C/C++ 오픈소스를 Tracing 하면 얻을 수 있는 것들

Tracing the C/C++ open-sources for profit and fun

HANBUM PARK

SOSCON 2018
SAMSUNG OPEN SOURCE CONFERENCE 2018

HANBUM PARK

KOSSLAB OPEN-FRONTIER

Uftrace Contributor
Reverse-Engineer

CONTENTS

The stories I want to share.

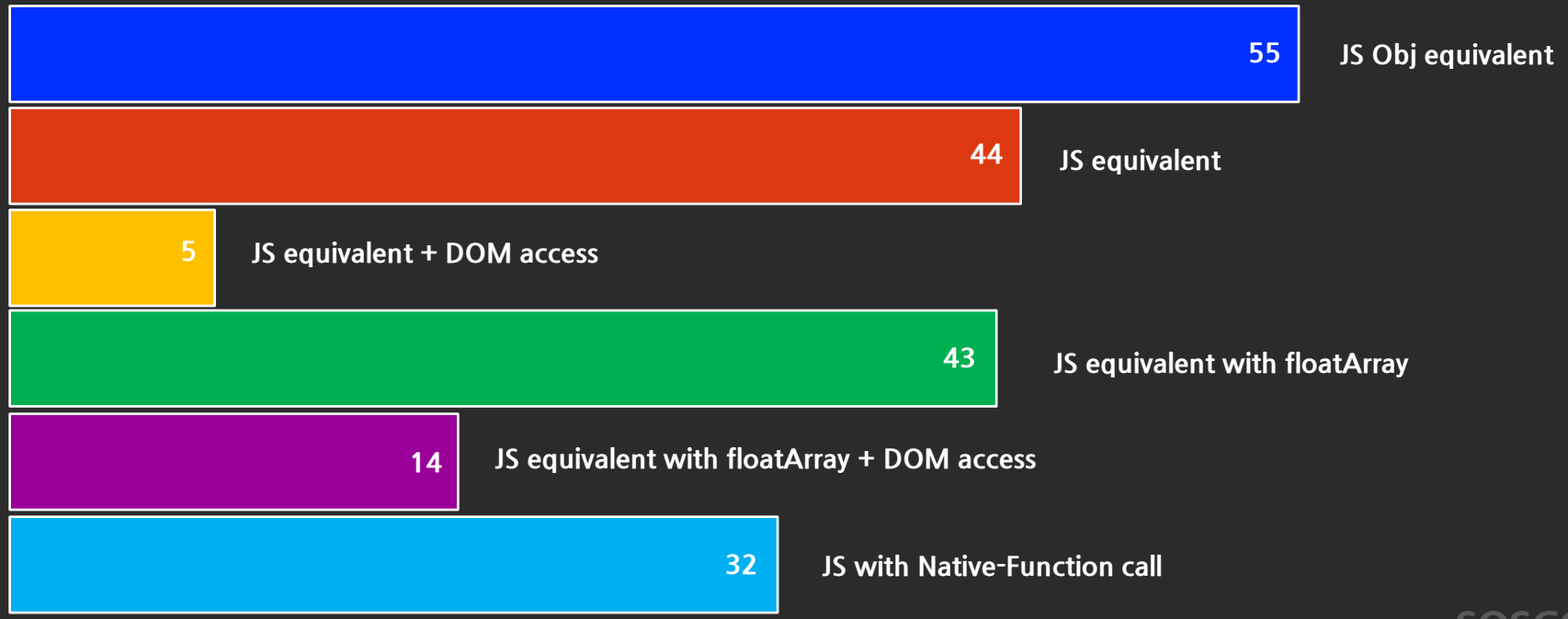
- Introduction
- Motivation
- Uftrace
- Tracing for profit
  - Android VM
- Tracing for fun
  - Gcc
- Retrospect
- Conclusion

Introduce

Tracing 사례 살펴보기

Tracing case study - Node.js

Native 함수 호출을 통한 연산이 순수 Javascript보다 느린 경우



Operation per second (higher is better)

DOMMatrix perf (https://jsperf.com/dommatrix-perf/9)

Tracing case study - Node.js

Uftrace로 Tracing 한 후 결과 로그를 통해 이유를 확인할 수 있습니다

Tracing case study - Node.js

Sum이란 함수로 배열 안의 값들의 합을 구한다고 가정합시다

```
let s = sum([1, 2, 3, 4, 5, 6, 7, 8, 9]);
```

Tracing case study - Node.js

배열 안의 값들에 합을 구하는 Sum 함수 - Javascript

```
function sum(elements) {
    let s = 0;
    elements.forEach(element => { s += element; });
    return s;
}
```

Tracing case study - Node.js

배열 안의 값들에 합을 구하는 Sum 함수 - Node.js N-API

```
napi_value Sum(napi_env, napi_callback_info info) {

        for (int i = 0; i < length; i++) {
            napi_value element;
            napi_get_element(env, i, &element);
            napi_valuetype valuetype;
            napi_typeof(env, element, &valuetype);
            double value;
            napi_get_value_double(env, element, &value);
            sum += value;
        }

        napi_value js_sum;
        napi_create_double(env, sum, &js_sum);
        return js_sum;

}
```

Tracing case study - Node.js

배열 안의 값들에 합을 구하는 Sum 함수는 두 부분으로 이뤄집니다

```
napi_value Sum(napi_env, napi_callback_info info) {
  for (int i = 0; i < length; i++) {
    napi_value element;
    napi_get_element(env, i, &element);
    napi_valuetype valuetype;
    napi_typeof(env, element, &valuetype);
    double value;
    napi_get_value_double(env, element, &value);
    sum += value;
  }
  napi_value js_sum;
  napi_create_double(env, sum, &js_sum);
  return js_sum;
}
```

연산

반환

Tracing case study - Node.js

Javascript 단에 저장된 값을 Native에서 사용하기 위해 읽어와야 합니다

```
napi_value Sum(napi_env, napi_callback_info info) {

    for (int i = 0; i < length; i++) {
      napi_value element;
      napi_get_element(env, i, &
      napi_valuetype valuetype;
      napi_typeof(env, element,
      double value;
      napi_get_value_double(env,
      sum += value;
    }

    napi_value js_sum;
    napi_create_double(env, sum, &js_sum);
    return js_sum;

}
```

Get Javascript value

```
napi_value element;
napi_get_element(env, i, &element);
```

Tracing case study - Node.js

이 과정에서 Javascript의 값에 대한 Type을 체크해야 합니다

```
napi_value Sum(napi_env, napi_callback_info info) {

    for (int i = 0; i < length; i++) {
      napi_value element;
      napi_get_element(env, i, &element);
      napi_valuetype valuetype;
      napi_typeof(env, element, &
      double value;
      napi_get_value_double(env
      sum += value;
    }

    napi_value js_sum;
    napi_create_double(env, sum, &js_sum);
    return js_sum;

}
```

Type checking

```
napi_valuetype valuetype;
napi_typeof(env, element, &valuetype);
```

Tracing case study - Node.js

Native에서 연산이 끝났으면, 결과 값을 Javascript에서 사용가능한 타입으로 바꿔야 합니다

```
napi_value Sum(napi_env, napi_callback_info info) {

    for (int i = 0; i < length; i++) {
        napi_value element;
        napi_get_element(env, i, &element);
        napi_valuetype valuetype;
        napi_typeof(env, element, &valuetype);
        double value;
        napi_get_value_double(env, element, &value);
        sum += value;
    }

    napi_value js_sum;
    napi_create_double(env, sum
    return js_sum;

}
```

Type converting

```
napi_valuetype valuetype;
napi_typeof(env, element, &valuetype);
```

Tracing case study - Node.js

Resolve : Chromium WebIDL to Node.js - automatic binding and checking

```
// N-API
napi_value Sum(napi_env, napi_callback_info info) {
    for (int i = 0; i < length; i++) {
        napi_value element;
        napi_get_element(env, i, &element);
        napi_valuetype valuetype;
        napi_typeof(env, element, &valuetype);
        double value;
        napi_get_value_double(env, element, &value);
        sum += value;
    }
    napi_value js_sum;
    napi_create_double(env, sum, &js_sum);
    return js_sum;
}
```

```
// WebIDL
[Constructor]
interface Calculator {
        double sum(sequence<long> elements);
};

// Native Code
class Calculator {
public:
    double Sum(std::vector<int> elements) {
        int sum = 0;
        for (int i = 0; I < elements.size(); i++)
            sum += elements[i];
        return sum;
    }
};
```

Tracing case study - Node.js

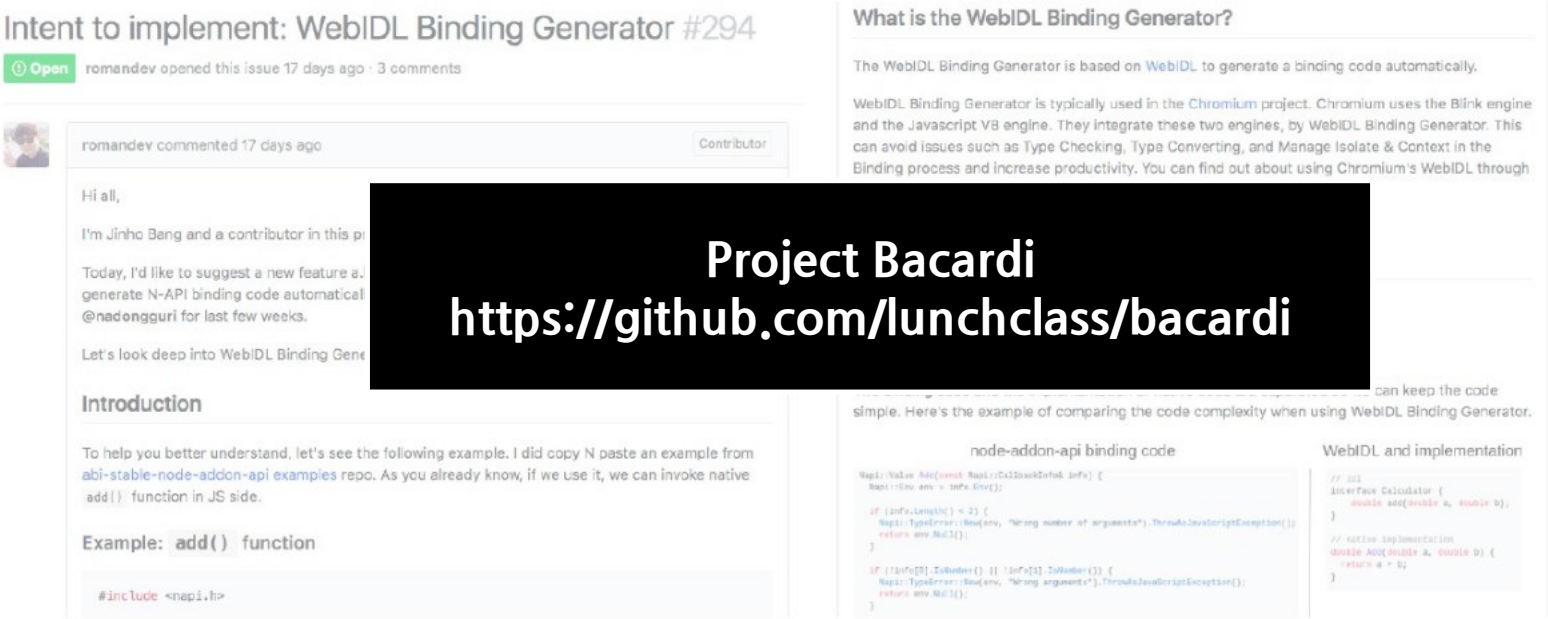
Resolve : Chromium WebIDL to Node.js - ignore type checking

```
// N-API
napi_value Sum(napi_env, napi_callback_info info) {
    for (int i = 0; i < length; i++) {
        napi_value element;
        napi_get_element(env, i, &element);
        napi_valuetype valuetype;
        napi_typeof(env, element, &valuetype);
        double value;
        napi_get_value_double(env, element, &value);
        sum += value;
    }
    napi_value js_sum;
    napi_create_double(env, sum, &js_sum);
    return js_sum;
}
```

```
// WebIDL
[Constructor]
interface Calculator {
        [NoTypeChecking] double sum(sequence<long> elements);
};

// Native Code
class Calculator {
public:
    double Sum(std::vector<int> elements) {
        int sum = 0;
        for (int i = 0; I < elements.size(); i++)
            sum += elements[i];
        return sum;
    }
};
```

Tracing case study - Node.js

방진호님은 WebIDL을 Node.js에 제의하였으며 개발…



Project Bacardi
https://github.com/lunchclass/bacardi

Tracing case study - Node.js

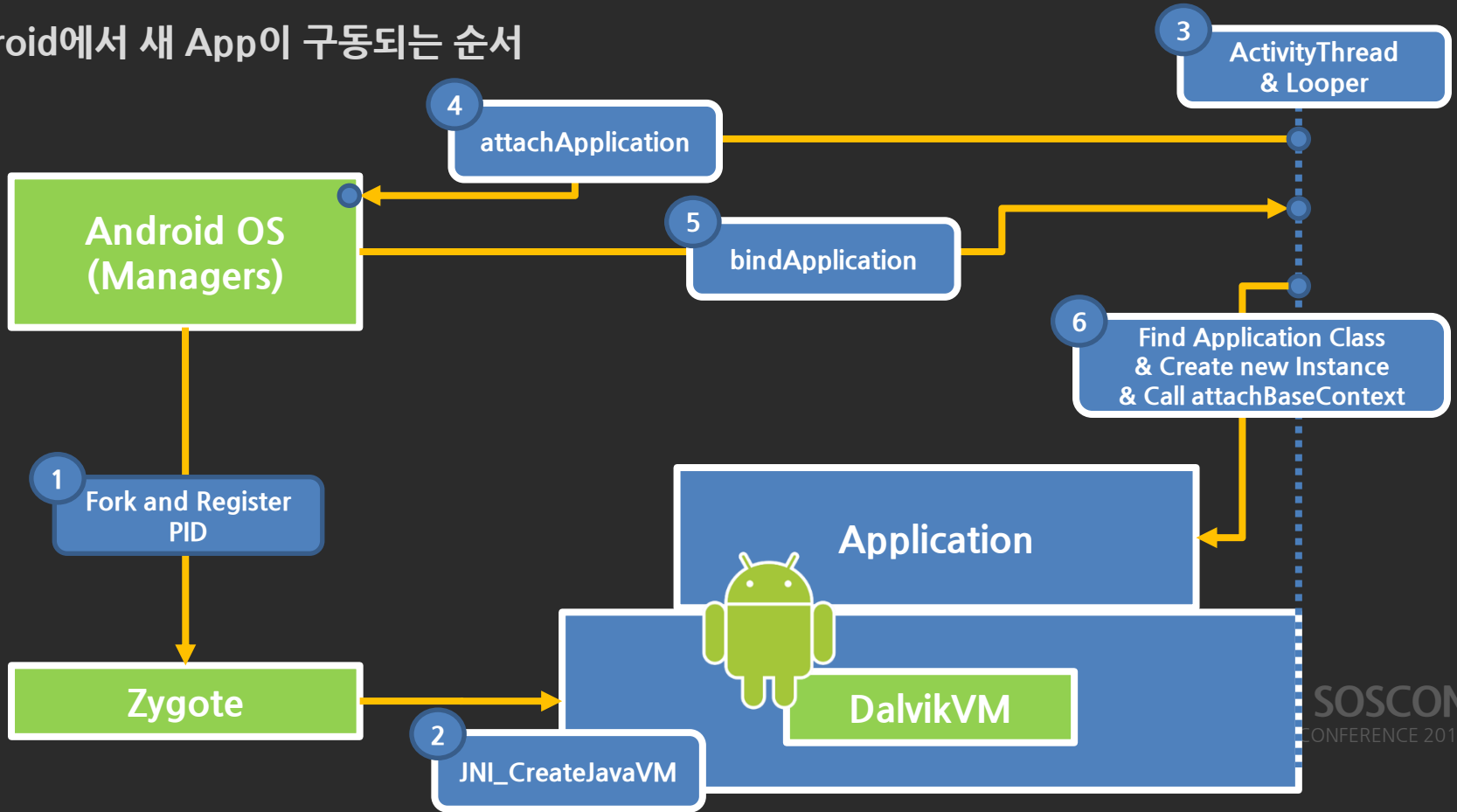
Tracing, “프로그램의 실행 동안의 변화를 다각도로 기록하고 이를 검토하는 일”

Motivation

시간은 금이라고 친구

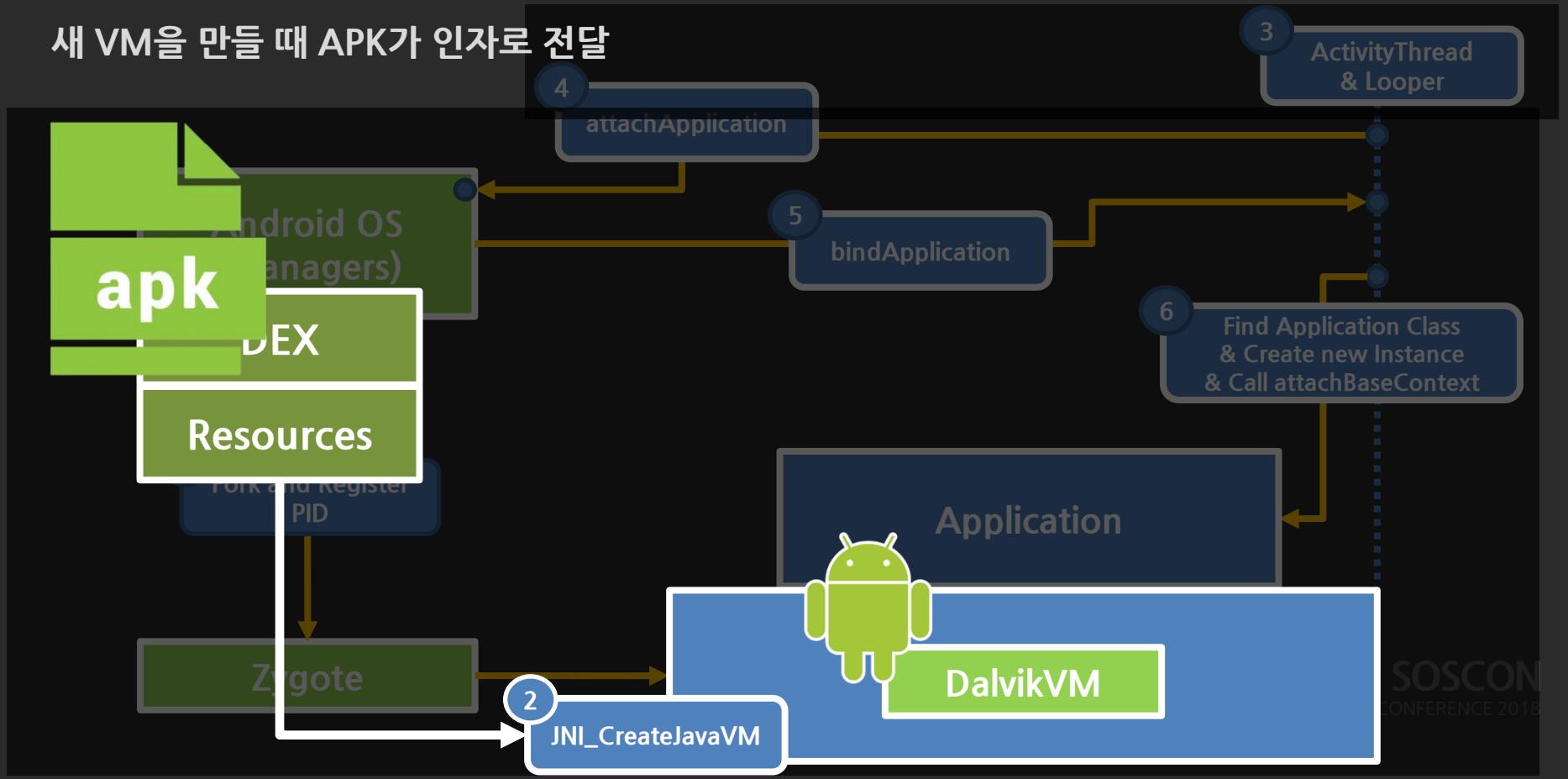
Motivation

Android에서 새 App이 구동되는 순서



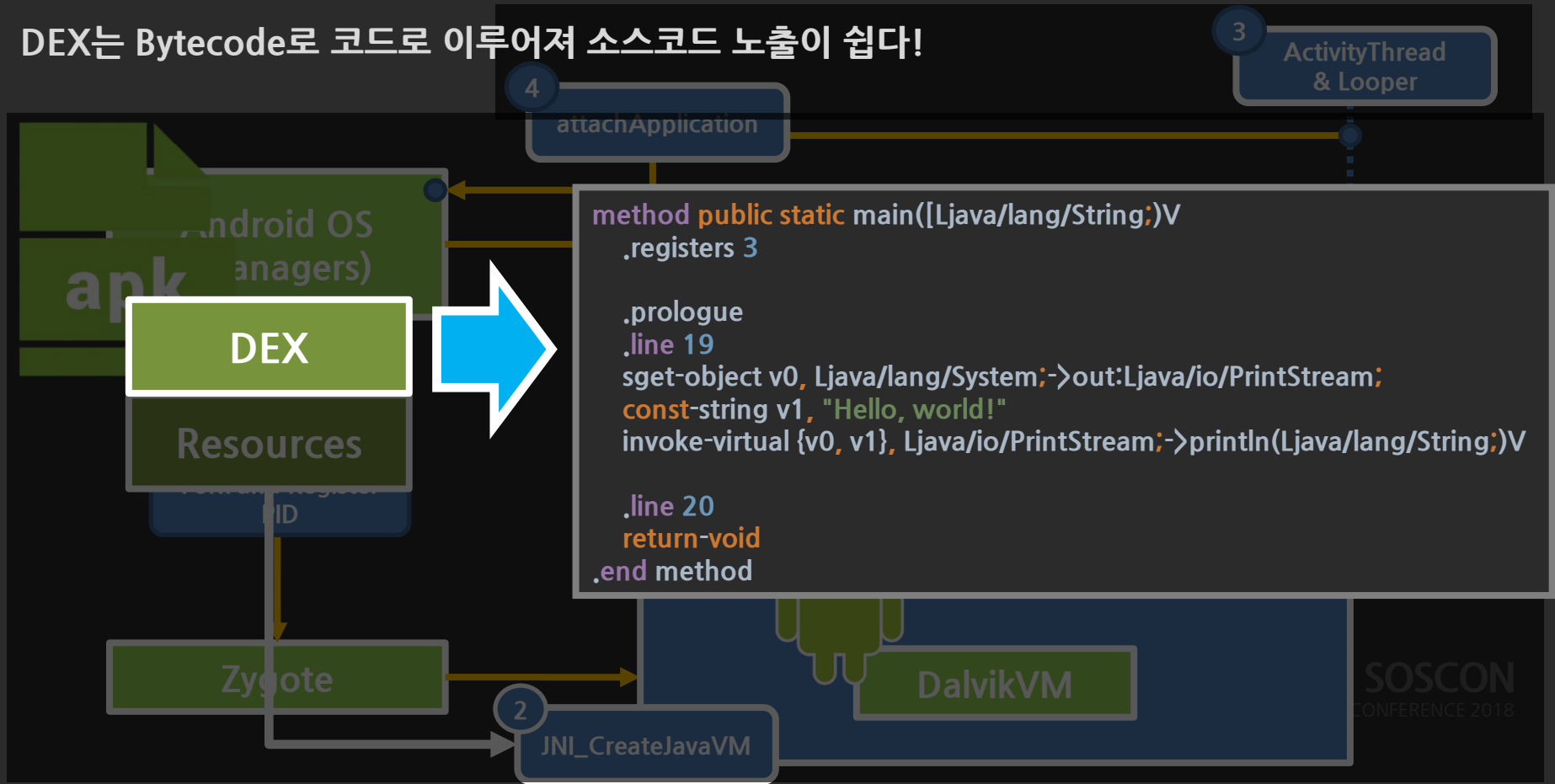
Motivation

새 VM을 만들 때 APK가 인자로 전달



Motivation

DEX는 Bytecode로 코드로 이루어져 소스코드 노출이 쉽다!



Motivation

DEX의 보호 방안 찾기 대작전!

원본 DEX를 보호하는 메커니즘의 필요

Motivation

You are right always

EntryPoint

Android
Open
Source
Project

Motivation

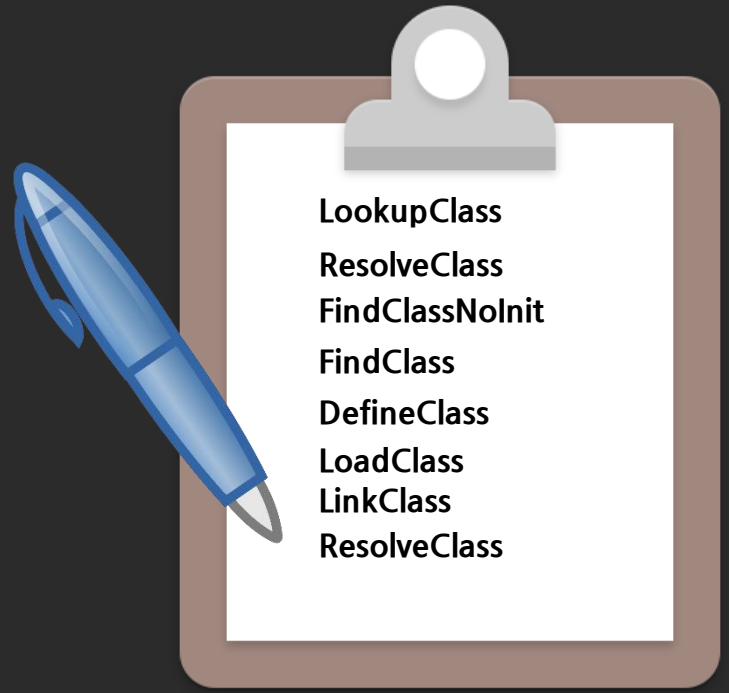
Everything grow up, except my salary

AOSP - art

Language	Files	Blank	Comment	Code Line
C	708	22952	28339	95886
C++	735	44751	49065	325759

Motivation

분석의 시작은 CC 형제, Ctags + Cscope와 함께 시작했습니다만



Motivation

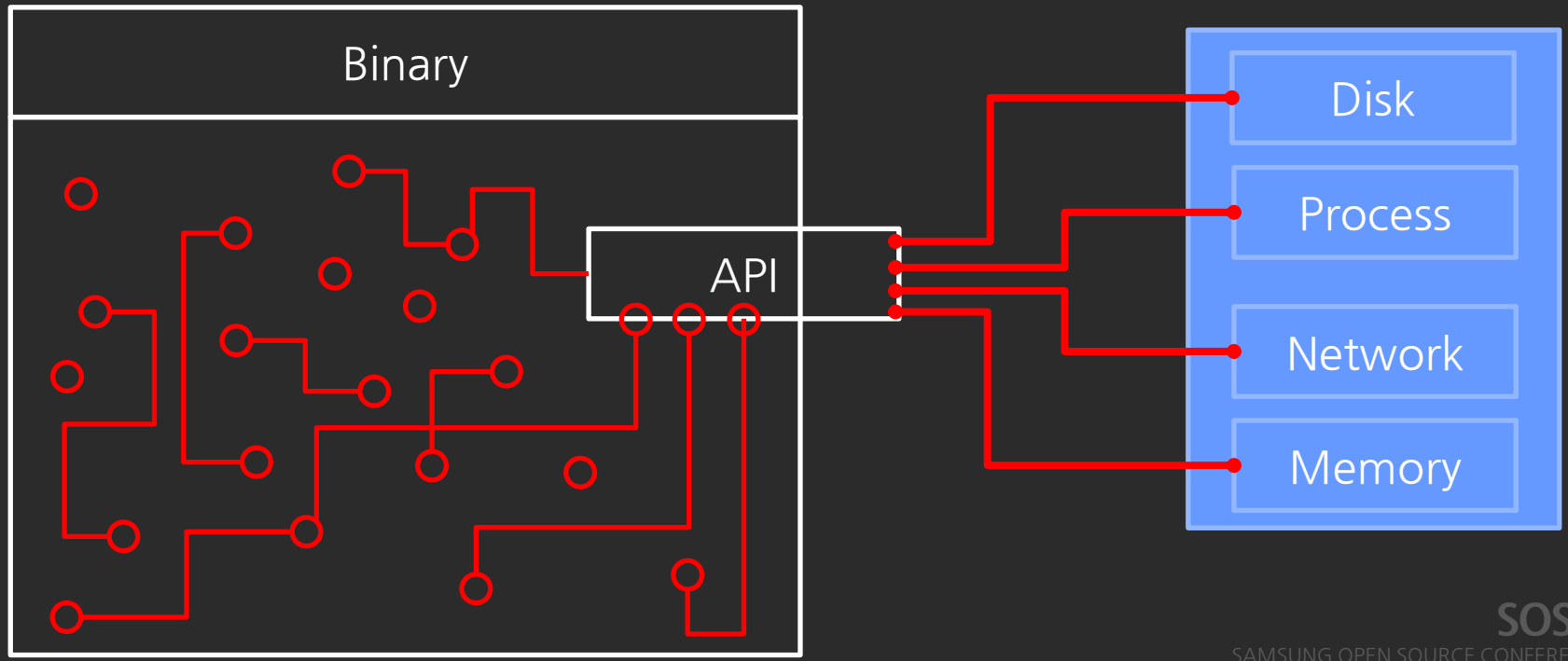
분석의 시작은 CC 형제, Ctags + Cscope와 함께 시작했습니다만

Motivation - time is gold, friends

You are right always, but need time

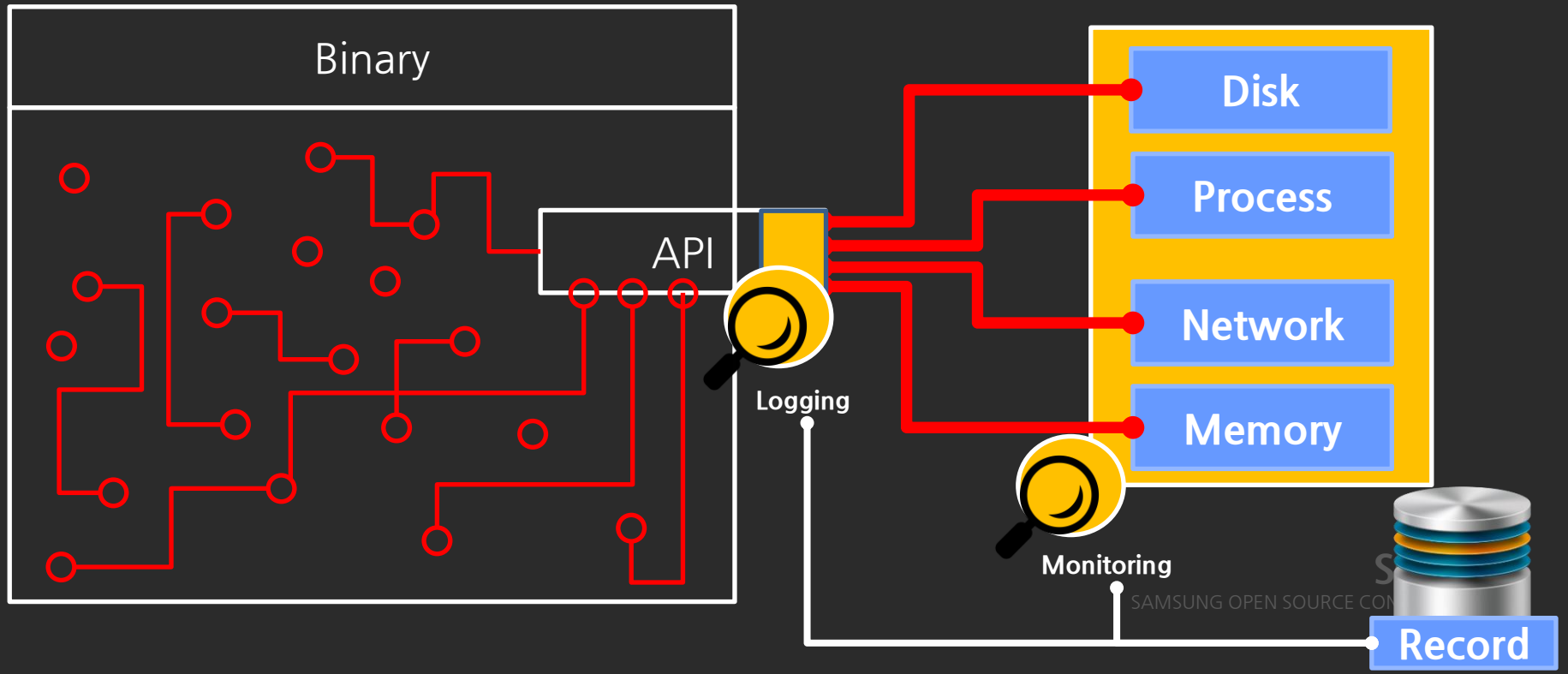
Motivation - time is gold, friends

Reverse Engineer을 하는 일반적인 방식 #1



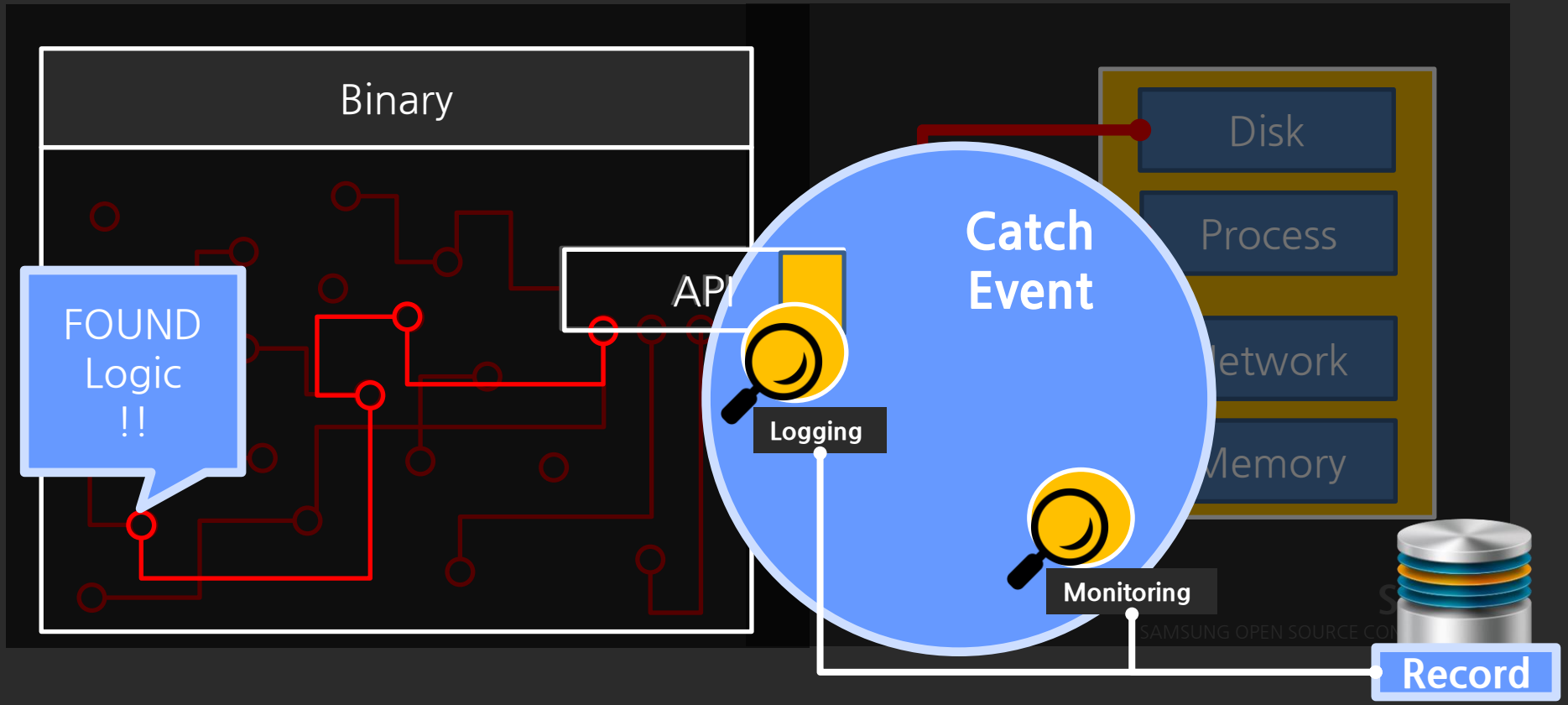
Motivation - time is gold, friends

Reverse Engineer을 하는 일반적인 방식 #2



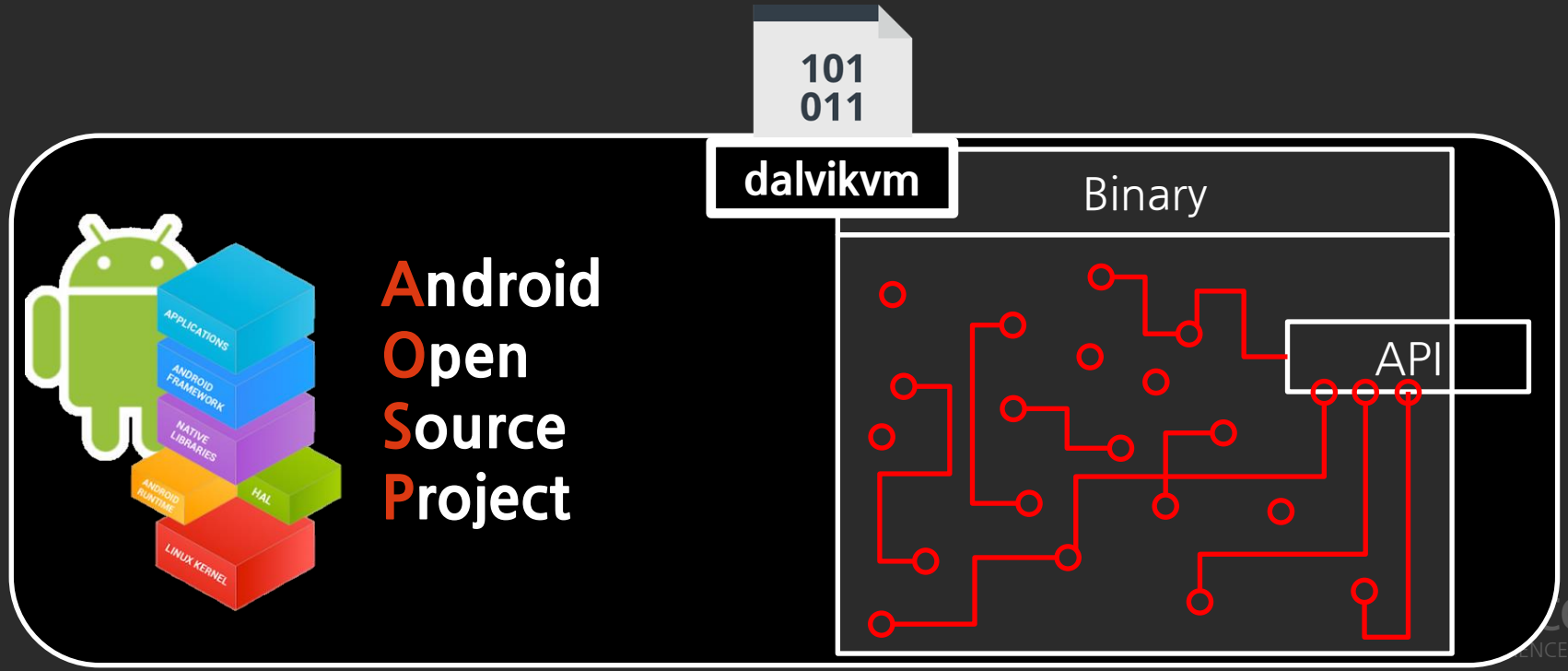
Motivation - time is gold, friends

Reverse Engineer을 하는 일반적인 방식 #3



Motivation - time is gold, friends

프로그램의 실행을 흐름을 로깅하면 분석에 필요한 최소 로직을 추려낼 수 있습니다



Uftrace

Function tracer for user-space

Introduce UFTRACE

일반적인 컴파일 결과

```
$ gcc foobar.c
```

```
void bar() {
}

void foo() {
  bar();
}

int main() {
    foo();
}
```

```
<bar>:
    ret

<foo>:
    call <bar>
    ret

<main>:
    call <foo>
    ret
```

Introduce UFTRACE

-pg 옵션으로 컴파일 했을 때의 결과

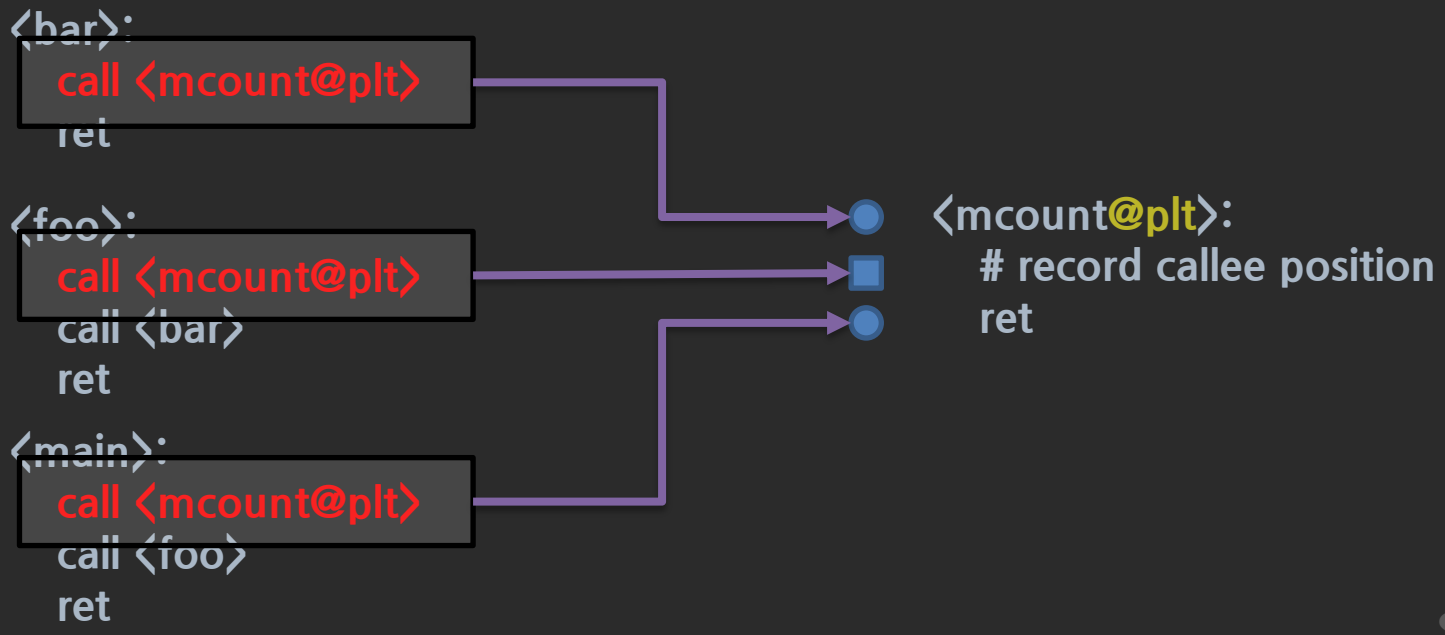
```
$ gcc -pg foobar.c
```

```
void bar() {
}

void foo() {
  bar();
}

int main() {
    foo();
}
```

```
<bar>:
    call <mcount@plt>
    ret

<foo>:
    call <mcount@plt>
    call <bar>
    ret

<main>:
    call <mcount@plt>
    call <foo>
    ret
```

Introduce UFTRACE

-pg로 컴파일 하면 컴파일러가 mcount 호출을 생성해 줍니다



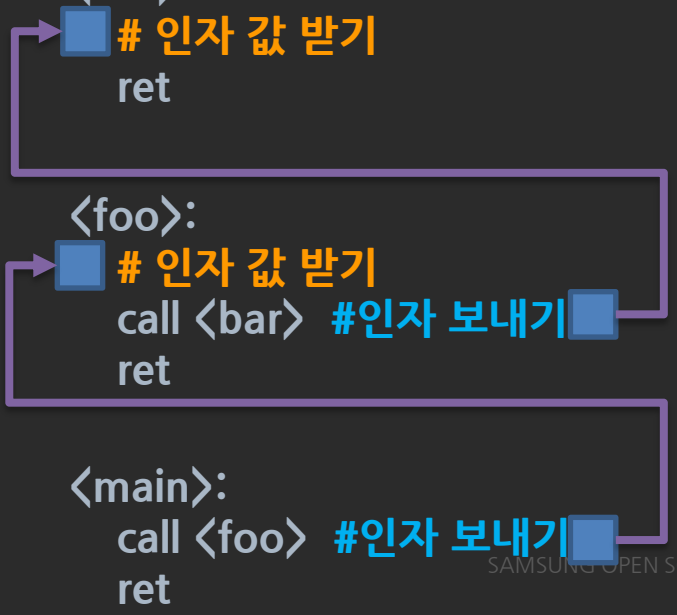
Introduce UFTRACE

함수 간에는 인자 값을 전달하는 방식은 약속되어 있습니다

```
$ gcc foobar.c  -o  output
```

```
void bar(int arg) {
}

void foo(int arg) {
  bar(arg);
}

int main() {
    foo(0x10);
}
```

```
<bar>:
    # 인자 값 받기
    ret

<foo>:
    # 인자 값 받기
    call <bar>  #인자 보내기
    ret

<main>:
    call <foo>  #인자 보내기
    ret
```

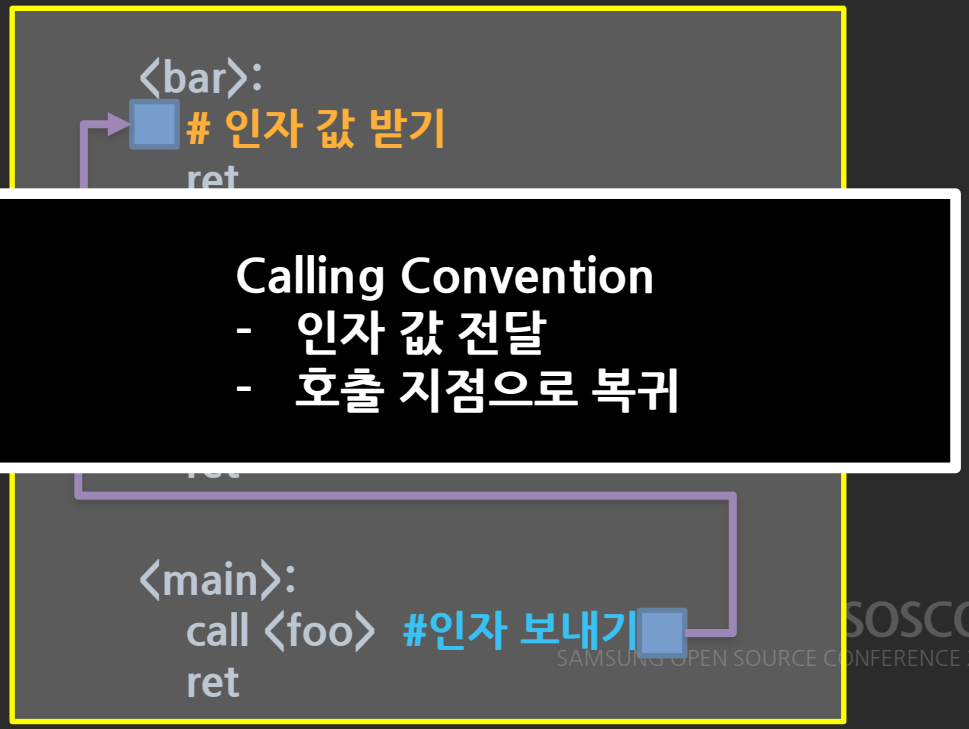


Introduce UFTRACE

함수 간에는 인자 값을 전달하는 방식은 약속되어 있습니다

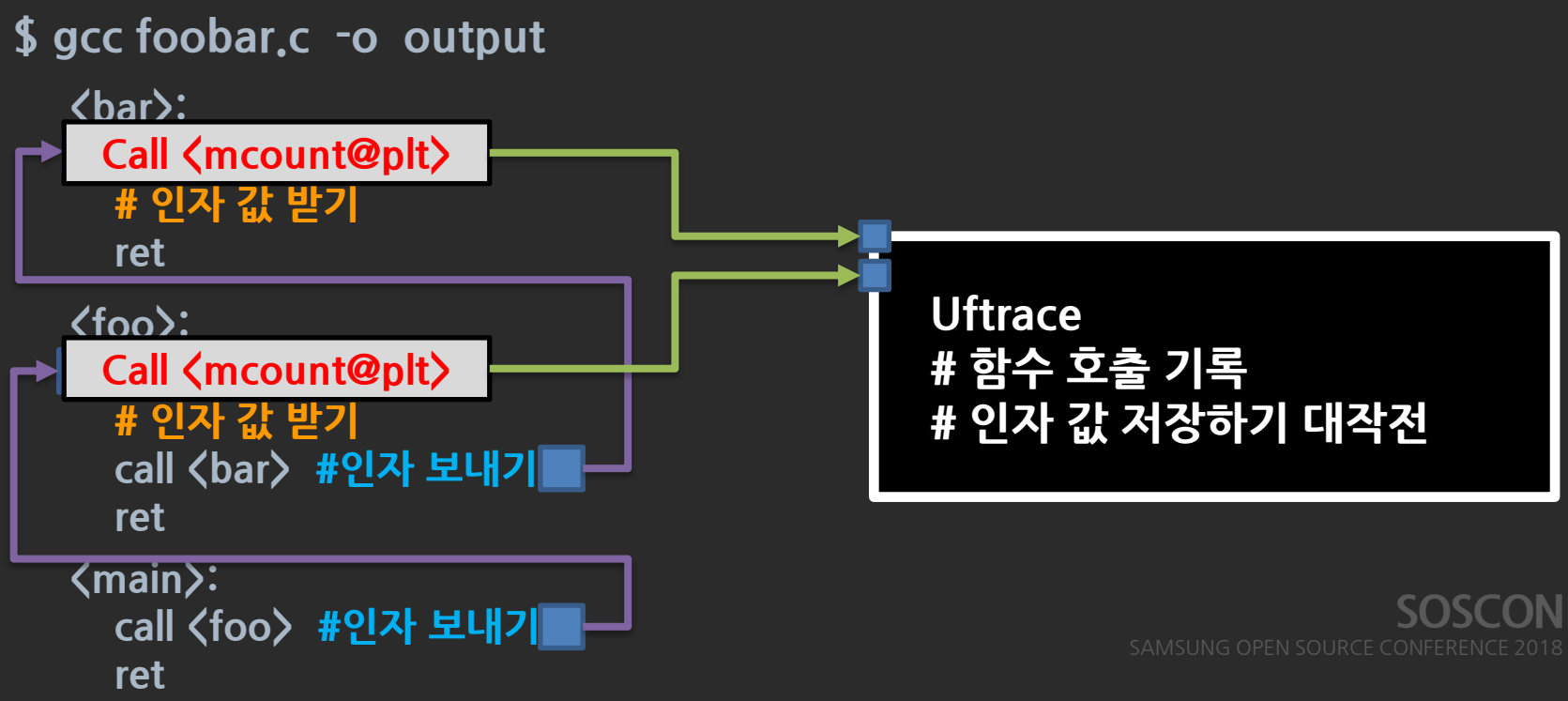
```
$ gcc foobar.c  -o  output
```

```
void bar(int arg) {
}

void foo(int arg) {
  bar(arg);
}

int main() {
    foo(0x10);
}
```



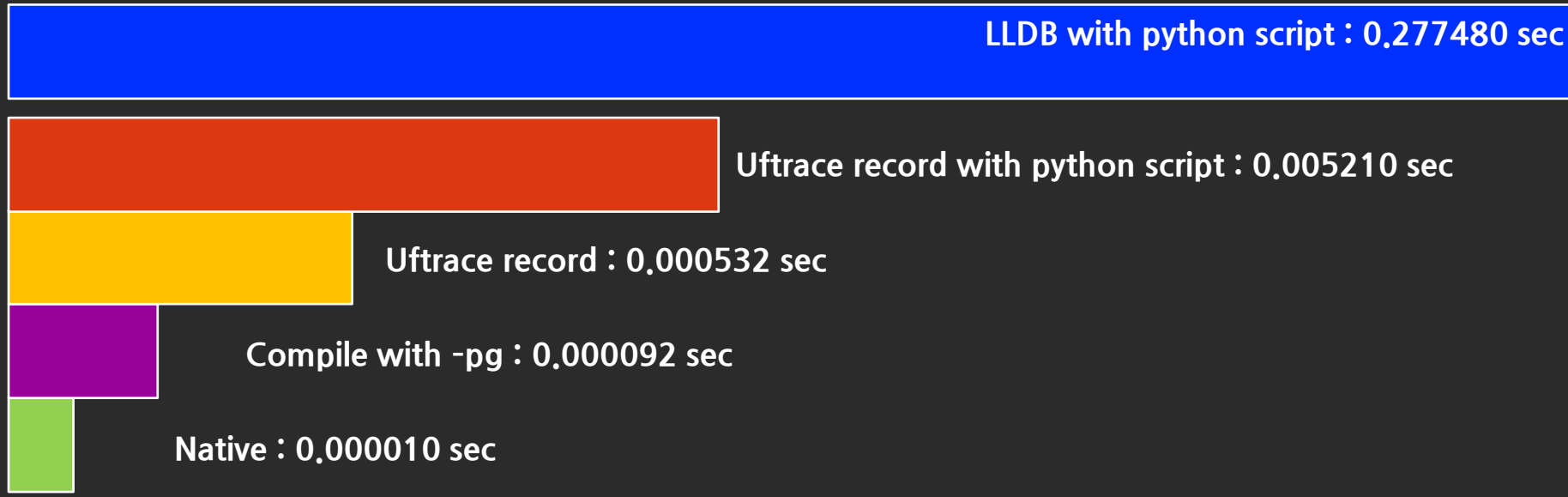
Introduce UFTRACE

호출 규약에 지정된 레지스터, 메모리를 읽어 인자 값을 알아낼 수 있습니다



Uftrace performance

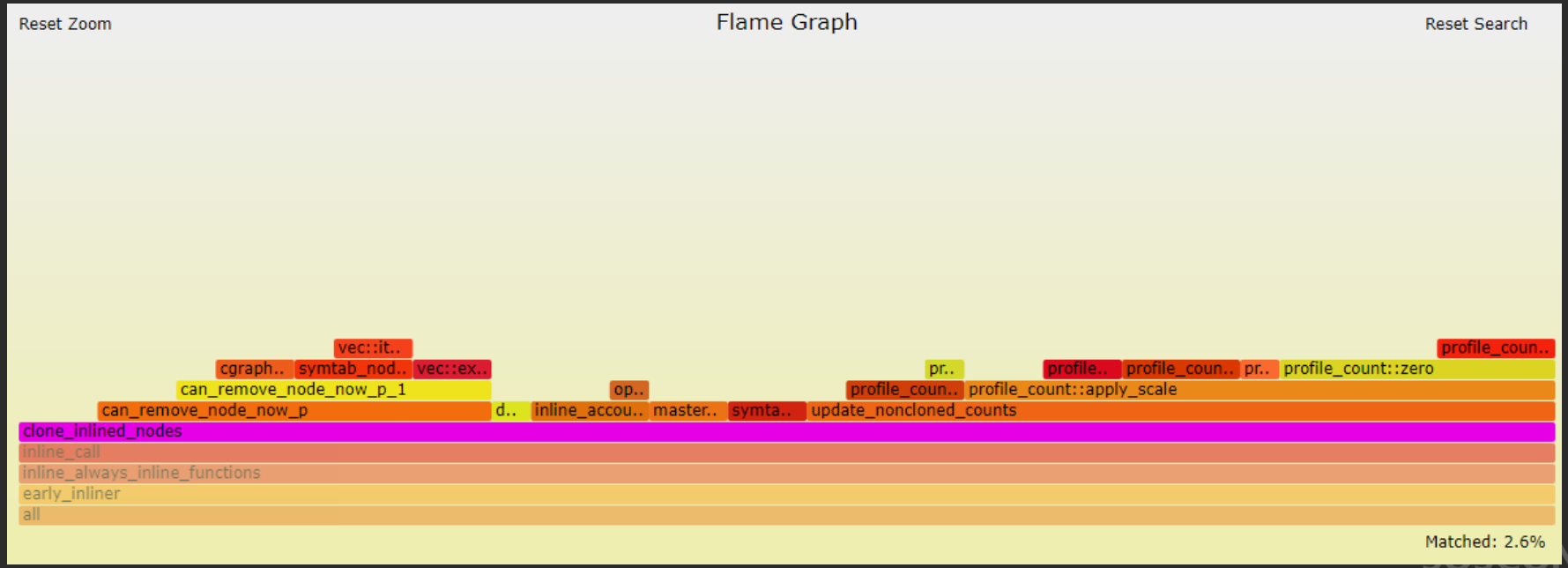
동일 작업을 한다고 가정하면, Debugger보다 부하가 월등히 적습니다



Appendix A. benchmark

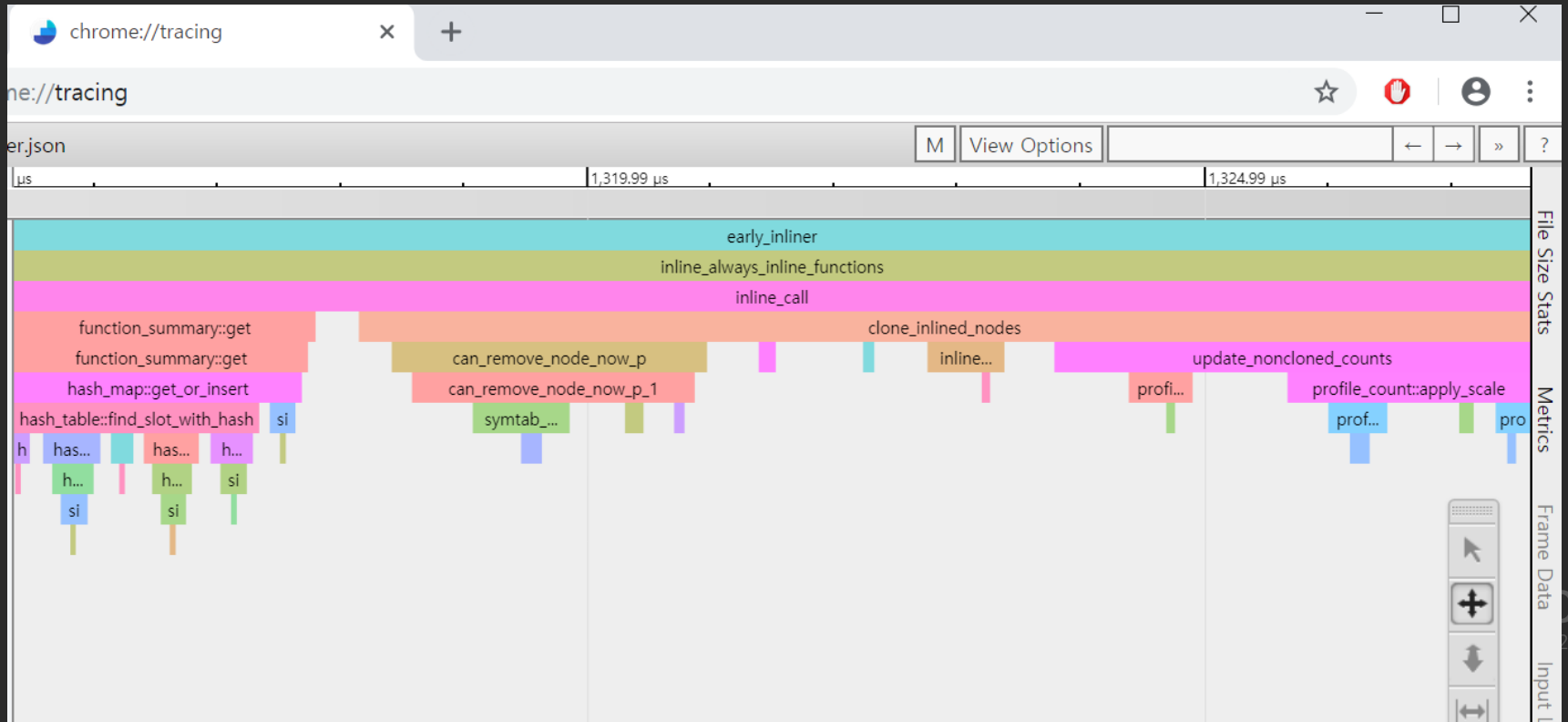
Uftrace visualization

$ uftrace dump --flame-graph -F clone_inlined_nodes > clone_inlined_nodes.flame
$ perl flamegraph.pl clone_inlined_nodes.flame > clone_inlined_nodes.svg



Uftrace visualization support

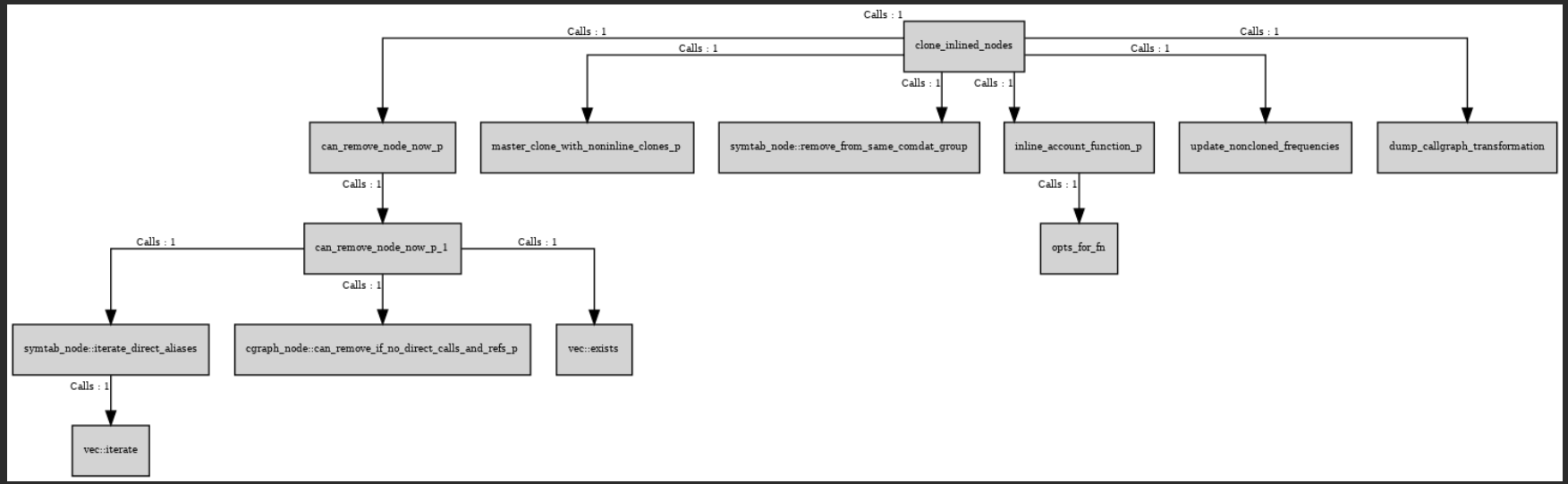
$ uftrace dump --chrome -F clone_inlined_nodes > clone_inlined_nodes.json



Uftrace visualization support

$ uftrace dump --graphviz -F clone_inlined_nodes > clone_inlined_nodes.dot

$ dot -Tpng clone_inlined_nodes.dot > clone_inlined_nodes.png



Tracing for profit

Android VM

Tracing for profit

Hello World를 출력하는 간단한 샘플 프로그램을 작성하여 Tracing!

```
public class Main {
    public static void main(String[] args) {
            System.out.println("Hello World!");
        }
}
```

```
$ javac Main.java
$ dx --dex --output=classes.dex Main.class
$ zip Main.apk classes.dex
$ uftrace record --auto-args `which dalvikvm64` -cp Main.apk Main
```

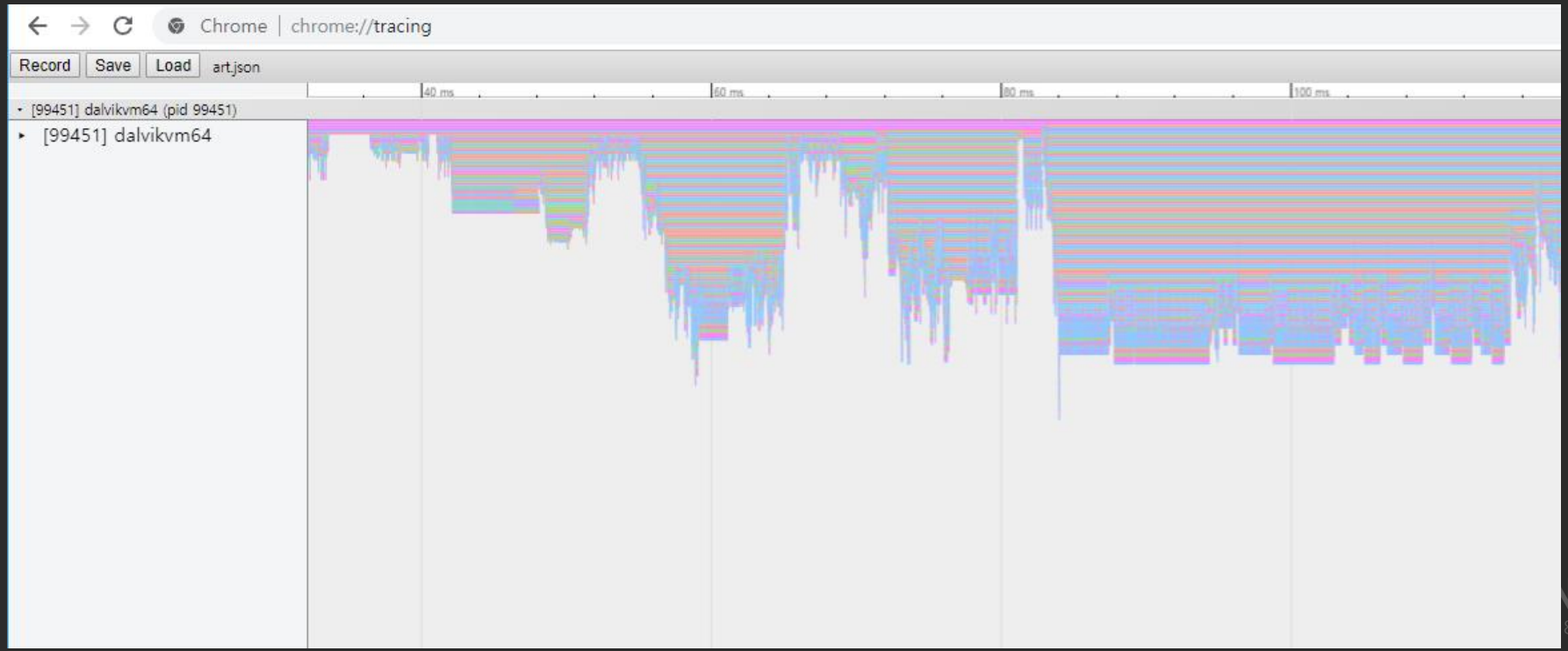
Tracing for profit

Results : 1,144,484 lines function call record

```
1 # DURATION     TID     FUNCTION
2               [ 99631] | main(4, 0x7ffc2b7a7bb8) {
3     0.595 us [ 99631] |   setvbuf();
4     0.922 us [ 99631] |   operator new[](48) = 0x55b99858f5b0;
5     0.415 us [ 99631] |   memset(0x55b99858f5b0, 0, 48) = 0x55b99858f5b0;
6   147.427 us [ 99631] |   strncmp("-cp", "-XXlib:", 7) = 11;
7     0.305 us [ 99631] |   strcmp("-cp", "-classpath") = 4;
8     0.166 us [ 99631] |   strcmp("-cp", "-cp") = 0;
9     0.198 us [ 99631] |   strncmp("Main.apk", "-XXlib:", 7) = 32;
10    0.146 us [ 99631] |   strcmp("Main.apk", "-classpath") = 32;
11    0.111 us [ 99631] |   strcmp("Main.apk", "-cp") = 32;


1144476    0.088 us [ 99631] | art::FaultManager::~FaultManager();
1144477    0.055 us [ 99631] | std::__1::__vector_base::~__vector_base();
1144478    0.060 us [ 99631] | std::__1::__vector_base::~__vector_base();
1144479    0.054 us [ 99631] | art::JDWP::JdwpOptions::~JdwpOptions();
1144480    0.052 us [ 99631] | std::__1::__vector_base::~__vector_base();
1144481    0.078 us [ 99631] | std::__1::__vector_base::~__vector_base();
1144482    0.053 us [ 99631] | std::__1::unique_ptr::~unique_ptr();
1144483    0.063 us [ 99631] | std::__1::unique_ptr::~unique_ptr();
1144484    0.150 us [ 99631] | std::__1::unique_ptr::~unique_ptr();
```

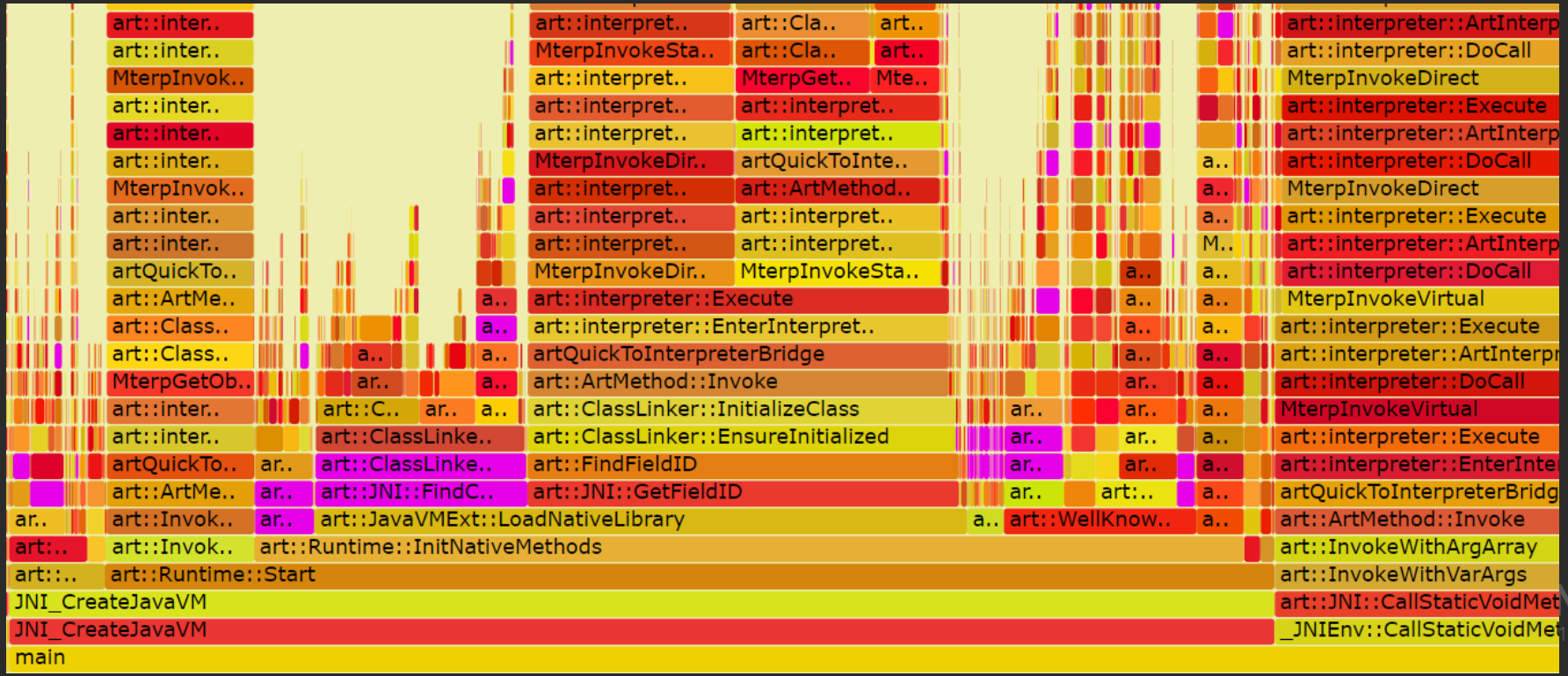
Tracing for profit

Results : Chrome Tracing Viewer



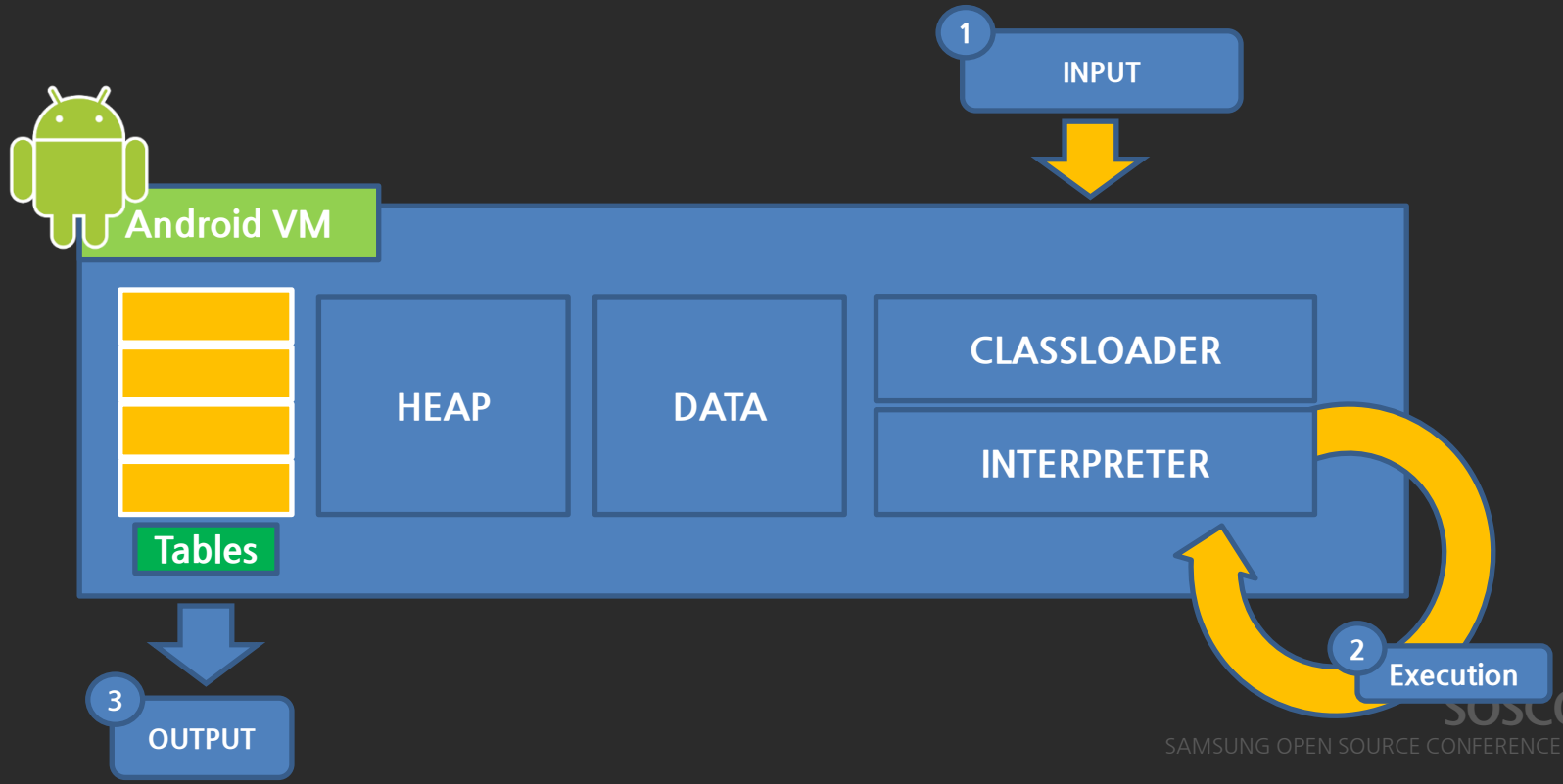
Tracing for profit

Results : Flamegraph



Tracing for profit

당황하지 마세요! 흔한 Interpreter 입니다.

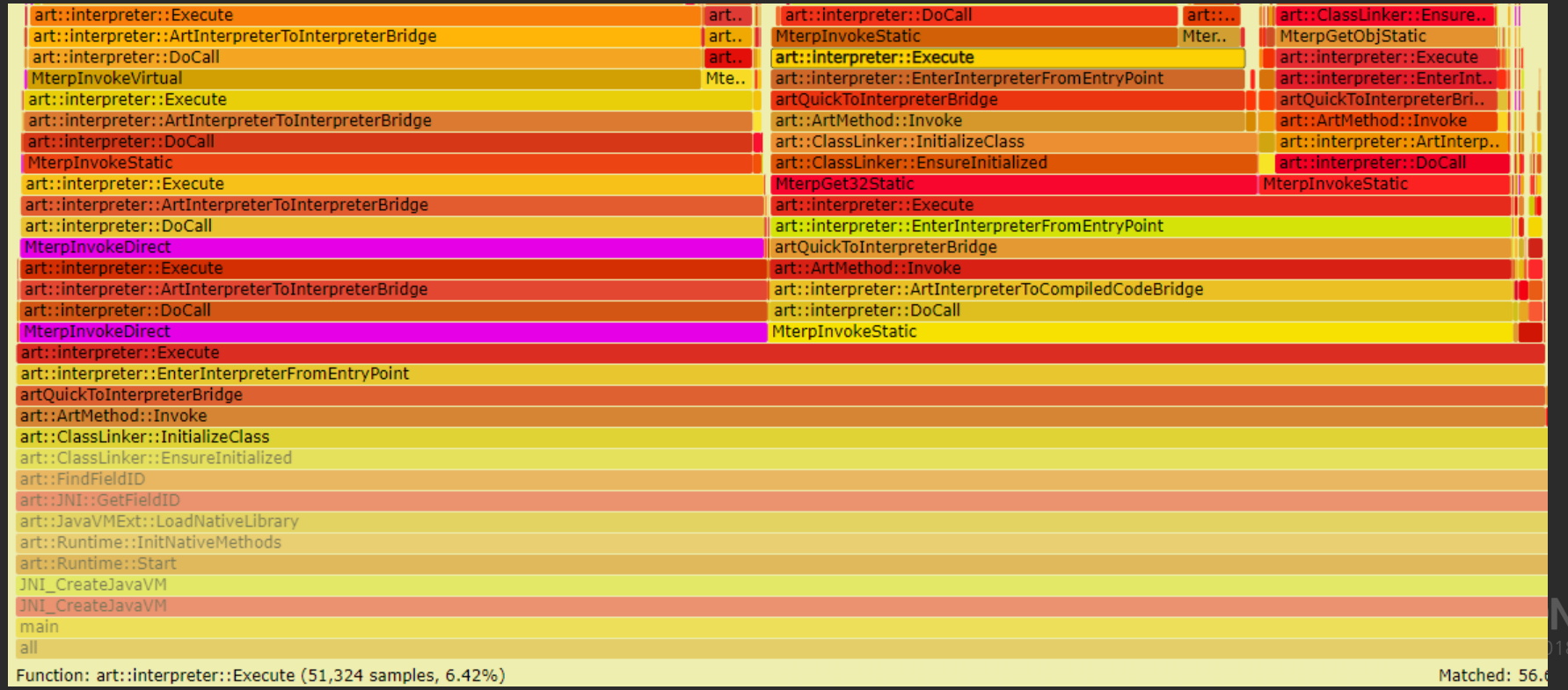


Tracing for profit

“Hello world”의 출력은 - 56.6%의 Interpreting과 36.2%의 Class Load으로 이뤄집니다

Tracing for profit

Flamegraph는 클릭된 지점에서의 호출 흐름을 보여줍니다

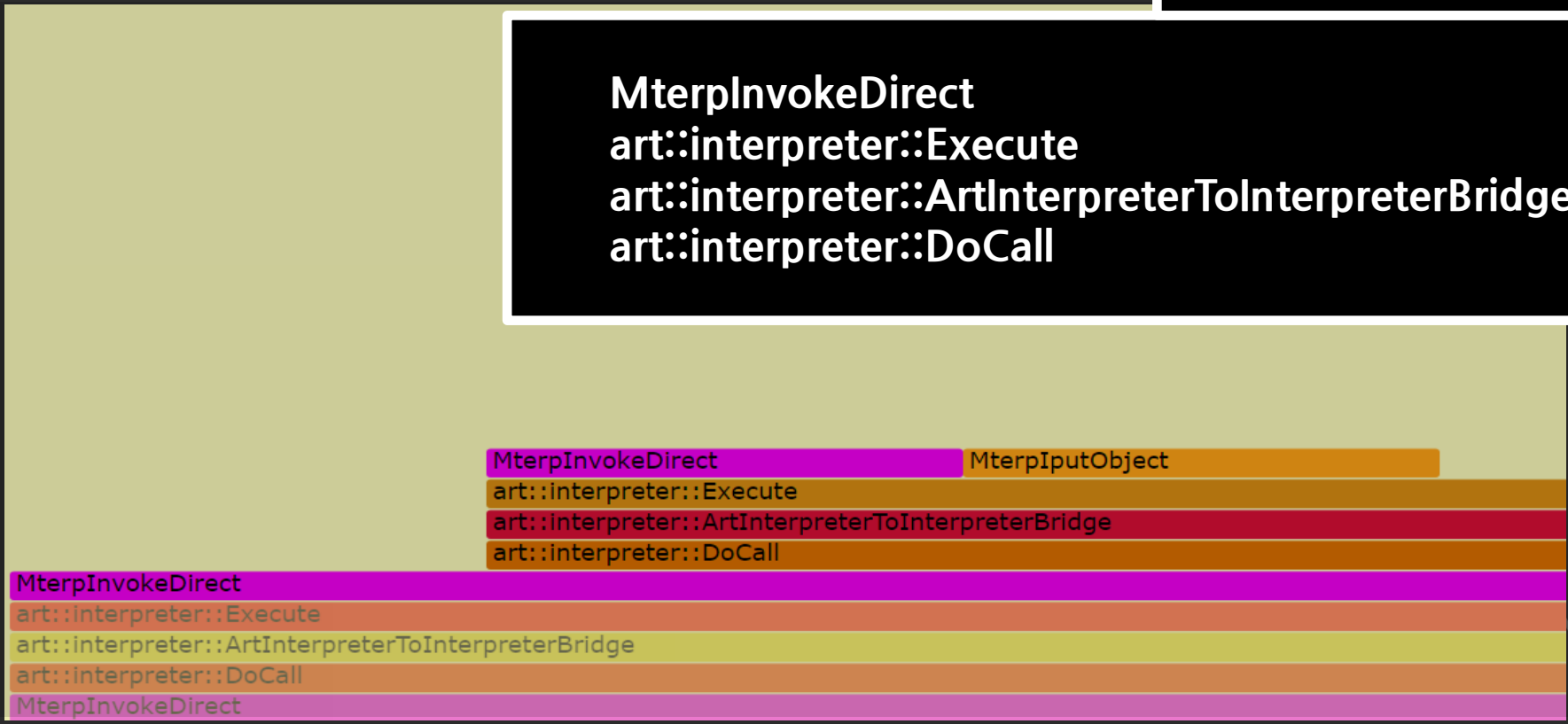


Tracing for profit

몇 번의 클릭으로 Interpreting이 이뤄지는 Path를 찾았습니다

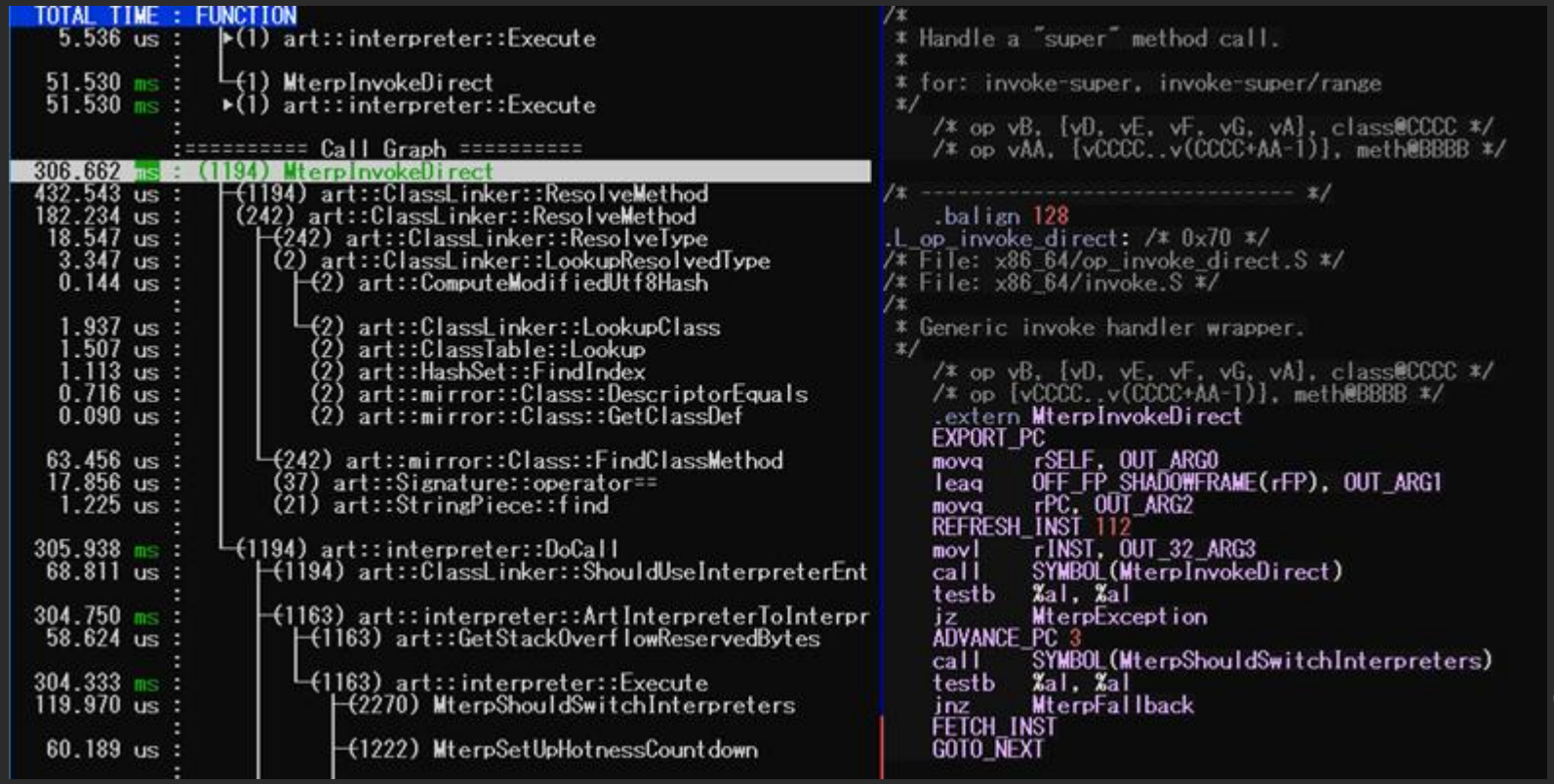
Interpreting Path

MterpInvokeDirect
art::interpreter::Execute
art::interpreter::ArtInterpreterToInterpreterBridge
art::interpreter::DoCall



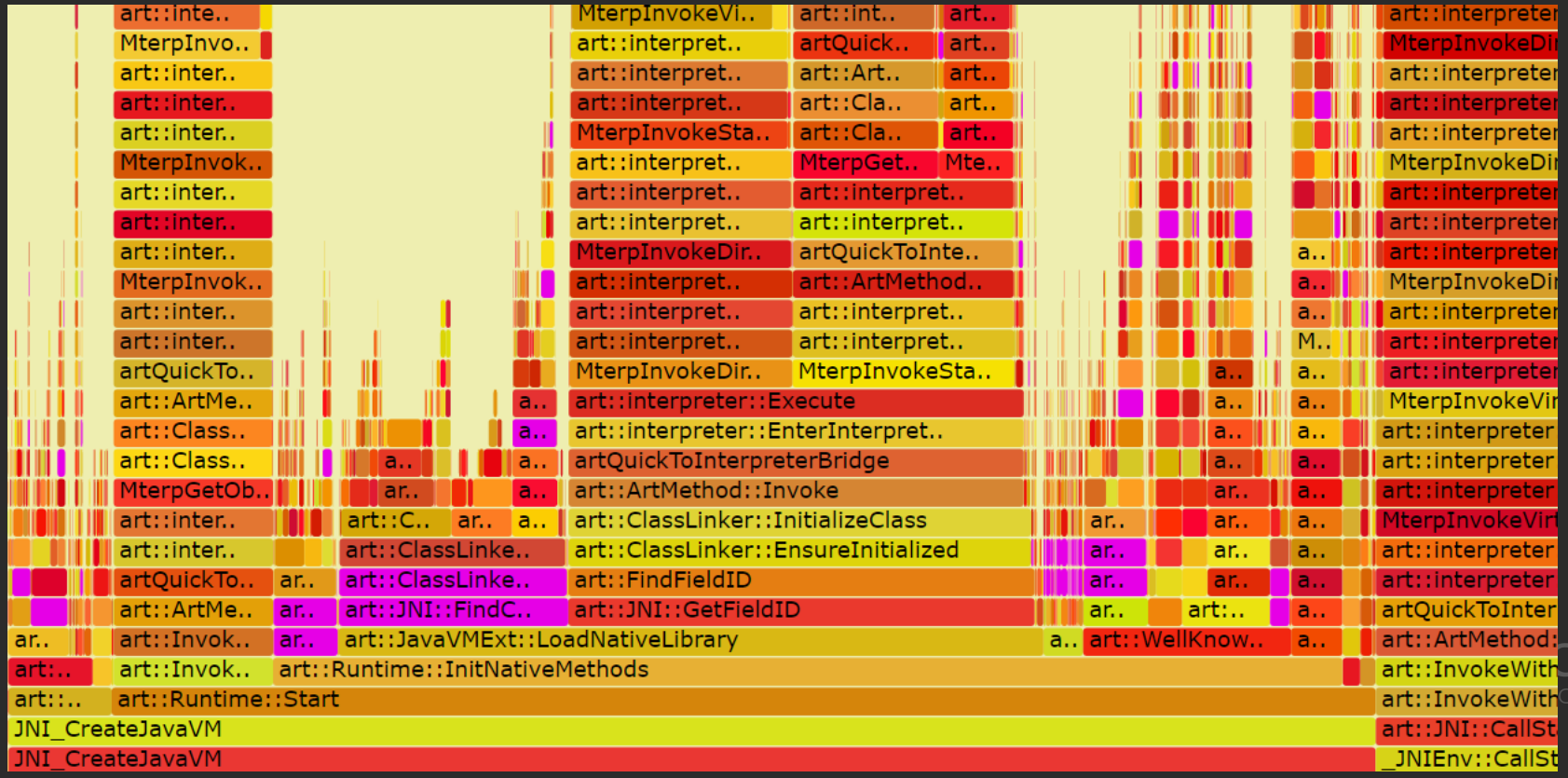
Tracing for profit

Uftrace TUI를 통해 소스 브라우징을 할 수 있다는 것!



Tracing for profit

36.2%의 Class Load, Class 지옥의 주역

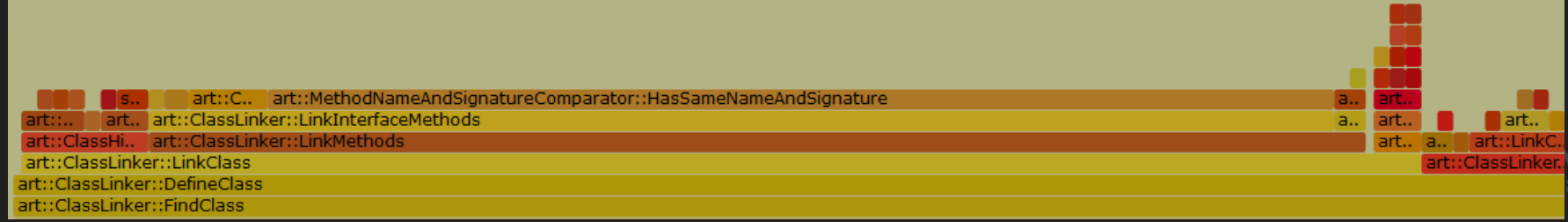


Tracing for profit

Class가 로드되는 Path를 찾았습니다!

Class Path

art::ClassLinker::LinkInterfaceMethods
art::ClassLinker::LinkMethods
art::ClassLinker::LinkClass
art::ClassLinker::DefineClass
art::ClassLinker::FindClass



Tracing for profit

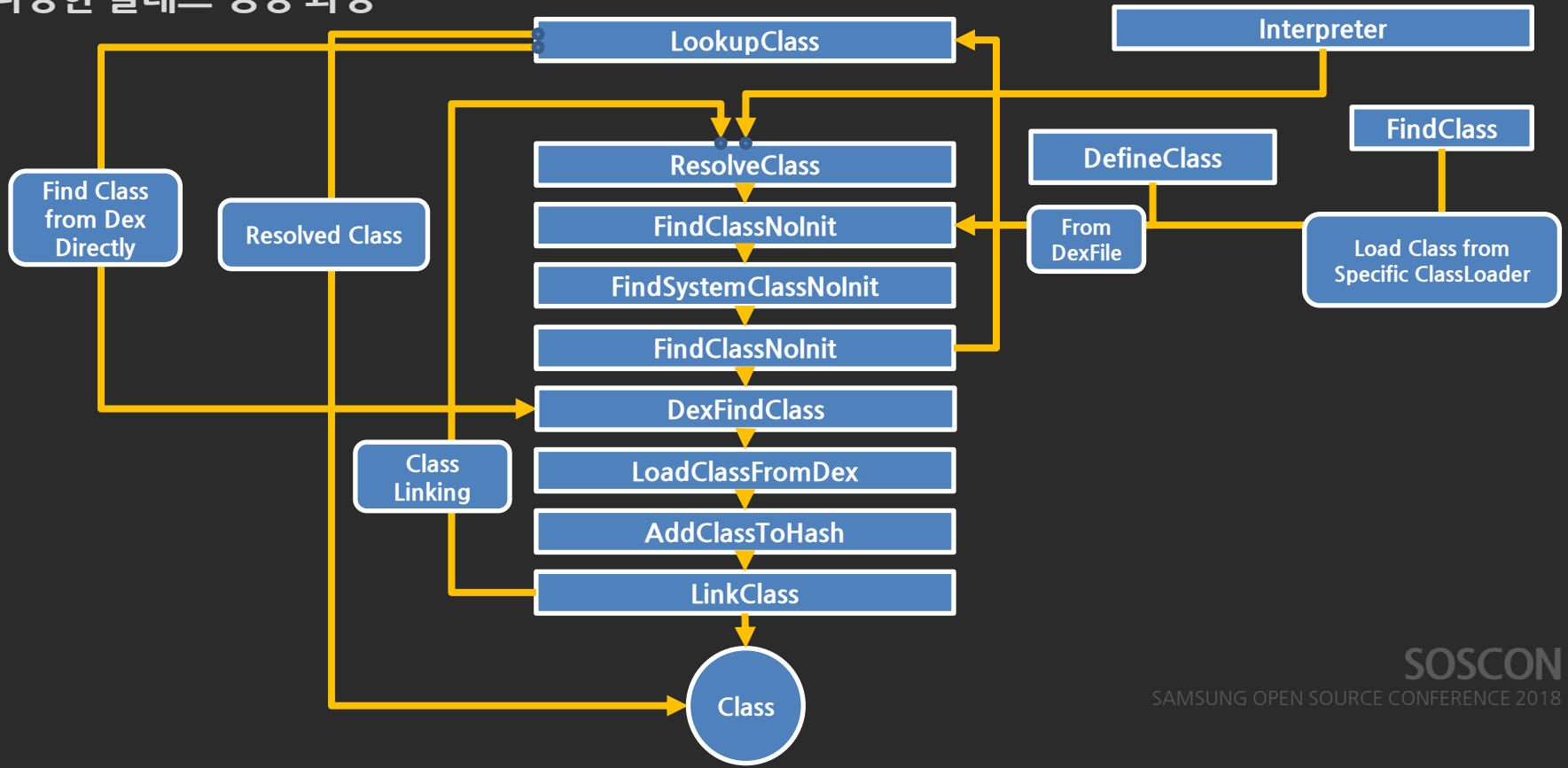
Uftrace TUI를 통해 소스 브라우징! 어메이징!

```
  TOTAL TIME : FUNCTION
             :========== Call Graph ==========
   66.328 ms : (589) art::ClassLinker::FindClass
   28.439 us :   ├─(589) art::Thread::AssertNoPendingException
             :   │
   42.686 us :   ├─(571) art::ComputeModifiedUtf8Hash
             :   │
  368.431 us :   ├─(571) art::ClassLinker::LookupClass
  265.979 us :   │ (570) art::ClassTable::Lookup
  172.374 us :   │ (570) art::HashSet::FindIndex
  106.421 us :   │ (253) art::mirror::Class::DescriptorEquals
   11.443 us :   │   ├─(236) art::mirror::Class::GetClassDef
             :   │   │
    0.417 us :   │   └─(6) art::Primitive::Descriptor
             :   │
  220.261 us :   ├─(491) art::OatDexFile::FindClassDef
   77.786 us :   │ (409) art::TypeLookupTable::IsStringsEquals
             :   │
   63.938 ms :   ├─(396) art::ClassLinker::DefineClass
   63.411 us :   │   ├─(1188) art::Runtime::GetRuntimeCallbacks
             :   │   │
  102.293 us :   │   ├─(396) art::RuntimeCallbacks::ClassPreDefine
   18.337 us :   │   │ (396) art::ClassLoadCallback::ClassPreDefine
             :   │   │
  247.186 us :   │   ├─(396) art::ClassLinker::RegisterDexFile
  178.648 us :   │   │ (396) art::Thread::DecodeJObject
  117.417 us :   │   │ (396) art::JavaVMExt::DecodeWeakGlobal
   30.997 us :   │   │ (396) art::IndirectReferenceTable::GetChecked
             :   │   │
   24.893 us :   │   ├─(396) art::mirror::Class::SetDexCache
             :   │   │
  291.060 us :   │   ├─(396) art::ClassLinker::SetupClass
   19.688 us :   │   │   ├─(396) art::mirror::Class::SetClassLoader
             :   │   │   │
   34.267 us :   │   │   ├─(396) art::mirror::Class::SetStatus
```

```
mirror::Class* ClassLinker::FindClass(Thread* self,
                                      const char* descriptor,
                                      Handle<mirror::ClassLoader> class_loader)
  DCHECK_NE(*descriptor, '¥0') << "descriptor is empty string";
  DCHECK(self != nullptr);
  self->AssertNoPendingException();
  self->PoisonObjectPointers();  // For DefineClass, CreateArrayClass, etc...
  if (descriptor[1] == '¥0') {
    // only the descriptors of primitive types should be 1 character long, also
    // for primitive classes that aren't backed by dex files.
    return FindPrimitiveClass(descriptor[0]);
  }
  const size_t hash = ComputeModifiedUtf8Hash(descriptor);
  // Find the class in the loaded classes table.
  ObjPtr<mirror::Class> klass = LookupClass(self, descriptor, hash, class_loader
  if (klass != nullptr) {
    return EnsureResolved(self, descriptor, klass);
  }
  // Class is not yet loaded.
  if (descriptor[0] != '[' && class_loader == nullptr) {
    // Non-array class and the boot class loader, search the boot class path.
    ClassPathEntry pair = FindInClassPath(descriptor, hash, boot_class_path_);
    if (pair.second != nullptr) {
      return DefineClass(self,
                         descriptor,
                         hash,
                         ScopedNullHandle<mirror::ClassLoader>(),
                         *pair.first,
                         *pair.second);
    } else {
      // The boot class loader is searched ahead of the application class loader
      // expected and will be wrapped in a ClassNotFoundException. Use the pre-a
      // trigger the chaining with a proper stack trace.
      ObjPtr<mirror::Throwable> pre_allocated =
```

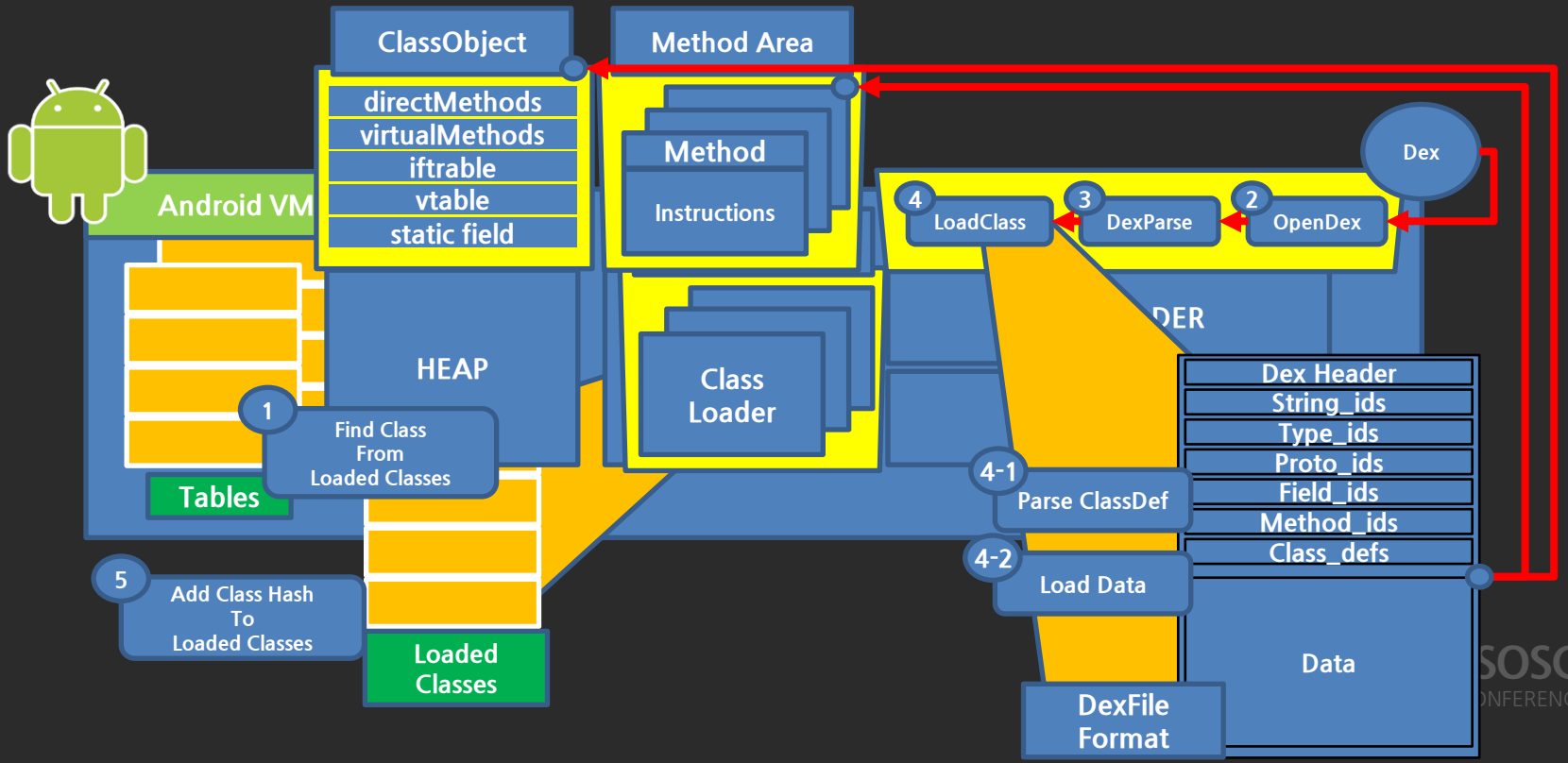
Tracing for profit

다양한 클래스 생성 과정



Tracing for profit

Class를 로드하는 Path를 Tracing 해냈습니다!



Tracing for profit

Hello world!

```
# direct methods
method public constructor <init>()V
    .registers 1

    .prologue
    .line 17
    invoke-direct {p0}, Ljava/lang/Object;-><init>()V
    return-void
.end method

method public static main([Ljava/lang/String;)V
    .registers 3

    .prologue
    .line 19
    sget-object v0,
    Ljava/lang/System;->out:Ljava/io/PrintStream;
    const-string v1, "Hello, world!"
    invoke-virtual {v0, v1},
    Ljava/io/PrintStream;->println(Ljava/lang/String;)V
    .line 20
    return-void
.end method
```

```
ScopedLocalRef<jclass> klass(env, env->FindClass(class_name.c_str()));
if (klass.get() == nullptr) {
    fprintf(stderr, "Unable to locate class '%s'₩n", class_name.c_str());
    env->ExceptionDescribe();
    return EXIT_FAILURE;
}

jmethodID method = env->GetStaticMethodID(klass.get(), "main",
"([Ljava/lang/String;)V");
if (method == nullptr) {
    fprintf(stderr, "Unable to find static main(String[]) in '%s'₩n",
class_name.c_str());
    env->ExceptionDescribe();
    return EXIT_FAILURE;
}

// Make sure the method is public.  JNI doesn't prevent us from
// calling a private method, so we have to check it explicitly.
if (!IsMethodPublic(env, klass.get(), method)) {
    fprintf(stderr, "Sorry, main() is not public in '%s'₩n",
class_name.c_str());
    env->ExceptionDescribe();
    return EXIT_FAILURE;
}
```

Tracing for Fun

Gcc

Tracing For Fun

Gcc를 사용해 컴파일 했는데 의도한 대로 동작하지 않는 상황

```
static inline __attribute__((always_inline)) char *get_msg()
{
    char str[64] = "Hello world₩n";
    return (char *)str;
}

int main()
{
    char *p;
    p = get_msg();
    puts(p);
    return 0;
}
```

```
$ ./compile_by_gcc
Segmentation fault (core dumped)
```

```
$ ./compile_by_clang
Hello world
```

Tracing For Fun

Clang과 비교해봤습니다

```
push   %rbp
mov    %rsp,%rbp
sub    $0x50,%rsp
movabs $0x6f77206f6c6c6548,%rax
mov    $0xa646c72,%edx
mov    %rax,-0x50(%rbp)
mov    %rdx,-0x48(%rbp)
movq   $0x0,-0x40(%rbp)
movq   $0x0,-0x38(%rbp)
movq   $0x0,-0x30(%rbp)
movq   $0x0,-0x28(%rbp)
movq   $0x0,-0x20(%rbp)
movq   $0x0,-0x18(%rbp)
mov    $0x0,%eax
mov    %rax,-0x8(%rbp)
mov    -0x8(%rbp),%rax
mov    %rax,%rdi
callq  0x400430 <puts@plt>
mov    $0x0,%eax
leaveq
retq
```

Gcc

```
push   %rbp
mov    %rsp,%rbp
sub    $0x60,%rsp
lea    -0x40(%rbp),%rax
movl   $0x0,-0x44(%rbp)
mov    %rax,%rcx
movaps 0x103(%rip),%xmm0      # 0x400650
movaps %xmm0,0x30(%rcx)
movaps 0xe8(%rip),%xmm0       # 0x400640
movaps %xmm0,0x20(%rcx)
movaps 0xcd(%rip),%xmm0       # 0x400630
movaps %xmm0,0x10(%rcx)
movaps 0xb2(%rip),%xmm0       # 0x400620
movaps %xmm0,(%rcx)
mov    %rax,-0x50(%rbp)
mov    -0x50(%rbp),%rdi
mov    $0x0,%al
callq  0x400410 <puts@plt>
xor    %edx,%edx
mov    %eax,-0x54(%rbp)
mov    %edx,%eax
add    $0x60,%rsp
pop    %rbp
retq
```

Clang

Tracing For Fun

Gcc 넌 나에게 0을 줬어…

```
push   %rbp
mov    %rsp,%rbp
sub    $0x50,%rsp
movabs $0x6f77206f6c6c6548,%rax
mov    $0xa646c72,%edx
mov    %rax,-0x50(%rbp)
mov    %rdx,-0x48(%rbp)
movq   $0x0,-0x40(%rbp)
movq   $0x0,-0x38(%rbp)
movq   $0x0,-0x30(%rbp)
movq   $0x0,-0x28(%rbp)
movq   $0x0,-0x20(%rbp)
movq   $0x0,-0x18(%rbp)
mov    $0x0,%eax
mov    %rax,-0x8(%rbp)
mov    -0x8(%rbp),%rax
mov    %rax,%rdi
callq  0x400430 <puts@plt>
mov    $0x0,%eax
leaveq
retq
```

```
mov    $0x0,%eax
mov    %rax,-0x8(%rbp)
mov    -0x8(%rbp),%rax
mov    %rax,%rdi
```

Why?

Tracing For Fun

코드를 변경해야 하나? #1 - 문자열 포인터 반환으로 변경

```
static inline __attribute__((always_inline)) char *get_msg()
{
    char *str = "Hello world₩n";
    return str;
}

int main()
{
    char *p;
    p = get_msg();
    puts(p);
    return 0;
}
```

Tracing For Fun

코드를 변경해야 하나? #1 - 문자열 포인터를 반환하게 바꾸면 내용을 수정 못함

```
static inline __attribute__((always_inline)) char *get_msg()
{
    char *str = "Hello world₩n";
    return str;
}

int main()
{
    char *p;
    p = get_msg();
    p[1] = 'a';
    puts(p);
    return 0;
}
```

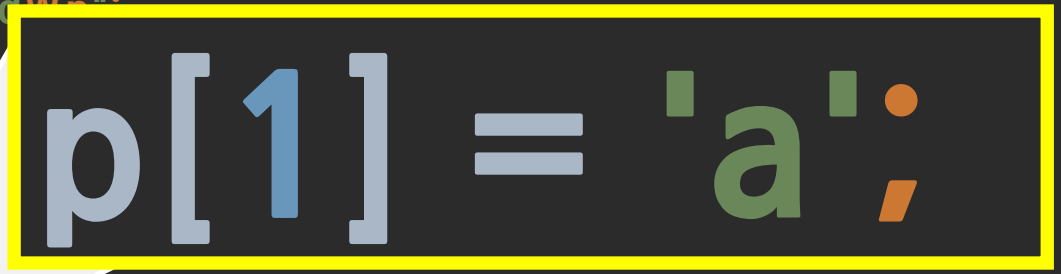
```
$ ./compile_by_gcc
Segmentation fault (core dumped)
```

```
$ ./compile_by_clang
Segmentation fault (core dumped)
```

Tracing For Fun

코드를 변경해야 하나? #1 - “Hallo world”를 출력하고 싶은데요...

```
static inline __attribute__((always_inline)) char *get_msg()
{
    char *str = "Hello world\n";
    return str;
}

int main()
{
    char *p;
    p = get_msg();
    p[1] = 'a';
    puts(p);
    return 0;
}
```



Tracing For Fun

코드를 변경해야 하나? #1 - 문자열은 Const Char * 형으로 Data Section에 저장

```
static inline __attribute__((always_inline)) char *get_msg()
{
    char *str = "Hello world₩n";
    return str;
}

int main()
{
    char *p;
    p = get_msg();
    p[1] = 'a';
    puts(p);
    return 0;
}
```

Const char *

```
$ objdump -s compile_by_gcc
...
Contents of section .rodata:
 4005f0 01000200 48656c6c 6f20776f 726c640a  ....Hello world.
 400600 00
...
```

Tracing For Fun

코드를 변경해야 하나? #2 - 동적 할당

```
static inline __attribute__((always_inline)) char *get_msg()
{
    const char *str = "Hello world₩n";

    char *p = malloc(strlen(str));
    strcpy(p, str);
    return p;
}

int main()
{
    char *p;
    p = get_msg();
    puts(p);
    free(p);
    return 0;
}
```

동적 할당으로 변경하면
해제를 해줘야...

Tracing For Fun

재미로 원인 추적하기 - Warning 메시지를 따라가보자

```
$ gcc inline.c
inline.c: In function ‘get_msg:
inline.c:6:9: warning: function returns address of local variable [-Wreturn-local-addr]
return (char *)str;
```

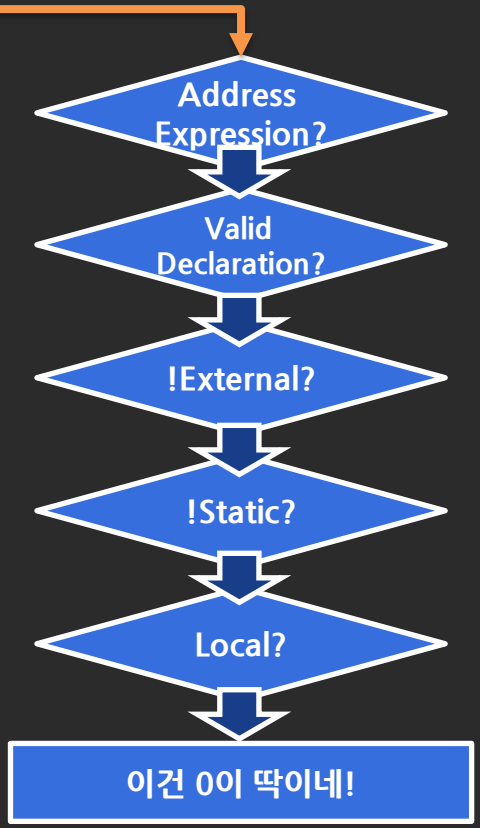
Tracing For Fun

```
10105 tree
10106 c_finish_return (location_t loc, tree retval, tree origtype)
...
10227           case ADDR_EXPR:
10228             inner = TREE_OPERAND (inner, 0);
10229
10230             while (REFERENCE_CLASS_P (inner)
10231                    && !INDIRECT_REF_P (inner))
10232               inner = TREE_OPERAND (inner, 0);
10233
10234             if (DECL_P (inner)
10235                 && !DECL_EXTERNAL (inner)
10236                 && !TREE_STATIC (inner)
10237                 && DECL_CONTEXT (inner) == current_function_decl)
10238               {
10239                 if (TREE_CODE (inner) == LABEL_DECL)
10240                   warning_at (loc, OPT_Wreturn_local_addr,
10241                               "function returns address of label");
10242                 else
10243                   {
10244                     warning_at (loc, OPT_Wreturn_local_addr,
10245                                 "function returns address of local variable");
10246                     tree zero = build_zero_cst (TREE_TYPE (res));
10247                     t = build2 (COMPOUND_EXPR, TREE_TYPE (res), t, zero);
10248                   }
10249               }
10250             break;
...
```

Tracing For Fun

str이 return 시 0이 되는 이유가 이렇습니다.

```
static inline __attribute__((always_inline))
char *get_msg()
{
    char str[64] = "Hello world₩n";

    return statement
        operand
    return (char *)str;

}
```



Tracing For Fun

```
10105 tree
10106 c_finish_return (location_t loc, tree retval, tree origtype)
...
10227         case ADDR_EXPR:
10228          inner = TREE_OPERAND (inner, 0);
10229
10230          while (REFERENCE_CLASS_P (inner)
10231                 && !INDIRECT_REF_P (inner))
10232            inner = TREE_OPERAND (inner, 0);
10233
10234          if (DECL_P (inner)
10235              && !DECL_EXTERNAL (inner)
10236              && !TREE_STATIC (inner)
10237              && DECL_CONTEXT (inner) == current_function_d
10238            {
10239              if (TREE_CODE (inner) == LABEL_DECL)
10240                warning_at (loc, OPT_Wreturn_local_addr,
10241                            "function returns address of label");
10242              else
10243                {
10244                  warning_at (loc, OPT_Wreturn_local_add
10245                              "function returns address of local variable");
10246                  tree zero = build_zero_cst (TREE_TYPE (res));
10247                  t = build2 (COMPOUND_EXPR, TREE_TYPE (res), t, zero);
10248                }
10249            }
10250          break;
...
```

```
mov    $0x0,%eax
mov    %rax,-0x8(%rbp)
mov    -0x8(%rbp),%rax
mov    %rax,%rdi
```

Tracing For Fun

왜 inline function 체크를 하지 않을까요?

```
(gdb) ptype tree
tree_node {
    tree_base base;
    tree_typed typed;
    tree_common common;
    tree_int_cst int_cst;
    tree_poly_int_cst poly_int_cst;
    tree_real_cst real_cst;
    tree_fixed_cst fixed_cst;
    tree_vector vector;
    tree_string string;
    tree_complex complex;
    tree_identifier identifier;
    tree_decl_minimal decl_minimal;
    tree_decl_common decl_common;
    tree_decl_with_rtl decl_with_rtl;
    tree_decl_non_common decl_non_common;
    tree_parm_decl parm_decl;
    tree_decl_with_vis decl_with_vis;
    tree_var_decl var_decl;
    tree_field_decl field_decl;
    tree_label_decl label_decl;
    tree_result_decl result_decl;
    tree_const_decl const_decl;
    tree_function_decl function_decl;
    tree_translation_unit_decl translation_unit_decl;
    tree_type_common type_common;
    tree_type_with_lang_specific type_with_lang_specific;
    tree_type_non_common type_non_common;
    tree_list list;
    tree_vec vec;
    tree_exp exp;
    tree_ssa_name ssa_name;
    tree_block block;
    tree_binfo binfo;
    tree_statement_list stmt_list;
    tree_constructor constructor;
    tree_omp_clause omp_clause;
    tree_optimization_option optimization;
    tree_target_option target_option;
} *
```

Tree_node 중 function을 관장하는 구조체

tree_function_decl
function_base

Tracing For Fun

str이 속한 함수 get_msg에는 inline_flag가 set되어 있습니다

```
static inline __attribute__((always_inline))
```

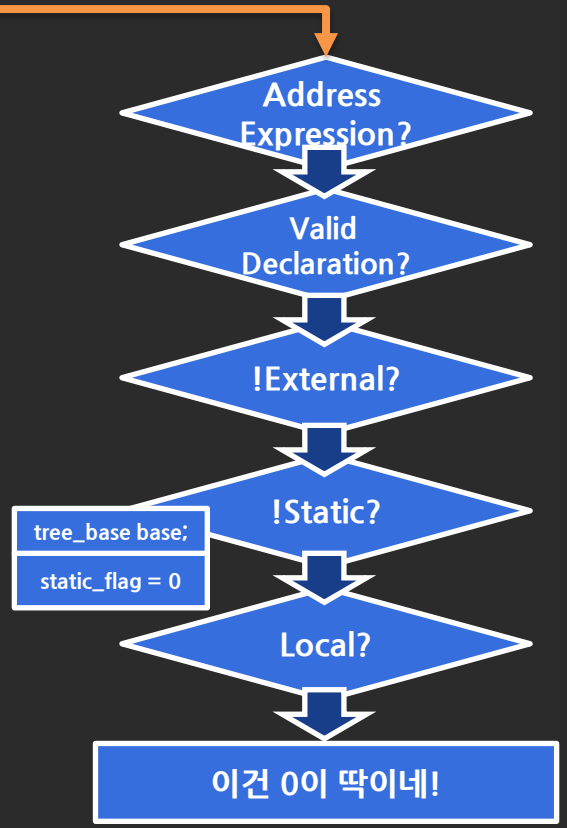
Current_function_decl.function_decl

static_flag = 1 | declared_inline_flag = 1

return statement

operand

```
return (char *)str;
}
```



Tracing For Fun

그래서 한번 바꿔봤습니다!

```
if (DECL_P (inner)
    && !DECL_EXTERNAL (inner)
    && !TREE_STATIC (inner)
    && !DECL_DECLARED_INLINE_P(current_function_decl)
    && DECL_CONTEXT (inner) == current_function_decl)
```

```
10237          && !DECL_DECLARED_INLINE_P(current_function_decl)
10236          && !TREE_STATIC (inner)
10237          && DECL_CONTEXT (inner) == current_function_decl)
10238        {
10239          if (TREE_CODE (inner) == LABEL_DECL)
10240            warning_at (loc, OPT_Wreturn_local_addr,
10241                        "function returns address of label");
10242          else
10243            {
10244              warning_at (loc, OPT_Wreturn_local_addr,
10245                          "function returns address of local variable");
10246              tree zero = build_zero_cst (TREE_TYPE (res));
10247              t = build2 (COMPOUND_EXPR, TREE_TYPE (res), t, zero);
10248            }
10249        }
10250      break;
```

return 될 tree가 속한 함수가
inline 으로 선언되어 있으면
0으로 바꾸지 마세요!

Tracing For Fun

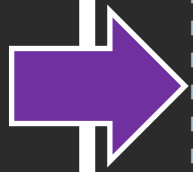
이런? 최적화 옵션을 주면 “Hello world”가 출력이 안됩니다!

```
$ gcc inline.c -o 1st_code_modify
$ ./1st_code_modify
Hello world

$ gcc -O1 inline.c 1st_code_modify
$ ./1st_code_modify
```

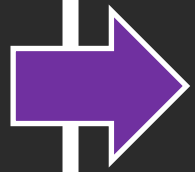
Tracing For Fun

디스어셈블을 통해 문제를 파악해 봅시다

```
push   %rbp
mov    %rsp,%rbp
sub    $0x50,%rsp
movabs $0x6f77206f6c6c6548,%rax
mov    $0xa646c72,%edx
mov    %rax,-0x50(%rbp)
mov    %rdx,-0x48(%rbp)
movq   $0x0,-0x40(%rbp)
movq   $0x0,-0x38(%rbp)
movq   $0x0,-0x30(%rbp)
movq   $0x0,-0x28(%rbp)
movq   $0x0,-0x20(%rbp)
movq   $0x0,-0x18(%rbp)
mov    $0x0,%eax
mov    %rax,-0x8(%rbp)
mov    -0x8(%rbp),%rax
mov    %rax,%rdi
callq  0x400430 <puts@plt>
mov    $0x0,%eax
leaveq
retq
```



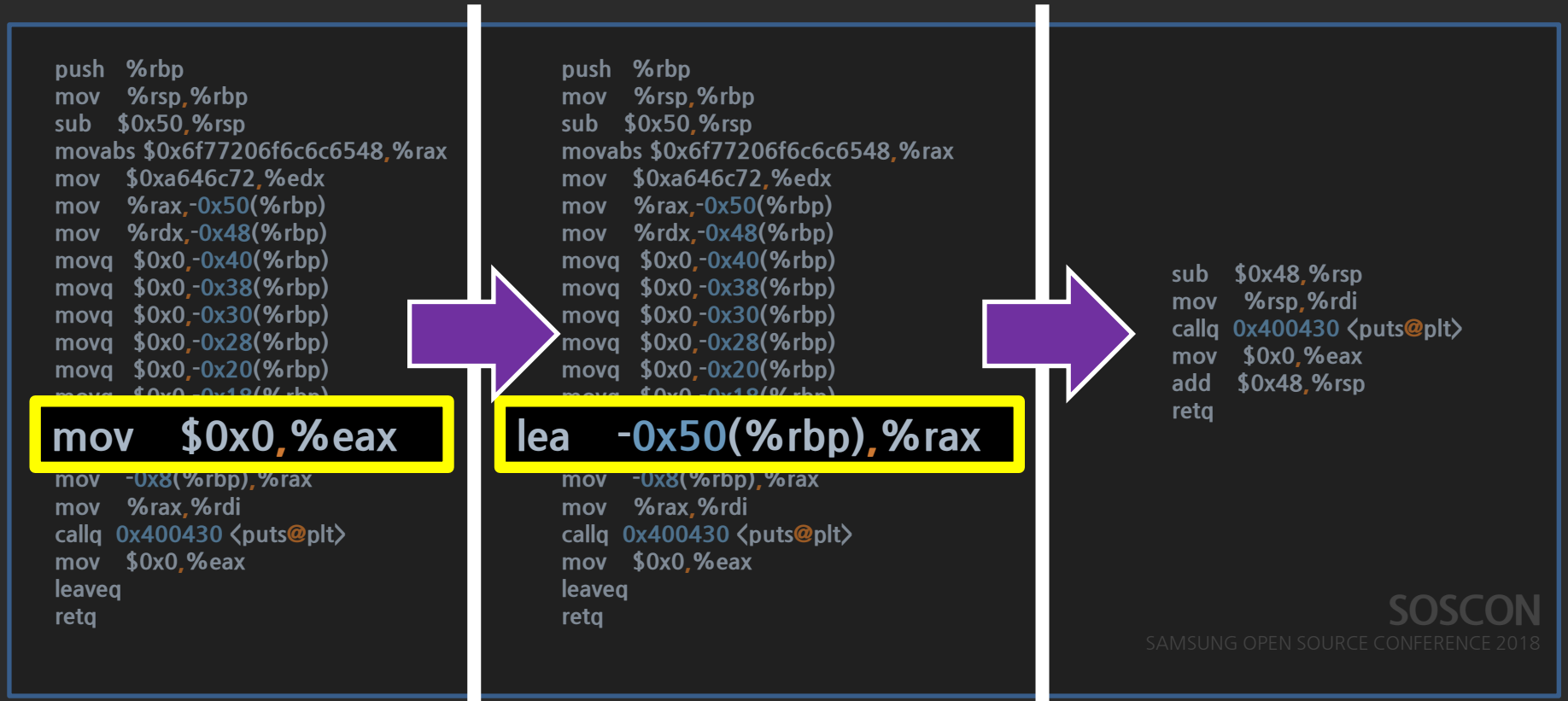
```
push   %rbp
mov    %rsp,%rbp
sub    $0x50,%rsp
movabs $0x6f77206f6c6c6548,%rax
mov    $0xa646c72,%edx
mov    %rax,-0x50(%rbp)
mov    %rdx,-0x48(%rbp)
movq   $0x0,-0x40(%rbp)
movq   $0x0,-0x38(%rbp)
movq   $0x0,-0x30(%rbp)
movq   $0x0,-0x28(%rbp)
movq   $0x0,-0x20(%rbp)
movq   $0x0,-0x18(%rbp)
lea    -0x50(%rbp),%rax
mov    %rax,-0x8(%rbp)
mov    -0x8(%rbp),%rax
mov    %rax,%rdi
callq  0x400430 <puts@plt>
mov    $0x0,%eax
leaveq
retq
```



```
sub    $0x48,%rsp
mov    %rsp,%rdi
callq  0x400430 <puts@plt>
mov    $0x0,%eax
add    $0x48,%rsp
retq
```

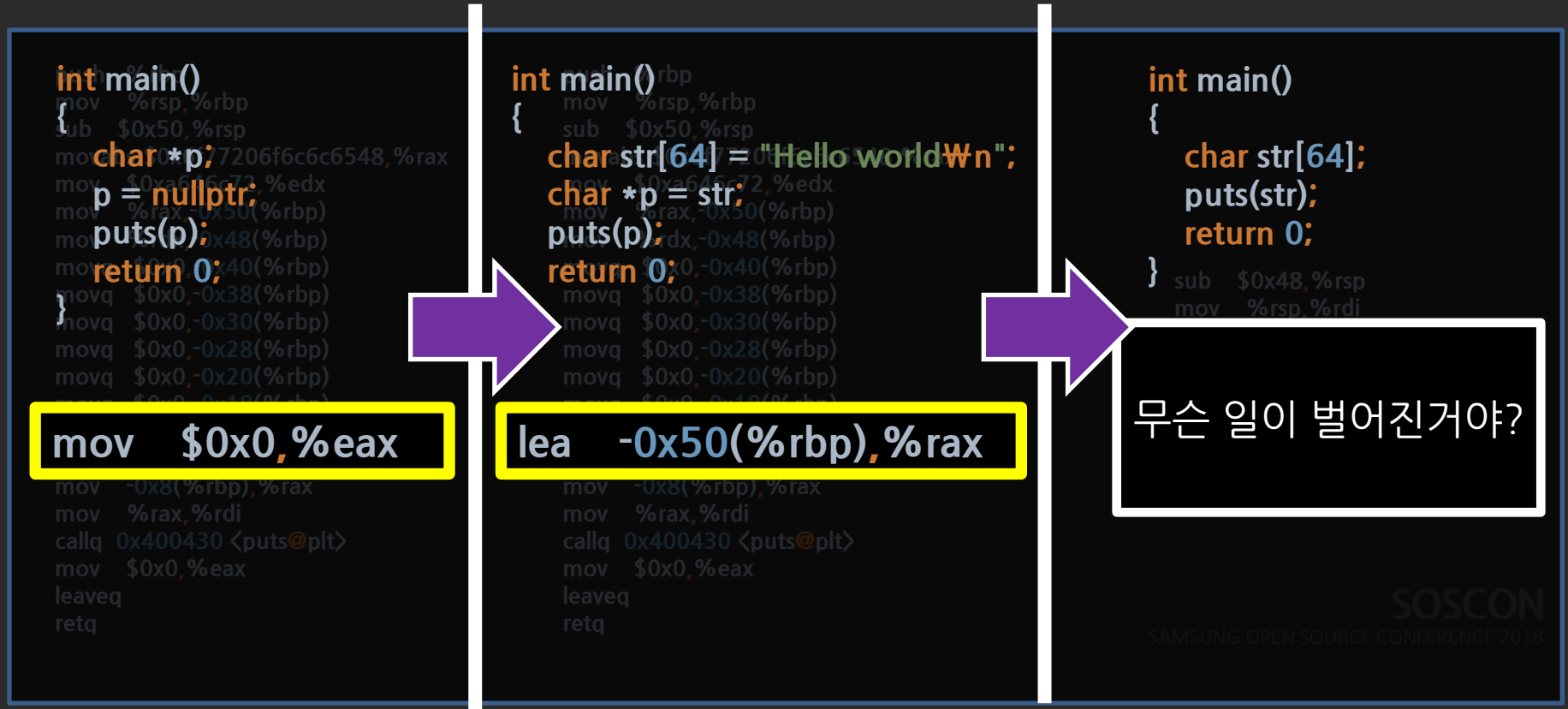
Tracing For Fun

0 이 되버리던 str을 제대로 반환하게 하는데 성공했지만



Tracing For Fun

최적화 옵션을 주면 “Hello world”가 아예 증발됩니다



Tracing For Fun

$ gcc -O1 -fdump-tree-all-details inline.c

```
;; Function get_msg (get_msg, funcdef_no=0, decl_uid=1792, cgraph_uid=0, symbol_order=0)
Deleted dead store: str = "Hello world₩n";
__attribute__((always_inline))  get_msg ()
{
  char str[64];
  <bb 2> [0.00%]:
  str ={v} {CLOBBER};
  return &str;
}

;; Function main (main, funcdef_no=1, decl_uid=1796, cgraph_uid=1, symbol_order=1)
main ()
{
  char str[64];
  char * D.1810;
  char * p;

  <bb 2> [100.00%]:
  str ={v} {CLOBBER};
  puts (&str);
  return 0;
}
```

Tracing For Fun

$ gcc -O1 -fdump-tree-all-details inline.c

```
;; Function get_msg (get_msg, funcdef_no=0, decl_uid=1792, cgraph_uid=0, symbol_order=0)
Deleted dead store: str = "Hello world₩n";
__attribute__((always_inline)) get_msg ()
{
  char str[64];
  <bb 2> [0.00%]:
  str ={v} {CLOBBER};
  return &str;
}

;; Function main (main, funcdef_no=1, decl_uid=1796, cgraph_uid=1, symbol_order=1)
main ()
{
  char str[64];
  char * D.1810;
  char * p;

  <bb 2> [100.00%]:
  str ={v} {CLOBBER};
  puts (&str);
  return 0;
}
```

inline.c.040t.dse1
Deleted dead store: str = "Hello world₩n";

Tracing For Fun

함수 인자 값을 함께 기록하도록 옵션을 지정하여 Tracing 해봅니다

```
$ uftrace record --auto-args -A fwrite@arg1/s ₩
                 `which gcc` -O1 -fdump-tree-all-details ₩
                 inline.c
```

Tracing For Fun

Tracing 결과에서 Deleted dead store 검색하면 위치를 찾을 수 있습니다

```
$ uftrace replay > replay.log
```

```
$ wc -lc replay.log
4157866 line 392760323 charater    replay.log
```

```
$ grep "Deleted" -B2 replay.log
[110797] |  delete_dead_assignment(0x7ffe0825b230) {
[110797] |      gsi_stmt();
[110797] |      fwrite("  Deleted dead store: ") = 22;
```

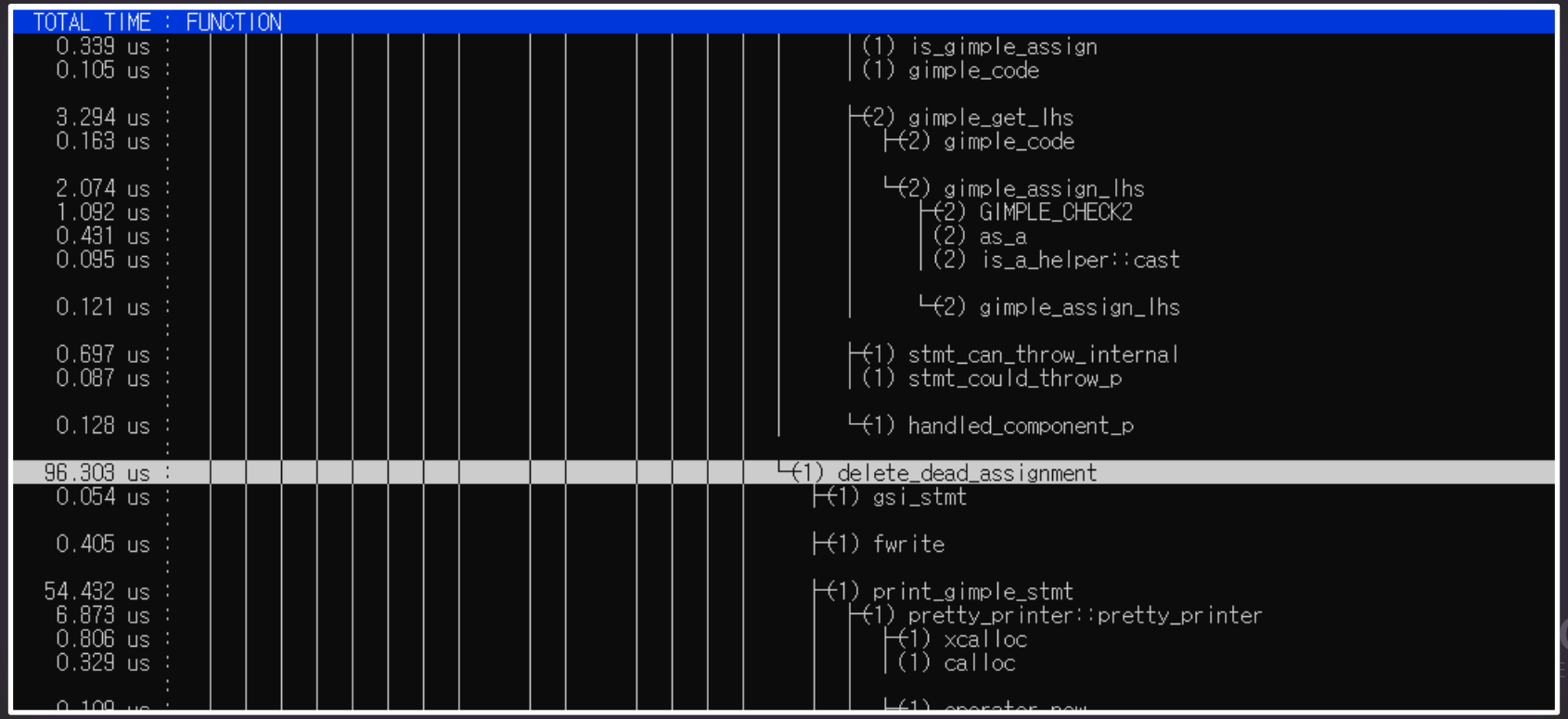
Tracing For Fun

Uftrace TUI를 통한 소스 브라우징! - delete_dead_assignment 검색

```
  TOTAL TIME : FUNCTION
  792.775 ms : (1) cc1
    1.322 us :   ├─(2) _dl_relocate_static_pie
    0.168 us :   │ (2) _dl_relocate_static_pie
             :   │
   79.360 us :   ├─(1) _GLOBAL__sub_I_last_id
   79.082 us :   │ (1) __static_initialization_and_destruction_0
   71.531 us :   │   ├─(1) mem_alloc_description::mem_alloc_description
    5.846 us :   │   │ ├─(3) operator new
    2.083 us :   │   │ │ (3) malloc
             :   │   │ │
   56.906 us :   │   │ └─(3) hash_map::hash_map
   56.033 us :   │   │   (3) hash_table::hash_table
    4.912 us :   │   │     ├─(3) hash_table_higher_prime_index
             :   │   │     │
   42.725 us :   │   │     └─(3) hash_table:lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
   11.295 us :   │   │       ├─(3) xcallocaxSearch function:                                              x
    4.297 us :   │   │       │ (3) xcalloc x (press ESC to exit)                                          x
    2.609 us :   │   │       │ (3) calloc  x                                                              x
             :   │   │       │             x                                                              x
   26.083 us :   │   │       └─(39) hash_tax  delete_dead_assignment_                                     x
   18.472 us :   │   │         (39) hash_max                                                              x
   13.318 us :   │   │         (39) simple_mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
    1.736 us :   │   │         (39) pointer_hash::mark_empty
             :   │   │
    0.165 us :   │   └─(1) __cxa_atexit
             :   │
   39.290 us :   ├─(1) _GLOBAL__sub_I_bitmap_mem_desc
   39.143 us :   │ (1) __static_initialization_and_destruction_0
   34.078 us :   │   ├─(1) mem_alloc_description::mem_alloc_description
```

Tracing For Fun

delete_dead_assignment 를 찾았습니다



Tracing For Fun

소스코드를 찾을 수 있을 때 까지 U키를 눌러 상위 함수로 이동합니다

```
TOTAL TIME : FUNCTION
           :========== Call Graph ==========
301.968 us : (12) dse_dom_walker::dse_optimize_stmt
  0.619 us :   ├─(12) gsi_stmt
           :
 16.375 us :   ├─▶(12) gimple_vdef
           :
  4.589 us :   ├─▶(8) gimple_has_volatile_ops
           :
 20.436 us :   ├─▶(5) gimple_clobber_p
           :
  3.761 us :   ├─▶(5) gimple_assign_lhs
           :
 35.591 us :   ├─▶(4) initialize_ao_ref_for_dse
           :
  0.475 us :   ├─▶(1) gimple_call_builtin_p
           :
  0.240 us :   ├─▶(1) is_gimple_assign
           :
  0.678 us :   ├─▶(1) gimple_assign_rhs1
           :
  2.066 us :   ├─▶(1) operand_equal_p
           :
 34.533 us :   ├─▶(1) setup_live_bytes_from_ref
           :
 73.969 us :   ├─▶(1) dse_classify_store
           :
 96.303 us :   └─(1) delete_dead_assignment
  0.054 us :     ├─(1) gsi_stmt
```

dse_dom_walker::dse_optimize_stmt

delete_dead_assignment

Tracing For Fun

한 번 더 하면, pass_dse 에 도달하게 됩니다

```
 TOTAL TIME : FUNCTION
            :=== Function Call Graph for '_GLOBAL__N_1::pass_dse::execute' ===
            :========== Back-trace ==========
  408.805 us :   ├─(2) _GLOBAL__N_1::pass_dse::execute
  408.805 us :   │ ▶(2) execute_one_pass
             :
  202.731 us :   └─(2) _GLOBAL__N_1::pass_d
  202.731 us :     ▶(2) execute_one_pass
             :
            :========== Call Graph ==========
  611.536 us : (4) _GLOBAL__N_1::pass_dse::execute
    2.080 us :   ├▶(4) bitmap_obstack_alloc_stat
             :
   33.851 us :   ├▶(4) renumber_gimple_stmt_uids
             :
  174.083 us :   ├▶(8) calculate_dominance_info
             :
    7.924 us :   ├▶(4) dse_dom_walker::dse_dom_walker
             :
  343.550 us :   ├▶(4) dom_walker::walk
             :
    2.889 us :   ├▶(4) dse_dom_walker::~dse_dom_walker
             :
    0.334 us :   ├─(4) bitmap_empty_p
             :
    1.393 us :   ├▶(4) bitmap_obstack_free
             :
uftrace graph: ../../gcc-7.3.0/gcc/tree-ssa-dse.c [line:839]
 0:vi-   1:uftrace*
```

_GLOBAL__N_1::pass_dse::execute

dse_dom_walker::dse_optimize_stmt

delete_dead_assignment

Tracing For Fun

/* This file implements dead store elimination.

A dead store is a store into a memory location which will later be overwritten by another store without any intervening loads. In this case the earlier store can be deleted.

In our SSA + virtual operand world we use immediate uses of virtual operands to detect dead stores. If a store's virtual definition is used precisely once by a later store to the same location which post dominates the first store, then the first store is dead.

The single use of the store's virtual definition ensures that there are no intervening aliased loads and the requirement that the second load post dominate the first ensures that if the earlier store executes, then the later stores will execute before the function exits.

It may help to think of this as first moving the earlier store to the point immediately before the later store. Again, the single use of the virtual definition and the post-dominance relationship ensure that such movement would be safe. Clearly if there are back to back stores, then the second is redundant.

Reviewing section 10.7.2 in Morgan's "Building an Optimizing Compiler" may also help in understanding this code since it discusses the relationship between dead store and redundant load elimination. In fact, they are the same transformation applied to different views of the CFG. */

DSE(Dead Store Elimination)

tree-ssa-dse.c

Tracing For Fun

최적화 때문이라면 최적화를 하지 못하게 강제하고 비교/확인해 볼까요?

```
static char *get_msg()
{

    char str[64] = "Hello world₩n";
    return (char *)str;
}

int main()
{
    char *p;
    p = get_msg();
    puts(p);
    return 0;
}
```

Tracing For Fun

최적화 때문이라면 최적화를 하지 못하게 강제하고 비교/확인해 볼까요?

```
static char *get_msg()
{
    volatile char str[64] = "Hello world₩n";
    r   har *)str;
}

int m
{
    char *p;
    p = get_msg();
    puts(p);
    return 0;
}
```

메모리 최적화 방지

```
static char *get_msg()
{
    char str[64] = "Hello world₩n";
    n (char *)str;
}

i
{
    char *p;
    p = get_msg();
    puts(p);
    return 0;
}
```

메모리 최적화

Tracing For Fun

$ gcc -O1 -fdump-tree-all-details inline.c

```
;; Function get_msg (get_msg, funcdef_no=0)
get_msg ()
{
  volatile char str[64];

  <bb 2> [0.00%]:
  str ={v} "Hello world₩n";
  return 0B;
}

;; Function main (main, funcdef_no=1)
main ()
{
  char * p;

  <bb 2> [0.00%]:
  p_3 = get_msg ();
  puts (p_3);
  return 0;

}
```

메모리 최적화 방지

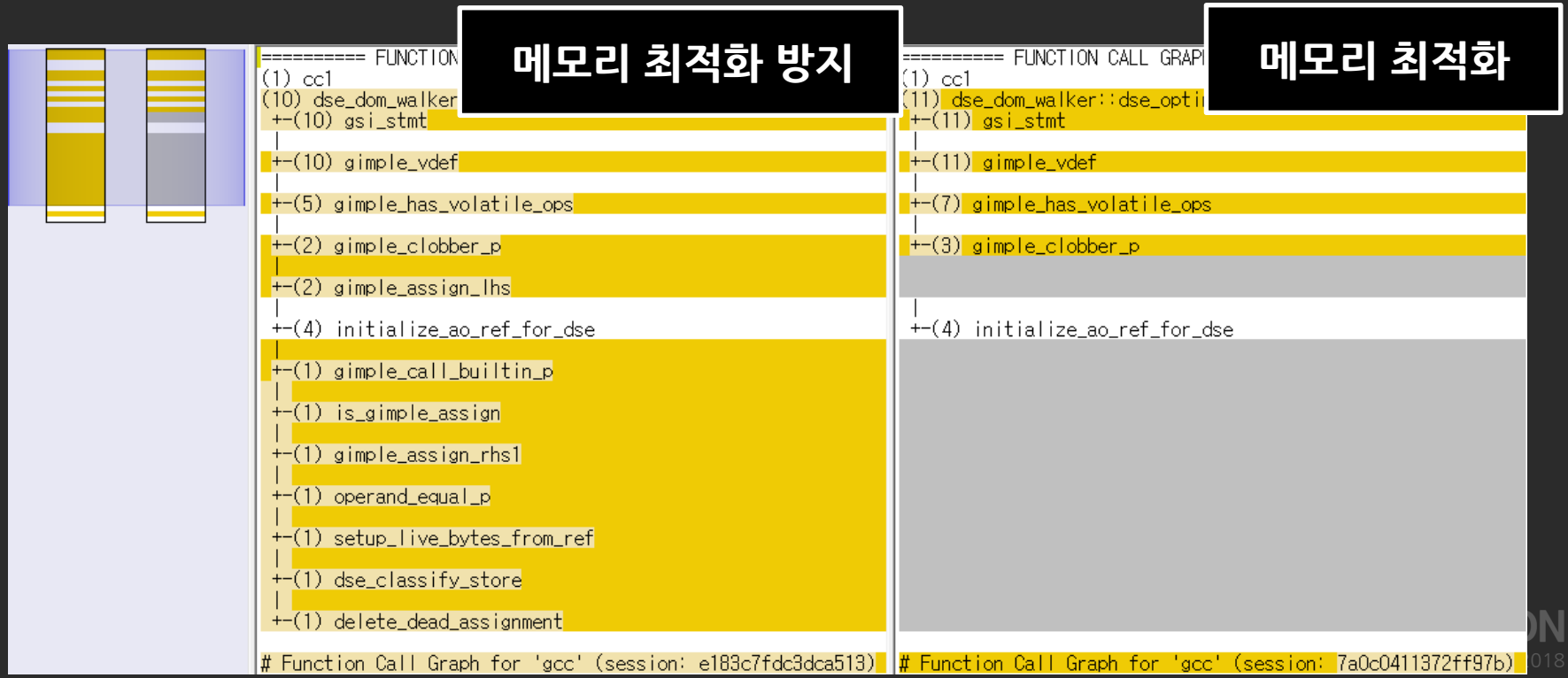
```
;; Function get_msg (get_msg, funcdef_no=0)
Deleted dead store: str = "Hello world₩n";
g
{
  s
  return 0B;

}

;; Function main (main, funcdef_no=1)
main ()
{
  char * p;

  <bb 2> [0.00%]:
  p_3 = get_msg ();
  puts (p_3);
  return 0;
}
```

inline.c.040t.dse1
Deleted dead store: str = "Hello world₩n";

메모리 최적화

Tracing For Fun

$ uftrace graph -D 2 -F dse_dom_walker::dse_optimize_stmt -f none



Tracing For Fun

```
10227        case ADDR_EXPR:
10228          inner = TREE_OPERAND (inner, 0);
10229
10230          while (REFERENCE_CLASS_P (inner)
10231                 && !INDIRECT_REF_P (inner))
10232            inner = TREE_OPERAND (inner, 0);
10233
10234          if (DECL_P (inner)
10235              && !DECL_EXTERNAL (inner)
10236              && !TREE_STATIC (inner)
10237              && DECL_CONTEXT (inner) == current_function_decl)
10238            warning_at (loc, OPT_Wreturn_local_addr,
10239                        "function returns address of local variable");
10240          else if (DECL_DECLARED_INLINE_P(current_function_decl))
10241            inner->base.volatile_flag = true;
10242          else
10243            {
10244              warning_at (loc, OPT_Wreturn_local_addr,
10245                          "function returns address of local variable");
10246              tree zero = build_zero_cst (TREE_TYPE (res));
10247              t = build2 (COMPOUND_EXPR, TREE_TYPE (res), t, zero);
10248            }
10249          }
10250        break;
10251
10252
```

```
else if (DECL_DECLARED_INLINE_P(current_function_decl))
  inner->base.volatile_flag = true;
```

return 될 tree가 속한 함수가
inline 으로 선언되어 있으면
0으로 바꾸지 마세요!

+ volatile_flag true set

Tracing For Fun

실종됐던 Hello world를 되찾았습니다

```
$ cat inline.c
static inline __attribute__((always_inline))
char *get_msg()
{
    char str[64] = "Hello world₩n";
    return (char *)str;
}

int main()
{
    char *p;
    p = get_msg();
    puts(p);
    return 0;
}

$ gcc -O2 inline.c
$ ./a.out
Hello world
```

-O1

```
Dump of assembler code for function main:
<+4>:    movabs $0x6f77206f6c6c6548,%rax
<+14>:   mov    $0xa646c72,%edx
<+19>:   mov    %rax,(%rsp)
<+23>:   mov    %rdx,0x8(%rsp)
<+28>:   movq   $0x0,0x10(%rsp)
<+37>:   movq   $0x0,0x18(%rsp)
<+46>:   movq   $0x0,0x20(%rsp)
<+55>:   movq   $0x0,0x28(%rsp)
<+64>:   movq   $0x0,0x30(%rsp)
<+73>:   movq   $0x0,0x38(%rsp)
<+82>:   mov    %rsp,%rdi
<+85>:   callq  0x400400 <puts@plt>
```

-O2

```
Dump of assembler code for function main:
<+4>:    movdqa 0x1b4(%rip),%xmm0
<+12>:   mov    %rsp,%rdi
<+15>:   movaps %xmm0,(%rsp)
<+19>:   pxor   %xmm0,%xmm0
<+23>:   movaps %xmm0,0x10(%rsp)
<+28>:   movaps %xmm0,0x20(%rsp)
<+33>:   movaps %xmm0,0x30(%rsp)
<+38>:   callq  0x400400 <puts@plt>
```

Retrospect

회고

Retrospect

문외한도 아는 척 할 수 있는 핵심은 - 가독성!

```
;; Function get_msg (get_msg, funcdef_no=0, decl_uid=1792, cgraph_uid=0, symbol_order=0)
Deleted dead store: str = "Hello world₩n";
__attribute__((always_inline)) get_msg ()
{
  char str[64];
  <bb 2> [0.00%]:
  str ={v} {CLOBBER};
  return &str;
}

;; Function main (main, funcdef_no=1, decl_uid=1796, cgraph_uid=1, symbol_order=1)
main ()
{
  char str[64];
  char * D.1810;
  char * p;

  <bb 2> [100.00%]:
  str ={v} {CLOBBER};
  puts (&str);
  return 0;
}
```

Retrospect

문외한도 아는 척 할 수 있는 핵심은 - 가독성!

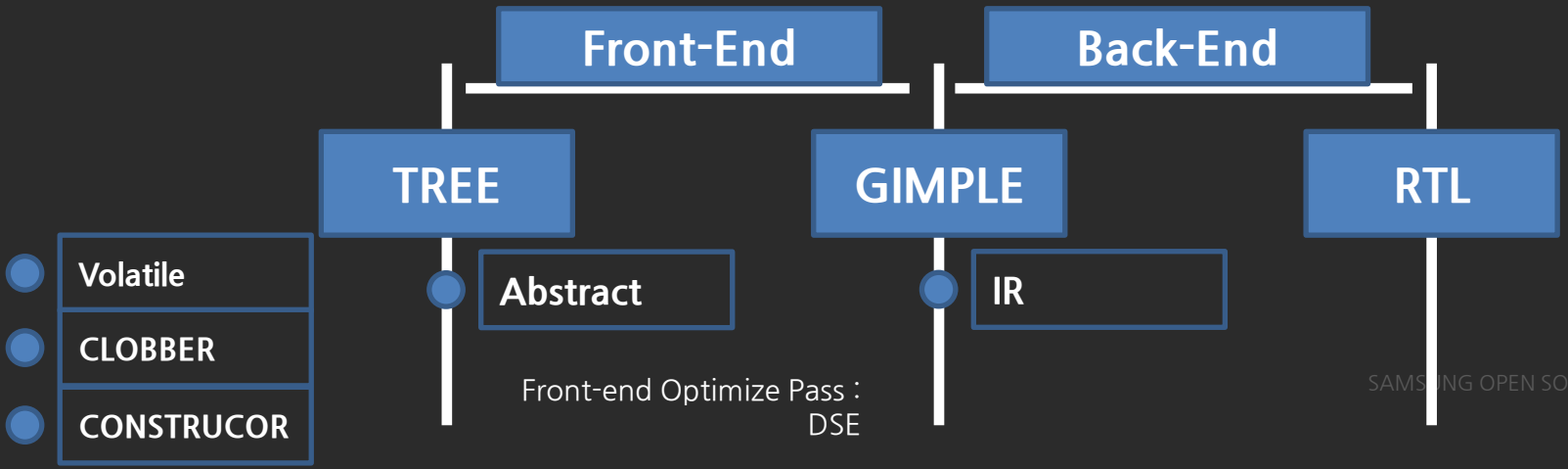
```
;; Function get_msg (get_msg, funcdef_no=0, decl_uid=1792, cgraph_uid=0, symbol_order=0)
Deleted dead store: str = "Hello world₩n";
__attribute__((always_inline)) get_msg ()
{
  char str[64];
  <bb 2> [0.00%]:
  str ={v} {CLOBBER};
  return;
}

;; Functi
main ()
{
  char str[64];
  char * D.1810;
  char * p;

  <bb 2> [100.00%]:
  str ={v} {CLOBBER};
  puts (&str);
  return 0;
}
```

```
(gdb) p *gsi->ptr
$4 = {code = GIMPLE_ASSIGN, no_warning = 0, visited = 0, nontemporal_move = 0, plf
= 1, modified = 0, has_volatile_ops = 0, pad = 0, subcode = 32, uid = 0,
  location = 2147483655, num_ops = 2, bb = 0x7ffff6ede478, next = 0x7ffff702d190,
prev = 0x7ffff702d140}
```

Retrospect

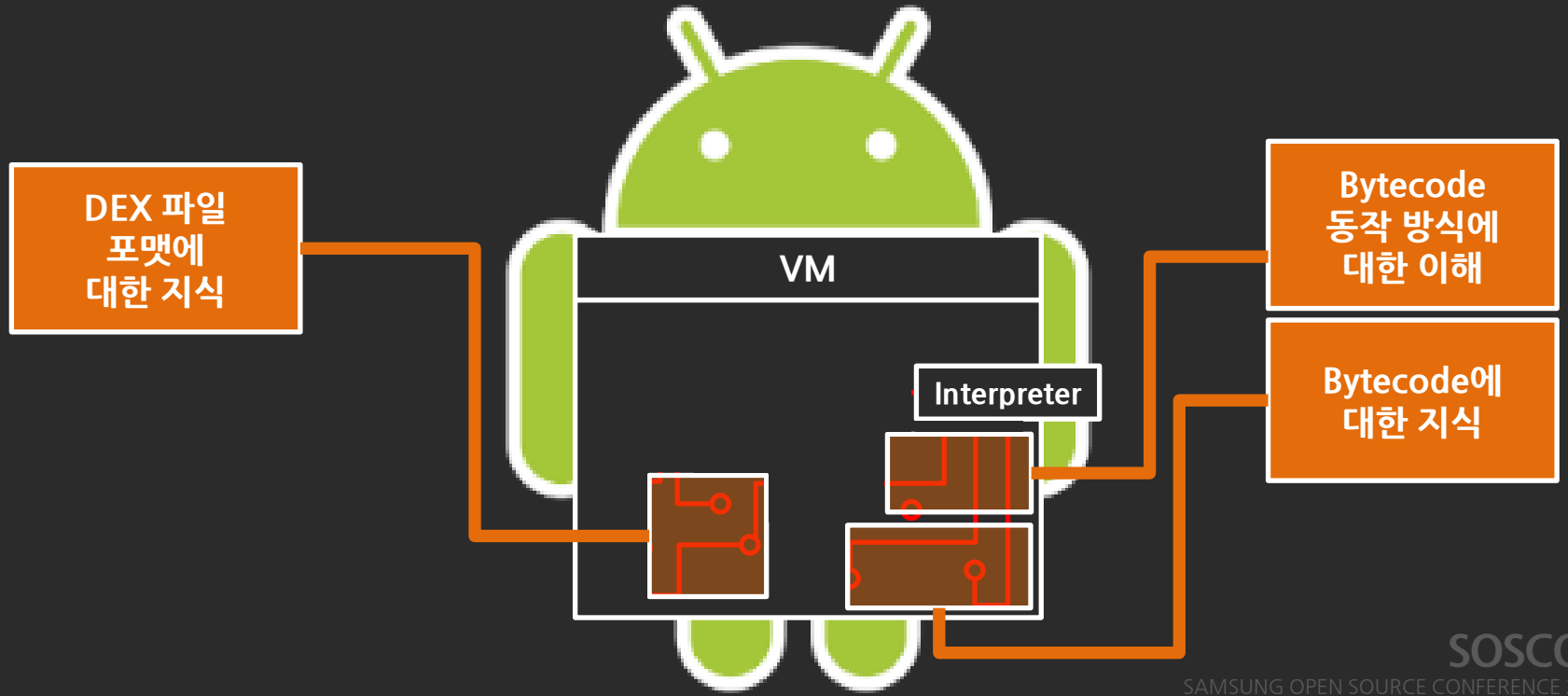
Gcc를 수정하는 과정에서 관련 지식도 조금 습득했습니다

```
else if (DECL_DECLARED_INLINE_P(current_function_decl))
    inner->base.volatile_flag = true;
```



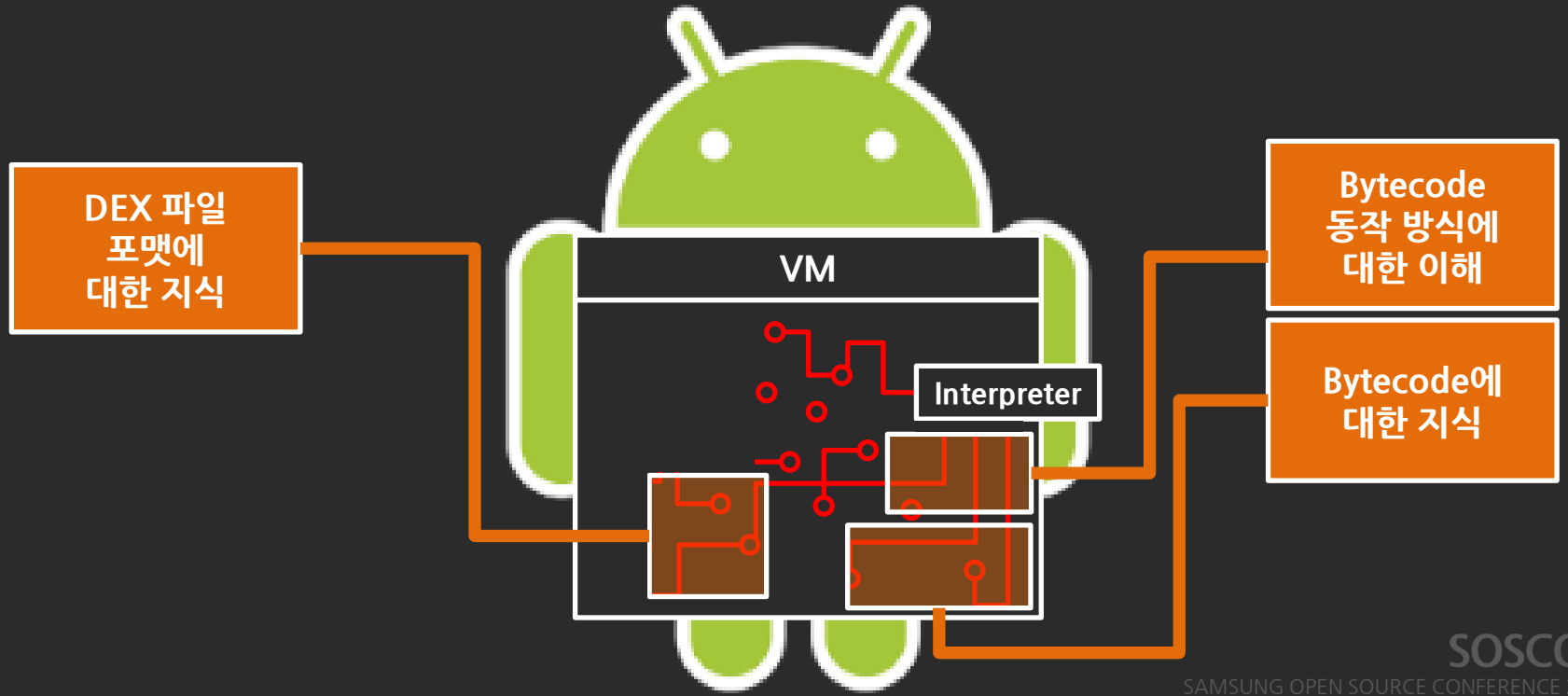
Retrospect

Android VM에 대한 부분 부분의 지식들이



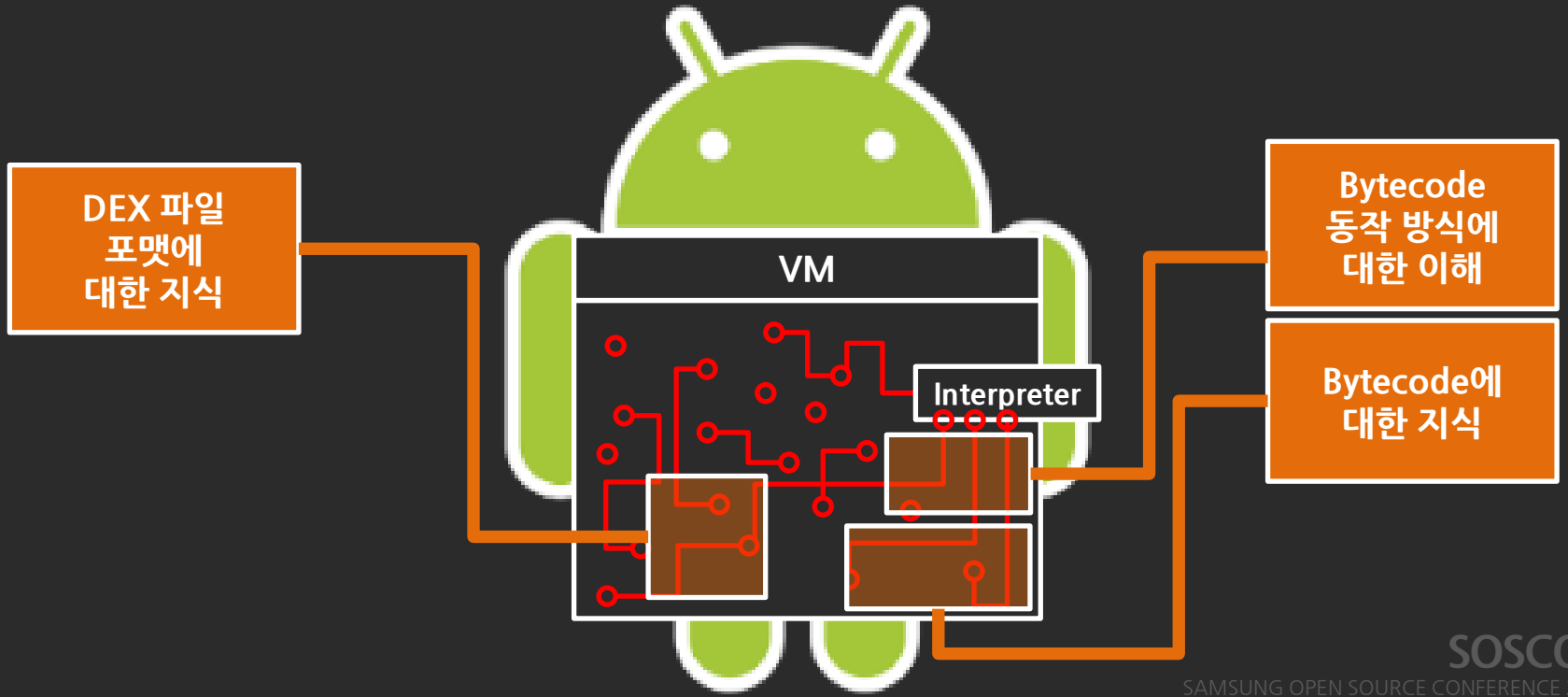
Retrospect

Tracing의 과정을 거치면서 연결되게 되면



Retrospect

답답하기만 했던 코드가 이해되는 시원함을 느끼게 됩니다

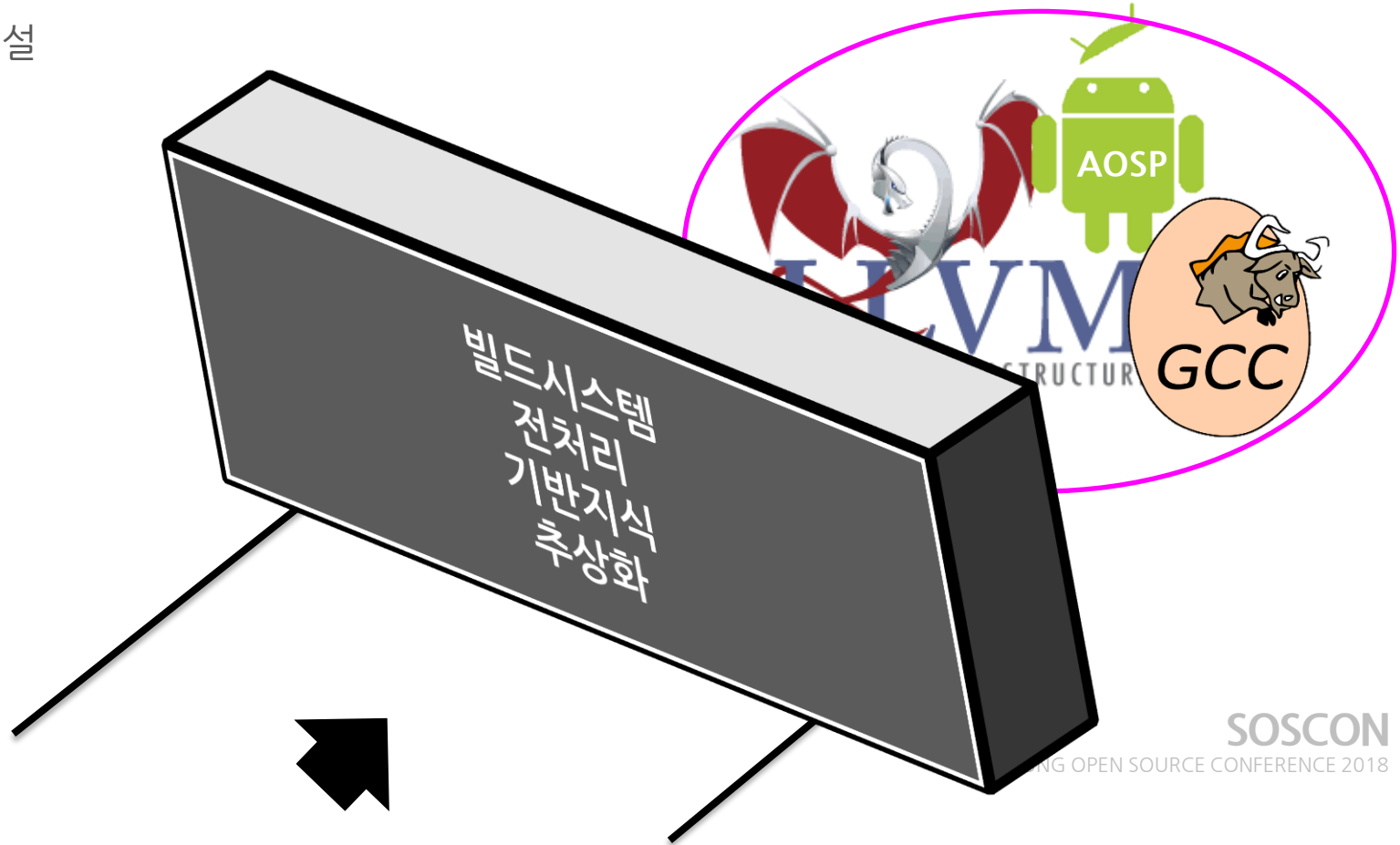


Conclusion

오픈소스의 접근성을 올려주는 요소

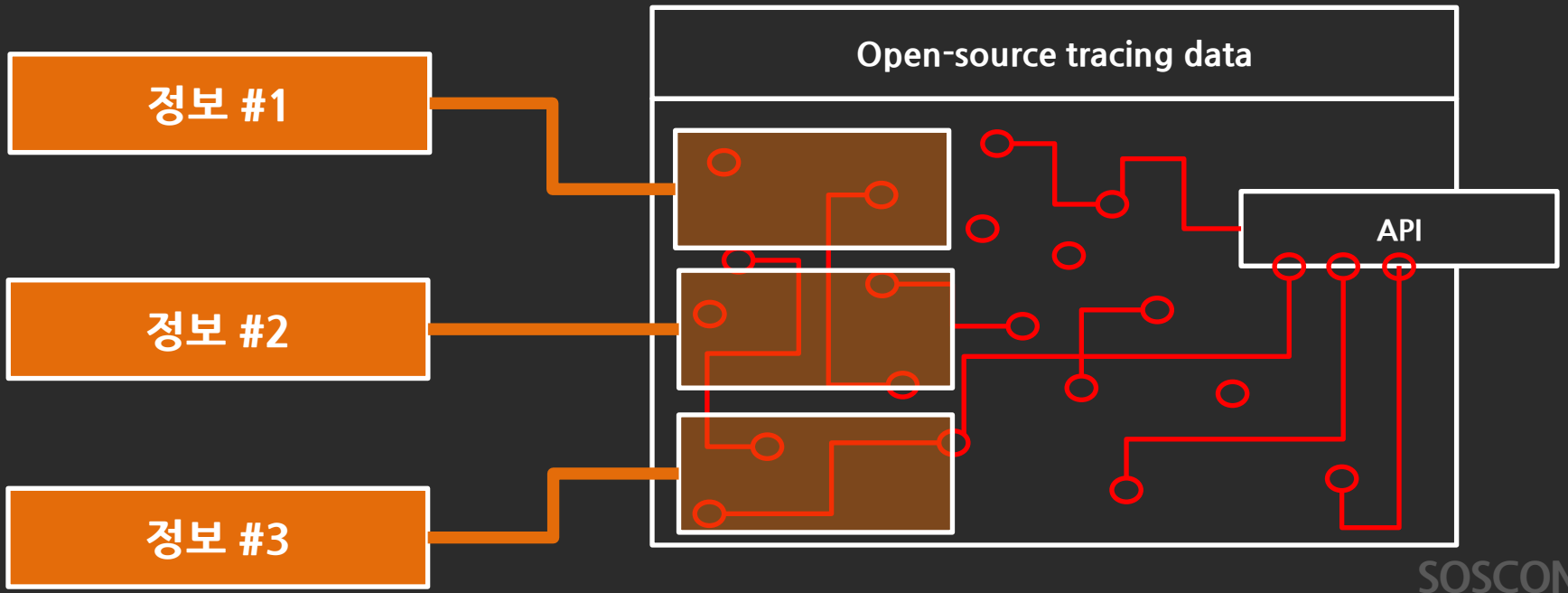
Conclusion

오픈소스의 역설



Conclusion

오픈소스의 접근성 올리기



Conclusion

오픈소스의 접근성 올리기

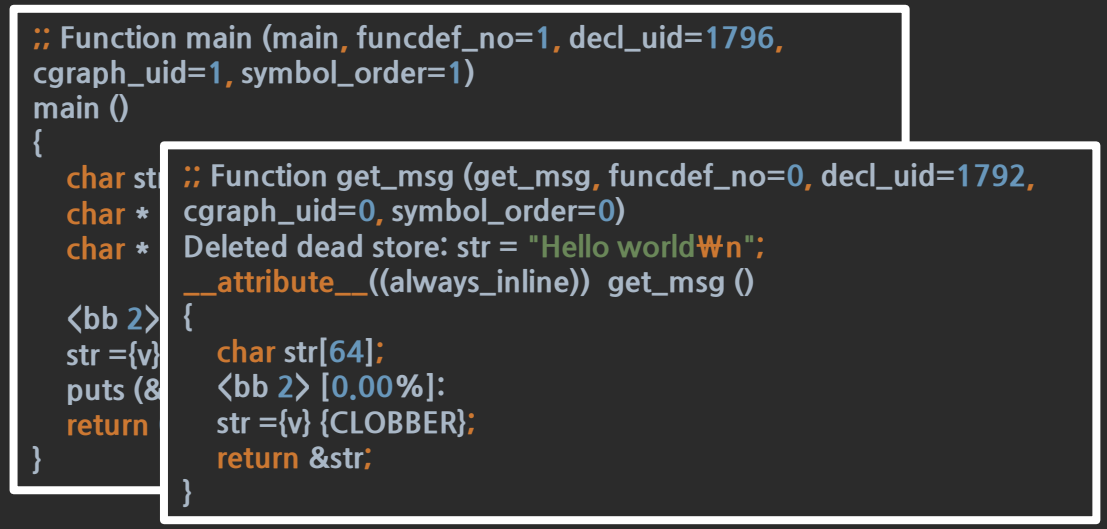
```
$ uftrace record -S clarity_gcc_opt_path.py `which gcc` -O1 inline.c
```

```
Global Declared Classes
Optimization step The actual step to run dse.
The optimization is actually performed when this function is called

A dead store is a store into a memory location which will later be
overwritten by another store without any intervening loads.  In this
case the earlier store can be deleted.
================================================================
                 [ 32314] |          _GLOBAL__N_1::pass_dse::pass_dse() {
  29.798 us [ 32314] |          } /* _GLOBAL__N_1::pass_dse::pass_dse */
```

Conclusion

오픈소스의 접근성 올리기

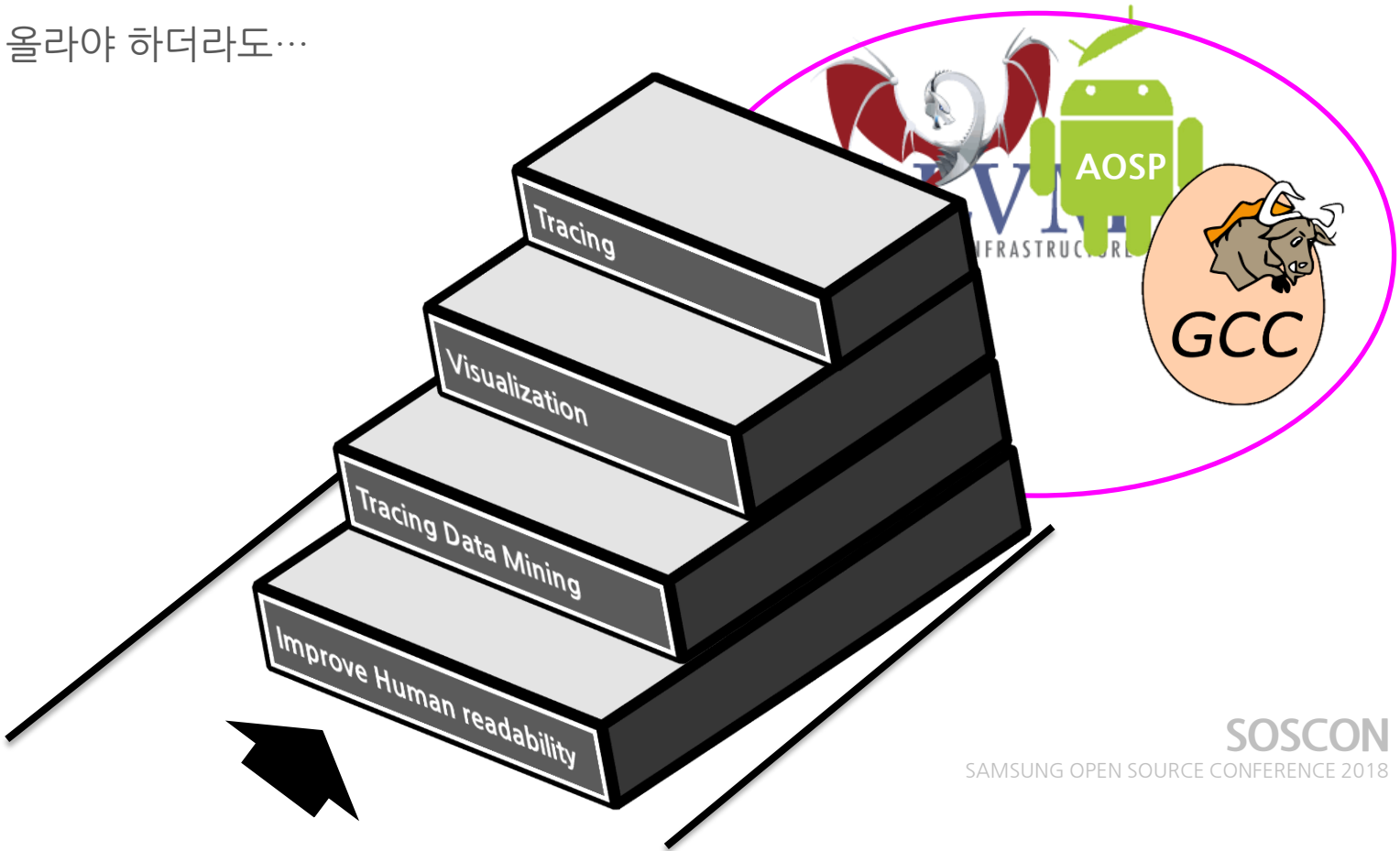
[32314] | _GLOBAL__N_1::pass_dse::pass_dse() {



[32314] | } /* _GLOBAL__N_1::pass_dse::pass_dse */

Conclusion

똑같은 장벽을 올라야 하더라도…



THANK YOU

Uftrace repository
https://github.com/namhyung/uftrace

Any Question?

Uftrace repository
https://github.com/namhyung/uftrace

Appendix A. benchmark

Target Source

```
#include <stdio.h>
#include <time.h>

void callme(int *r)
{
    (*r)++;
}

int main()
{
    clock_t begin = clock();

    for(int i=0;i<0xFFF;)
    {
        callme(&i);
    }

    clock_t end = clock();

    printf("Elapsed: %lf seconds₩n", (double)(end - begin) / CLOCKS_PER_SEC);
    return 0;
}
```

LLDB python script
https://github.com/ParkHanbum/lldb_tracer