CS259 - LAB2

Objective:
The aim of this lab is to familiarize you with the process of extending GPU microarchitecture by implementing a new software prefetch instruction to reduce global memory latency. We will use the Vortex OpenGPU platform (Vortex OpenGPU GitHub) for this experiment. Please ensure you are working with the latest code from the `main` branch.

Assignment Breakdown:
This assignment consists of four key steps:

1. **ISA Extension**: Extend the GPU ISA by adding a new prefetch instruction.
2. **Testing**: Write a sample kernel program to verify your prefetch extension.
3. **Simulation**: Implement the new instruction in the SimX cycle-level simulator.
4. **Evaluation**: Assess your prefetching effectiveness through performance analysis.

Part1 - ISA Extension (2 points)
You will add a new RISC-V custom instruction for memory prefetch: **VX_PREFETCH.**
This new instruction takes a single source operand *rs1* which represents the memory address to prefetch and has no destination operand: *Vx_prefetch(addr)*.

You will need to modify ***Vortex/runtime/include/vx_intrinsics.h*** to define your new VX_PREFETCH instruction.

```
// prefetch
inline void vx_prefetch(uint32_t a) {
    asm volatile (".insn r ?, ?, ?, ?, ?, ?" : "=r"(?) : "i"(?), "r"(?),
"r"(?));
}
```

Use the R-Type RISC-V instruction format.

| funct7 | rs2 | rs1 | funct3 | rd | opcode |
| 7 bits | 5 bit| 5 bit| 3 bits |5 bit| 7 bits |

- **opcode**: 0x0B.
- **func7**: 1
- **func3**: 0
- **rs1**: <address>
- **rs2**: 0
- **rd**: 0

Read the following doc to understand **insn** speudo instruction format:
https://sourceware.org/binutils/docs/as/RISC_002dV_002dFormats.html

We recommend checking out how other custom GPU instructions with a similar signature (like **VX_TMC**) are implemented and decoded in the codebase.

Part2 - Test kernel implementation (2 point)
You need to create a test kernel program to validate your new prefetch extension. We recommend that you clone one of the GPU regression tests and modify both the host and kernel source code. Use the existing **mstress** test program as a baseline since they are both memory-centric, and modify it to add your prefetch instruction call. You will need to modify the host program such that the input size is directly set from the command line.

Part3 - Simulation support (4 points)
Extend the GPU's cycle-level simulator to support the new prefetch instruction.
Your new instruction should be implemented as a new operation for the LSU unit.
Extend the decode (*decode.cpp*) and execute (*func_unit.cpp*) stages of SimX to support the new prefetch instruction. Your software prefetch instruction can be implemented like a load instruction that doesn't have a return value. The idea is to have the LSU issue a load to the cache and not wait for the result to come back. For this, you will have to also modify the cache (*cache_sim.cpp*) and extend its request interface such that prefetch requests do not return a response.
Use VX_TMC as reference for decoding your new instruction and VX_FENCE as reference for implementing the execution of your new instruction..

Part4 - Evaluation (2 points)
You will add a new performance counter to SimX for tracking the effectiveness of your prefetch extension. The counter should be used to report the prefetch efficiency (useful prefetches / total prefetch calls) x 100. A useful prefetch is a prefetch that has generated a cache hit. Use a class2 performance counter for this addition. You will plot a graph that compares the IPC of your application with and without software prefetching. Also report the prefetch efficiency. Use an input size of 4096 elements with an 8x8 single-core. Use the following data cache sizes: 32KB, 16KB, 8KB, and 4KB.

Submission Guidelines:

Please submit the following two files with **"exact"** names:

1. **source.diff:** A diff file showing the changes you made to the source code. Ensure the diff reflects all modifications, including the kernel and host code changes.
2. **report.pdf:** A written report detailing your evaluation results. The report should include a summary of your approach, performance analysis, graphs, and insights from your profiling experiments.