Compilation:

List of all compiler options:
https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/compiler-options/alphabetical-option-list.html

- scalar optimizations: https://www.cs.utexas.edu/~pingali/CS380C/2016/lectures/dataflow.pdf
- loop transformations: https://en.wikipedia.org/wiki/Loop_interchange | https://en.wikipedia.org/wiki/Loop_unrolling
- Cache blocking: https://www.intel.com/content/www/us/en/developer/articles/technical/cache-blocking-techniques.html
- Loop fusion and fission: https://en.wikipedia.org/wiki/Loop_fission_and_fusion
- Cache prefetching: https://en.wikipedia.org/wiki/Cache_prefetching
- Function inlining: https://en.wikipedia.org/wiki/Inline_function
- Constant propagation: https://en.wikipedia.org/wiki/Constant_folding
- Floating-point exceptions: https://www.ibm.com/docs/en/aix/7.2?topic=concepts-floating-point-exceptions
- FMA: https://en.wikipedia.org/wiki/Multiply–accumulate_operation
- SIMD: https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-1-manual.html
  - **MMX**
    Introduce eight 64 bit registers (MM0-MM7) and instructions to work with eight signed/unsigned bytes, four signed/unsigned words, two signed/unsigned dwords.
  - **3DNow!**
    Add support for single precision floating point operand to MMX. Few operation supported, for example addition, subtraction, multiplication.
  - **SSE**
    Introduce eight/sixteen 128 bit registers (XMM0-XMM7/15) and instruction to work with four single precision floating point operands. Add integer operations on MMX registers too. (The MMX-integer part of SSE is sometimes called MMXEXT, and was implemented on a few non-Intel CPUs without xmm registers and the floating point part of SSE.)
  - **SSE2**
    Introduces instruction to work with 2 double precision floating point operands, and with packed byte/word/dword/qword integers in 128-bit xmm registers.
  - **SSE3**
    Add a few varied instructions (mostly floating point), including a special kind of unaligned load (lddqu) that was better on Pentium 4, synchronization instruction, horizontal add/sub.

- **SSSE3**
  Again a varied set of instructions, mostly integer. The first shuffle that takes its control operand from a register instead of hard-coded (pshufb). More horizontal processing, shuffle, packing/unpacking, mul+add on bytes, and some specialized integer add/mul stuff.
- **SSE4 (SSE4.1, SSE4.2)**
  Add a lot of instructions: Filling in a lot of the gaps by providing min and max and other operations for all integer data types (especially 32-bit integer had been lacking), where previously integer min was only available for unsigned bytes and signed 16-bit. Also scaling, FP rounding, blending, linear algebra operation, text processing, comparisons. Also a non temporal load for reading video memory, or copying it back to main memory. (Previously only NT stores were available.)
- **AESNI**
  Add support for accelerating AES symmetric encryption/decryption.
- **AVX** Add eight/sixteen 256 bit registers (YMM0-YMM7/15).
  Support all previous floating point datatype. Three operand instructions.
- **FMA**
  Add Fused Multiply Add and correlated instructions.
- **AVX2**
  Add support for integer data types.
- **AVX512F**
  Add eight/thirty-two 512 bit registers (ZMM0-ZMM7/31) and eight 64-bit mask register (k0-k7). Promote most previous instruction to 512 bit wide. Optional parts of AVX512 add instruction for exponentials & reciprocals (AVX512ER), scatter/gather prefetching (AVX512PF), scatter conflict detection (AVX512CD), compress, expand.
- **IMCI (Intel Xeon Phi)**
  Early development of AVX512 for the first-gen Intel Xeon Phi (Knight's Corner) coprocessor.
- -ax = https://community.intel.com/t5/Intel-C-Compiler/Full-optimization-of-binaries-vs-ax-options/td-p/1077821
- Xhost = https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/compiler-options/code-generation-options/xhost-qxhost.html

- SPECint: https://en.wikipedia.org/wiki/SPECint
- SPECfp: https://en.wikipedia.org/wiki/SPECfp
- SPEC HPC: https://en.wikipedia.org/wiki/Standard_Performance_Evaluation_Corporation

- Loop dependence analysis: https://en.wikipedia.org/wiki/Loop_dependence_analysis
- Aliasing: https://en.wikipedia.org/wiki/Aliasing_(computing)
- Ivdep: https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/pragmas/intel-specific-pragma-reference/ivdep.html
- Sin vs sinf: sin takes a double and returns a double - sinf takes a float and returns a float. In other words sin is double precision and sinf is single precision.
- Cache inclusion policy: https://en.wikipedia.org/wiki/Cache_inclusion_policy
- SIMD Data Layout Templates (SDLT): https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top/compiler-reference/libraries/simd-data-layout-templates.html
- Clang: https://en.wikipedia.org/wiki/Clang
- LLVM: https://en.wikipedia.org/wiki/LLVM