CIS 6010 Final Project

Saahil Gupta & Dylan Li

ShadeIR

ShadeIR is short for *shader intermediate representation*. It is a lightweight, domain-specific shader language designed to compile down to NVIDIA's PTX (Parallel Thread Execution) assembly. The intent behind ShadeIR is to explore the architecture of modern GPU compilation pipelines and create a simplified, educational model of how high-level shader code can be lowered into low-level GPU-executable code.

At its core, ShadeIR will serve as a minimal yet expressive intermediate representation language that sits between high-level shading logic and PTX, enabling experimentation with compiler optimizations, register tracking, and control flow management in a GPU-centric context.

Below is an example of the proposed syntax:

```
[vector] i32 input = @loadbuf(0)
[vector] i32 accum

#while_exec
  #exec_mask input > 0
  accum += input
  input -= 1
#end_exec

@writebuf(0) <= accum
```

In this snippet, ShadeIR demonstrates a simple compute operation that loads a vector buffer, performs iterative accumulation under an execution mask, and writes the results back to memory.

Main Project Goals

The primary goal of ShadeIR is to design and implement a working compiler toolchain that translates ShadeIR source code into valid PTX output. Beyond simple code translation, we aim to implement mid-level optimization passes that reflect real-world compiler strategies, including:

- **Loop Unrolling** to use registers to represent indexable memory
- **Live Register Tracking** to manage GPU register allocation and reduce unnecessary data movement
- **Execution Masking** to mimic SIMD-style thread predication commonly found in GPU execution models

Technology

For this project, we will be using C++/Rust, along with a number of resources mimicking the LLVM tech stack.

- ANTR4
- clang
- PTX

Milestones

- **Milestone 1**: Complete the language spec and frontend language parser
- **Milestone 2**: Complete the middle PTX translator, output to a PTX file
- **Milestone 3**: Complete the backend PTX testing environment, optimize compiler
- **Milestone 4**: More optimizations, create test environments, shared memory

Expected Results

By project completion, ShadeIR will be capable of compiling small, efficient compute kernels into PTX that can execute directly on NVIDIA GPUs. Users will be able to write concise ShadeIR code, compile it through our toolchain, and run the result through a C++ host program interacting with CUDA buffers.

The project will demonstrate a working understanding of:

- The structure and semantics of shader intermediate representations
- GPU execution models and memory hierarchies
- The compilation process from high-level IR to low-level GPU assembly

Stretch Goals

If time permits, we plan to extend ShadeIR beyond PTX by adding a backend that targets **SPIR-V**, the intermediate language used in Vulkan and OpenCL. This would significantly broaden ShadeIR's applicability across different GPU vendors and APIs. Additionally, future work could include a visual IR inspector or an interactive compiler debugger to visualize execution masks and register states.