Introduction to Commodity SIMD

Robert D. Cameron

School of Computing Science
Simon Fraser University

May 8, 2024
What Are Vector Instructions?

Single Instruction Multiple Data (SIMD)

- Vector instructions implement the SIMD concept.
- Multiple vector elements are combined in a single operation.

Example:

\[
\begin{align*}
\text{Vector Addition} & \\
A_0 & + B_0 + C_0 \\
A_1 & + B_1 + C_1 \\
A_2 & + B_2 + C_2 \\
A_3 & + B_3 + C_3
\end{align*}
\]

Four 16-bit additions at once. Replaces the loop:

\[
\text{for (int i=0; i<4; i++) } \\
\text{C}[i] = A[i] + B[i];
\]

Intel \textit{paddw} MMX instruction.
What Are Vector Instructions?

Single Instruction Multiple Data (SIMD)
- Vector instructions implement the SIMD concept.
- Multiple vector elements are combined in a single operation.

Example: \(<4 \times i16>\) Vector Addition

- Four 16-bit additions at once.
- Replaces the loop:
  ```c
  for (int i=0; i<4; i++) {
      C[i] = A[i] + B[i];
  }
  ```
- Intel paddw MMX instruction.
Two Decades of Intel SIMD Extensions

MMX, SSE, AVX, AVX-512

- 1999: 128-bit floating point SIMD vectors (SSE).
  - Widespread Intel/AMD standard: all x86-64 processors.
Two Decades of Intel SIMD Extensions

**MMX, SSE, AVX, AVX-512**

- 1999: 128-bit floating point SIMD vectors (SSE).
  - Widespread Intel/AMD standard: all x86-64 processors.

**Not Just Intel/AMD**

- Altivec/VMX (2004): 128-bit SIMD
- ARM SVE: scalable SIMD up to 2048 bits (in progress)
Doubling Register Size, Repeatedly

128-bit SSE2: \(<8 \times i16>\) Vector Addition

\[
\begin{align*}
A_0 & \leftarrow A_0 + B_0 + C_0 \\
A_1 & \leftarrow A_1 + B_1 + C_1 \\
A_2 & \leftarrow A_2 + B_2 + C_2 \\
A_3 & \leftarrow A_3 + B_3 + C_3 \\
A_4 & \leftarrow A_4 + B_4 + C_4 \\
A_5 & \leftarrow A_5 + B_5 + C_5 \\
A_6 & \leftarrow A_6 + B_6 + C_6 \\
A_7 & \leftarrow A_7 + B_7 + C_7
\end{align*}
\]
Doubling Register Size, Repeatedly

128-bit SSE2: \(<8 \times i16>\) Vector Addition

AVX2, AVX-512
- 256-bit AVX2: \(<16 \times i16>\) vector addition in a single operation.
- 512-bit AVX-512: \(<32 \times i16>\) vector addition in a single operation.
Why SIMD?

SIMD Costs

- Intel has added hundreds of SIMD instructions over two decades.
  - More SIMD instructions added than any other kind.
  - Substantial transistor count, chip area devoted to SIMD.
  - Larger cores, so fewer cores per package possible.

SIMD Benefits

- SIMD naturally supports data parallel applications.
  - Graphics, signal and image processing.
  - Physical simulation.
  - Database queries, financial analytics.

SIMD has natural advantages over multicore.
- Cost of instruction fetch/decode divided by SIMD vector length.
- SIMD ALUs share common control and data path logic.
- Synchronization of parallel execution is automatic.
Why SIMD?

**SIMD Costs**
- Intel has added hundreds of SIMD instructions over two decades.
  - More SIMD instructions added than any other kind.
  - Substantial transistor count, chip area devoted to SIMD.
  - Larger cores, so fewer cores per package possible.

**SIMD Benefits**
- SIMD naturally supports data parallel applications.
  - Graphics, signal and image processing.
  - Physical simulation.
  - Database queries, financial analytics.
- SIMD has natural advantages over multicore.
  - Cost of instruction fetch/decode divided by SIMD vector length.
  - SIMD ALUs share common control and data path logic.
  - Synchronization of parallel execution is automatic.
### What’s new in AVX-512?

**Several New Instruction Families**

- **AVX-512F**: Foundation - core 32/64 bit operations (extending AVX).
- **AVX-512DQ**: New doubleword/quadword (32/64-bit) operations.
- **AVX-512BW**: AVX-2 byte/word operations extended to 512 bits.
- **AVX-512VBMI**: Full byte-level permutation selecting from 128 bytes.
- Several additional small families of specialized instructions.
What’s new in AVX-512?

Several New Instruction Families

- AVX-512F: Foundation - core 32/64 bit operations (extending AVX).
- AVX-512DQ: New doubleword/quadword (32/64-bit) operations.
- AVX-512BW: AVX-2 byte/word operations extended to 512 bits.
- AVX-512VBMI: Full byte-level permutation selecting from 128 bytes.
- Several additional small families of specialized instructions.

Systematic New Features

- Systematic masking and blending using bitmask registers.
- Constant parameter broadcasting, rounding and exception control.
- Register count increased from 16 to 32.
- Ternary logic - all possible 3-bit Boolean functions.
Opportunity and Challenge

Opportunity

- Extensive SIMD parallelism offers the potential to dramatically speed-up applications.
- The expected speed-up is potentially very large.
- Considerable data rearrangement overhead can be tolerated.
Opportunity and Challenge

Opportunity

- Extensive SIMD parallelism offers the potential to dramatically speed-up applications.
- The expected speed-up is potentially very large.
- Considerable data rearrangement overhead can be tolerated.

Challenge

- Existing sequential programs generally cannot be auto-vectorized.
  - Too many sequential dependencies between data elements.
  - Programmer code optimizations often obscure parallelizable logic.
- Language technology may limit access to SIMD capabilities.
- Text processing may involve variable-length items not well-matched to fixed SIMD field and register widths.
- Data parallel algorithmic approaches may be hard to find.