GLSL compiler: Where we've been and where we're going

Matt Turner

intel Software
Intel OpenSource TECHNOLOGY CENTER

In the last year

- Committed long awaited geometry shader support (recently for Sandybridge too!)
- Jumped from GLSL 1.40 to GLSL 3.30
- Tons of new extensions
  - separate_shader_objects (4.1) gpu_shader5 (4.0)
  - shader_atomic_counters (4.2) viewport_array (4.1)
  - sample_shading (4.0) explicit_uniform_location (4.3)
  - derivative_control (4.5) ...

In the last year

- Tons of easy algebraic optimizations
  - Amazing (and a bit disappointing) how many programs these help
- “Vectorizing” multiple scalar operations
  - Amazing how bad code from DX translators can be
- Finally implemented common subexpression elimination (kind of...)
  - Only works on constants and uniforms
- Realizing more and more that a tree-based IR makes things difficult

In the last year... in the i965 backend

- New SEL instruction peephole, dead control flow elimination
- Significant improvements to register allocation and instruction scheduling
- Rewritten vec4 and scalar dead code elimination passes
- Lots of register coalescing improvements
- New vec4 CSE pass
- Preserving the control flow graph across all optimization passes
- Realizing more and more that we want an SSA-based IR

How do we measure compiler improvements?

- Benchmarking games is often tedious and has a lot of variability
- apitraces don't work for benchmarking for a number of reasons
- Optimizations often individually too small to detect FPS changes
- Would like to measure improvements in generated code more directly

shader-db

- Collection of shaders gathered from games and benchmarks
  - Plus scripts to compile them and collect statistics
- 19599 *.shader_test files in my local checkout (GLSL and ARB vp/fp)
- Quick and easy to check whether an optimization helps or hurts real applications

```
glsl: Optimize open-coded lrp into lrp.

total instructions in shared programs: 1498191 -> 1487051 (-0.74%)
instructions in affected programs:     669388 -> 658248 (-1.66%)
GAINED:                                1
LOST:                                  0
```

GLSL code from DX translators

- What we get:

```
r1.w = inversesqrt( r7.x );
r2.w = inversesqrt( r7.y );
r0.w = inversesqrt( r7.z );
r7.x = 1.0 / r1.w;
r1.w = inversesqrt( r7.w );
r7.y = 1.0 / r2.w;
r7.w = 1.0 / r1.w;
r7.z = 1.0 / r0.w;
```

```
vec4 cmp(in vec4 src0, in vec4 src1, in vec4 src2)
{
vec4 result;
result.x = src0.x >= 0.0 ? src1.x : src2.x;
result.y = src0.y >= 0.0 ? src1.y : src2.y;
result.z = src0.z >= 0.0 ? src1.z : src2.z;
result.w = src0.w >= 0.0 ? src1.w : src2.w;
return result;
}
```

- What we'd like to get:

```
r7 = sqrt(r7);
```

```
vec4 cmp(in vec4 src0, in vec4 src1, in vec4 src2)
{
return mix(src2, src1, greaterThanEqual(src0, 0.0));
}
```

A year's worth of compiler improvements

```
total instructions in shared programs: 5777098 -> 4823707 (-16.50%)
instructions in affected programs:     5558170 -> 4604779 (-17.15%)
GAINED:                                1717
LOST:                                  14
```

- SIMD16 programs increased from 88.6% (16401/18497) to 97.8%
- 43559 programs helped, 9512 unchanged, 110 hurt
- Cut number of loops in programs by ~10%
- Cut number of basic blocks by 16.49%
- Cut number of CFG calculations by 92%

Questions (so far)

The fires are (mostly?) out. What to do now?

- Have been reactionary for a long time
- New Steam games usually just work these days
  - And if not, usually only small fixes required
- Can afford to think about longer term investments
- Lack of compiler infrastructure has hurt us in the past
  - i965's fs dead code elimination pass without a CFG

What do we actually want? (i965 backend)

- SSA
  - Existing optimization passes become more efficient and more effective
  - Allows for new optimizations like GCM-GVN and divergence analysis
- An SSA-based register allocator
  - Can register allocate in polynomial time! (Maybe!)
  - Can make better decisions about register usage

What do we actually want? (glsl compiler)

- A flat (non-tree-based) SSA IR
  - Wouldn't it be nice to do GCM-GVN in a place common to all drivers?
- To translate both to and **from** TGSI
  - For drivers that don't want to write all of the same optimizations again
- Something other people (i.e., non-Intel) will also work on

Questions after Connor's talk