LIFE WITH OPAQUE POINTERS FROM A FRONTEND PERSPECTIVE

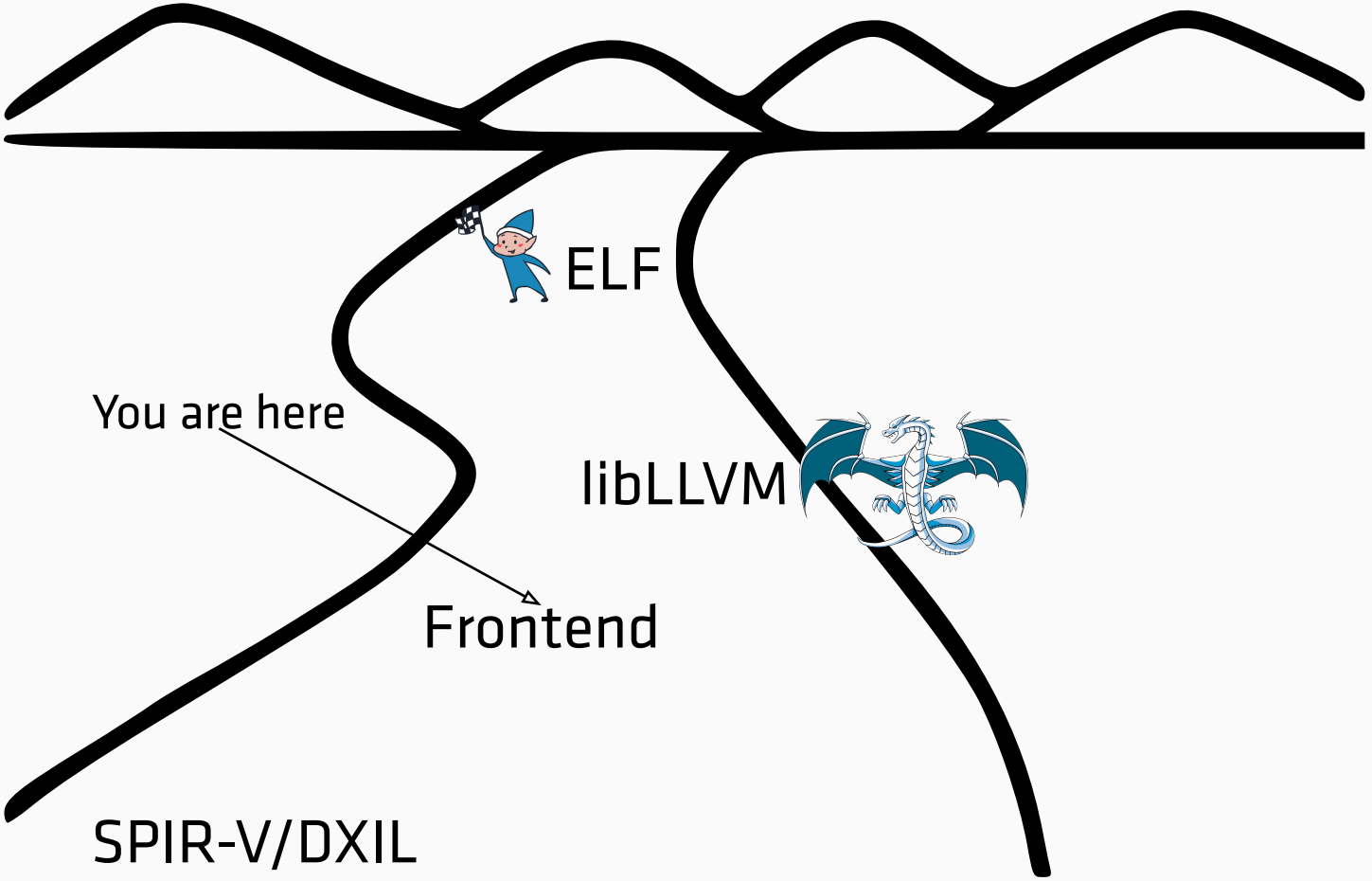
Sebastian Neubauer

April 10th, 2024

AMD

together we advance_

OVERVIEW

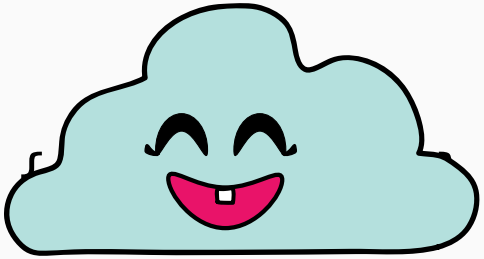


PROBLEM

```
{ i32, float } *
```

```
getPointerElementType
```

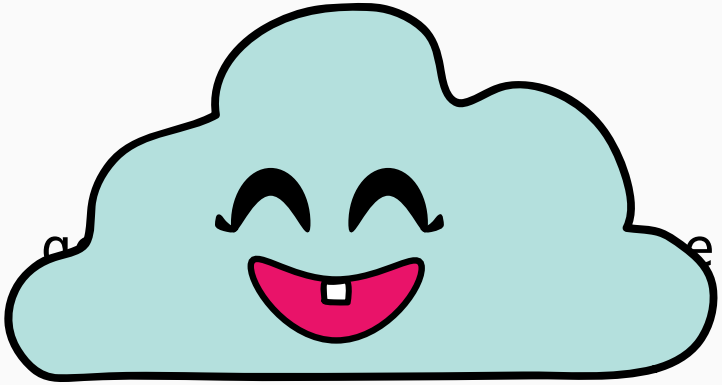
PROBLEM



*

Memory is untyped

⚡ Attack of Opaque Pointers



→ ???

CHANGES (GENERAL)

getPointerElementType

```
getValueType           // GlobalValue
getAllocatedType       // AllocaInst
getResultElementType   // GetElementPtrInst
…
```

```
Value *allocVal();
Value *createVal();
define void @func([10 x i32]* %arg)
getPointerElementType
```

```
AllocaInst *allocVal();
std::pair<Value *, Type *> createVal();
define void @func([10 x i32] %arg)
DenseMap<Value *, Type *> ElementTypes
```

CHANGES (SPIR-V)

Problem:

Type `[10 x i32]` has no semantic meaning

```
getelementptr [10 x i32],
              ptr %arr,
              i32 0,
              i32 5
```

Goal:

Want to know stride in later pass

Solution:

Use custom “intrinsic” to preserve array stride

```
call ptr @array.gep(ptr %arr,
                    /* stride */ i32 4,
                    /* index */ i32 5)
```

This slide is brought to you by `llvm-dialects`

CHANGES (DXIL)

CHANGES (DXIL)

DirectX 12 ULTIMATE

What is DXIL?

- Bitcode from ~LLVM 3.7

Example

```
type %struct.Payload = { i32, float }
void @shader(%struct.Payload* %payload) {}
```

CHANGES (DXIL)

DirectX XII ULTIMATE

What is DXIL?

- Bitcode from ~LLVM 3.7

Example

```
type %struct.Payload = { i32, float }
void @shader(%struct.Payload* %payload) {}
```

⚡ Attack of Opaque Pointers (+ Bitcode auto-upgrader)

```
void @shader(ptr %payload) {}
```

Problem:

Need type information from signatures

```
void @shader(ptr %payload) {}
```

Solution:

`BitcodeReader` hook saves types in metadata

```
type %struct.Payload = { i32, float }
void @shader(ptr %payload) !types !0 {}
!0 = !{%struct.Payload poison}
```

Problem:

Need type information from signatures

```
void @shader(ptr %payload) {}
```

Solution:

`BitcodeReader` hook saves types in metadata

```
type %struct.Payload = { i32, float }
void @shader(ptr %payload) !types !0 {}
!0 = !{%struct.Payload poison}
```

In C++:

```
shader->getArgOperand(0)
      ->getType()
      ->getPointerElementType()
```

```
cast<ConstantAsMetadata>(
  shader->getMetadata("types")
        ->getOperand(0))
  ->getType()
```

Implementation in `unittests/Bitcode/BitReaderTest.cpp`

SUMMARY

- Change function signatures and pass arguments by value
- Use intrinsics to preserve information
- `BitcodeReader` hook to save argument types in metadata
- **Tip**: Support opaque and typed pointers at the same time to easily switch back on regressions

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

QUESTIONS?


AMD