Compilers

About Me

- IRC: tstellar
- GSOC student while at the University of Oregon
- Added features and optimization to the r300 compiler
- I am currently employed by AMD.

Summary

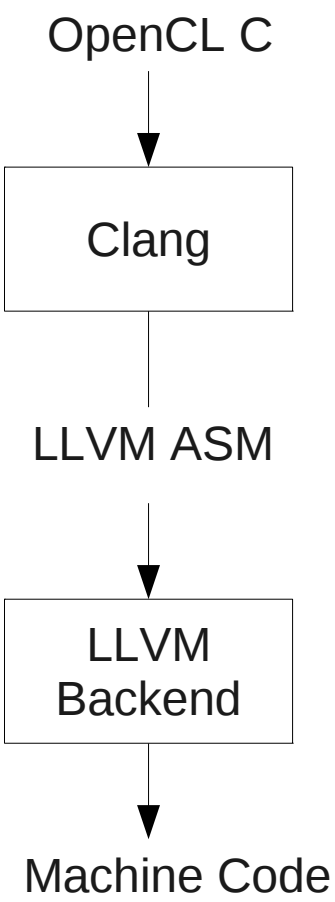
- OpenCL compiler stack
  - What it looks like
  - How to integrate with Mesa
- Sharing code between OpenCL and GLSL compilers
- What I have been working on lately

Mesa Compiler Goals

- Share code
- Compiler infrastructure for drivers
- Comprehensive test suites

OpenCL compiler

- LLVM Compiler Infrastructure
  - Clang frontend (C, C++, OpenCL)
  - LLVM Assembly
  - Optimization library
  - Target specific code generators (X86, PPC, ARM, etc.)
  - JIT execution



LLVM Backend

- TargetMachine
  - Registers definitions
    - Registers and sub-registers
    - Register Classes (f32, i32, v4f32, v4i32, etc.)
  - Instructions definitions
    - Pattern matching
    - Assembly string
    - HW encoding

Register Definitions

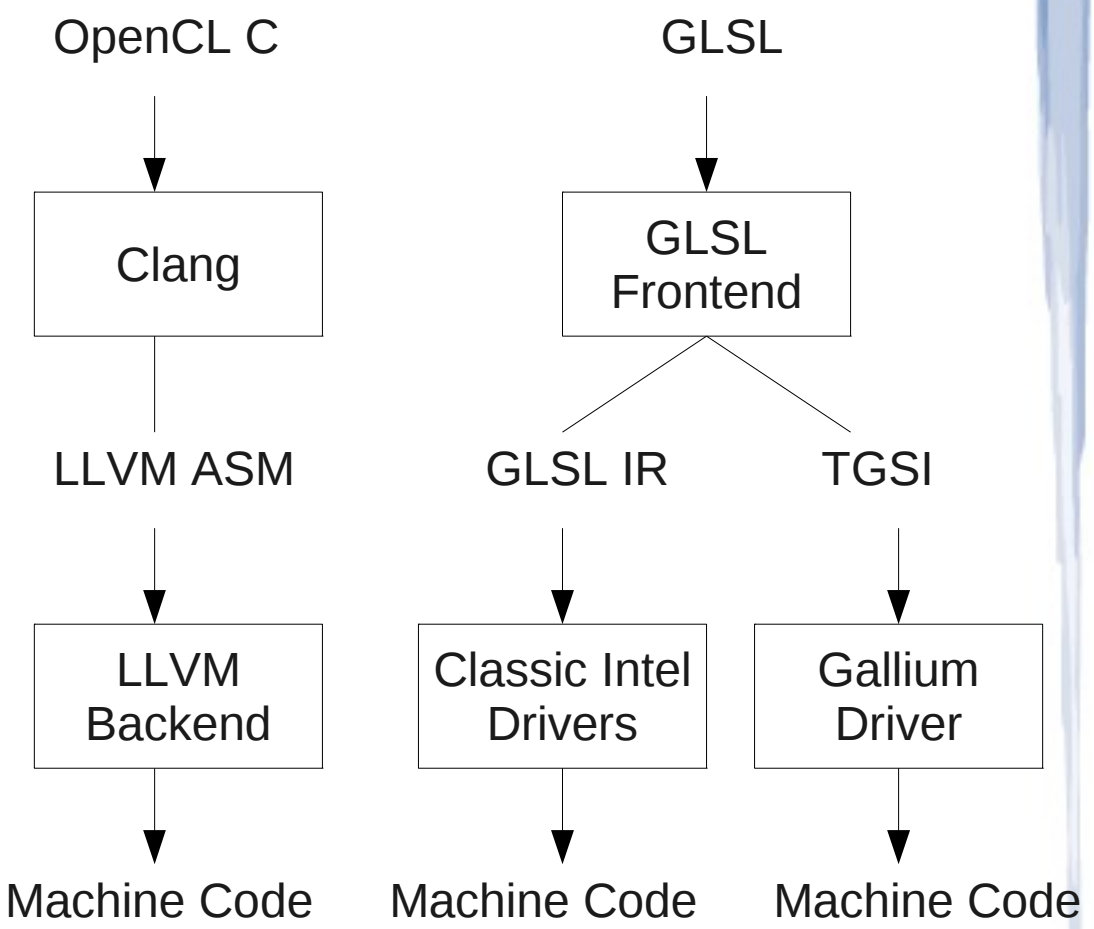
```
def R1x : R600Reg<1, "R1.x">
def R1y : R600Reg<2, "R1.y">
def R1z : R600Reg<3, "R1.z">
```

Instruction Definitions

```
def  MULLIT : R600Instruction <
  (outs GPRv4f32:$dst),
  (ins GPRf32:$src0, GPRf32:$src1,
GPRf32:$src2),
  "MULLIT $dst, $src0, $src1",
  [(set GPRv4f32:$dst, (int_R600_mullit
GPRf32:$src0, GPRf32:$src1,
GPRf32:$src2))]
>;
```

Integrating OpenCL compiler with Mesa

- Ideas:
  - Integrate LLVM Backend into Gallium driver
  - Write an LLVM backend that generates TGSI
  - Other ideas??



Ideas

- Integrating LLVM backend into Gallium drivers
  - Pros:
    - Easier to write GPU specific optimizations
    - A better trivial shader compiler
  - Cons:
    - Potential for 2 code emitters per driver
    - Might need to use LLVM for shaders
    - Each driver needs to implement an LLVM backend

Ideas

- LLVM -> TGSI Conversion
  - Pros:
    - We can reuse current code emitters
    - Only one LLVM backend is needed (TGSI)
  - Cons:
    - We need to extend TGSI to support OpenCL features (Pointers, Integers, Vector Types)
    - We have to roll our own compiler infrastructure for backends

Can we combine GLSL and OpenCL compilers?

- One backend for GLSL and OpenCL with LLVM
- Why should we do this?
  - Code sharing
  - More testing
  - LLVM Compiler Infrastructure
- How can we do this?
  - TGSI -> LLVM converter

TGSI->LLVM converter

- Already exists in Gallium llvmpipe
- We can pull code out of llvmpipe for a TGSI->LLVM converter
- Llvmpipe
  - Two ways to convert TGSI->LLVM:
  - Split vector int scalar elements
  - Pass vector types to LLVM (untested)

Challenges in using LLVM for shaders

- LLVM IR is designed for “traditional” hardware
  - However, infrastructure makes it easy to write own optimization passes
- We need to extend the instruction set using intrinsics
- Swizzles / Writemasks
  - We can implement these with intrinsics
  - We can also use creative instruction definitions
- Input / Output registers
  - We need to add instructions to emulate GPU operations

What I have been working on

- TGSI -> LLVM Converter
- LLVM backend for r600g