Case Study on LLVM as suitable intermediate language for binary analysis

Florian Märkl
Technische Universität München

Wintersemester 2016/17
7. Februar 2017

ESIL

REIL

BAP

Binary Ninja's IL

VEX

MAIL

TCG

Technische Universität München

ESIL

REIL

BAP

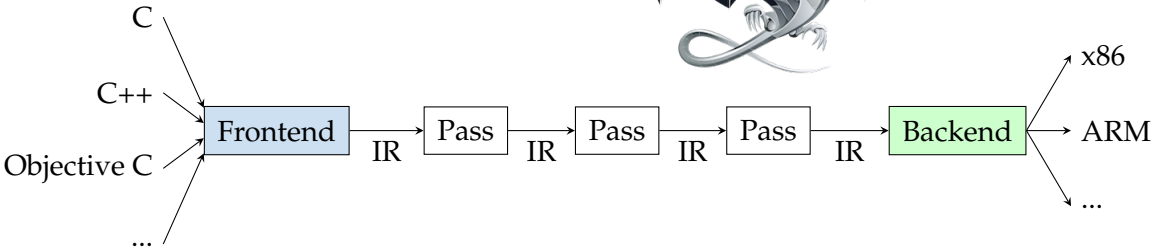
Binary Ninja's IL

Why not try LLVM IR?

VEX

MAIL

TCG


C
C++
Objective C
...
Frontend
IR
Pass
IR
Pass
IR
Pass
IR
Backend
x86
ARM
...

```
1  @.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"
2
3  define i32 @main()
4  {
5    %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0
6
7    call i32 @puts(i8* %cast210)
8    ret i32 0
9  }
10
11 declare i32 @puts(i8* nocapture) nounwind
12
13 !0 = !{i32 42, null, !"string"}
14 !foo = !{!0}
```

```
1  @.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"
2
3  define i32 @main()
4  {
5    %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0
6
7    call i32 @puts(i8* %cast210)
8    ret i32 0
9  }
10
11 declare i32 @puts(i8* nocapture) nounwind
12
13 !0 = !{i32 42, null, !"string"}
14 !foo = !{!0}
```

Global variable, starting with @

```
1  @.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"
2
3  define i32 @main()
4  {
5    %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0
6
7    call i32 @puts(i8* %cast210)
8    ret i32 0
9  }
10
11 declare i32 @puts(i8* nocapture) nounwind
12
13 !0 = !{i32 42, null, !"string"}
14 !foo = !{!0}
```

Function with return value

```
1  @.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"
2
3  define i32 @main()
4  {
5    %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0
6
7    call i32 @puts(i8* %cast210)
8    ret i32 0
9  }
10
11 declare i32 @puts(i8* nocapture) nounwind
12
13 !0 = !{i32 42, null, !"string"}
14 !foo = !{!0}
```

Assign local variable, starting with % (SSA)

```
1  @.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"
2
3  define i32 @main()
4  {
5    %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0
6
7    call i32 @puts(i8* %cast210)
8    ret i32 0
9  }
10
11 declare i32 @puts(i8* nocapture) nounwind
12
13 !0 = !{i32 42, null, !"string"}
14 !foo = !{!0}
```

Call function with parameter

```
1  @.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"
2
3  define i32 @main()
4  {
5    %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0
6
7    call i32 @puts(i8* %cast210)
8    ret i32 0
9  }
10
11 declare i32 @puts(i8* nocapture) nounwind
12
13 !0 = !{i32 42, null, !"string"}
14 !foo = !{!0}
```

Declaration of external function

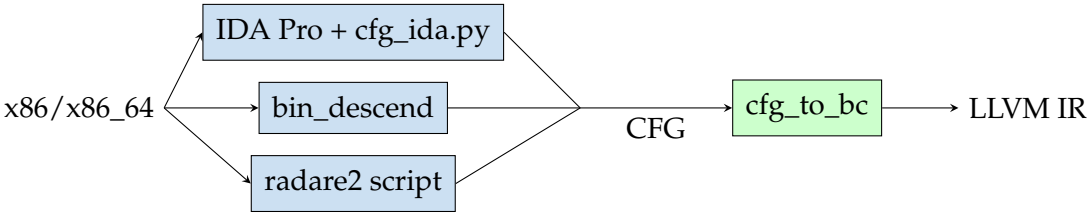
```
1  @.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"
2
3  define i32 @main()
4  {
5    %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0
6
7    call i32 @puts(i8* %cast210)
8    ret i32 0
9  }
10
11 declare i32 @puts(i8* nocapture) nounwind
12
13 !0 = !{i32 42, null, !"string"}
14 !foo = !{!0}
```

Some metadata

```
@.str = private unnamed_addr constant [13 x i8] c"hello world\0A\00"

define i32 @main()
{
  %cast210 = getelementptr [13 x i8], [13 x i8]* @.str, i64 0, i64 0

  call i32 @puts(i8* %cast210)
  ret i32 0
}

declare i32 @puts(i8* nocapture) nounwind

!0 = !{i32 42, null, !"string"}
!foo = !{!0}
```

Much more high-level than e.g. VEX!

```
1  @str = private unnamed_addr constant [52 x i8] c"higher or equal.\00"
2  @str2 = private unnamed_addr constant [27 x i8] c"higher.\00"
3
4  declare i32 @puts(i8* nocapture readonly) #1
5
6  define void @nil_recurring(i32 %a, i32 %b) #0 {
7    entry:
8      %cmp = icmp sgt i32 %a, %b
9      br i1 %cmp, label %if.then, label %if.else
10
11   if.then:
12     %puts2 = tail call i32 @puts(i8* getelementptr inbounds ([27 x i8]* @str2, i64 0, i64 0))
13     br label %if.end
14
15   if.else:
16     %puts = tail call i32 @puts(i8* getelementptr inbounds ([52 x i8]* @str, i64 0, i64 0))
17     br label %if.end
18
19   if.end:
20     ret void
21 }
```


IDA Pro + cfg_ida.py
x86/x86_64
bin_descend
radare2 script
CFG
cfg_to_bc
LLVM IR

```
internal_funcs {
  blocks {
    insts {
      inst_bytes: "U"
      inst_addr: 134217728
      inst_len: 1
    }
    ...
    base_address: 134217728
  }
  entry_address: 134217728
  symbol_name: "deadwing"
}
module_name: "demo_test.o"
entries {
  entry_name: "deadwing"
  entry_address: 134217728
  entry_extra {
    entry_argc: 0
    entry_cconv: CalleeCleanup
    does_return: true
  }
}
```

```
%RegState = type <{ i32, i32, i32, i32, i32,
      i32, i32, i32, i32, i8, i8, i8, i8, i8,
      i8, i8, [8 x x86_fp80], i8, i8, i8, i8,
      i8, i8, i8, i8, i8, i8, i8, i8, i8, i8,
      i8, i8, i8, i8, i8, i8, i8, i8, i8, [8 x
       i8], i16, i32, i16, i32, i16, i128,
      i128, i128, i128, i128, i128, i128, i128
      , i128, i128, i128, i128, i128, i128,
      i128, i128, i32, i32 }>
```

=

```
typedef uint32_t reg_t;

typedef struct _RegState
{
  reg_t EIP;
  reg_t EAX;
  reg_t EBX;
  reg_t ECX;
  reg_t EDX;
  reg_t ESI;
  reg_t EDI;
  reg_t ESP;
  reg_t EBP;

  // ... more registers following ...

} __attribute__((packed)) RegState;
```

```
example:
    add eax, 1
    ret
```

```
define internal void @example(%RegState*) #1
{
  %XAX = getelementptr %RegState* %0, i32 0, i32 1, !mcsema_real_eip !0

  %EAX_val.0 = load i32* %XAX, align 4, !mcsema_real_eip !0

  %uadd = tail call { i32, i1 }  @llvm.uadd.with.overflow.i32(i32 %EAX_val.0, i32 1)
  %1 = extractvalue { i32, i1 } %uadd, 0

  store i32 %1, i32* %XAX, align 4, !mcsema_real_eip !0

  ret void, !mcsema_real_eip !1
}
```

Assembly, C, ...

Lifted LLVM Code

External Functions


__mcsema_attach_call
Assembly, C, ...
Lifted LLVM Code
External Functions
__mcsema_detach_ret


__mcsema_attach_call
__mcsema_detach_call
Assembly, C, ...
Lifted LLVM Code
External Functions
__mcsema_detach_ret
__mcsema_attach_ret

```
module asm "  .globl sub_8000000;"
module asm "  .globl deadwing;"
module asm "  .type deadwing,@function"
module asm "deadwing:"
module asm "  .cfi_startproc;"
module asm "  pushl $sub_8000000;"
module asm "  jmp __mcsema_attach_call_cdecl;"
module asm "0:"
module asm "  .size deadwing,0b-deadwing;"
module asm "  .cfi_endproc;"


define void @sub_8000000(%RegState*) #1 {
  ; ... some code ...
}
```

Demo

Future Outlook

Technische Universität München

Passes to make code more readable?

Decompiler?

ARM!

Binary Analysis framework?

radare2 for CFG recovery?

Th4nk5 f0r l15t3n1ng!