CS259 - LAB2

The objective of this lab is to introduce you to the basics of extending GPU microarchitecture to accelerate a kernel in hardware

.
We will be using the Vortex OpenGPU (https://github.com/vortexgpgpu/vortex) for this experiment.

You will be using the latest code from branch**: develop**

Part1 - ISA extension (1 points)

You will add a new RISC-V custom instruction for computing the integer dot product; **VX_DOT8.**

VX_DOT8 calculates the dot product of two small vectors of int8 integers.
Each source register (rs1 and rs2) holds four int8 elements. The operation proceeds as follows:

Extract four int8 values from each source register:
rs1 holds A1, A2, A3, A4
rs2 holds B1, B2, B3, B4
Perform the dot product operation:
Dot Product = (A1*B1 + A2*B2 + A3*B3 + A4*B4)
The result, which could potentially be a 32-bit integer due to overflow, is stored in the destination register (rd).

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: Specific opcode reserved for custom instructions.
- rs1, rs2: Source registers containing vectors of four `int8` values each.
- rd: Destination register.
- funct3 and funct7: Used for further specifying the operation (like saturation and accumulation options).

Use custom extension opcode=0x0B with func7=1 and func3=0;
Your new instruction will be implemented as a new operation for the ALU unit.

You will need to modify vx_instrinsics.h to define your new VX_DOTP8 instruction

```
// DOT8
inline int vx_dot8(int a, int b) {
    size_t ret;
    asm volatile (".insn r ?, ?, ?, ?, ?, ?" : "=r"(?) : "i"(?), "r"(?),
"r"(?));
    return ret;
}
```

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 VX_SPLIT and VX_PRED instructions are decoded in SimX as reference.

Part2 - Sample matrix multiplication (2 points)

Implement a simple matrix multiplication GPU code that uses your new H/W extension.
Here is a basic CPU implementation example:

```
void matrixMultiply(int8_t A[][N], int8_t B[][N], int32_t C[][N], int N) {
   int i, j, k;
   int packedA, packedB;
   for (i = 0; i < N; i++) {
      for (j = 0; j < N; j++) {
         C[i][j] = 0;
         for (k = 0; k < N; k += 4) {
            // Pack 4 int8_t elements from A and B into 32-bit integers
            packedA = *((int*)(A[i] + k));
            packedB = *(int*)(B[k] + j)
                        | (*(int*)(B[k+1] + j) << 8)
                        | (*(int*)(B[k+2] + j) << 16)
                        |  (*(int*)(B[k+3] + j) << 24);
            // Accumulate the dot product result into the C matrix
            C[i][j] += vx_dot8(packedA, packedB);
         }
      }
   }
}
```

Part3 - SimX implementation (3 points)

Modify the cycle level simulator to implement the custom ISA extension.
You will need to modify decode.cpp, execute.cpp, func_unit,cpp primarily for this to work.
You will assume 2 cycles latency for the operation.
We recommend checking out how VX_SPLIT and VX_PRED instructions are decoded in SimX as reference.

Part4 - RTL implementation (4 points)

Modify the RTL code to implement the custom ISA extension.
You will need to modify VX_decode.sv, VX_alu_unit.sv primarily.
You will add a new module called VX_alu_dot8 and add it VX_alu_unit.sv similar to how we added aVX_alu_muldiv.
You will assume 2 cycles latency for the operation.
We recommend checking out how MULT instructions are decoded in RTL as reference.

Report:

You will compare your new accelerated sgemm program with the existing sgemm kernel under the regression codebase. You will use N=128 and (warps=4, threads=4) and (Warps=16, threads=16) for 1 and 4 cores on the SimX and RTLSIM driver. Plot the results reporting the IPC,

Bonus (2 points):
You will get a bonus if you write an optimized kernel that leverages the GPU local memory to tile the computation on 16x16 elements (256 bytes tiles).

What to submit?
1- zip file containing report.pdf and changes.diff.
The report should be about 2 pages showing the generated plots.