GLSL compiler:
Where we've been and where we're going
(2015 Edition)

Matt Turner
In the last year

- Added SSA-based NIR (the **New Intermediate** Representation)
- NIR in use by default in i965/vec4, i965/fs, vc4, and freedreno
  - Net reduction of 2 backends in i965 (fs, fp → NIR/fs, vs, vp → NIR/vec4)
- Mostly stopped working on the tree-based “GLSL IR”
  - 48 optimization patches to NIR
  - 9 optimization patches to GLSL IR
In the last year... in the i965 backend

- NIR enabled by default
  - Cut 12% of instructions in ARB fragment programs
- Added pass to combine immediate-value loads
  - Packs 8 values into each register
  - Allows unconditional use of MAD instructions!
- New conditional-modifier propagation pass
- Added flag-register dead code elimination
  - Rewrote vec4 dead code elimination pass
(New!) shader-db

- Still a collection of 25k *.shader_test files gathered from games and benchmarks
  - Plus scripts to compile them and collect statistics
- “Runner” script replaced by nice C program using the latest goodness
  - Render nodes, EGL, GBM, libepoxy
  - Single process, uses OpenMP to compile shaders in parallel
  - Feeds compiler stats (instruction counts, loops, spills, etc) back via KHR_debug
  - 300 second runtime reduced to 90
Another year's worth of compiler improvements

- Previous year was -16.50%, so about -25% in two years
- Broadwell and newer uses scalar mode for vertex shaders
- Support for SIMD16 on Gen4 (Improved FPS of Shadowrun Returns by 20%)
- Support for SIMD16 with control flow added on Gen4 and Gen5 (ILK and older)

### Instruction Counts

<table>
<thead>
<tr>
<th>Description</th>
<th>Before</th>
<th>After</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total instructions in shared programs</td>
<td>6615500</td>
<td>5996928</td>
<td>-9.35%</td>
</tr>
<tr>
<td>Instructions in affected programs</td>
<td>6165481</td>
<td>5575266</td>
<td>-9.57%</td>
</tr>
<tr>
<td>GAINED</td>
<td>236</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOST</td>
<td>154</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Questions (so far)
IRs (2014)
IRs (2015)
NIR is here. What to do now?

- NIR is an **additional** IR. Hasn't simplified situation overall
- Reduce or remove optimizations in GLSL IR
  - GLSL IR optimizations are inefficient and slow (and sometimes not even effective)
  - Shadertoy.com shader compiles in 25 seconds... or 5
- GLSL linking in NIR
  - Would allow better pre-linking optimizations
- Get rid of Mesa IR
IRs (future)
What do we need?

- Buy-in from TGSI consumers to go through NIR
- How they would benefit:
  - Share more code, optimizations, lowering passes (gl_ClipVertex, GL_CLAMP, texture rectangle scaling, texture projection, integer division)
  - Get to delete `st_glsI_to_tgsi.cpp` and `ir_to_mesa.cpp` (NIR ↔ TGSI remaining)
  - Better compile times, in some cases significantly
  - SPIR-V support
  - Maybe better generated code
What do we need to do?

- Port Mesa IR consumers to NIR (i915, r200, swrast)
- Port Mesa IR producers to NIR (FF vertex pipeline, ARB fp, ARB vp)
- Port FF fragment pipeline to NIR
- Fix-up NIR ↔ TGSI translators…?
- Of course, testing and benchmarking
Discussion