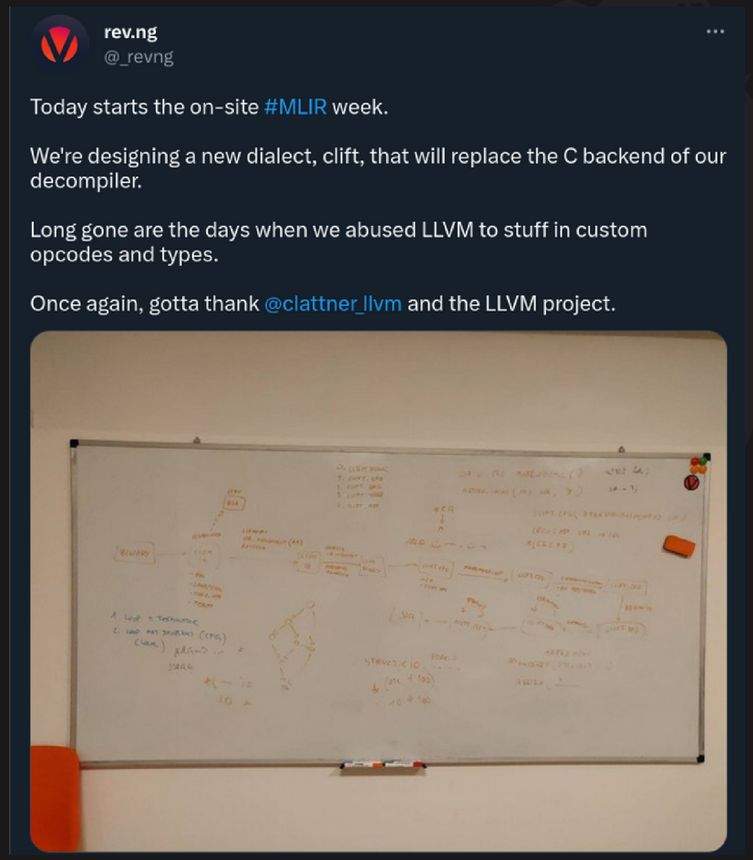
Eduardo Blázquez

EUROLLVM 2023

USING MLIR FOR DALVIK BYTECODE ANALYSIS

Motivation

Compiling Ruby with MLIR



rev.ng clift (MLIR Dialect)

Static Binary Analysis

Binary/Bytecode

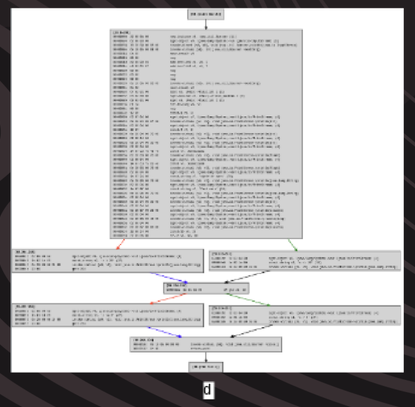
```
62000300
70200900
08006e10
0b000800
0a000000
```

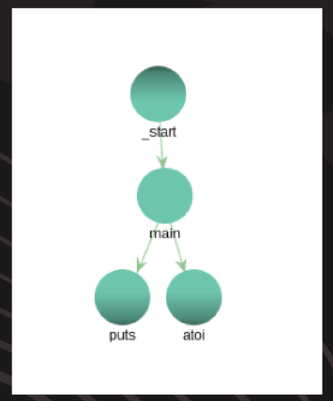
Disassembly

Baksmali

Assembly/Smali

```
00000000  new-instance v8, Scanner
00000004  sget-object v0, System;->in
00000008  invoke-direct {v8, v0}, Scanner-><init>
0000000e  invoke-virtual {v8}, Scanner->nextInt()
00000014  move-result v0
00000016  nop
```





Binary Lifting

IR Code

```
%__c1.addr = alloca ptr, align 8
%__c2.addr = alloca ptr, align 8
store ptr %__c1, ptr %__c1.addr, align 8
store ptr %__c2, ptr %__c2.addr, align 8
%0 = load ptr, ptr %__c1.addr, align 8
%1 = load i8, ptr %0, align 1
%conv = sext i8 %1 to i32
%2 = load ptr, ptr %__c2.addr, align 8
%3 = load i8, ptr %2, align 1
%conv1 = sext i8 %3 to i32
%cmp = icmp eq i32 %conv, %conv1
ret i1 %cmp
```

Decompilation

Pseudocode

```
ref = new Scanner(System.in);
iVar2 = ref.nextInt();
iVar4 = iVar2 + 10;
iVar3 = ref.nextInt();
Main.field_int = iVar3;
bVar1 = Main.field_boolean;
iVar3 = Main.field_int;
```

Static Binary Analysis

Binary/Bytecode

```
62000300
70200900
08006e10
0b000800
0a000000
```

Disassembly

Baksmali

Assembly/Smali

```
00000000  new-instance v8, Scanner
00000004  sget-object v0, System;->in
00000008  invoke-direct {v8, v0}, Scanner-><init>
0000000e  invoke-virtual {v8}, Scanner->nextInt()
00000014  move-result v0
00000016  nop
```

Binary Lifting

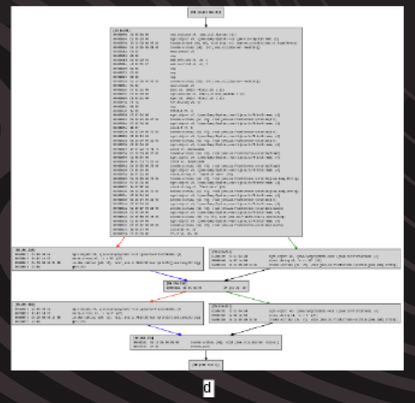
IR Code

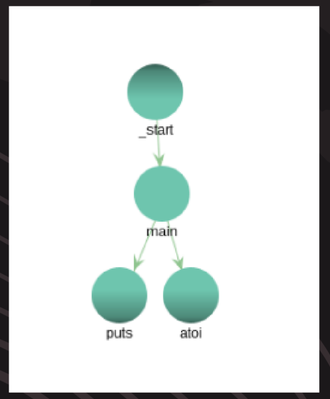
```
%__c1.addr = alloca ptr, align 8
%__c2.addr = alloca ptr, align 8
store ptr %__c1, ptr %__c1.addr, align 8
store ptr %__c2, ptr %__c2.addr, align 8
%0 = load ptr, ptr %__c1.addr, align 8
%1 = load i8, ptr %0, align 1
%conv = sext i8 %1 to i32
%2 = load ptr, ptr %__c2.addr, align 8
%3 = load i8, ptr %2, align 1
%conv1 = sext i8 %3 to i32
%cmp = icmp eq i32 %conv, %conv1
ret i1 %cmp
```

Decompilation

Pseudocode

```
ref = new Scanner(System.in);
iVar2 = ref.nextInt();
iVar4 = iVar2 + 10;
iVar3 = ref.nextInt();
Main.field_int = iVar3;
bVar1 = Main.field_boolean;
iVar3 = Main.field_int;
```





- Visual representation
- Useful for analysts
- Needs from patterns and heuristics

Static Binary Analysis

Binary/Bytecode

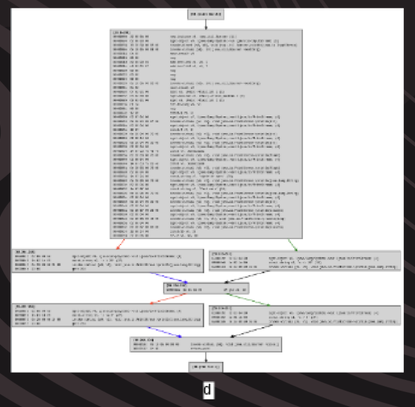
```
62000300
70200900
08006e10
0b000800
0a000000
```

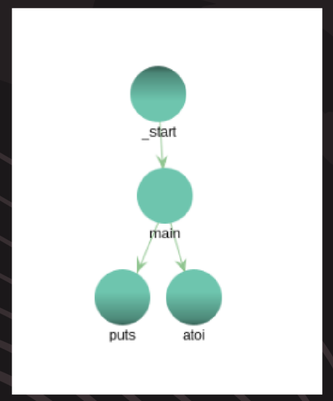
Disassembly

Baksmali

Assembly/Smali

```
00000000  new-instance v8, Scanner
00000004  sget-object v0, System;->in
00000008  invoke-direct {v8, v0}, Scanner-><init>
0000000e  invoke-virtual {v8}, Scanner->nextInt()
00000014  move-result v0
00000016  nop
```





Binary Lifting

IR Code

```
%__c1.addr = alloca ptr, align 8
%__c2.addr = alloca ptr, align 8
store ptr %__c1, ptr %__c1.addr, align 8
store ptr %__c2, ptr %__c2.addr, align 8
%0 = load ptr, ptr %__c1.addr, align 8
%1 = load i8, ptr %0, align 1
%conv = sext i8 %1 to i32
%2 = load ptr, ptr %__c2.addr, align 8
%3 = load i8, ptr %2, align 1
%conv1 = sext i8 %3 to i32
%cmp = icmp eq i32 %conv, %conv1
ret i1 %cmp
```

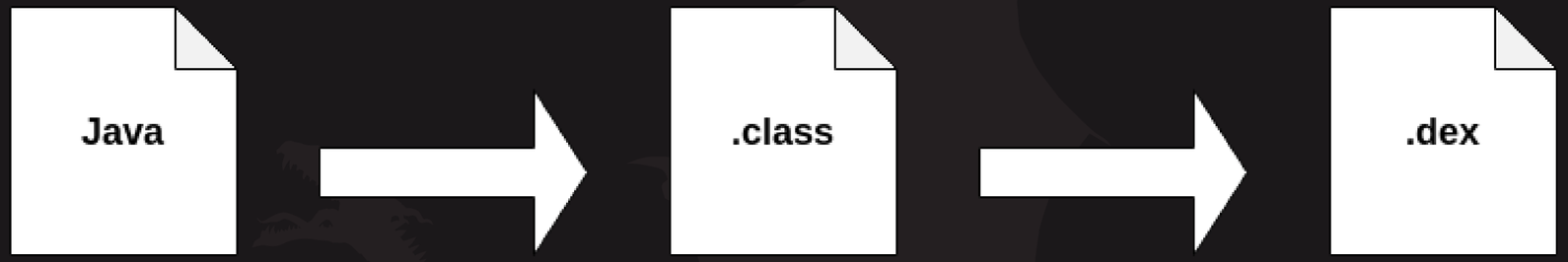
Decompilation

Pseudocode

```
ref = new Scanner(System.in);
iVar2 = ref.nextInt();
iVar4 = iVar2 + 10;
iVar3 = ref.nextInt();
Main.field_int = iVar3;
bVar1 = Main.field_boolean;
iVar3 = Main.field_int;
```

- It can be represented in SSA form
- Easier analysis than assembly
- Many existing and useful frameworks for working with IR (e.g., llvm IR)
- Topic of this talk

Dalvik Bytecode



- Dalvik is a virtual machine register-based (64k virtual registers of 32 bits)
- Adjacent registers can be used for representing 64 bit values
- References to strings, fields, methods and classes are done through index values
- Pseudo-instructions and Instructions with side effects handled (e.g., exceptions) by the ART Runtime
- Virtual Registers do not have type specified, this can be inferred from instruction (represented by a suffix) or by the instruction type.
- More than 270 operation codes for the different instructions from Dalvik

Dalvik Bytecode

```
double Main->test(int)
BB-Start Block
BB-0
00000000  00 00                    nop
00000002  da 05 05 28              mul-int/lit8 v5, v5, 40
0000000a  52 45 02 00              iget v5, v4, LMain;->r I (2)
0000000e  83 50                    int-to-double v0, v5
0000001a  71 20 03 00 10 00        invoke-static {v0, v1}, boolean java.lang.Double->isNaN(double)
00000024  5a 40 00 00              iput-wide v0, v4, LMain;->d D (0)
00000028  62 05 03 00              sget-object v5, Ljava/lang/System;->out Ljava/io/PrintStream; (3)
00000036  1a 01 0f 00              const-string v1, "The result is equals to: " (15)
00000044  71 20 04 00 21 00        invoke-static {v1, v2}, java.lang.String
                                   java.lang.Double->toString(double)
0000004a  0c 01                    move-result-object v1
00000052  6e 10 08 00 00 00        invoke-virtual {v0}, java.lang.String
                                   java.lang.StringBuilder->toString()
00000058  0c 00                    move-result-object v0
00000064  10 00                    return-wide v0
BB-End Block
```

Dalvik Bytecode

```
double Main->test(int)
BB-Start Block
BB-0
00000000  00 00                    nop
00000002  da 05 05 28              mul-int/lit8 v5, v5, 40
0000000a  52 45 02 00              iget v5, v4, LMain;->r I (2)
0000000e  83 50                    int-to-double v0, v5
0000001a  71 20 03 00 10 00        invoke-static {v0, v1}, boolean java.lang.Double->isNaN(double)
00000024  5a 40 00 00              iput-wide v0, v4, LMain;->d D (0)
00000028  62 05 03 00              sget-object v5, Ljava/lang/System;->out Ljava/io/PrintStream; (3)
00000036  1a 01 0f 00              const-string v1, "The result is equals to: " (15)
00000044  71 20 04 00 21 00        invoke-static {v1, v2}, java.lang.String
                                   java.lang.Double->toString(double)
0000004a  0c 01                    move-result-object v1
00000052  6e 10 08 00 00 00        invoke-virtual {v0}, java.lang.String
                                   java.lang.StringBuilder->toString()
00000058  0c 00                    move-result-object v0
00000064  10 00                    return-wide v0
BB-End Block
```

Dalvik Bytecode

```
double Main->test(int)
BB-Start Block
BB-0
00000000  00 00                    nop
00000002  da 05 05 28              mul-int/lit8 v5, v5, 40
0000000a  52 45 02 00              iget v5, v4, LMain;->r I (2)
0000000e  83 50                    int-to-double v0, v5
0000001a  71 20 03 00 10 00        invoke-static {v0, v1}, boolean java.lang.Double->isNaN(double)
00000024  5a 40 00 00              iput-wide v0, v4, LMain;->d D (0)
00000028  62 05 03 00              sget-object v5, Ljava/lang/System;->out Ljava/io/PrintStream; (3)
00000036  1a 01 0f 00              const-string v1, "The result is equals to: " (15)
00000044  71 20 04 00 21 00        invoke-static {v1, v2}, java.lang.String
                                   java.lang.Double->toString(double)
0000004a  0c 01                    move-result-object v1
00000052  6e 10 08 00 00 00        invoke-virtual {v0}, java.lang.String
                                   java.lang.StringBuilder->toString()
00000058  0c 00                    move-result-object v0
00000064  10 00                    return-wide v0
BB-End Block
```

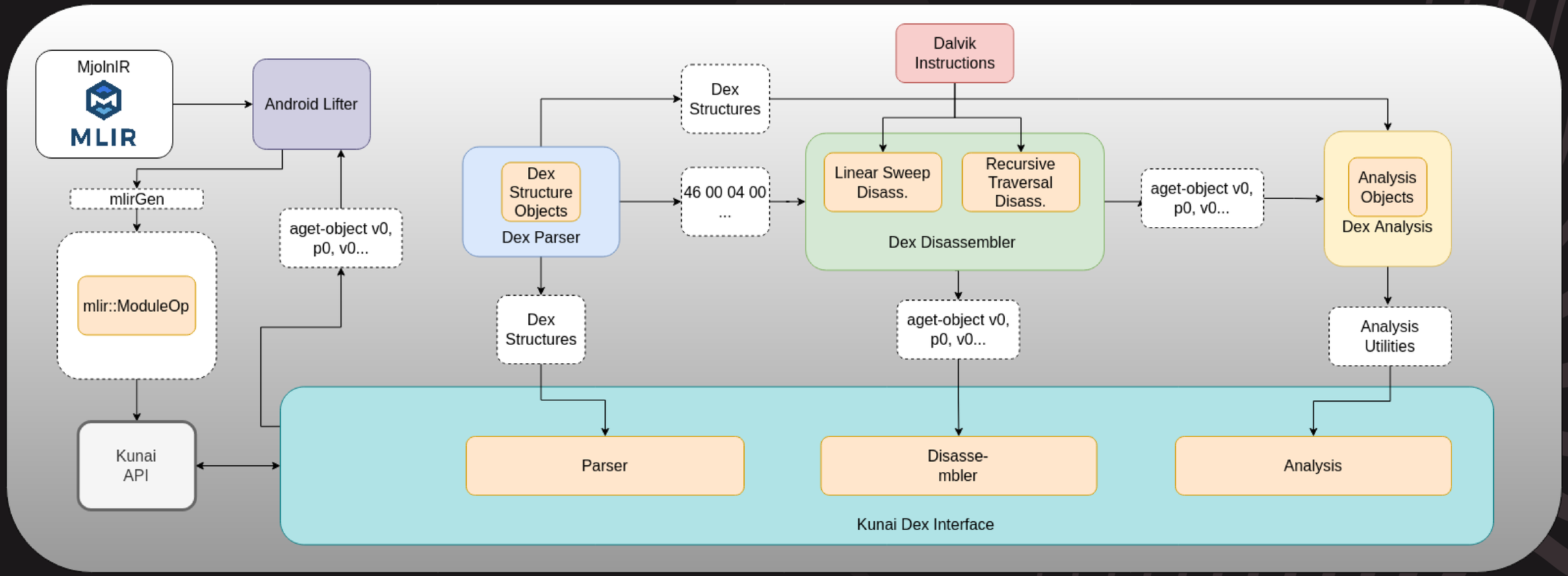
Dalvik Bytecode

```
double Main->test(int)
BB-Start Block
BB-0
00000000  00 00                    nop
00000002  da 05 05 28              mul-int/lit8 v5, v5, 40
0000000a  52 45 02 00              iget v5, v4, LMain;->r I (2)
0000000e  83 50                    int-to-double v0, v5
0000001a  71 20 03 00 10 00        invoke-static {v0, v1}, boolean java.lang.Double->isNaN(double)
00000024  5a 40 00 00              iput-wide v0, v4, LMain;->d D (0)
00000028  62 05 03 00              sget-object v5, Ljava/lang/System;->out Ljava/io/PrintStream; (3)
00000036  1a 01 0f 00              const-string v1, "The result is equals to: " (15)
00000044  71 20 04 00 21 00        invoke-static {v1, v2}, java.lang.String
                                   java.lang.Double->toString(double)
0000004a  0c 01                    move-result-object v1
00000052  6e 10 08 00 00 00        invoke-virtual {v0}, java.lang.String
                                   java.lang.StringBuilder->toString()
00000058  0c 00                    move-result-object v0
00000064  10 00                    return-wide v0
BB-End Block
```

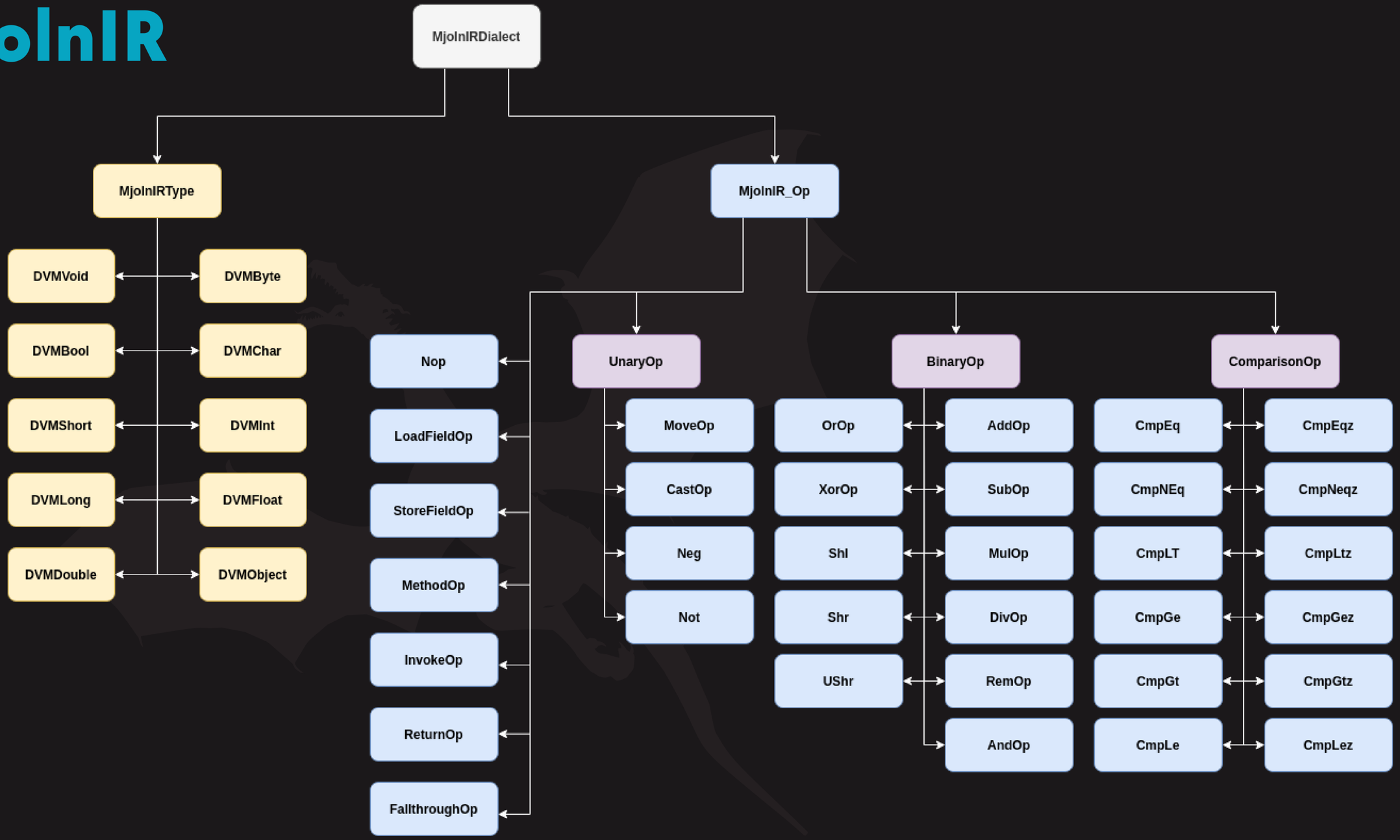
MLIR

Bytecode Analysis → IR Analysis

Kunai


MjolnIR
MLIR
Android Lifter
mlirGen
mlir::ModuleOp
Kunai API
aget-object v0, p0, v0...
Dex Structure Objects
Dex Parser
Dex Structures
Dex Structures
46 00 04 00 ...
Dalvik Instructions
Linear Sweep Disass.
Recursive Traversal Disass.
Dex Disassembler
aget-object v0, p0, v0...
aget-object v0, p0, v0...
Analysis Objects
Dex Analysis
Analysis Utilities
Parser
Disasse-mbler
Analysis
Kunai Dex Interface

MjolnIR


MjolnIRDialect
MjolnIRType
MjolnIR_Op
DVMVoid
DVMByte
DVMBool
DVMChar
DVMShort
DVMInt
DVMLong
DVMFloat
DVMDouble
DVMObject
Nop
LoadFieldOp
StoreFieldOp
MethodOp
InvokeOp
ReturnOp
FallthroughOp
UnaryOp
MoveOp
CastOp
Neg
Not
BinaryOp
OrOp
XorOp
Shl
Shr
UShr
AddOp
SubOp
MulOp
DivOp
RemOp
AndOp
ComparisonOp
CmpEq
CmpNEq
CmpLT
CmpGe
CmpGt
CmpLe
CmpEqz
CmpNeqz
CmpLtz
CmpGez
CmpGtz
CmpLez

```
BB-Start Block
BB-0
00000000  90 00 02 03            add-int v0, v2, v3
00000004  92 02 02 00            mul-int v2, v2, v0
00000008  b3 32                  div-int/2addr v2, v3
0000000a  58 13 00 00            iget-short v3, v1, LSimple;->test_field S (0)
0000000e  5f 13 00 00            iput-short v3, v1, LSimple;->test_field S (0)
00000012  00 00                  nop
00000014  33 20 04 00            if-ne v0, v2, 4
BB-18
00000018  01 02                  move v2, v0
0000001a  28 02                  goto 30
BB-1c
0000001c  00 00                  nop
BB-1e
0000001e  00 00                  nop
00000020  0f 02                  return v2
BB-End Block
```

Dalvik

MjolnIR

```
module @"LSimple;" {
  MjolnIR.method @my_add(%arg0: !MjolnIR.dvmint, %arg1: !MjolnIR.dvmint) -> !MjolnIR.dvmint {
    %0 = "MjolnIR.add"(%arg0, %arg1) : (!MjolnIR.dvmint, !MjolnIR.dvmint) -> !MjolnIR.dvmint
    %1 = "MjolnIR.mul"(%arg0, %0) : (!MjolnIR.dvmint, !MjolnIR.dvmint) -> !MjolnIR.dvmint
    %2 = "MjolnIR.div"(%1, %arg1) : (!MjolnIR.dvmint, !MjolnIR.dvmint) -> !MjolnIR.dvmint
    %3 = MjolnIR.loadfield @"LSimple;" -> @test_field(0) : !MjolnIR.dvmshort
    MjolnIR.storefield %3 : !MjolnIR.dvmshort, @"LSimple;" -> @test_field(0)
    "MjolnIR.nop"() : () -> ()
    %4 = "MjolnIR.cmp-neq"(%0, %2) : (!MjolnIR.dvmint, !MjolnIR.dvmint) -> i1
    cf.cond_br %4, ^bb2(%2 : !MjolnIR.dvmint), ^bb1(%0 : !MjolnIR.dvmint)
  ^bb1(%5: !MjolnIR.dvmint):  // pred: ^bb0
    %6 = "MjolnIR.move"(%5) : (!MjolnIR.dvmint) -> !MjolnIR.dvmint
    cf.br ^bb3(%6 : !MjolnIR.dvmint)
  ^bb2(%7: !MjolnIR.dvmint):  // pred: ^bb0
    "MjolnIR.nop"() : () -> ()
    MjolnIR.fallthrough ^bb3(%7 : !MjolnIR.dvmint)
  ^bb3(%8: !MjolnIR.dvmint):  // 2 preds: ^bb1, ^bb2
    "MjolnIR.nop"() : () -> ()
    MjolnIR.return %8 : !MjolnIR.dvmint
  }
}
```


HTTPS://GITHUB.COM/FARE9/KUNAI
-STATIC-
ANALYZER/TREE/REFACTORING/KUN
AI-LIB/MJOLNIR

TRUST aWARE

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101021377

uc3m | Universidad Carlos III de Madrid

THANK YOU

Eduardo Blázquez
eduardo.blazquez@uc3m.es

Special Thanks to:
- **Alex Denisov** for his help with Dialect and the SSA algorithm.
- To all those who give of their time to offer **office hours**
- **MLIR community and discord channel** for the answers to my questions.

https://www.linkedin.com/in/eduardo-blazquez-gonz/ https://twitter.com/Farenain