libc++ <experimental/simd>

Introduction and Progress

Yin Zhang, 2023.08.19

Outline

- **Introduction**
  - Synopsis
  - Documentations and Publications
  - Features
  - How to Use
- **Progress**
  - libstdc++ <experimental/simd>
  - libc++ <experimental/simd>

Outline

- **Introduction**
    - Synopsis
    - Documentations and Publications
    - Features
    - How to Use
- **Progress**
    - libstdc++ <experimental/simd>
    - libc++ <experimental/simd>

Synopsis

- The *<experimental/simd>* header in the C++ Standard Library was introduced to provide experimental support for SIMD operations directly within C++.
- It aimed to provide a higher-level abstraction for SIMD operations, making it easier for developers to write SIMD code without dealing with low-level assembly instructions.
- It provided portable, zero-overhead C++ types for explicitly data-parallel programming.
- std::experimental::simd is going to move on to std::simd for C++26.

Outline

- **Introduction**
  - Synopsis
  - Documentations and Publications
  - Features
  - How to Use
- **Progress**
  - libstdc++ <experimental/simd>
  - libc++ <experimental/simd>

Documentations and Publications

Publications

- M. Kretz, "Extending C++ for Explicit Data-Parallel Programming via SIMD Vector Types", Goethe University Frankfurt, Dissertation, 2015.
- P. Esterie, M. Gaunard, J. Falcou and J. Lapresté, "Exploiting Multimedia Extensions in C++: A Portable Approach," in Computing in Science & Engineering, vol. 14, no. 5, pp. 72-77, Sept.-Oct. 2012.
- M. Kretz and V. Lindenstruth, "Vc: A C++ library for explicit vectorization", Software: Practice and Experience, 2011.
- J. Falcou and J. Serot, "E.V.E., An Object Oriented SIMD Library.", Scalable Computing: Practice and Experience, vol. 6, no. 4, pp. 72-77, 2005.

Documentations and Publications

Documentations

- https://en.cppreference.com/w/cpp/experimental/simd
- https://github.com/cplusplus/parallelism-ts
- https://wg21.link/P1928
- https://wg21.link/N4808

Outline

- **Introduction**
  - Synopsis
  - Documentations and Publications
  - Features
  - How to Use
- **Progress**
  - libstdc++ <experimental/simd>
  - libc++ <experimental/simd>

Features

Header only

- #include <experimental/simd>
- Easy to use

Features

Abstraction

- *<experimental/simd>* provide a higher-level abstraction that allows developers to write SIMD code without dealing directly with low-level assembly instructions. This makes SIMD programming more accessible to a wider range of developers.

Features

Portability

- SIMD support varies across different hardware architectures and compilers. *<experimental/simd>* can abstract away some of these differences, allowing developers to write SIMD code that works across multiple platforms without having to write and maintain separate code paths for each platform.
- X86: SSE, AVX, AVX512
- ARM: NEON
- PowerPC: Altivec, VSX
- RISC-V: V-Extension
- ……

Features

Simd/Mask types

- template <class Tp, class Abi>

  class simd;

- template <class Tp, class Abi>

  class simd_mask;

- template <class Mask, class Simd>

  class where_expression;

Features

Supported element types and ABI tags

- Element types (vectorizable types): vectorizable types for a data-parallel type comprises all cv-unqualified arithmetic types other than bool.

- ABI tags:

    - scalar

    - fixed_size<N>

    - native<Tp>

    - compatible<Tp>

    - deduced_t<Tp, N, … Abis>

Features

Operations - 1

- simd subscript operators:
    - [ ]
- simd unary operators
    - +, -, ++, --, !, ~
- simd binary operators
    - +, -, *, /, %, &, |, ^, <<, >>

Features

Operations - 2

- simd compound assignment
    - +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=
- simd compare operators
    - ==, !=, >=, <=, >, <

Features

Operations - 3

- Reductions
  - reduce (plus<>, multiplies<>, bit_and<>, bit_or<>, bit_xor<>)
  - hmin, hmax
- Casts
  - simd_cast, static_simd_cast, to_fixed_size, to_native, to_compatible
  - split, resize_simd

Features

Operations - 4

- Algorithms
    - min, max, minmax, clamp

Features

Math Library

- acos, asin, atan, cos, sin, tan, ……
- exp, log, log2, log10, ……
- abs, pow, sqrt, ……
- ceil, floor, round, ……

Outline

- **Introduction**
  - Synopsis
  - Documentations and Publications
  - Features
  - How to Use
- **Progress**
  - libstdc++ <experimental/simd>
  - libc++ <experimental/simd>

How to Use

- https://godbolt.org/z/dPWe5e4xf

Outline

- **Introduction**
  - Synopsis
  - Documentations and Publications
  - Features
  - How to Use
- **Progress**
  - libstdc++ <experimental/simd>
  - libc++ <experimental/simd>

Differences between libstdc++ and libc++

- libstdc++ and libc++ are both C++ standard libraries, but they are associated with different C++ compiler implementations.
- Compiler Associations:
    - libstdc++: This is the C++ standard library that comes with the GNU Compiler Collection (GCC).
    - libc++: This is the C++ standard library developed by the LLVM project, primarily associated with the Clang compiler.

libstdc++ <experimental/simd>

Current status

- Implemented based on N4808
- https://github.com/VcDevel/std-simd
- https://gcc.gnu.org/git/?p=gcc.git;a=blob;f=libstdc%2B%2B-v3/include/experimental/simd;hb=HEAD

Outline

- **Introduction**
  - Synopsis
  - Documentations and Publications
  - Features
  - How to Use
- **Progress**
  - libstdc++ <experimental/simd>
  - libc++ <experimental/simd>

libc++ <experimental/simd>

Current status - 1

- An incomplete implementation in the main branch of llvm-project:
    - https://github.com/llvm/llvm-project/blob/main/libcxx/include/experimental/simd
- The implementation of this version comes from Tim Shen (timshen91@gmail.com)

libc++ <experimental/simd>

Current status - 2

- A rough initial implementation (without math library) :
  - https://github.com/plctlab/llvm-project/tree/simd_for_upstream
- A Single Header Library of libcxx simd:
  - https://github.com/plctlab/simd_for_godbolt

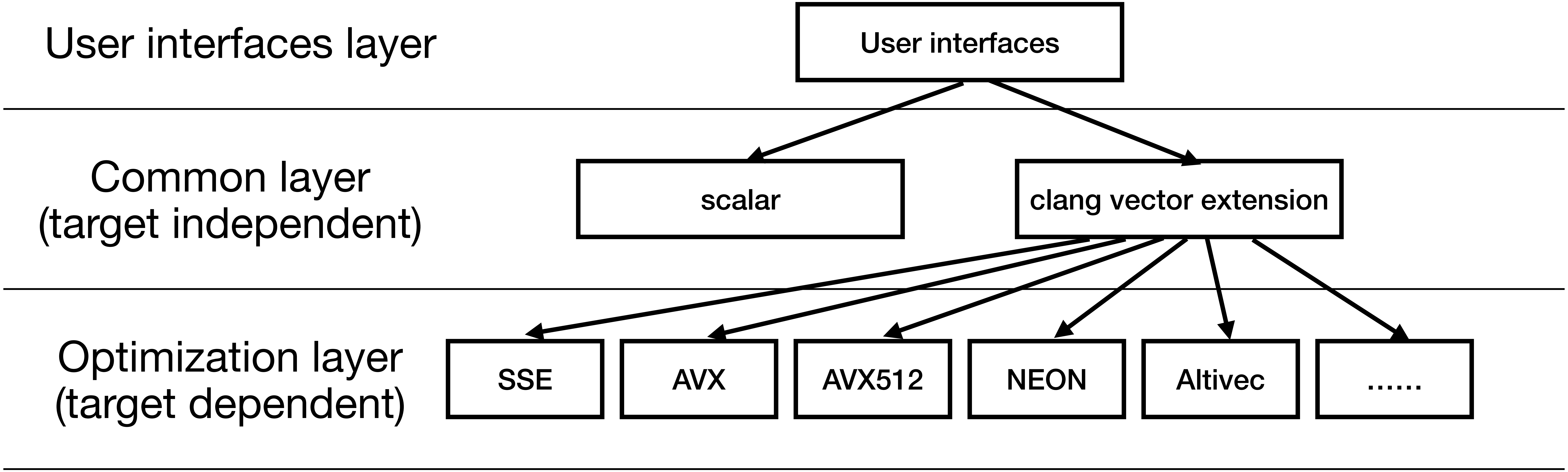
libc++ <experimental/simd>

Current status - 2

- Submitting upstream:
    - https://reviews.llvm.org/D144698
    - https://reviews.llvm.org/D144362
    - https://reviews.llvm.org/D144363
    - https://reviews.llvm.org/D153319
    - https://reviews.llvm.org/D144364
    - https://reviews.llvm.org/D156225

libc++ <experimental/simd>

Implementation architecture



User interfaces layer

User interfaces

Common layer (target independent)

scalar

clang vector extension

Optimization layer (target dependent)

SSE

AVX

AVX512

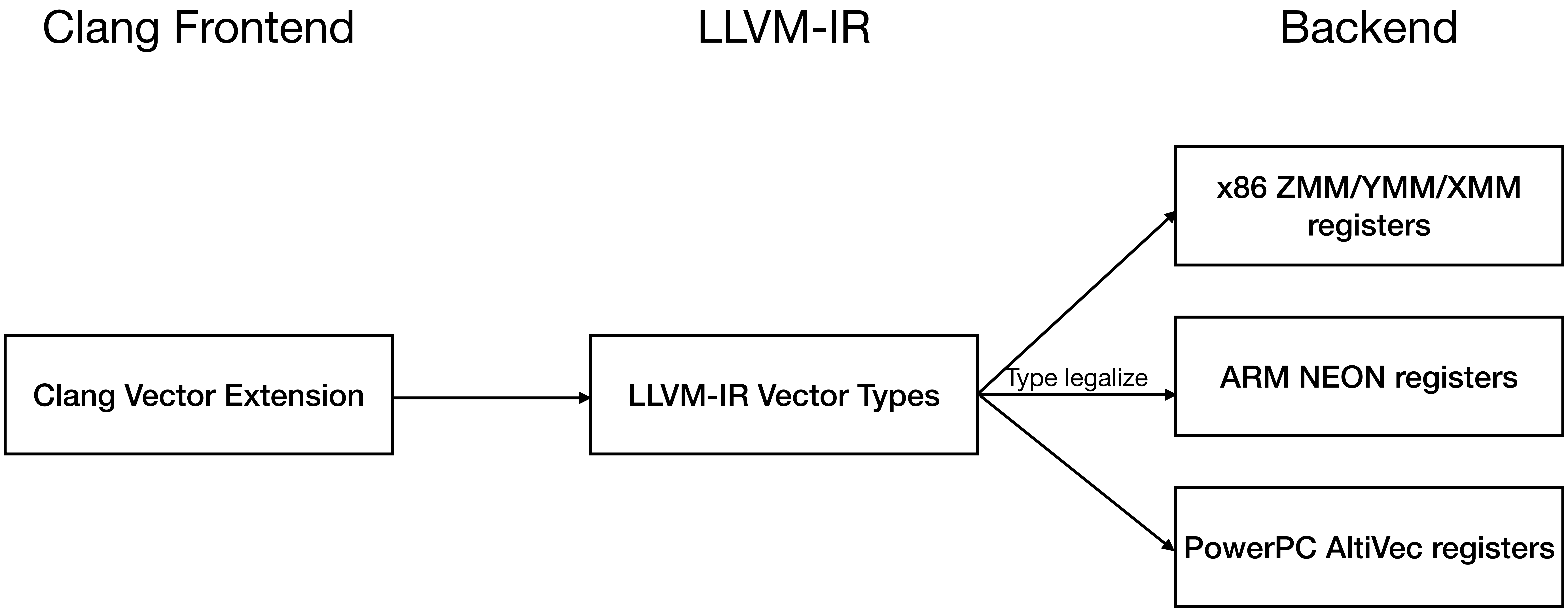
NEON

Altivec

......

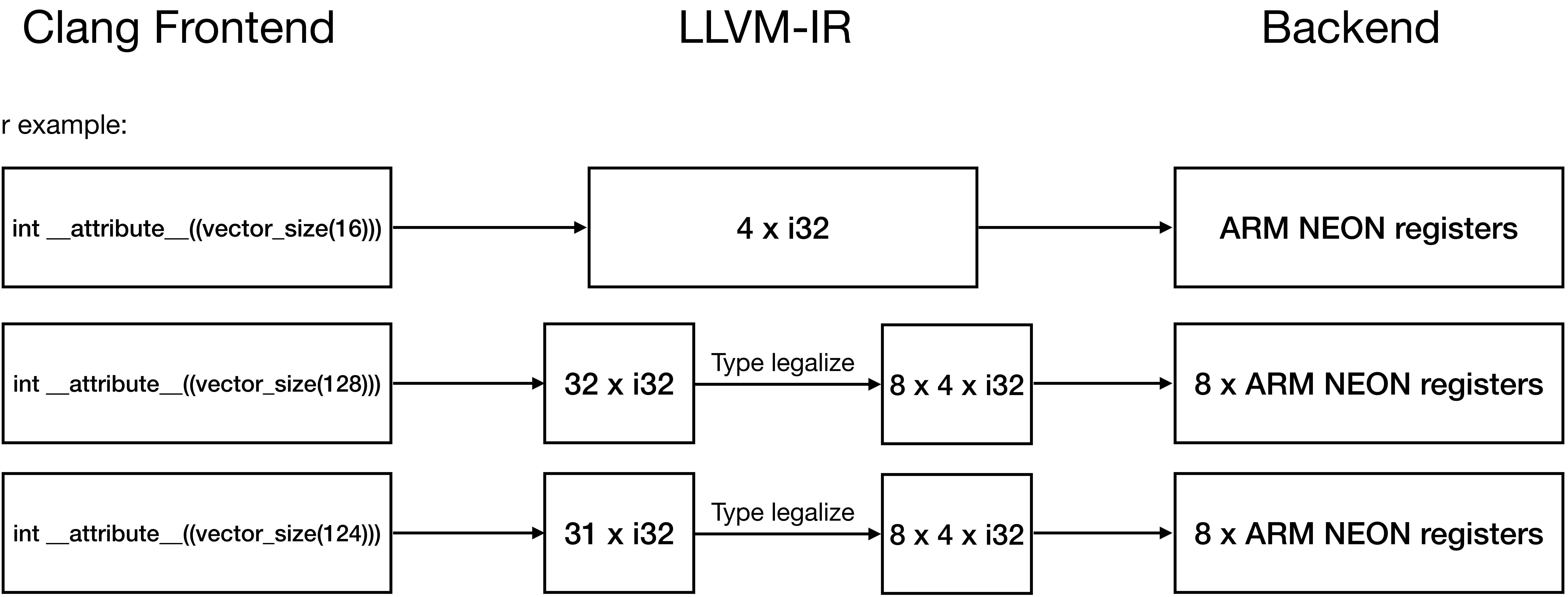
Clang Vector Extension

Vector types



Clang Vector Extension

Vector types example



Clang Frontend

LLVM-IR

Backend

For example:

int __attribute__((vector_size(16)))

4 x i32

ARM NEON registers

int __attribute__((vector_size(128)))

32 x i32

Type legalize

8 x 4 x i32

8 x ARM NEON registers

int __attribute__((vector_size(124)))

31 x i32

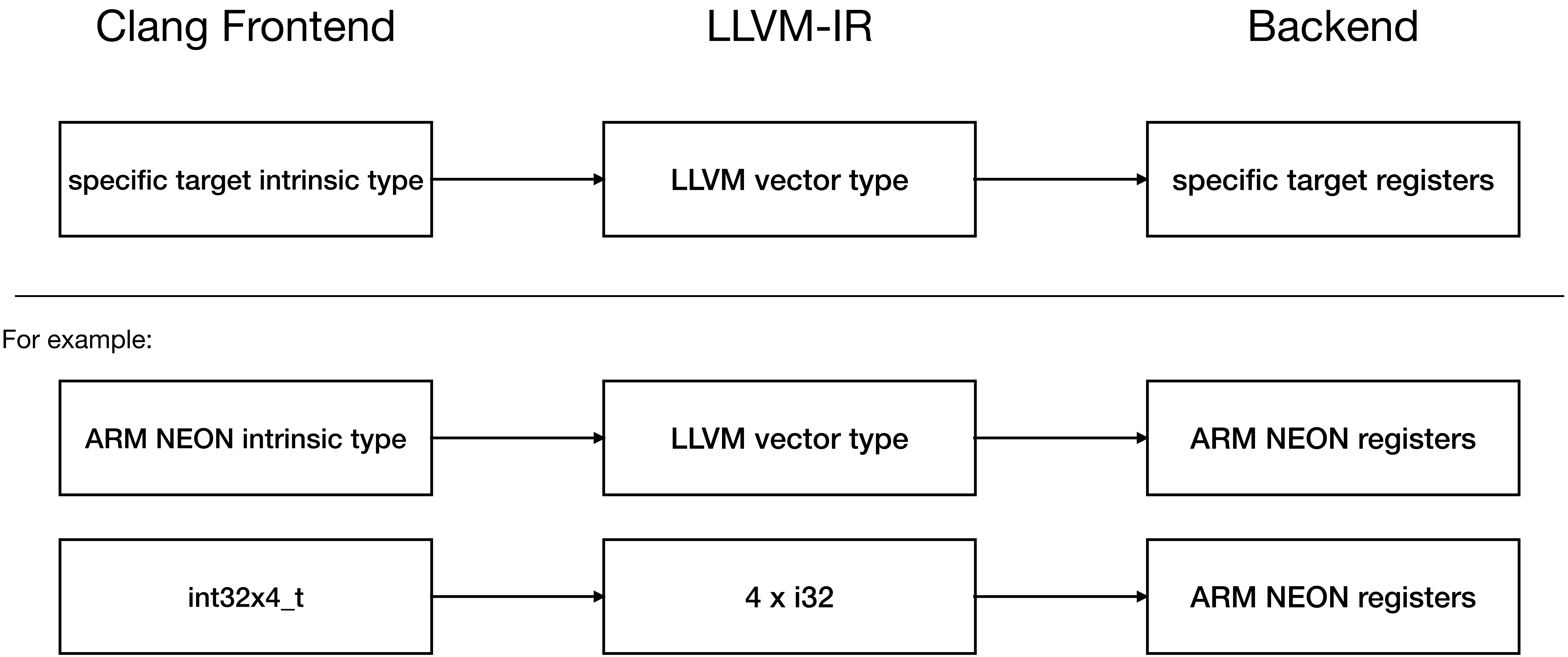
Type legalize

8 x 4 x i32

8 x ARM NEON registers

intrinsic type

Vector types example



Clang Frontend

LLVM-IR

Backend

specific target intrinsic type → LLVM vector type → specific target registers

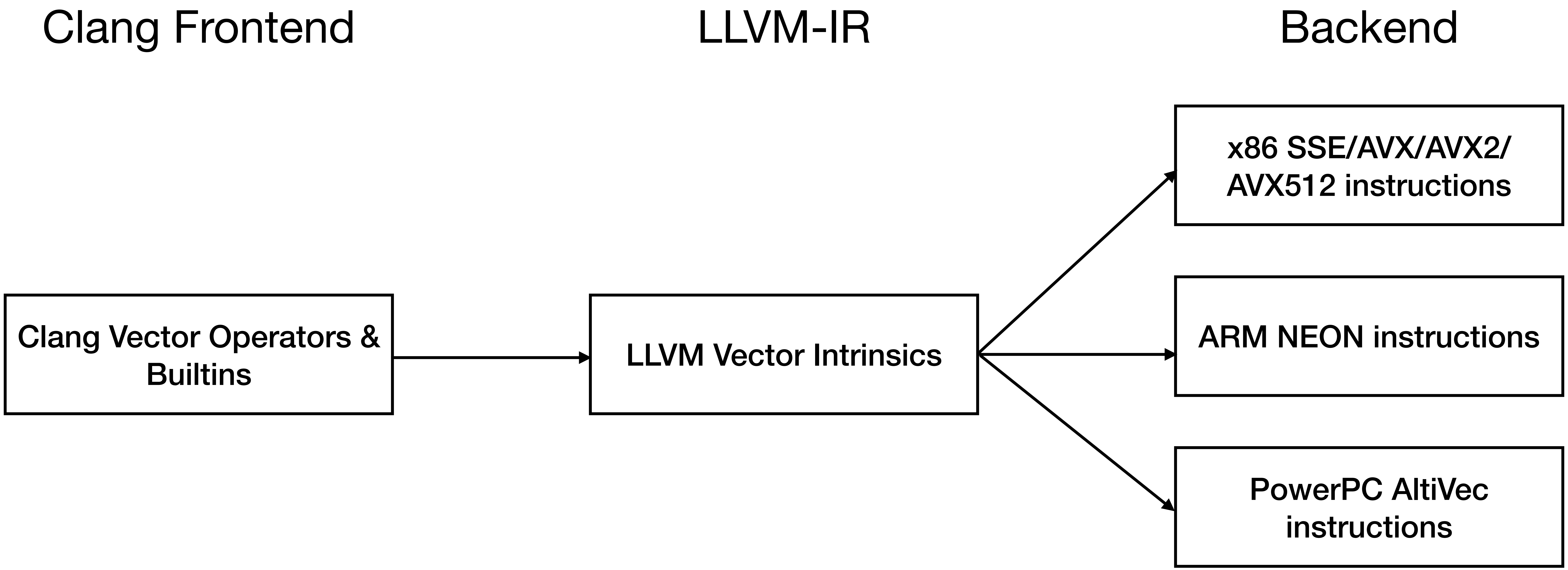
For example:

ARM NEON intrinsic type → LLVM vector type → ARM NEON registers

int32x4_t → 4 x i32 → ARM NEON registers

Vector Operations

Clang Vector Operators & Builtins



Clang Frontend

LLVM-IR

Backend

Clang Vector Operators & Builtins

LLVM Vector Intrinsics

x86 SSE/AVX/AVX2/ AVX512 instructions

ARM NEON instructions

PowerPC AltiVec instructions

https://clang.llvm.org/docs/LanguageExtensions.html#vectors-and-extended-vectors

libc++ <experimental/simd>

Tests

- There are a total of 43 test files and 1 test framework file that fully cover all implemented external user interfaces.
- Test each combination of vectorizable types and ABI tags through multi-layer template nested. Fully improved testing coverage.
- An experimental implementation on OpenCV:
  - https://github.com/plctlab/opencv/pull/1
- Benchmarks working in progress.

libc++ <experimental/simd>

Contributors

Yin Zhang - zhangyin2018@iscas.ac.cn

Yiliang He - QuarticCat@protonmail.com

Yi Zhang - zhangyi216@mails.ucas.ac.cn

Haolin Pan - panhaolin21@mails.ucas.ac.cn

Jiatai He - jiatai2021@iscas.ac.cn

Heda Chen - marcythm@gmail.com

Haichuan Hu - huhaichuan0704@126.com

Haohang Shi - shyhot@outlook.com

PLCT Lab, Intelligent Software Research Center

Institute of Software Chinese Academy of Sciences

Thanks!