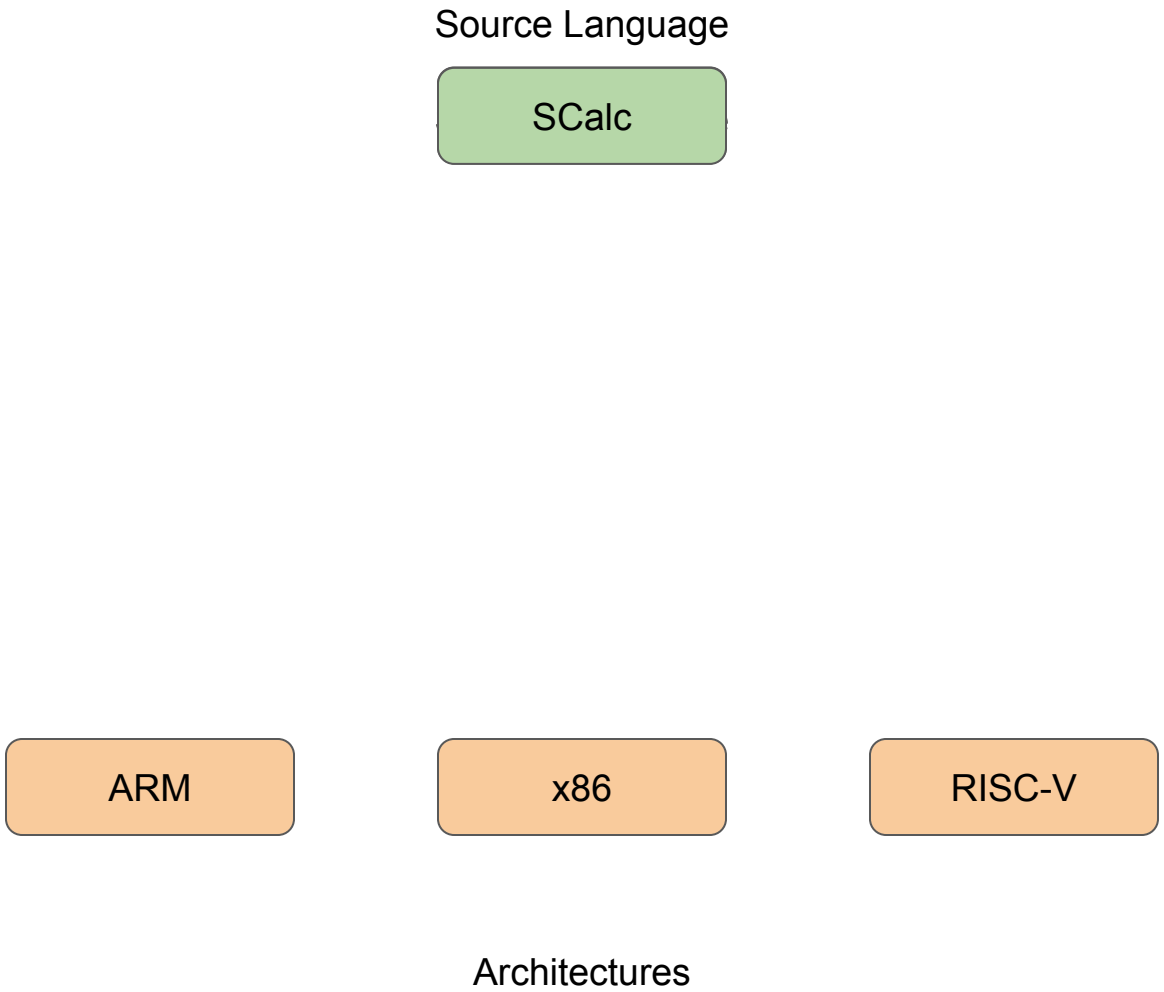
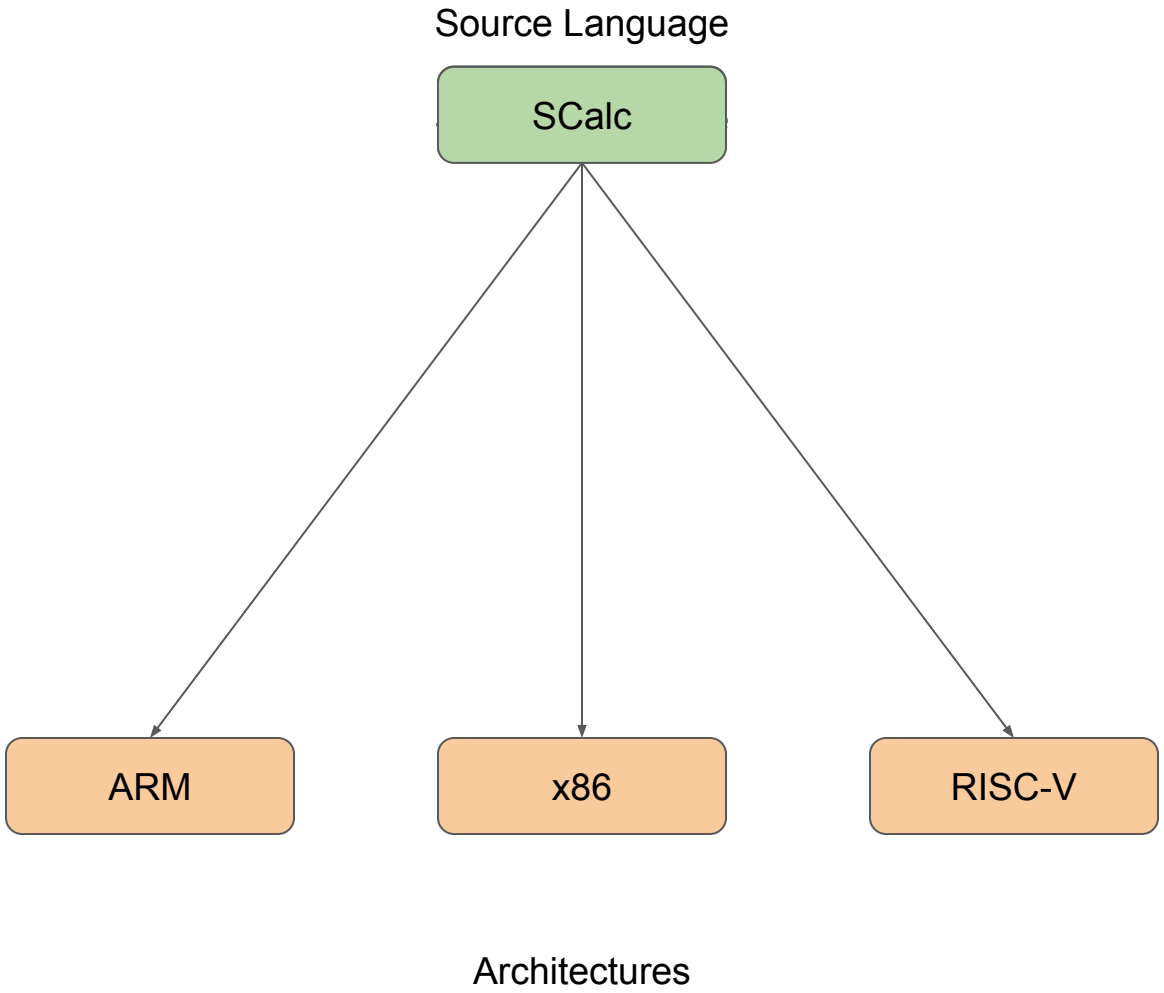
What is LLVM (and MLIR)?

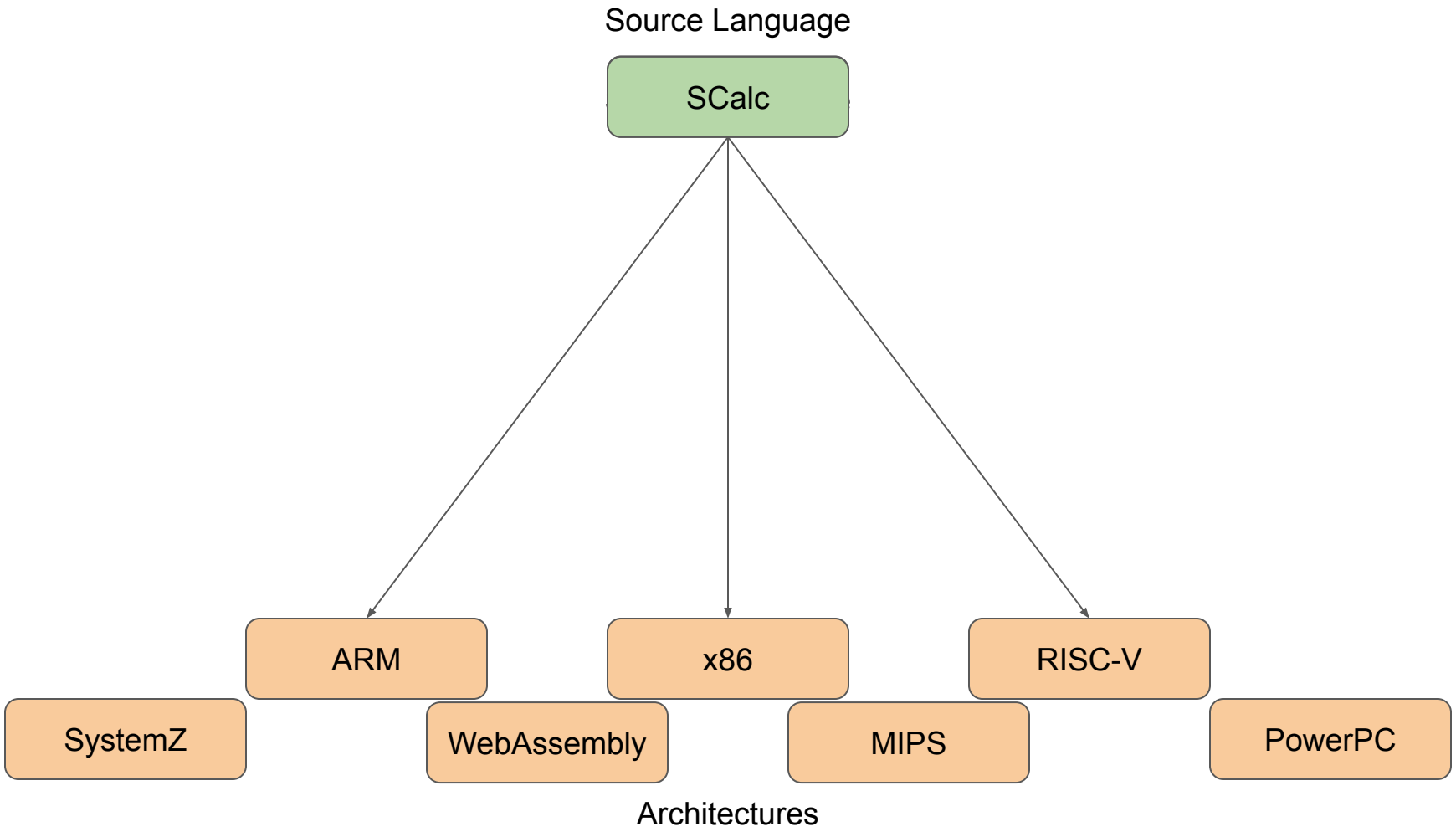
Quinn Pham
Adapted from slides by Braedy Kuzma, Caio Salvador Rohwedder, and Nelson Amaral

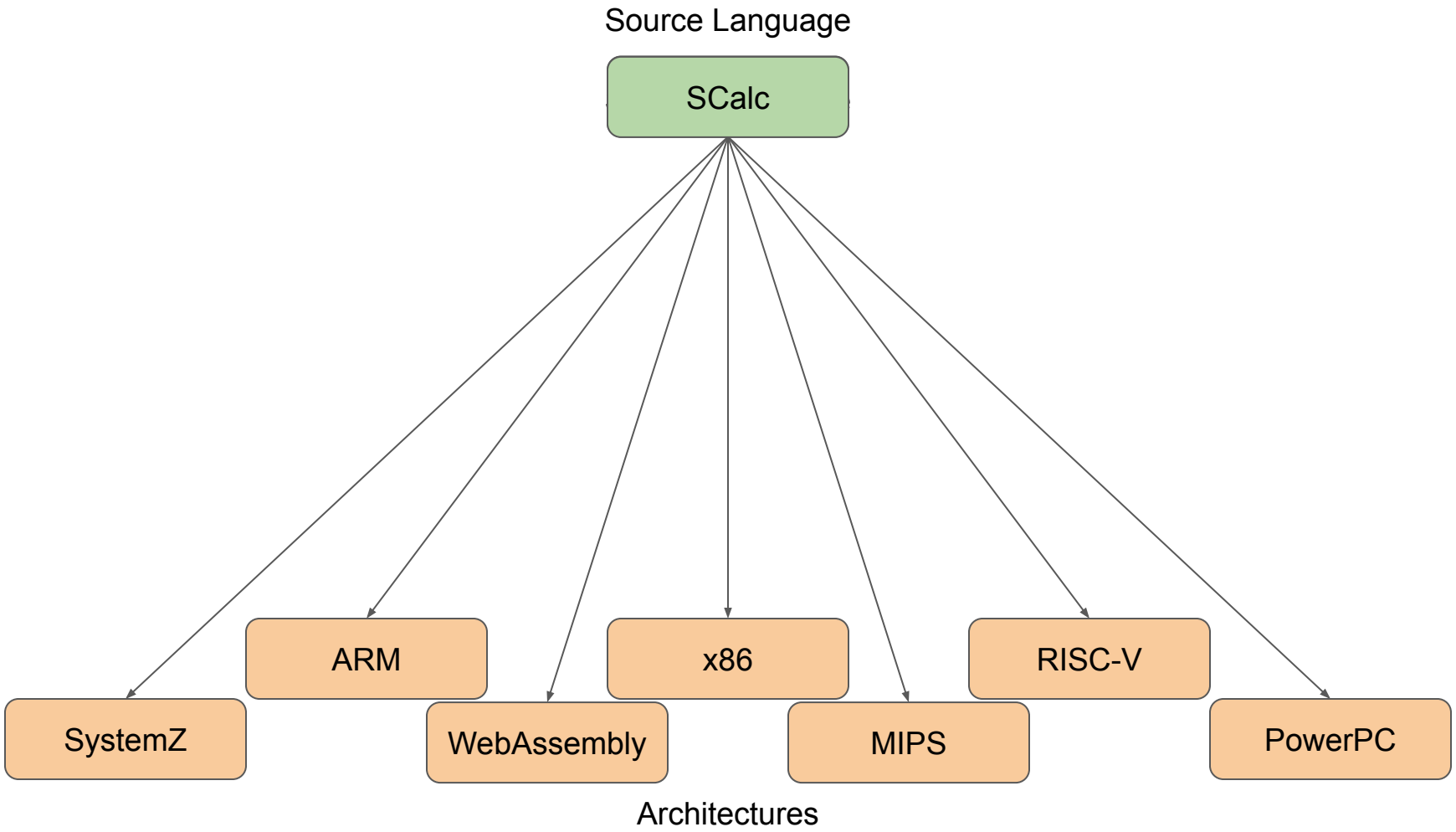
Your Life Right Now

SCalc


Source Language
SCalc
ARM
x86
RISC-V
Architectures


Source Language
SCalc
ARM
x86
RISC-V
Architectures


Source Language
SCalc
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures


Source Language
SCalc
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures

Source Languages

Swift

Julia

C

C++

Rust

ARM

x86

RISC-V

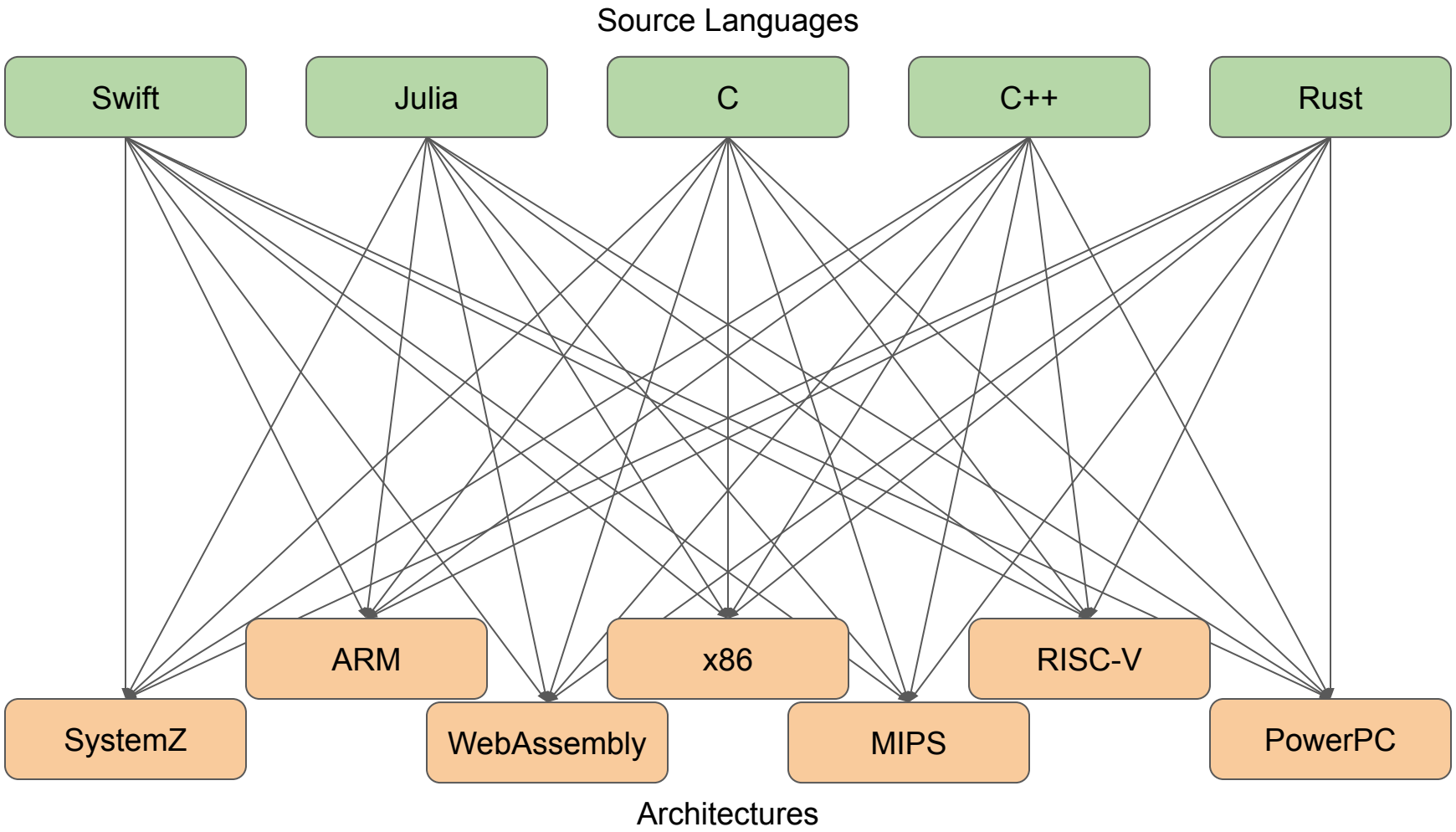
SystemZ

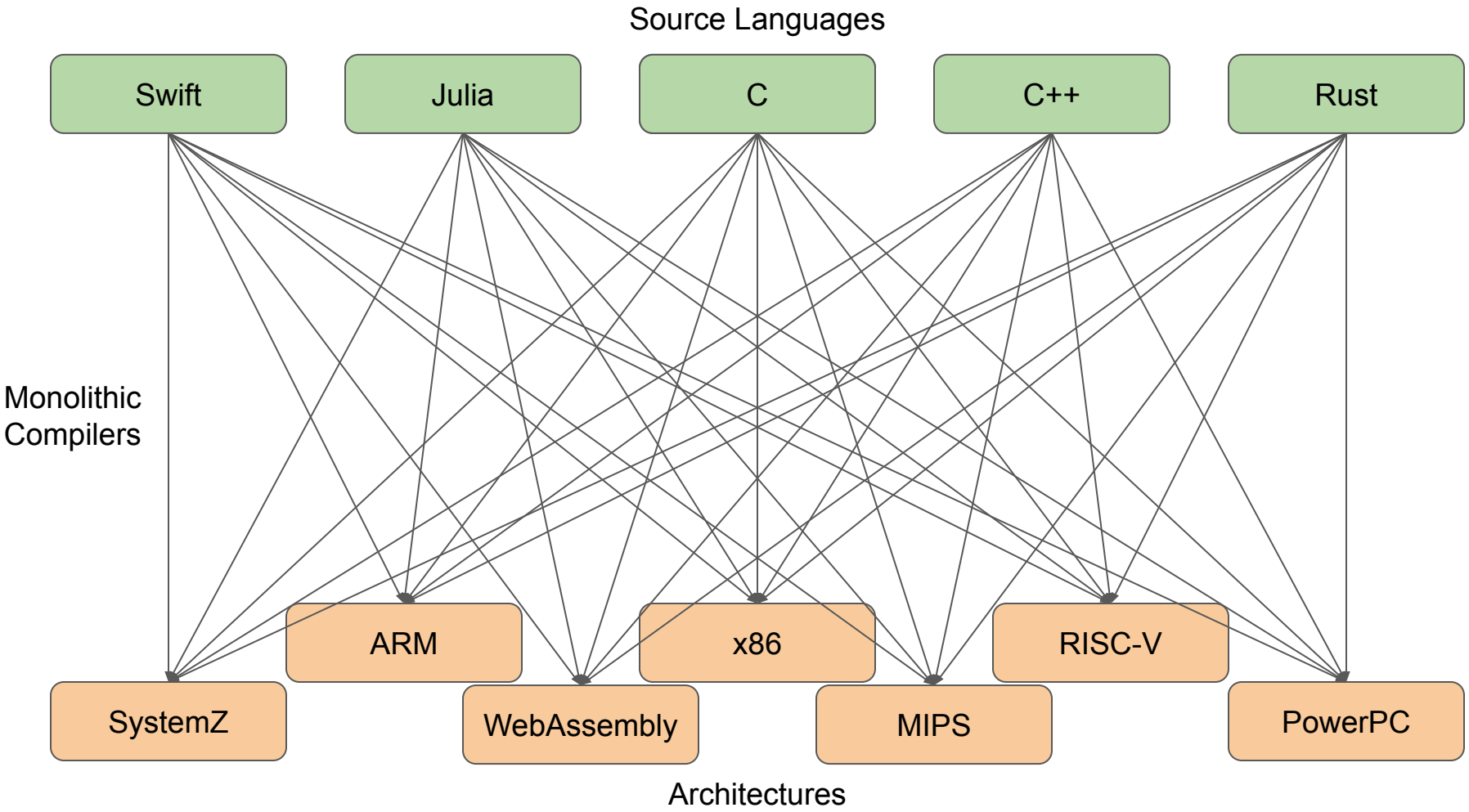
WebAssembly

MIPS

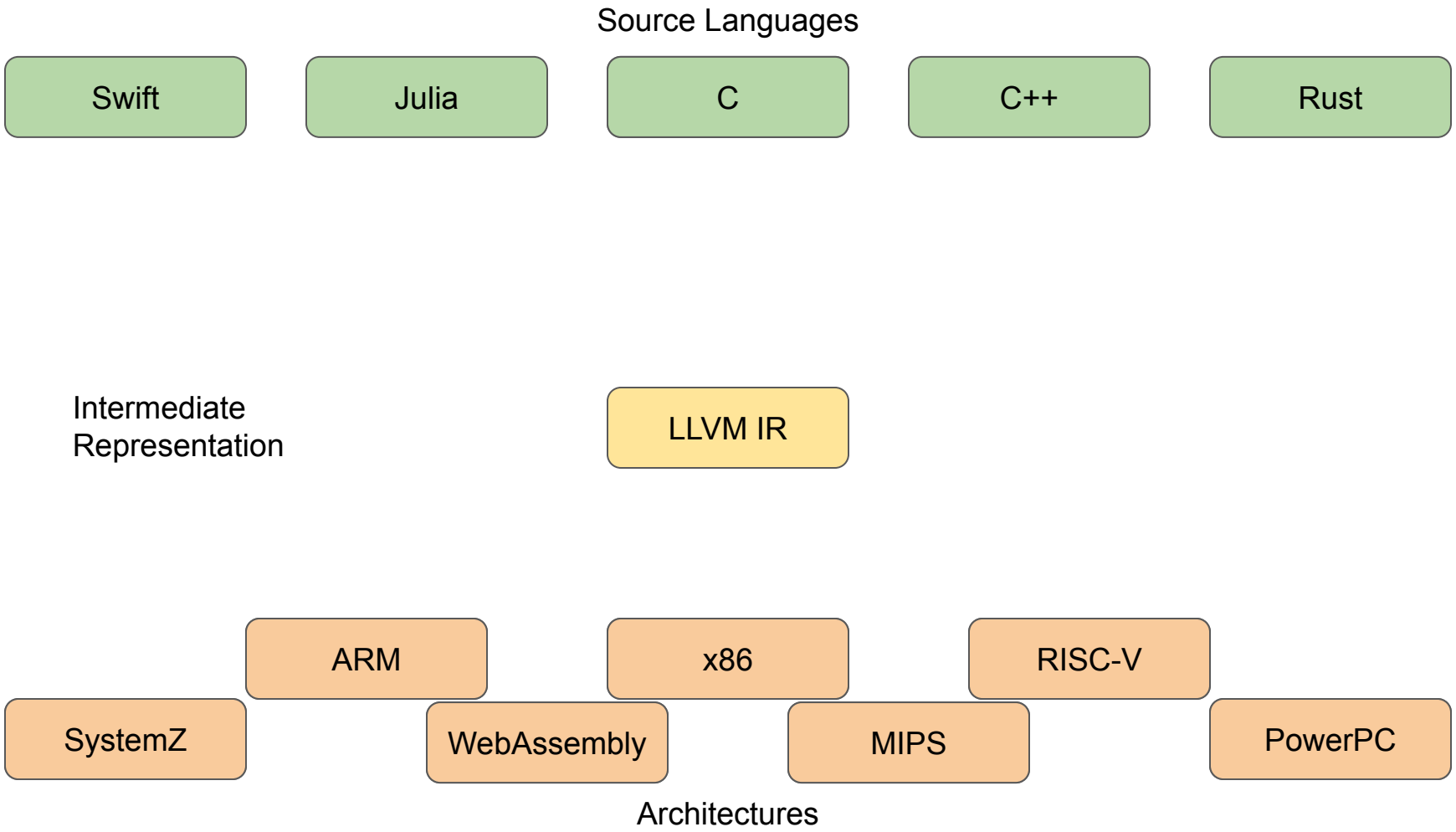
PowerPC

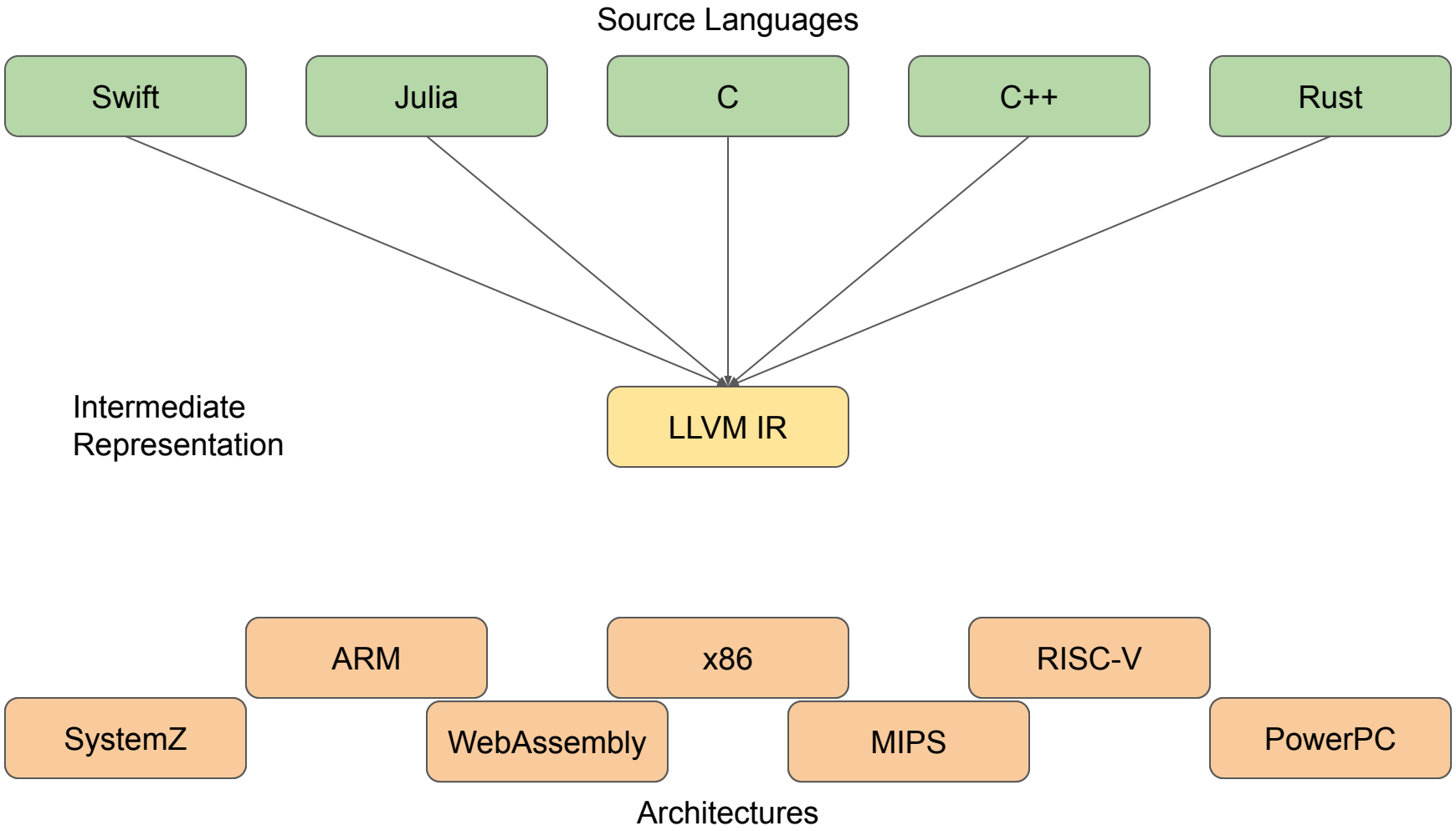
Architectures

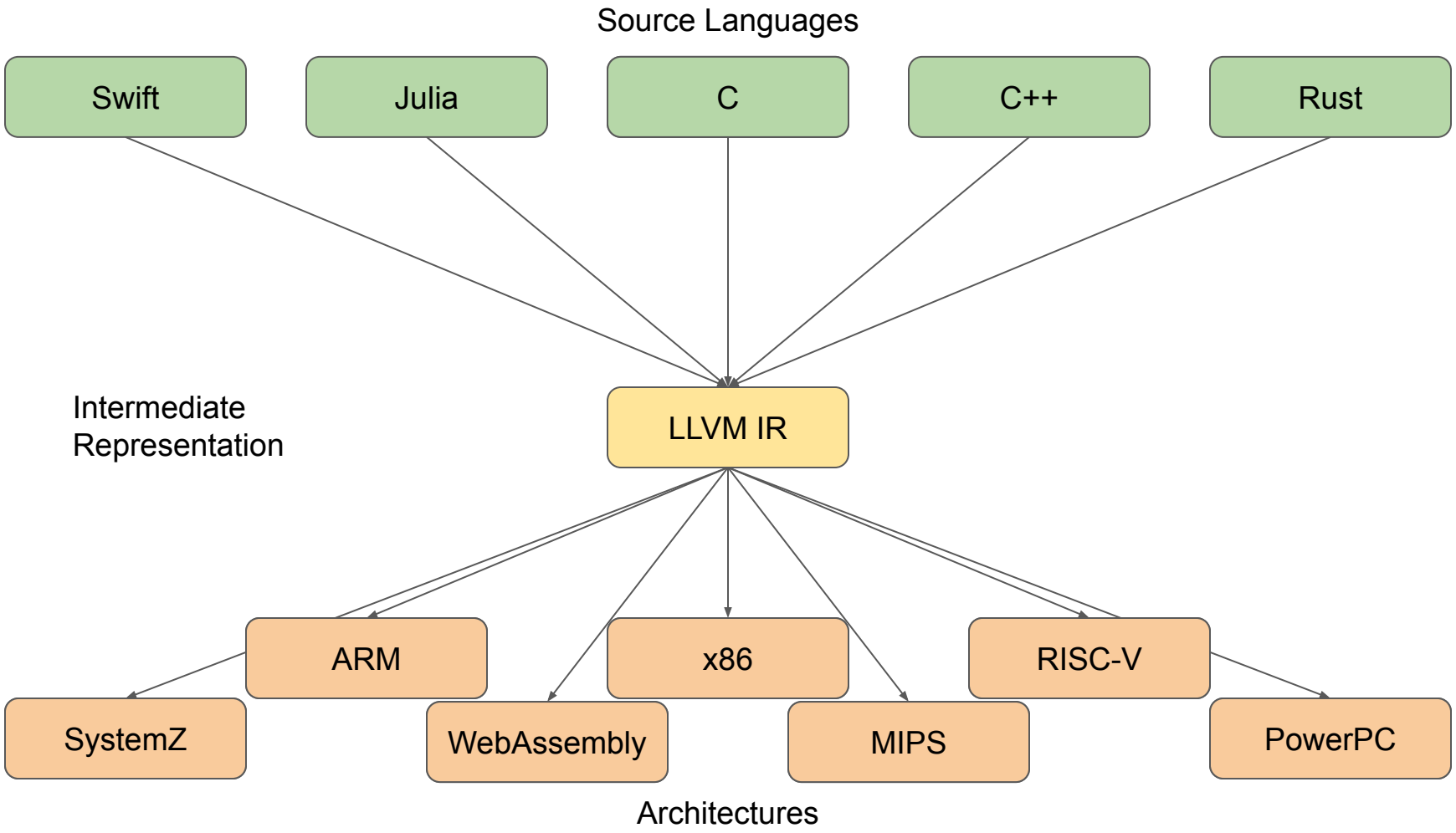

Source Languages
Swift
Julia
C
C++
Rust
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures

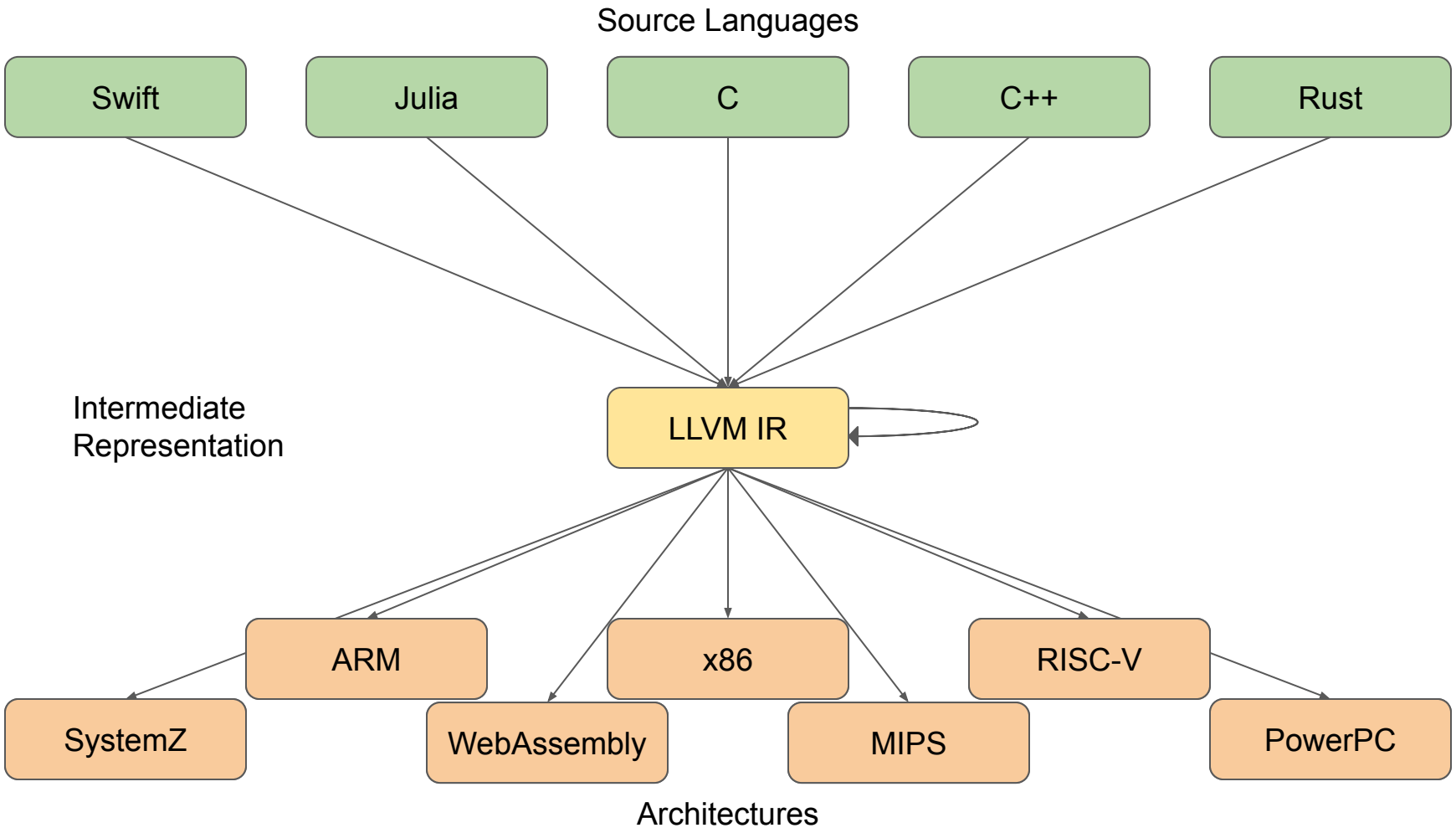

Source Languages
Swift
Julia
C
C++
Rust
Monolithic Compilers
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures

LLVM

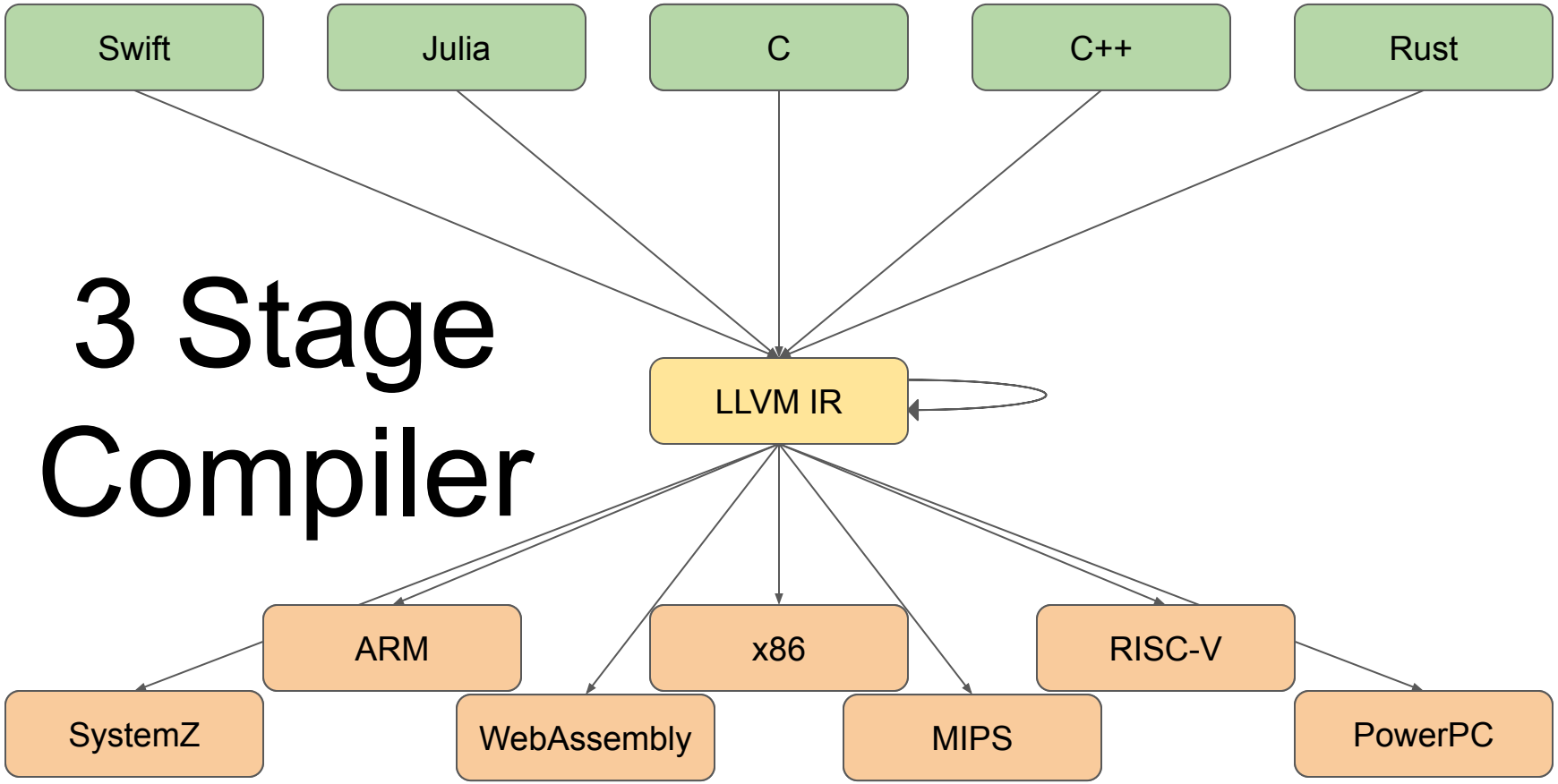

Source Languages
Swift
Julia
C
C++
Rust
Intermediate Representation
LLVM IR
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures

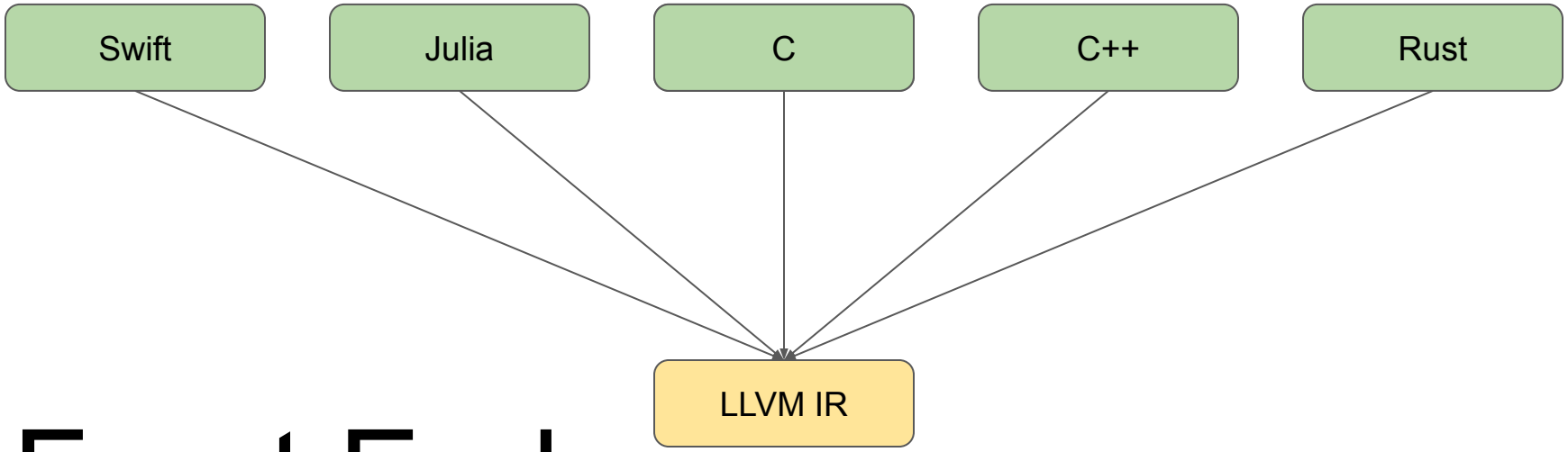

Source Languages
Swift
Julia
C
C++
Rust
Intermediate Representation
LLVM IR
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures


Source Languages
Swift
Julia
C
C++
Rust
Intermediate Representation
LLVM IR
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures

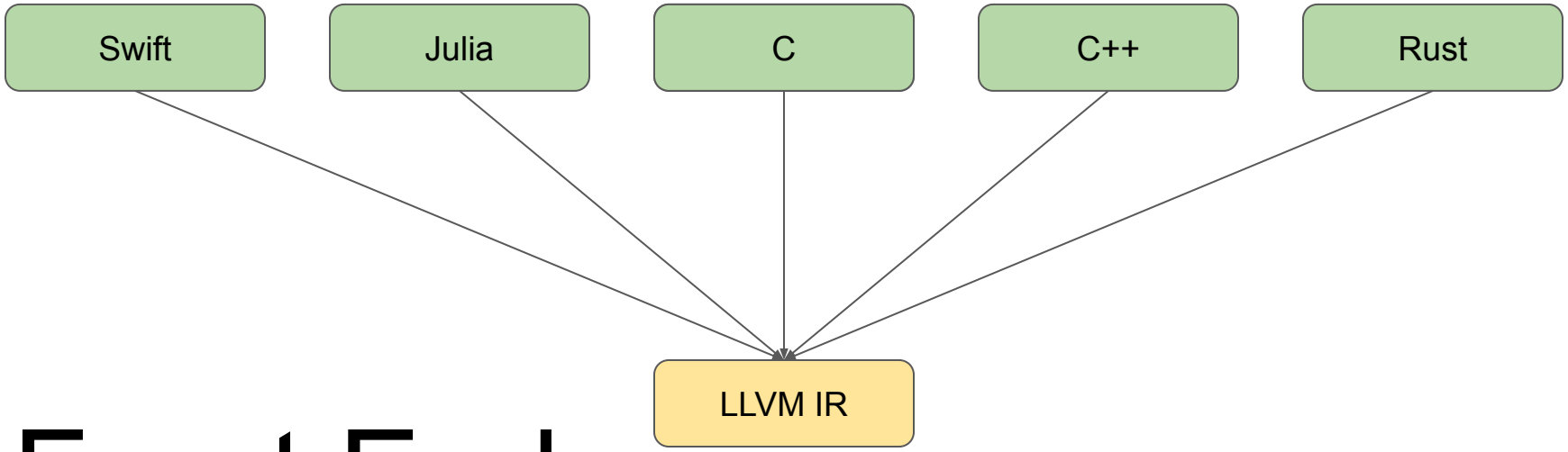

Source Languages
Swift
Julia
C
C++
Rust
Intermediate Representation
LLVM IR
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC
Architectures

3 Stage Compiler





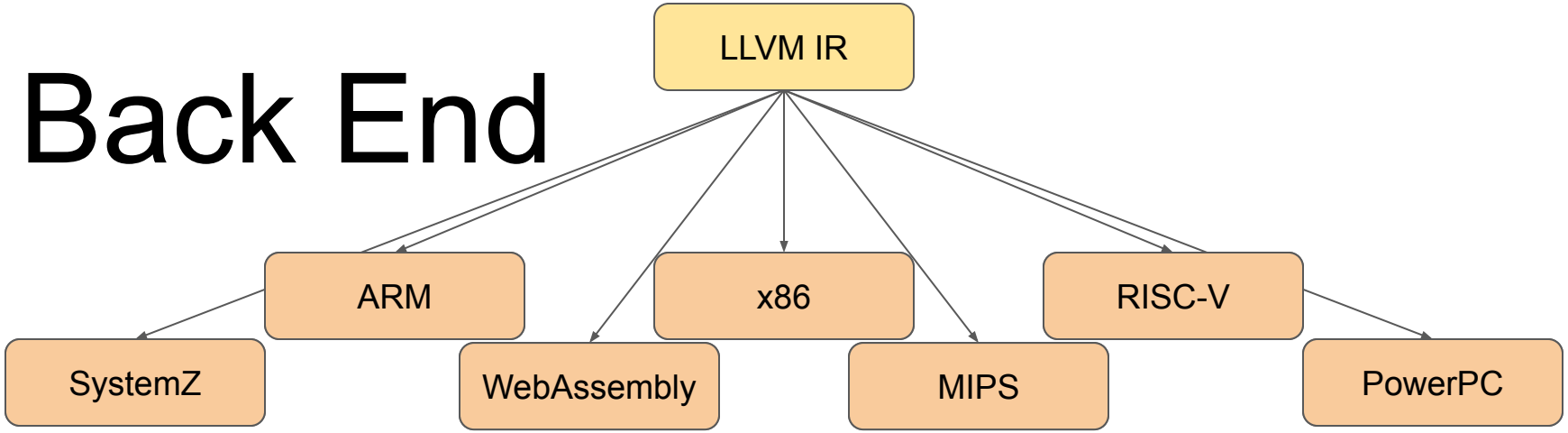
Front End



Front End

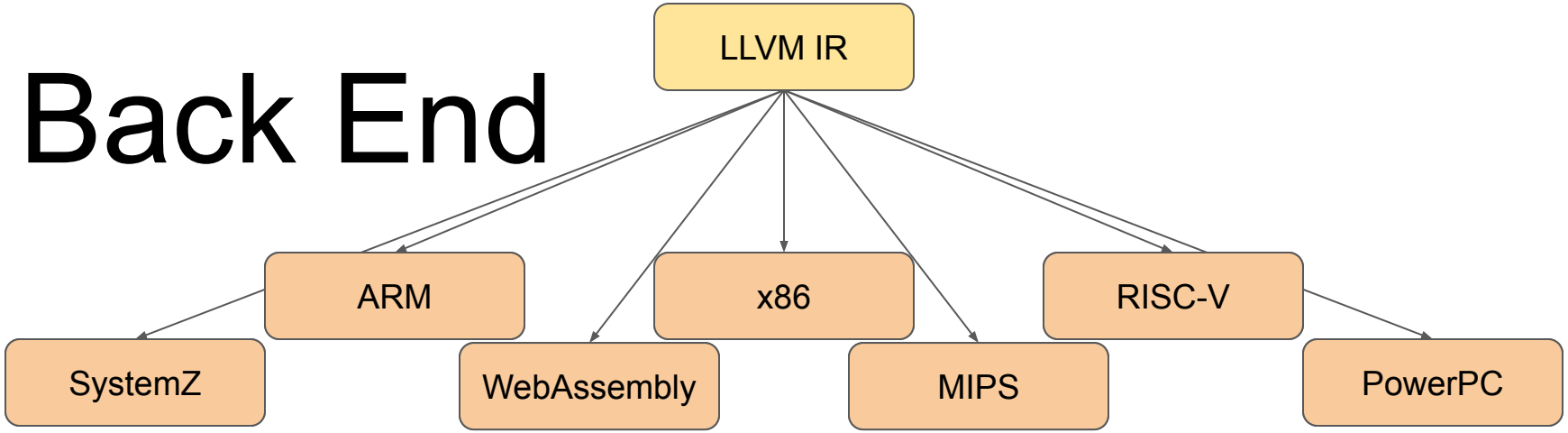
Analyzes the source code and builds the intermediate representation for the program.

Back End



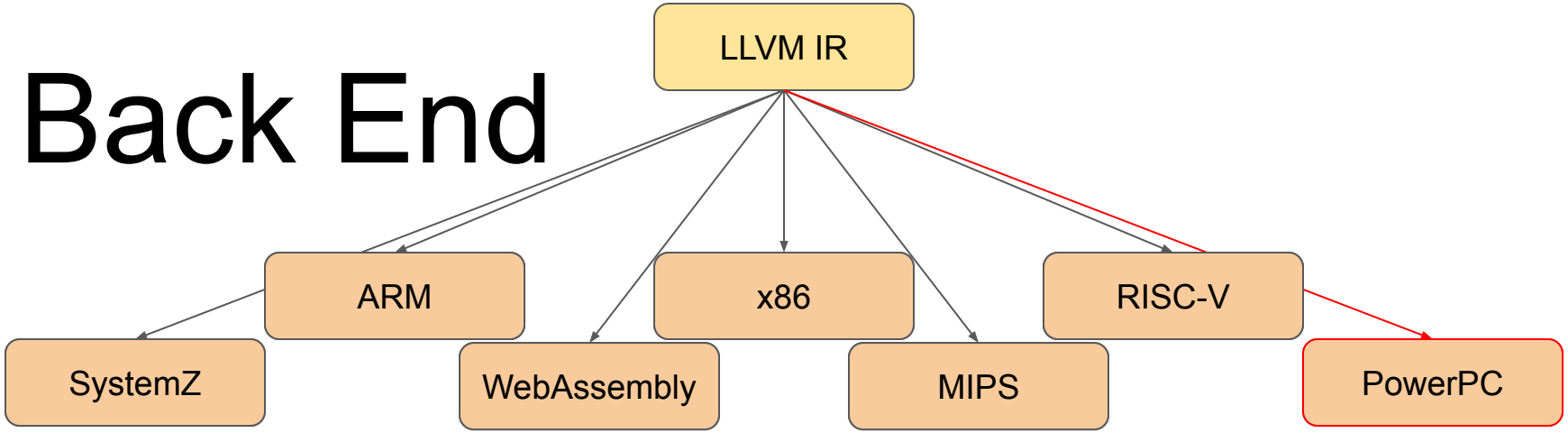
Performs architecture specific optimizations and generates assembly code for the program.

Back End



Performs architecture specific optimizations and generates assembly code for the program.

Back End

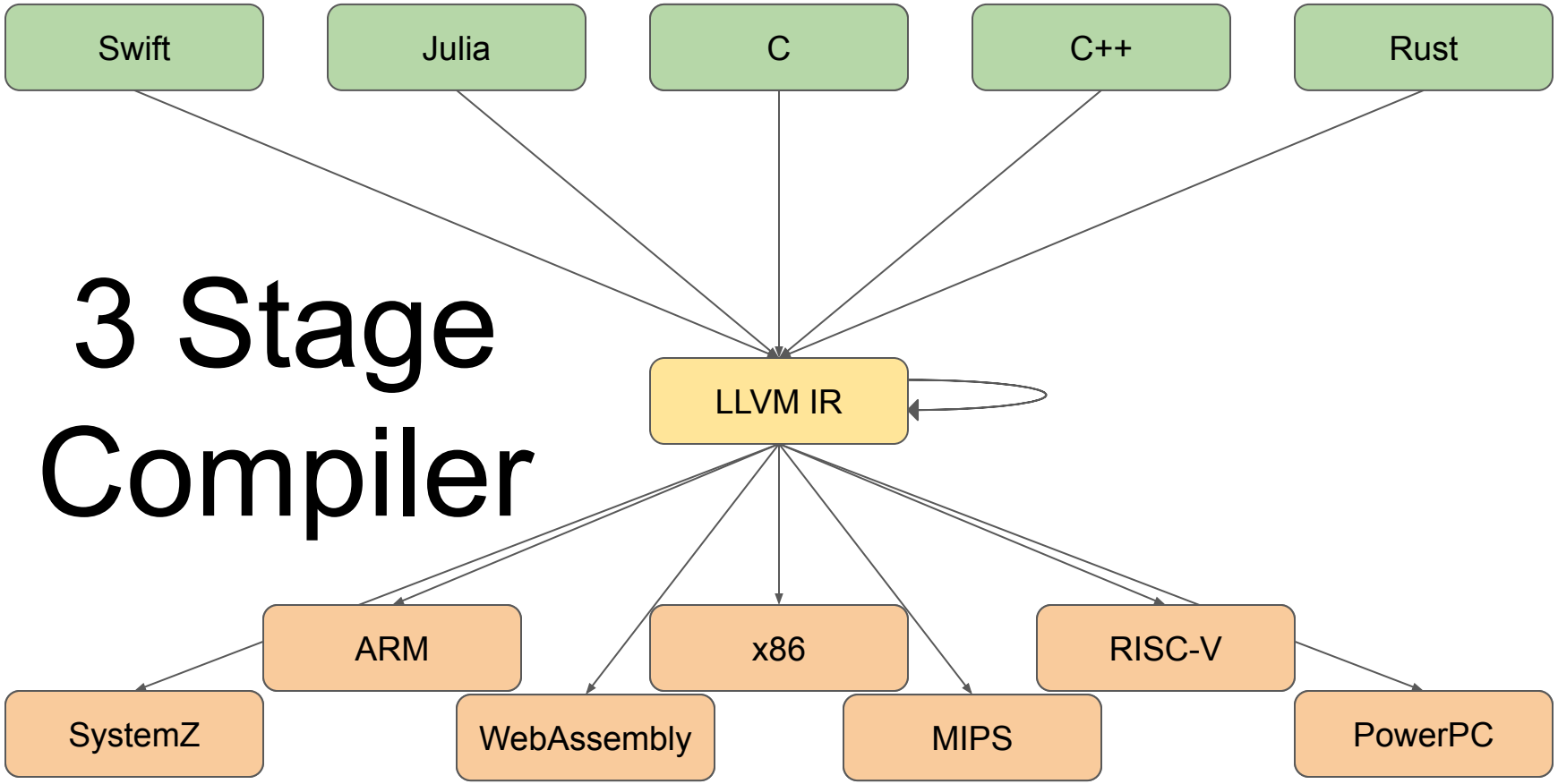


Middle End

Middle End

Performs analyses and transformations on the intermediate representation.

3 Stage Compiler



LLVM

LLVM

- Used to stand for Low Level Virtual Machine
  - No longer has meaning as an initialism

LLVM

- Used to stand for Low Level Virtual Machine
  - No longer has meaning as an initialism
- Collection of modular and reusable compiler and toolchain technologies

LLVM

- Used to stand for Low Level Virtual Machine
  - No longer has meaning as an initialism
- Collection of modular and reusable compiler and toolchain technologies
- “LLVM” can refer to the entire project or the middle & back ends

LLVM

- Used to stand for Low Level Virtual Machine
  - No longer has meaning as an initialism
- Collection of modular and reusable compiler and toolchain technologies
- “LLVM” can refer to the entire project or the middle & back ends
- Open source

llvm / llvm-project

Type / to search

Code | Issues 5k+ | Pull requests 576 | Actions | Security | Insights

llvm-project

Public

Watch 610 | Fork 8.6k | Star 22k

main | 45 branches | 249 tags | Go to file | Add file | Code

dtcxzyw [ValueTracking] Simplify uaddo pattern (#65910) ... 9f2fc88 1 minute ago 476,266 commits

.ci	[ci] buildkite don't escape windows targets (#66192)	2 weeks ago
.github	[Workflow] Update clang-format to 17.0.1 (#67402)	11 hours ago
bolt	[BOLT] Update for rename of MemLifetimePolicy in e994f84.	17 minutes ago
clang-tools-extra	[clangd][CodeComplete] Improve FunctionCanBeCall	4 hours ago
clang	[clang][Parse][NFC] Remove dead if statement	22 minutes ago
cmake	[CMake] Switch the CMP0091 policy (MSVC_RUNTIME_LIBRARY) t...	2 months ago
compiler-rt	[scudo] Use MemMap in BufferPool and RegionPageMap (#66788)	3 hours ago
cross-project-tests	[Dexter] Associate parser errors with correct file (#66765)	last week
flang	[Flang] [OpenMP] [Semantics] Add semantic support for IS_DEVIC...	1 hour ago
libc	[libc] Fix wrapper headers for some ctype macros and C++ decls	3 hours ago
libclc	libclc: Fix signed integer underflow in abs_diff	last month
libcxx	[libc++][NFC] Simplify checks for static assertions in .verify.cpp te...	4 hours ago
libcxxabi	[libc++][lit] Allow overriding the executor for tests (#66545)	2 days ago
libunwind	[libunwind][nfc] Avoid type warning of debug printf (#67390)	9 hours ago
lld	[lld] Fix REQUIRES line in new test	18 hours ago
lldb	[LLDB] Skip TestTlsGlobals.py for Linux Arm/AArch64	6 hours ago
llvm-libgcc	[llvm-libgcc][CMake] Refactor llvm-libgcc (#65455)	last week
llvm	[ValueTracking] Simplify uaddo pattern (#65910)	1 minute ago
mlir	[mlir][SCF] Bufferize scf.index_switch (#67666)	1 hour ago
openmp	[Libomptarget] Fix Nvidia offloading hanging on dataRetrieve using...	2 days ago
polly	Move CallInst::CreateFree to IRBuilderBase	last week
pstl	Clear release notes for 18.x	2 months ago

About

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies.

llvm.org

Readme
View license
Code of conduct
Security policy
Activity
22k stars
610 watching
8.6k forks
Report repository

Releases 91

LLVM 17.0.1 Latest
last week

+ 90 releases

Used by 5

Contributors 3,194

+ 3,183 contributors

https://github.com/llvm/llvm-project

llvm / llvm-project

Type / to search

Code | Issues 5k+ | Pull requests 576 | Actions | Security | Insights

llvm-project

Public

Watch 610 | Fork 8.6k | Star 22k

main | 45 branches | 249 tags | Go to file | Add file | Code

dtcxzyw [ValueTracking] Simplify uaddo pattern (#65910) ... 9f2fc88 1 minute ago 476,266 commits

.ci	[ci] buildkite don't escape windows targets (#66192)	2 weeks ago
.github	[Workflow] Update clang-format to 17.0.1 (#67402)	11 hours ago
bolt	[BOLT] Update for rename of MemLifetimePolicy in e994f84.	17 minutes ago
clang-tools-extra	[clangd][CodeComplete] Improve FunctionCanBeCall	4 hours ago
clang	[clang][Parse][NFC] Remove dead if statement	22 minutes ago
cmake	[CMake] Switch the CMP0091 policy (MSVC_RUNTIME_LIBRARY) t...	2 months ago
compiler-rt	[scudo] Use MemMap in BufferPool and RegionPageMap (#66788)	3 hours ago
cross-project-tests	[Dexter] Associate parser errors with correct file (#66765)	last week
flang	[Flang] [OpenMP] [Semantics] Add semantic support for IS_DEVIC...	1 hour ago
libc	[libc] Fix wrapper headers for some ctype macros and C++ decls	3 hours ago
libclc	libclc: Fix signed integer underflow in abs_diff	last month
libcxx	[libc++][NFC] Simplify checks for static assertions in .verify.cpp te...	4 hours ago
libcxxabi	[libc++][lit] Allow overriding the executor for tests (#66545)	2 days ago
libunwind	[libunwind][nfc] Avoid type warning of debug printf (#67390)	9 hours ago
lld	[lld] Fix REQUIRES line in new test	18 hours ago
lldb	[LLDB] Skip TestTlsGlobals.py for Linux Arm/AArch64	6 hours ago
llvm-libgcc	[llvm-libgcc][CMake] Refactor llvm-libgcc (#65455)	last week
llvm	[ValueTracking] Simplify uaddo pattern (#65910)	1 minute ago
mlir	[mlir][SCF] Bufferize scf.index_switch (#67666)	1 hour ago
openmp	[Libomptarget] Fix Nvidia offloading hanging on dataRetrieve using...	2 days ago
polly	Move CallInst::CreateFree to IRBuilderBase	last week
pstl	Clear release notes for 18.x	2 months ago

About

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies.

llvm.org

Readme
View license
Code of conduct
Security policy
Activity
22k stars
610 watching
8.6k forks
Report repository

Releases 91

LLVM 17.0.1 Latest
last week

+ 90 releases

Used by 5

Contributors 3,194

+ 3,183 contributors

https://github.com/llvm/llvm-project

LLVM Language Reference Manual

https://llvm.org/docs/LangRef.html

LLVM Tools

LLVM Tools

clang - C front end

.c -> .ll | .bc

LLVM Tools

clang - C front end

.c -> .ll | .bc

llvm-dis - disassembler

.bc -> .ll

LLVM Tools

clang - C front end

.c -> .ll | .bc

llvm-dis - disassembler

.bc -> .ll

llvm-as - assembler

.ll -> .bc

LLVM Tools

clang - C front end

.c -> .ll | .bc

llvm-dis - disassembler

.bc -> .ll

llvm-as - assembler

.ll -> .bc

opt - optimizer and analyzer

.ll | .bc -> .ll | .bc

LLVM Tools

clang - C front end

.c -> .ll | .bc

llvm-dis - disassembler

.bc -> .ll

llvm-as - assembler

.ll -> .bc

opt - optimizer and analyzer

.ll | .bc -> .ll | .bc

lli - interpreter

interpret .ll or .bc

LLVM Tools

clang - C front end

.c -> .ll | .bc

llvm-dis - disassembler

.bc -> .ll

llvm-as - assembler

.ll -> .bc

opt - optimizer and analyzer

.ll | .bc -> .ll | .bc

lli - interpreter

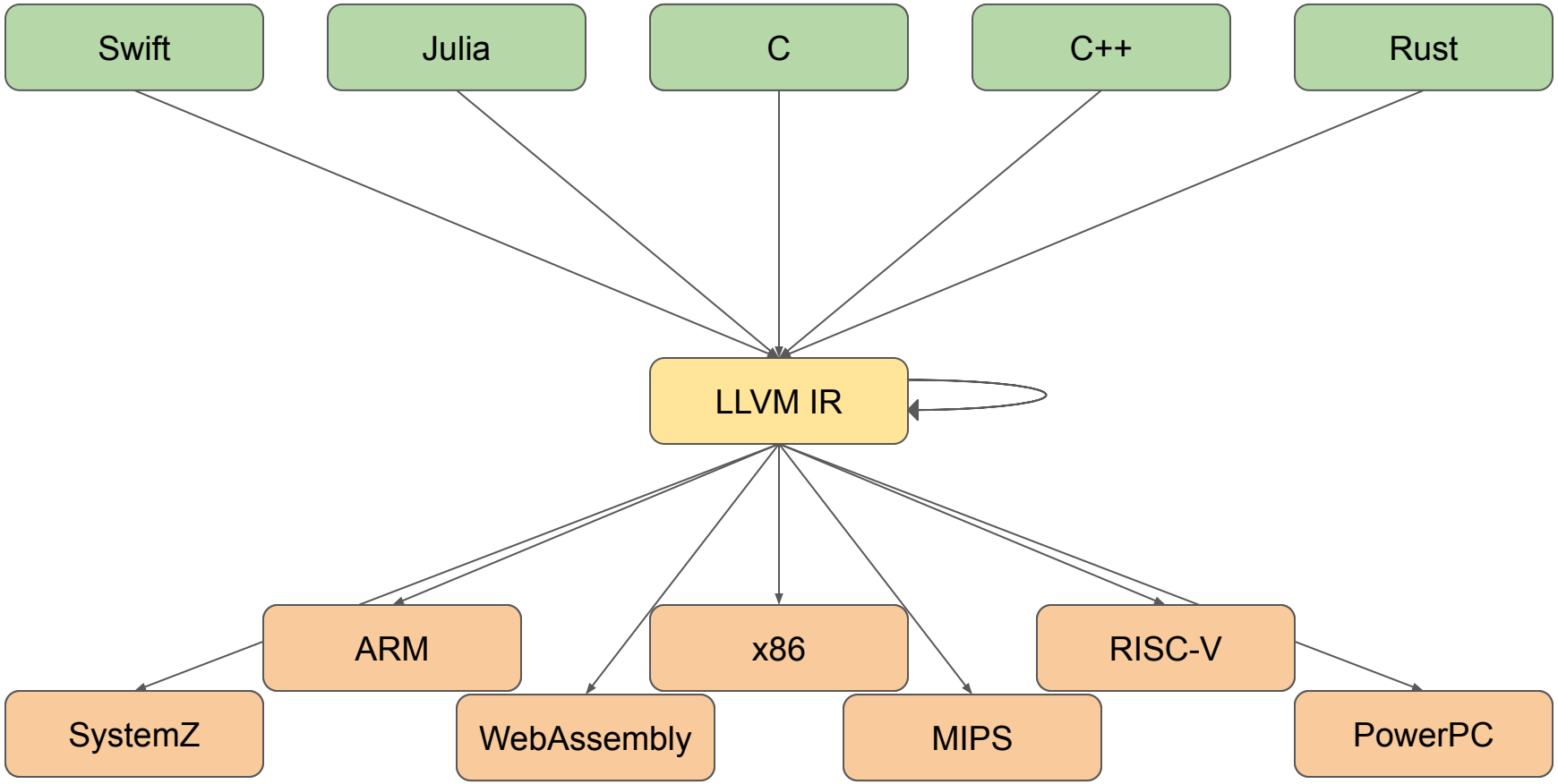
interpret .ll or .bc

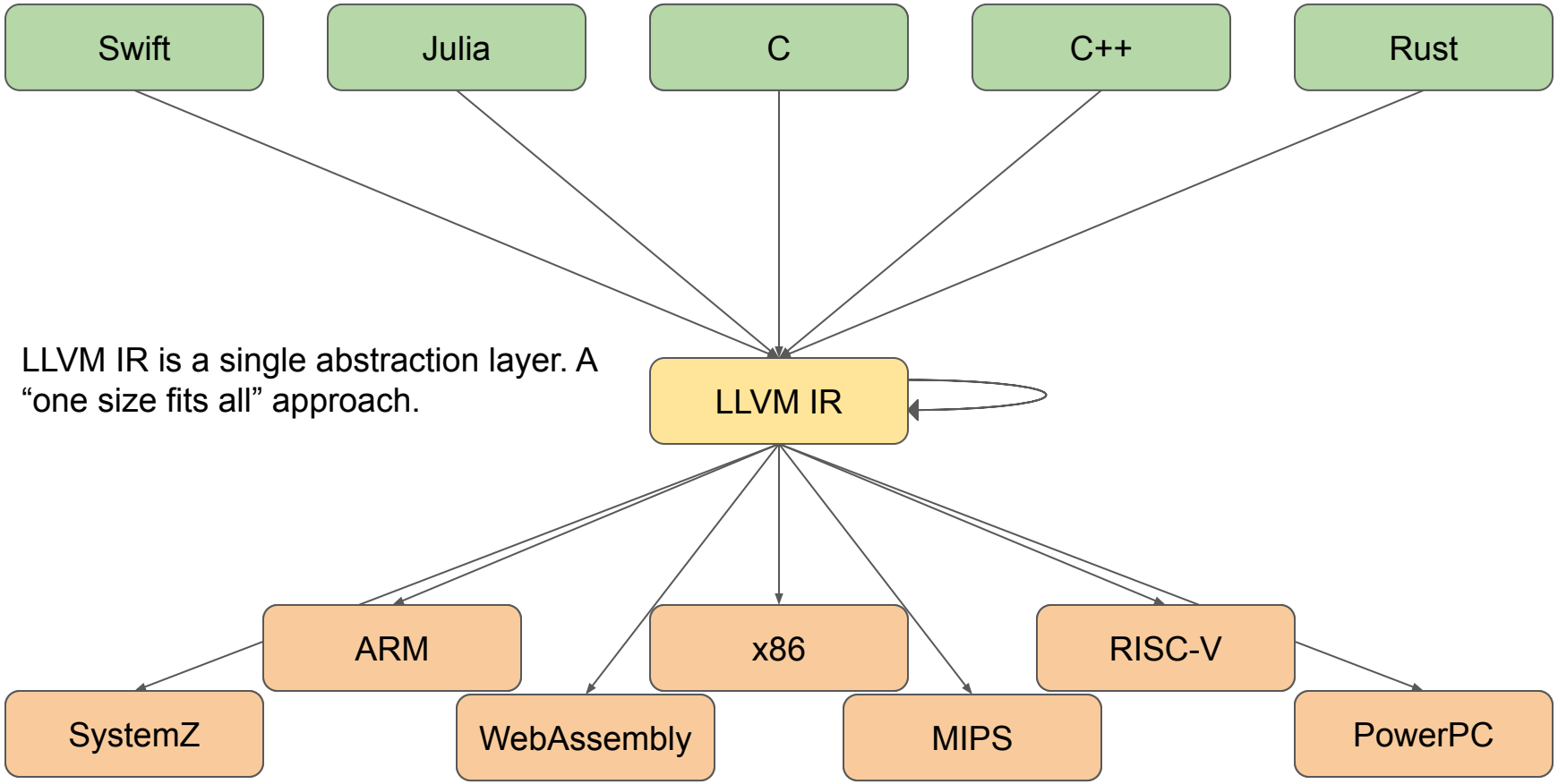
llc - compiler

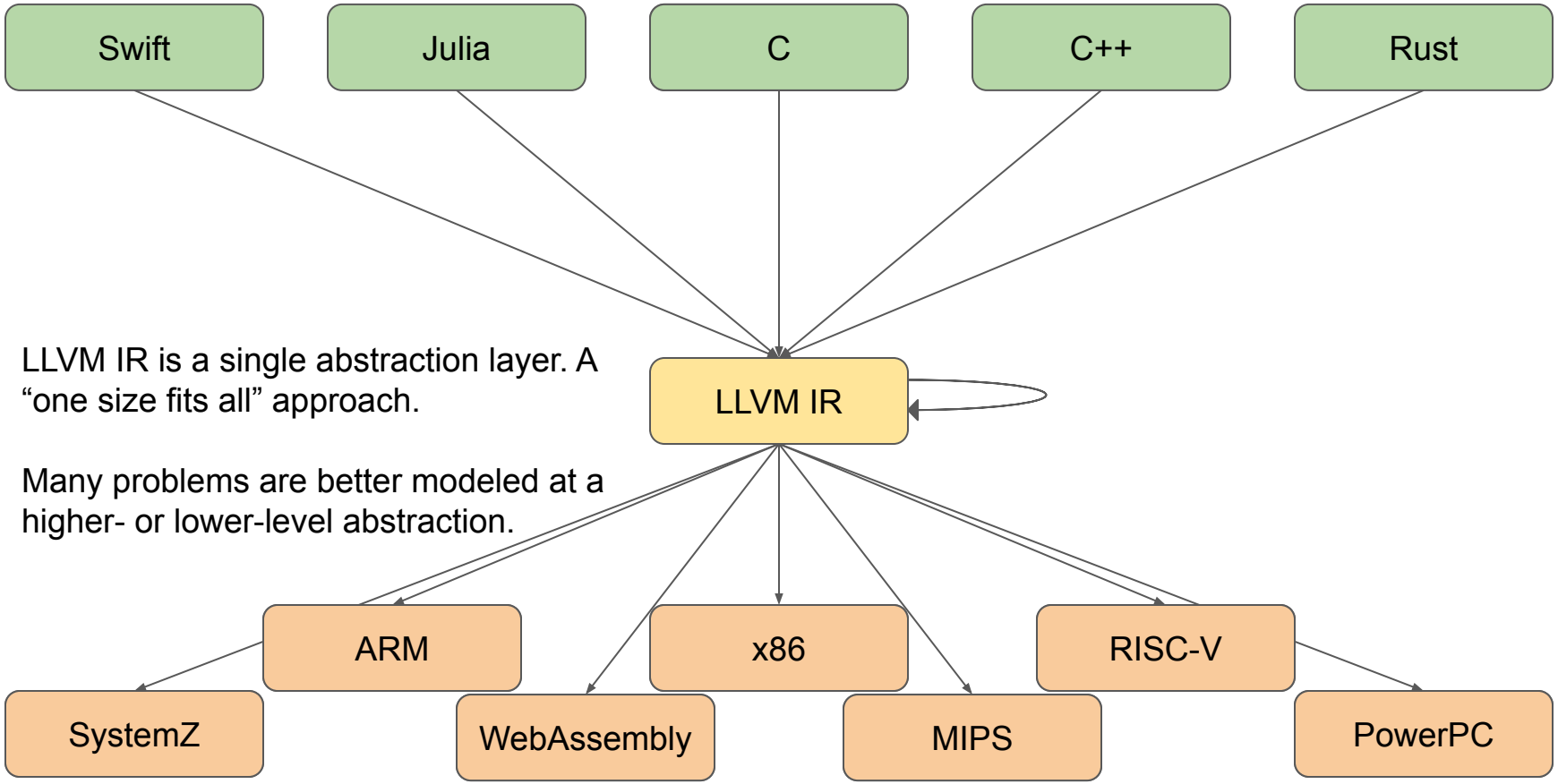
.ll | .bc -> .o

What is LLVM (and MLIR)?

What is ~~LLVM~~ (and MLIR)?

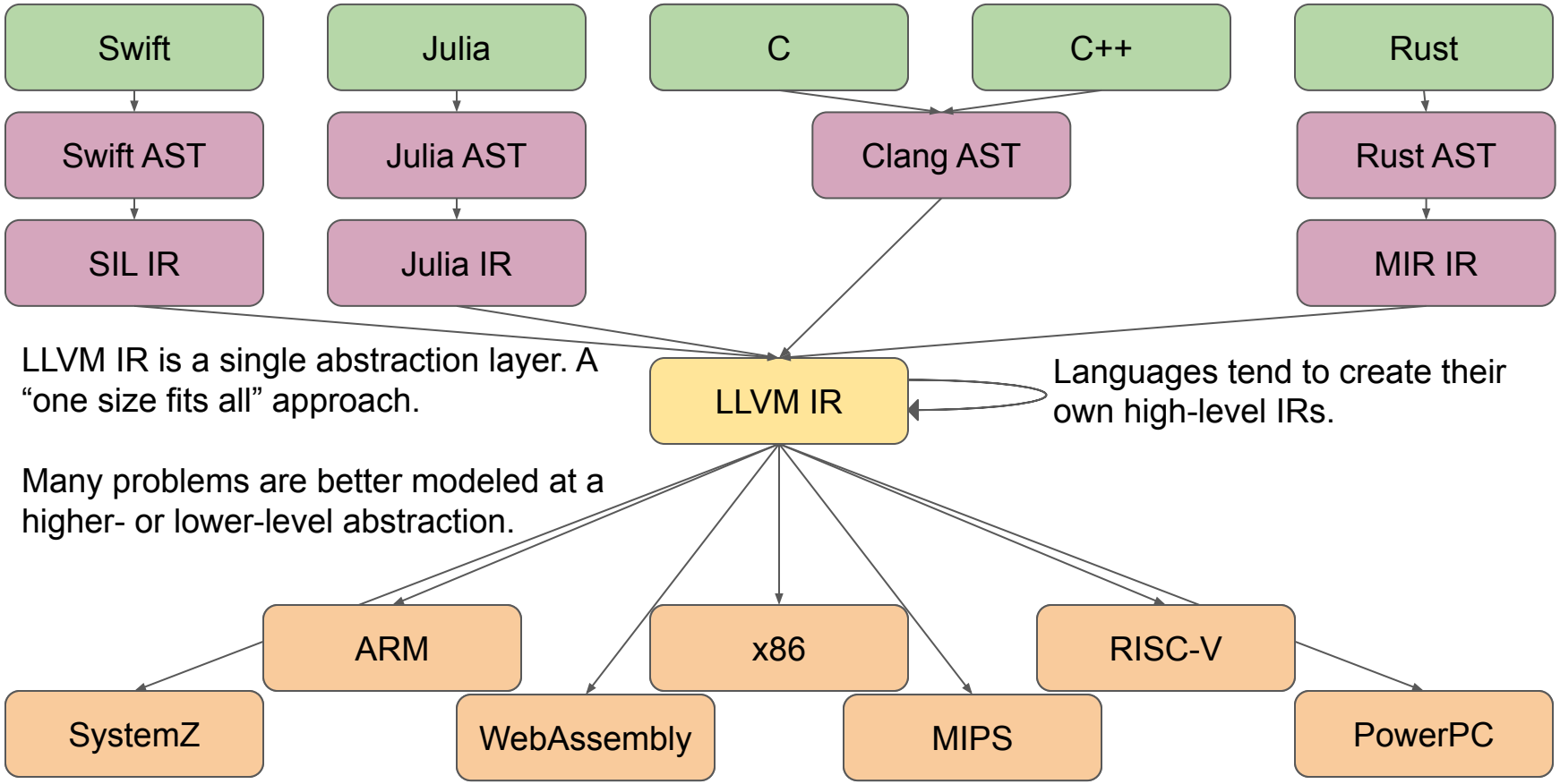

Swift
Julia
C
C++
Rust
LLVM IR
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC

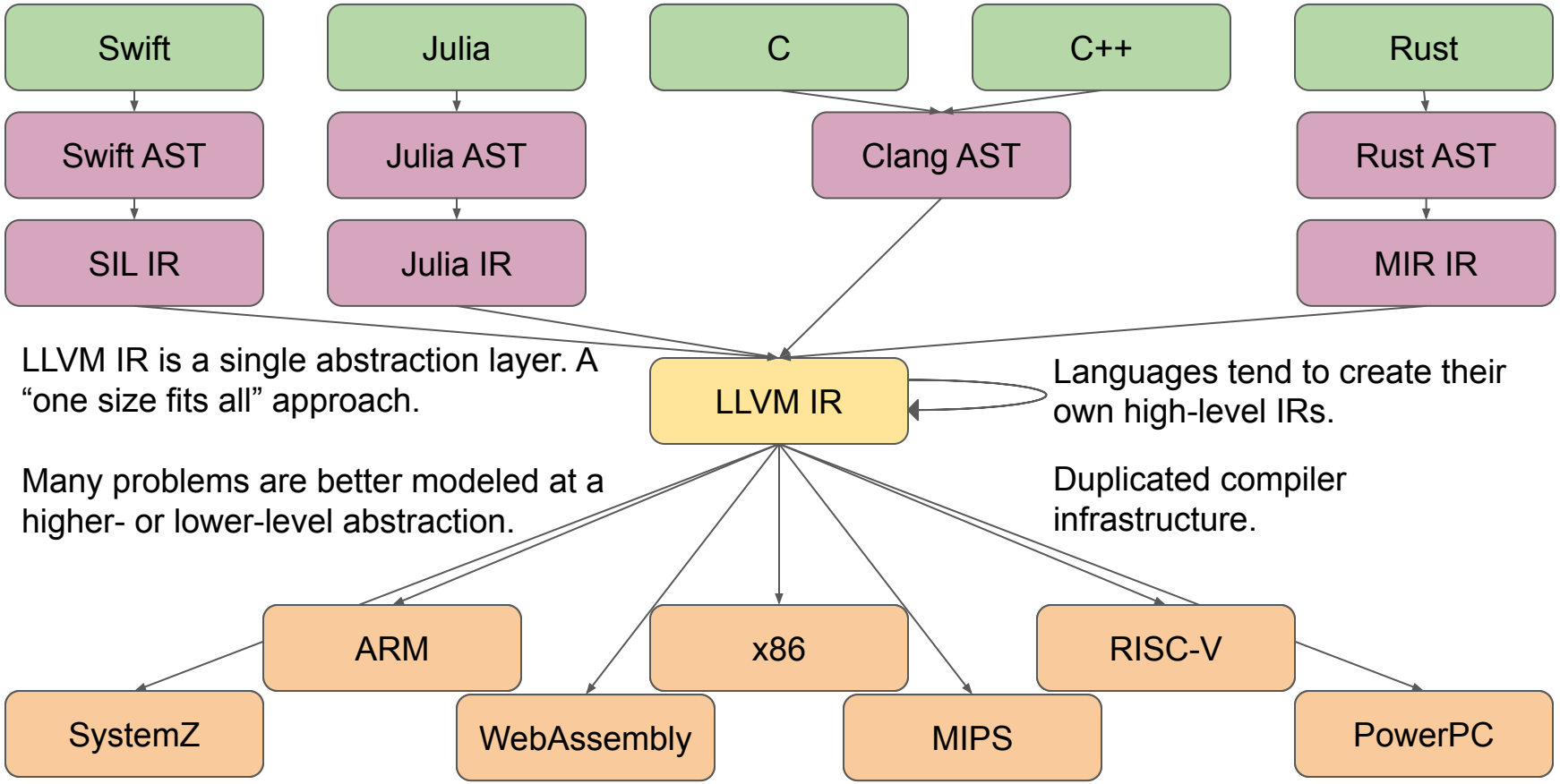

Swift
Julia
C
C++
Rust
LLVM IR is a single abstraction layer. A “one size fits all” approach.
LLVM IR
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC



LLVM IR is a single abstraction layer. A “one size fits all” approach.

Many problems are better modeled at a higher- or lower-level abstraction.


Swift
Julia
C
C++
Rust
Swift AST
Julia AST
Clang AST
Rust AST
SIL IR
Julia IR
MIR IR
LLVM IR is a single abstraction layer. A "one size fits all" approach.
LLVM IR
Languages tend to create their own high-level IRs.
Many problems are better modeled at a higher- or lower-level abstraction.
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC


Swift
Julia
C
C++
Rust
Swift AST
Julia AST
Clang AST
Rust AST
SIL IR
Julia IR
MIR IR
LLVM IR is a single abstraction layer. A “one size fits all” approach.
LLVM IR
Languages tend to create their own high-level IRs.
Many problems are better modeled at a higher- or lower-level abstraction.
Duplicated compiler infrastructure.
ARM
x86
RISC-V
SystemZ
WebAssembly
MIPS
PowerPC

MLIR

Multi-Level Intermediate Representation (MLIR)

Multi-Level Intermediate Representation (MLIR)

- Define intermediate representations, called dialects, for new abstraction levels

Multi-Level Intermediate Representation (MLIR)

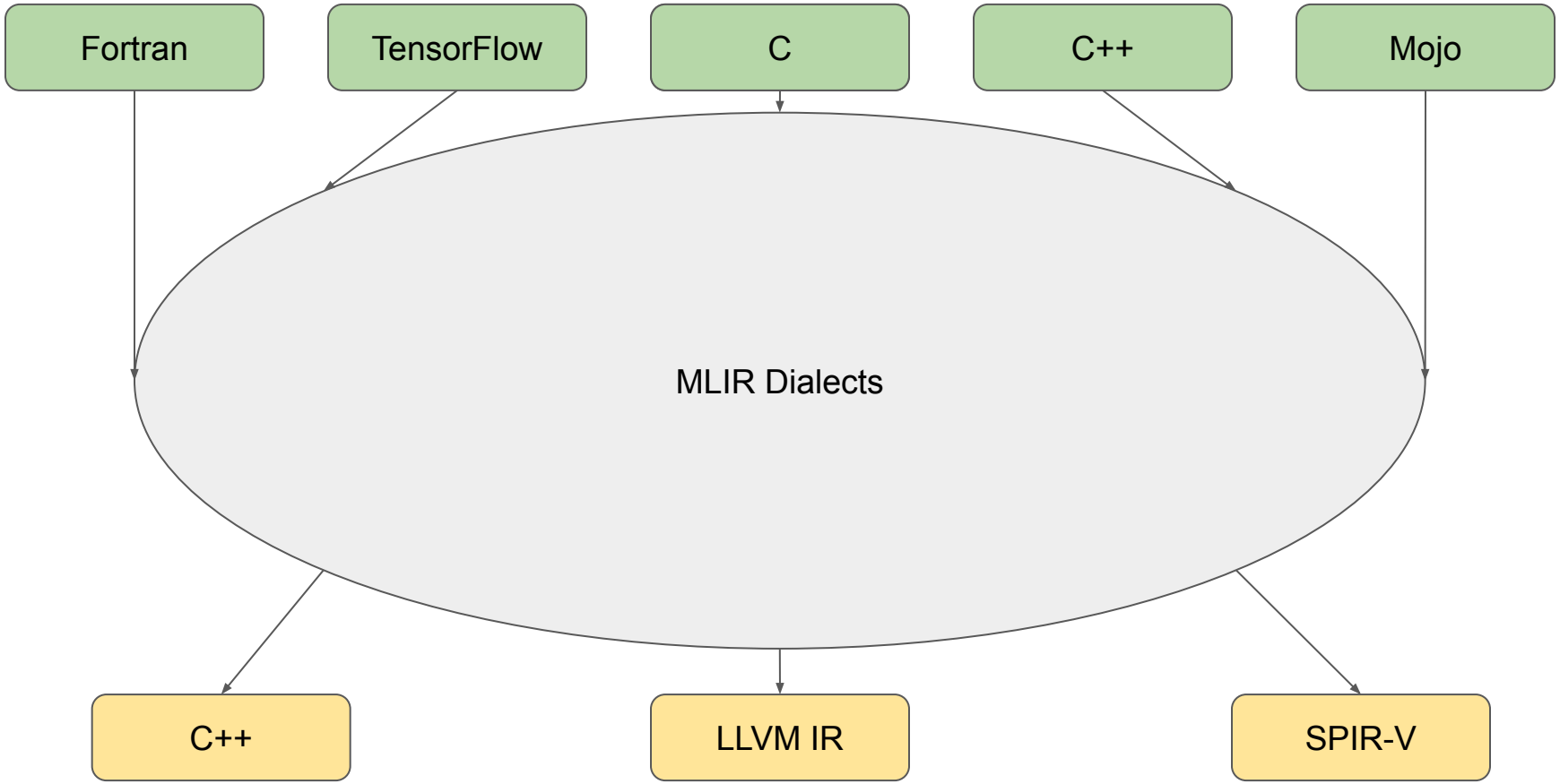
- Define intermediate representations, called dialects, for new abstraction levels
- Mix dialects.

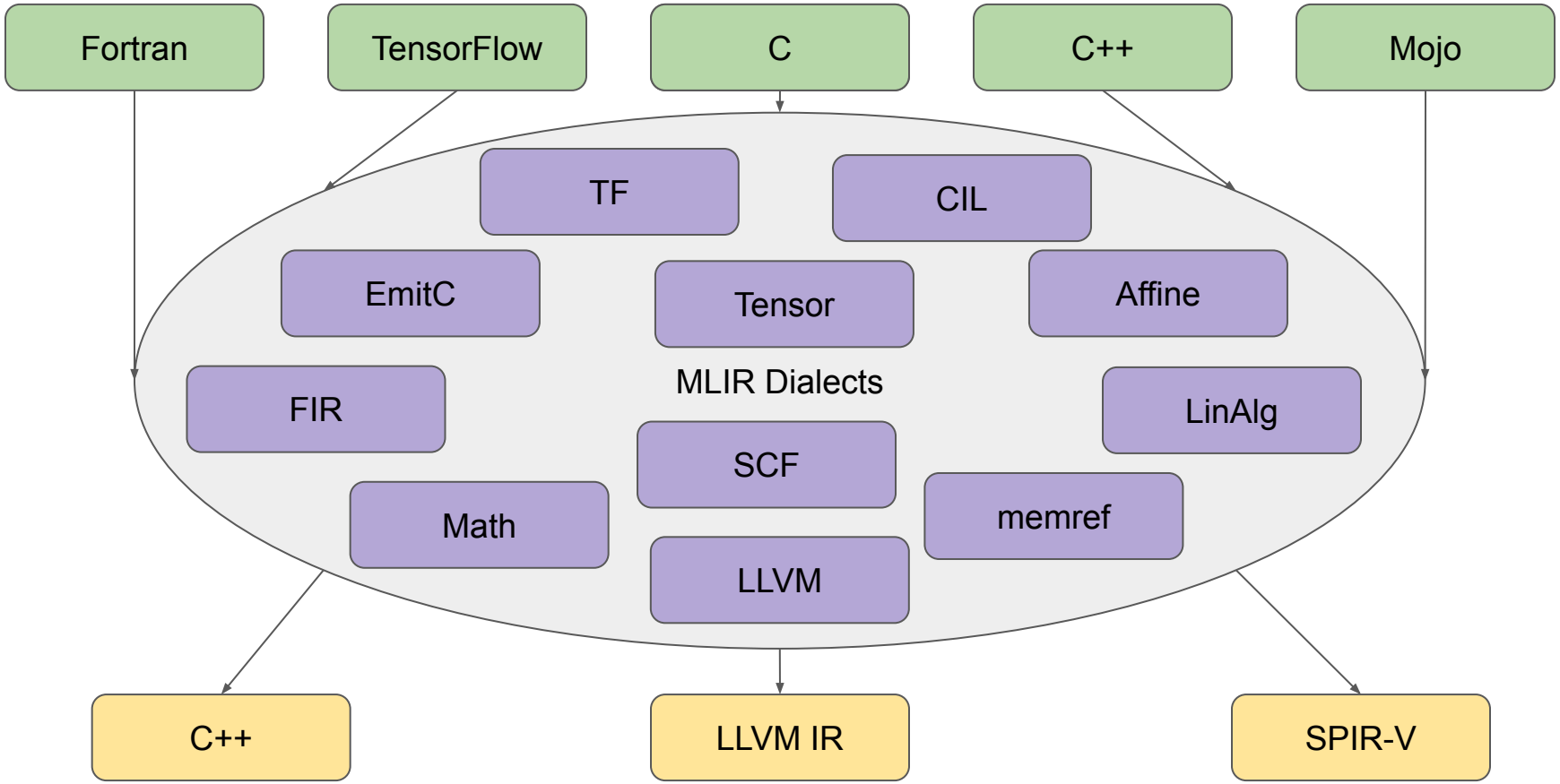
Multi-Level Intermediate Representation (MLIR)

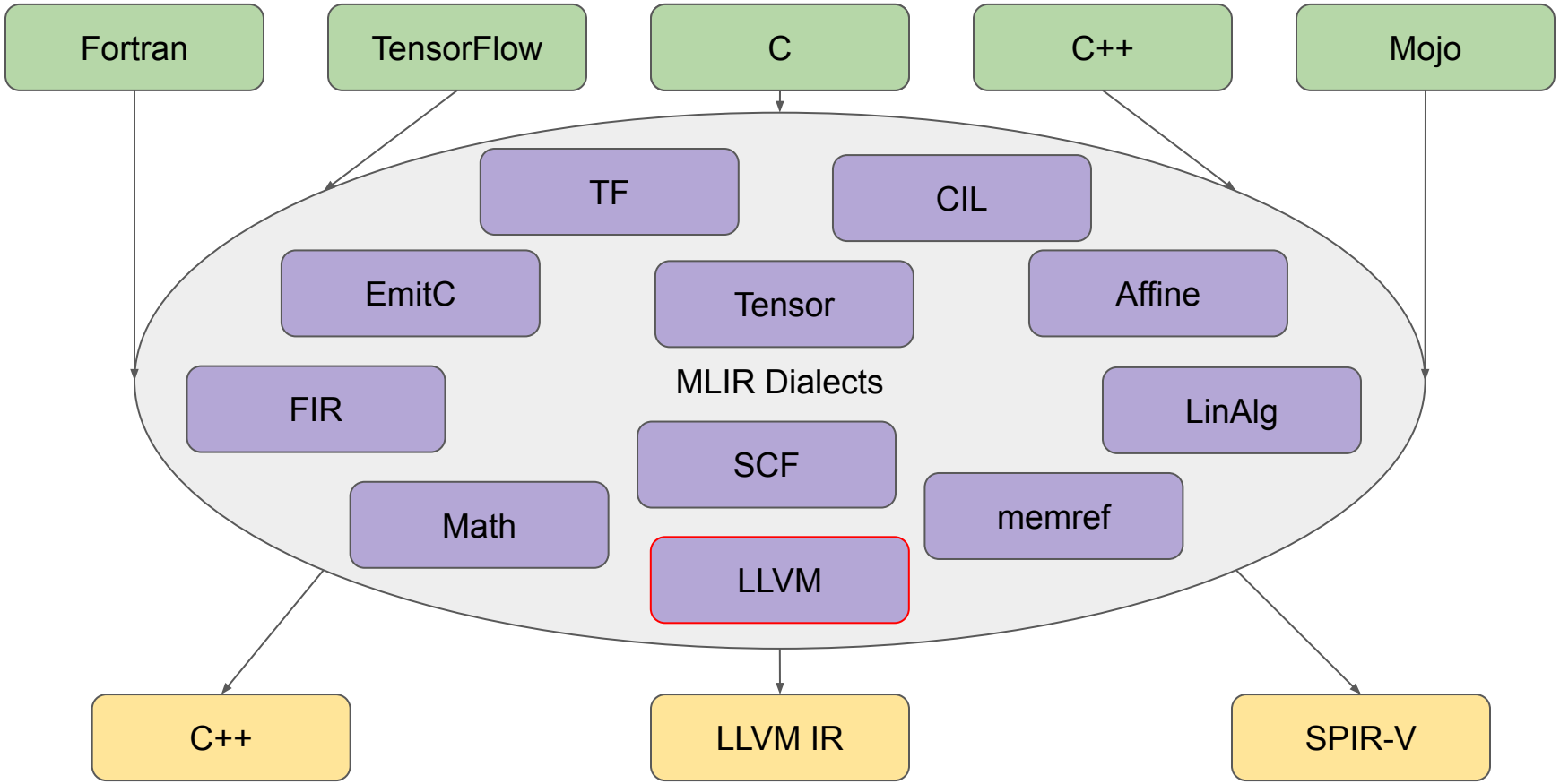
- Define intermediate representations, called dialects, for new abstraction levels
- Mix dialects.
- Define passes to transform/analyze mlir and convert between dialects.

Multi-Level Intermediate Representation (MLIR)

- Define intermediate representations, called dialects, for new abstraction levels
- Mix dialects.
- Define passes to transform/analyze mlir and convert between dialects.
- Create pass pipelines.

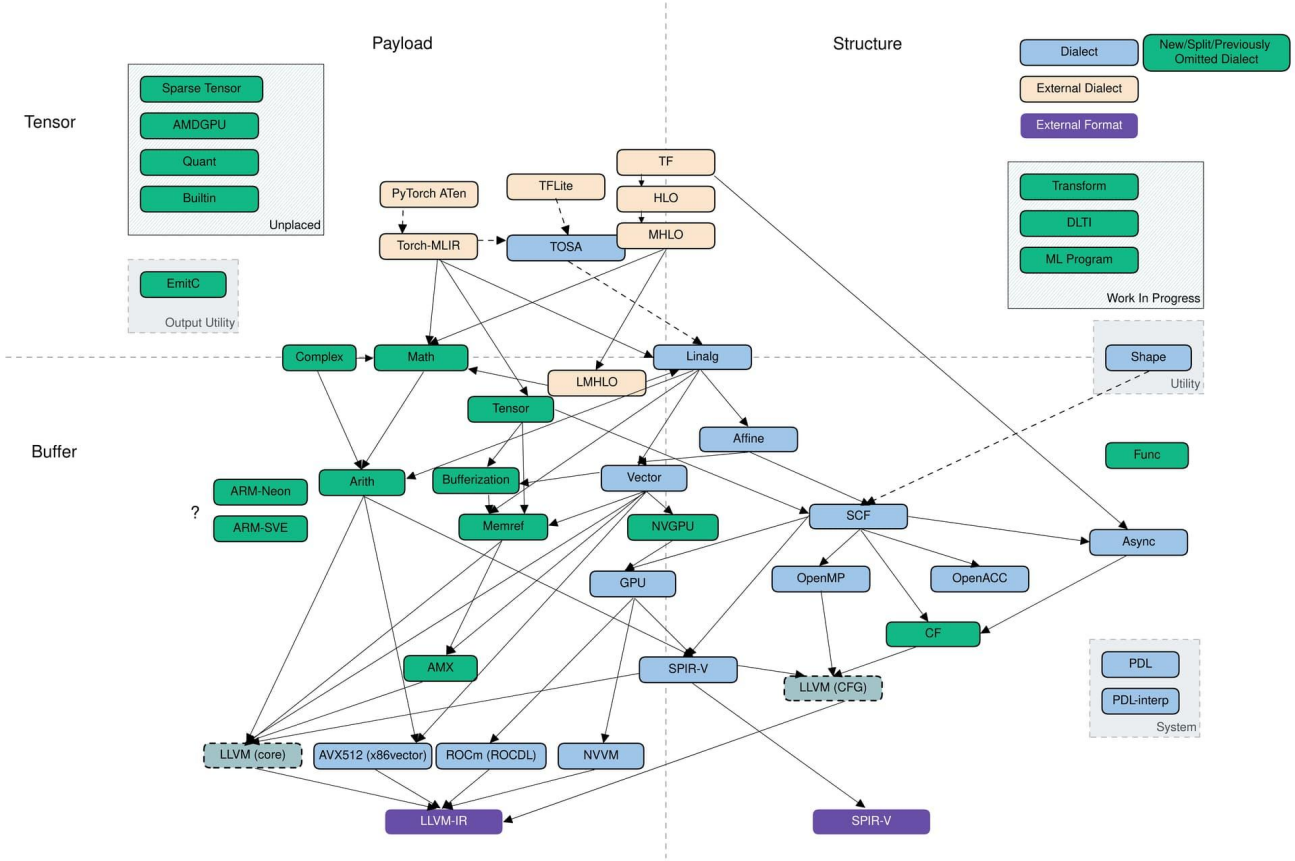

Fortran
TensorFlow
C
C++
Mojo
MLIR Dialects
C++
LLVM IR
SPIR-V


Fortran
TensorFlow
C
C++
Mojo
TF
CIL
EmitC
Tensor
Affine
MLIR Dialects
FIR
LinAlg
SCF
Math
memref
LLVM
C++
LLVM IR
SPIR-V


Fortran
TensorFlow
C
C++
Mojo
TF
CIL
EmitC
Tensor
Affine
MLIR Dialects
FIR
LinAlg
SCF
Math
memref
LLVM
C++
LLVM IR
SPIR-V

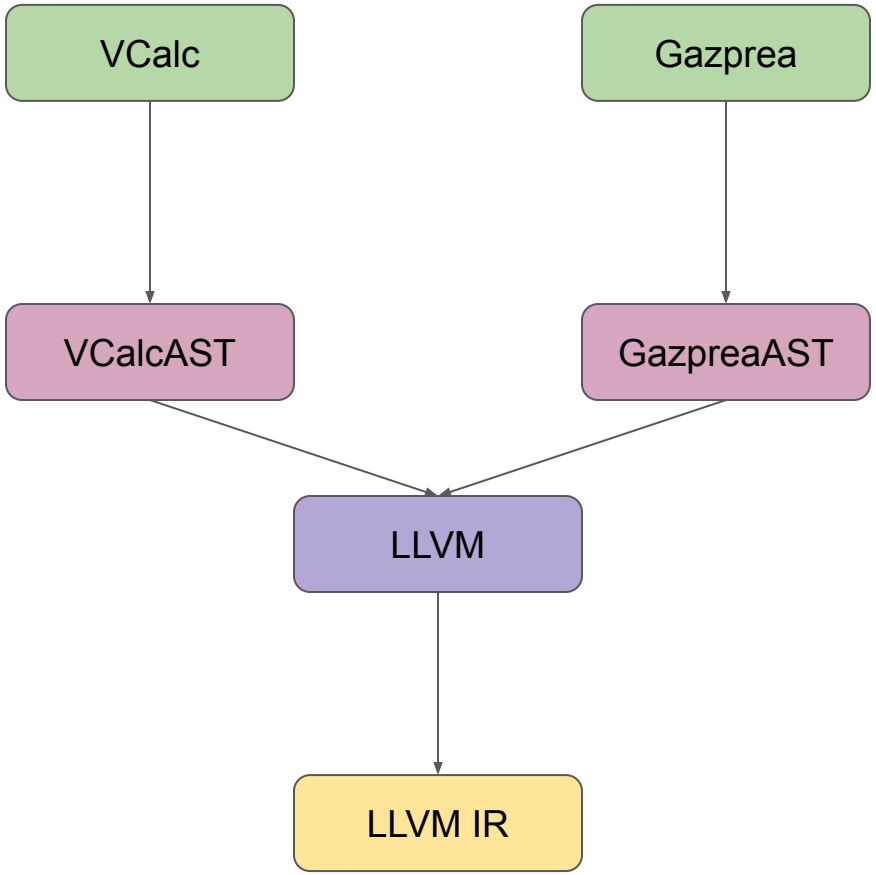
What really goes on in MLIR…

What really goes on in MLIR…



What you
need to do…

What you need to do…



MLIR Tools

MLIR Tools

mlir-opt - optimizer and lowerer

optimize mlir

lower mlir to lower level dialect

MLIR Tools

mlir-opt - optimizer and lowerer

optimize mlir

lower mlir to lower level dialect

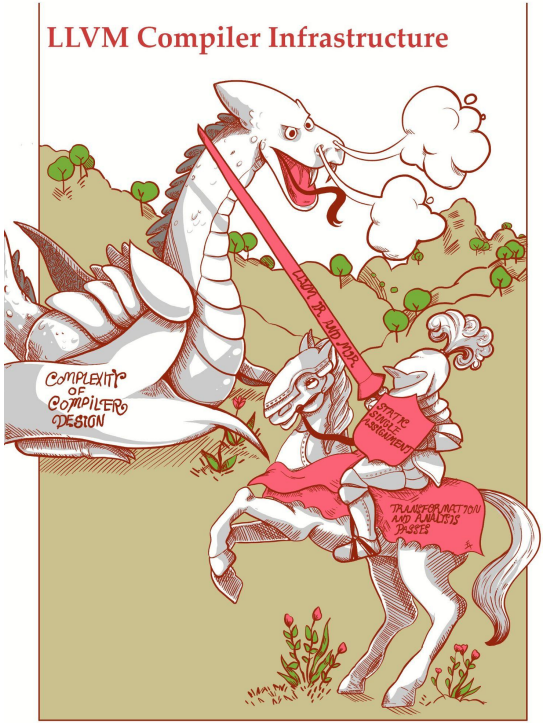
mlir-translate - translation tool

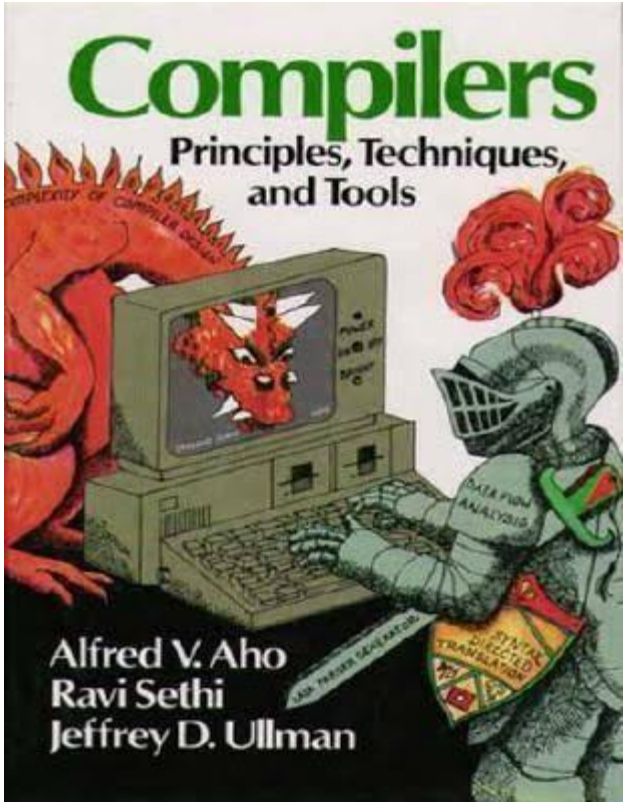
mlir -> external representation

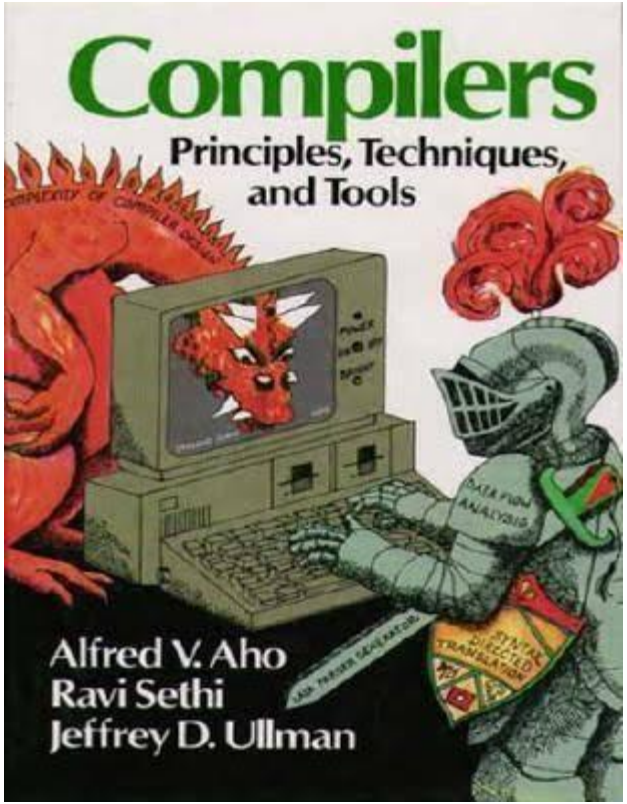
external representation -> mlir


Principles of Compiler Design
Alfred V. Aho
Jeffrey D. Ullman


Principles of Compiler Design
Alfred V. Aho
Jeffrey D. Ullman


LLVM Compiler Infrastructure
COMPLEXITY OF COMPILER DESIGN
LLVM IR AND APIS
STATE MACHINE ASSIGNMENT
TRANSFORMATION AND ANALYSIS PASSES


Compilers
Principles, Techniques, and Tools
Alfred V. Aho
Ravi Sethi
Jeffrey D. Ullman


Compilers
Principles, Techniques, and Tools
Alfred V. Aho
Ravi Sethi
Jeffrey D. Ullman


LLVM
Compiler Infrastructure
COMPLEXITY OF COMPILER DESIGN
"NFC" COMMITS
ADDRESS SANITIZER
STATIC SINGLE ASSIGNMENT
JESSICA PAQUETTE 2018

Next Time

- LLVM IR & MLIR examples
- Working with LLVM & MLIR
- How to emit MLIR LLVM dialect in C++

https://llvm.org/docs/LangRef.html

```
int add(int left, int right) {
  return left + right;
}
```

```
define i32 @add(i32 %0, i32 %1) {
  %3 = alloca i32
  %4 = alloca i32
  store i32 %0, ptr %3
  store i32 %1, ptr %4
  %5 = load i32, ptr %3
  %6 = load i32, ptr %4
  %7 = add i32 %5, %6
  ret i32 %7
}
```

- Instruction Reference
  - Terminator Instructions
    - 'ret' Instruction
    - 'br' Instruction
    - 'switch' Instruction
    - 'indirectbr' Instruction
    - 'invoke' Instruction
    - 'callbr' Instruction
    - 'resume' Instruction
    - 'catchswitch' Instruction
    - 'catchret' Instruction
    - 'cleanupret' Instruction
    - 'unreachable' Instruction
  - Unary Operations
    - 'fneg' Instruction
  - Binary Operations
    - 'add' Instruction
    - 'fadd' Instruction
    - 'sub' Instruction
    - 'fsub' Instruction
    - 'mul' Instruction
    - 'fmul' Instruction
    - 'udiv' Instruction
    - 'sdiv' Instruction
    - 'fdiv' Instruction
    - 'urem' Instruction
    - 'srem' Instruction
    - 'frem' Instruction

```
int add(int left, int right) {
  return left + right;
}
define i32 @add(i32 %0, i32 %1) {
  %3 = alloca i32
  %4 = alloca i32
  store i32 %0, ptr %3
  store i32 %1, ptr %4
  %5 = load i32, ptr %3
  %6 = load i32, ptr %4
  %7 = add i32 %5, %6
  ret i32 %7
}
```