ECE-5580 Computer Architecture: Final Project

Superscalar Fixed-Length Vector Pipeline

Dylan-Matthew Garza Elijah Sargeant

Department of Electrical and Computer Engineering
Western Michigan University

November 16, 2023

Table of Contents

Background on Vector Processors

- Vector architectures implement SIMD (Single Instruction Multiple Data) by operating on sequential register files that contain multiple elements.
- Increases performance by reducing the total number of instructions needed on repeated computations.
- Vector architectures usually are not standalone implementations in modern applications and usually are implemented in MME (Multimedia Extensions):
  - Intel's AVX (Advanced Vector Extensions) & AMD's SSE (Streaming SIMD Extensions)
  - GPUs (SIMD with vector operations)
- Intel and AMD vector extension's support fixed-vectors

Implementation

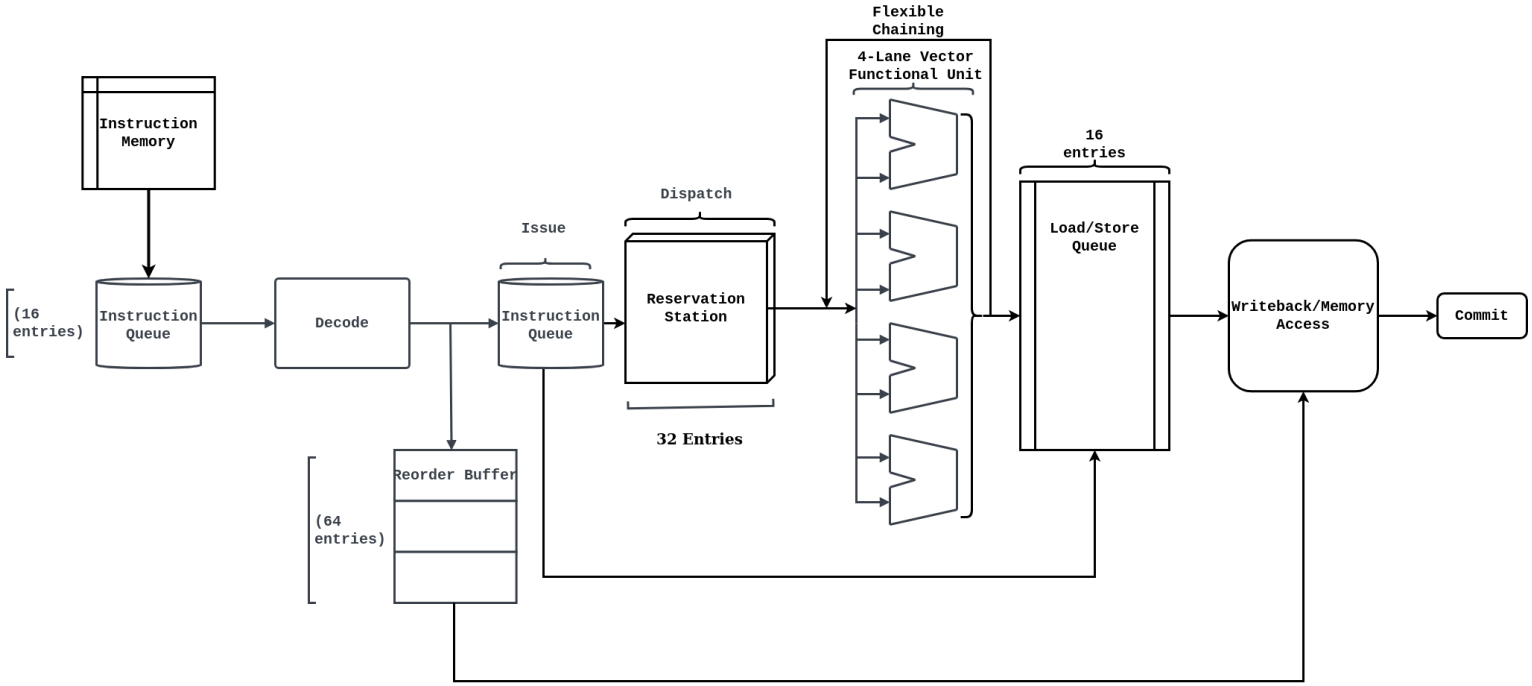

Figure: Block Diagram of proposed system's pipeline

- Fixed Length vector operands (8 elements)
- Out-of-Order Execution width multi-issue width(4 instructions)
- Reorder Buffer and Reservation Station
- Multi-lane execution with Chaining

SIMD Trace

- Using Intel's Pin tool to dynamically analyze a running binary
- Profiled AVX/AVX2 instructions to collect vector instructions
- Scalar instructions are excluded since vector architecures only handle vector instructions.

```
33  VOID
34  Instruction(INS ins, VOID *v)
35  {
36      unsigned category = INS_Category(ins);
37    if (category == XED_CATEGORY_AVX || category == XED_CATEGORY_AVX2 || category == XED_CATEGORY_AVX512)
38      {
39        std::string instruction = INS_Mnemonic(ins);
40        std::string dstReg = GetRegName(INS_RegW(ins, 0));
41        std::string srcReg1 = INS_OperandCount(ins) > 0 ? GetRegName(INS_RegR(ins, 0)) : "";
42        std::string srcReg2 = INS_OperandCount(ins) > 1 ? GetRegName(INS_RegR(ins, 1)) : "";
43        std::string cat = CATEGORY_StringShort(INS_Category(ins));
44
45        INS_InsertCall(ins, IPOINT_BEFORE, (AFUNPTR)LogSIMDInstruction,
46                       IARG_PTR, new std::string(instruction),
47                       IARG_PTR, new std::string(dstReg),
48                       IARG_PTR, new std::string(srcReg1),
49                       IARG_PTR, new std::string(srcReg2),
50                       IARG_PTR, new std::string(cat),
51                       IARG_END);
52      }
53  }
```

Figure: Written pin tool to filter intel Vector instructions

SIMD Trace

```c
void
multiply_matrices(float *A, float *B, float *C, int n) {
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j += 8)
        {
            // Process 8 elements at a time
            volatile __m256 sum = _mm256_setzero_ps(); // Initialize sum vector to 0
            for (int k = 0; k < n; k++)
            {
                volatile __m256 a_vec = _mm256_set1_ps(A[i * n + k]);
                volatile __m256 b_vec = _mm256_loadu_ps(&B[k * n + j]);
                sum = _mm256_add_ps(sum, _mm256_mul_ps(a_vec, b_vec));
            }

            _mm256_storeu_ps(&C[i * n + j], sum);
        }
    }
}
```

Figure: C program utilizing AVX intrinsics

```
65532    VMULPS DEST ymm0 SRC ymm0 ymm2
65533    VADDPS DEST ymm0 SRC ymm0 ymm1
65534    VMULPS DEST ymm0 SRC ymm0 ymm2
65535    VADDPS DEST ymm0 SRC ymm0 ymm1
65536    VMULPS DEST ymm0 SRC ymm0 ymm2
65537    VADDPS DEST ymm0 SRC ymm0 ymm1
65538    VMULPS DEST ymm0 SRC ymm0 ymm2
65539    VADDPS DEST ymm0 SRC ymm0 ymm1
65540    VMULPS DEST ymm0 SRC ymm0 ymm2
65541    VADDPS DEST ymm0 SRC ymm0 ymm1
65542    VMULPS DEST ymm0 SRC ymm0 ymm2
65543    VADDPS DEST ymm0 SRC ymm0 ymm1
65544    VMULPS DEST ymm0 SRC ymm0 ymm2
65545    VADDPS DEST ymm0 SRC ymm0 ymm1
```

Figure: trace file generated

- Will assume a latency of 2 cycles for the vector addition and 8 cycles for vector multiplication (latency applied to each execution unit)
- Will entire vector data is loaded simultaneously
- Simulator will vary the amount of lanes in each set of execution units, but will not be bottlenecked by the amount of units

Goals & Challenges

Goals

- Vary lane amount to see how average IPC and max IPC is affected. Will varry lanes from 1 to 8.

Challenges

- Writing the pipeline simulator
  - Potentially challenging