[ART-7ff5373] LSE of Host ART Removes an NonEnvironmentUse Incorrectly

Assigned Bug P3 [AOSP] assigned

STATUS UPDATE No update yet. Edit

DESCRIPTION an...@gmail.com created issue #1

```
import dalvik.artemis.Artemis;

// NPE
class BugOOOP3 {
    int field = 0;

    static int foo() {
        BugOOOP3 m = new BugOOOP3();
        Artemis.ensureDeoptimized();
        return m.field;                        // Line 10
    }

    public static void main(String[] args) throws Throwable {
        Artemis.ensureMethodJitCompiled(BugOOOP3.class, "foo");
        foo();
    }
}
```

The above code makes *host* ART throw an *unexpected* NullPointerException at Line 10 as:

```
Exception in thread "main" java.lang.NullPointerException: Attempt to read from field 'int BugOOOP3.field' on a null object reference in method 'int BugOOOP3
        at BugOOOP3.foo(BugOOOP3.java:10)
        at BugOOOP3.main(BugOOOP3.java:15)
```

The exception is not expected to be thrown though.

Compilation and Analysis

I'm sorry the provided code DOES NOT compile since we modified ART to add our own code (i.e., `dalvik.artemis.Artemis`) to force a method to be JIT compiled or deoptimized. Though our ov

We have debugged this bug for a while, it seems that LSE (LoadStoreElimination) directly removes an HNewInstance instruction for variable `m` (at Line 8) because the LoadStoreAnalysis tells `m` h

We reckon this to be a bug of LSE because

1. when `m`'s HNewInstance instruction is added to the analysis (load_store_elimination.cc:1248), it DOES hold that NonEnvironmentUse
2. unfortunately, after the analysis, the NonEnvironmentUse is removed and the HNewInstance is also removed (load_store_elimination.cc#3941)

Our Environment

Arch: x86_64

ART Commit: a 12-day-ago commit:

```
$ art -showversion
ART version 2.1.0 x86_64
```

d8:

```
$ d8 --version
D8 3.3.20-dev+aosp1 (build 1fc252939072b72d3948ac2e8b6e5c8a0d319f7d from go/r8bot (luci-r8-custom-ci-xenial-27-5ysn))
```

Java compiler: Java 11

```
$ javac -version
javac 11.0.15
```

What happened

ART throws NPE

What you think the correct behavior should be

ART doesn't throw

Links (11)

"…he provided code DOES NOT compile since we modified ART to add our own code (i.e., dalvik.artemis.Artemis) to force a method to be JIT compiled or deoptimized. Though our own logic, we utilize
"… compile since we modified ART to add our own code (i.e., dalvik.artemis.Artemis) to force a method to be JIT compiled or deoptimized. Though our own logic, we utilized existing interfaces like art
" …mpile since we modified ART to add our own code (i.e., dalvik.artemis.Artemis) to force a method to be JIT compiled or deoptimized. Though our own logic, we utilized existing interfaces like artDe
"… since we modified ART to add our own code (i.e., dalvik.artemis.Artemis) to force a method to be JIT compiled or deoptimized. Though our own logic, we utilized existing interfaces like artDeoptim
"when m's HNewInstance instruction is added to the analysis (load_store_elimination.cc:1248), it DOES hold that NonEnvironmentUse"

See all related links

COMMENTS

an...@gmail.com <an...@gmail.com> #2

FYI, `Artemis.ensureDeoptimized()` and `Artemis.ensureMethodJitCompiled()` are two native methods.

an...@gmail.com <an...@gmail.com> #3

If you very much slightly change the test case by adding `for (int i = 0; i < Integer.MAX_VALUE; i++) {}` right before Line 10 like following

```
import dalvik.artemis.Artemis;

// NPE
class BugOOOP3 {
    int field = 0;

    static int foo() {
        BugOOOP3 m = new BugOOOP3();
        Artemis.ensureDeoptimized();
        for (int i = 0; i < Integer.MAX_VALUE; i++) {}     // Add here
        return m.field;
    }

    public static void main(String[] args) throws Throwable {
        Artemis.ensureMethodJitCompiled(BugOOOP3.class, "foo");
        foo();
    }
}
```

the OSR (`art_quick_osr_stub()`) crashes with the following log:

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
Fatal signal 11 (SIGSEGV), code 1 (SEGV_MAPERR) fault addr 0xebadde11
OS: Linux 5.4.0-100-generic (x86_64)
Cmdline: <unset>
Thread: 766670 "main"
Registers:
    rax: 0x00000000ebadde09    rbx: 0x00007ffd9bad5ef8    rcx: 0x0000000000000000    rdx: 0x0000000043ca6658
    rdi: 0x00007ffd9bad5cd8    rsi: 0x000055ee1ac345a8    rbp: 0x00007ffd9bad5ce0    rsp: 0x00007ffd9bad5c70
    r8 : 0x00007f38a6d49899    r9 : 0x000055ee1ac191c0    r10: 0x0000000000000118    r11: 0x000000000000538a
    r12: 0x000055ee1ac19270    r13: 0x000055ee1ac191c0    r14: 0x000055ee1ac34530    r15: 0x00007f38a6d4b040
    rip: 0x0000000043ca6608    eflags: 0x00010246 [ PF ZF IF ]
     cs: 0x00000033     gs: 0x00000000     fs: 0x00000000

Backtrace:
        #00 pc 0000000002000608  /memfd:jit-cache (deleted) (offset 2000000) (BugOOOP3.foo+200)
/usr/bin/addr2line: '/memfd:jit-cache (deleted)': No such file
        #01 pc 0000000000a7b66b  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art_quick_osr_stub+27)
/usr/bin/addr2line: DWARF error: invalid or unhandled FORM value: 0x25
          art_quick_osr_stub
           crtstuff.c:?
        #02 pc 0000000000681585  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art::jit::Jit::MaybeDoOnStackReplacement(art::Thread*, art::Art
          art::jit::Jit::MaybeDoOnStackReplacement(art::Thread*, art::ArtMethod*, unsigned int, int, art::JValue*)
           ??:?
        #03 pc 00000000005f1407  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (void art::interpreter::ExecuteSwitchImplCpp<false, false>(art::
          void art::interpreter::ExecuteSwitchImplCpp<false, false>(art::interpreter::SwitchImplContext*)
           ??:?
        #04 pc 0000000000a7b965  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (ExecuteSwitchImplAsm+5)
          ExecuteSwitchImplAsm
           crtstuff.c:?
        #05 pc 0000000000001498  <anonymous:7f38a6d45000> (BugOOOP3.foo)
        #06 pc 00000000005d9e49  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccess
          art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFrame&, art::JValue, bool, bool)
           interpreter.cc:?
        #07 pc 00000000005e1092  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art::interpreter::EnterInterpreterFromDeoptimize(art::Thread*,
          art::interpreter::EnterInterpreterFromDeoptimize(art::Thread*, art::ShadowFrame*, art::JValue*, bool, art::DeoptimizationMethodType)
           ??:?
```

```
        #08 pc 0000000000a36fe6  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art::HandleDeoptimization(art::JValue*, art::ArtMethod*, art::S
          art::HandleDeoptimization(art::JValue*, art::ArtMethod*, art::ShadowFrame*, art::ManagedStack*)
           quick_trampoline_entrypoints.cc:?
        #09 pc 0000000000a36a48  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (artQuickToInterpreterBridge+1352)
          artQuickToInterpreterBridge
           ??:?
        #10 pc 0000000000a798ec  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art_quick_to_interpreter_bridge+140)
          art_quick_to_interpreter_bridge
           crtstuff.c:?
        #11 pc 0000000000a55315  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (nterp_helper+165)
          NterpCommonInvokeStatic
           nterp.cc:?
        #12 pc 0000000000001504  <anonymous:7f38a6d45000> (BugOOOP3.main+20)
        #13 pc 0000000000a60536  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art_quick_invoke_static_stub+806)
          art_quick_invoke_static_stub
           crtstuff.c:?
        #14 pc 000000000043faec  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art::ArtMethod::Invoke(art::Thread*, unsigned int*, unsigned in
          art::ArtMethod::Invoke(art::Thread*, unsigned int*, unsigned int, art::JValue*, char const*)
           ??:?
        #15 pc 00000000008f04e0  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art::JValue art::InvokeWithVarArgs<art::ArtMethod*>(art::Scoped
          art::JValue art::InvokeWithVarArgs<art::ArtMethod*>(art::ScopedObjectAccessAlreadyRunnable const&, _jobject*, art::ArtMethod*, __va_list_tag*)
           ??:?
        #16 pc 0000000000724afb  /zdata/congli/my-aosp/out/host/linux-x86/lib64/libart.so (art::JNI<false>::CallStaticVoidMethodV(_JNIEnv*, _jclass*, _jme
          art::JNI<false>::CallStaticVoidMethodV(_JNIEnv*, _jclass*, _jmethodID*, __va_list_tag*)
           ??:?
        #17 pc 0000000000003bbb  /zdata/congli/my-aosp/out/host/linux-x86/bin/dalvikvm64 (_JNIEnv::CallStaticVoidMethod(_jclass*, _jmethodID*, ...)+139)
/usr/bin/addr2line: DWARF error: invalid or unhandled FORM value: 0x25
          _JNIEnv::CallStaticVoidMethod(_jclass*, _jmethodID*, ...)
           ??:?
        #18 pc 000000000000380d  /zdata/congli/my-aosp/out/host/linux-x86/bin/dalvikvm64 (art::dalvikvm(int, char**)+2077)
          art::dalvikvm(int, char**)
           dalvikvm.cc:?
        #19 pc 0000000000002fe8  /zdata/congli/my-aosp/out/host/linux-x86/bin/dalvikvm64 (main+8)
          main
           ??:?
        #20 pc 0000000000024082  /usr/lib/x86_64-linux-gnu/libc-2.31.so (__libc_start_main+242) (BuildId: 1878e6b475720c7c51969e69ab2d276fae6d1dee)
/usr/bin/addr2line: DWARF error: section .debug_info is larger than its filesize! (0x93ef57 vs 0x530ea0)
          __libc_start_main
           ??:?
        #21 pc 0000000000002ec8  /zdata/congli/my-aosp/out/host/linux-x86/bin/dalvikvm64 (???)

Fault message:
exiting due to SIG_DFL handler for signal 11, ucontext 0x55ee1ac22780
art F art/runtime/runtime_common.cc] HandleUnexpectedSignal reentered

art F art/runtime/runtime_common.cc]  S06
```

Don't know for sure whether they are the same bug, but they seem similar.

---

vi...@google.com <vi...@google.com> #4

Assigned to vi...@google.com.

Thank you for reporting this issue. For us to further investigate this issue, please provide the following additional information:

Android build
Which Android build are you using? (e.g. PPP5.180610.010)

Device used -- Device Make, Model, Android OS Version
Which device did you use to reproduce this issue?

Android bug report (to be captured after reproducing the issue)
For steps to capture a bug report, please refer: https://developer.android.com/studio/debug/bug-report#bugreportdevice

Screen record of the issue, if possible
Please capture screen record or video of the issue using following steps:
adb shell screenrecord /sdcard/video.mp4
Subsequently use following command to pull the recorded file:
adb pull /sdcard/video.mp4
Attach the file to this issue.

Capture the issue in a screenshot, if possible
Press the volume down and power buttons simultaneously. The image will appear in the picture gallery. Attach the screenshot image to this issue.

Note: Please avoid uploading directly to the issue using attachments. Please upload to google drive and share the folder to android-bugreport@google.com, then share the link here.

---

an...@gmail.com <an...@gmail.com> #5

Hi ART's developers, sorry that we are using Host-ART and does not need any Android devices. And so we cannot capture any bugreport/screenshot.

vi...@google.com <vi...@google.com> #6

We've shared this with our product and engineering teams and will continue to provide updates as more information becomes available.

ng...@google.com <ng...@google.com>

Reassigned to ng...@google.com.

ng...@google.com <ng...@google.com> #7

Verified by ng...@google.com.

Thanks for the report! It looks like we've fixed the issue as I could not reproduce it on a similar test case: https://android-review.googlesource.com/c/platform/art/+/2101709

Also, note that you should not use artDeoptimizeFromCompiledCode from C++ code. It's intended to only be used by the compiler, so the compiler knows that it need to keep an environment

an...@gmail.com <an...@gmail.com> #8

Thanks #7. Pretty helpful. We will re-confirm it in future ART commits.

On the other hand, we have kept the environment for the deoptimization in the SSA Liveness Analysis and Code Generator. See this commit.

Did you find any other possible flaws in our code? Or are there any other conditions that (you find we do not consider but) would affect deoptimization?

Thanks!

ng...@google.com <ng...@google.com> #9

#8: Yes, we had a bug in the compiler on LSE and deopts, which got fixed here: https://android-review.googlesource.com/c/platform/art/+/2072271.

an...@gmail.com <an...@gmail.com> #10

Dear comment #9, thanks! But we can still reproduce this bug on 24710b3c (the latest commit by 22/05/2022) using our provided test (rather than yours mentioned in comment #7).

Can you help to revisit this issue?

FYI. We have

1. kept the environment of those `HInvokeStaticOrDirect` instructions who invokes `Artemis.ensureDeoptimized()` in `SsaLivenessAnalysei:: ShouldBeLiveForEnvironment()`; an
2. written their environment (vregs) to `CodeInfo` by returning true in `NeedsVregInfo()`.

Please check the following code for the above description or this commit:

```
// ssa_liveness_analysis.h
static bool ShouldBeLiveForEnvironment(HInstruction* env_holder, HInstruction* instruction) {
  // ...
  return instruction->GetType() == DataType::Type::kReference ||
         // When invoking Artemis' specifc ensureXxx() methods, should keep the
         // environment since we need them to force self JIT/DEOPT (i.e., OSR).
         // Always put this line of code as the last condition to decrease the
         // performance degradation induced by Artemis.
         env_holder->IsArtemisEnsureJitCompiledOrDeoptimizedStaticInvoke();
}

// code_generator.cc
static bool NeedsVregInfo(HInstruction* instruction, bool osr) {
  HGraph* graph = instruction->GetBlock()->GetGraph();
  return ... ||
         instruction->IsArtemisEnsureJitCompiledOrDeoptimizedStaticInvoke();
}


// nodes.h
inline bool HInstruction::IsArtemisEnsureJitCompiledOrDeoptimizedStaticInvoke() {
  if (!IsInvokeStaticOrDirect()) {
    return false;
  }
  ArtMethod* method = AsInvokeStaticOrDirect()->GetResolvedMethod();
  // Either Artemis#ensureJitCompiled() or Artemis#ensureJitCompiled()
  return artemis::IsArtemisEnsureJitCompiled(method) ||
         artemis::IsArtemisEnsureDeoptimized(method);
}
```

ng...@google.com <ng...@google.com> #11

Status: Assigned (reopened)

Could you craft a test case that doesn't require artemis calls? There's lots of handling of explicit deopts in the compiler, and the fact it's hidden through your artemis method is probably maki