SLEIGH Disassembler Backend

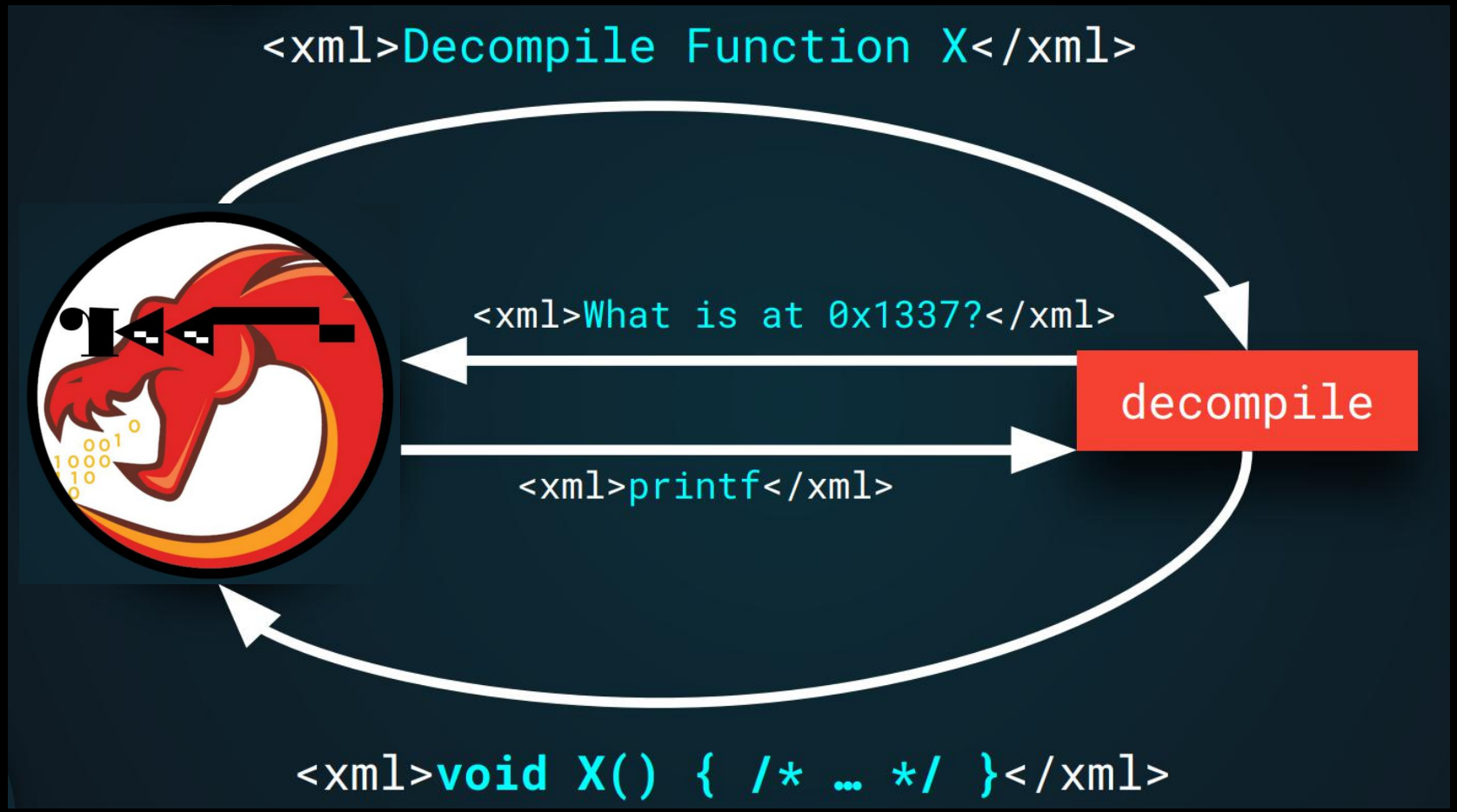
Bridge from SLEIGH to Radare2

Jiaxiang Zhou | @FXTi
September 4, 2020
r2con - Online

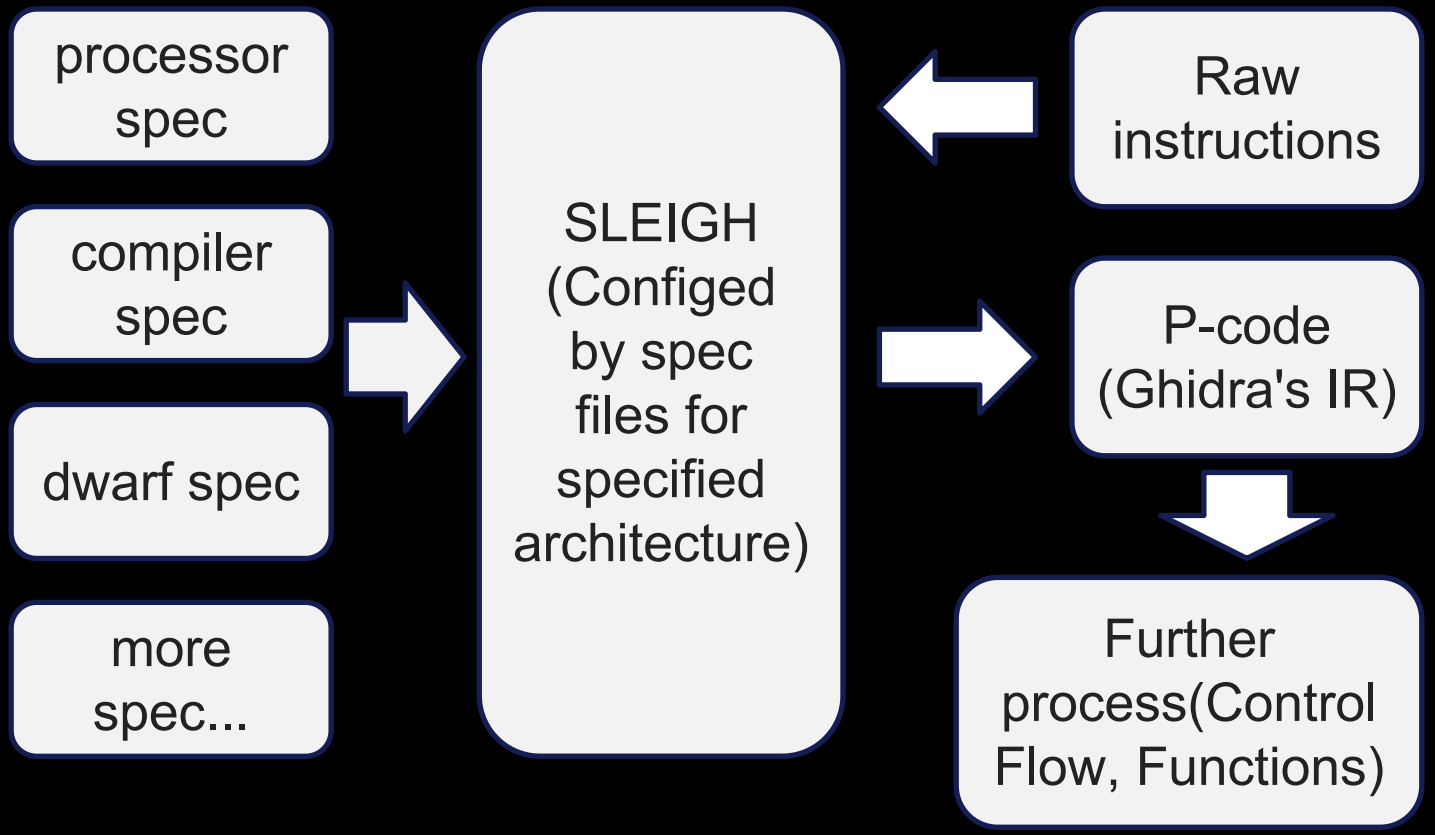
About Me

- FXTi on Github and Telegram
- Just finish 3rd year CS major in CUMT
- Refactor pyc plugin and r_big in R2
- Likes compilers and decompilers

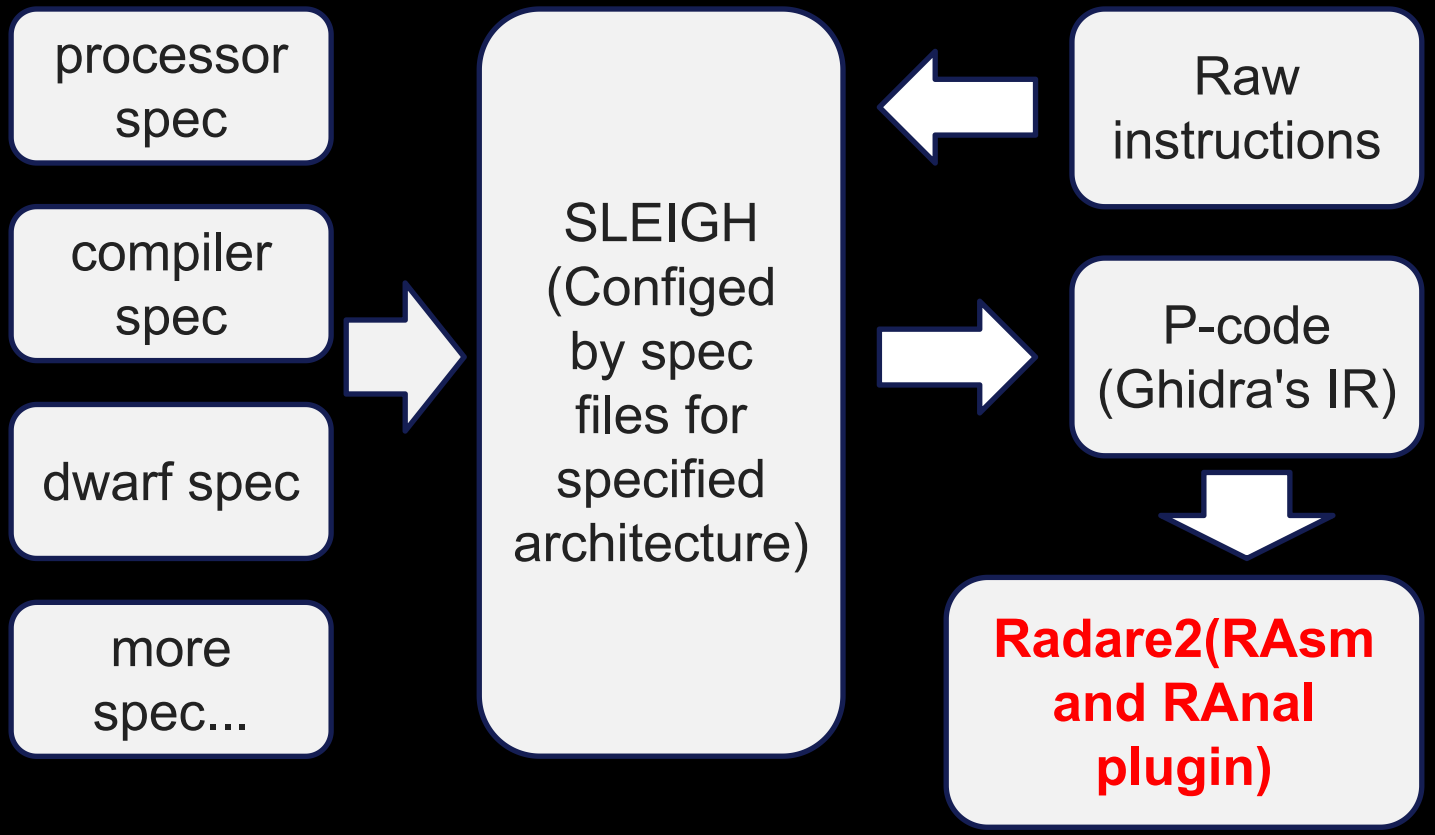
r2ghidra-dec



Decompiler in C++



What do we want?



Why?

```
Processors [master *] % pwd;ll
/home/fx-ti/git/ghidra/Ghidra/Processors
total 116K
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 6502
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 68000
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 6805
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 8048
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 8051
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 8085
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 AARCH64
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 ARM
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 Atmel
drwxr-xr-x 6 fx-ti fx-ti 4.0K Jul 27 13:57 CR16
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 Dalvik
drwxr-xr-x 8 fx-ti fx-ti 4.0K Jul 27 13:59 DATA
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 HCS08
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 HCS12
drwxr-xr-x 8 fx-ti fx-ti 4.0K Jul 27 13:59 JVM
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 MCS96
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 MIPS
drwxr-xr-x 6 fx-ti fx-ti 4.0K Jul 27 13:57 PA-RISC
drwxr-xr-x 8 fx-ti fx-ti 4.0K Jul 27 13:59 PIC
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 PowerPC
drwxr-xr-x 8 fx-ti fx-ti 4.0K Jul 27 13:59 RISCV
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 Sparc
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 SuperH
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 SuperH4
drwxr-xr-x 5 fx-ti fx-ti 4.0K Jul 27 13:57 TI_MSP430
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 Toy
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 tricore
drwxr-xr-x 7 fx-ti fx-ti 4.0K Jul 27 13:59 x86
drwxr-xr-x 6 fx-ti fx-ti 4.0K Jul 27 13:57 Z80
```

Disassemble

Dive into spec file parsing process and export all interfaces needed

- Registers
- Disassemble

Disassemble

```
[0x080483d0]> pdga
[0x080483d0]> pd 50
            ;-- entry0:
            ;-- section..text:
            ;-- .text:
            ;-- _start:
            ;-- eip:
            0x080483d0      31ed            XOR ebp,ebp                 ; [13
.text
            0x080483d2      5e              POP esi
            0x080483d3      89e1            MOV ecx,esp
            0x080483d5      83e4f0          AND esp,0xfffffff0
            0x080483d8      50              PUSH eax
            0x080483d9      54              PUSH esp
            0x080483da      52              PUSH edx
            0x080483db      6810860408      PUSH sym.__libc_csu_fini
            0x080483e0      68a0850408      PUSH sym.__libc_csu_init
            0x080483e5      51              PUSH ecx
            0x080483e6      56              PUSH esi
            0x080483e7      6840850408      PUSH main
            0x080483ec      e873ffffff      CALL sym.imp.__libc_start_main
            0x080483f1      f4              HLT
            0x080483f2      90              NOP
```

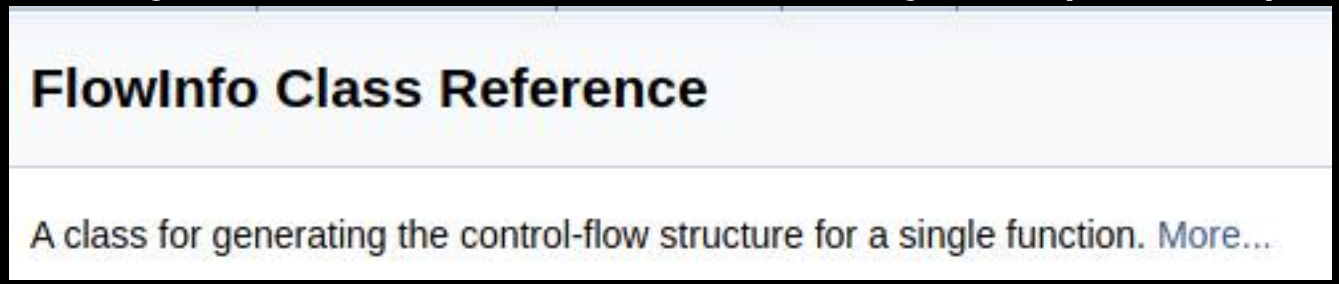
Analysis

- Extract control flow from P-code
- Indentify type of instruction
- Translate P-code into ESIL

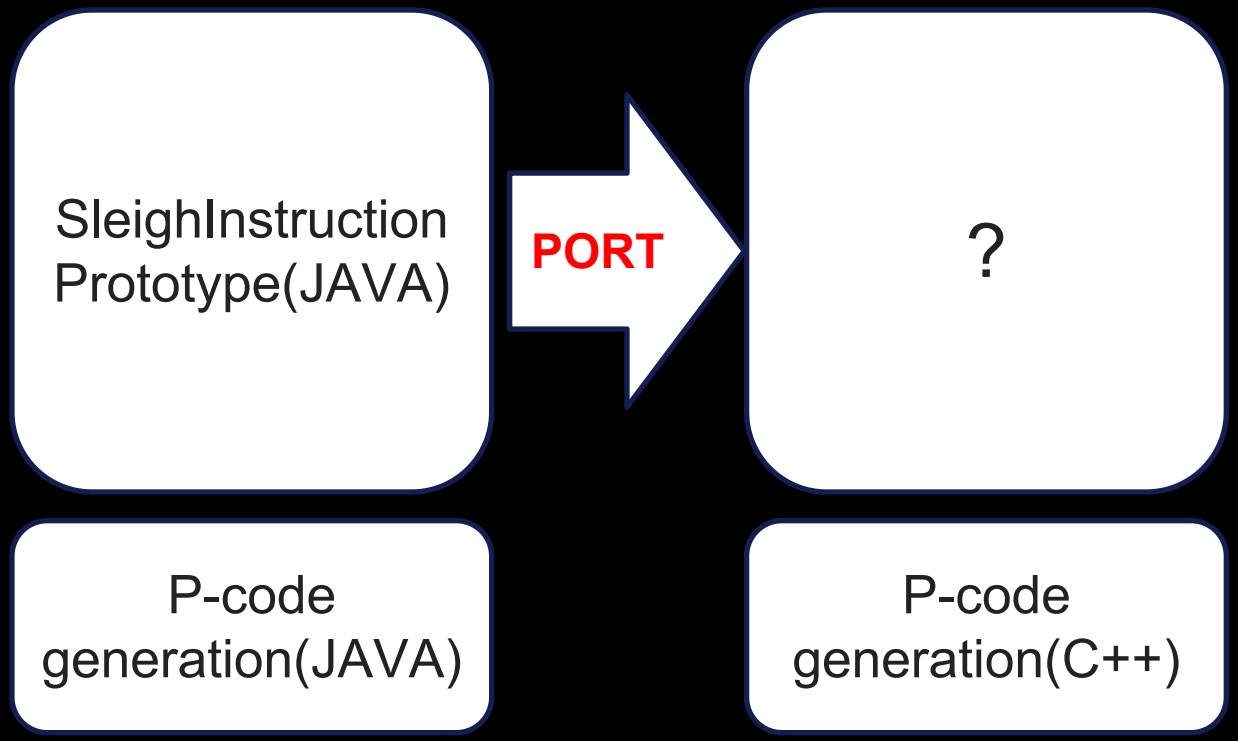
Extract control flow from P-code

- SleighInstuctionPrototype(JAVA!)

- Only function-level analysis(C++!)



Extract control flow from P-code



Extract control flow from P-code

Showing 2 changed files with 3 additions and 0 deletions. Unified Split

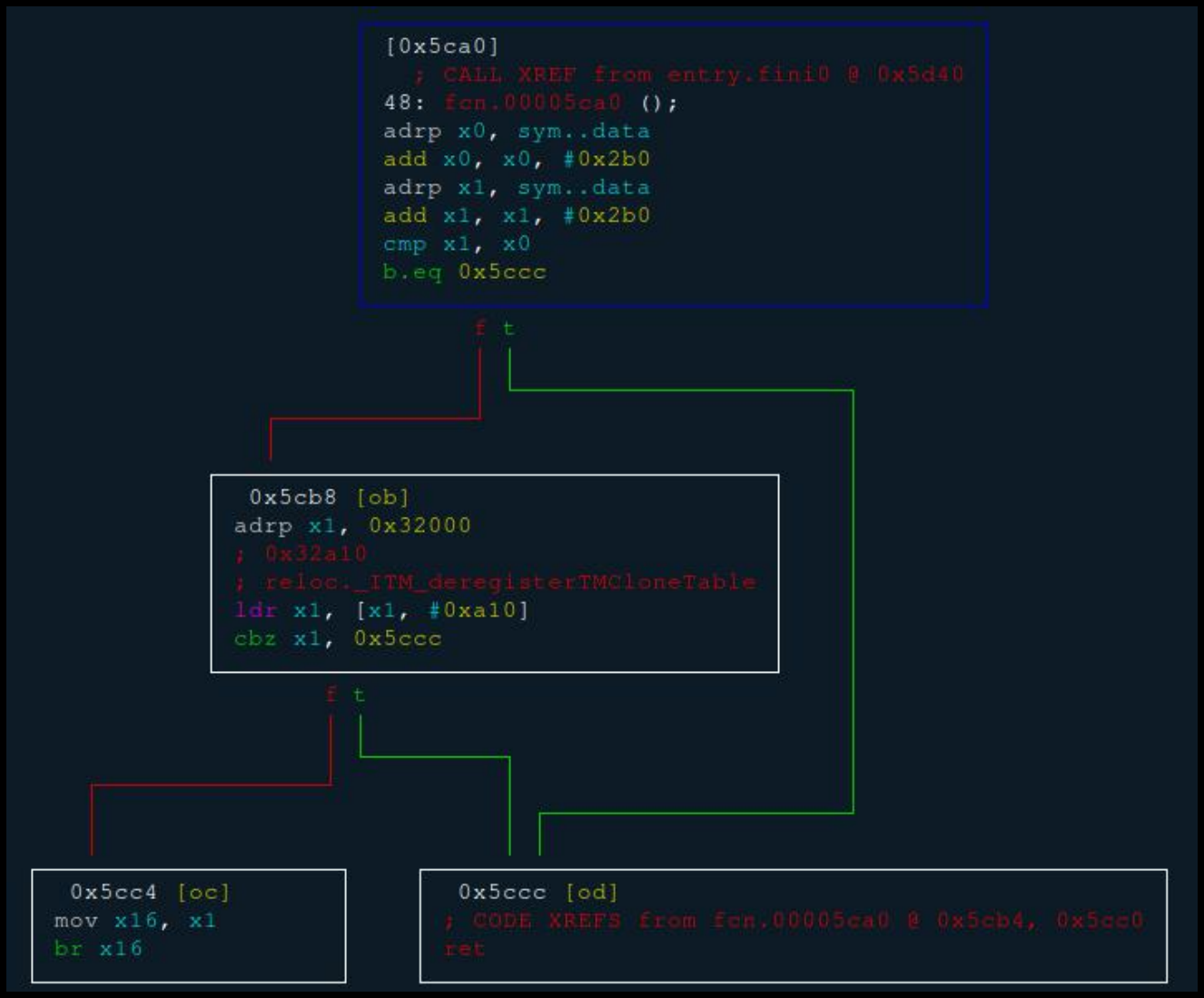
2 Ghidra/Features/Decompiler/src/decompile/cpp/context.hh

```
@@ -89,6 +89,8 @@ private:
    ConstructState *base_state;
    int4 alloc;                // Number of ConstructState's allocated
    int4 delayslot;            // delayslot depth
+ protected:
+   ConstructState **getBaseState(void) { return &base_state; }
  public:
    ParserContext(ContextCache *ccache);
    ~ParserContext(void) { if (context != (uintm *)0) delete [] context; }
```

1 Ghidra/Features/Decompiler/src/decompile/cpp/sleigh.hh

```
@@ -113,6 +113,7 @@ protected:
    ParserContext *obtainContext(const Address &addr,int4 state) const;
    void resolve(ParserContext &pos) const;
    void resolveHandles(ParserContext &pos) const;
+   ContextCache *getContextCache(void) { return cache; }
  public:
    Sleigh(LoadImage *ld,ContextDatabase *c_db);
    virtual ~Sleigh(void);
```

Extract control flow from P-code



Indentify type of instruction

```
[0x00000000]> wx d1f8
[0x00000000]> pdga
[0x00000000]> pdgsd 1
0x00000000: SAR EAX,1
    (unique,0x17d0,1) = COPY 0x1
    (unique,0xa400,4) = INT_AND EAX, 0x1
    CF = INT_NOTEQUAL (unique,0xa400,4), 0x0
    OF = COPY 0x0
    EAX = INT_SRIGHT EAX, 0x1
    RAX = INT_ZEXT EAX
    SF = INT_SLESS EAX, 0x0
    ZF = INT_EQUAL EAX, 0x0
    (unique,0x2560,4) = INT_AND EAX, 0xff
    (unique,0x2570,1) = POPCOUNT (unique,0x2560,4)
    (unique,0x2580,1) = INT_AND (unique,0x2570,1), 0x1
    PF = INT_EQUAL (unique,0x2580,1), 0x0
```

Indentify type of instruction

FXTi commented 21 days ago — Author — Member

P-code list for your reference: https://ghidra.re/courses/languages/html/pcodedescription.html

R_ANAL_OP_TYPE_MOV = 9, /* register move */

Pattern:

- Has `COPY`
- Input and output come from registers appear in assembly

Example:

```
0x00001191: MOV EBP,ESP
    EBP = COPY ESP
0x00001194: MOV dword ptr [EBP + -0x4],0x0
    (unique,0x3a0,4) = INT_ADD EBP, 0xfffffffc
    (unique,0xf40,4) = COPY 0x0
    (unique,0xf40,4) = STORE [(unique,0x3a0,4)]
0x0000119b: MOV dword ptr [EAX + -0x64],EAX
    (unique,0x3a0,4) = INT_ADD EAX, 0xffffff9c
    (unique,0xf30,4) = COPY EAX
    (unique,0xf30,4) = STORE [(unique,0x3a0,4)]
```

Indentify type of instruction

```
if(anal_type_XCHG(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_SINGLE(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_XSWI(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_XCMP(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_NOR(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_XPUSH(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_POP(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_STORE(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_LOAD(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_INT_XXX(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_NOT(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
if(anal_type_MOV(anal, anal_op, pcode_slg.pcodes, arg_set))
    return;
```

Indentify type of instruction

PUSH qword ptr [RCX + -0x4]

(unique,0x620,8) = INT_ADD RCX, 0xfffffffffffffffc

(unique,0x17b0,8) = LOAD ram[(unique,0x620,8)]

(unique,0x1cd0,8) = COPY (unique,0x17b0,8)

RSP = INT_SUB RSP, 0x8

(unique,0x1cd0,8) = STORE ram[RSP]

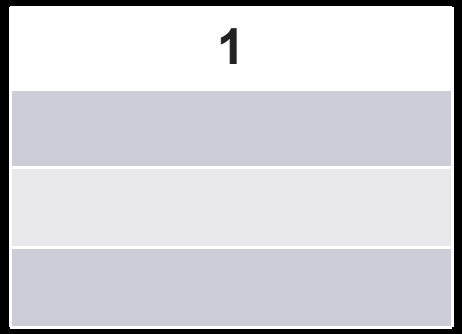
Indentify type of instruction

```
0x08048543: R_ANAL_OP_TYPE_POP dst: eax
0x08048552: R_ANAL_OP_TYPE_ADD dst: esi in0: esi in1: 0x1
0x08048552: R_ANAL_OP_TYPE_ADD dst: esi in0: esi in1: 0x1
 in0: esi in1: edi
 in0: esi in1: edi
            ;-- __libc_csu_init:
            0x08048500      55              PUSH ebp
            0x08048501      57              PUSH edi
            0x08048502      56              PUSH esi
            0x08048503      53              PUSH ebx
            0x08048504      e869000000      CALL sym.__i686.get_pc_thunk.bx
            0x08048509      81c3eb1a0000    ADD ebx,0x1aeb
            0x0804850f      83ec1c          SUB esp,0x1c
            0x08048512      8b6c2430        MOV ebp,dword ptr [esp + 0x30]
            0x08048516      8dbb20ffffff    LEA edi,[ebx + 0xffffff20]
            0x0804851c      e817fefffe      CALL sym._init
            0x08048521      8d8320ffffff    LEA eax,[ebx + 0xffffff20]
            0x08048527      29c7            SUB edi,eax
            0x08048529      c1ff02          SAR edi,0x2
            0x0804852c      85ff            TEST edi,edi
        ,=< 0x0804852e      7429            JZ 0x8048559
        |   0x08048530      31f6            XOR esi,esi
```

Translate P-code into ESIL

(unique,0x17d0,1) = COPY 0x1

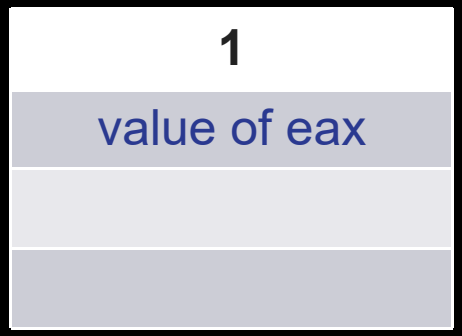
1,=



Translate P-code into ESIL

(unique,0xa400,4) = INT_AND EAX,1

1,eax,GET,&



Translate P-code into ESIL

- When register is one char, like “a”, it will be treated as a value when “,==”
- Float register needs a uniform interface to fetch value onto stack
- If I choose to leave middle variables on stack, how to fetch them?

Translate P-code into ESIL

- Add “GET”, to push register onto stack
- All elements on ESIL stack involved in calculation will be prefetch, thus only pure value will be on stack waiting for operation
- Add “PICK”, to fetch element on stack in random access

Translate P-code into ESIL

- Override “=”, to store value from stack back to register
- Override “[4]” and “[8]” to read float number from memory
- Override “=[4]” and “=[8]” to store float number to memory
- Add serials of float operation.

Translate P-code into ESIL

```
[0x00000000]> pdgsd 1
0x00000000: SAR EAX,1
    (unique,0x17d0,1) = COPY 0x1
    (unique,0xa400,4) = INT_AND EAX, 0x1
    CF = INT_NOTEQUAL (unique,0xa400,4), 0x0
    OF = COPY 0x0
    EAX = INT_SRIGHT EAX, 0x1
    RAX = INT_ZEXT EAX
    SF = INT_SLESS EAX, 0x0
    ZF = INT_EQUAL EAX, 0x0
    (unique,0x2560,4) = INT_AND EAX, 0xff
    (unique,0x2570,1) = POPCOUNT (unique,0x2560,4)
    (unique,0x2580,1) = INT_AND (unique,0x2570,1), 0x1
    PF = INT_EQUAL (unique,0x2580,1), 0x0
[0x00000000]> pdgsd 1; ao 1 | grep esil
esil: 1,1,eax,GET,&,32,1,<<,1,SWAP,-,&,0,2,PICK,==,!,cf,=,0,of,=,1,eax
,GET,32,SWAP,SIGN,>>,32,1,<<,1,SWAP,-,&,eax,=,eax,GET,rax,=,0,eax,GET,
32,SWAP,SIGN,SWAP,32,SWAP,SIGN,SWAP,<,sf,=,0,eax,GET,==,zf,=,255,eax,G
ET,&,32,1,<<,1,SWAP,-,&,1,PICK,POPCOUNT,1,2,PICK,&,8,1,<<,1,SWAP,-,&,0
,2,PICK,==,pf,=,CLEAR
```

DEMO

Thanks

- Florian Märkl (@thestr4ng3r)
- Giovanni (@wargio)
- Anton (@XVilka)