How to Reduce an LLVM Bug

Matt Arsenault

AMD
together we advance_

Demo Plugin

https://github.com/arsenm/llvm-reduce-tutorial

LLVM built with `-DLLVM_BUILD_LLVM_DYLIB=ON -DLLVM_LINK_LLVM_DYLIB=ON`

```
$ mkdir build; cd build
$ cmake llvm-reduce-tutorial –G Ninja –DCMAKE_PREFIX_PATH=/path/to/llvm-project/build
–DLLVM_PROJECT_SRC=/path/to/llvm-project
$ ninja
```

Demo Plugin Usage

```
$ opt --load-pass-plugin=/path/to/llvm-reduce-tutorial/build/buggy_plugin.so \
 –passes=‘buggy< crash-on-i1-select;crash-on-aggregate-phis>’ example.ll
```

```
$ BUGGY_PLUGIN_OPTS=“crash-on-i1-select;crash-on-aggregate-phis” clang++ \
       –fpass-plugin=/path/to/llvm-reduce-tutorial/build/buggy_plugin.so \
       example.cpp
```

X86ISelLowering Example

```
$ touch llvm/lib/Target/X86/X86ISelLowering.cpp
$ ninja –v –j1 llc
$ export BUGGY_PLUGIN_OPTS=crash-load-of-inttoptr
$ # Add -fpass-plugin=/path/to/build.so argument to clang invocation
```

Generated Reproducer Script

```
clang++: error: clang frontend command failed with exit code 70 (use -v to see invocation)
clang version 21.0.0git (https://github.com/llvm/llvm-project.git faefb70c7a771ae646df3d5defe122cfff2aac7c)
Target: x86_64-unknown-linux-gnu
Thread model: posix
InstalledDir: /home/marsenau/src/llvm-project/build/bin
Build config: +assertions
clang++: note: diagnostic msg:
********************


PLEASE ATTACH THE FOLLOWING FILES TO THE BUG REPORT:
Preprocessed source(s) and associated run script(s) are located at:
clang++: note: diagnostic msg: /tmp/X86ISelLowering-5a21d8.cpp
clang++: note: diagnostic msg: /tmp/X86ISelLowering-5a21d8.sh
clang++: note: diagnostic msg:


********************
```

Where did the crash occur?

- Step 1 is getting your failure into a self-contained command
- Best results if you can get it down to a single pass invocation
- Usually works out alright for IR passes to just take the failing pass, extract the IR before it, and run
- Repeat the process as you refine your testcase
- Some failures (i.e., register allocation) are very sensitive to any perturbations

```
fatal error: error in backend: load of inttoptr is broken
PLEASE submit a bug report to https://github.com/llvm/llvm-project/issues/ and include the crash backtrace, preprocessed source, and associated run script.
Stack dump:
0. Program arguments: /home/marsenau/src/llvm-project/build/bin/clang++ -fpass-plugin=/home/marsenau/src/llvm-reduce-tutorial/build/buggy_plugin.so -DGTEST_HAS_RTTI=0 -
   DLLVM_EXPORTS … -o lib/Target/X86/CMakeFiles/LLVMX86CodeGen.dir/X86ISelLowering.cpp.o -c /home/marsenau/src/llvm-project/llvm/lib/Target/X86/X86ISelLowering.cpp
1. <eof> parser at end of file
2. Optimizer
3. Running pass "function<eager-inv>(float2int,lower-constant-intrinsics,buggy<crash-load-of-inttoptr;>,loop(loop-rotate<header-duplication;no-prepare-for-lto>,loop-
   deletion),loop-distribute,inject-tli-mappings,loop-vectorize<no-interleave-forced-only;no-vectorize-forced-only;>,infer-alignment,loop-load-elim,instcombine<max-
   iterations=1;no-verify-fixpoint>,simplifycfg<bonus-inst-threshold=1;forward-switch-cond;switch-range-to-icmp;switch-to-lookup;no-keep-loops;hoist-common-insts;no-hoist-
   loads-stores-with-cond-faulting;sink-common-insts;speculate-blocks;simplify-cond-branch;no-speculate-unpredictables>,slp-vectorizer,vector-combine,instcombine<max-
   iterations=1;no-verify-fixpoint>,loop-unroll<O2>,transform-warning,sroa<preserve-cfg>,infer-alignment,instcombine<max-iterations=1;no-verify-fixp`oint>,loop-
   mssa(licm<allowspeculation>),alignment-from-assumptions,loop-sink,instsimplify,div-rem-pairs,tailcallelim,simplifycfg<bonus-inst-threshold=1;no-forward-switch-cond;switch-
   range-to-icmp;no-switch-to-lookup;keep-loops;no-hoist-common-insts;hoist-loads-stores-with-cond-faulting;no-sink-common-insts;speculate-blocks;simplify-cond-
   branch;speculate-unpredictables>)" on module "/home/marsenau/src/llvm-project/llvm/lib/Target/X86/X86ISelLowering.cpp"
4. Running pass "buggy<crash-load-of-inttoptr;>" on function "_ZL19FindSingleBitChangePN4llvm5ValueE"
#0 0x00007096a6a87928 llvm::sys::PrintStackTrace(llvm::raw_ostream&, int) /home/marsenau/src/llvm-project/llvm/lib/Support/Unix/Signals.inc:804:13
#1 0x00007096a6a857f0 llvm::sys::RunSignalHandlers() /home/marsenau/src/llvm-project/llvm/lib/Support/Signals.cpp:106:18
#2 0x00007096a69befb7 (anonymous namespace)::CrashRecoveryContextImpl::HandleCrash(int, unsigned long) /home/marsenau/src/llvm-
   project/llvm/lib/Support/CrashRecoveryContext.cpp:73:5
#3 0x00007096a69bef4f llvm::CrashRecoveryContext::HandleExit(int) /home/marsenau/src/llvm-project/llvm/lib/Support/CrashRecoveryContext.cpp:446:3
#4 0x00007096a6a826d7 llvm::sys::Process::Exit(int, bool) /home/marsenau/src/llvm-project/llvm/lib/Support/Process.cpp:117:5
#5 0x00005be361edba56 (/home/marsenau/src/llvm-project/build/bin/clang+++0x16a56)
#6 0x00007096a69d2689 std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::_M_data() const /usr/lib/gcc/x86_64-linux-
   gnu/14/../../../../include/c++/14/bits/basic_string.h:228:28
#7 0x00007096a69d2689 std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::_M_is_local() const /usr/lib/gcc/x86_64-linux-
   gnu/14/../../../../include/c++/14/bits/basic_string.h:269:6
#8 0x00007096a69d2689 std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::_M_dispose() /usr/lib/gcc/x86_64-linux-
   gnu/14/../../../../include/c++/14/bits/basic_string.h:287:7
#9 0x00007096a69d2689 std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string() /usr/lib/gcc/x86_64-linux-
   gnu/14/../../../../include/c++/14/bits/basic_string.h:809:9
#10 0x00007096a69d2689 llvm::report_fatal_error(llvm::Twine const&, bool) /home/marsenau/src/llvm-project/llvm/lib/Support/ErrorHandling.cpp:105:5
#11 0x00007096a69d2576 (/home/marsenau/src/llvm-project/build/bin/../lib/libLLVM.so.21.0git+0x4dd2576)
#12 0x000070969f1ce9fb (anonymous namespace)::BuggyPass::run(llvm::Function&, llvm::AnalysisManager<llvm::Function>&) buggy_plugin.cpp:0:0
#13 0x000070969f1d208f llvm::detail::PassModel<llvm::Function, (anonymous namespace)::BuggyPass, llvm::AnalysisManager<llvm::Function>>::run(llvm::Function&,
   llvm::AnalysisManager<llvm::Function>&) buggy_plugin.cpp:0:0
#14 0x00007096a6c6a9c7 llvm::PassManager<llvm::Function, llvm::AnalysisManager<llvm::Function>>::run(llvm::Function&, llvm::AnalysisManager<llvm::Function>&)
   /home/marsenau/src/llvm-project/llvm/include/llvm/IR/PassManagerImpl.h:85:8
#15 0x00007096a9531f6d llvm::detail::PassModel<llvm::Function, llvm::PassManager<llvm::Function, llvm::AnalysisManager<llvm::Function>>,
   llvm::AnalysisManager<llvm::Function>>::run(llvm::Function&, llvm::AnalysisManager<llvm::Function>&) /home/marsenau/src/llvm-
   project/llvm/include/llvm/IR/PassManagerInternal.h:91:5
#16 0x00007096a6c6f181 llvm::ModuleToFunctionPassAdaptor::run(llvm::Module&, llvm::AnalysisManager<llvm::Module>&) /home/marsenau/src/llvm-
   project/llvm/lib/IR/PassManager.cpp:129:23
#
```

Two Debugging Paths

- Go direct to the IR at the point of failure
- Happy path which works most of the time
- Run the singular failing pass with `opt`

- Reproduce the invocation as closely as possible from the start
- Global options set by frontend
- Most sensitive bugs, and usually encountering infrastructure issues

Obtain Original clang Output

- -emit-llvm -Xclang -disable-llvm-passes
- clang -Xclang -disable-llvm-passes != clang –O0
- clang –O0 produces different IR
- clang –O0 -disable-llvm-passes != clang –O0

Reproduce with opt

- `$ buggy_opt –O2 -disable-output X86ISelLowering.cpp.clang-disable-llvm-passes.bc`

Extract the IR before the failure

Brute force approach

```
$ buggy_clang++ -mllvm –print-before-all 2> debug_log.txt
```

```
; *** IR Dump Before SROAPass on _ZN4llvm2cl3optIiLb0ENS0_6parserIiEEEC1IJA47_cNS0_11initializerIiEENS0_4descENS0_12OptionHiddenEEEEDpRKT_ ***
; Function Attrs: mustprogress nounwind ssp uwtable(sync)
define linkonce_odr hidden noundef ptr @_ZN4llvm2cl3optIiLb0ENS0_6parserIiEEEC1IJA47_cNS0_11initializerIiEENS0_4descENS0_12OptionHiddenEEEEDpRKT_(ptr noundef nonnull returned align 8 dereferenceable(192) %this, ptr noundef nonnull align 1 dereferenceable(47) %Ms, ptr noundef nonnull align 8 dereferenceable(8) %Ms1, ptr noundef nonnull align 8 dereferenceable(16) %Ms3, ptr noundef nonnull align 4 dereferenceable(4) %Ms5) unnamed_addr #2 !dbg !212411 {
entry:
  %this.addr = alloca ptr, align 8
  %Ms.addr = alloca ptr, align 8
  %Ms.addr2 = alloca ptr, align 8
  %Ms.addr4 = alloca ptr, align 8
  %Ms.addr6 = alloca ptr, align 8
…
```

Printing All the IRs

- `$ buggy_opt –O2 –print-before-all -print-module-scope 2> debug_log.txt`

- `$ buggy_opt –O2 –print-on-crash -print-module-scope 2> debug_log.txt`

Refined Printed Context

```
-print-before=buggy

  Running pass "buggy<crash-load-of-inttoptr;>" on function …


-print-after=other-pass-name

  Running pass "function<eager-inv>(float2int,lower-constant-intrinsics,buggy<crash-load-of-
  inttoptr;>,loop(loop-rotate<header-duplication;no-prepare-for-lto>,loop-deletion),loop-
  distribute


-print-after=lower-constant-intrinsics


-print-module-scope -filter-print-funcs=_ZNK4llvm17X86TargetLowering23LowerINTRINSIC_...
```

Refined Printed Context

- -print-before-pass-number is the best way to get precise output before the failure point
- Use –print-pass-numbers to get the pass number at the failure point

```
Running pass 466737 SimplifyCFGPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466738 SLPVectorizerPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466739 VectorCombinePass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466740 InstCombinePass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466741 LoopUnrollPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466742 WarnMissedTransformationsPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466743 SROAPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466744 InferAlignmentPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466745 InstCombinePass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466746 LoopSimplifyPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466747 LCSSAPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466748 AlignmentFromAssumptionsPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466749 LoopSinkPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466750 InstSimplifyPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466751 DivRemPairsPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466752 TailCallElimPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466753 SimplifyCFGPass on _ZNK4llvm17X86TargetLowering30shouldExpandLogicAtomicRMWInIREPNS_13AtomicRMWInstE
Running pass 466754 Float2IntPass on _ZN4llvm11Instruction9user_backEv
Running pass 466755 LowerConstantIntrinsicsPass on _ZN4llvm11Instruction9user_backEv
Running pass 466756 buggy on _ZN4llvm11Instruction9user_backEv
Running pass 466757 LoopSimplifyPass on _ZN4llvm11Instruction9user_backEv
Running pass 466758 LCSSAPass on _ZN4llvm11Instruction9user_backEv
Running pass 466759 LoopDistributePass on _ZN4llvm11Instruction9user_backEv
Running pass 466760 InjectTLIMappings on _ZN4llvm11Instruction9user_backEv
Running pass 466761 LoopVectorizePass on _ZN4llvm11Instruction9user_backEv
Running pass 466762 InferAlignmentPass on _ZN4llvm11Instruction9user_backEv
Running pass 466763 LoopLoadEliminationPass on _ZN4llvm11Instruction9user_backEv
Running pass 466764 InstCombinePass on _ZN4llvm11Instruction9user_backEv
Running pass 466765 SimplifyCFGPass on _ZN4llvm11Instruction9user_backEv
Running pass 466766 SLPVectorizerPass on _ZN4llvm11Instruction9user_backEv
Running pass 466767 VectorCombinePass on _ZN4llvm11Instruction9user_backEv
Running pass 466768 InstCombinePass on _ZN4llvm11Instruction9user_backEv
Running pass 466769 LoopUnrollPass on _ZN4llvm11Instruction9user_backEv
Running pass 466770 WarnMissedTransformationsPass on _ZN4llvm11Instruction9user_backEv
Running pass 466771 SROAPass on _ZN4llvm11Instruction9user_backEv
Running pass 466772 InferAlignmentPass on _ZN4llvm11Instruction9user_backEv
Running pass 466773 InstCombinePass on _ZN4llvm11Instruction9user_backEv
Running pass 466774 LoopSimplifyPass on _ZN4llvm11Instruction9user_backEv
Running pass 466775 LCSSAPass on _ZN4llvm11Instruction9user_backEv
Running pass 466776 AlignmentFromAssumptionsPass on _ZN4llvm11Instruction9user_backEv
Running pass 466777 LoopSinkPass on _ZN4llvm11Instruction9user_backEv
Running pass 466778 InstSimplifyPass on _ZN4llvm11Instruction9user_backEv
Running pass 466779 DivRemPairsPass on _ZN4llvm11Instruction9user_backEv
Running pass 466780 TailCallElimPass on _ZN4llvm11Instruction9user_backEv
Running pass 466781 SimplifyCFGPass on _ZN4llvm11Instruction9user_backEv
Running pass 466782 Float2IntPass on _ZL19FindSingleBitChangePN4llvm5ValueE
Running pass 466783 LowerConstantIntrinsicsPass on _ZL19FindSingleBitChangePN4llvm5ValueE
Running pass 466784 buggy on _ZL19FindSingleBitChangePN4llvm5ValueE
LLVM ERROR: load of inttoptr is broken
PLEASE submit a bug report to https://github.com/llvm/llvm-project/issues/ and include the crash backtrace.
Stack dump:
0.      Program arguments: /home/marsenau/src/llvm-project/build_reldebinfo/bin/opt --load-pass-plugin=/home/marsenau/src/llvm-reduce-tutorial/build/buggy_plugin.so -disable-output -O2 X86ISelLowering.cpp.clang-disable-llvm-passes.bc
 -print-pass-numbers
1.      Running pass "function<eager-inv>(float2int,lower-constant-intrinsics,buggy<crash-load-of-inttoptr;>,loop(loop-rotate<header-duplication;no-prepare-for-lto>,loop-deletion),loop-distribute,inject-tli-mappings,loop-vectorize<no
-interleave-forced-only;no-vectorize-forced-only;>,infer-alignment,loop-load-elim,instcombine<max-iterations=1;no-verify-fixpoint>,simplifycfg<bonus-inst-threshold=1;forward-switch-cond;switch-range-to-icmp;switch-to-lookup;no-keep-l
oops;hoist-common-insts;no-hoist-loads-stores-with-cond-faulting;sink-common-insts;speculate-blocks;simplify-cond-branch;no-speculate-unpredictables>,slp-vectorizer,vector-combine,instcombine<max-iterations=1;no-verify-fixpoint>,loop
-unroll<O2>,transform-warning,sroa<preserve-cfg>,infer-alignment,instcombine<max-iterations=1;no-verify-fixpoint>,loop-mssa(licm<allowspeculation>),alignment-from-assumptions,loop-sink,instsimplify,div-rem-pairs,tailcallelim,simplify
cfg<bonus-inst-threshold=1;no-forward-switch-cond;switch-range-to-icmp;no-switch-to-lookup;keep-loops;no-hoist-common-insts;hoist-loads-stores-with-cond-faulting;no-sink-common-insts;speculate-blocks;simplify-cond-branch;speculate-un
predictables>)" on module "X86ISelLowering.cpp.clang-disable-llvm-passes.bc"
2.      Running pass "buggy<crash-load-of-inttoptr;>" on function "_ZL19FindSingleBitChangePN4llvm5ValueE"
```

Running pass 466784 buggy on _ZL19FindSingleBitChangePN4llvm5ValueE
LLVM ERROR: load of inttoptr is broken

-print-before-pass-number

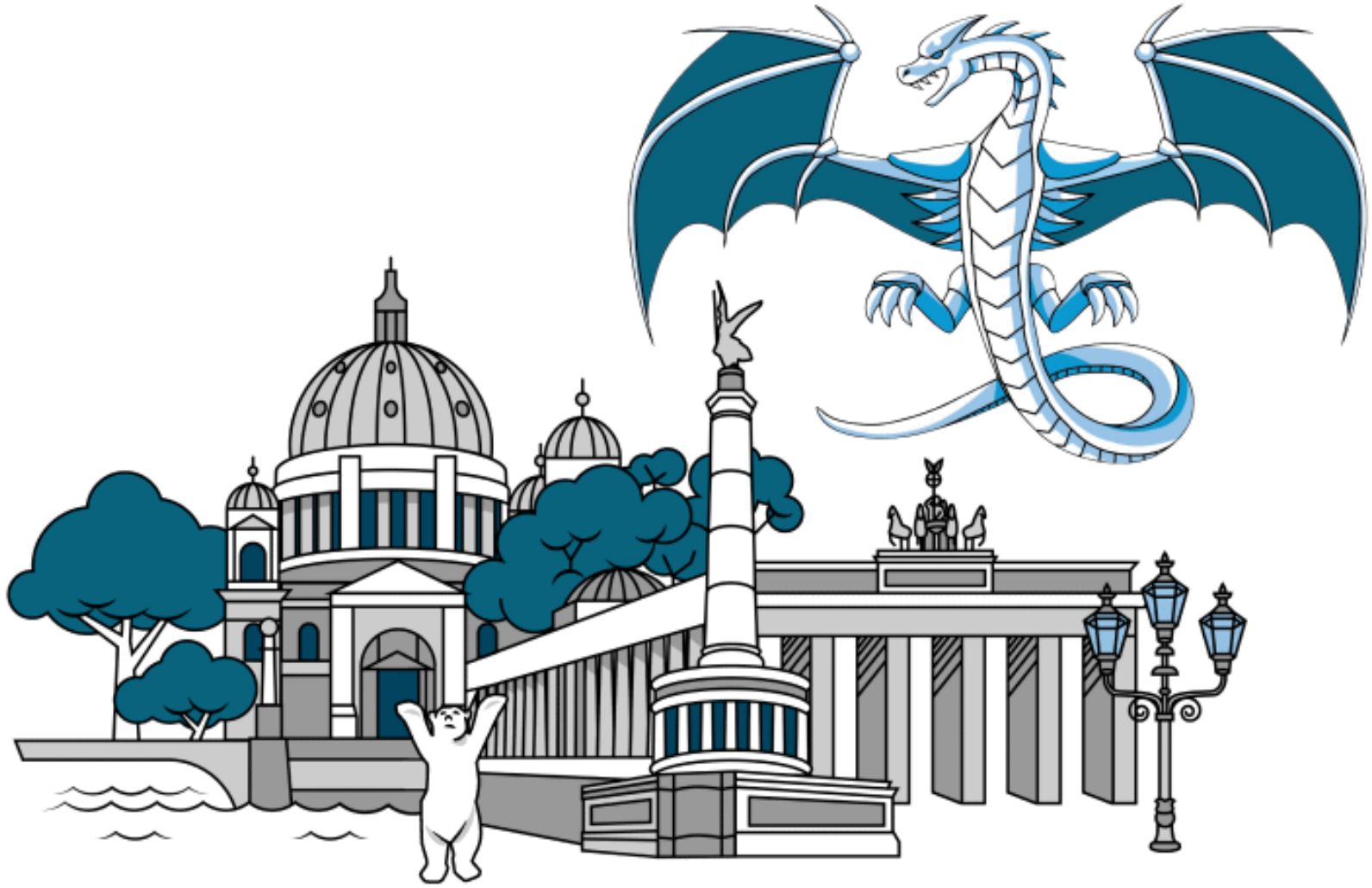
```
opt -disable-output -O2 -print-before-pass-number=466784  \
  X86ISelLowering.cpp.clang-disable-llvm-passes.bc 2> before-466784.ll
```

- Raw stderr output includes the backtrace and errors at the end
- You can manually trim it off to get a parsable input

-ir-dump-directory

```
$ opt -disable-output -O2 -print-before-pass-number=466784 \
  –print-module-scope \
  -ir-dump-directory=/tmp/my_debug_output \
  X86ISelLowering.cpp.clang-disable-llvm-passes.bc
$ ls /tmp/my_debug_output
466784-153b4460d50ce649-function-a71200593fa6b29a-buggy-before.ll
```

- New since 2023
- Works with all of the –print-* flags
- No more backtrace to trim out, directly usable files
- Did not work correctly with `–print-before-pass-number` when I made the slides
- https://github.com/llvm/llvm-project/pull/130983 fixes it



llvm-reduce

- Now you are ready to handle the reduction
- Reduces any binary bitcode or text IR input
- Create an “interestingness” script with the IR based reproduction command

Interestingness Script

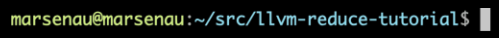
- An exit value of 0 is considered interesting
- This is backwards from what you would hope for a crash, but it matches FileCheck / grep
- Don’t forget shebang line
- Don’t forget to chmod +x
- Write to be multiprocess safe

```
#!/usr/bin/env sh


! /path/to/opt -disable-output --load-pass-plugin=/path/to/buggy_plugin.so \
  -passes=‘buggy<crash-load-of-inttoptr>’ $@ 2> /dev/null
```

llvm-reduce

```
$ llvm-reduce –test=interestingness.sh before-466784.ll
```



```
define fastcc i1
@"_ZZNK4llvm17X86TargetLowering17LowerBUILD_VECTORENS_7SDValueERNS_12Selection
DAGEENK3$_0clES1_jNS_8ArrayRefIS1_EE"() {
entry:
  %0 = inttoptr i64 0 to ptr
  ret i1 false

_ZNK4llvm8ArrayRefINS_7SDValueEEixEm.exit.peel: ; No predecessors!
  %1 = load ptr, ptr %0, align 8
  br label %for.body

for.body: ; preds = %for.body, %_ZNK4llvm8ArrayRefINS_7SDValueEEixEm.exit.peel
  br label %for.body
}
```

-O2 vs. Single Pass Reproducer

```
define i32
@_ZNK4llvm17X86TargetLowering17getConstraintTypeENS_9StringRefE(i64
%Constraint.coerce.fca.0.extract, ptr %Constraint11) {
entry:
  store i64 %Constraint.coerce.fca.0.extract, ptr %Constraint11, align 8
  %call2 = call i8 @_ZNK4llvm9StringRefixEm(ptr %Constraint11)
  switch i8 %call2, label %return [
    i8 65, label %sw.bb3
    i8 68, label %sw.bb3
    i8 83, label %sw.bb3
    i8 100, label %sw.bb3
    i8 1, label %sw.bb3
    i8 98, label %sw.bb3
    i8 0, label %sw.bb3
  ]
sw.bb3:
  br label %return
return:
  %retval.0 = phi i32 [ 0, %sw.bb3 ], [ 1, %entry ]
  ret i32 %retval.0
}

define i8 @_ZNK4llvm9StringRefixEm(ptr %this) {
  %call2 = call ptr @_ZNK4llvm9StringRef4dataEv(ptr %this)
  %0 = load i8, ptr %call2, align 1
  ret i8 %0
}

define ptr @_ZNK4llvm9StringRef4dataEv(ptr %this) {
  %0 = load ptr, ptr %this, align 8
  ret ptr %0
}
```

```
define fastcc i1
@"_ZZNK4llvm17X86TargetLowering17LowerBUILD_VECTORENS
_7SDValueERNS_12SelectionDAGEENK3$_0clES1_jNS_8ArrayR
efIS1_EE"() {
entry:
  %0 = inttoptr i64 0 to ptr
  ret i1 false

; No predecessors!
_ZNK4llvm8ArrayRefINS_7SDValueEEixEm.exit.peel:
  %1 = load ptr, ptr %0, align 8
  br label %for.body

for.body:
  br label %for.body
}
```

Adjusted Interestingness Script

```
! opt -disable-output --load-pass-plugin=buggy_plugin.so \
-passes='instsimplify,simplifycfg,buggy<crash-load-of-inttoptr>' $@ \
2> /dev/null
```

```
define fastcc i1
@"_ZZNK4llvm17X86TargetLowering17LowerBUILD_VECTORENS_7SDValueERNS_12SelectionDAGEENK3$_0
clES1_jNS_8ArrayRefIS1_EE"(i64 %Ops.coerce.fca.0.extract) {
entry:
  %0 = inttoptr i64 %Ops.coerce.fca.0.extract to ptr
  %1 = load ptr, ptr %0, align 8
  %cmp.i.i.peel = icmp eq ptr %1, null
  ret i1 %cmp.i.i.peel
}
```

Debugging Hangs

```
#!/usr/bin/env sh

! timeout 10 opt -disable-output -load-pass-plugin=buggy_plugin.so \
-passes='simplifycfg,loop-combine,buggy-indirect-call>' $@ 2> /dev/null

      define ptr @_ZN4llvm2cl3optIiLb0ENS0_6parserIiEEED1Ev(ptr %this) {
      entry:
        br label %_ZN4llvm9StringRefC2EPKc.exit
        tail call void %0(ptr null)
      cond.true.i: ; No predecessors!
      } %call.i.i.i = tail call i64 null(ptr null)
        br label %_ZN4llvm9StringRefC2EPKc.exit

      _ZN4llvm9StringRefC2EPKc.exit: ; preds = %cond.true.i, %entry
        ret ptr null
      }
```


AMD
together we advance_

Multiple Bugs

```
#!/usr/bin/env sh
! opt -disable-output --load-pass-plugin=buggy_plugin.so \
  -passes='instsimplify,simplifycfg,buggy<crash-on-i1-select;crash-on-repeated-phi-
  predecessor;bug-only-if-internal-func>’ $@
```

```
$ interestingness-multi-crash.sh before-466784.ll
```

LLVM ERROR: i1 typed select is broken

PLEASE submit a bug report to https://github.com/llvm/llvm-project/issues/ and include the crash backtrace.

Stack dump:

Multiple Bugs

```
$ llvm-reduce -o reduced.ll --test=interestingness-multi-crash.sh before-466784.ll

$ interestingness-multi-crash.sh reduced.ll

LLVM ERROR: phi with repeated predecessor is broken
PLEASE submit a bug report to https://github.com/llvm/llvm-project/issues/ and include
the crash backtrace.
```

Filtering Error Messages

```
#!/usr/bin/env sh
! opt -disable-output --load-pass-plugin=buggy_plugin.so \
  -passes='instsimplify,simplifycfg,buggy<crash-on-i1-select;crash-on-repeated-phi-predecessor;bug-only-if-internal-func>' $@ \
  2>&1 | grep -q "i1 typed select is broken"
```

```
#!/usr/bin/env sh
! opt -disable-output --load-pass-plugin=buggy_plugin.so \
  -passes='instsimplify,simplifycfg,buggy<crash-on-i1-select;crash-on-repeated-phi-predecessor;bug-only-if-internal-func>' $@ \
  2>&1 | FileCheck --allow-empty --dump-input=never $0

# CHECK: i1 typed select is broken
define internal fastcc [2 x i64]
  @_ZL19LowerPoHorizontalOpPKN4llvm17BuildVectorSDNodeERKNS_5SDLocERKNS_12X86SubtargetERNS_
  12SelectionDAGE(i1 %cond, i1 %cmp.i91) {
entry:
  %or.cond311 = select i1 %cond, i1 %cmp.i91, i1 false
  br i1 %or.cond311, label %if.then39, label %common.ret

common.ret: ; preds = %if.then39, %entry
  ret [2 x i64] zeroinitializer

if.then39: ; preds = %entry
  call void @llvm.lifetime.start.p0(i64 0, ptr null)
  br label %common.ret
}
```

Controlling specific reduction techniques

- Sometimes you need additional controls to reduce successfully
- Some reduction strategies may be counterproductive to your bug
- Work around `llvm-reduce` bugs
- llvm-reduce `--print-delta-passes` shows the set of reduction passes
- Not useful to run `llvm-reduce` repeatedly
  - `--max-pass-iterations=<int>`

Controlling specific reductions

- `--delta-passes=functions,instructions`
- `--skip-delta-passes=reduction-name` for opt-out
  - `--skip-delta-passes=attributes`
  - `--skip-delta-passes=ir-passes`

- `llvm-reduce --delta-passes=ir-passes --ir-passes='function(sroa)’`

llvm-reduce options

- `-j=1`
- `--write-tmp-files-as-bitcode`
- `--skip-verify-interesting-after-counting-chunks`
- `--abort-on-invalid-reduction`
- `--preserve-debug-environment`

Debugging Compiler Invocation

```
$ clang “-###” -fuse-ld=lld -O2 a.c b.c
clang –cc1 "-emit-obj” /* many arguments */ a.c –o a-f0d8ae.o
clang –cc1 "-emit-obj” /* many arguments */ b.c –o b-a12565.o
ld.lld /* many arguments */ -o a.out a-f0d8ae.o b-a12565.o –l …

$ clang “-###” -flto -fuse-ld=lld -O2 a.c b.c
clang –cc1 "-emit-llvm-bc” /* many arguments */ a.c –o a-f0d8ae.o
clang –cc1 "-emit-llvm-bc” /* many arguments */ b.c –o b-a12565.o
ld.lld /* many arguments */ -o a.out a-f0d8ae.o b-a12565.o –l …
```

Debugging Compiler Invocation

```
$ clang –save-temps -flto -fuse-ld=lld -O2 a.c b.c
$ ls
a.bc a.i a.o a.out b.bc b.i b.o
```

```
$ clang -fuse-ld=lld -O2 a.c b.c –mllvm –print-before=prologepilog

$ clang –flto -fuse-ld=lld -O2 a.c b.c –Wl,-mllvm -Wl,print-before=prologepilog

$ clang –flto -fuse-ld=lld -O2 a.c b.c –Wl,-plugin-opt=save-temps
$ ls
a.bc  a.o    a.out.0.0.preopt.bc      a.out.0.4.opt.bc         a.out.lto.o
b.bc  b.o
a.i   a.out  a.out.0.2.internalize.bc  a.out.0.5.precodegen.bc  a.out.resolution.txt
b.i
```

Debugging Compiler Invocation

```
$ clang -### -flto=full -fuse-ld=lld -O2 -save-temps=cwd inputs/a.c
```

```
clang version 21.0.0git (git@github.com:llvm/llvm-project.git 1a95215e5019edc858f234c95477ab0a6b164d5)
Target: x86_64-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/local/bin
Build config: +assertions
"clang" "-cc1" "-E" "-dumpdir" "a-" "-save-temps=cwd" "-o" "a.i" "-x" "c" "inputs/a.c"
"clang" "-cc1" "-emit-llvm-bc" "-emit-llvm-uselists" "-flto=full" "-flto-unit" "-dumpdir" "a-" "-save-temps=cwd" "-O2" "-disable-llvm-passes" "-o" "a.bc" "-x" "cpp-output" "a.i"
"ld.lld" "--hash-style=gnu" "--eh-frame-hdr" "-m" "elf_x86_64" "-pie" "-dynamic-linker" "/lib64/ld-linux-x86-64.so.2" "-o" "a.out" "Scrt1.o" "crti.o" "crtbeginS.o" "-L/usr/lib" "-plugin-opt=mcpu=x86-64" "-plugin-opt=O2" "-plugin" "a.bc"
```

clang version 21.0.0git (git@github.com:llvm/llvm-project.git 1a95215e5019edc858f234c95477ab0a6b164d5)
Target: x86_64-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/local/bin
Build config: +assertions
"clang" "-cc1" "-E" "-dumpdir" "a-" "-save-temps=cwd" "-o" "a.i" "-x" "c" "inputs/a.c"
"clang" "-cc1" "-emit-llvm-bc" "-emit-llvm-uselists" "-flto=full" "-flto-unit" "-dumpdir" "a-" "-save-temps=cwd" "-O2" "-disable-llvm-passes" "-o" "a.bc" "-x" "cpp-output" "a.i"
"ld.lld" "--hash-style=gnu" "--eh-frame-hdr" "-m" "elf_x86_64" "-pie" "-dynamic-linker" "/lib64/ld-linux-x86-64.so.2" "-o" "a.out" "Scrt1.o" "crti.o" "crtbeginS.o" "-L/usr/lib" "-plugin-opt=mcpu=x86-64" "-plugin-opt=O2" "-plugin" "a.bc"

```
"clang" "-cc1" "-emit-llvm-bc" "-flto=full" "-flto-unit" "-dumpdir" "a-" "-save-temps=cwd" "-o" "a.i" "-x" "c" "inputs/a.c"$tmpdir/a-b21c52.o" "-x" "c"
"clang" "-cc1" "-emit-llvm-bc" "-emit-llvm-uselists" "-flto=full" "-flto-unit" "-dumpdir" "a-" "-save-temps=cwd" "-O2" "-disable-llvm-passes" "-o" "a.bc" "-x" "cpp-output" "a.i"
"ld.lld" "--hash-style=gnu" "--eh-frame-hdr" "-m" "elf_x86_64" "-pie" "-dynamic-linker" "/lib64/ld-linux-x86-64.so.2" "-o" "a.out" "Scrt1.o" "crti.o" "crtbeginS.o" "-L/usr/lib" "-plugin-opt=mcpu=x86-64" "-plugin-opt=O2" "a.o" "-lgcc" "--as-needed" "-lgcc_s" "--no-as-needed" "-lc" "-lgcc" "--as-needed" "-lgcc_s" "--no-as-needed" "crtendS.o" "crtn.o"
```

Debugging Offload Invocations

```
$ clang++ ”-fopenmp -offload-arch=gfx942 -O3 test.cpp
```

```
"clang" "-cc1" "-triple" "x86_64-pc-linux-gnu" "-emit-llvm-bc" "-fopenmp" "-disable-llvm-passes" \
          "-fopenmp-targets=amdgcn-amd-amdhsa" "-x" "c++" "test.cpp" -o ".../test-f66534.bc"
"clang" "-cc1" "-triple" "amdgcn-amd-amdhsa" "-aux-triple" "x86_64-pc-linux-gnu" "-emit-llvm-bc" "-mlink-builtin-bitcode" "..." ”\
          -target-cpu" "gfx942" "-include" "__clang_openmp_device_functions.h" "-O3" "-o" "...test-gfx942-f0a302.bc" "-x" "c++” \
          "test.cpp"
"clang-offload-packager" "-o" "test-5145b4.out" "--image=file=test-gfx942-f0a302.bc,triple=amdgcn-amd-amdhsa,arch=gfx942,kind=openmp"
"clang" "-cc1" "-triple" "x86_64-pc-linux-gnu" "-emit-obj" "-target-cpu" "x86-64" "-fopenmp" "-fgnuc-version=4.2.1" \
          "-fembed-offload-object=test-5145b4.out" "-fopenmp-targets=amdgcn-amd-amdhsa" "-o" "test-a2836a.o" "-x" "ir" "test-
f66534.bc” \
"clang-linker-wrapper" "--device-compiler=amdgcn-amd-amdhsa=-O3" "--host-triple=x86_64-pc-linux-gnu" "-o" "a.out” "-lm"
          "-lomp" "-lomptarget" "-lomptarget.devicertl"
```

Debugging Offload Invocations

clang++ ”-###” –save-temps -fopenmp --offload-arch=gfx942 -O3 test.cpp

```
"clang" "-cc1" "-triple" "x86_64-pc-linux-gnu" "-E" "-dumpdir" "a-" "-save-temps=cwd" "-O3" "-fopenmp-targets=amdgcn-amd-amdhsa" "-o"
"test-host-x86_64-pc-linux-gnu.ii" "-x" "c++" "test.cpp"
"clang" "-cc1" "-triple" "x86_64-pc-linux-gnu" "-emit-llvm-bc" "-dumpdir" "a-" "-save-temps=cwd" "-fopenmp" "-disable-llvm-passes" "-
fopenmp-targets=amdgcn-amd-amdhsa""-o" "test-host-x86_64-pc-linux-gnu.bc" "-x" "c++-cpp-output" "test-host-x86_64-pc-linux-gnu.ii"
"clang" "-cc1" "-triple" "amdgcn-amd-amdhsa" "-aux-triple" "x86_64-pc-linux-gnu" "-E" "-dumpdir" "a-" "-save-temps=cwd" "-mlink-
builtin-bitcode" "example.bc" "-target-cpu" "gfx942" "-include" "__clang_openmp_device_functions.h" "-O3" "-fvisibility=protected" "-
fopenmp" "-fopenmp-is-target-device" "-o" "test-openmp-amdgcn-amd-amdhsa-gfx942.ii" "-x" "c++" "test.cpp"
"clang" "-cc1" "-triple" "amdgcn-amd-amdhsa" "-aux-triple" "x86_64-pc-linux-gnu" "-emit-llvm-bc" "-dumpdir" "a-" "-save-temps=cwd" "-
mlink-builtin-bitcode" "example.bc" "-target-cpu" "gfx942" "-O3" "-fopenmp" "-disable-llvm-passes" "-fopenmp-is-target-device" "-
fopenmp-host-ir-file-path" "test-host-x86_64-pc-linux-gnu.bc" "-o" "test-openmp-amdgcn-amd-amdhsa-gfx942.tmp.bc" "-x" "c++-cpp-output"
"test-openmp-amdgcn-amd-amdhsa-gfx942.ii"
"clang" "-cc1" "-triple" "amdgcn-amd-amdhsa" "-aux-triple" "x86_64-pc-linux-gnu" "-emit-llvm-bc" "-emit-llvm-uselists" "-dumpdir" "a-"
"-save-temps=cwd" "-mlink-builtin-bitcode" "example.bc" "-target-cpu" "gfx942" "-O3" "-fvisibility=protected" "-fopenmp" "-fopenmp-is-
target-device" "-o" "test-openmp-amdgcn-amd-amdhsa-gfx942.bc" "-x" "ir" "test-openmp-amdgcn-amd-amdhsa-gfx942.tmp.bc"
"/usr/local/bin/clang-offload-packager" "-o" "test-openmp-x86_64-pc-linux-gnu.out" "--image=file=test-openmp-amdgcn-amd-amdhsa-
gfx942.bc,triple=amdgcn-amd-amdhsa,arch=gfx942,kind=openmp"
"clang" "-cc1" "-triple" "x86_64-pc-linux-gnu" "-S" "-dumpdir" "a-" "-save-temps=cwd" "-O3" "-fopenmp" "-fembed-offload-object=test-
openmp-x86_64-pc-linux-gnu.out" "-fopenmp-targets=amdgcn-amd-amdhsa" "-o" "test-host-x86_64-pc-linux-gnu.s" "-x" "ir" "test-host-
x86_64-pc-linux-gnu.bc"
"clang" "-cc1as" "-triple" "x86_64-pc-linux-gnu" "-filetype" "obj" "-main-file-name" "test.cpp" "-target-cpu" "x86-64" "-o" "test-host-
x86_64-pc-linux-gnu.o" "test-host-x86_64-pc-linux-gnu.s"
"clang-linker-wrapper" "--device-compiler=amdgcn-amd-amdhsa=-O3" "--device-compiler=amdgcn-amd-amdhsa=-save-temps=cwd" "--host-
triple=x86_64-pc-linux-gnu" "--save-temps" "--linker-path=/opt/local/bin/ld" "--hash-style=gnu" "--eh-frame-hdr" "-m" "elf_x86_64" "-
pie" "-dynamic-linker" "/lib64/ld-linux-x86-64.so.2" "-o" "a.out" "Scrt1.o" "crti.o" "crtbeginS.o" "-L/usr/lib" "test-host-x86_64-pc-
linux-gnu.o" "-lstdc++" "-lm" "-lomp" "-lomptarget" "-lomptarget.devicertl" "-L/usr/local/lib" "-lgcc_s" "-lgcc" "-lpthread" "-lc" "-
lgcc_s" "-lgcc" "crtendS.o" "crtn.o"
```

Pass pipeline reduction

```
static bool BuggyGlobalFlag = false;

PreservedAnalyses
BuggyAttrPass::run(Module &M, ModuleAnalysisManager &) {
  BuggyGlobalFlag = true;

  for (Function &F : M) {
    if (!F.isDeclaration())
      F.addFnAttr("buggy-attr");
  }

  return PreservedAnalyses::all();
}
```

```
$ export BUGGY_PLUGIN_OPTS=crash-on-buggy-global-state
$ buggy_opt X86ISelLowering.cpp.clang-disable-llvm-passes.bc
```

Pass pipeline reduction

```
export BUGGY_PLUGIN_OPTS=crash-on-buggy-global-state

llvm-project/llvm/utils/reduce_pipeline.py \
  --opt-binary ~/llvm-project/build_reldebinfo/bin/opt \
  --passes='default<O2>' \
  --input X86ISelLowering.cpp.clang-disable-llvm-passes.bc \
  --output reduced-pipeline.out.ll \
  --load-pass-plugin=buggy_plugin.so
```

Pass pipeline reduction

```
$ llvm-reduce -o reduced-pipeline.ll --test=interestingness-multi-pass.sh \
  reduced-pipeline-output.ll
```

```
#!/usr/bin/env sh
! buggy-opt --disable-output \
    -passes="buggy-attr,function<eager-inv>(buggy<crash-on-buggy-global-state;>)" $@
```

```
The following extra args will be passed to opt: ['--load-pass-plugin=/home/marsenau/src/llvm-reduce-tutorial/build/buggy-plugin.so']
Expanded pass sequence: annotation2metadata,forceattrs,inferattrs,coro-early,function<eager-inv>(...
---Starting step #0---
-passes="annotation2metadata,forceattrs,inferattrs,coro-early,function<eager-inv>(...
---Starting step #1---
-passes="buggy-attr,function<eager-inv>(float2int,lower-constant-intrinsics,buggy<crash-on-buggy-global-state;>),loop(...
---Starting step #2---
-passes="buggy-attr,function<eager-inv>(float2int,lower-constant-intrinsics,buggy<crash-on-buggy-global-state;>)"
---Starting step #3---
-passes="buggy-attr,function<eager-inv>(buggy<crash-on-buggy-global-state;>)"
}--FINISHED---
Wrote output to 'reduced-pipeline-output.ll'.
-passes="buggy-attr,function<eager-inv>(buggy<crash-on-buggy-global-state;>)"
```

```
define void @_ZN4llvm17X86TargetLoweringC2ERKNS_16X86TargetMachineERKNS_12X86SubtargetE() {
ret void
}
```

Reducing Backend Issues

- Start debugging the same way as any middle end problem.
- Machine pass issues are a second phase
- Cannot do better than an IR reproducer for SelectionDAG issues
- Ideally get to reproducer command using llc
- Everything is generally worse in the backend
- Modularity breakdown and fixed pass orders
- CodeGen still using legacy pass manager

Reducing Backend Issues

```
$ clang –c –O3 –emit-llvm -o test.bc test.c
$ llc test.bc
```

Reducing Backend Issues

```
#!/usr/bin/env sh

! llc -filetype=obj "$@" -o- | FileCheck verify-machineinstrs $@
```

Reducing Backend Issues

- There are IR passes that run in the codegen pipeline
- These extra cleanups sometimes complicate reductions
- Can use `-start-after=passname` to skip extra passes
- Note not all passes should be skipped, and you will run into different problems if you skip the wrong ones
- Try using `-verify-machineinstrs`, though often prohibitively expensive inside the interestingness test

Reducing Backend Issues

```
$ llc -mtriple=x86_64-pc-linux-gnu -debug-pass=Structure test.bc

Pass Arguments:  -targetlibinfo -targetpassconfig -machinemoduleinfo -tti …
Target Pass Configuration
Machine Module Information
Target Transform Information
Type-Based Alias Analysis
Scoped NoAlias Alias Analysis
Assumption Cache Tracker
Profile summary info
Create Garbage Collector Module Metadata
Machine Branch Probability Analysis
Default Regalloc Eviction Advisor
Default Regalloc Priority Advisor
  ModulePass Manager
    Pre-ISel Intrinsic Lowering
    FunctionPass Manager
      Expand large div/rem
      Expand fp
…
   Prologue/Epilogue Insertion & Frame Finalization  # Pass name is PrologEpilogInserter, usable name is prologepilog
```

Reducing Backend Issues

- `llvm-reduce` supports MIR, but still rough
- MIR infrastructure could use improvement
- Routinely difficult to reproduce problems in machine passes
- `llc –stop-after=finalize-isel -o test.mir; llc –start-after=finalize-isel -filetype=null test.mir`
- `-stop-before=phi-node-elimination, -stop-before=register-coalescer, -stop-before=machine-scheduler, -stop-before=greedy`

Can go straight from source to MIR if you really need to with `–mllvm –stop-before`

Reducing Backend Issues

```
#!/usr/bin/env sh

! llc -x mir -mtriple=amdgcn-amd-amdhsa -start-before=machine-scheduler \
       -stop-after=virtregrewriter -verify-regalloc $@ 2> /dev/null
```

```
#!/usr/bin/env sh

! llc –run-pass=peephole-opt $@ 2> /dev/null
```

Debugging Miscompiles

- Good luck
- Mostly a function of debugging your program, not the compiler
- Sanitizers first if possible
- `opt -passes=lint` can report on some obvious undefined behavior in the IR
- Finding any kind of pass/fail diff is very helpful
- Executing misbehaving code in `llvm-reduce` script probably not that useful
- `llvm-reduce` interestingness script which compiles good/bad and runs llvm-diff
- `--opt-bisect` flag

Opt Bisect

- Just add `-opt-bisect-limit=<integer>` to any tool running passes
- Selectively disables optional passes

Opt Bisect

```
int main(int argc, const char* argv[]) {
  return argc < 2;
}
```

```
#!/usr/bin/env sh

export BUGGY_PLUGIN_OPTS="miscompile-icmp-slt-to-sle"
clang –O3 -fpass-plugin=buggy_plugin.so test.c -o test $@
./test 1
echo $?
1
```

Opt Bisect

```
$ ./miscompile_and_run.sh -opt-bisect-limit=-1
```

```
BISECT: running pass (1) Annotation2MetadataPass on [module]
BISECT: running pass (2) ForceFunctionAttrsPass on [module]
BISECT: running pass (3) AssignmentTrackingPass on [module]
BISECT: running pass (4) InferFunctionAttrsPass on [module]
BISECT: running pass (5) LowerExpectIntrinsicPass on main
BISECT: running pass (6) SimplifyCFGPass on main
BISECT: running pass (7) SROAPass on main
…
BISECT: running pass (67) Float2IntPass on main
BISECT: running pass (68) LowerConstantIntrinsicsPass on main
BISECT: running pass (69) ControlHeightReductionPass on main
BISECT: running pass (70) buggy on main
BISECT: running pass (71) LoopSimplifyPass on main
BISECT: running pass (72) LCSSAPass on main
…
BISECT: running pass (142) X86 LEA Fixup on function (main)
```

Opt Bisect

```
$ ./miscompile_and_run.sh -mllvm -opt-bisect-limit=70
0
```

```
…
BISECT: running pass (66) RecomputeGlobalsAAPass on [module]
BISECT: running pass (67) Float2IntPass on main
BISECT: running pass (68) LowerConstantIntrinsicsPass on main
BISECT: running pass (69) ControlHeightReductionPass on main
BISECT: NOT running pass (70) buggy on main
BISECT: NOT running pass (71) LoopSimplifyPass on main
BISECT: NOT running pass (72) LCSSAPass on main
BISECT: NOT running pass (73) LoopDistributePass on main
…
```

Miscellaneous Tools for Interestingness Scripts

- Shortcut with `llvm-extract –recursive --func=name`
- `llvm-extract –bb=func:bb1,bb2`
- `llvm-diff` to find minimal example for what a pass does
- `opt –passes=normalize` to reorder and rename instructions to shrink diffs

Cleaning Up Results for Tests

- `$ opt -S -passes=strip,instnamer`
- `$ opt -S -passes=strip,metarenamer`
- Prefer to avoid undefined behavior in test if not relevant to reproducer
  - e.g. replace `load` from `ptr null`

Useful options

clang

-save-temps -Wl,-plugin-opt=-save-temps

-mllvm -print-module-scope

-Xclang

-###

-save-temps -###

opt

-disable-output

-passes='lint<abort-on-error>'

-passes='strip,metarenamer'

General LLVM

-ir-dump-directory

-print-on-crash

-print-module-scope

-print-before, -print-before-pass-number=N

llc

-filetype=null, -filetype=obj

-O0 (-O2 is default)

-verify-machineinstrs -verify-regalloc, -verify-misched

-start-before=name –stop-after=name

llvm-extract

--recursive

–func=some-func

--bb=func:some-bb;bb2

llvm-diff

Globals to compare

llvm-reduce bugs

- uselistorder errors when using text IR
  - https://github.com/llvm/llvm-project/issues/58629
- “`input module no longer interesting after counting chunks`”
  - Can be symptomatic of a flaky test but could be a bug in a delta pass
- Errors with `-abort-on-invalid-reduction`
- Many in MIR reduction
  - Does multi threaded work correctly?
  - Missing block reduction

- If you do hit a bug in `llvm-reduce`, you know what to do

```
#!/usr/bin/env sh
INPUT=$1

# Must have at least one function in the module.
if ! grep -q "^define" $INPUT; then
  exit 1
fi
# Ignore cases that have dead code in the input
if grep -q "No predecessors" $INPUT; then
  exit 1
fi

TMPFILE=`mktemp` || exit 1

llvm-reduce --abort-on-invalid-reduction \
  --skip-delta-passes=unreachable-basic-blocks -o $TMPFILE \
  --test=interestingness.sh $@

reduce_status=$?
if [ $reduce_status -eq 2 ]; then
  # Exit code 2 indicates the input was not interesting.
  exit 1
fi

# Looking for case that introduced unreachable code.
grep -q "No predecessors" $TMPFILE
```

```
$ llvm-reduce -o meta-reduced.ll -v \
                --abort-on-invalid-reduction \
                --test="yo_dawg.sh" before-466784.ll
```

llvm-reduce llvm-reduce

```
define fastcc i1
@"_ZZNK4llvm17X86TargetLowering17LowerBUILD_VECTORENS_7SDValueERNS_12SelectionDAGEENK3$_0c
lES1_jNS_8ArrayRefIS1_EE"() {
entry:
  %0 = inttoptr i64 0 to ptr
  br i1 false, label %cond.true.i, label %_ZNK4llvm8ArrayRefINS_7SDValueEEixEm.exit.peel

_ZNK4llvm8ArrayRefINS_7SDValueEEixEm.exit.peel: ; preds = %entry
  %1 = load ptr, ptr %0, align 8
  ret i1 false

cond.true.i: ; preds = %entry
  unreachable
}
```

https://github.com/llvm/llvm-project/pull/133842

llvm-reduce: Fix introducing unreachable code in simplify conditionals

Reduction Suggestions

- llvm-reduce label on github

DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18


AMD