ACO, a new compiler backend for GCN GPUs

2019-10-02

Bas Nieuwenhuizen
Daniel Schürmann

Who are we?

- **Bas Nieuwenhuizen**
  - RADV maintainer since it started existing (Summer 2016)

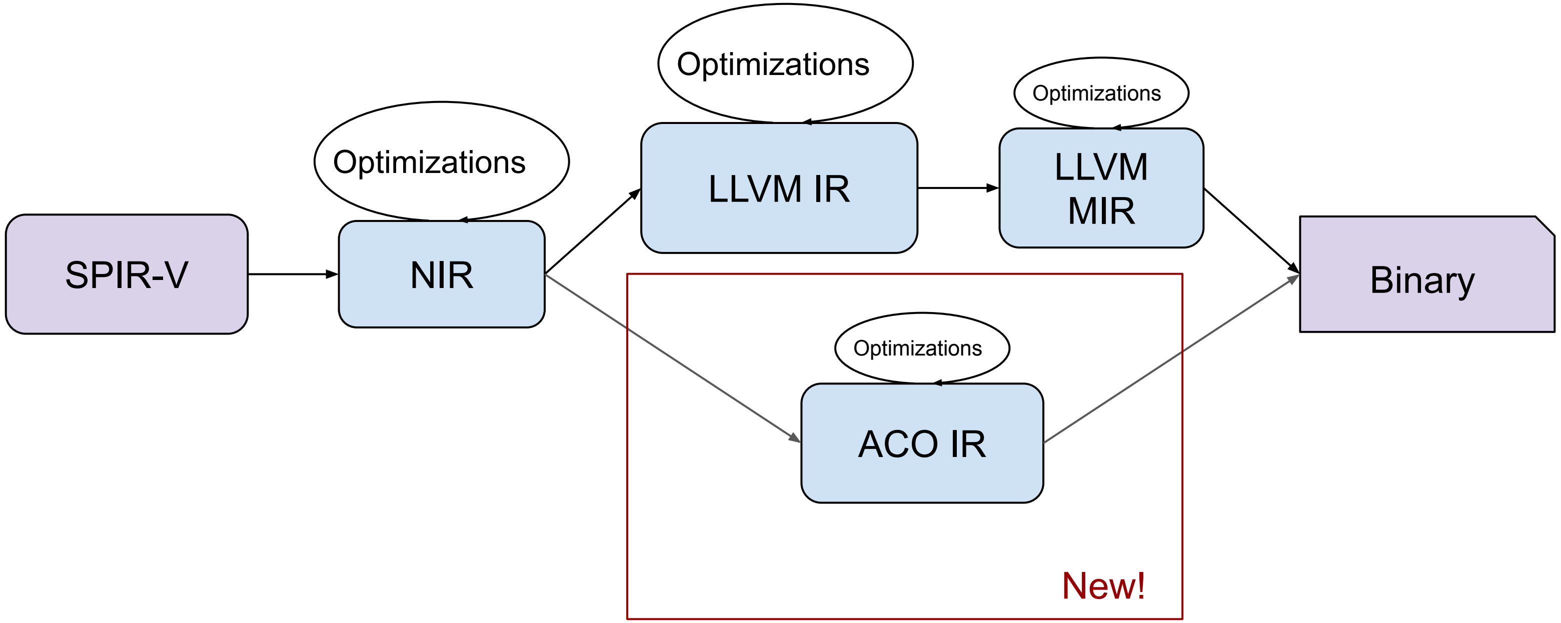
- **Daniel Schürmann**
  - Contracted by Valve Corporation
  - Working on RADV since February 2018

Outline

- A new compiler backend
- High level overview
- Register Pressure Control
- Current Status
- Performance results
- Challenges & Future

A new compiler backend?

- RADV: Independent Vulkan Driver for AMD GCN GPUs



A new compiler backend?

1. Improve control flow handling by directly using NIR
   - NIR stores the structured control flow of shaders

2. Improve compile time performance

Using NIR - Control Flow

- In hardware multiple invocations get executed using SIMD

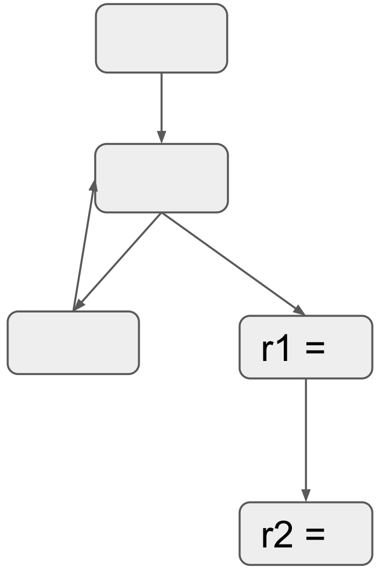
- Scalar registers sometimes needed for
  - Performance
  - Correctness

Using NIR - Control Flow

```
uniform texture2d textures[64];

void main() {
   int i;
   vec4 r1, r2;

   for (i = 0;; ++i) {
       if (i == g_localInvocationIndex) {
          r1 = texture(texture2d[i], vec2(0.0)); OK
          break;
       }
   }
   r2 = texture(texture2d[i], vec2(0.0)); Not OK
}
```

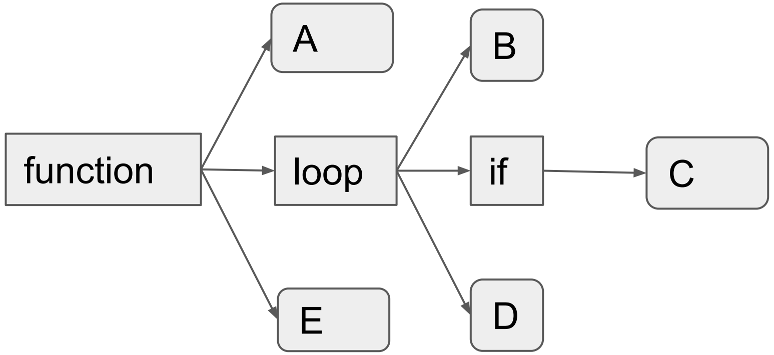
Using NIR - Control Flow

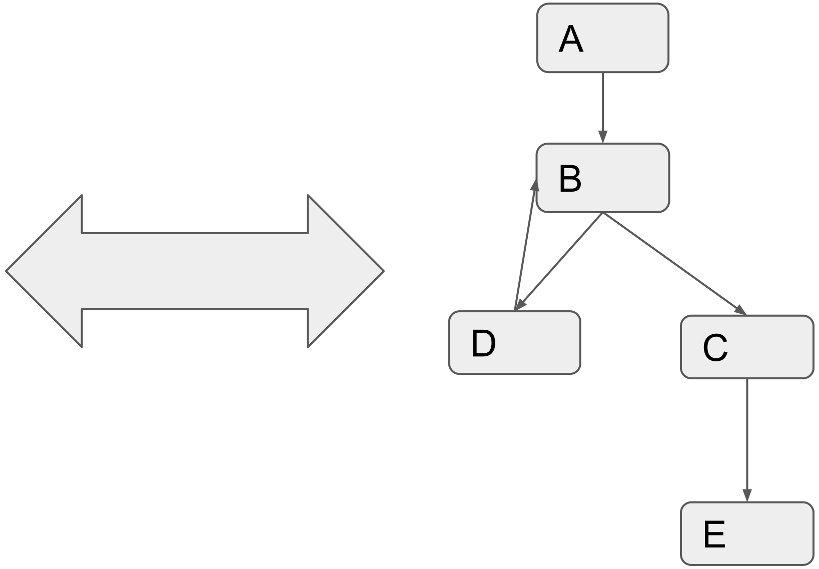
```
uniform texture2d textures[64];

void main() {
   int i;
   vec4 r1, r2;

   for (i = 0;; ++i) {
       if (i == g_localInvocationIndex) {
          r1 = texture(texture2d[i], vec2(0.0)); OK
          break;
       }
   }
   r2 = texture(texture2d[i], vec2(0.0)); Not OK
}
```



Using NIR - Control Flow

NIR stores structure control flow:



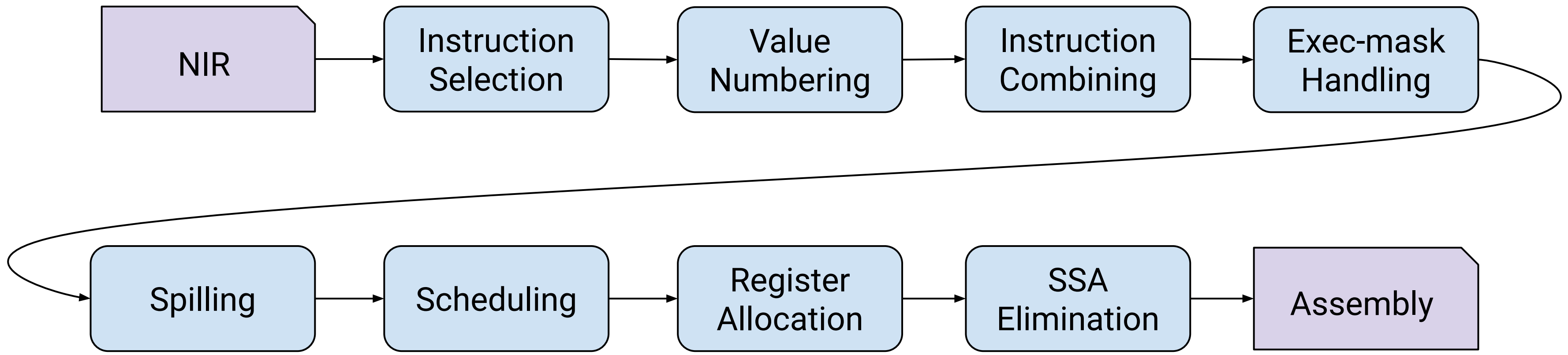


With LLVM this information was thrown away

High level overview

- “Modern” Compiler Construction Principles
- Written in C++
- Divergence-aware instruction selection
- Logical vs linear CFG
- SSA-IR based on hardware ISA
- SSA-based Register Allocation

High level overview



Register Pressure Control

Problem:

- Scheduling might increase register pressure
- High register pressure might lead to spilling
- On AMD GPUs, high register pressure lowers parallelism (occupancy)

Register Pressure Control

Solution: Control the register pressure!

- Via SSA-based Spilling/RA
- And scheduling under register pressure constraints

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  2)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs:  2)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  2)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs:  2)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder          <-----
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  2)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs:  2)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0                  ←——— current
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder          ←——— insertion
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  2)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs:  2)     v2: %5 = p_create_vector %4, %3                    ←——current
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder          ←——insertion
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  2)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0                    <---current
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
                                                                               <---insertion
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  2)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x     <------current
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
                                                                                  <------insertion
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0                  ←——current
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
                                                                             ←——insertion
(vgprs:  4, sgprs: 11)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  3, sgprs:  2)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs:  2)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs:  2)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  4, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  4, sgprs: 11)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x     <-----insertion
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)    s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)    s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)   p_logical_start                                           <———current
(vgprs:  2, sgprs:  3)    s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)    s8: %29 = s_load_dwordx8 %28, 32 reorder
                                                                                    <———insertion
(vgprs:  3, sgprs: 11)    v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 11)    v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 11)    v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 11)    v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 11)    v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)    s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)    v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)    v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)    v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)    v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  2, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  3, sgprs: 11)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 11)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 11)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 11)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder  ←insertion
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  2, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  3, sgprs: 11)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 11)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 11)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 11)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder  <------ insertion
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z         <------ current
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  2, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  3, sgprs: 11)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 11)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 11)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 11)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  5, sgprs: 13)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
                                                                                  ←———insertion
(vgprs:  4, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z      ←———current
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  2, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  3, sgprs: 11)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 11)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 11)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 11)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 11)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  5, sgprs: 13)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)    s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)    s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)   p_logical_start
(vgprs:  2, sgprs:  3)    s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)    s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  2, sgprs: 13)    s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  3, sgprs: 13)    v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 13)    v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 13)    v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 13)    v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 13)    v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)    v2: %33 = p_wqm %5
(vgprs:  5, sgprs: 13)    v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  5, sgprs: 13)    v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
(vgprs:  4, sgprs:  1)    v1: %32 = image_sample %33, %29, %31 dmask:w reorder
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)    s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)    s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)   p_logical_start
(vgprs:  2, sgprs:  3)    s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)    s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  2, sgprs: 13)    s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  3, sgprs: 13)    v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 13)    v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 13)    v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 13)    v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 13)    v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)    v2: %33 = p_wqm %5
(vgprs:  5, sgprs: 13)    v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  5, sgprs: 13)    v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
(vgprs:  4, sgprs:  1)    v1: %32 = image_sample %33, %29, %31 dmask:w reorder
...
```

Register Pressure Control

```
BB0
(vgprs:  2, sgprs:  2)     s1: %19:s[2],  s1: %20:s[3],  v1: %21:v[0],  v1: %22:v[1],  s2: %23:exec = p_startpgm
(vgprs:  2, sgprs:  3)     s2: %45:exec,  s1: %44:scc = s_wqm_b64 %23:exec
(vgprs:  2, sgprs:  2)    p_logical_start
(vgprs:  2, sgprs:  3)     s2: %28 = p_create_vector %19, 0
(vgprs:  2, sgprs: 11)     s8: %29 = s_load_dwordx8 %28, 32 reorder
(vgprs:  2, sgprs: 13)     s4: %31 = s_load_dwordx4 %28, 0 reorder
(vgprs:  3, sgprs: 13)     v1: %26 = v_interp_p1_f32 %21, %20:m0 attr0.y
(vgprs:  3, sgprs: 13)     v1: %3 = v_interp_p2_f32 %22, %20:m0, %26 attr0.y
(vgprs:  4, sgprs: 13)     v1: %27 = v_interp_p1_f32 %21, %20:m0 attr0.x
(vgprs:  4, sgprs: 13)     v1: %4 = v_interp_p2_f32 %22, %20:m0, %27 attr0.x
(vgprs:  4, sgprs: 13)     v2: %5 = p_create_vector %4, %3
(vgprs:  4, sgprs: 13)     v2: %33 = p_wqm %5
(vgprs:  3, sgprs:  1)     v1: %32 = image_sample %33, %29, %31 dmask:w reorder
(vgprs:  4, sgprs:  1)     v1: %34 = v_interp_p1_f32 %21, %20:m0 attr0.z
(vgprs:  4, sgprs:  1)     v1: %9 = v_interp_p2_f32 %22, %20:m0, %34 attr0.z
...
```

Register Pressure Control

Results: LLVM -> ACO

- +9.40 % needed SGPRs
- +2.65 % needed VGPRs
- -95.91 % less SPGR Spilling
- -100.00 % less VGPR Spilling
- -5.24 % max Waves
- -7.90 % less Code Size

Current Status

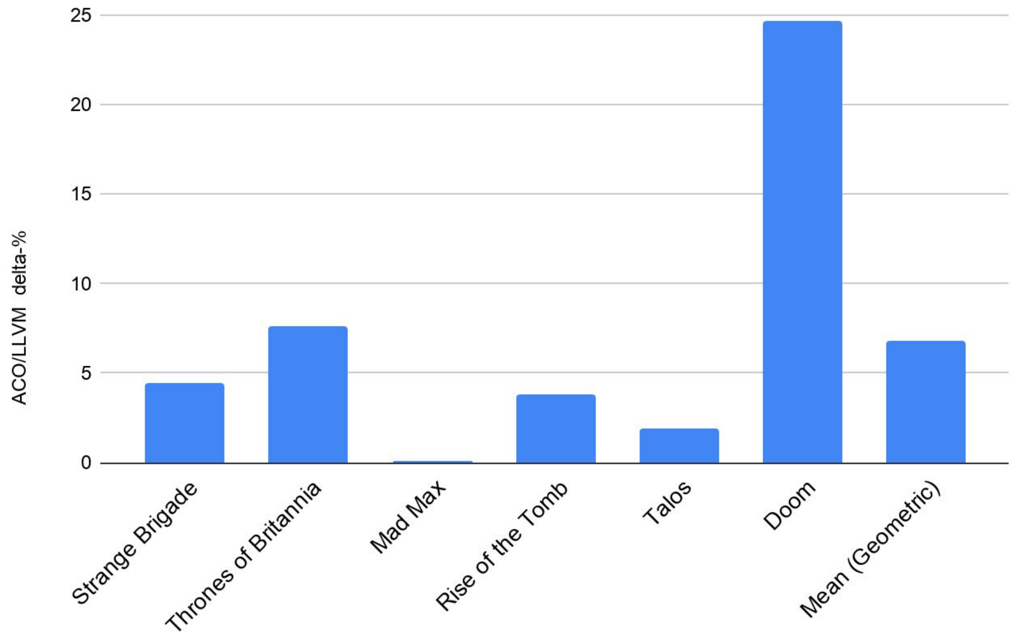
- Fully working VS, FS & CS
- Supports same extensions as RADV/LLVM except for < 32bit types.
- Currently GFX8 & GFX9 only
- Same CTS pass rate as RADV/LLVM

Current Status

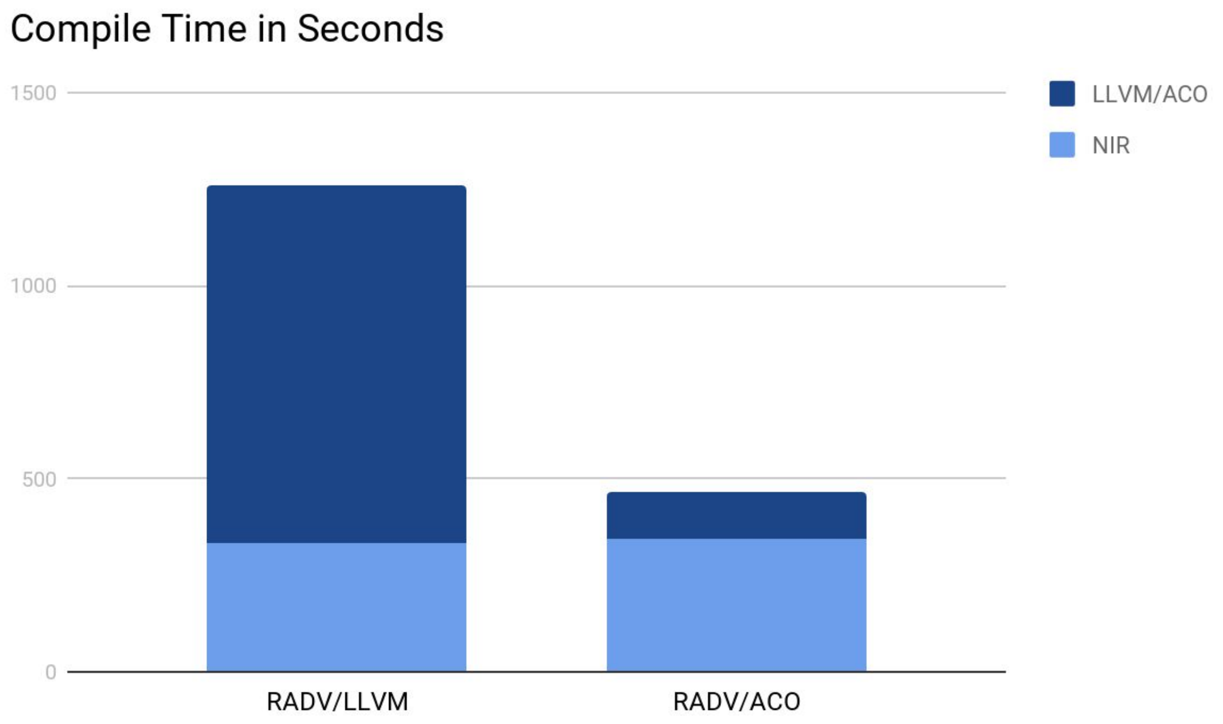
- ACO has been widely announced by Valve in July 2019
    - Resulted in 118 bug reports and tons of feedback

- Now merged into upstream Mesa
    - Included in 19.3 release
    - … but behind RADV_PERFTEST=aco

Runtime Performance



Compile Performance



Future: Making ACO the default

- Complete support for GPUs
  - Navi support in progress
- Finish remaining Vulkan extensions

- No date on making ACO the default yet
    - First have to make sure it has bug-parity

Challenges

- Wider adoption
  - Radeonsi?
- More performance
  - Alias analysis
- Testing
  - Game traces?
  - Unit tests?
  - both?

End

Questions?