SSE2

SSE2 (**Streaming SIMD Extensions 2**) is one of the Intel SIMD (Single Instruction, Multiple Data) processor supplementary instruction sets first introduced by Intel with the initial version of the Pentium 4 in 2000. It extends the earlier SSE instruction set, and is intended to fully replace MMX. Intel extended SSE2 to create SSE3 in 2004. SSE2 added 144 new instructions to SSE, which has 70 instructions. Competing chip-maker AMD added support for SSE2 with the introduction of their Opteron and Athlon 64 ranges of AMD64 64-bit CPUs in 2003.

Contents

Features

Most of the SSE2 instructions implement the integer vector operations also found in MMX. Instead of the MMX registers they use the XMM registers, which are wider and allow for significant performance improvements in specialized applications. Another advantage of replacing MMX with SSE2 is avoiding the mode switching penalty for issuing x87 instructions present in MMX because it is sharing register space with the x87 FPU. The SSE2 also complements the floating-point vector operations of the SSE instruction set by adding support for the double precision data type.

Other SSE2 extensions include a set of cache control instructions intended primarily to minimize cache pollution when processing infinite streams of information, and a sophisticated complement of numeric format conversion instructions.

AMD's implementation of SSE2 on the AMD64 (x86-64) platform includes an additional eight registers, doubling the total number to 16 (XMM0 through XMM15). These additional registers are only visible when running in 64-bit mode. Intel adopted these additional registers as part of their support for x86-64 architecture (or in Intel's parlance, "Intel 64") in 2004.

Differences between x87 FPU and SSE2

FPU (x87) instructions provide higher precision by calculating intermediate results with 80 bits of precision, by default, to minimise roundoff error in numerically unstable algorithms (see IEEE 754 design rationale and references therein). However, the x87 FPU is a scalar unit only whereas SSE2 can process a small vector of operands in parallel.

If codes designed for x87 are ported to the lower precision double precision SSE2 floating point, certain combinations of math operations or input datasets can result in measurable numerical deviation, which can be an issue in reproducible scientific computations, e.g. if the calculation results must be compared against results generated from a different machine architecture. A related issue is that, historically, language standards and compilers had been inconsistent in their handling of the x87 80-bit registers implementing double extended precision variables, compared with the double and single precision formats implemented in SSE2: the rounding of extended precision intermediate values to double precision variables was not fully defined and was dependent on implementation details such as when registers were spilled to memory.

Differences between MMX and SSE2

SSE2 extends MMX instructions to operate on XMM registers. Therefore, it is possible to convert all existing MMX code to an SSE2 equivalent. Since an SSE2 register is twice as long as an MMX register, loop counters and memory access may need to be changed to accommodate this. However, 8 byte loads and stores to XMM are available, so this is not strictly required.

Although one SSE2 instruction can operate on twice as much data as an MMX instruction, performance might not increase significantly. Two major reasons are: accessing SSE2 data in memory not aligned to a 16-byte boundary can incur significant penalty, and the throughput of SSE2 instructions in older x86 implementations was half that for MMX instructions. Intel addressed the first problem by adding an instruction in SSE3 to reduce the overhead of accessing unaligned data and improving the overall performance of misaligned loads, and the last problem by widening the execution engine in their Core microarchitecture in Core 2 Duo and later products.

Since MMX and x87 register files alias one another, using MMX will prevent x87 instructions from working as desired. Once MMX has been used, the programmer must use the emms instruction (C: _mm_empty()) to restore operation to the x87 register file. On some operating systems, x87 is not used very much, but may still be used in some critical areas like pow() where the extra precision is needed. In such cases, the corrupt floating-point state caused by failure to emit emms may go undetected for millions of instructions before ultimately causing the floating-point routine to fail, returning NaN. Since the problem is not locally apparent in the MMX code, finding and correcting the bug can be very time consuming. As SSE2 does not have this problem, usually provides much better throughput and provides more registers in 64-bit code, it should be preferred for nearly all vectorization work.

Compiler usage

When first introduced in 2000, SSE2 was not supported by software development tools. For example, to use SSE2 in a Microsoft Visual Studio project, the programmer had to either manually write inline-assembly or import object-code from an external source. Later the Visual C++ Processor Pack added SSE2 support to Visual C++ and MASM.

The Intel C++ Compiler can automatically generate SSE4, SSSE3, SSE3, SSE2, and SSE code without the use of hand-coded assembly.

Since GCC 3, GCC can automatically generate SSE/SSE2 scalar code when the target supports those instructions. Automatic vectorization for SSE/SSE2 has been added since GCC 4.

The Sun Studio Compiler Suite can also generate SSE2 instructions when the compiler flag -xvector=simd is used.

Since Microsoft Visual C++ 2012, the compiler option to generate SSE2 instructions is turned on by default.

CPU support

SSE2 is an extension of the IA-32 architecture, based on the x86 instruction set. Therefore, only x86 processors can include SSE2. The AMD64 architecture supports the IA-32 as a compatibility mode and includes the SSE2 in its specification.[1][2] It also doubles the number of XMM registers, allowing for better performance. SSE2 is also a requirement for installing Windows 8[3] (and later) or Microsoft Office 2013 (and later) "to enhance the reliability of third-party apps and drivers running in Windows 8".[4]

The following IA-32 CPUs support SSE2:

- Intel NetBurst-based CPUs (Pentium 4, Xeon, Celeron, Pentium D, Celeron D)
- Intel Pentium M and Celeron M
- Intel Atom
- AMD Athlon 64
- Transmeta Efficeon
- VIA C7

The following IA-32 CPUs were released after SSE2 was developed, but did not implement it:

- AMD CPUs prior to Athlon 64, such as Athlon XP
- VIA C3
- Transmeta Crusoe
- Intel Quark

See also

- SSE2 instructions

References

1. Matz, Michael; Hubicka, Jan; Jaeger, Andreas; Mitchell, Mark (January 2010). "System V Application Binary Interface - AMD64 Architecture Processor Supplement - Draft Version 0.99.4" (https://www.cs.washington.edu/education/courses/351/12wi/supp-docs/abi.pdf) (PDF). Retrieved April 26, 2013.
2. Fog, Agner. "Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms" (http://www.agner.org/optimize/optimizing_cpp.pdf) (PDF). Retrieved April 26, 2013.
3. "DirectXMath Programming Guide/Library Internals" (https://docs.microsoft.com/en-us/windows/desktop/dxmath/pg-xnamath-internals).
4. Microsoft Corporation. "What is PAE, NX, and SSE2 and why does my PC need to support them to run Windows 8 ?" (https://web.archive.org/web/20130411004411/http://windows.microsoft.com/en-GB/windows-8/what-is-pae-nx-sse2). Archived from the original (http://windows.microsoft.com/en-GB/windows-8/what-is-pae-nx-sse2) on April 11, 2013. Retrieved March 19, 2013.

Retrieved from "https://en.wikipedia.org/w/index.php?title=SSE2&oldid=955547901"

This page was last edited on 8 May 2020, at 12:18 (UTC).