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

Matt Turner

In the last year

- Added SSA-based NIR (the **N**ew **I**ntermediate **R**epresentation)
- 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

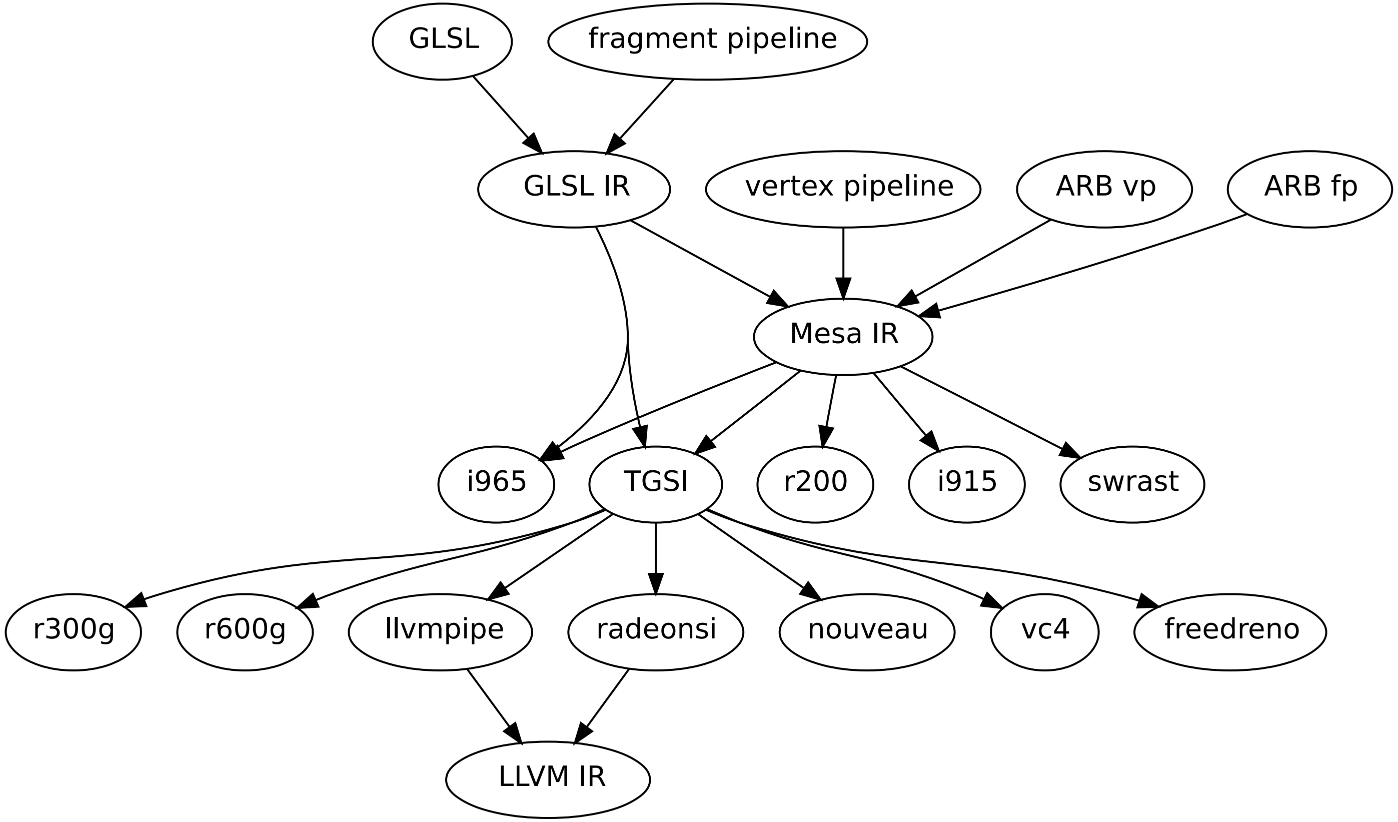
Another year's worth of compiler improvements

```
total instructions in shared programs: 6615500 -> 5996928 (-9.35%)
instructions in affected programs:     6165481 -> 5575266 (-9.57%)
GAINED:                                236
LOST:                                  154
```

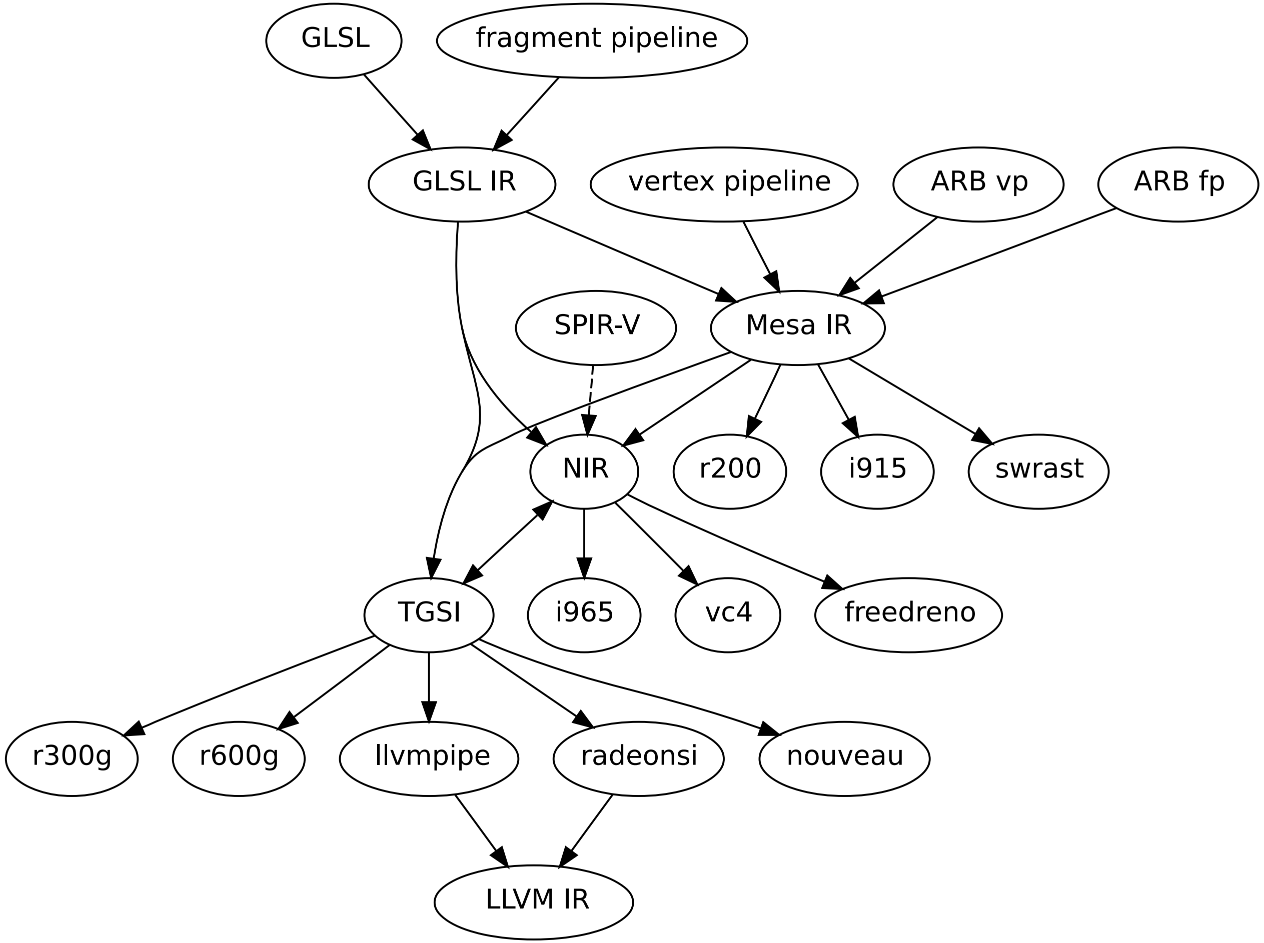
- Previous year was -16.50%, so about -25% in two years
- Broadwell and newer uses scalar mode for vertex shaders
- Support for SIMD16 on Gen4 (Improved FPS of Shadowrun Returns by 20%)
- Support for SIMD16 with control flow added on Gen4 and Gen5 (ILK and older)

Questions (so far)

IRs (2014)


GLSL
fragment pipeline
GLSL IR
vertex pipeline
ARB vp
ARB fp
Mesa IR
i965
TGSI
r200
i915
swrast
r300g
r600g
llvmpipe
radeonsi
nouveau
vc4
freedreno
LLVM IR

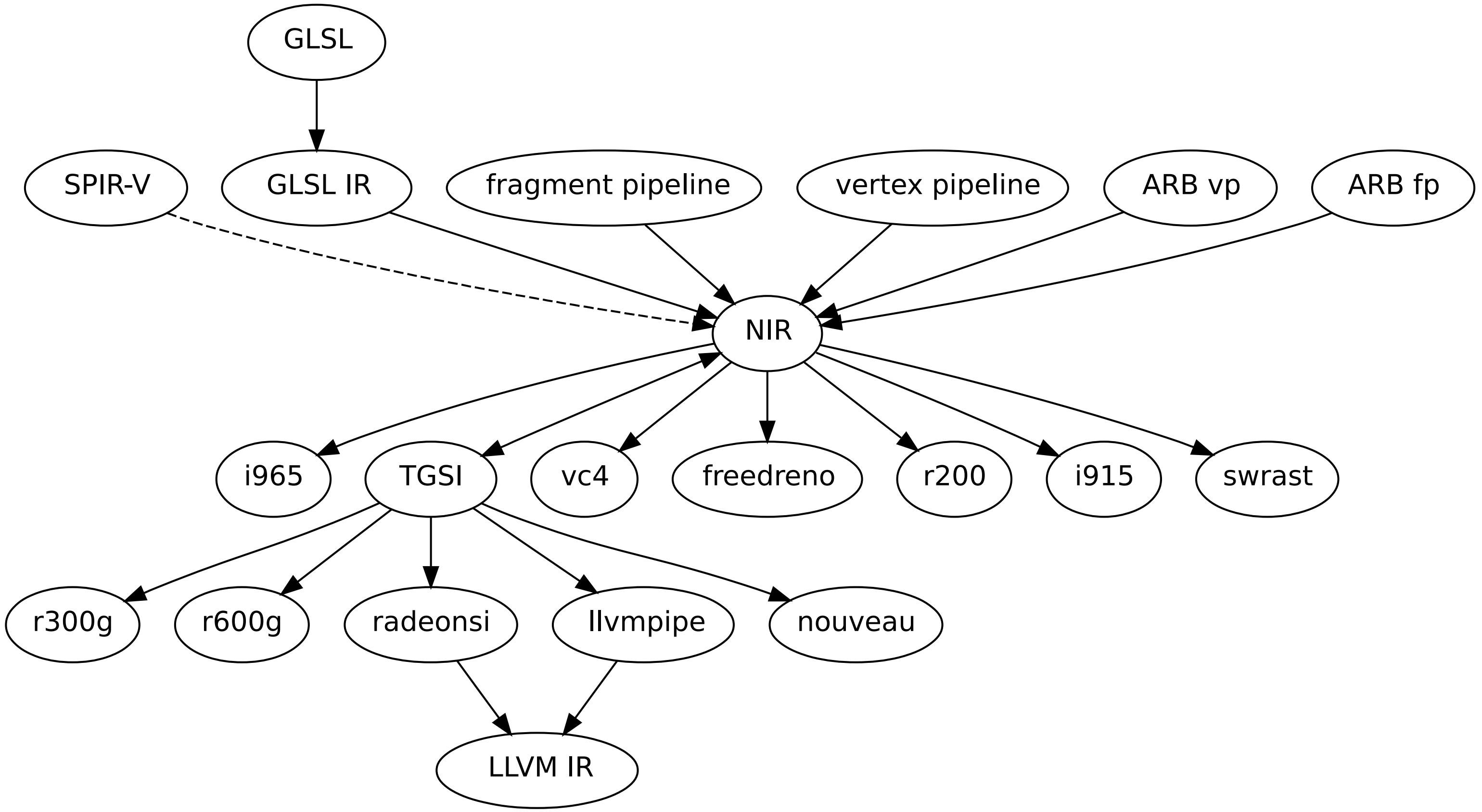
IRs (2015)


GLSL
fragment pipeline
GLSL IR
vertex pipeline
ARB vp
ARB fp
SPIR-V
Mesa IR
NIR
r200
i915
swrast
TGSI
i965
vc4
freedreno
r300g
r600g
llvmpipe
radeonsi
nouveau
LLVM IR

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)


GLSL
SPIR-V
GLSL IR
fragment pipeline
vertex pipeline
ARB vp
ARB fp
NIR
i965
TGSI
vc4
freedreno
r200
i915
swrast
r300g
r600g
radeonsi
llvmpipe
nouveau
LLVM IR

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 rectange scaling, texture projection, integer division)
  - Get to delete **st_glsl_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