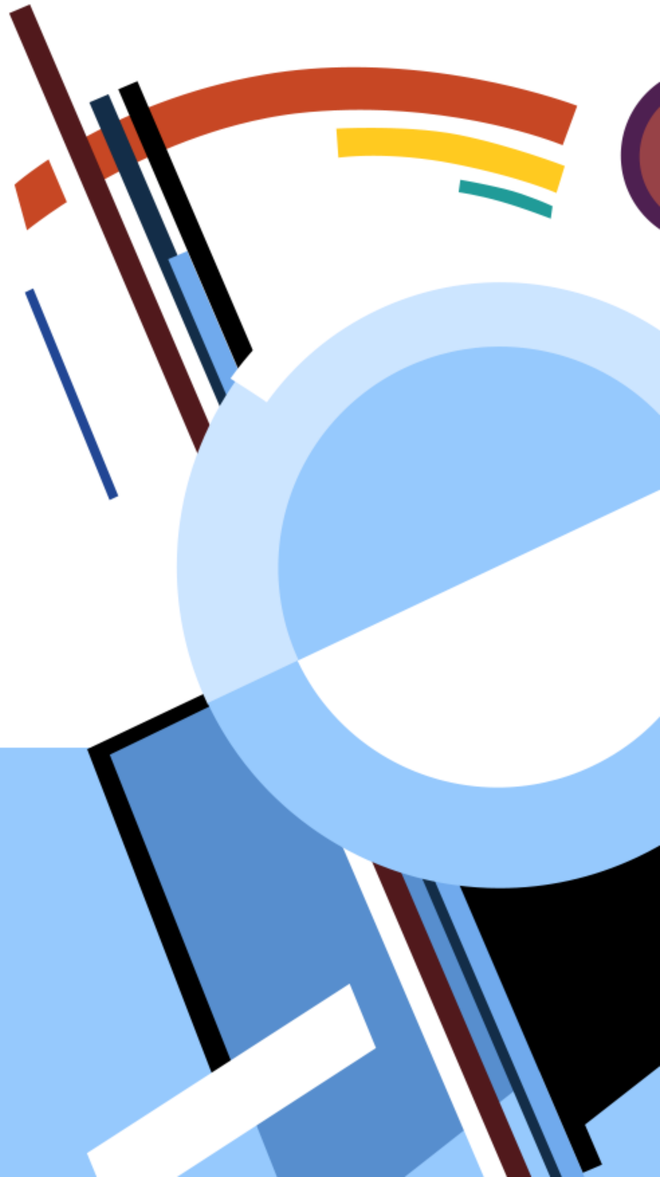
Getting started with LLVM using Swift

Alex Denisov,
http://lowlevelbits.org



App's 2016 Apps Conf

whoami

- iOS Apps Developer
- Compiler Hobbyist
- Internet User:

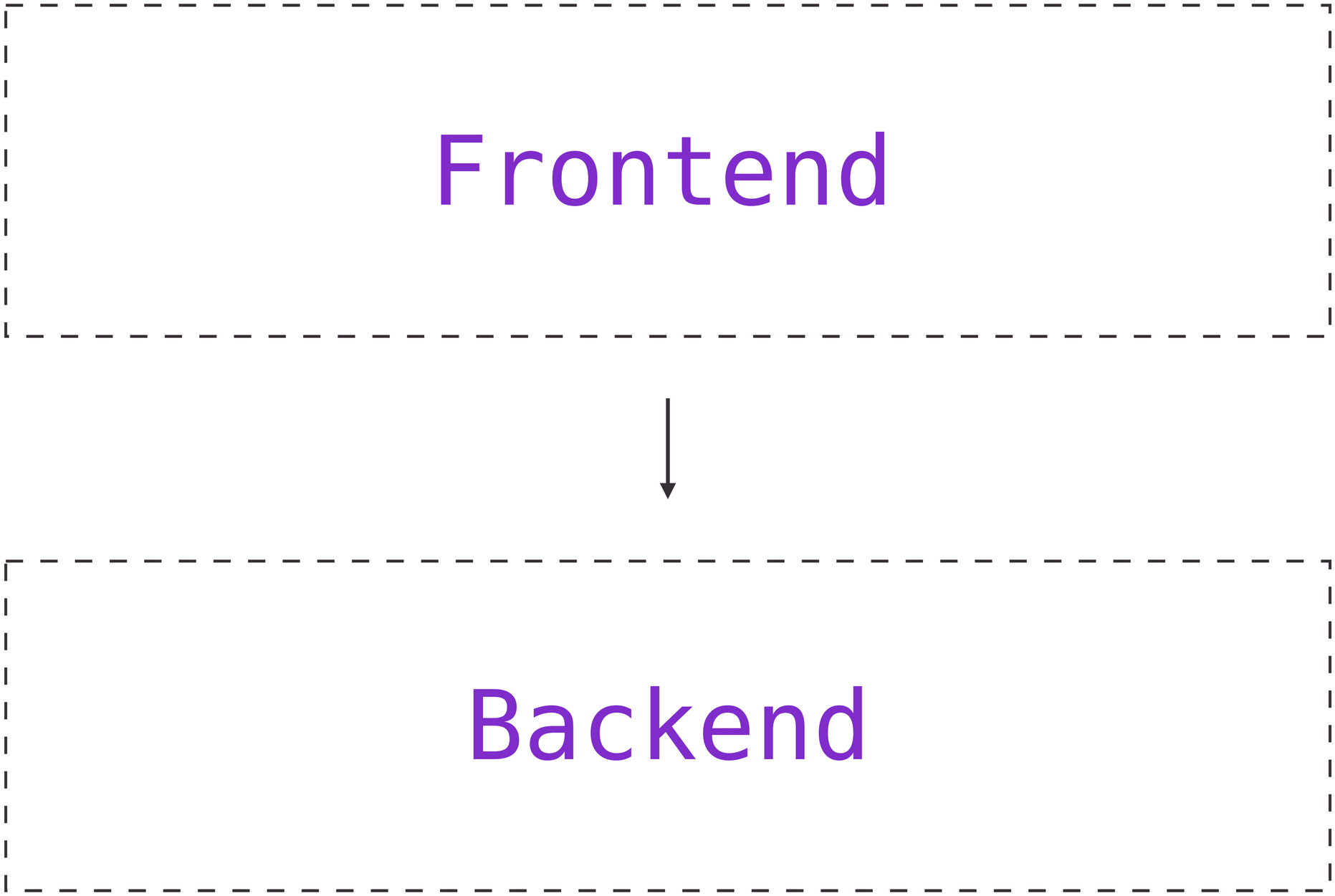
  https://twitter.com/1101_debian
  https://github.com/AlexDenisov
  http://lowlevelbits.org

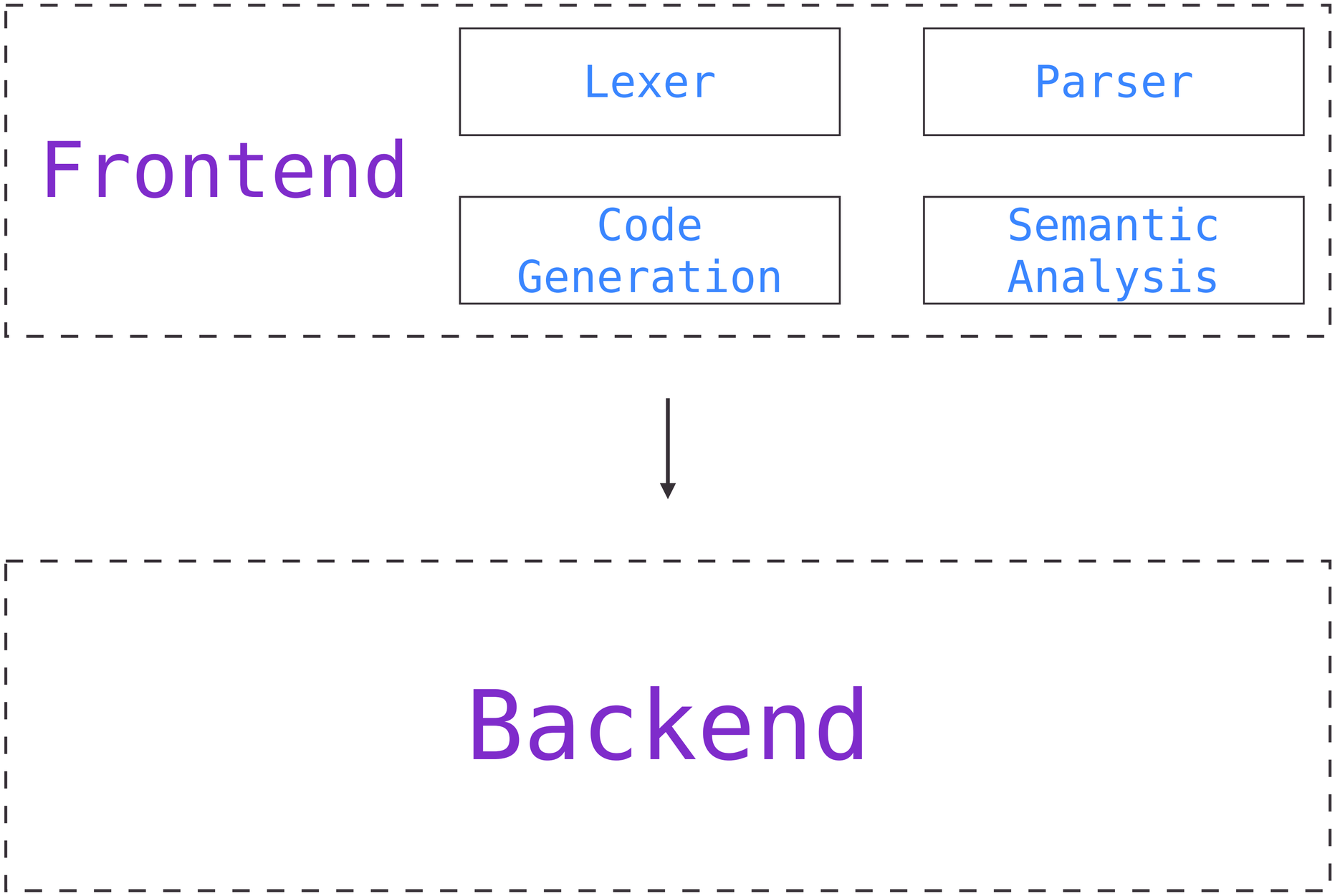
AppsConf

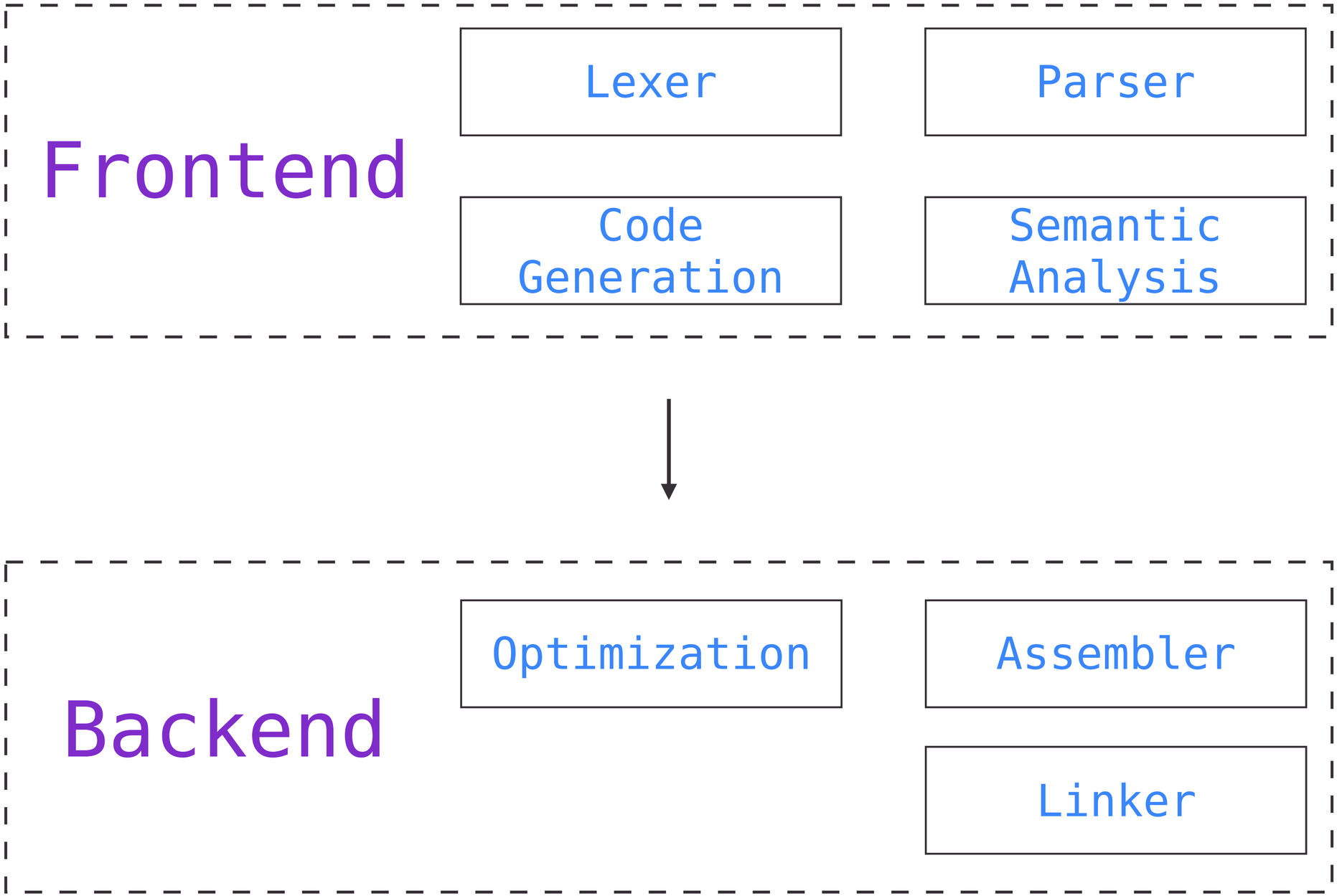
Outline

- What is LLVM
- Compilation Process
- Practical Applications
- LLVM C API
- QA

LLVM




Frontend
Lexer
Parser
Code Generation
Semantic Analysis
Backend


Frontend
Lexer
Parser
Code Generation
Semantic Analysis
Backend
Optimization
Assembler
Linker

Frontend

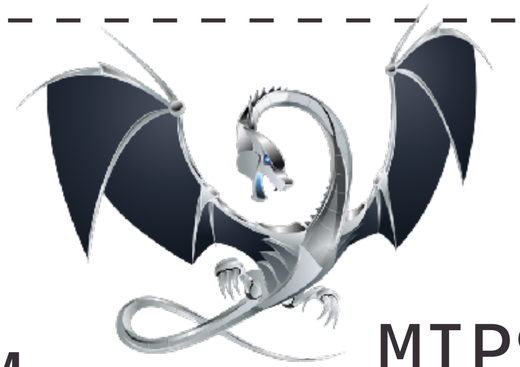
Backend

Optimization

Assembler

Linker

Frontend



Backend

ARM

x86

MIPS

...

Compilation Process

```
int main(){
  return 0;
}
```

```
0000000 cf fa ed fe 07 00 00 01 03 00 00 80 02 00 00 00
0000010 0f 00 00 00 38 03 00 00 85 00 20 00 00 00 00 00
0000020 19 00 00 00 48 00 00 00 5f 5f 50 41 47 45 5a 45
0000030 52 4f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000040 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00
0000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000060 00 00 00 00 00 00 00 00 19 00 00 00 38 01 00 00
0000070 5f 5f 54 45 58 54 00 00 00 00 00 00 00 00 00 00
0000080 00 00 00 00 01 00 00 00 00 10 00 00 00 00 00 00
0000090 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00
00000a0 07 00 00 00 05 00 00 00 03 00 00 00 00 00 00 00
00000b0 5f 5f 74 65 78 74 00 00 00 00 00 00 00 00 00 00
00000c0 5f 5f 54 45 58 54 00 00 00 00 00 00 00 00 00 00
00000d0 98 0f 00 00 01 00 00 00 08 00 00 00 00 00 00 00
00000e0 98 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000f0 00 04 00 80 00 00 00 00 00 00 00 00 00 00 00 00
0000100 5f 5f 75 6e 77 69 6e 64 5f 69 6e 66 6f 00 00 00
0000110 5f 5f 54 45 58 54 00 00 00 00 00 00 00 00 00 00
0000120 a0 0f 00 00 01 00 00 00 48 00 00 00 00 00 00 00
0000130 a0 0f 00 00 02 00 00 00 00 00 00 00 00 00 00 00
0000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000150 5f 5f 65 68 5f 66 72 61 6d 65 00 00 00 00 00 00
0000160 5f 5f 54 45 58 54 00 00 00 00 00 00 00 00 00 00
0000170 e8 0f 00 00 01 00 00 00 18 00 00 00 00 00 00 00
0000180 e8 0f 00 00 03 00 00 00 00 00 00 00 00 00 00 00
0000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001a0 19 00 00 00 48 00 00 00 5f 5f 4c 49 4e 4b 45 44
00001b0 49 54 00 00 00 00 00 00 00 10 00 00 01 00 00 00
00001c0 00 10 00 00 00 00 00 00 00 10 00 00 00 00 00 00
00001d0 d8 00 00 00 00 00 00 00 07 00 00 00 01 00 00 00
```

```
const float factor = 42.f;

int calc(float x) {
    return factor * x;
}
```

Lexer

```
const float factor = 42.f;

int calc(float x) {
    return factor * x;
}
```

(KW ‘const’)

```
const float factor = 42.f;

int calc(float x) {
    return factor * x;
}
```

AppsConf

(KW ‘const’) (TYPE ‘float’)

```
const float factor = 42.f;

int calc(float x) {
    return factor * x;
}
```

```
(KW ‘const’) (TYPE ‘float’) (ID ‘factor’) (EQ ‘=‘)
(NUM ’42.f’) (SEMI ‘;’)
```

```
const float factor = 42.f;

int calc(float x) {
    return factor * x;
}
```

AppsConf

```
(KW ‘const’) (TYPE ‘float’) (ID ‘factor’) (EQ ‘=‘)
(NUM ’42.f’) (SEMI ‘;’) (TYPE ‘int’) (ID ‘calc’)
(L_PAREN ‘(‘) (TYPE ‘float’) (ID ‘x’) (R_PAREN ‘)’)
(L_BRACE ‘{‘) (KW ‘return’) (ID ‘factor’) (STAR ‘*’)
(ID ‘x’) (SEMI ‘;’) (R_BRACE ‘}’) (EOF ‘’)
```

```
const float factor = 42.f;

int calc(float x) {
    return factor * x;
}
```

AppsConf

Parser

```
return factor * x
```

```
return factor * x
```

```
(KW ‘return’) (ID ‘factor’) (STAR ‘*’) (ID ‘x’)
```

```
return factor * x
```

↓

```
(KW 'return') (ID 'factor') (STAR '*') (ID 'x')
```

↓

```
(KW 'return')
```

↓

```
(STAR '*')
```

↙ ↘

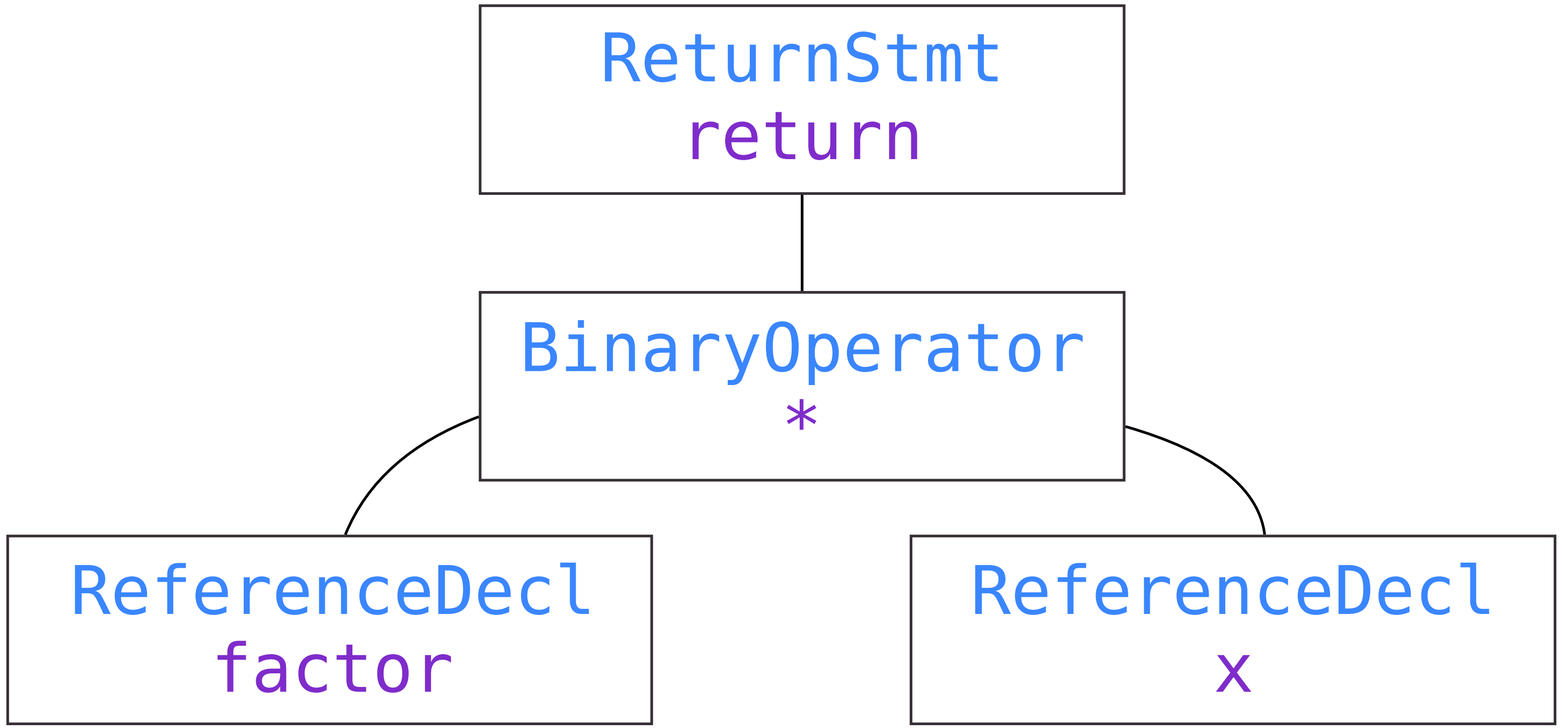
```
(ID 'factor')   (ID 'x')
```

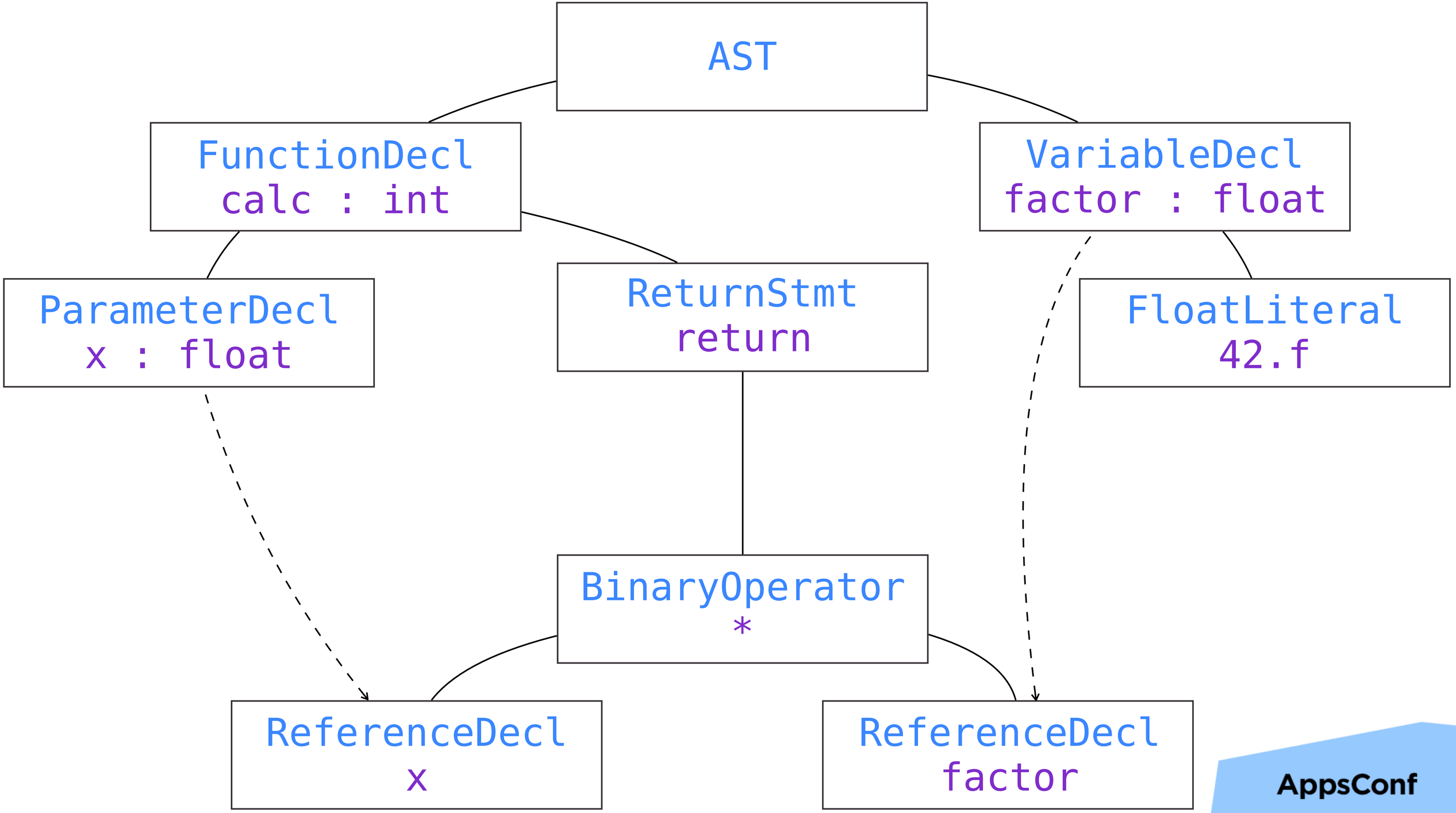
```
return factor * x
```

↓

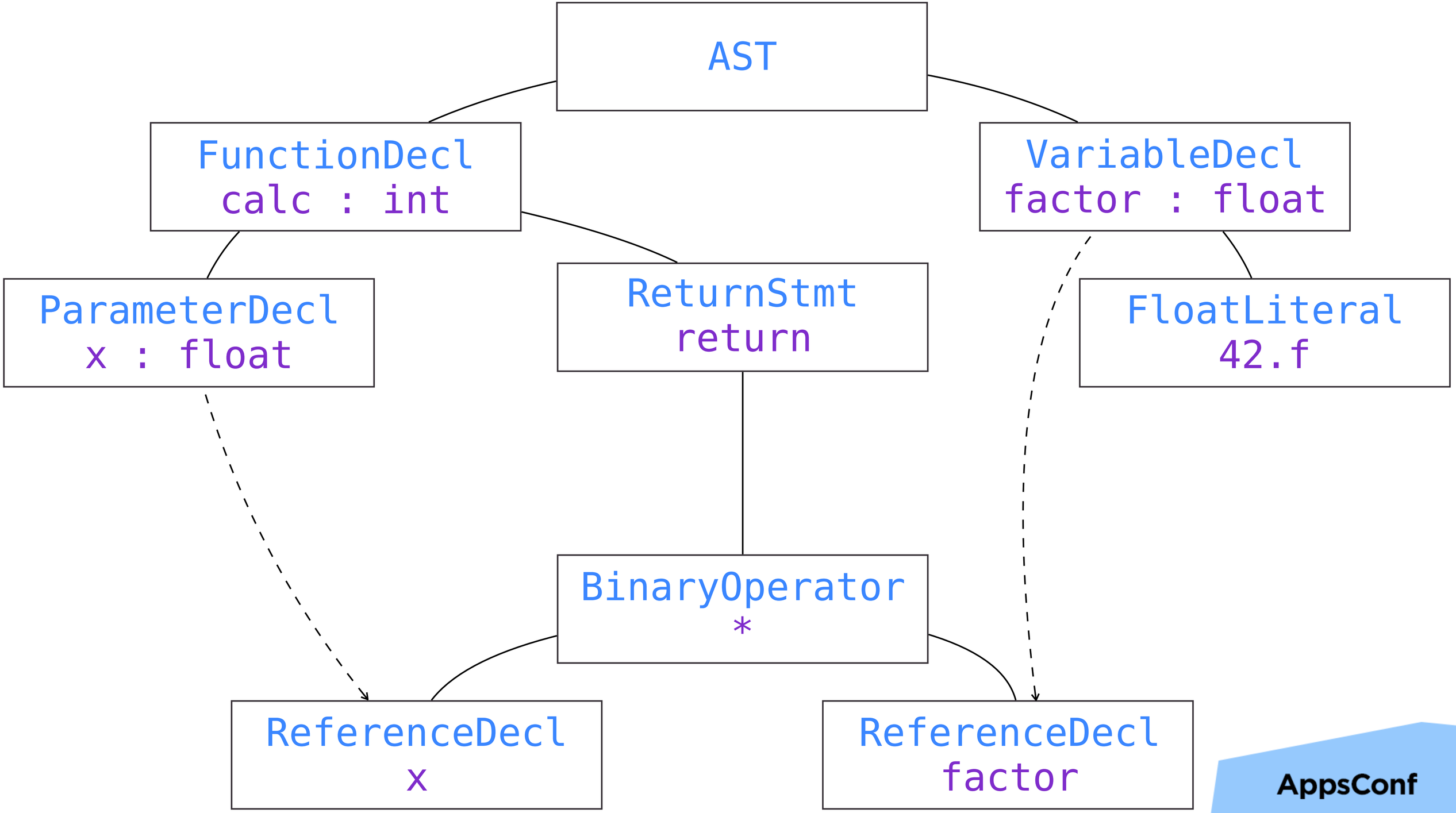
```
(KW 'return') (ID 'factor') (STAR '*') (ID 'x')
```

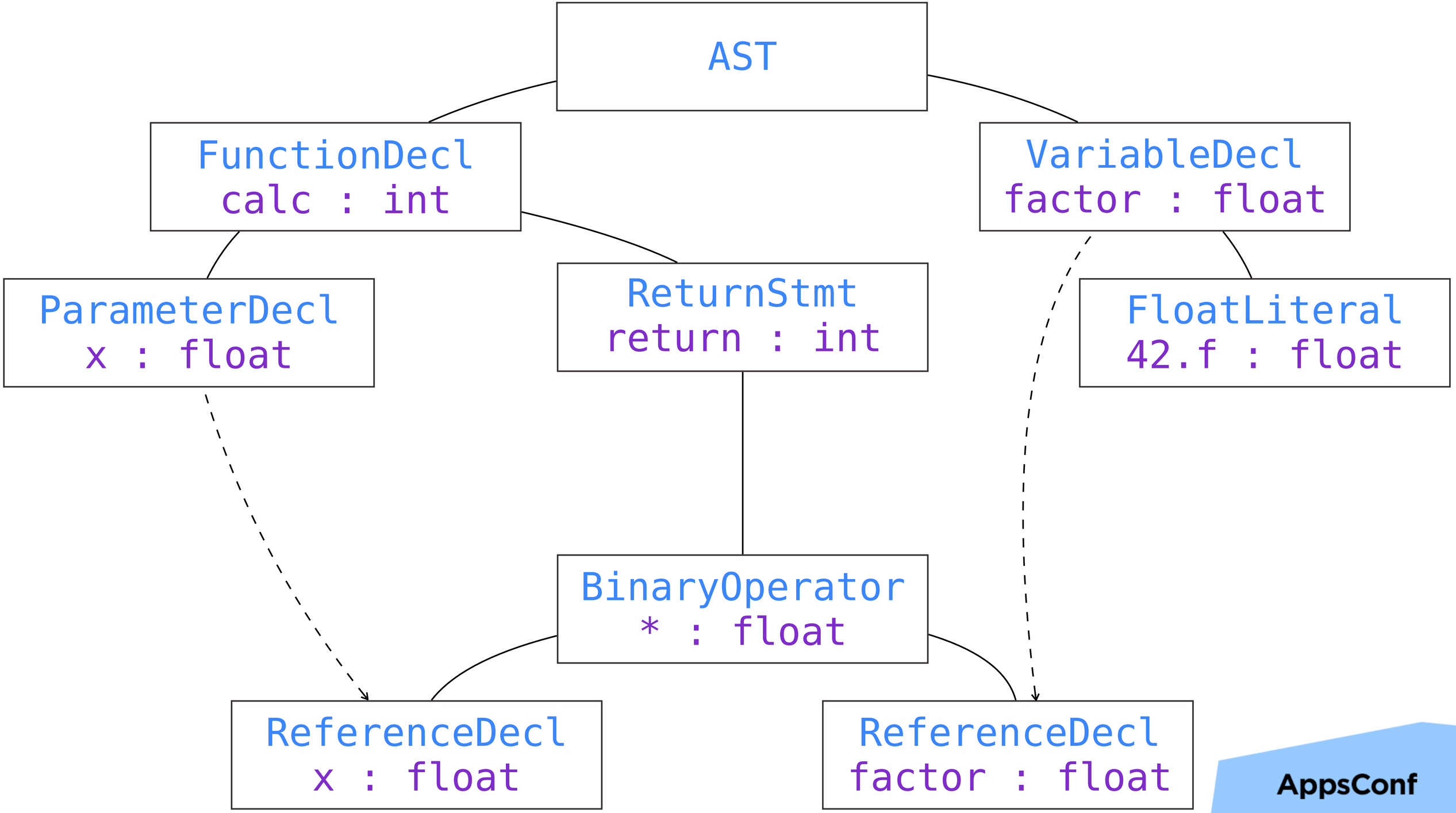
↓



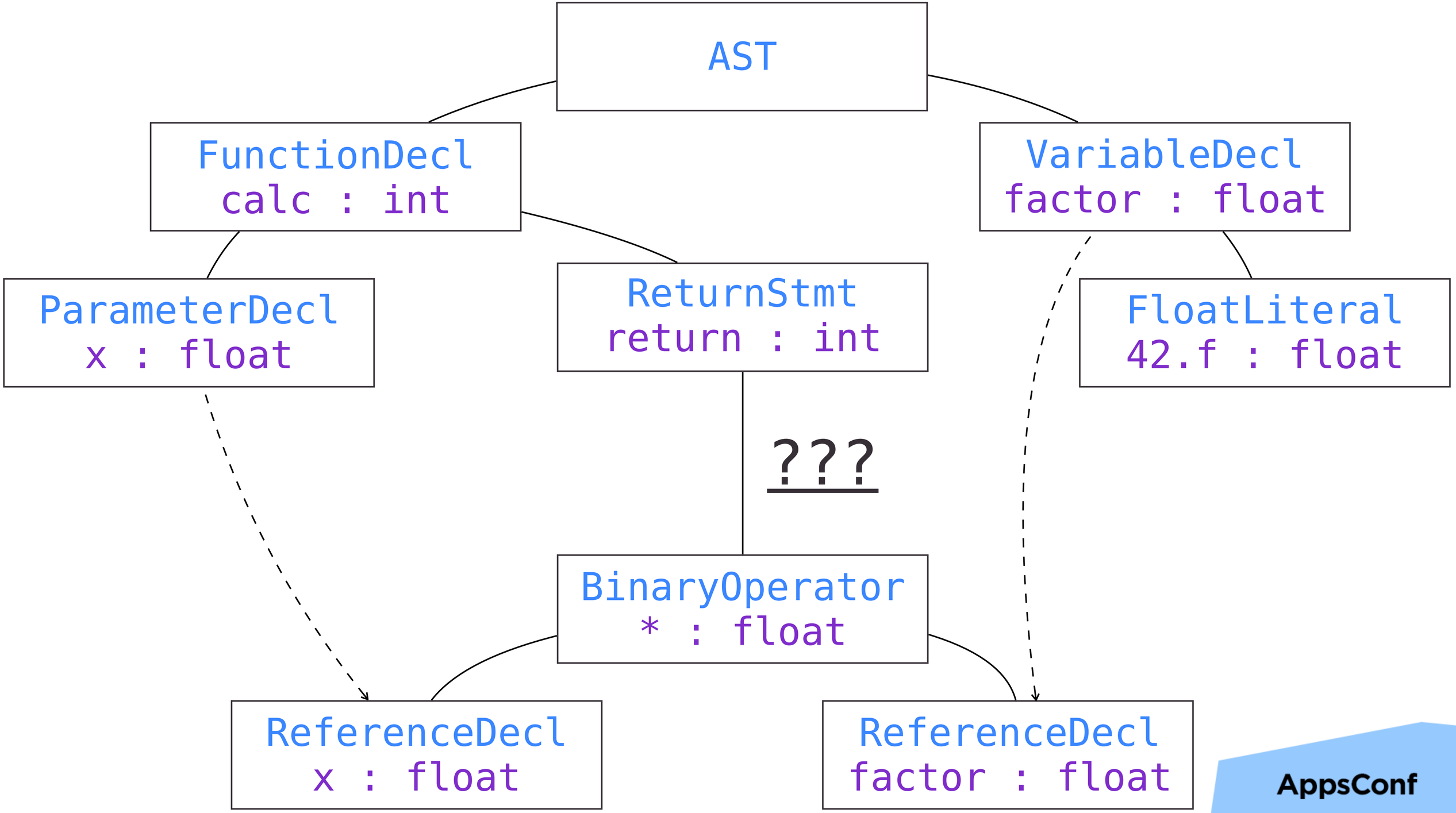

AST
FunctionDecl
calc : int
VariableDecl
factor : float
ParameterDecl
x : float
ReturnStmt
return
FloatLiteral
42.f
BinaryOperator
*
ReferenceDecl
x
ReferenceDecl
factor

Semantic Analysis

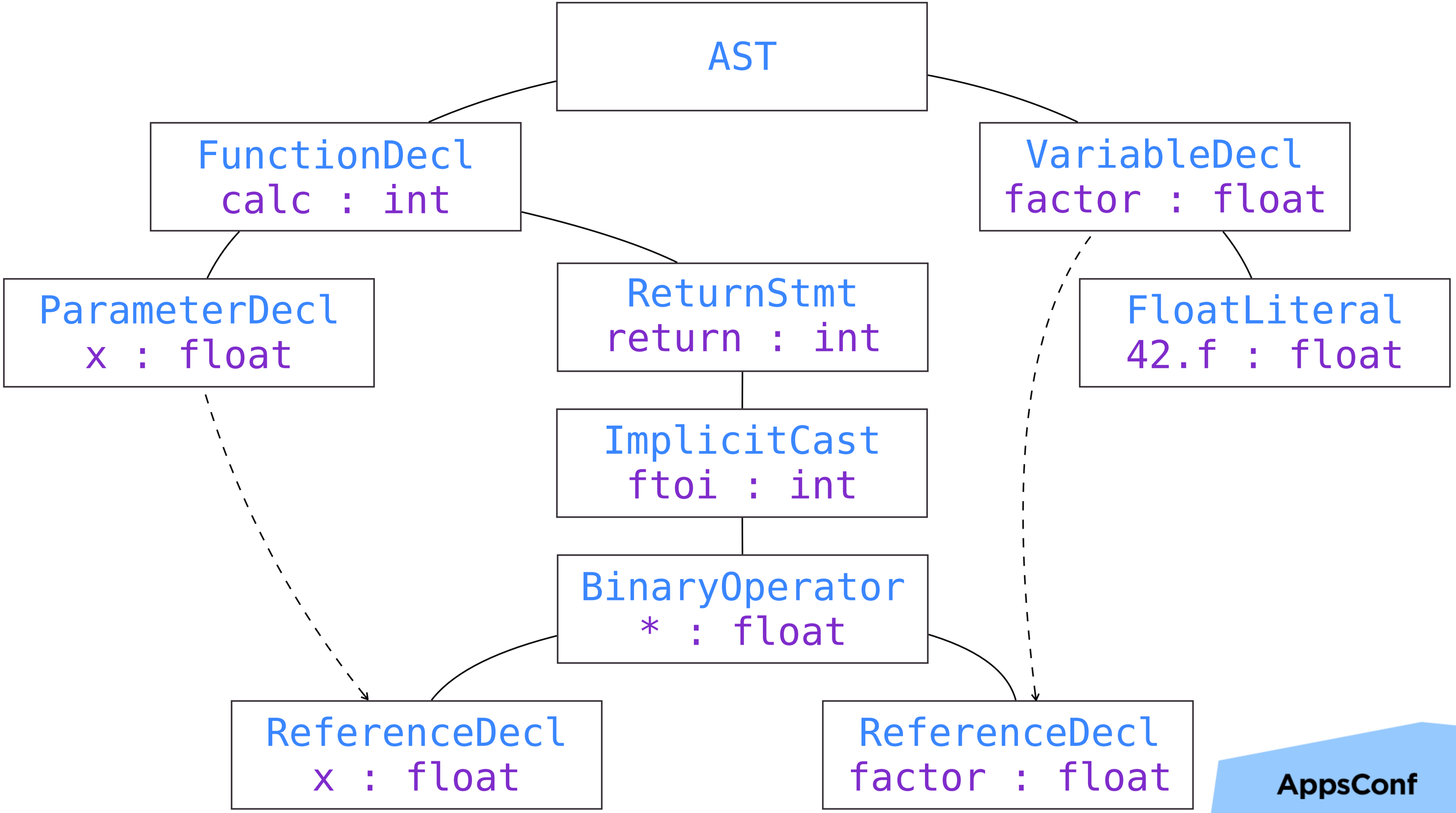

AST
FunctionDecl
calc : int
VariableDecl
factor : float
ParameterDecl
x : float
ReturnStmt
return
FloatLiteral
42.f
BinaryOperator
*
ReferenceDecl
x
ReferenceDecl
factor


AST
FunctionDecl
calc : int
VariableDecl
factor : float
ParameterDecl
x : float
ReturnStmt
return : int
FloatLiteral
42.f : float
BinaryOperator
* : float
ReferenceDecl
x : float
ReferenceDecl
factor : float

AppsConf


AST
FunctionDecl
calc : int
VariableDecl
factor : float
ParameterDecl
x : float
ReturnStmt
return : int
FloatLiteral
42.f : float
???
BinaryOperator
* : float
ReferenceDecl
x : float
ReferenceDecl
factor : float

AppsConf


AST
FunctionDecl
calc : int
VariableDecl
factor : float
ParameterDecl
x : float
ReturnStmt
return : int
FloatLiteral
42.f : float
ImplicitCast
ftoi : int
BinaryOperator
* : float
ReferenceDecl
x : float
ReferenceDecl
factor : float

Code Generation

```
@factor = constant float 42.0

define calc(float %x) {
entry:
  movf %x, %r1
  movf @factor, %r2
  %r3 = fmul %r1, %r2
  movf %r3, %r0
  ret
}
```

AppsConf

Optimization

```
@factor = constant float 42.0

define calc(float %x) {
entry:
  movf %x, %r1
  movf @factor, %r2
  %r3 = fmul %r1, %r2
  movf %r3, %r0
  ret
}
```

AppsConf

```
@factor = constant float 42.0

define calc(float %x) {
entry:
  %r0 = fmul @factor, %x
  ret
}
```

Assembler

```
_calc:
  push  {r7, lr}
  mov r7, sp
  mov r1, #36175872
  orr r1, r1, #1073741824
  bl  ___mulsf3
  bl  ___fixsfsi
  pop {r7, lr}
  mov pc, lr

  .section  __TEXT,__const
  .globl  _factor @ @factor
  .align  2
_factor:
  .long 1109917696 @ float 42
```

AppsConf

Linker

```
const float factor = 42.f;

int calc(float x) {
    return factor * x;
}

 $ clang -c calc.c -o calc.o
```

```
extern int calc(float);

int main() {
  printf("%d\n", calc(2.f));
  return 0;
}

$ clang -c main.c -o main.o
```

```
$ nm main.o
                 U _calc
0000000000000000 T _main
                 U _printf
```

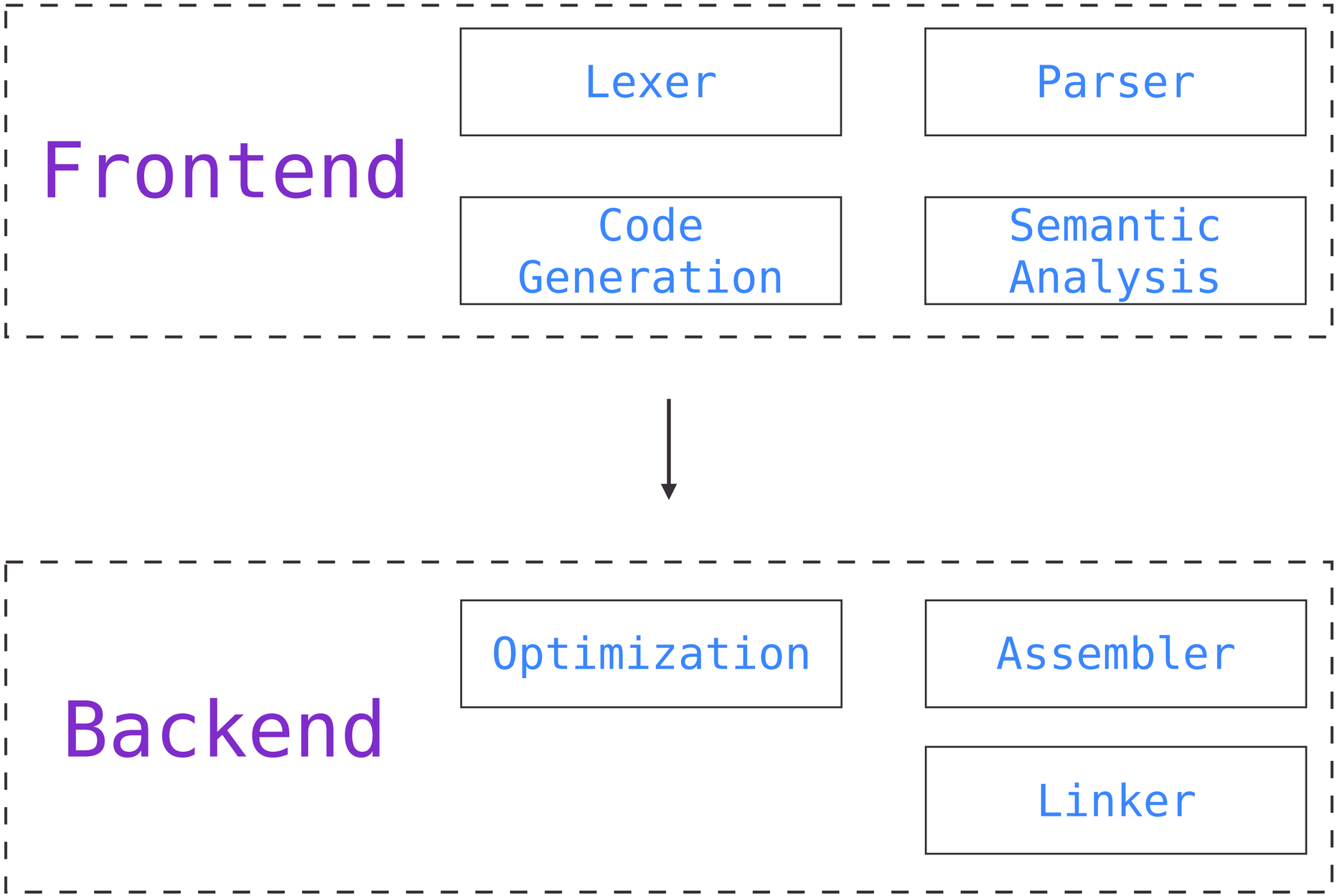
AppsConf

```
$ nm main.o
                 U _calc
0000000000000000 T _main
                 U _printf

$ ld -lc calc.o main.o -o main

$ nm main
0000000000001f30 T _calc
0000000000001fc8 S _factor
0000000000001f60 T _main
                 U _printf
```

AppsConf


Frontend
Lexer
Parser
Code Generation
Semantic Analysis
Backend
Optimization
Assembler
Linker

Applications

Application #1

UI Development

SearchResults.slim

SlimUI 〉 SlimUI 〉 SearchResults.slim 〉 No Selection

```
- unless results.empty?
  table_view
  - results.each do |result|
    cell.text = result.name
    cell.details = result.description
- else
  label
  | No items found

```

AppsConf

SearchResults.slim

SlimUI 〉 SlimUI 〉 SearchResults.slim 〉 No Selection

iPhone 6s Plus | SlimUI: **Ready** | Today at 07:33

SlimUI

SlimUI | General | Capabilities | Resource Tags | Info | Build Settings | Build Phases | Build Rules

All | Custom

Files '*.slim' using Script

Process: Source files with names matching: *.slim

Using: Custom script:

```
1 /usr/bin/slim-compiler $(INPUT_FILE_PATH)
```

Output Files	Compiler Flags
$(DERIVED_FILE_DIR)/$(INPUT_FILE_BASE).o	

AppsConf

Application #2

Mocks

User.swift

mockery 〉 mockery 〉 User.swift 〉 No Selection

```swift
class User {
  func getName() -> String {
    return "John"
  }
}

class UserConsumer {
  func doTheJob(u: User) {
    print(u.getName())
  }
}
```

AppsConf

User.swift

mockery 〉 mockery 〉 User.swift 〉 No Selection

main.swift

mockery 〉 mockery 〉 main.swift 〉 No Selection

```swift
import Foundation

let m = Mock()
          .stub(User.getName)
          .andReturn("Not a John")

let uc = UserConsumer()
uc.doTheJob(m)
```

AppsConf

User.swift

mockery 〉 mockery 〉 User.swift 〉 No Selection

main.swift

mockery 〉 mockery 〉 main.swift 〉 No Selection

```swift
import Foundation

let m = Mock()
          .stub(User.getName)
          .andReturn("Not a John")

let uc = UserConsumer()
uc.doTheJob(m as AnyObject as! User)
```

AppsConf

Application #3

Mutation Testing

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
func sum(a: Int, b: Int) -> Int {
  return a * b
}
```

AppsConf

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
func sum(a: Int, b: Int) -> Int {
  return a * b
}
```

test.swift

MutationTesting 〉 MutationTesting 〉 test.swift 〉 test()

```swift
func test() {
  let result = sum(5, b: 6)
  t_assert(result > 0)
}
```

AppsConf

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
func sum(a: Int, b: Int) -> Int {
  return a * b
}
```

test.swift

MutationTesting 〉 MutationTesting 〉 test.swift 〉 test()

```swift
func test() {
  let result = sum(5, b: 6)
  t_assert(result > 0)
}
```

AppsConf

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
func sum(a: Int, b: Int) -> Int {
  return a * b
}
```

test.swift

MutationTesting 〉 MutationTesting 〉 test.swift 〉 test()

```swift
func test() {
  let result = sum(5, b: 6)
  t_assert(result > 0)
```

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
/// sum'
func sum(a: Int, b: Int) -> Int {
  return a + b
}
```

AppsConf

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
1 func sum(a: Int, b: Int) -> Int {
2   return a * b
3 }
4
```

test.swift

MutationTesting 〉 MutationTesting 〉 test.swift 〉 test()

```swift
1 func test() {
2   let result = sum(5, b: 6)
3   t_assert(result > 0)
```

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
1 /// sum'
2 func sum(a: Int, b: Int) -> Int {
3   return a + b
4 }
5
```

testee.swift

MutationTesting 〉 MutationTesting 〉 testee.swift 〉 No Selection

```swift
1 /// sum''
2 func sum(a: Int, b: Int) -> Int {
3   return a
4 }
5
```

AppsConf

Applications

- Haml/Slim compiler
- Swift + Mocks
- Mutation Testing
- You name it!

LLVM Template

```
$ make setup
$ make build
$ open LLVMTemplate.xcodeproj
```

LLVM.xcconfig

LLVMTemplate 〉 LLVMTemplate 〉 LLVM.xcconfig 〉 No Selection

```
LLVM_SOURCE_DIR = $(HOME)/LLVM/src
LLVM_BUILD_DIR  = $(HOME)/LLVM/build

LLVM_LINKER_FLAGS = -lLLVMCore -lLLVMSupport

```

main.swift

LLVMTemplate 〉 LLVMTemplate 〉 main.swift 〉 No Selection

```swift
//
//  main.swift
//  LLVMTemplate
//

import LLVM_C

let module = sampleMathModule()
LLVMDumpModule(module)
LLVMDisposeModule(module)

```

main.swift

LLVMTemplate 〉 LLVMTemplate 〉 main.swift 〉 No Selection

```swift
//
//  main.swift
//  LLVMTemplate
//

import LLVM_C

let module = sampleMathModule()

iterateOverFunctions(module) { (function) in
    let name = LLVMGetValueName(function)
    print(String.fromCString(name)!)
}

LLVMDisposeModule(module)
```

AppsConf

main.swift

LLVMTemplate 〉 LLVMTemplate 〉 main.swift 〉 No Selection

```swift
//
//  main.swift
//  LLVMTemplate
//

import LLVM_C

let module = sampleMathModule()
let function = LLVMGetFirstFunction(module)

iterateOverInstructions(function) { (instruction) in
    LLVMDumpValue(instruction)
}

LLVMDisposeModule(module)
```

AppsConf

What’s next?

Design decisions that shaped LLVM
http://aosabook.org/en/llvm.html

Implementing a Language with LLVM
http://llvm.org/docs/tutorial/index.html

Various articles about LLVM
http://lowlevelbits.org/categories/llvm/

Kaleidoscope in Swift
https://github.com/AlexDenisov/SwiftKaleidoscope

LLVM + Swift Xcode Template
https://github.com/AlexDenisov/LLVMTemplate

Thank You!

Alex Denisov,
http://lowlevelbits.org

AppsConf