 1 root root   343528 Nov 14 12:10 localedef*
-rwxr-xr-x. 1 root root    69544 Nov 14 12:10 pldd*
-rwxr-xr-x. 1 root root       32 Nov 18 06:27 read*
-rwxr-xr-x. 1 root root  1208792 Jul 20 20:22 riscv64-linux-gnu-addr2line*
-rwxr-xr-x. 2 root root  1142584 Jul 20 20:22 riscv64-linux-gnu-ar*
-rwxr-xr-x. 2 root root  1803384 Jul 20 20:21 riscv64-linux-gnu-as*
-rwxr-xr-x. 1 root root   136848 Jul 20 20:21 riscv64-linux-gnu-c++filt*
-rwxr-xr-x. 1 root root  1211344 Jul 21 01:09 riscv64-linux-gnu-cpp*
-rwxr-xr-x. 1 root root    70368 Jul 20 20:22 riscv64-linux-gnu-elfedit*
-rwxr-xr-x. 1 root root  1211360 Jul 21 01:09 riscv64-linux-gnu-gcc*
-rwxr-xr-x. 1 root root    71224 Jul 21 01:09 riscv64-linux-gnu-gcc-ar*
-rwxr-xr-x. 1 root root    71216 Jul 21 01:09 riscv64-linux-gnu-gcc-nm*
-rwxr-xr-x. 1 root root    71216 Jul 21 01:09 riscv64-linux-gnu-gcc-ranlib*
-rwxr-xr-x. 1 root root   613824 Jul 21 01:09 riscv64-linux-gnu-gcov*
-rwxr-xr-x. 1 root root   545464 Jul 21 01:09 riscv64-linux-gnu-gcov-dump*
-rwxr-xr-x. 1 root root   546344 Jul 21 01:09 riscv64-linux-gnu-gcov-tool*
-rwxr-xr-x. 1 root root  1275984 Jul 20 20:22 riscv64-linux-gnu-gprof*
-rwxr-xr-x. 4 root root  3647792 Jul 20 20:22 riscv64-linux-gnu-ld*
-rwxr-xr-x. 4 root root  3647792 Jul 20 20:22 riscv64-linux-gnu-ld.bfd*
-rwxr-xr-x. 1 root root 18262800 Jul 21 01:09 riscv64-linux-gnu-lto-dump*
-rwxr-xr-x. 2 root root  1209952 Jul 20 20:22 riscv64-linux-gnu-nm*
-rwxr-xr-x. 2 root root  1345712 Jul 20 20:21 riscv64-linux-gnu-objcopy*
-rwxr-xr-x. 2 root root  2727960 Jul 20 20:21 riscv64-linux-gnu-objdump*
-rwxr-xr-x. 2 root root  1142608 Jul 20 20:22 riscv64-linux-gnu-ranlib*
-rwxr-xr-x. 2 root root   936320 Jul 20 20:21 riscv64-linux-gnu-readelf*
-rwxr-xr-x. 1 root root  1076616 Jul 20 20:22 riscv64-linux-gnu-size*
-rwxr-xr-x. 1 root root  1076784 Jul 20 20:22 riscv64-linux-gnu-strings*
-rwxr-xr-x. 2 root root  1345744 Jul 20 20:21 riscv64-linux-gnu-strip*
lrwxrwxrwx. 1 root root        4 Nov 18 06:27 sh → bash*
-rwxr-xr-x. 1 root root     4282 Nov 14 12:06 sotruss*
-rwxr-xr-x. 1 root root    69608 Nov 14 12:10 sprof*
-rwxr-xr-x. 1 root root       32 Nov 18 06:27 type*
-rwxr-xr-x. 1 root root    15352 Nov 14 12:06 tzselect*
-rwxr-xr-x. 1 root root       34 Nov 18 06:27 ulimit*
-rwxr-xr-x. 1 root root       33 Nov 18 06:27 umask*
-rwxr-xr-x. 1 root root       35 Nov 18 06:27 unalias*
-rwxr-xr-x. 1 root root       32 Nov 18 06:27 wait*
-rwxr-xr-x. 1 root root    69544 Nov 14 12:10 zdump*
[mydev@fedora /]$
```

2.2.2 Build from source

2.2.2.1 GCC

- https://github.com/riscv-collab/riscv-gnu-toolchain

```
[submodule "binutils"]
        path = binutils
        url = https://sourceware.org/git/binutils-gdb.git
        branch = binutils-2_39-branch
[submodule "gcc"]
        path = gcc
        url = https://gcc.gnu.org/git/gcc.git
        branch = releases/gcc-12
[submodule "glibc"]
        path = glibc
        url = https://sourceware.org/git/glibc.git
[submodule "dejagnu"]
        path = dejagnu
        url = https://git.savannah.gnu.org/git/dejagnu.git
        branch = dejagnu-1.6.3
[submodule "newlib"]
        path = newlib
        url = https://sourceware.org/git/newlib-cygwin.git
        branch = master
[submodule "gdb"]
        path = gdb
        url = https://sourceware.org/git/binutils-gdb.git
        branch = gdb-12-branch
[submodule "qemu"]
        path = qemu
        url = https://gitlab.com/qemu-project/qemu.git
[submodule "musl"]
        path = musl
        url = https://git.musl-libc.org/git/musl
        branch = master
[submodule "spike"]
        path = spike
        url = https://github.com/riscv-software-src/riscv-isa-sim.git
        branch = master
[submodule "pk"]
        path = pk
        url = https://github.com/riscv-software-src/riscv-pk.git
        branch = master
```

Getting the sources

This repository uses submodules, but submodules will fetch automatically on demand, so `--recursive` or `git submodule update --init --recursive` is not needed.

Warning: git clone takes around 6.65 GB of disk and download size

for ELF with multilib enabled:

```
./configure --prefix=/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/ELF --enable-multilib
```

make -j1:

```
[mydev@fedora /]$ tree -L 1 /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/ELF
/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/ELF
├── bin
├── include
├── lib
├── lib64
├── libexec
├── riscv64-unknown-elf
└── share
```

```
[mydev@fedora /]$ tree /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/ELF/bin
/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/ELF/bin
├── riscv64-unknown-elf-addr2line
├── riscv64-unknown-elf-ar
├── riscv64-unknown-elf-as
├── riscv64-unknown-elf-c++
├── riscv64-unknown-elf-c++filt
├── riscv64-unknown-elf-cpp
├── riscv64-unknown-elf-elfedit
├── riscv64-unknown-elf-g++
├── riscv64-unknown-elf-gcc
├── riscv64-unknown-elf-gcc-12.2.0
├── riscv64-unknown-elf-gcc-ar
├── riscv64-unknown-elf-gcc-nm
├── riscv64-unknown-elf-gcc-ranlib
├── riscv64-unknown-elf-gcov
├── riscv64-unknown-elf-gcov-dump
├── riscv64-unknown-elf-gcov-tool
├── riscv64-unknown-elf-gdb
├── riscv64-unknown-elf-gdb-add-index
├── riscv64-unknown-elf-gprof
├── riscv64-unknown-elf-ld
├── riscv64-unknown-elf-ld.bfd
├── riscv64-unknown-elf-lto-dump
├── riscv64-unknown-elf-nm
├── riscv64-unknown-elf-objcopy
├── riscv64-unknown-elf-objdump
├── riscv64-unknown-elf-ranlib
├── riscv64-unknown-elf-readelf
├── riscv64-unknown-elf-run
├── riscv64-unknown-elf-size
├── riscv64-unknown-elf-strings
└── riscv64-unknown-elf-strip
```

directly built on RPi4 against “riscv64-unknown-elf-” for ~6.5h (depends on network speed).

for Linux with multilib enabled:

```
./configure --prefix=/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/Linux --enable-multilib
```

make linux -j1

```
[mydev@fedora /]$ tree -L 1 /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/Linux
/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/Linux
├── bin
├── include
├── lib
├── lib64
├── libexec
├── riscv64-unknown-linux-gnu
├── share
└── sysroot
```

```
[mydev@fedora /]$ tree /opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/Linux/bin
/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/Linux/bin
├── riscv64-unknown-linux-gnu-addr2line
├── riscv64-unknown-linux-gnu-ar
├── riscv64-unknown-linux-gnu-as
├── riscv64-unknown-linux-gnu-c++
├── riscv64-unknown-linux-gnu-c++filt
├── riscv64-unknown-linux-gnu-cpp
├── riscv64-unknown-linux-gnu-elfedit
├── riscv64-unknown-linux-gnu-g++
├── riscv64-unknown-linux-gnu-gcc
├── riscv64-unknown-linux-gnu-gcc-12.2.0
├── riscv64-unknown-linux-gnu-gcc-ar
├── riscv64-unknown-linux-gnu-gcc-nm
├── riscv64-unknown-linux-gnu-gcc-ranlib
├── riscv64-unknown-linux-gnu-gcov
├── riscv64-unknown-linux-gnu-gcov-dump
├── riscv64-unknown-linux-gnu-gcov-tool
├── riscv64-unknown-linux-gnu-gdb
├── riscv64-unknown-linux-gnu-gdb-add-index
├── riscv64-unknown-linux-gnu-gfortran
├── riscv64-unknown-linux-gnu-gprof
├── riscv64-unknown-linux-gnu-ld
├── riscv64-unknown-linux-gnu-ld.bfd
├── riscv64-unknown-linux-gnu-lto-dump
├── riscv64-unknown-linux-gnu-nm
├── riscv64-unknown-linux-gnu-objcopy
├── riscv64-unknown-linux-gnu-objdump
├── riscv64-unknown-linux-gnu-ranlib
├── riscv64-unknown-linux-gnu-readelf
├── riscv64-unknown-linux-gnu-run
├── riscv64-unknown-linux-gnu-size
├── riscv64-unknown-linux-gnu-strings
└── riscv64-unknown-linux-gnu-strip
```

directly built on RPi4 against “riscv64-unknown-linux-gnu-” for ~9.5 hours (depends on network speed).

Build a special version for IREE

- git checkout rvv-next

```
./configure --prefix=/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/Linux --with-arch=rv64gcv --with-abi=lp64d --enable-multilib
```

Retest when the build is ready…

2.2.2.2 LLVM

- No available binary packages currently

Build from source on RPi4

- https://llvm.org/docs/CMake.html
- https://llvm.org/docs/GettingStarted.html
- Build

1. git clone https://github.com/llvm/llvm-project
   (On main branch, last commit: 910ad36e1a2592915b32941844cf089442972d7a)
   cd $SRC_LLVM;mkdir -p build/Install;cd build
2. our configurations:

```
cmake -G Ninja -DCMAKE_BUILD_TYPE=Release -DLLVM_TARGETS_TO_BUILD="RISCV;BPF;WebAssembly" -DLLVM_ENABLE_PROJECTS="
clang;clang-tools-extra;libclc;libcxx;libcxxabi;lld;lldb;mlir;cross-project-tests" -DLLVM_DEFAULT_TARGET_TRIPLE="r
iscv64-unknown-linux-gnu" -DCMAKE_INSTALL_PREFIX=$PWD/Install ../llvm
```

$SRC_LLVM/llvm/CMakeLists.txt

```
# List of all targets to be built by default:
set(LLVM_ALL_TARGETS
  AArch64
  AMDGPU
  ARM
  AVR
  BPF
  Hexagon
  Lanai
  Mips
  MSP430
  NVPTX
  PowerPC
  RISCV
  Sparc
  SystemZ
  VE
  WebAssembly
  X86
  XCore
  )
```

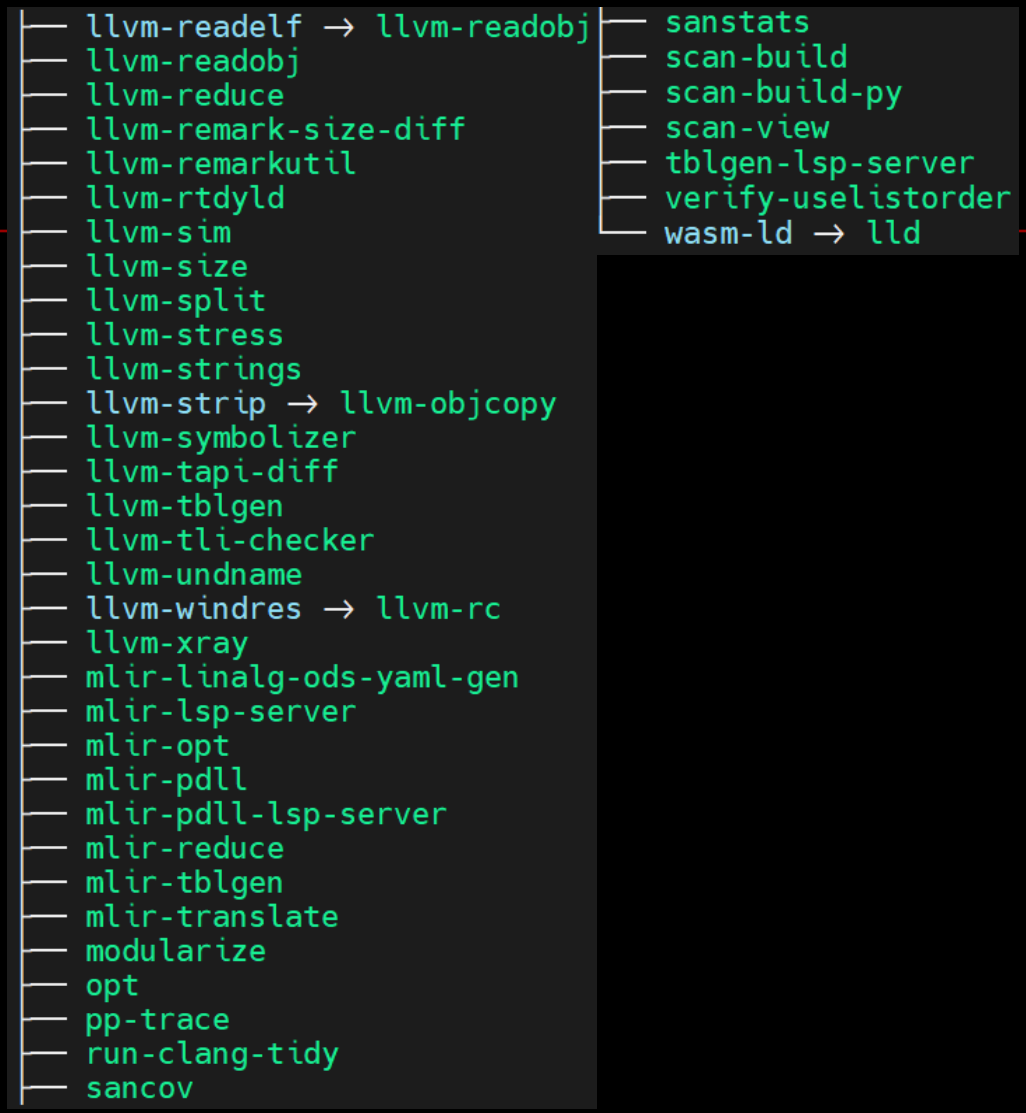
3. build command:

```
ninja -j$(nproc)
```

it costed ~11 hours on my RPi4 for finish the full build

```
[mydev@fedora llvm-project-main]$ tree -L 1 build//Install/
build//Install/
├── bin
├── include
├── lib
├── libexec
├── local
└── share
```

```
[mydev@fedora llvm-project-main]$ tree build//Install/bin
build//Install/bin
├── analyze-build
├── bugpoint
├── c-index-test
├── clang → clang-16
├── clang++ → clang
├── clang-16
├── clang-apply-replacements
├── clang-change-namespace
├── clang-check
├── clang-cl → clang
├── clang-cpp → clang
├── clangd
├── clang-doc
├── clang-extdef-mapping
├── clang-format
├── clang-include-fixer
├── clang-linker-wrapper
├── clang-move
├── clang-offload-bundler
├── clang-offload-packager
├── clang-pseudo
├── clang-query
├── clang-refactor
├── clang-rename
├── clang-reorder-fields
├── clang-repl
├── clang-scan-deps
├── clang-tblgen
├── clang-tidy
├── diagtool
├── dsymutil
├── find-all-symbols
├── git-clang-format
├── hmaptool
├── intercept-build
├── ld64.lld → lld
├── ld.lld → lld
├── llc
├── lld
├── lldb
├── lldb-argdumper
├── lldb-instr
├── lldb-server
├── lldb-vscode
├── lld-link → lld
├── lli
├── llvm-addr2line → llvm-symbolizer
├── llvm-ar
├── llvm-as
├── llvm-bcanalyzer
├── llvm-bitcode-strip → llvm-objcopy
├── llvm-cat
├── llvm-cfi-verify
├── llvm-config
├── llvm-cov
├── llvm-c-test
├── llvm-cvtres
├── llvm-cxxdump
├── llvm-cxxfilt
├── llvm-cxxmap
├── llvm-debuginfod
├── llvm-debuginfod-find
├── llvm-dis
├── llvm-dlltool → llvm-ar
├── llvm-dwarfdump
├── llvm-dwarfutil
├── llvm-dwp
├── llvm-exegesis
├── llvm-extract
├── llvm-gsymutil
├── llvm-ifs
├── llvm-install-name-tool → llvm-objcopy
├── llvm-jitlink
├── llvm-lib → llvm-ar
├── llvm-libtool-darwin
├── llvm-link
├── llvm-lipo
├── llvm-lto
├── llvm-lto2
├── llvm-mc
├── llvm-mca
├── llvm-ml
├── llvm-modextract
├── llvm-mt
├── llvm-nm
├── llvm-objcopy
├── llvm-objdump
├── llvm-opt-report
├── llvm-otool → llvm-objdump
├── llvm-pdbutil
├── llvm-profdata
├── llvm-profgen
├── llvm-ranlib → llvm-ar
└── llvm-rc
```


├── llvm-readelf → llvm-readobj
├── llvm-readobj
├── llvm-reduce
├── llvm-remark-size-diff
├── llvm-remarkutil
├── llvm-rtdyld
├── llvm-sim
├── llvm-size
├── llvm-split
├── llvm-stress
├── llvm-strings
├── llvm-strip → llvm-objcopy
├── llvm-symbolizer
├── llvm-tapi-diff
├── llvm-tblgen
├── llvm-tli-checker
├── llvm-undname
├── llvm-windres → llvm-rc
├── llvm-xray
├── mlir-linalg-ods-yaml-gen
├── mlir-lsp-server
├── mlir-opt
├── mlir-pdll
├── mlir-pdll-lsp-server
├── mlir-reduce
├── mlir-tblgen
├── mlir-translate
├── modularize
├── opt
├── pp-trace
├── run-clang-tidy
├── sancov
├── sanstats
├── scan-build
├── scan-build-py
├── scan-view
├── tblgen-lsp-server
├── verify-uselistorder
└── wasm-ld → lld
...

II. Architecture & Design

Existing ML stack challenges

- ML stacks face huge problem space with combinatorial complexity due to
  - Evolving ML model architectures; various frameworks with shifting user interests
  - Growing heterogeneous hardware (CPUs/GPUs w/ vector/matrix, AI accelerators)
  - Different deployment scenarios (server, desktop/laptop, mobile/edge, web, etc.)
- ML stacks' in-house hardware "interfaces" at ML graph/op level leads to
  - Hardware needs to build full API/runtime/kernel/compiler story to integrate
  - Stack needs to have full story for all hardware and deployment scenarios
- So we see fragmented solution space with
  - Solutions specialize towards a subset and often lack adaptability and generality
  - Extensive duplicated manual engineering efforts within/across various stacks

Source: https://open-src-soc.org/2022-05/media/slides/4th-RISC-V-Meeting-2022-05-03-14h00-Michael-Gielda-and-Adam-Jesionowski.pdf

Towards generalizable and performant ML stack

- We have seen similar challenges in graphics
  - Varying rendering techniques, game engines, GPU vendors, machine form factors
- ML inference stack can draw experiences from decades of learnings in graphics
  - Standards to support various hardware for both commonality and optionality
  - Compilers to handle different architectures for reusability and performance
- Vulkan and SPIR-V presents a modern clean base solution
  - Explicitly exposing hardware functionalities; not opinionated with high-level constructs
  - Low-level; suitable for auto generation (both host scheduling and device executable)
  - Readily available on many platforms; meeting various deployment needs

Source: https://open-src-soc.org/2022-05/media/slides/4th-RISC-V-Meeting-2022-05-03-14h00-Michael-Gielda-and-Adam-Jesionowski.pdf

1) Overview
1.1 What is it

- https://iree-org.github.io/iree/

IREE (**I**ntermediate **R**epresentation **E**xecution **E**nvironment[1]) is an MLIR-based end-to-end compiler and runtime that lowers Machine Learning (ML) models to a unified IR that scales up to meet the needs of the datacenter and down to satisfy the constraints and special considerations of mobile and edge deployments.

Key features

- Ahead-of-time compilation of scheduling and execution logic together
- Support for dynamic shapes, flow control, streaming, and other advanced model features
- Optimized for many CPU and GPU architectures
- Low overhead, pipelined execution for efficient power and resource usage
- Binary size as low as 30KB on embedded systems
- Debugging and profiling support

Support matrix

IREE supports importing from a variety of ML frameworks:

- TensorFlow
- TensorFlow Lite
- JAX
- PyTorch
- ONNX (hoped for)

The IREE compiler tools run on Linux, Windows, and macOS and can generate efficient code for a variety of runtime platforms:

- ✅ Linux
- ✅ Windows
- ✅ Android
- ⬜ macOS (planned)
- ⬜ iOS (planned)
- ✅ Bare metal
- ⬜ WebAssembly (planned)

and architectures:

- ✅ ARM
- ✅ x86
- ✅ RISC-V

Support for hardware accelerators and APIs is also included:

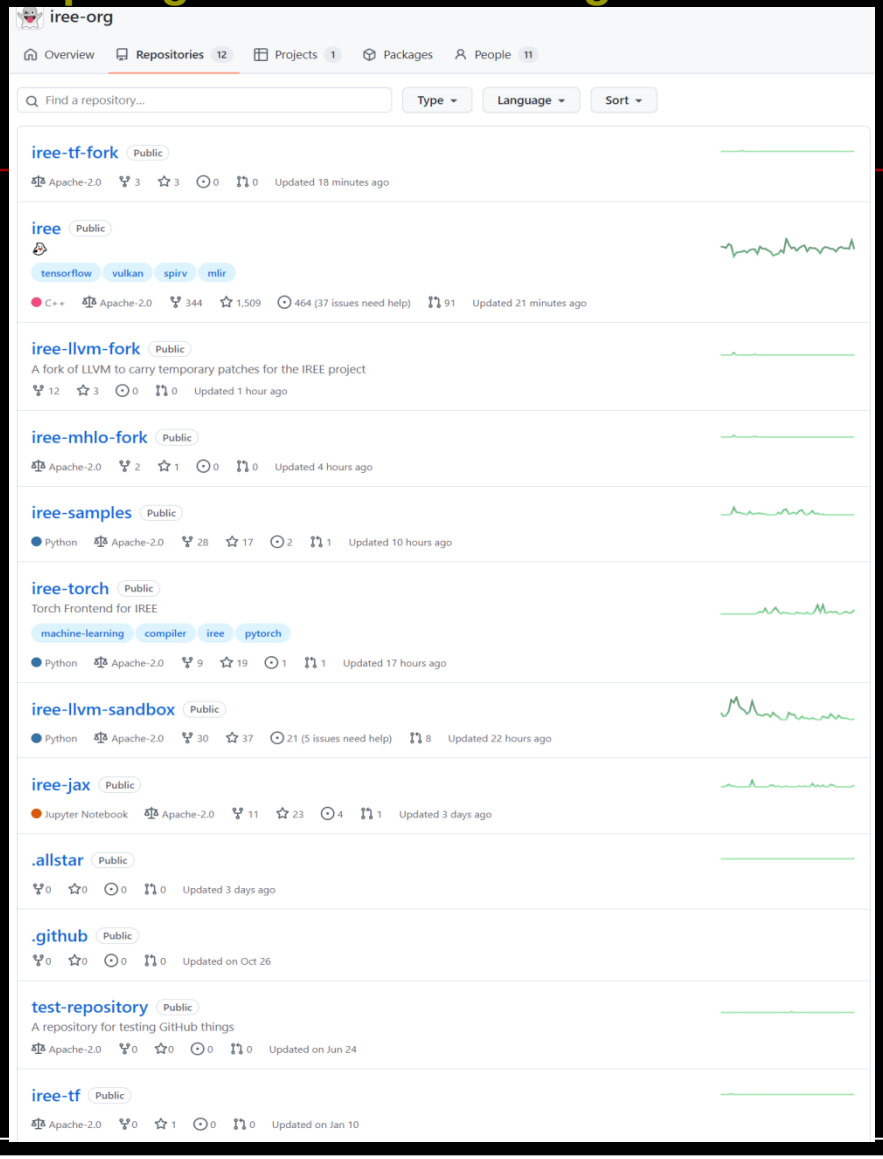
- ✅ Vulkan
- ✅ CUDA
- ⬜ Metal (planned)
- ⬜ WebGPU (planned)

Project status

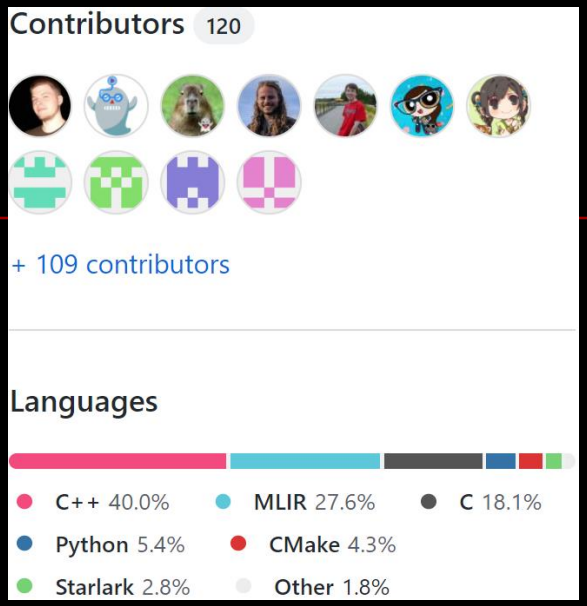
IREE is still in its early phase. We have settled down on the overarching infrastructure and are actively improving various software components as well as project logistics. It is still quite far from ready for everyday use and is made available without any support at the moment. With that said, we welcome any kind of feedback on any communication channels!

Src

- https://github.com/iree-org/

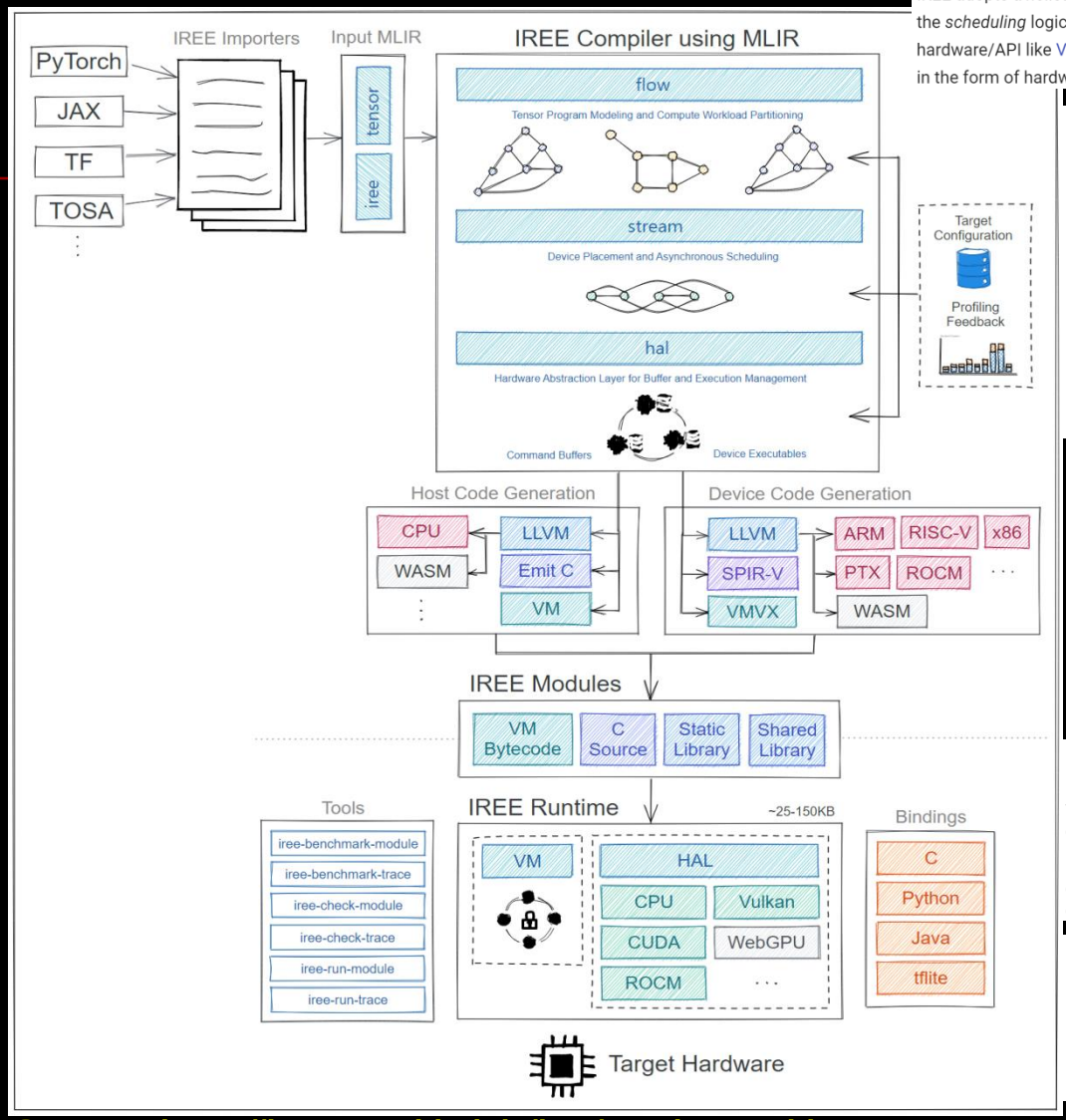


- https://github.com/iree-org/iree




```
[submodule "third_party/googletest"]
        path = third_party/googletest
        url = https://github.com/google/googletest.git
[submodule "third_party/llvm-project"]
        path = third_party/llvm-project
        url = https://github.com/iree-org/iree-llvm-fork.git
[submodule "third_party/vulkan_headers"]
        path = third_party/vulkan_headers
        url = https://github.com/KhronosGroup/Vulkan-Headers.git
[submodule "third_party/vulkan_memory_allocator"]
        path = third_party/vulkan_memory_allocator
        url = https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator.git
[submodule "third_party/spirv_headers"]
        path = third_party/spirv_headers
        url = https://github.com/KhronosGroup/SPIRV-Headers.git
[submodule "third_party/pybind11"]
        path = third_party/pybind11
        url = https://github.com/pybind/pybind11.git
        branch = stable
[submodule "third_party/benchmark"]
        path = third_party/benchmark
        url = https://github.com/google/benchmark.git
[submodule "third_party/tracy"]
        path = third_party/tracy
        url = https://github.com/wolfpld/tracy.git
[submodule "third_party/flatcc"]
        path = third_party/flatcc
        url = https://github.com/dvidelabs/flatcc.git
[submodule "third_party/spirv_cross"]
        path = third_party/spirv_cross
        url = https://github.com/KhronosGroup/SPIRV-Cross.git
[submodule "third_party/cpuinfo"]
        path = third_party/cpuinfo
        url = https://github.com/pytorch/cpuinfo.git
[submodule "third_party/liburing"]
        path = third_party/liburing
        url = https://github.com/axboe/liburing.git
[submodule "third_party/mlir-hlo"]
        path = third_party/mlir-hlo
        url = https://github.com/iree-org/iree-mhlo-fork.git
[submodule "third_party/libyaml"]
        path = third_party/libyaml
        url = https://github.com/yaml/libyaml.git
[submodule "third_party/webgpu-headers"]
        path = third_party/webgpu-headers
        url = https://github.com/webgpu-native/webgpu-headers.git
[submodule "third_party/musl"]
        path = third_party/musl
        url = https://github.com/powderluv/musl.git
```

1.2 Overall Architecture and flow



IREE adopts a *holistic* approach towards ML model compilation: the IR produced contains both the *scheduling* logic, required to communicate data dependencies to low-level parallel pipelined hardware/API like Vulkan, and the *execution* logic, encoding dense computation on the hardware in the form of hardware/API-specific binaries like SPIR-V.

Workflow overview

Specific examples outlining IREE's workflow can be found in the User Getting Started Guide. Using IREE involves the following general steps:

1. **Import your model**

   Develop your program using one of the supported frameworks, then run your model using one of IREE's import tools.
2. **Select your deployment configuration**

   Identify your target platform, accelerator(s), and other constraints.
3. **Compile your model**

   Compile through IREE, picking compilation targets based on your deployment configuration.
4. **Run your model**

   Use IREE's runtime components to execute your compiled model.

Selecting deployment configurations

IREE provides a flexible set of tools for various deployment scenarios. Fully featured environments can use IREE for dynamic model deployments taking advantage of multi-threaded hardware, while embedded systems can bypass IREE's runtime entirely or interface with custom accelerators.

- What platforms are you targeting? Desktop? Mobile? An embedded system?
- What hardware should the bulk of your model run on? CPU? GPU?
- How fixed is your model itself? Can the weights be changed? Do you want to support loading different model architectures dynamically?

IREE supports the full set of these configurations using the same underlying technology.

Compiling models

Model compilation is performed ahead-of-time on a *host* machine for any combination of *targets*. The compilation process converts from layers and operators used by high level frameworks down into optimized native code and associated scheduling logic.

For example, compiling for GPU execution using Vulkan generates SPIR-V kernels and Vulkan API calls. For CPU execution, native code with static or dynamic linkage and the associated function calls are generated.

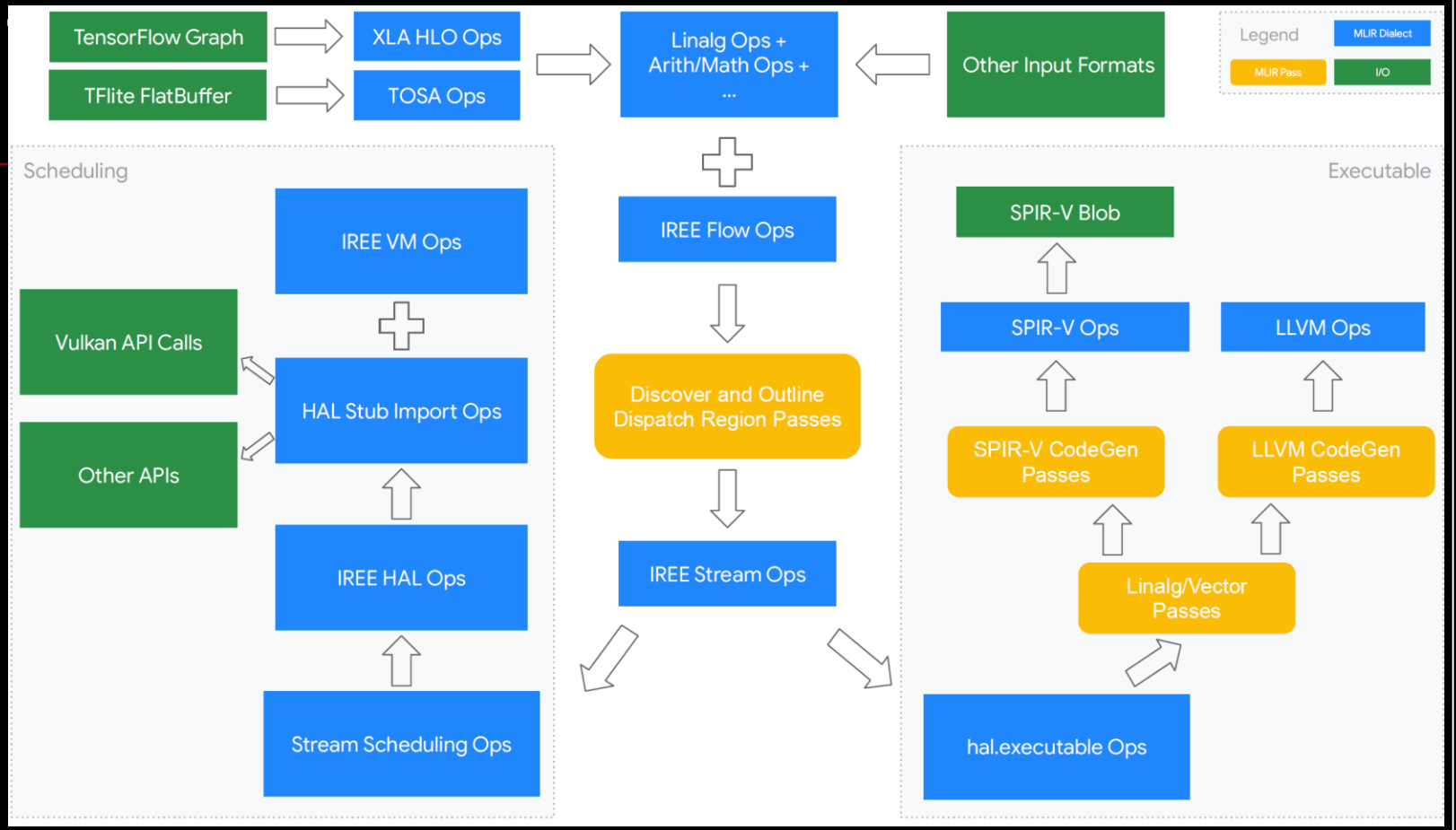
Running models

IREE offers a low level C API, as well as several specialized sets of bindings for running IREE models using other languages:

- C API
- Python
- TensorFlow Lite

Source: https://iree-org.github.io/iree/#project-architecture

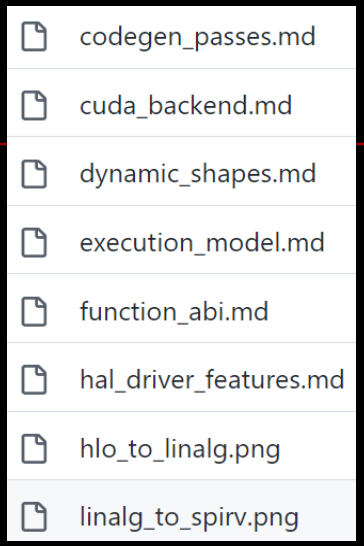
IREE core compilation flow



Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

Good resources

- **https://github.com/iree-org/iree/tree/main/docs/developers/design_docs**



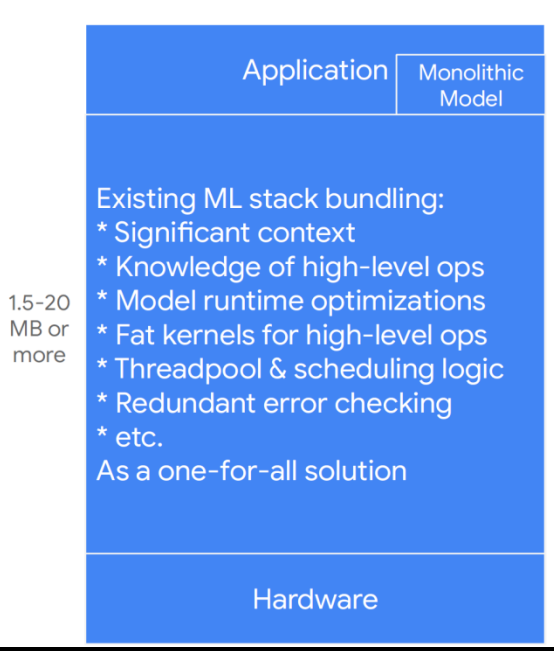
- **https://github.com/iree-org/iree/blob/main/docs/developers/design_roadmap.md**
- **…**

2) Runtime
2.1 Overview

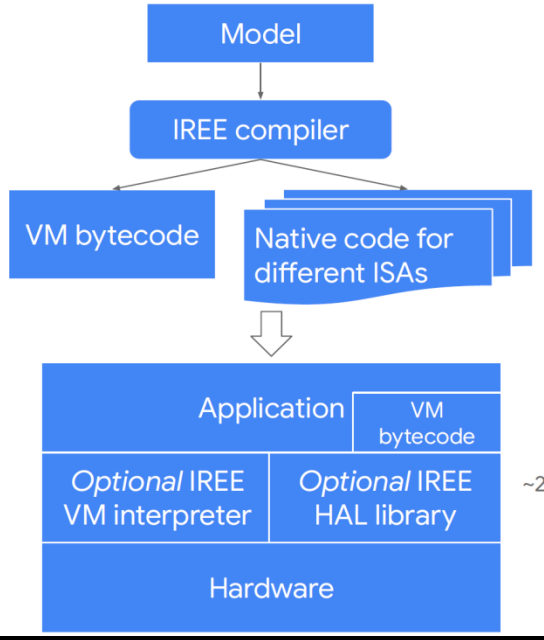
- **IREE runtime**

IREE does not have a traditional "fat" runtime that bundles everything.

IREE provides an almost zero-cost virtual machine for interpreting host scheduling ops compiled from ML models. It just performs lightweight math for workload size calculation and performs task scheduling.



1.5-20 MB or more



~25-150 KB

Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

2.2 VM
2.2.1 Overview

VM: Lightweight User-mode Processes and an FFI

- Responsible for “host” programs
- Workload is mostly calls into the HAL and some simple buffer offset arithmetic
- Isolates loaded module instances and their state (ala processes)
- Dynamic runtime linking of modules implemented in C or compiler-generated
  - Used for versioning/compatibility shimming
  - Enables host code sharing (similar model architectures/etc)
- Reference-counted type-safe handles: no void*, no need for an MMU
- Bytecode module format runs anywhere
  - Minimal implementation is i32-only, ~10KB
  - Extensions for i64, f32, and f64
- Coroutines: cellular batching, wider parallelism, reduced thrashing

Source: “IREE Runtime Design”, benvanik@google.com, 2021.

2.2.1.1 ISA

- Virtual register-based instructions
- Direct execution (no load-time processing)
- Efficiently dispatchable with computed gotos
- Support for external ref counted types
  - VM tracks references to avoid use-after-free
  - Opaque type round-tripping
  - Built-in types for lists and byte buffers
- Memory access only though checked buffers
- Instruction set extension mechanism
  - Optional 64-bit integer type
  - Optional 32/64-bit floating-point types
- Stack frames and stack walking support
- MLIR is easily lowered to C, LLVM-IR, etc

```
vm.export @loop_sum
vm.func @loop_sum(%count : i32) -> i32 {
  %c1 = vm.const.i32 1 : i32
  %i0 = vm.const.i32.zero : i32
  vm.br ^loop(%i0 : i32)
^loop(%i : i32):
  %in = vm.add.i32 %i, %c1 : i32
  %cmp = vm.cmp.lt.i32.s %in, %count : i32
  vm.cond_br %cmp, ^loop(%in : i32),
                   ^loop_exit(%in : i32)
^loop_exit(%ie : i32):
  vm.return %ie : i32
}
```

Source: "IREE Runtime Design", benvanik@google.com, 2021.

2.2.2 Module
2.2.2.1 Overview

- Modules are like an ELF shared library/DLL. The IREE runtime provides a dynamic linker.

  VM modules are just a small C interface: a module can be defined as a FlatBuffer with bytecode, native C/C++ (handwritten or generated with emitc), or anything you can reach from the C ABI (python, javascript, etc).

  Built-in IREE functionality is also done via modules, like the HAL, allowing multi-versioning and compatibility shims in the same way one would do it for a normal application.

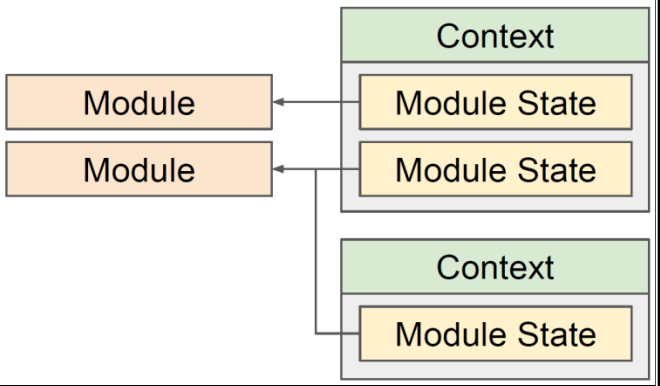
  The most general and well-supported scenario has the compiler produce FlatBuffers containing the IREE VM bytecode (.vmfb). Provides a dynamically deployable mmap-able binary with near zero load-time overhead.

Source: "IREE Runtime Design", benvanik@google.com, 2021.

$SRC_IREE/runtime/src/iree/vm/module.h
$SRC_IREE/runtime/src/iree/vm/bytecode_module.h
$SRC_IREE/runtime/src/iree/vm/native_module.h

VM module:context :: shared object:process

- **Contexts** perform dynamic linking and instantiate **module state**
- **Modules** import and export **functions**
  - Virtual interface with a defined FFI, part of the public API
  - Single unified interface for built-in, compiler-generated, and user-defined modules
  - Reflection queries for plumbing higher-level calling convention info
  - Type system and import signature verification
  - Supports coroutine-style continuation
- **Abstract call stack**
  - Cross-environment backtraces
  - Reference-counted object lifetime tracking
  - Storage for coroutine state



Source: "IREE Runtime Design", benvanik@google.com, 2021.

Native Module

VM: Low-level Native Modules

Low-level C interface; good for small code and interop (exporting to python/etc):

```
// vm.import @semaphore.create(
//   %device : !vm.ref<!hal.device>,
//   %initial_value : i32
// ) -> !vm.ref<!hal.semaphore>

IREE_VM_ABI_EXPORT(iree_hal_module_semaphore_create,
                   iree_hal_module_state_t,
                   ri, r) {
  iree_hal_device_t* device = NULL;
  IREE_RETURN_IF_ERROR(iree_hal_device_check_deref(args->r0, &device));
  uint32_t initial_value = (uint32_t)args->i1;
  ..
  rets->r0 = iree_hal_semaphore_move_ref(semaphore);
  return iree_ok_status();
}
```

VM: High-level Native Modules

High-level C++ interface; much easier to use with larger binary size:

```
// vm.import @semaphore.create(
//   %device : !vm.ref<!hal.device>,
//   %initial_value : i32
// ) -> !vm.ref<!hal.semaphore>

StatusOr<vm::ref<iree_hal_semaphore_t>> SemaphoreCreate(
    vm::ref<iree_hal_device_t> device, int32_t initial_value) {
  ...
  return semaphore;
}
```

Source: "IREE Runtime Design", benvanik@google.com, 2021.

Bytecode Module

The most general and well-supported scenario uses FlatBuffers to provide a mmap-able binary with near zero load-time overhead containing the IREE VM bytecode (.vmfb).

Just one implementation of the module interface; WebAssembly, Lua, Python, etc can be implemented with equal fidelity.

Goals:

- Portability (runs anywhere C can run)
- Manageable implementation (~15KB)
- Near 1:1 mapping with MLIR dialect
- Rich type and FFI verification
- Safe isolation even without OS and hardware support

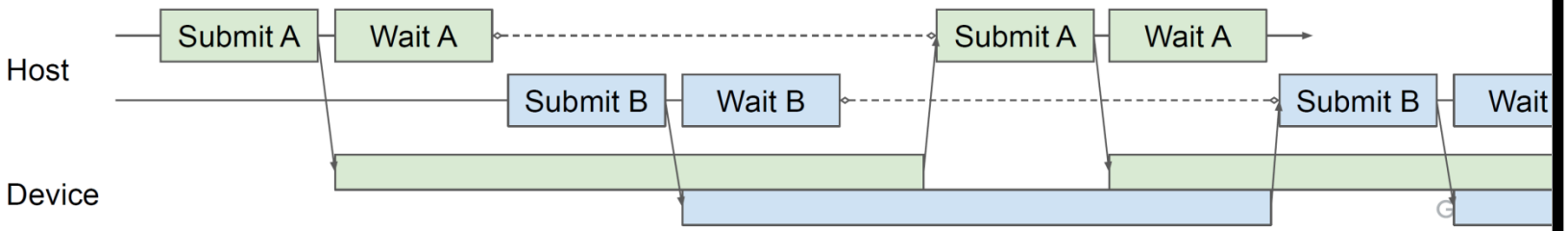
BytecodeModuleDef
Header for runtime versioning
Import functions from other modules
Export functions to other modules
Reflection metadata for FFI
Read-only data (.rodata) for constants and compiled HAL device executables
Initialized **read-write data** (.data) for variables
VM bytecode for the host program (.text)

Source: "IREE Runtime Design", benvanik@google.com, 2021.

2.2.3 Concurrency
2.2.3.1 Coroutine

ABI has built-in coroutine support; works across languages/runtimes.

- Enables multiple programs - or invocations of the same - to overlap
- VM execution yields on wait handles; multi-wait for epoll-style waiting
- Compiler determines what work may overlap based on data flow
- Enables dispatch coalescing in the HAL



Source: "IREE Runtime Design", benvanik@google.com, 2021.

1.3.2 HAL

HAL: A Compute-only Subset of Vulkan + Some Tweaks

Modern GPU APIs like Vulkan, Metal, Direct3D 12, and (lower-level) CUDA all effectively do the same thing with different spelling:

- **Enumerate and query devices** and their capabilities
- **Define executable code** that runs on the device
- **Allocate** unified or discrete memory and provide cache control
- **Organize work into sequences** for deferred submission
- Provide **explicit synchronization primitives** for ordering submissions

The IREE HAL wraps these up in Vulkan-themed functions (as it's the only open and cross-platform/vendor solution) - but it's just style.

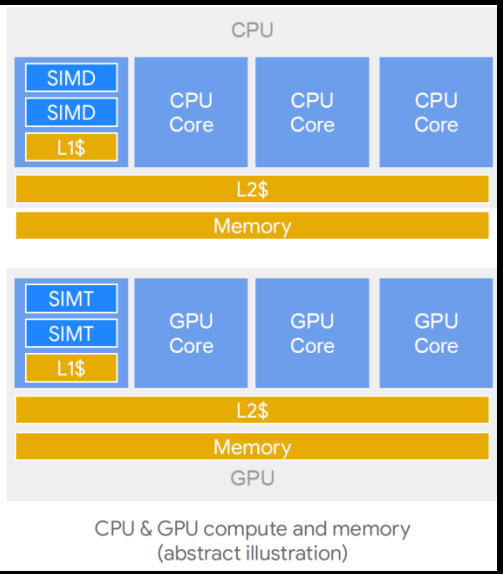
- Common abstraction for CPU, GPU, and beyond
  - All with multi-level memory/compute hierarchies
  - All meant for compute in tiled fashion
- Building pipelines to exploit the scheduling hierarchy
  - Submissions (workload + coarse-grained sync)
    - Command buffers (workload + fine-grained sync)

On host

    - Dispatches (⟶ GPU; ⟶ CPU)
  - Workgroups (⟶ GPU cores; ⟶ CPU threads)
    - Subgroups (⟶ GPU SIMT; ⟶ CPU: SIMD)

In executable
(On hardware; target CodeGen)

      - Instr. (⟶ GPU thread; ⟶ CPU: lane)



CPU & GPU compute and memory (abstract illustration)

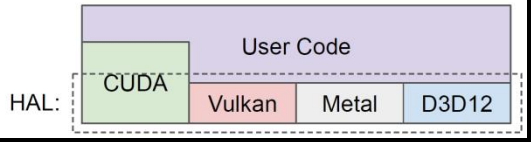
Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

HAL: A Low-Level Abstraction

Predictable, portable, efficient, and small.

- No implicit state tracking: users own only the state they need
- No implicit cross-device memory transfer: DMA is a first-class concept
- No unmanaged void* access: explicit buffer and binding APIs
- No implicit synchronization: barriers are opt-in

For larger full-stack solutions like CUDA only the lower-level primitives are used.



HAL: Drivers and Devices

- Drivers provide device enumeration
  - Driver examples: CUDA, Vulkan, local CPU
  - Available drivers change over time: external nvidia GPU plugged in, remote host available
- Devices model logical resource/execution scopes
  - Device examples: GPU 0, GPU 1, GPU 0+1 (mirrored), CPU package 0
  - Available devices change over time: discrete GPU powered on, pooled device unused
- Devices are primarily factories for resources like buffers and executables
- Capability query interface to allow compiled modules to select/adapt
- Hosting applications can wrap existing device handles (thread pools, VkDevice, CUdevice, etc) for interoperability
- Note: a CPU is a device! (more later)

Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

HAL: Allocators

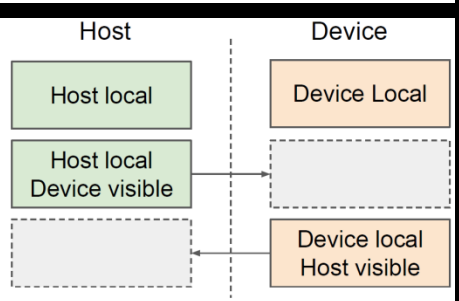
All buffer handles obtained via a device-provided allocator handle:

- Explicit host/device placement and visibility control
- Wrap host buffers (void*) if able
- Import/export for interoperability (import VkBuffer, etc)
- Buffer constraint queries (min alignment, max size, etc)

Implementation can be simple (malloc/free), perform pooling (VMA for Vulkan), or defer to external allocators reusing application memory or memory shared across devices.

Compiler emits VM code to perform tensor-level packing and allocation.

HAL: Buffers



Memory type:

- `HOST|DEVICE_LOCAL`: where the buffer lives
- `HOST|DEVICE_VISIBLE`: who can access it
- `TRANSIENT`: queue-local pooled memory

Allowed Usage: `CONSTANT|TRANSFER|MAPPING|DISPATCH`

Allowed Access: `READ|WRITE`

Mapping: explicit checked map/unmap to get host pointer access

- Random void* access is a non-optimal convenience for *humans*
- Invalidation and flushing critical for non-coherent memory types

Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

HAL: Command Buffers

Reusable recordings of a sequence of commands to execute. Like a CUstream that can be replayed.

Execution order and concurrency defined by synchronization commands.

Primary/secondary nesting: primary is dynamic, secondary is static and reused.

Commands:

- Synchronization: `ExecutionBarrier, SetEvent/WaitEvent`
- DMA: `CopyBuffer, FillBuffer, UpdateBuffer`
- Execution: `Dispatch, DispatchIndirect, Execute` (command buffer)
- Recording-only state: `PushConstants, BindDescriptorSet`

HAL: Executables

Executables provide one or more entry points that have well-defined signatures:

- Loaded via an optionally-stateful device-defined cache handle
  - Enables asynchronous/batched preparation (JITing/translation)
  - Enables persistent caching (VkPipelineCache) to reduce cold-start time
- Explicit I/O layout enables validation during calls
  - Only buffers specified in the layout may be accessed during execution
  - Compiler determines the layout and can trade off performance/size/tracking overhead
- Implementation can multi-version/specialize (CPU uarch, GPU shared memory size/extension availability)

Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

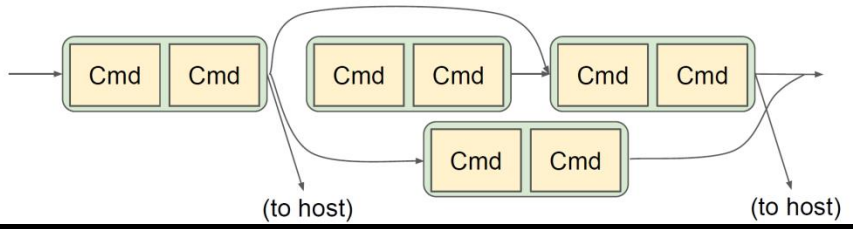
HAL: Semaphores (aka Fences)

Each semaphore is a monotonically increasing timeline, submissions can specify:

- [Wait for a semaphore to reach >=t]
- [Signal a semaphore to t=t']

Defines a DAG. Both host and device can signal and wait. No round-trips!

Lazy updating: fine for latency between signal and availability.



Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

1.3.2.1 HAL IR

- …

- **Example**

HAL IR example

HAL has scheduling ops that map to new generation explicit GPU APIs like Vulkan.

These ops effectively expose Vulkan C APIs as compiler IRs for automatic transformation, to enable codify best practices via compilers.

This is where we materialize concrete (binding-based) ABIs between executables and scheduling.

```
module {
 hal.executable @executable_module {
  hal.interface @abi {
   hal.interface.binding @ret, set=0, binding=0, type="StorageBuffer", access="Read"
   ...
  }
  hal.executable.binary {data = dense<[...]> : vector<1620xi8>, format = "SPIR-V"} ...
```

Executable: SPIR-V

```
func @main(%arg0: !iree.ref<!hal.buffer>, %arg1: !iree.ref<!hal.buffer>) -> !iree.ref<!hal.buffer> {
 %dev = hal.ex.shared_device : !iree.ref<!hal.device>
 %allocator = hal.device.allocator %dev : !iree.ref<!hal.allocator>
 %buffer = hal.allocator.allocate %allocator, ..., shape=[...], element_size=4 : !iree.ref<!hal.buffer>
 %cmd = hal.command_buffer.create %dev, "OneShot", "Transfer|Dispatch" : !iree.ref<!hal.command_buffer>
 hal.command_buffer.begin %cmd
 hal.ex.push_descriptor_set %cmd, ...
 hal.device.switch(%dev: !hal.device)
 #hal.device.match.id<"vulkan*"> : !iree.ref<!hal.buffer> {
   ...
   %exe = hal.executable.look_up, ...
   hal.command_buffer.dispatch %cmd, %exe, entry_point=0, workgroup_xyz=[%c1, %c5, %c1]
 }
 %memory_barrier = hal.make_memory_barrier "DispatchWrite", "DispatchRead" : tuple<i32, i32>
 hal.command_buffer.execution_barrier %cmd, "CommandRetire", "CommandIssue",
                                      memory_barriers=[%memory_barrier]
 ...
 hal.command_buffer.end %cmd
 hal.ex.submit_and_wait %dev, %cmd
 return %buffer : !iree.ref<!hal.buffer>
```

Scheduling

Source: https://www.khronos.org/assets/uploads/developers/presentations/IREE_targeting_Vulkan_Zhang_May22.pdf

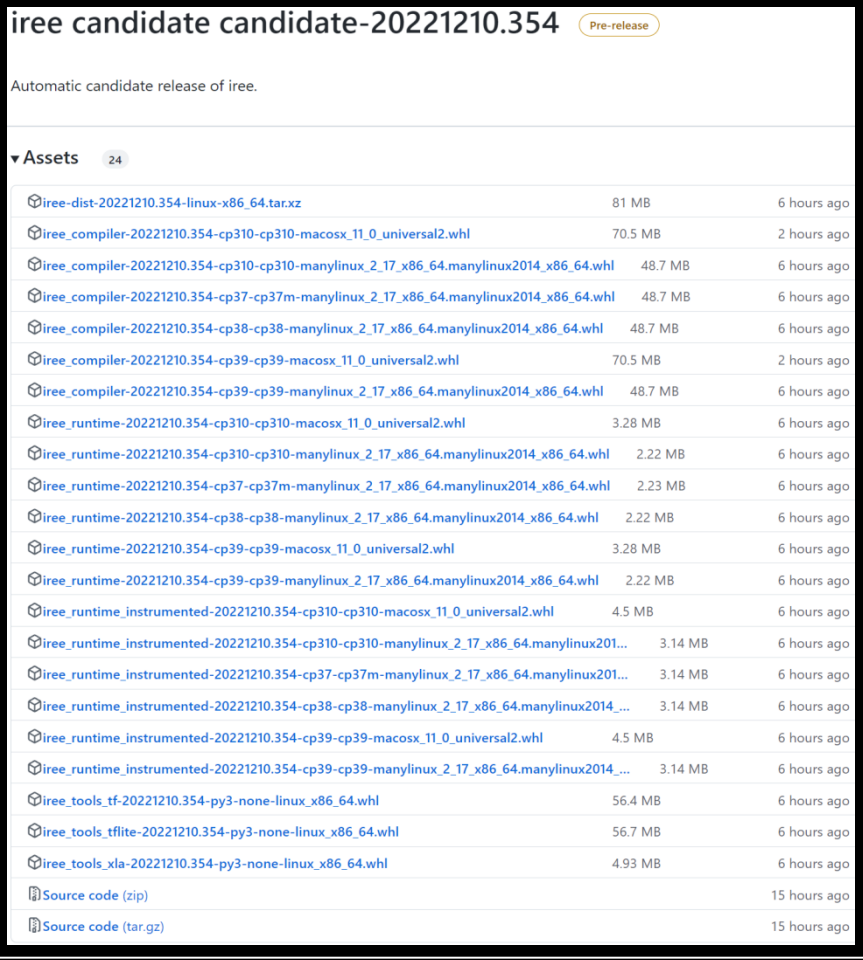
III. Practice on ARM

Prerequisites

- …

1) RPi4
1.1 Setup IREE
1.1.1 Package install (Only available for X64)

- https://github.com/iree-org/iree/releases



iree candidate candidate-20221210.354 Pre-release

Automatic candidate release of iree.

▾ Assets 24

iree-dist-20221210.354-linux-x86_64.tar.xz	81 MB	6 hours ago
iree_compiler-20221210.354-cp310-cp310-macosx_11_0_universal2.whl	70.5 MB	2 hours ago
iree_compiler-20221210.354-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	48.7 MB	6 hours ago
iree_compiler-20221210.354-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	48.7 MB	6 hours ago
iree_compiler-20221210.354-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	48.7 MB	6 hours ago
iree_compiler-20221210.354-cp39-cp39-macosx_11_0_universal2.whl	70.5 MB	2 hours ago
iree_compiler-20221210.354-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	48.7 MB	6 hours ago
iree_runtime-20221210.354-cp310-cp310-macosx_11_0_universal2.whl	3.28 MB	6 hours ago
iree_runtime-20221210.354-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	2.22 MB	6 hours ago
iree_runtime-20221210.354-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	2.23 MB	6 hours ago
iree_runtime-20221210.354-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	2.22 MB	6 hours ago
iree_runtime-20221210.354-cp39-cp39-macosx_11_0_universal2.whl	3.28 MB	6 hours ago
iree_runtime-20221210.354-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl	2.22 MB	6 hours ago
iree_runtime_instrumented-20221210.354-cp310-cp310-macosx_11_0_universal2.whl	4.5 MB	6 hours ago
iree_runtime_instrumented-20221210.354-cp310-cp310-manylinux_2_17_x86_64.manylinux201...	3.14 MB	6 hours ago
iree_runtime_instrumented-20221210.354-cp37-cp37m-manylinux_2_17_x86_64.manylinux201...	3.14 MB	6 hours ago
iree_runtime_instrumented-20221210.354-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_...	3.14 MB	6 hours ago
iree_runtime_instrumented-20221210.354-cp39-cp39-macosx_11_0_universal2.whl	4.5 MB	6 hours ago
iree_runtime_instrumented-20221210.354-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_...	3.14 MB	6 hours ago
iree_tools_tf-20221210.354-py3-none-linux_x86_64.whl	56.4 MB	6 hours ago
iree_tools_tflite-20221210.354-py3-none-linux_x86_64.whl	56.7 MB	6 hours ago
iree_tools_xla-20221210.354-py3-none-linux_x86_64.whl	4.93 MB	6 hours ago
Source code (zip)		15 hours ago
Source code (tar.gz)		15 hours ago

1.1.1.1 Prebuilt toolchains from Google for RISC-V

For IREE

- https://github.com/iree-org/iree/blob/main/build_tools/riscv/riscv_bootstrap.sh
  1. https://storage.googleapis.com/iree-shared-files/toolchain_iree_20220918.tar.gz

```
[mydev@fedora Google]$ tree -L 1 toolchain_iree_20220918
toolchain_iree_20220918
├── bin
├── include
├── lib
├── libexec
├── riscv64-unknown-linux-gnu
├── share
└── sysroot
```

```
[mydev@fedora Google]$ tree -L 2 toolchain_iree_20220918/lib
toolchain_iree_20220918/lib
├── bfd-plugins
│   └── libdep.so
├── clang
│   └── 16.0.0
├── cmake
│   └── llvm
├── gcc
│   └── riscv64-unknown-linux-gnu
├── libcc1.la
├── libcc1.so → libcc1.so.0.0.0
├── libcc1.so.0 → libcc1.so.0.0.0
├── libcc1.so.0.0.0
├── libclang.so → libclang.so.16
├── libclang.so.16 → libclang.so.16.0.0git
├── libclang.so.16.0.0git
├── libear
│   ├── config.h.in
│   ├── ear.c
│   └── __init__.py
├── libLTO.so → libLTO.so.16git
├── libLTO.so.16git
├── libRemarks.so → libRemarks.so.16git
├── libRemarks.so.16git
├── libriscv64-unknown-linux-gnu-sim.a
└── libscanbuild
    ├── analyze.py
    ├── arguments.py
    ├── clang.py
    ├── compilation.py
    ├── __init__.py
    ├── intercept.py
    ├── report.py
    ├── resources
    └── shell.py
```

```
[mydev@fedora Google]$ file qemu-riscv/qemu-riscv64
qemu-riscv/qemu-riscv64: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, with debug_info, not stripped
[mydev@fedora Google]$
[mydev@fedora Google]$ file toolchain_iree_20220918/bin/riscv64-unknown-linux-gnu-ld
toolchain_iree_20220918/bin/riscv64-unknown-linux-gnu-ld: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, with debug_info, not stripped
[mydev@fedora Google]$
[mydev@fedora Google]$ file toolchain_iree_20220918/bin/clang
toolchain_iree_20220918/bin/clang: symbolic link to clang-16
[mydev@fedora Google]$ file toolchain_iree_20220918/bin/clang-16
toolchain_iree_20220918/bin/clang-16: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, not stripped
[mydev@fedora Google]$
```

```
[mydev@fedora Google]$ tree toolchain_iree_20220918/bin
toolchain_iree_20220918/bin
├── analyze-build
├── clang → clang-16
├── clang++ → clang
├── clang-16
├── clang-check
├── clang-cl → clang
├── clang-cpp → clang
├── clang-extdef-mapping
├── clang-format
├── clang-linker-wrapper
├── clang-offload-bundler
├── clang-offload-packager
├── clang-refactor
├── clang-rename
├── clang-repl
├── clang-scan-deps
├── diagtool
├── git-clang-format
├── hmaptool
├── intercept-build
├── ld64.lld → lld
├── ld.lld → lld
├── lld
├── lld-link → lld
├── llvm-ar
├── llvm-cov
├── llvm-cxxfilt
├── llvm-dwp
├── llvm-lib → llvm-ar
├── llvm-ml
├── llvm-nm
├── llvm-objcopy
├── llvm-objdump
├── llvm-pdbutil
├── llvm-profdata
├── llvm-ranlib → llvm-ar
├── llvm-rc
├── llvm-readobj
├── llvm-size
├── llvm-strings
├── llvm-strip → llvm-objcopy
├── llvm-symbolizer
├── riscv64-unknown-linux-gnu-addr2line
├── riscv64-unknown-linux-gnu-ar
├── riscv64-unknown-linux-gnu-as
├── riscv64-unknown-linux-gnu-c++
├── riscv64-unknown-linux-gnu-c++filt
├── riscv64-unknown-linux-gnu-cpp
├── riscv64-unknown-linux-gnu-elfedit
├── riscv64-unknown-linux-gnu-g++
├── riscv64-unknown-linux-gnu-gcc
├── riscv64-unknown-linux-gnu-gcc-10.2.0
├── riscv64-unknown-linux-gnu-gcc-ar
├── riscv64-unknown-linux-gnu-gcc-nm
├── riscv64-unknown-linux-gnu-gcc-ranlib
├── riscv64-unknown-linux-gnu-gcov
├── riscv64-unknown-linux-gnu-gcov-dump
├── riscv64-unknown-linux-gnu-gcov-tool
├── riscv64-unknown-linux-gnu-gdb
├── riscv64-unknown-linux-gnu-gdb-add-index
├── riscv64-unknown-linux-gnu-gfortran
├── riscv64-unknown-linux-gnu-gprof
├── riscv64-unknown-linux-gnu-ld
├── riscv64-unknown-linux-gnu-ld.bfd
├── riscv64-unknown-linux-gnu-lto-dump
├── riscv64-unknown-linux-gnu-nm
├── riscv64-unknown-linux-gnu-objcopy
├── riscv64-unknown-linux-gnu-objdump
├── riscv64-unknown-linux-gnu-ranlib
├── riscv64-unknown-linux-gnu-readelf
├── riscv64-unknown-linux-gnu-run
├── riscv64-unknown-linux-gnu-size
├── riscv64-unknown-linux-gnu-strings
├── riscv64-unknown-linux-gnu-strip
├── scan-build
├── scan-build-py
├── scan-view
└── wasm-ld → lld
```

```
[mydev@fedora Google]$ tree toolchain_iree_20220918/libexec
toolchain_iree_20220918/libexec
├── analyze-c++
├── analyze-cc
├── c++-analyzer
├── ccc-analyzer
├── gcc
│   └── riscv64-unknown-linux-gnu
│       └── 10.2.0
│           ├── cc1
│           ├── cc1plus
│           ├── collect2
│           ├── f951
│           ├── install-tools
│           │   ├── fixincl
│           │   ├── fixinc.sh
│           │   ├── mkheaders
│           │   └── mkinstalldirs
│           ├── liblto_plugin.la
│           ├── liblto_plugin.so → liblto_plugin.so.0.0.0
│           ├── liblto_plugin.so.0 → liblto_plugin.so.0.0.0
│           ├── liblto_plugin.so.0.0.0
│           ├── lto1
│           ├── lto-wrapper
│           └── plugin
│               └── gengtype
├── intercept-c++
└── intercept-cc
```

```
[mydev@fedora Google]$ tree -L 1 toolchain_iree_20220918/sysroot
toolchain_iree_20220918/sysroot
├── etc
├── lib
├── lib32
├── lib64
├── sbin
├── usr
└── var
```

...

2. https://storage.googleapis.com/iree-shared-files/qemu-riscv.tar.gz

```
[mydev@fedora Google]$ tree qemu-riscv
qemu-riscv
├── qemu-bridge-helper
├── qemu-edid
├── qemu-img
├── qemu-io
├── qemu-keymap
├── qemu-nbd
├── qemu-pr-helper
├── qemu-riscv32
├── qemu-riscv64
└── qemu-system-riscv32

0 directories, 10 files
[mydev@fedora Google]$
[mydev@fedora Google]$ file qemu-riscv/qemu-riscv64
qemu-riscv/qemu-riscv64: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, with deb
ug_info, not stripped
[mydev@fedora Google]$
```

For Springbok

- https://github.com/AmbiML/iree-rv32-springbok/blob/main/build_tools/install_toolchain.sh

1. https://storage.googleapis.com/shodan-public-artifacts/toolchain_iree_rv32.tar.gz

```
[mydev@fedora Google]$ tree -L 1 toolchain_iree_rv32imf
toolchain_iree_rv32imf
├── bin
├── include
├── lib
├── libexec
├── riscv32-unknown-elf
└── share
```

```
[mydev@fedora Google]$ file toolchain_iree_rv32imf/bin/clang-16
toolchain_iree_rv32imf/bin/clang-16: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=628
ef6b713435cae68aba5e601a645ec0abee509, for GNU/Linux 3.2.0, not stripped
[mydev@fedora Google]$
[mydev@fedora Google]$ file toolchain_iree_rv32imf/bin/riscv32-unknown-elf-ld
toolchain_iree_rv32imf/bin/riscv32-unknown-elf-ld: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, Bui
ldID[sha1]=76d6ab36725954c0d6861035c9ad4772ba732140, for GNU/Linux 3.2.0, with debug_info, not stripped
[mydev@fedora Google]$
```

...

```
[mydev@fedora Google]$ tree toolchain_iree_rv32imf/bin
toolchain_iree_rv32imf/bin
├── analyze-build
├── clang → clang-16
├── clang++ → clang
├── clang-16
├── clang-check
├── clang-cl → clang
├── clang-cpp → clang
├── clang-extdef-mapping
├── clang-format
├── clang-linker-wrapper
├── clang-offload-bundler
├── clang-offload-packager
├── clang-refactor
├── clang-rename
├── clang-repl
├── clang-scan-deps
├── diagtool
├── git-clang-format
├── hmaptool
├── intercept-build
├── ld64.lld → lld
├── ld.lld → lld
├── lld
├── lld-link → lld
├── llvm-ar
├── llvm-cov
├── llvm-cxxfilt
├── llvm-dwp
├── llvm-lib → llvm-ar
├── llvm-ml
├── llvm-nm
├── llvm-objcopy
├── llvm-objdump
├── llvm-pdbutil
├── llvm-profdata
├── llvm-ranlib → llvm-ar
├── llvm-rc
├── llvm-readobj
├── llvm-size
├── llvm-strings
├── llvm-strip → llvm-objcopy
├── llvm-symbolizer
├── riscv32-unknown-elf-addr2line
├── riscv32-unknown-elf-ar
├── riscv32-unknown-elf-as
├── riscv32-unknown-elf-c++
├── riscv32-unknown-elf-c++filt
├── riscv32-unknown-elf-cpp
├── riscv32-unknown-elf-elfedit
├── riscv32-unknown-elf-g++
├── riscv32-unknown-elf-gcc
├── riscv32-unknown-elf-gcc-10.2.0
├── riscv32-unknown-elf-gcc-ar
├── riscv32-unknown-elf-gcc-nm
├── riscv32-unknown-elf-gcc-ranlib
├── riscv32-unknown-elf-gcov
├── riscv32-unknown-elf-gcov-dump
├── riscv32-unknown-elf-gcov-tool
├── riscv32-unknown-elf-gdb
├── riscv32-unknown-elf-gdb-add-index
├── riscv32-unknown-elf-gprof
├── riscv32-unknown-elf-ld
├── riscv32-unknown-elf-ld.bfd
├── riscv32-unknown-elf-lto-dump
├── riscv32-unknown-elf-nm
├── riscv32-unknown-elf-objcopy
├── riscv32-unknown-elf-objdump
├── riscv32-unknown-elf-ranlib
├── riscv32-unknown-elf-readelf
├── riscv32-unknown-elf-run
├── riscv32-unknown-elf-size
├── riscv32-unknown-elf-strings
├── riscv32-unknown-elf-strip
├── scan-build
├── scan-build-py
├── scan-view
└── wasm-ld → lld
```

1.1.2 Build from Src on RPi4

1.1.2.1 For host development

Official guide

- https://iree-org.github.io/iree/building-from-source/getting-started

Configure CMake:

Linux and MacOS | Windows

```
# Recommended for simple development using clang and lld:
cmake -GNinja -B ../iree-build/ -S . \
    -DCMAKE_BUILD_TYPE=RelWithDebInfo \
    -DIREE_ENABLE_ASSERTIONS=ON \
    -DCMAKE_C_COMPILER=clang \
    -DCMAKE_CXX_COMPILER=clang++ \
    -DIREE_ENABLE_LLD=ON

# Alternately, with system compiler and your choice of CMake generator:
# cmake -B ../iree-build/ -S .

# Additional quality of life CMake flags:
# Enable ccache:
# See https://github.com/iree-org/iree/blob/main/docs/developers/developing_iree/ccache.md
#   -DCMAKE_C_COMPILER_LAUNCHER=ccache
#   -DCMAKE_CXX_COMPILER_LAUNCHER=ccache
```

Build:

```
cmake --build ../iree-build/
```

What's next?

Running tests

Build test dependencies:

```
cmake --build ../iree-build --target iree-test-deps
```

Run all built tests through CTest:

```
ctest --test-dir ../iree-build/ --output-on-failure
```

Take a look around

Check out the contents of the 'tools' build directory:

```
ls ../iree-build/tools/
../iree-build/tools/iree-compile --help
```

1.1.2.2 For RISC-V development

Official guide

- https://iree-org.github.io/iree/building-from-source/riscv/

Running on a platform like RISC-V involves cross-compiling from a *host* platform (e.g. Linux) to a *target* platform (a specific RISC-V CPU architecture and operating system):

- IREE's *compiler* is built on the host and is used there to generate modules for the target
- IREE's *runtime* is built on the host for the target. The runtime is then pushed to the target to run natively.

1. Host environment setup

Prerequisites

Host environment setup

You should already be able to build IREE from source on your host platform. Please make sure you have followed the getting started steps.

Install RISC-V cross-compile toolchain and emulator

You'll need a RISC-V LLVM compilation toolchain and a RISC-V enabled QEMU emulator.

See instructions in the following links

- Clang getting started
- RISC-V GNU toolchain
- QEMU
- RISC-V Linux QEMU

Note

The `RISCV_TOOLCHAIN_ROOT` environment variable needs to be set to the root directory of the installed GNU toolchain when building the RISC-V compiler target and the runtime library.

Install prebuilt RISC-V tools (RISC-V 64-bit Linux toolchain)

Execute the following script to download the prebuilt RISC-V toolchain and QEMU from the IREE root directory:

```
./build_tools/riscv/riscv_bootstrap.sh
```

Support vector extension

For RISC-V vector extensions support, see additional instructions

Configure and build

Host configuration

Build and install on your host machine:

```
cmake -GNinja -B ../iree-build/ \
  -DCMAKE_C_COMPILER=clang \
  -DCMAKE_CXX_COMPILER=clang++ \
  -DCMAKE_INSTALL_PREFIX=../iree-build/install \
  -DCMAKE_BUILD_TYPE=RelWithDebInfo \
  .
cmake --build ../iree-build/ --target install
```

Try it on RPi4

On main branch, last commit: b21b732dc17f0722a41e96ca10a4fa4c46fbc0df

```
=============== start configuring IREE on ARM64 ===============
-- The ASM compiler identification is Clang with GNU-like command-line
-- Found assembler: /usr/bin/clang
-- The C compiler identification is Clang 15.0.4
-- The CXX compiler identification is Clang 15.0.4
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/clang - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/clang++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- IREE HAL drivers:
--   - local-sync
--   - local-task
--   - vulkan
-- IREE HAL local executable library loaders:
--   - embedded-elf
--   - system-library
--   - vmvx-module
-- IREE compiler input dialects:
--   - MHLO
--   - Torch MLIR
--   - TOSA
-- IREE compiler output formats:
--   - C source module
--   - VM Bytecode
--   - VM MLIR Assembly
-- Found Python3: /usr/bin/python3.11 (found version "3.11.0") found components: Interpreter
-- Looking for PyYAML - found
-- Found Git: /usr/bin/git (found version "2.38.1")
-- Adding bundled LLVM source dependency
-- IREE compiler target backends:
--   - llvm-cpu
--   - llvm-cpu (wasm)
--   - metal-spirv
--   - vulkan-spirv
--   - vmvx
...
```

```
-- Adding LLVM external project mlir-iree-dialects (MLIR_IREE_DIALECTS) → /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-main/llvm-external-projects/iree-di
alects
-- Adding LLVM external project mlir-hlo (MLIR_HLO) → /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-main/third_party/mlir-hlo
-- Adding LLVM external project torch-mlir-dialects (TORCH_MLIR_DIALECTS) → /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-main/third_party/torch-mlir-diale
cts
-- bolt project is disabled
-- clang project is disabled
-- clang-tools-extra project is disabled
-- compiler-rt project is disabled
-- cross-project-tests project is disabled
-- libc project is disabled
-- libclc project is disabled
-- lld project is enabled
-- lldb project is disabled
-- mlir project is enabled
-- openmp project is disabled
-- polly project is disabled
-- pstl project is disabled
-- flang project is disabled
-- mlir-iree-dialects project is enabled
-- mlir-hlo project is enabled
-- torch-mlir-dialects project is enabled
-- Found Python3: /usr/bin/python3.11 (found suitable version "3.11.0", minimum required is "3.0") found components: Interpreter
-- Performing Test LLVM_LIBSTDCXX_MIN
-- Performing Test LLVM_LIBSTDCXX_MIN - Success
-- Performing Test LLVM_LIBSTDCXX_SOFT_ERROR
-- Performing Test LLVM_LIBSTDCXX_SOFT_ERROR - Success
-- Looking for dlfcn.h
-- Looking for dlfcn.h - found
```

...

```
-- Performing Test HAVE_POSIX_REGEX
-- Performing Test HAVE_POSIX_REGEX
-- Performing Test HAVE_POSIX_REGEX -- success
-- Performing Test HAVE_STEADY_CLOCK
-- Performing Test HAVE_STEADY_CLOCK
-- Performing Test HAVE_STEADY_CLOCK -- success
-- SPIRV-Cross: Finding Git version for SPIRV-Cross.
-- SPIRV-Cross: Git hash: 50b4d538
-- Performing Test IREE_UK_BUILD_ARM_64_DOTPROD
-- Performing Test IREE_UK_BUILD_ARM_64_DOTPROD - Success
-- Performing Test IREE_UK_BUILD_ARM_64_I8MM
-- Performing Test IREE_UK_BUILD_ARM_64_I8MM - Success
-- IREE custom_dispatch/cpu/embedded ignored -- only builds for x86_64 (today)
-- IREE custom_dispatch/vulkan/shaders ignored -- glslc not found
-- Configuring done
-- Generating done
-- Build files have been written to: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build
```

...

```
=============== start building-installing IREE on ARM64 ===============
[0/2] Re-checking globbed directories...
[1/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/bitfield.c.o
[2/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/allocator.c.o
[3/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/loop.c.o
[4/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/string_builder.c.o
[5/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/loop_inline.c.o
[6/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/time.c.o
[7/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/status.c.o
[8/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/wait_source.c.o
[9/4934] Building C object runtime/src/iree/base/CMakeFiles/iree_base_base.objects.dir/string_view.c.o
[10/4934] Linking C static library runtime/src/iree/base/libiree_base_base.a
```

...

```
[4929/4934] Building C object samples/static_library/CMakeFiles/iree_samples_static_library_static_library_demo_c.dir/static_library_demo.c.o
[4930/4934] Linking CXX executable samples/static_library/static_library_demo
[4931/4934] Building C object samples/static_library/CMakeFiles/iree_samples_static_library_static_library_demo_c.dir/create_c_module.c.o
[4932/4934] Linking CXX executable samples/static_library/static_library_demo_c
[4933/4934] Linking CXX executable tools/iree-run-mlir
[4933/4934] Install the project...
-- Install configuration: "RelWithDebInfo"
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-flatcc-cli
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/generate_embed_data
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/lib64/libIREECompiler.so.0
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/lib64/libIREECompiler.so
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/flags_demo
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/fpu_state_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/synchronization_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/libdevice_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/mmt4d_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/pack_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/elf_module_test_binary
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/executable_library_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/resource_set_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/hello_world_file
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/hello_world_embedded
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/hello_world_terse
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/native_module_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/bytecode_module_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/bytecode_module_size_benchmark
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/lld
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-benchmark-module
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-benchmark-trace
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-check-module
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-dump-module
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-run-module
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-run-trace
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-e2e-matmul-test
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-tblgen
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-compile
-- Set runtime path of "/opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-compile" to "$ORIGIN:$ORIGIN/../lib64"
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-opt
-- Set runtime path of "/opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-opt" to "$ORIGIN:$ORIGIN/../lib64"
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-mlir-lsp-server
-- Set runtime path of "/opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-mlir-lsp-server" to "$ORIGIN:$ORIGIN/../lib64"
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/iree-run-mlir
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/tests/bin/FileCheck
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/tests/bin/not
```

```
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/static_library_demo
-- Installing: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build/install/bin/static_library_demo_c
[0/2] Re-checking globbed directories...
[1/594] Generating check_llvm-cpu-host_local-task_constant.mlir_module.vmfb from constant.mlir
[2/594] Generating check_llvm-cpu-host_local-task_divide.mlir_module.vmfb from divide.mlir
[3/594] Generating check_llvm-cpu-host_local-task_cosine.mlir_module.vmfb from cosine.mlir
[4/594] Generating check_llvm-cpu-host_local-task_concatenate.mlir_module.vmfb from concatenate.mlir
[5/594] Generating check_llvm-cpu-host_local-task_convert.mlir_module.vmfb from convert.mlir
[6/594] Generating check_llvm-cpu-host_local-task_exponential.mlir_module.vmfb from exponential.mlir
[7/594] Generating check_llvm-cpu-host_local-task_exponential_fp16.mlir_module.vmfb from exponential_fp16.mlir
[8/594] Generating check_llvm-cpu-host_local-task_dynamic_update_slice.mlir_module.vmfb from dynamic_update_slice.mlir
[9/594] Generating check_llvm-cpu-host_local-task_dynamic_slice.mlir_module.vmfb from dynamic_slice.mlir
[10/594] Generating check_llvm-cpu-host_local-task_convolution.mlir_module.vmfb from convolution.mlir
[11/594] Generating check_llvm-cpu-host_local-task_exponential_minus_one.mlir_module.vmfb from exponential_minus_one.mlir
[12/594] Generating check_llvm-cpu-host_local-task_dot_general.mlir_module.vmfb from dot_general.mlir
Unhandled aarch64 CPU feature: +neon
Unhandled aarch64 CPU feature: Unhandled aarch64 CPU feature: +neon
+neon
Unhandled aarch64 CPU feature: +neon
Unhandled aarch64 CPU feature: +neon
Unhandled aarch64 CPU feature: +neon
Unhandled aarch64 CPU feature: +neon
[13/594] Generating check_llvm-cpu-host_local-task_finite.mlir_module.vmfb from finite.mlir
[14/594] Generating check_llvm-cpu-host_local-task_log_plus_one.mlir_module.vmfb from log_plus_one.mlir
[15/594] Generating check_llvm-cpu-host_local-task_multiply.mlir_module.vmfb from multiply.mlir
...
[593/594] Generating check_llvm-cpu_local-task_mobilenetv3_fake_weights.mlir_module.vmfb from mobilenetv3_fake_weights.mlir
[594/594] Generating check_regression_llvm-cpu_lowering_config.mlir_module.vmfb from lowering_config.mlir
Internal ctest changing into directory: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build
Test project /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build
        Start    1: iree/build_tools/embed_data/c_embed_data_test
  1/1134 Test    #1: iree/build_tools/embed_data/c_embed_data_test ..........................................................   Passed    0.03 sec
        Start    2: iree/tests/compiler_driver/executable_benchmarks.mlir.test
  2/1134 Test    #2: iree/tests/compiler_driver/executable_benchmarks.mlir.test .............................................   Passed    2.09 sec
        Start    3: iree/tests/compiler_driver/hal_executable.mlir.test
  3/1134 Test    #3: iree/tests/compiler_driver/hal_executable.mlir.test ....................................................   Passed    0.75 sec
        Start    4: iree/tests/compiler_driver/smoketest.mlir.test
  4/1134 Test    #4: iree/tests/compiler_driver/smoketest.mlir.test .........................................................   Passed    1.22 sec
...
1133/1134 Test #1133: iree/samples/static_library/static_library_demo_test ..................................................   Passed    0.55 sec
        Start 1134: iree/samples/static_library/static_library_demo_c_test
1134/1134 Test #1134: iree/samples/static_library/static_library_demo_c_test ................................................   Passed    0.54 sec

95% tests passed, 58 tests failed out of 1134

Label Time Summary:
+dotprod                                                    =   0.07 sec*proc (4 tests)
+i8mm                                                       =   0.07 sec*proc (4 tests)
driver=local-sync                                           =   1.36 sec*proc (60 tests)
driver=local-task                                           = 408.12 sec*proc (378 tests)
driver=vulkan                                               = 630.16 sec*proc (162 tests)
hostonly                                                    = 405.01 sec*proc (92 tests)
iree/base                                                   =   0.91 sec*proc (6 tests)
iree/base/internal                                          =   2.37 sec*proc (13 tests)
iree/base/testing                                           =   0.05 sec*proc (1 test)
...
```

```
vulkan_uses_shader_int64                                        =   0.34 sec*proc (2 tests)
vulkan_uses_vk_khr_shader_float16_int8                          =   0.32 sec*proc (2 tests)

Total Test time (real) = 1005.04 sec

The following tests FAILED:
        30 - iree/tests/e2e/linalg_ext_ops/check_vulkan-spirv_vulkan_scatter.mlir (Failed)
        32 - iree/tests/e2e/linalg_ext_ops/check_vulkan-spirv_vulkan_winograd_input.mlir (Failed)
        33 - iree/tests/e2e/linalg_ext_ops/check_vulkan-spirv_vulkan_winograd_output.mlir (Failed)
        57 - iree/tests/e2e/matmul/e2e_matmul_direct_f32_gpu_large_ampere-unknown-linux_vulkan-spirv_vulkan (Failed)
        59 - iree/tests/e2e/models/collatz.mlir.test (Failed)
        61 - iree/tests/e2e/models/fragment_000.mlir.test (Failed)
        65 - iree/tests/e2e/models/unidirectional_lstm.mlir.test (Failed)
        68 - iree/tests/e2e/models/check_vulkan-spirv_vulkan_bert_encoder_unrolled_fake_weights.mlir (Failed)
        69 - iree/tests/e2e/models/check_vulkan-spirv_vulkan_mobilenetv3_fake_weights.mlir (Failed)
       119 - iree/tests/e2e/regression/check_regression_vulkan-spirv_dynamic_torch_index_select_scalar.mlir (Failed)
       122 - iree/tests/e2e/regression/check_regression_vulkan-spirv_linalg_ops.mlir (Failed)
       123 - iree/tests/e2e/regression/check_regression_vulkan-spirv_reduction_broadcast_elementwise.mlir (Failed)
       124 - iree/tests/e2e/regression/check_regression_vulkan-spirv_softmax.mlir (Failed)
       125 - iree/tests/e2e/regression/check_regression_vulkan-spirv_strided_slice.mlir (Failed)
       263 - iree/tests/e2e/tosa_ops/check_vulkan-spirv_vulkan_fully_connected.mlir (Failed)
       267 - iree/tests/e2e/tosa_ops/check_vulkan-spirv_vulkan_if.mlir (Failed)
       287 - iree/tests/e2e/tosa_ops/check_vulkan-spirv_vulkan_while.mlir (Failed)
       288 - iree/tests/e2e/vulkan_specific/check_vulkan-spirv_vulkan_f16_add_f16.mlir (Failed)
       289 - iree/tests/e2e/vulkan_specific/check_vulkan-spirv_vulkan_f16_dot_f16.mlir (Failed)
       404 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_abs.mlir (Failed)
       406 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_batch_norm_inference.mlir (Failed)
       407 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_broadcast.mlir (Failed)
       408 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_broadcast_add.mlir (Failed)
       409 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_broadcast_in_dim.mlir (Failed)
       410 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_clamp.mlir (Failed)
       411 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_compare.mlir (Failed)
       415 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_convolution.mlir (Failed)
       416 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_cosine.mlir (Failed)
       418 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_dot.mlir (Failed)
       419 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_dot_general.mlir (Failed)
       420 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_dynamic_slice.mlir (Failed)
       421 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_dynamic_update_slice.mlir (Failed)
       422 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_exponential.mlir (Failed)
       424 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_finite.mlir (Failed)
       425 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_floor.mlir (Failed)
       426 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_gather.mlir (Failed)
       428 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_log.mlir (Failed)
       430 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_maximum.mlir (Failed)
       431 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_minimum.mlir (Failed)
       432 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_multiply.mlir (Failed)
       433 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_negate.mlir (Failed)
       434 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_pad.mlir (Failed)
       435 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_pow.mlir (Failed)
       436 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_reduce.mlir (Failed)
       437 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_reduce_window.mlir (Failed)
       438 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_remainder.mlir (Failed)
       441 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_rng_normal.mlir (Failed)
       442 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_rng_uniform.mlir (Failed)
       444 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_rsqrt.mlir (Failed)
       445 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_scatter.mlir (Failed)
       447 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_select.mlir (Failed)
       448 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_sine.mlir (Failed)
       451 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_sqrt.mlir (Failed)
       454 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_torch_index_select.mlir (Failed)
       456 - iree/tests/e2e/xla_ops/check_vulkan-spirv_vulkan_while.mlir (Failed)
       513 - iree/tests/transform_dialect/cpu/matmul.mlir.test (Failed)
      1064 - iree/modules/check/test/success.mlir.test (Failed)
      1067 - iree/modules/check/test/check_vulkan-spirv_vulkan_success.mlir (Failed)
Errors while running CTest
```

total time for building plus testing cost ~8h

take a look around:

```
[mydev@fedora Official]$ tree iree-build/tools
iree-build/tools
├── android
│   ├── CMakeFiles
│   ├── cmake_install.cmake
│   ├── CTestTestfile.cmake
│   └── run_module_app
│       ├── CMakeFiles
│       ├── cmake_install.cmake
│       └── CTestTestfile.cmake
├── build_config.txt
├── CMakeFiles
│   ├── iree-benchmark-module.dir
│   │   └── iree-benchmark-module-main.cc.o
│   ├── iree-benchmark-trace.dir
│   │   └── iree-benchmark-trace-main.c.o
│   ├── iree-check-module.dir
│   │   └── iree-check-module-main.cc.o
│   ├── iree-compile.dir
│   │   └── iree-compile-main.cc.o
│   ├── iree-dump-module.dir
│   │   └── iree-dump-module-main.c.o
│   ├── iree-e2e-matmul-test.dir
│   │   └── iree-e2e-matmul-test.c.o
│   ├── iree-mlir-lsp-server.dir
│   │   └── iree-mlir-lsp-server.cc.o
│   ├── iree-opt.dir
│   │   └── iree-opt-main.cc.o
│   ├── iree-run-mlir.dir
│   │   └── iree-run-mlir-main.cc.o
│   ├── iree-run-module.dir
│   │   └── iree-run-module-main.cc.o
│   ├── iree-run-trace.dir
│   │   └── iree-run-trace-main.c.o
│   └── iree-tblgen.dir
│       └── ┌
│           ├── compiler
│           │   └── src
│           │       └── iree
│           │           └── compiler
│           │               └── Dialect
│           │                   └── VM
│           │                       └── Tools
│           │                           ├── VMOpEncoderGen.cpp.o
│           │                           └── VMOpTableGen.cpp.o
│           └── third_party
│               └── llvm-project
│                   └── mlir
│                       └── tools
│                           └── mlir-tblgen
│                               └── mlir-tblgen.cpp.o
├── cmake_install.cmake
├── CTestTestfile.cmake
├── iree-benchmark-module
├── iree-benchmark-trace
├── iree-check-module
├── iree-compile
├── iree-dump-module
├── iree-e2e-matmul-test
├── iree-mlir-lsp-server
├── iree-opt
├── iree-run-mlir
├── iree-run-module
├── iree-run-trace
├── iree-tblgen
├── test
│   ├── CMakeFiles
│   ├── cmake_install.cmake
│   ├── CTestTestfile.cmake
│   ├── iree_run_module_bytecode_module_vmvx.vmfb
│   └── iree_run_module_correctness_test_flagfile
└── test-iree-compiler-api-test-binary
```

```
[mydev@fedora Official]$ iree-build/tools/iree-compile --help
OVERVIEW: IREE compilation driver

USAGE: iree-compile [options] <input file or '-' for stdin>

OPTIONS:

Color Options:

  --color                                          - Use colors in output (default=autodetect)

General options:

  --aarch64-neon-syntax=<value>                    - Choose style of NEON code to emit from AArch64 backend:
    =generic                                       -   Emit generic NEON assembly
    =apple                                         -   Emit Apple-style NEON assembly
  --aarch64-use-aa                                 - Enable the use of AA during codegen.
  --abort-on-max-devirt-iterations-reached         - Abort when the max iterations for devirtualization CGSCC repeat pass is reached
  --allow-ginsert-as-artifact                      - Allow G_INSERT to be considered an artifact. Hack around AMDGPU test infinite loops.
  --arm-add-build-attributes                       -
  --arm-implicit-it=<value>                        - Allow conditional instructions outdside of an IT block
    =always                                        -   Accept in both ISAs, emit implicit ITs in Thumb
    =never                                         -   Warn in ARM, reject in Thumb
    =arm                                           -   Accept in ARM, reject in Thumb
    =thumb                                         -   Warn in ARM, emit implicit ITs in Thumb
  --atomic-counter-update-promoted                 - Do counter update using atomic fetch add  for promoted counters only
  --atomic-first-counter                           - Use atomic fetch add for first counter in a function (usually the entry counter)
  --bounds-checking-single-trap                    - Use one trap block per function
  --cfg-hide-cold-paths=<number>                   - Hide blocks with relative frequency below the given value
  --cfg-hide-deoptimize-paths                      -
  --cfg-hide-unreachable-paths                     -
  --compile-to=<value>                             - Compilation phase to run up until before emitting output.
    =input                                         -   Performs input processing and lowering into core IREE input dialects (linalg/etc).
    =abi                                           -   Adjusts program ABI for the specified execution environment.
    =flow                                          -   Compiles up to the `flow` dialect.
    =stream                                        -   Compiles up to the `stream` dialect.
    =hal                                           -   Compiles up to the `hal` dialect, including codegen.
    =vm                                            -   Compiles up to the `vm` dialect.
    =end                                           -   Complete the full compilation pipeline.
  --cost-kind=<value>                              - Target cost kind
    =throughput                                    -   Reciprocal throughput
    =latency                                       -   Instruction latency
    =code-size                                     -   Code size
    =size-latency                                  -   Code size and latency
  --debug-info-correlate                           - Use debug info to correlate profiles.
  --debugify-func-limit=<ulong>                    - Set max number of processed functions per pass.
  --debugify-level=<value>                         - Kind of debug info to add
    =locations                                     -   Locations only
    =location+variables                            -   Locations and Variables
```

▪ ▪ ▪

```
IREE options for controlling host/device scheduling.:

  --iree-execution-model=<value>                   - Specifies the execution model used for scheduling tensor compute operations.
    =host-only                                     -   Host-local code only that does not need execution scheduling.
    =async-internal                                -   Full HAL using asynchronous host/device execution internally but exporting functions as if synchronous.
    =async-external                                -   Full HAL using asynchronous host/device execution both internally and externally.
    =inline-static                                 -   Inline host-local in-process execution with executable code statically linked into the host program.
    =inline-dynamic                                -   Inline host-local in-process execution using dynamic executables.
  --iree-scheduling-dump-statistics-file=<string>  - File path to write statistics to; or `` for stderr or `-` for stdout.
  --iree-scheduling-dump-statistics-format=<value> - Dumps statistics in the specified output format.
    =pretty                                        -   Human-readable pretty printed output.
    =verbose                                       -   Pretty printed output with additional IR.
    =csv                                           -   Comma separated values.
    =json                                          -   JSON output with structures for data exchange

IREE options for controlling the input transformations to apply.:

  --iree-input-type=<value>                        - Specifies the input program representation.
    =none                                          -   No input dialect transformation.
    =mhlo                                          -   Legalize from MHLO ops.
    =xla                                           -   Legalize from MHLO ops (with XLA cleanup preprocessing).
    =tm_tensor                                     -   Legalize from TMTensor ops.
    =tosa                                          -   Legalize from TOSA ops.

IREE translation binding support options.:

  --iree-native-bindings-support                   - Include runtime support for native IREE ABI-compatible bindings.
  --iree-tflite-bindings-support                   - Include runtime support for the IREE TFLite compatibility bindings.
[mydev@fedora Official]$
```

2. Target environment setup

Target configuration

The following instruction shows how to build for a RISC-V 64-bit Linux machine. For other RISC-V targets, please refer to riscv.toolchain.cmake as a reference of how to set up the cmake configuration.

RISC-V 64-bit Linux target

```
cmake -GNinja -B ../iree-build-riscv/ \
  -DCMAKE_TOOLCHAIN_FILE="./build_tools/cmake/riscv.toolchain.cmake" \
  -DIREE_HOST_BINARY_ROOT=$(realpath ../iree-build/install) \
  -DRISCV_CPU=rv64 \
  -DIREE_BUILD_COMPILER=OFF \
  -DRISCV_TOOLCHAIN_ROOT=${RISCV_TOOLCHAIN_ROOT} \
  -DIREE_ENABLE_CPUINFO=OFF \
  .
cmake --build ../iree-build-riscv/
```

Try it on RPi4

2.1 prepare for the RISC-V GNU toolchain

base on the LLVM and GNU toolchain for RISC-V that we built in previous Testbed section

```
[mydev@fedora /]$ tree -L 1 /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install
/opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install
├── bin
├── include
├── lib
├── libexec
├── local
├── riscv64-unknown-linux-gnu
├── share
└── sysroot
```

directory “sysroot” and “riscv64-unknown-linux-gnu” are completely copied from the previously built GNU toolchain for RISC-V in previous Testbed section with the following configuration:

```
./configure --prefix=/opt/MyWorkSpace/DevSW/Toolchain/RISC-V/GCC/Official/Linux --enable-multilib
```

and add the riscv64-unknown-linux-gnu related binaries to directory “bin”:

```
[mydev@fedora /]$ ll /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin |grep -i  riscv64-unknown-linux-gnu
-rwxr-xr-x. 1 mydev mydev   4875992 Dec  9 10:43 riscv64-unknown-linux-gnu-addr2line*
-rwxr-xr-x. 1 mydev mydev   5079768 Dec  9 10:43 riscv64-unknown-linux-gnu-ar*
-rwxr-xr-x. 1 mydev mydev   6758872 Dec  9 10:43 riscv64-unknown-linux-gnu-as*
-rwxr-xr-x. 1 mydev mydev   6657312 Dec  9 10:43 riscv64-unknown-linux-gnu-c++*
-rwxr-xr-x. 1 mydev mydev   4825272 Dec  9 10:43 riscv64-unknown-linux-gnu-c++filt*
-rwxr-xr-x. 1 mydev mydev   6654400 Dec  9 10:43 riscv64-unknown-linux-gnu-cpp*
-rwxr-xr-x. 1 mydev mydev    155248 Dec  9 10:43 riscv64-unknown-linux-gnu-elfedit*
-rwxr-xr-x. 1 mydev mydev   6657312 Dec  9 10:43 riscv64-unknown-linux-gnu-g++*
-rwxr-xr-x. 1 mydev mydev   6652336 Dec  9 10:43 riscv64-unknown-linux-gnu-gcc*
-rwxr-xr-x. 1 mydev mydev   6652336 Dec  9 10:43 riscv64-unknown-linux-gnu-gcc-12.2.0*
-rwxr-xr-x. 1 mydev mydev    185640 Dec  9 10:43 riscv64-unknown-linux-gnu-gcc-ar*
-rwxr-xr-x. 1 mydev mydev    185472 Dec  9 10:43 riscv64-unknown-linux-gnu-gcc-nm*
-rwxr-xr-x. 1 mydev mydev    185472 Dec  9 10:43 riscv64-unknown-linux-gnu-gcc-ranlib*
-rwxr-xr-x. 1 mydev mydev   4671296 Dec  9 10:43 riscv64-unknown-linux-gnu-gcov*
-rwxr-xr-x. 1 mydev mydev   3118608 Dec  9 10:43 riscv64-unknown-linux-gnu-gcov-dump*
-rwxr-xr-x. 1 mydev mydev   3255784 Dec  9 10:43 riscv64-unknown-linux-gnu-gcov-tool*
-rwxr-xr-x. 1 mydev mydev  96997288 Dec  9 10:43 riscv64-unknown-linux-gnu-gdb*
-rwxr-xr-x. 1 mydev mydev      4045 Dec  9 10:43 riscv64-unknown-linux-gnu-gdb-add-index*
-rwxr-xr-x. 1 mydev mydev   6659816 Dec  9 10:43 riscv64-unknown-linux-gnu-gfortran*
-rwxr-xr-x. 1 mydev mydev   5390016 Dec  9 10:43 riscv64-unknown-linux-gnu-gprof*
-rwxr-xr-x. 1 mydev mydev   9783320 Dec  9 10:59 riscv64-unknown-linux-gnu-ld*
-rwxr-xr-x. 1 mydev mydev   9783320 Dec  9 10:43 riscv64-unknown-linux-gnu-ld.bak*
-rwxr-xr-x. 1 mydev mydev   9783320 Dec  9 10:43 riscv64-unknown-linux-gnu-ld.bfd*
-rwxr-xr-x. 1 mydev mydev 189686600 Dec  9 10:43 riscv64-unknown-linux-gnu-lto-dump*
-rwxr-xr-x. 1 mydev mydev   4914208 Dec  9 10:43 riscv64-unknown-linux-gnu-nm*
-rwxr-xr-x. 1 mydev mydev   5573800 Dec  9 10:43 riscv64-unknown-linux-gnu-objcopy*
-rwxr-xr-x. 1 mydev mydev   7974640 Dec  9 10:43 riscv64-unknown-linux-gnu-objdump*
-rwxr-xr-x. 1 mydev mydev   5079768 Dec  9 10:43 riscv64-unknown-linux-gnu-ranlib*
-rwxr-xr-x. 1 mydev mydev   3762488 Dec  9 10:43 riscv64-unknown-linux-gnu-readelf*
-rwxr-xr-x. 1 mydev mydev   7090960 Dec  9 10:43 riscv64-unknown-linux-gnu-run*
-rwxr-xr-x. 1 mydev mydev   4866016 Dec  9 10:43 riscv64-unknown-linux-gnu-size*
-rwxr-xr-x. 1 mydev mydev   4874168 Dec  9 10:43 riscv64-unknown-linux-gnu-strings*
-rwxr-xr-x. 1 mydev mydev   5573800 Dec  9 10:43 riscv64-unknown-linux-gnu-strip*
[mydev@fedora /]$
```

2.2 try to build the target

base on the LLVM and GNU toolchain for RISC-V that we built in Testbed section

```
cmake -GNinja -B ../iree-build-riscv/ -DCMAKE_TOOLCHAIN_FILE="./build_tools/cmake/riscv.toolchain.cmake" -DIREE_HOST_BINARY_ROOT=$(realpath ../iree-build/install) -DRISCV_CPU=rv64 -DIREE_BUILD_COMPILER=OFF -DRISCV_TOOLCHAIN_ROOT=/opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install -DIREE_ENABLE_CPUINFO=OFF .
```

build failed at configuration stage while doing compiler test:

```
=============== start configuring IREE-RISCV on ARM64 ===============
-- The ASM compiler identification is Clang with GNU-like command-line
-- Found assembler: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/clang
-- The C compiler identification is Clang 16.0.0
-- The CXX compiler identification is Clang 16.0.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - failed
-- Check for working C compiler: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/clang
-- Check for working C compiler: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/clang - broken
CMake Error at /usr/share/cmake/Modules/CMakeTestCCompiler.cmake:69 (message):
  The C compiler

    "/opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/clang"

  is not able to compile a simple test program.

  It fails with the following output:

    Change Dir: /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/IR/IREE/Official/iree-build-riscv/CMakeFiles/CMakeTmp

    Run Build Command(s):/usr/bin/ninja-build cmTC_4f438 && [1/2] Building C object CMakeFiles/cmTC_4f438.dir/testCCompiler.c.o
    [2/2] Linking C executable cmTC_4f438
    FAILED: cmTC_4f438
    : && /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/clang -march=rv64i2p0ma2p0f2p0d2p0c2p0 -mabi=lp64d -lstdc++ -lpthread -lm -ldl CMakeFiles/cmTC_4f438.dir/testCCompiler.c.o -o cmTC_4f438   && :
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: Relocations in generic ELF (EM: 183)
    /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/bin/riscv64-unknown-linux-gnu-ld: /lib/../lib64/Scrt1.o: error adding symbols: file in wrong format
    clang-16: error: linker command failed with exit code 1 (use -v to see invocation)
    ninja: build stopped: subcommand failed.
```

check https://github.com/iree-org/iree/blob/main/build_tools/cmake/riscv.toolchain.cmake:

```
42  if(RISCV_CPU STREQUAL "rv64")
43    set(CMAKE_SYSTEM_PROCESSOR riscv64)
44    set(CMAKE_SYSTEM_NAME Linux)
45    set(CMAKE_SYSTEM_LIBRARY_PATH "${RISCV_TOOLCHAIN_ROOT}/sysroot/usr/lib")
46    # Specify ISP spec for march=rv64gc. This is to resolve the mismatch between
47    # llvm and binutil ISA version.
48    set(RISCV_COMPILER_FLAGS "${RISCV_COMPILER_FLAGS} \
49        -march=rv64i2p0ma2p0f2p0d2p0c2p0 -mabi=lp64d")
50    set(RISCV_LINKER_FLAGS "${RISCV_LINKER_FLAGS} -lstdc++ -lpthread -lm -ldl")
51    set(RISCV64_TEST_DEFAULT_LLVM_FLAGS
52      "--iree-llvm-target-triple=riscv64"
53      "--iree-llvm-target-cpu=generic-rv64"
54      "--iree-llvm-target-abi=lp64d"
55      "--iree-llvm-target-cpu-features=+m,+a,+f,+d,+c,+v"
56      "--riscv-v-fixed-length-vector-lmul-max=8"
57      "--riscv-v-vector-bits-min=512"
58      CACHE INTERNAL "Default llvm codegen flags for testing purposes")
```

that’s why we need to build a GNU toolchain for with this configs…

```
--with-arch=rv64gcv --with-abi=lp64d
```

2.3 anyway, try to build the target with specify the

```
-DCMAKE_TOOLCHAIN_FILE="./build_tools/cmake/riscv.toolchain.cmake" \
```

build ok, but…

```
[mydev@fedora Official]$ tree -L 1 iree-build-riscv
iree-build-riscv
├── bin
├── build.ninja
├── build_tools
├── CMakeCache.txt
├── CMakeFiles
├── cmake_install.cmake
├── compile_commands.json
├── CTestCustom.cmake
├── CTestTestfile.cmake
├── lib
├── runtime
├── samples
├── tests
├── third_party
└── tools
```

```
[mydev@fedora Official]$ tree iree-build-riscv/tools
iree-build-riscv/tools
├── android
│   ├── CMakeFiles
│   ├── cmake_install.cmake
│   ├── CTestTestfile.cmake
│   └── run_module_app
│       ├── CMakeFiles
│       ├── cmake_install.cmake
│       └── CTestTestfile.cmake
├── build_config.txt
├── CMakeFiles
│   ├── iree-benchmark-module.dir
│   │   └── iree-benchmark-module-main.cc.o
│   ├── iree-benchmark-trace.dir
│   │   └── iree-benchmark-trace-main.c.o
│   ├── iree-check-module.dir
│   │   └── iree-check-module-main.cc.o
│   ├── iree-dump-module.dir
│   │   └── iree-dump-module-main.c.o
│   ├── iree-e2e-matmul-test.dir
│   │   └── iree-e2e-matmul-test.c.o
│   ├── iree-run-module.dir
│   │   └── iree-run-module-main.cc.o
│   └── iree-run-trace.dir
│       └── iree-run-trace-main.c.o
├── cmake_install.cmake
├── CTestTestfile.cmake
├── iree-benchmark-module
├── iree-benchmark-trace
├── iree-check-module
├── iree-dump-module
├── iree-e2e-matmul-test
├── iree-run-module
├── iree-run-trace
└── test
    ├── CMakeFiles
    ├── cmake_install.cmake
    ├── CTestTestfile.cmake
    ├── iree_run_module_bytecode_module_vmvx.vmfb
    └── iree_run_module_correctness_test_flagfile
```

…

```
[mydev@fedora Official]$ tree iree-build-riscv/bin
iree-build-riscv/bin

0 directories, 0 files
```

```
[mydev@fedora Official]$ tree iree-build-riscv/lib
iree-build-riscv/lib
├── libgmock.a
└── libgtest.a
```

```
[mydev@fedora Official]$ tree -L 1 iree-build-riscv/runtime
iree-build-riscv/runtime
├── bindings
├── CMakeFiles
├── cmake_install.cmake
├── CTestTestfile.cmake
└── src
```

```
[mydev@fedora Official]$ tree -L 1 iree-build-riscv/tests
iree-build-riscv/tests
├── CMakeFiles
├── cmake_install.cmake
├── compiler_driver
├── CTestTestfile.cmake
├── e2e
├── microbenchmarks
└── transform_dialect
```

```
[mydev@fedora Official]$ tree -L 1 iree-build-riscv/samples
iree-build-riscv/samples
├── CMakeFiles
├── cmake_install.cmake
├── CTestTestfile.cmake
├── custom_dispatch
├── custom_module
├── dynamic_shapes
├── emitc_modules
├── py_custom_module
├── simple_embedding
├── static_library
└── variables_and_state
```

```
[mydev@fedora Official]$ tree -L 2 iree-build-riscv/third_party
iree-build-riscv/third_party
├── benchmark
│   ├── CMakeFiles
│   ├── cmake_install.cmake
│   └── src
├── googletest
│   ├── CMakeFiles
│   ├── cmake_install.cmake
│   ├── CTestTestfile.cmake
│   ├── googlemock
│   └── googletest
└── vulkan_headers
    ├── CMakeFiles
    ├── cmake_install.cmake
    └── cmake_uninstall.cmake
```

run the official test at

https://iree-org.github.io/iree/building-from-source/riscv/

Running IREE bytecode modules on the RISC-V system

> **Note**
>
> The following instructions are meant for the RISC-V 64-bit Linux target. For the bare-metal target, please refer to simple_embedding to see how to build a ML workload for a bare-metal machine.

Set the path to qemu-riscv64 Linux emulator binary in the `QEMU_BIN` environment variable. If it is installed with `riscv_bootstrap.sh`, the path is default at ${HOME}/riscv/qemu/linux/RISCV/bin/qemu-riscv64.

```
export QEMU_BIN=<path to qemu-riscv64 binary>
```

Invoke the host compiler tools to produce a bytecode module FlatBuffer:

```
../iree-build/install/bin/iree-compile \
  --iree-hal-target-backends=vmvx \
  samples/models/simple_abs.mlir \
  -o /tmp/simple_abs_vmvx.vmfb
```

Run the RISC-V emulation:

```
${QEMU_BIN} \
  -cpu rv64 \
  -L ${RISCV_TOOLCHAIN_ROOT}/sysroot/ \
  ../iree-build-riscv/tools/iree-run-module \
  --device=local-task \
  --module_file=/tmp/simple_abs_vmvx.vmfb \
  --entry_function=abs \
  --function_input=f32=-5
```

```
[mydev@fedora Official]$ cat iree-main/samples/models/simple_abs.mlir
func.func @abs(%input : tensor<f32>) -> (tensor<f32>) {
  %result = math.absf %input : tensor<f32>
  return %result : tensor<f32>
}
```

```
[mydev@fedora Official]$ iree-build/install/bin/iree-compile --iree-hal-target-backends=vmvx  iree-main/samples/models/simple_abs.mlir -o Test/RISC-V/simple_abs_vmvx.vmfb
[mydev@fedora Official]$
[mydev@fedora Official]$ file Test/RISC-V/simple_abs_vmvx.vmfb
Test/RISC-V/simple_abs_vmvx.vmfb: Zip archive data, at least v4.5 to extract, compression method=store
[mydev@fedora Official]$
```

check the test file simple_abs_vmvx.vmfb

```
[mydev@fedora Official]$ cd Test/RISC-V/
[mydev@fedora RISC-V]$ ll
total 8
drwxr-xr-x. 1 mydev mydev   40 Dec  9 07:43 ./
drwxr-xr-x. 1 mydev mydev   12 Dec  9 07:41 ../
-rw-r--r--. 1 mydev mydev 5174 Dec  9 07:43 simple_abs_vmvx.vmfb
[mydev@fedora RISC-V]$
[mydev@fedora RISC-V]$
[mydev@fedora RISC-V]$ 7z x simple_abs_vmvx.vmfb

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,4 CPUs LE)

Scanning the drive for archives:
1 file, 5174 bytes (6 KiB)

Extracting archive: simple_abs_vmvx.vmfb
--
Path = simple_abs_vmvx.vmfb
Type = zip
Physical Size = 5174
64-bit = +

Everything is Ok

Files: 2
Size:       4721
Compressed: 5174
[mydev@fedora RISC-V]$ ll
total 16
drwxr-xr-x. 1 mydev mydev  126 Dec  9 07:44 ./
drwxr-xr-x. 1 mydev mydev   12 Dec  9 07:41 ../
-rw-r--r--. 1 mydev mydev  949 Jan  1  1980 abs_dispatch_0_vmvx_bytecode_fb.fb
-rw-r--r--. 1 mydev mydev 3772 Jan  1  1980 module.fb
-rw-r--r--. 1 mydev mydev 5174 Dec  9 07:43 simple_abs_vmvx.vmfb
[mydev@fedora RISC-V]$
```

run the test and got failed not suprisingly

```
[mydev@fedora Official]$ /usr/bin/qemu-riscv64 -cpu rv64 -L /opt/MyWorkSpace/MyProjs/Toolchain/LLVM/Official/llvm-project-main/build/Install/sysroot/ iree-build-riscv/tools
odule --device=local-task --module_file=Test/RISC-V/simple_abs_vmvx.vmfb --entry_function=abs --function_input=f32=-5
qemu-riscv64: iree-build-riscv/tools/iree-run-module: Invalid ELF image for this architecture
[mydev@fedora Official]$
```

```
[mydev@fedora Official]$ /usr/bin/qemu-riscv64 --help
usage: qemu-riscv64 [options] program [arguments...]
Linux CPU emulator (compiled for riscv64 emulation)
```

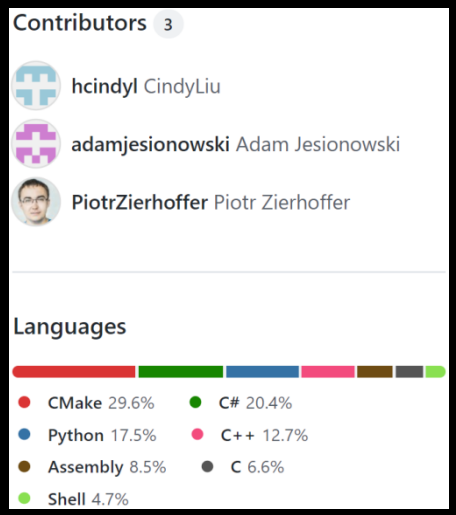
```
[mydev@fedora Official]$ /usr/bin/qemu-riscv64 -version
qemu-riscv64 version 7.0.0 (qemu-7.0.0-11.fc37)
Copyright (c) 2003-2022 Fabrice Bellard and the QEMU Project developers
[mydev@fedora Official]$
```

1.2 Springbok
1.2.1 Overview

- https://github.com/AmbiML/iree-rv32-springbok

RISC-V 32-Bit Bare-Metal ML Deployment on Springbok via IREE.

This project demonstrates how to compile RISC-V 32-bit ML workloads via IREE, and deploy the workloads with IREE's c API to generate the bare-matal executables. The built artifacts are targeted for Springbok, a RISC-V 32-bit bare-metal platform, and can be simulated with Renode.



Code structure

- build_tools: Utility scripts for the project
- cmake: CMake Macros for the project
- samples: Codegen and execution of ML models based on IREE
  - device: Device HAL driver library
  - float_model: float model examples
  - quant_model: quantized model examples
  - simple_vec_mul: Point-wise vector multiplication examples
  - util: Runtime utility library for model execution
- sim/config: Renode configuration and infrastructure
- springbok: Low-level code and linker scripts for the Springbok machine
- third_party/iree: IREE codebase

- **Springbok is an RISC-V core with the Vector extension (RVV) that runs machine learning (ML) workloads**
- …

1.2.2 Design

Python to RVV

The majority of machine learning modelling is performed in Python using frameworks like PyTorch, Tensorflow, or JAX

But Springbok is a bare-metal environment, we can't run a Python interpreter!

Solution: IREE

Source: https://open-src-soc.org/2022-05/media/slides/4th-RISC-V-Meeting-2022-05-03-14h00-Michael-Gielda-and-Adam-Jesionowski.pdf

Springbok HAL

- IREE's output consists of a virtual machine and the compiled ML output

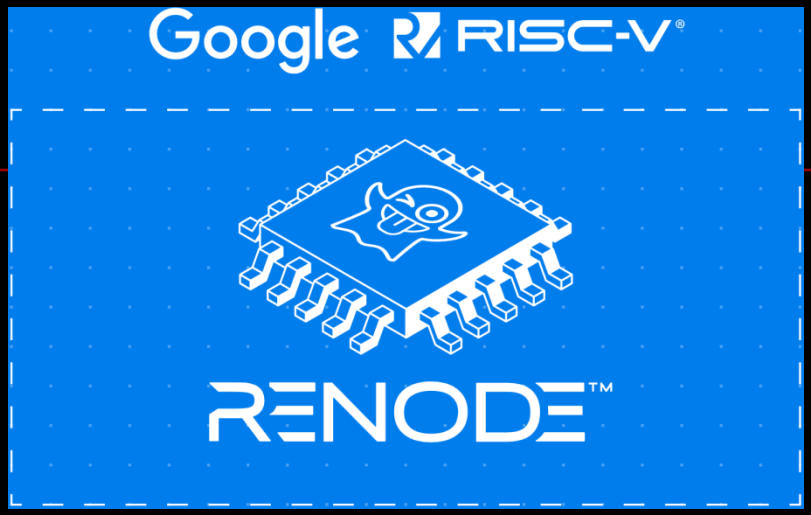
  It needs a Hardware Abstraction Layer (HAL) to operate on RISC-V and a scheduler

  Our code provides an example of bare-metal execution on RISC-V

  Source: https://open-src-soc.org/2022-05/media/slides/4th-RISC-V-Meeting-2022-05-03-14h00-Michael-Gielda-and-Adam-Jesionowski.pdf

- ...

1.2.3 Springbok with Renode



Source: https://renode.io/news/co-developing-ml-with-risc-v-using-renode-for-google-research/

- https://zephyrproject.org/renode-1-13-for-improved-machine-learning-and-pre-silicon-development/

RISC-V support for Vector Extensions, new platforms and more

One of the most exciting advancements coming with Renode 1.13 is the overall progress with RISC-V support. As part of our long time collaboration with Google, we introduced features such as RISC-V Vector Extensions v1.0 and Custom Function Units support, opening up even more options in developing ML-enabled products with RISC-V. Other additions include platforms such as the JH7100 used in BeagleV and ARVSOM, as well as new peripherals and improvements for the OpenTitan platform. Thanks to our collaboration with Google and the Sound Open Firmware project we also added initial support for the Xtensa architecture, extending our simulation framework's capabilities to even more use cases.

Other changes include improvement of ARM architecture support, namely:

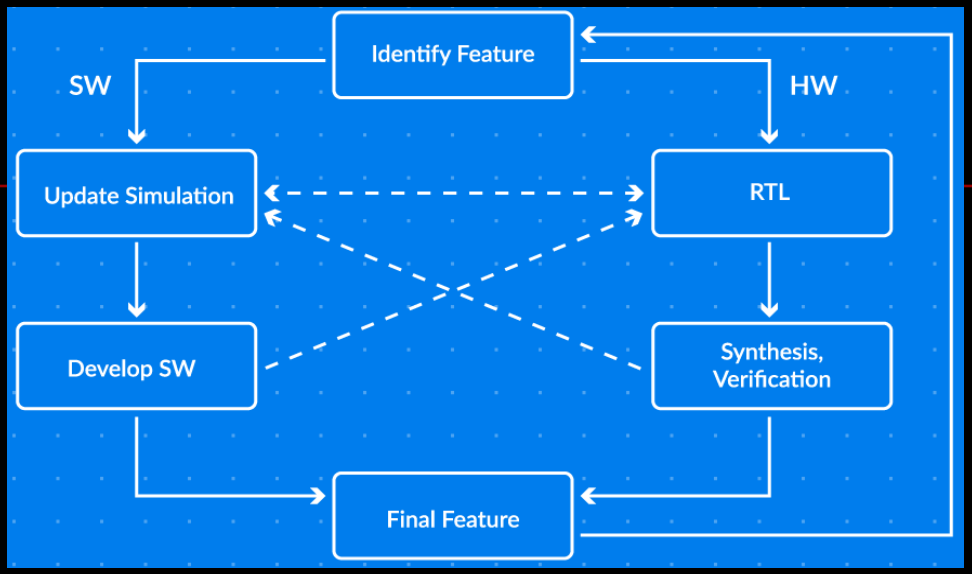
- Floating Point Unit (FPU) support in Cortex-M
- Cortex-M33 stub support
- Basic support for 64-bit registers

- …

1.2.3.1 RVV support in Renode

- https://antmicro.com/blog/2021/12/riscv-vector-instructions-in-renode/
- https://renode.io/news/co-developing-ml-with-risc-v-using-renode-for-google-research/
- https://riscv.org/blog/2022/07/co-developing-machine-learning-with-a-risc-v-vector-core-using-renode-for-google-research-antmicro-2/
- https://github.com/antmicro/renode-riscv-rvv-stress-test
- …

1.2.3.2 Co-simulating with Renode



Renode

File Edit View Search Terminal Help

```
12:09:39.1567 [WARNING] The debug mode now has no effect - connect a debugger, a
nd switch to stepping mode.
12:09:39.4753 [INFO] sysbus: Loading segment of 329144 bytes length at 0x3200000
0.
12:09:39.4835 [INFO] sysbus: Loading segment of 16777216 bytes length at 0x34000
000.
12:09:39.5279 [INFO] cpu2: Setting PC value to 0x32000000.
Starting emulation...
12:09:39.5568 [INFO] springbok: Machine started.
12:09:39.6809 [WARNING] cpu2: The debug mode now has no effect - connect a debug
ger, and switch to stepping mode.
(springbok) 12:09:40.7074 [INFO] cpu2: simprint: "INFO |Image prediction result
is: id: 178", 0 (0x0)
12:09:40.7100 [INFO] cpu2: simprint: "[[ iree_hal_allocator_t memory statistics
]]", 0 (0x0)
12:09:40.7101 [INFO] cpu2: simprint: "  HOST_LOCAL:       150528B peak /       1
50528B allocated /       150528B freed /            0B live", 0 (0x0)
12:09:40.7102 [INFO] cpu2: simprint: "DEVICE_LOCAL:       760937B peak /       7
60937B allocated /       760937B freed /            0B live", 0 (0x0)
12:09:40.7105 [INFO] cpu2: simprint: "INFO |mobilenet_v1_0.25_224_quant finished
 successfully", 0 (0x0)
12:09:40.7107 [INFO] cpu2: simprint: "main returned: ", 0 (0x0)

(springbok)
```

Source: https://opensource.googleblog.com/2022/09/co-simulating-ml-with-springbok-using-renode.html

1.2.4 Our effort for make Renode running on ARM

1.2.4.1 Methodologies

Directly porting

- A heavy work…

Try to make a prebuilt version running

- Could not resolve some issues by now…

Virtualization

- Trying some new methods and getting a little progress, will summarize it soon…

Renode on GraalVM

- Also working on it…

  From our point of view:

  1. The most popular Polyglot runtimes: .Net, Wasm, GraalVM.
  2. Cross-platform ecological integration for Polyglot programming is the future.

1.2.4.2 Summary

- For more details, you may refer to our previous talk "Renode: a Swiss Army Knife for RISC-V development" at .Net Conf China 2021(Online) and the upcoming follow-up "Revisiting Renode as a Swiss Army Knife for RISC-V development"
- Continuously updating "Exploration of running C# program on GraalVM" at .Net Conf China 2022(Online) as a more broader topic "Cross-platform integration for Polyglot Programming" (the latest slides is put at: https://github.com/XianBeiTuoBaFeng2015/MySlides/blob/master/LTS/Cross-platform%20integration%20for%20Polyglot%20Programming__20221203p.pdf)
- …

IV. TinyIREE

- …

  An ML Execution Environment for Embedded Systems from Compilation to Deployment.

- The last step in the device-side compilation is lowering the program into the LLVM dialect, which allows the program to be mechanically converted into LLVM IR. The LLVM compilation stack can then generate the binary code for the target architecture. The LLVM target flags can be used to select which CPU architecture, ABI, and ISA extensions to use. For example, to build the module for x86_64 CPU, apply the following flag:

  – –iree–llvm–target–triple=x86_64–pc–linux–elf

  Whereas for RISC-V 32-bit CPU with multiplication and floating point ISA extension support, use these flags:

  – –iree–llvm–target–triple=riscv32–pc–linux–elf
  – –iree–llvm–target–cpu=generic–rv32
  – –iree–llvm–target–cpu–features=+m,+f
  – –iree–llvm–target–abi=ilp32

  In addition, for Armv7E-M CPU, use the following flags:

  – –iree–llvm–target–triple=armv7em–pc–linux–elf
  – –iree–llvm–target–float–abi=hard

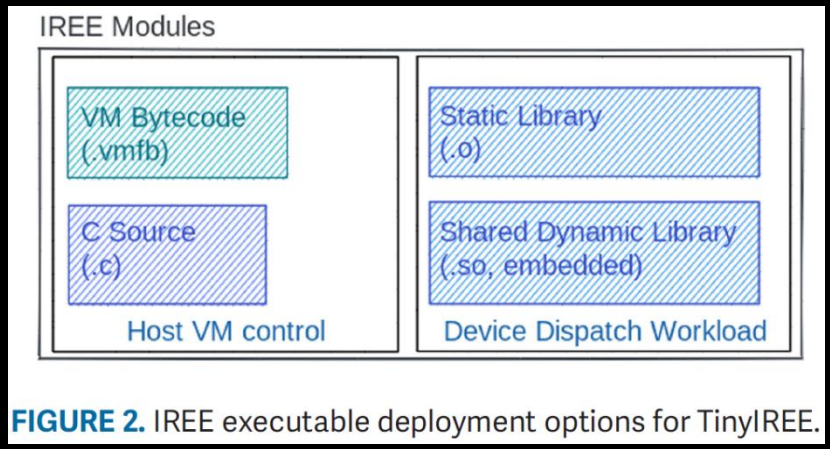


FIGURE 2. IREE executable deployment options for TinyIREE.

Source: https://csdl-downloads.ieeecomputer.org/mags/mi/2022/05/09782563.pdf

- **current result**

TABLE 1. IREE Module FlatBuffer and the embedded device executable workload size for mobilenet V2 SSD model compiled for different CPU architecture targets.

Mode	FlatBuffer size (kB)	Workload size (kB)
x86_64		
Debug-dylib	5,588	256.55
Dylib	5,525	208.35
Embedded	5,521	203.80
Static	5,317	–
Armv7E-M		
Debug-dylib	5,468	136.55
Dylib	5,420	102.98
Embedded	5,406	88.84
Static	5,317	–
RISC-V32imf		
Debug-dylib	5,500	168.42
Dylib	5,437	120.18
Embedded	5,437	120.18
Static	5,317	–

TABLE 2. Runtime peak memory usage and the host library size for running mobilenet V2 SSD model with TFLite interpreter versus IREE runtime application.

	Peak memory (MB)	Host library (kB)
TFLite		
x86_64	20.05	2971.06
IREE		
(Embedded)		
x86_64	5.93	96.31
Armv7E-M	5.93	79.96
RISC-V32imf	5.93	152.86

Source: https://csdl-downloads.ieeecomputer.org/mags/mi/2022/05/09782563.pdf

- **https://github.com/iml130/iree-bare-metal-arm**

IREE Bare-Metal Arm Sample

Build and Test passing

DISCLAIMER: This project is not intended for everyday use and made available without any support. However, we welcome any kind of feedback via the issue tracker or if appropriate via IREE's communication channels, e.g. via the Discord server.

This projects demonstrates how to build IREE with the Arm GNU Toolchain for bare-metal Arm targets using either the open-source firmware library libopencm3 or CMSIS.

Wasm on Embedded Systems

- You may refer to our previous talk **"AOT compilation based Wasm compiler and runtime for Serverless Edge computing"** at OpenInfra Days China 2021(Beijing) and the upcoming follow-ups.
- …

V. Wrap-up

- A new Golden Age for AI compiler and runtime!
- You may look forward to our upcoming follow-ups "Revisiting IREE as a MLIR-based end-to-end compiler & runtime for Machine Learning " and more.

Q & A

Reference

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- http://www.slideshare.net/
- https://en.wikipedia.org/wiki/Standard_Portable_Intermediate_Representation
- https://en.wikipedia.org/wiki/Vulkan
- https://www.renesas.com/us/en/products/microcontrollers-microprocessors/rz-mpus/rzfive-general-purpose-microprocessors-risc-v-cpu-core-andes-ax45mp-single-10-ghz-2ch-gigabit-ethernet
- …