LLVM toolchains

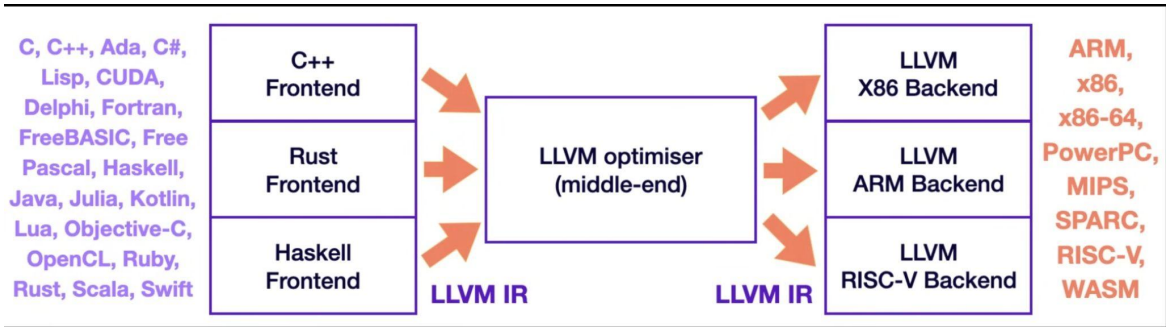
Davide Grassano (@Crivella)
CECAM Lausanne - Switzerland

Outline

- A brief recap on LLVM
- New updates in LLVM
- New updates in Easybuild
  - New toolchains
  - Software available
  - What are the challenges?
- Where to go from here

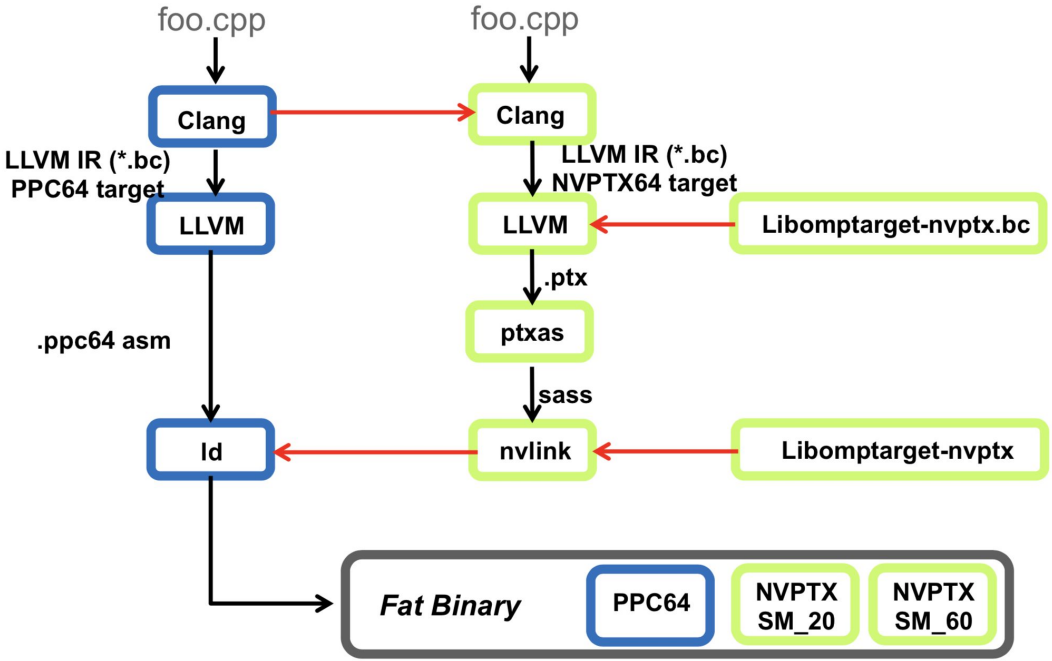
Previously… - A recap on LLVM

- Decoupling (LLVM IR)
  - Completeness of the IR enables total decoupling of the front and back-end
- Modularization
  - Organized as a set of libraries
- Reusability
  - APIs allow to reuse/mix functionalities from several existing components by new tools



Previously… - A recap on LLVM

Previously… - A recap on LLVM



Previously… - LLVM toolchain

- Standalone
  - No leftover GCC dependencies
  - With a more mature libc could allow for a more system independent toolchain?
  - Needs to redo all the GCCcore EC files
- On top of GCCcore
  - Reuse existing GCCcore dependencies
  - Risc mixing of OpenMP and compiler runtime libraries
    - Seems that calling a **libgomp** compiled libraries from a **libomp** linked executable works as intended

Previously… - LLVM toolchain

- Standalone
  - No leftover GCC dependencies
  - With a more mature libc could allow for a more system independent toolchain?
  - Needs to redo all the GCCcore EC files
- On top of GCCcore
  - Reuse existing GCCcore dependencies
  - Risc mixing of OpenMP and compiler runtime libraries
    - Seems that calling a **libgomp** compiled libraries from a **libomp** linked executable works as intended

What’s new in LLVM - 20.1.x

- Target specific changes
  - AMD
    - Support for ***gfx950***
    - Headers for common GPU builtins for OpenMP/CUDA/HIP/OpenCL/C/C++
  - NVPTX
    - Headers for common GPU builtins for OpenMP/CUDA/HIP/OpenCL/C/C++
  - x86
    - Support ISA: AMX-FP8, AMX-TRANSPONSE, AMX-MOVRS, AMX-AVX512, AMX-TF32, MOVRS
    - ***-march/tune*** for diamondrapids Support
  - Arm/AArch64
    - Implementation of SVE2.1 and SME2.1
  - RISC-V
    - The option -mcmodel=large for the large code model is supported.
  - CUDA
    - Support up to CUDA SDK 12.6
    - Support for ***sm_100***
    - Uses offload driver by default
  - Others

What’s new in LLVM - 20.1.x

- Target specific changes
- Improved OpenMP support
- Clang
  - Several changes in flag behaviors of Clang
  - Implemented several missing features for C++2c, C++23, C++20, C++17, C2y, C23
  - Improvements to clang diagnostic/static-analyzer/sanitizers

What’s new in LLVM - 21.1.x

- Target specific changes
  - AMD
    - Code object version bumped to 6. ROCm 6.3 required for programs compiled with COV6
  - x86
    - The 256-bit maximum vector register size control was removed from AVX10
    - Re-target m[no-]avx10.1 to enable AVX10.1 with 512-bit maximum vector register size
  - Arm/AArch64
    - Implementation of modal 8-bit floating point intrinsics
    - Support for Cortex-A320 processor
    - The +nosimd attribute is now fully supported for ARM. Previously, this had no effect when being used with ARM targets, however this will now disable NEON instructions being generated.
    - Fix SVE regression llvm-project#130973
    - Includes also FPU features in assembly files for the specific target architecture
  - RISC-V
    - Add support for -mtune=generic-ooo
    - Add support for several __attribute__ interrupts
  - Others

What’s new in LLVM - 21.1.x

- Clang
  - Changes in flag behaviors of Clang
  - Implemented several missing features for C++2c, C++20, C2y, C23, C11
  - Improvements to clang diagnostic/static-analyzer/sanitizers
- Flang
  - Better tracking of feature support
    - 2003: complete with a few feature with partial support
    - 2008: one missing feature and one with partial support
    - 2023: WIP
  - Changes to runtime libraries names
  - Allow building flang runtime for multiple target triples
  - Support for flags: -floop-interchange, -fveclib=libmvec (aarch64)

What’s new in LLVM - 22.1.x

- Target specific changes
  - x86
    - More SSE, AVX, AVX512 intrinsics
    - Add ***-march=wildcatlake*** and ***-march=novalake***
  - Arm/AArch64
    - Implementation of modal 8-bit floating point intrinsics
    - Support for processors: Ampere1C, Arm C1-Nano/Pro/Premium/Ultra
    - More intrinsics for several AArch64 instructions
  - RISC-V
    - Completed support for ***Smrnmi*** extension
    - Add ***-march=unset***
  - CUDA/HIP
    - Suppot for C++17 Class Template Argument Deduction (CTAD)
  - CUDA
    - Fix for assertions failures with ***-fsyntax-only*** or device architecture flags
  - Others
    - Add support for new OpenMP clauses

What’s new in LLVM - 22.1.x

- Clang
  - Changes in flag behaviors of Clang
  - Implemented several missing features for C++2c, C++20, C2y, C23
  - Improvements to clang diagnostic/static-analyzer/sanitizers
  - New warning in GCC toolchain detection if one does not have *libstdc++* available (llvm-project#145056)
- Flang
  - Better support for ***do concurrent*** mapping to OpenMP
  - EXPERIMENTAL Multi-image program launch (***-fcoarray***) (2008/2018 standard)
  - Support for LOWER= argument for C_F_POINTER (2023 standard)

Feature comparison - PREVIOUSLY

C++

	LLVM	GNU
c++11	DONE	DONE
c++14	DONE	DONE
c++17	ALMOST	ALMOST
c++20	ALMOST	EXPERIMENTAL
c++23	EXPERIMENTAL	EXPERIMENTAL
c++2c	WIP	EXPERIMENTAL

FORTRAN

	LLVM	GNU
1995	DONE	DONE
2003	~DONE	DONE
2008	ALMOST	ALMOST
2018	EXPERIMENTAL	EXPERIMENTAL
2023	WIP	WIP

Feature comparison - NOW

C++

	LLVM	GNU
c++11	DONE	DONE
c++14	DONE	DONE
c++17	DONE	DONE
c++20	~DONE	DONE
c++23	ALMOST	ALMOST
c++2c	EXPERIMENTAL	EXPERIMENTAL

FORTRAN

	LLVM	GNU
1995	DONE	DONE
2003	~DONE	DONE
2008	ALMOST	ALMOST
2018	ALMOST	ALMOST
2023	EXPERIMENTAL	WIP

What’s new in Easybuild (w.r.t LLVM)

- Framework: ~6 PRs
- Easyblocks: ~40 PRs
- Easyconfigs: ~50 PRs

What’s new in Easybuild - Framework

- Add files for the new llvm toolchain definition

lfbf	llvm-compilers	*(none)*	FlexiBLAS	FFTW
lfoss	llvm-compilers	OpenMPI	FlexiBLAS, ScaLAPACK	FFTW
llvm-compilers	llvm-compilers	*(none)*	*(none)*	*(none)*
lmpflf	llvm-compilers	MPICH	FlexiBLAS, ScaLAPACK	FFTW
lmpich	llvm-compilers	MPICH	*(none)*	*(none)*
lompi	llvm-compilers	OpenMPI	*(none)*	*(none)*
lpsflf	llvm-compilers	psmpi	FlexiBLAS, ScaLAPACK	FFTW
lpsmpi	llvm-compilers	psmpi	*(none)*	*(none)*

What's new in Easybuild - Framework

- Add files for the new llvm toolchain definition
- Generalize linkers for toolchains
  - Add support for LLVM specific linkers
- Toolchain specific options
  - *lld_undefined_version*: False[/True] adds -Wl,--undefined-version to C/Fortran
  - *no_unused_args*: True[/False] adds -Wno-unused-command-line-argument to C
  - *no_int_conversion_error*: True[/False] adds -Wno-error=int-conversion
- easybuild-framework#4914

What’s new in Easybuild - Easyblocks

- Adapted Easyblocks with behavior specific to the Toolchain to include LLVM
  - Boost
  - ScoreP
  - FlexiBlas
  - Psmpi
  - QuantumEspresso

What's new in Easybuild - Easyblocks

- Adapted Easyblocks with behavior specific to the Toolchain to include LLVM
- Improvements/fixes and new features to the EB_LLVM easyblock
  - RISC-V support and improved Aarch64 support (#3676, #4059)
  - *sysroot* support (tested on EESSI) (#3741, …)
    - *--sysroot*, *-Wl,-dynamic-linker*
  - Improved support for GPU targets and tests
  - Improvement to installed libomp libraries (and more configurations via easyblock options) (#3815, …)
  - Improved sanity checks w.r.t runtime vs build dependencies and system dependencies (#3877)
  - Allow embedding ECs using EB_LLVM in a Bundle (support for ROCm-LLVM) (#3781)
  - Can now be used with *--rebuild* + *--sanity-check-only* or *--module-only* (#3977, …)
  - support for newer LLVM versions (up to 21.1.x and open PR for 22.1.x)
  - Several minor fixes

What’s new in Easybuild - Easyconfigs

- Add new ECs for versions 19.1.x, 20.1.x. and 21.1.x
- Patches for RISC-V support
- Add more required dependencies explicitly to EC files
- Add patches for better CUDA compatibility
- 48 new ECs using LLVM as a dependency
  - Mesa/OpenGL
  - numba
  - Triton
  - …
- New files for 2025b and 2026.1 LLVM toolchains

Software available in the LLVM toolchain

- 2025b (GCCcore-14.3.0, LLVM-20.1.8)
  - llvm-compilers
    - LinAlg: FFTW, BLIS, OpenBLAS, AOCL-BLAS, FlexiBLAS
    - MPI: OpenMPI, MPICH
    - others: Boost, AdaptiveCpp, pybind11, GSL, libxc, WCSLIB
  - lfbf:
    - others: networkx, SciPy-bundle, xarray
  - lompi:
    - LinAlg: FFTW.MPI, ScaLAPACK
    - others: netCDF, HDF5, ScoreP, Scalasca, MUST, mpi4py
  - lfoss
    - others: arpack-ng, casacore, ELPA, GROMACS, Hypre, HPL, h5py, HPCG, QuantumEspresso
  - lmpich:
    - others: ScoreP, Scalasca, MUST, mpi4py
  - lmpflf:
    - others: HPL, HPCG, GROMACS

Software available in the LLVM toolchain

- 2026.1 (GCCcore-15.3.0, LLVM-21.1.8)
  - llvm-compilers
    - LinAlg: FFTW, BLIS, OpenBLAS, AOCL-BLAS, FlexiBLAS
    - MPI: OpenMPI
  - lfbf:
  - lompi:
    - LinAlg: FFTW.MPI, ScaLAPACK
  - lfoss
    - others: HPL
  - lmpich:
  - lmpflf:

Challenges - HDF5

- Previously (using autoconf)
  - Configure script runs **flang** and uses the output to determine input options for CFLAGS
    - *-target-feature -lwp* ends up being interpreted as a non-existing library by clang
    - This is emitted only in case of **-march=native** (temporary fix disable this)
  - Weird one: double quotes in the compiler version string (extracted from **mpicc+clang**) gets injected into a template file for h5cc without being escaped

Challenges - HDF5

- Previously (using autoconf)
  - Configure script runs **flang** and uses the output to determine input options for CFLAGS
    - *-target-feature -lwp* ends up being interpreted as a non-existing library by clang
    - This is emitted only in case of **-march=native** (temporary fix disable this)
  - Weird one: double quotes in the compiler version string (extracted from **mpicc+clang**) gets injected into a template file for h5cc without being escaped
- Switch to CMake builds

```
error: loc("/software/OpenMPI/5.0.8-llvm-compilers-20.1.8/lib/mpi_f08.mod":429:24):
    'hlfir.declare' op of numeric, logical, or assumed type entity must not have length parameters
error: verification of lowering to FIR failed
...
make[2]: *** [fortran/testpar/CMakeFiles/parallel_test.dir/build.make:120:
    fortran/testpar/CMakeFiles/parallel_test.dir/mpi_param.F90.o] Error 1
```

Challenges - HDF5

- Previously (using autoconf)
  - Configure script runs **flang** and uses the output to determine input options for CFLAGS
    - *-target-feature -lwp* ends up being interpreted as a non-existing library by clang
    - This is emitted only in case of **-march=native** (temporary fix disable this)
  - Weird one: double quotes in the compiler version string (extracted from **mpicc+clang**) gets injected into a template file for h5cc without being escaped
- Switch to CMake builds
- Back-port patch from LLVM 21.1.x for handling common blocks used as BIND(C) to 20.1.8 for the 2025b toolchain

Challenges - ELPA

- libtool does not properly set ***$wl***
  - flag used to pass flags to the compiler to thee linker
- Patch libtool in the prebuildopts to replace ***$wl*** with ***-Wl***,
- Configure script runs flang and uses the output to determine input options for CFLAGS
  - ***-target-feature -lwp*** ends up being interpreted as a non-existing library by clang
  - This is emitted only in case of ***-march=native*** (temporary fix disable this)

Challenges - ELPA

- libtool does not properly set ***$wl***
  - flag used to pass flags to the compiler to thee linker
- Patch libtool in the prebuildopts to replace ***$wl*** with ***-Wl***,
- Configure script runs flang and uses the output to determine input options for CFLAGS
  - ***-target-feature -lwp*** ends up being interpreted as a non-existing library by clang
  - This is emitted only in case of ***-march=native*** (temporary fix disable this) -> problem on AMD Zen4

```
checking whether we can compile AVX gcc intrinsics in C... no
configure: error: Could not compile a test program with AVX, try with
--disable-avx, or adjust the C compiler or CFLAGS. Possibly (some of) the flags
"   -mmmx -msse -msse2 -msse3 -mssse3 -msse4.1 -msse4.2 -msse4a -msha -maes
-mavx -mfma -mavx2 -mavx512f -mavx512cd -mavx512vl -mavx512bw -mavx512dq
-mavx512ifma -mavx512vbmi " solve this issue
```

Challenges - ELPA

- libtool does not properly set ***$wl***
  - flag used to pass flags to the compiler to thee linker
- Patch libtool in the prebuildopts to replace ***$wl*** with ***-Wl***,
- Configure script runs flang and uses the output to determine input options for CFLAGS
  - ***-target-feature -lwp*** ends up being interpreted as a non-existing library by clang
  - This is emitted only in case of ***-march=native*** (temporary fix disable this) -> problem on AMD Zen4
  - Patch Autoconf to not parse flags passed through ***-target-feature*** (@Thyre #24669)
- Precompiler problems
  - Some solved by moving to 20.1.8 from the earlier 20.1.2
  - Some solved by using a newer version of ELPA (tested from 2025.01.002)

Challenges - QuantumEspresso

- Try with all feature/plugins

```
loc("../7.5/lfoss-2023b/qe-7.5/external/d3q/src/d3_shuffle.f90":190:1): .../
LLVM/20.1.5/GCCcore-13.2.0/llvm-project-20.1.5.src/flang/lib/Lower/
CallInterface.cpp:1115: not yet implemented: VOLATILE in procedure interface
```

Challenges - QuantumEspresso

- Try with all feature/plugins
  - Try back-porting PATCH from PR implementing VOLATILE in procedure interface merged in 21.1.x
  - Too many conflicts + requires features from other PRs to work
- Disable d3q plugin until LLVM>=21.1.x (2026.1 toolcahin uses 21.1.8)

Challenges - QuantumEspresso

- Try with all feature/plugins
  - Try back-porting PATCH from PR implementing VOLATILE in procedure interface merged in 21.1.x
  - Too many conflicts + requires features from other PRs to work
- Disable d3q plugin until LLVM>=21.1.x (2026.1 toolcahin uses 21.1.8)
- Build on Arm Neoverse V1

```
ld.lld: error: undefined symbol: __trampoline_setup
...
flang-20: error: linker command failed with exit code 1 (use -v to see
invocation)
```

Challenges - QuantumEspresso

- Try with all feature/plugins
  - Try back-porting PATCH from PR implementing VOLATILE in procedure interface merged in 21.1.x
  - Too many conflicts + requires features from other PRs to work
- Disable d3q plugin until LLVM>=21.1.x (2026.1 toolcahin uses 21.1.8)
- Build on Arm Neoverse V1
  - __trampoline_setup requires compiler-rt to be used together with flang
  - Back-port patch from 21.1.x to always link compiler-rt to flang

Challenges - ScipyBundle

- Numpy and flang
  - Patches to numpy
    - Numpy does not select the correct real kind using flang
    - Fix compiler flags in f2py for mixing C linker with fortran sources
- Architecture specific problems
  - Neoverse V1
    - OpenBLAS + numpy: *RuntimeWarning: divide by zero*
    - BLIS: *fatal error: instruction requires: sve or sme*
      - Disable ***-march=native*** in favor of -***march=armv8-a+sve*** added by BLIS by default
  - Neoverse V2
    - OpenBLAS + numpy: *RuntimeWarning: divide by zero*
    - BLIS + scipy: tolerance for test result exceeded (was already being increased via a patch)
- See easybuild-easyconfigs/issues/24764 for more details

Where to go from here?

- Toolchain
  - Continue building more software with the toolchain
  - Test on effect of current optimization flags
  - Streamline toolchain reusability/compatibility with LLVM-derived compilers
- LLVM
  - Switch from CMakeMake to CMakeNinja to support more control over the build
  - Add checks/warnings for potential issues related to low RAM/core or filesystems
  - Add support for experimental/external features such as llvm-libc and llvm-spirv (possible SYCL toolchain)
  - Keep supporting new LLVM versions
    - New CMake configuration options
    - New project/runtimes
    - New project and install structure

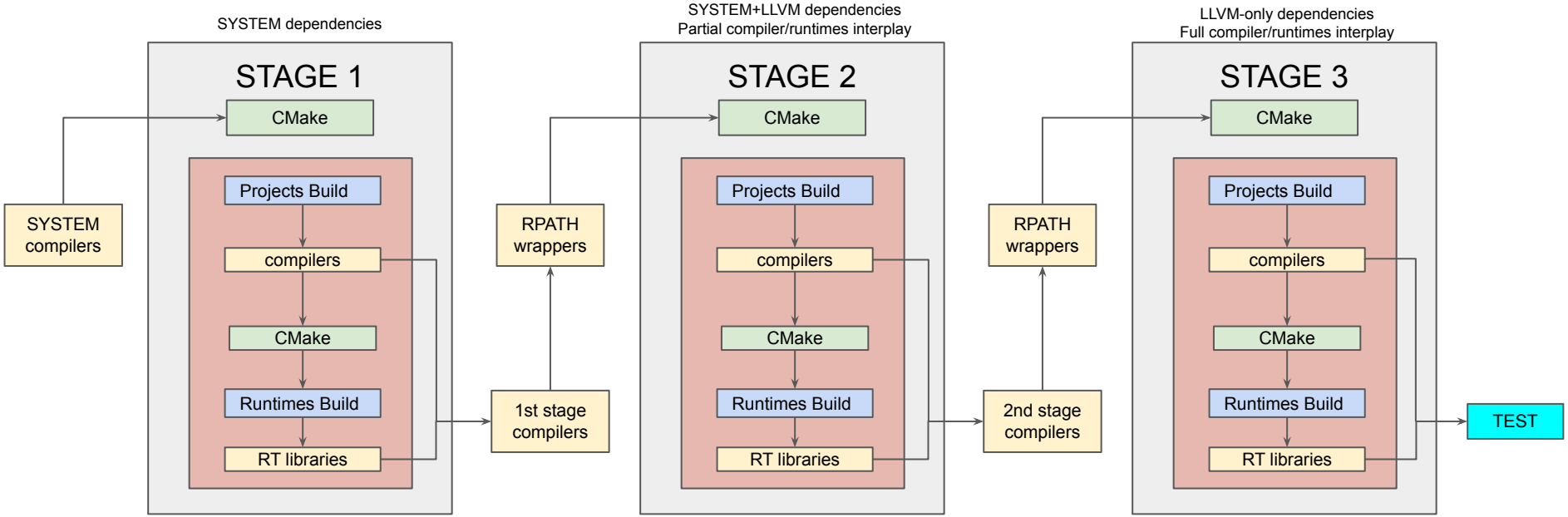
Particular thanks

- @Thyre: Many insightful discussion and very active development and testing of the toolchain
- @Flamefire: Many contribution to the easyblock logic and testing
- @julianmorillo: Add support for building LLVM on RISC-V
- Easybuild maintainers

Thanks for your attention. Questions?

LLVM in EasyBuild - EasyBlock Challenges

- RPATHING (multi-stage build)
  - Easybuild creates wrappers scripts for every compiler that takes the content of LD_LIBRARY_PATH and transform its to a sequence of -Wl,-rpath=XXX options



LLVM - Workflow example - Flang + OpenMP

- Fortran parser
- MLIR dialects
  - HL-FIR
  - FIR
  - OpenMP
- LLVM IR + OpenMP code-generation