XDC 2018
Clover
this time with SPIR-V and NIR

Rob Clark & Karol Herbst

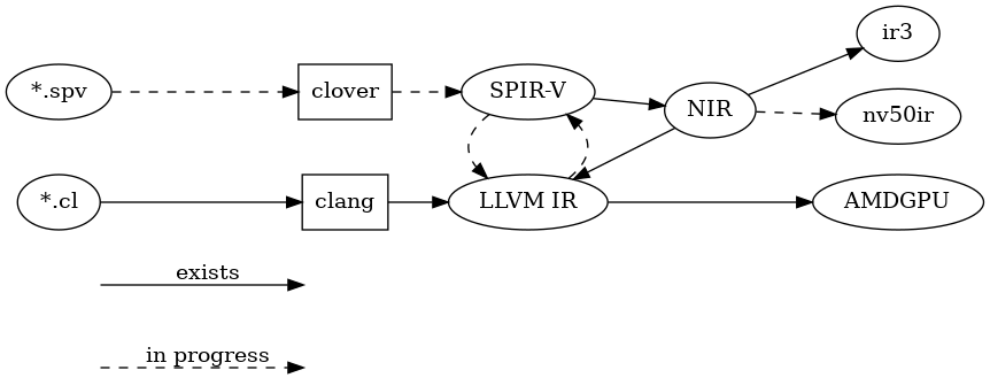
What we have

- mesa: Clover Gallium state tracker (r600, radeonsi)
- clang: OpenCL C frontend
- LLVM: LLVM to SPIR-V (upstream khronos project: SPIRV-LLVM-Translator)
- mesa: SPIR-V to NIR

The idea

- Why not use NIR as the general purpose backend IR within mesa?
- Translate various inputs into NIR
- Driver only needs to implement NIR to get all the goodies
- support Compute Shaders
- and `set_compute_resources` (but we have `set_constant_buffer` and `set_shader_images`

IR Overview


*.spv
clover
SPIR-V
NIR
ir3
nv50ir
*.cl
clang
LLVM IR
AMDGPU
exists
in progress

What is missing

- NIR
  - support real pointer (they are just ordinary values)
  - some intrinsics for memory operations and other random stuff
  - alu opcodes and lowering for OpenCL builtins
  - support for vec8 and vec16
  - rounding modes for conversions
- clover
  - support for SPIR-V files (required by OpenCL 2.1)
  - convert LLVM IR to SPIR-V

SPIR-V in Clover

- implementing `cl_khr_il_program` to accept SPIR-V
- required with OpenCL 2.1
- using spirv-tools for parsing and linking
- but, why not converting LLVM to SPIR-V and have one IR to support?
- great work by Pierre!

Pointers inside NIR!

- SPIR-V Graphics Profile only has opaque pointers
  - similar to GLSL
- SPIR-V Compute Profile adds “real” pointers
  - ie. everything you’d expect in C
  - pointer arithmetic
  - pointer casting
  - dereferencing
  - etc
- Wondering: what about ARB_bindless_texture?

Address Spaces

- But it gets worse..
- four disjoint address spaces:
  - global - what you would expect
  - local - shared by threads in a workgroup
  - constant - similar to global but read-only
    - implementation can optimize
    - ie. turn into push constant, etc
  - private - visible to thread
- SPIR-V adds function address space for function local memory
- You cannot cast pointers to different address space[*]

Generic Pointers

- But it gets even worse..
- generic points (OpenCL 2.0)
  - Pointers declared without address space qualifier are generic
  - Cast global, local or private pointer to generic
  - Implement function taking pointers once
- But we need different instructions to load/store different address spaces :-(
  - different intrinsic instructions in NIR
  - turns into different native instructions on most hardware
- How do we do this?

Fat Pointers!

- Turn pointers into vec2
  - fptr.x - pointer address
  - fptr.y - address space
- `nir_lower_io` turns pointer load/store into if/else ladder
- constant folding, etc, turns things back into something reasonable
  - at least in most cases
  - not if we stop inlining all the function calls
    - value range tracking might be helpful?
  - the fptr.y values are optimized out before coming out of SSA (assuming scalar arch)

The Details

- `nir_deref_ptr_as_array` - pointer to deref chain
  - starts a deref chain from a fat-ptr
  - foo->bar is same as foo[0].bar
- `nir_intrinsic_address_from_deref` - deref to pointer
  - gets a fat-ptr back from a deref chain
  - avoid having to use result of deref instruction as input to random ALU instructions, etc
  - keeps deref instruction result as opaque, ie. not having to fixup all the places where 32b vec1 is used

Problems

- `glsl_type` size vs actual data size vs vec2 size
- what if we have to store a generic pointer inside memory? (local or global)
- was running into several issues when fixing issues with nested pointers and SVM
- maybe address space translation is a nice way to workaround some issues?

Driver support

- WIP for Nouveau and freedreno
- little changes to drivers needed if they support NIR:
  - new Compute specific NIR intrinsics
  - may implement new NIR alu instructions to prevent lowering
  - require Clover related Gallium functions and caps
- Nouveau: additionally requires NIR to nv50ir (lacks review, Pierre is working on it!)

State of Work

- around 150 Patches pending!
- NIR path is slower than TGSI for Nouveau
- current pointer solution doesn't work out for edge cases
- breaks graphics :(
- waiting on review of other patches

Constant Folding of conversions

- int to float _rtz in pure C (no FPU or SSE):

```
'_rtz':
   "__typeof__(src0+0) max = ~(__typeof__(src0))0;\n" +
   "if ((__typeof__(src0))-1 < 0) max ^= (__typeof__(src0))1 << ((sizeof(src0) * 8) - 1);\n" +
   "dst = src0;\n" +
   "__typeof__(src0+0) y;\n" +
   "if (dst >= 2.0*(max/2 + 1)) y = max; else y = dst;\n" +
   "__typeof__(src0) abs_src0 = ((__typeof__(src0))-1 < 0) ? imaxabs(src0) : src0;\n" +
   "__typeof__(src0) abs_y = ((__typeof__(src0))-1 < 0) ? imaxabs(y) : y;\n" +
   "if (abs_y > abs_src0)\n" +
   "   dst = nextafter(dst, (__typeof__(dst))(dst > 0.0 ? -INFINITY : (__typeof__(dst))(dst <
   "else\n" +
   "   dst = nextafter(dst, (__typeof__(dst))0);\n",
```

- there has to be a more simple solution, right?

besides OpenCL

- SPIR-V could be used to support other languages
- OpenMP state tracker maybe?
- HMM