Compute Support for Nouveau

Creating a LLVM to TGSI and a SPIR-V to NV50 IR backend

Hans de Goede, Pierre Moreau
About Us

Hans de Goede
- Software engineer for Red Hat's graphics team
- Nouveau developer since 2015

Pierre Moreau
- PhD Student in Computer Graphics at Lunds Tekniska Högskola, Sweden
- Nouveau developer since 2013
Summary

I. Recap of Mesa's Compute Stack
II. Converting SPIR-V to NV50 IR
III. Converting LLVM IR to TGSI
IV. Conclusion
Recap of Mesa's Compute Stack

Application
- SPIR-V binary
- OpenCL

Mesa
- Clover
  - LLVM IR
  - SPIR-V
    - Hans' work
    - Pierre's work
- Nouveau's TGSI converter
  - NV50 IR
  - GPU code
  - Nouveau's lowering pass

clang
Presentation of NV50 IR

• Custom *Intermediate Representation* (IR) used by Nouveau internally for all shaders (and now kernels)

• Keeps track of *Control Flow Graph* and variables' uses

• The Nouveau compiler performs multiple optimisation passes on NV50 IR, before lowering it to machine code
Presentation of SPIR-V

- Introduced by Khronos in 2015 as the IR fed into Vulkan, for shaders and kernels
- Binary format, supports extensions
- Is in Static Single-Assignment form, and might have gone through optimisation passes
Presentation of SPIR-V (cont.)

- Required capabilities and extensions
- Memory model and entry points
- Some debug information
- Types
  - Constants
  - Functions

Code snippet:
```
; SPIR-V
; Version: 1.0
; Generator: Khronos LLVM/SPIR-V Translator; 10
; Bound: 13
; Schema: 0

OpCapability Addresses
OpCapability Kernel
%1 = OpExtInstImport "OpenCL.std"
OpMemoryModel Physical32 OpenCL
OpEntryPoint Kernel %6 "pi"
OpSource OpenCL_C 102000
OpName %7 "out"
OpName %8 "entry"
OpName %11 "arrayidx"
%2 = OpTypeVoid
%3 = OpTypeFloat 32
%4 = OpTypePointer CrossWorkgroup %3
%5 = OpTypeFunction %2 %4
%9 = OpTypeInt 32 0
%10 = OpConstant %9 0
%12 = OpConstant %3 3.14159
%6 = OpFunction %2 None %5
%7 = OpFunctionParameter %2 None %5
%8 = OpLabel
%11 = OpInBoundsPtrAccessChain %4 %7 %10
OpStore %11 %12 Aligned 4
OpReturn
OpFunctionEnd
```
NV50 IR (and Mesa) Befriends SPIR-V

- Uses KhronosGroup/\{SPIRV-LLVM, SPIR\} from GitHub
- Integrate with clover: SPIR-V generation
- Integrate with Nouveau: advertise compute and SPIR-V support
- Need to design new storage class for non-vec4 elements, and of different sizes
SPIR-V → NV50 IR: Current Status

What Works:
• Arithmetic and comparison ops
• Branching without phi nodes
• Some builtins
• Array/pointer indexing
• Vector support
• Casts (not all of them)

What Doesn't Work:
• Phi nodes
• Images
• Atomics
• Loops
• Swizzles
• Function calling (almost)
• Some builtins and ops
Presentation of TGSI

- Tungsten Graphics Shader Infrastructure
- Intermediate language for shaders used in gallium (mesa), modelled after DX9 shader-ir
- Uses four component vector registers and operations, following the SIMD design of (DX9) GPUs at the time
- Somewhat cumbersome for current Nvidia GPUs which are not SIMD.
LLVM Befriends TGSI

• Based on Francisco Jerez' TGSI llvm backend work from 2013

• Several issues due to TGSI differences from typical assembly syntax:
  – Using a single vector component requires adding swizzling postfixes
  – Immediates need to be declared before the program and then addressed as IMM[x] rather then just writing the immediate value
  – Used registers need to be declared beforehand

• libclc support for get_local_id() and friends
LLVM → TGSI: Current Status

- clang can now compile this:

```c
__kernel void test_kern(__global uint *vals, __global uint *buf) {
    uint id = get_local_id(0);
    buf[32 * id] -= vals[id];
}
```

- Into:

```c
...
```
DCL SV[0], BLOCK_ID[0]
DCL SV[1], BLOCK_SIZE[0]
DCL SV[2], GRID_SIZE[0]
DCL SV[3], THREAD_ID[0]
DCL TEMP[0]
DCL TEMP[1]
...
DCL TEMP[31]
IMM UINT32 { 7, 0, 0, 0 }
IMM UINT32 { 4, 0, 0, 0 }
IMM UINT32 { 2, 0, 0, 0 }
IMM UINT32 { 0, 0, 0, 0 }
BGNSUB

SHL TEMP[1].x, SV[3].xxxx, IMM[0].xxxx
LOAD TEMP[1].y, RINPUT.xxxx, IMM[1]
UADD TEMP[1].x, TEMP[1].yyyy, TEMP[1].xxxx
SHL TEMP[1].y, SV[3].xxxx, IMM[2].xxxx
LOAD TEMP[1].z, RINPUT.xxxx, IMM[3]
UADD TEMP[1].y, TEMP[1].zzzz, TEMP[1].yyyy
LOAD TEMP[1].z, RGLOBAL.xxxx, TEMP[1].yyyy
INEG TEMP[1].y, TEMP[1].yyyy
LOAD TEMP[1].z, RGLOBAL.xxxx, TEMP[1].xxxxx
UADD TEMP[1].y, TEMP[1].yyyy, TEMP[1].zzzz
STORE RGLOBAL.x, TEMP[1].xxxxx, TEMP[1].yyyy
RET
ENDSUB
LLVM → TGSI: What is missing?

• TGSI backend:
  – Support for doubles, vectors
  – Control flow (if / for / while) support
  – Function call support
  – Support for multi-dimensional input / output data

• clover:
  – Integration of clang/llvm TGSI support into clover

• libclc:
  – Currently only supports get_local_id
  – everything else is missing
Nouveau and OpenCL: What Is missing?

- Image support (being worked on by Ilia Mirkin and Samuel Pitoiset)
- Atomics support (being worked on by Ilia Mirkin)
- Memory barriers / fences
- Support more GPU models
Questions?

- **Git:**
  - **SPIR-V:**
    - https://phabricator.pmoreau.org/diffusion/MESA
  - **LLVM → TGSI:**
    - http://cgit.freedesktop.org/~jwrdegoede/llvm
    - http://cgit.freedesktop.org/~jwrdegoede/clang
    - http://cgit.freedesktop.org/~jwrdegoede/libclc

- **Contact:**
  - Hans de Goede <hdegoede@redhat.com>
  - Pierre Moreau <pierre.morrow@free.fr>