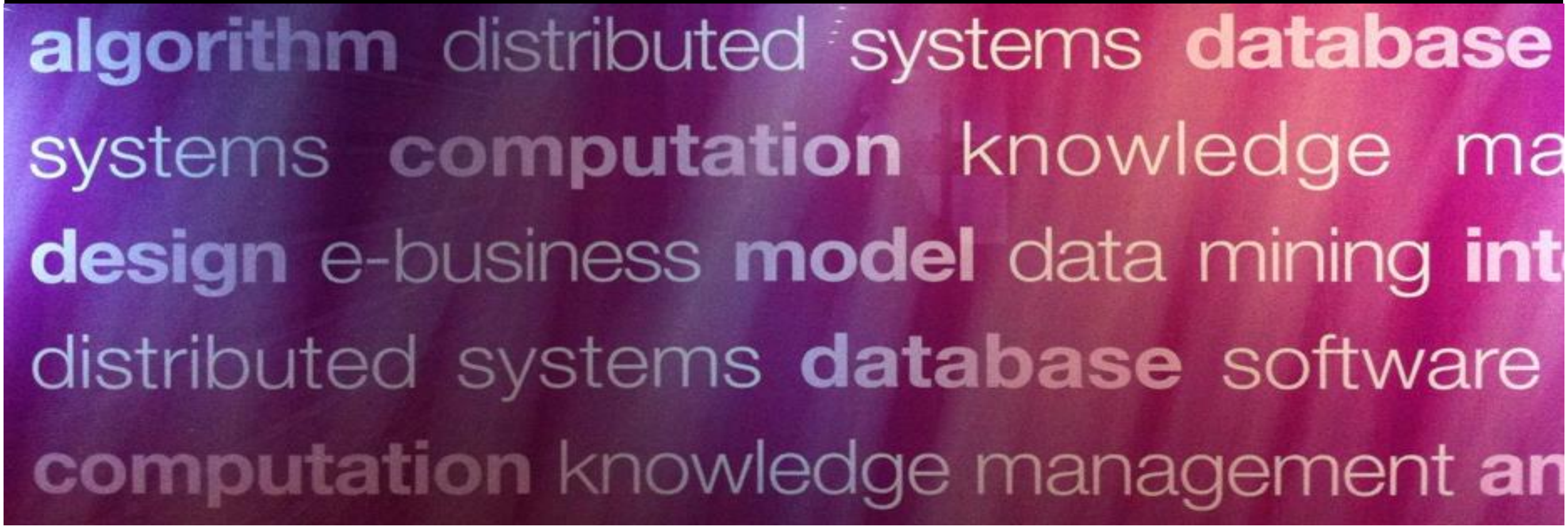
MONASH University

Information Technology

# FIT3143 - LECTURE WEEK 10

SIMD AND DATA PARALLEL ARCHITECTURES


algorithm distributed systems database
systems computation knowledge ma
design e-business model data mining int
distributed systems database software
computation knowledge management an

# Topic Overview

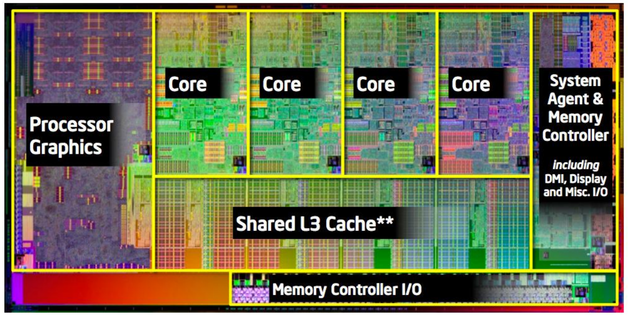
- Flynn’s Taxonomy
- Definition of SIMD
- Streaming SIMD Extensions (SSE)
- SSE vs MMX
- SSE, SSE2, SSE3, SSE4 & AVX

## Learning outcome(s) related to this topic

- **Explain the fundamental principles of parallel computing architectures and algorithms (LO1)**

- **Design and develop parallel algorithms for various parallel computing architectures (LO3)**

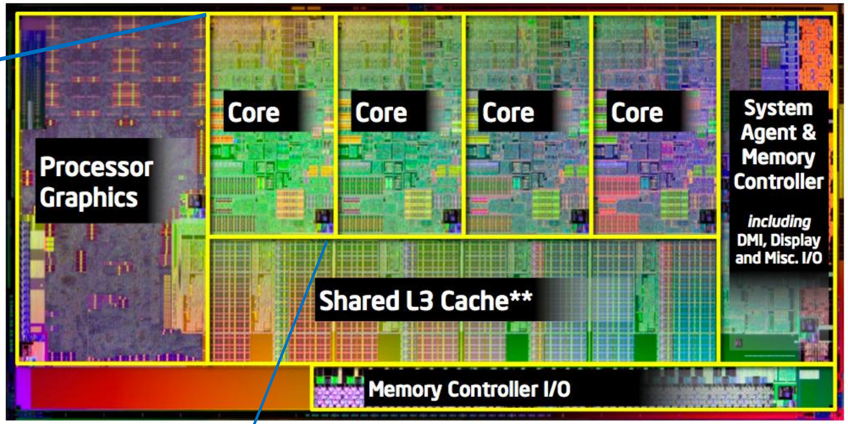
# Quick Recap of Parallel Computing Thus Far

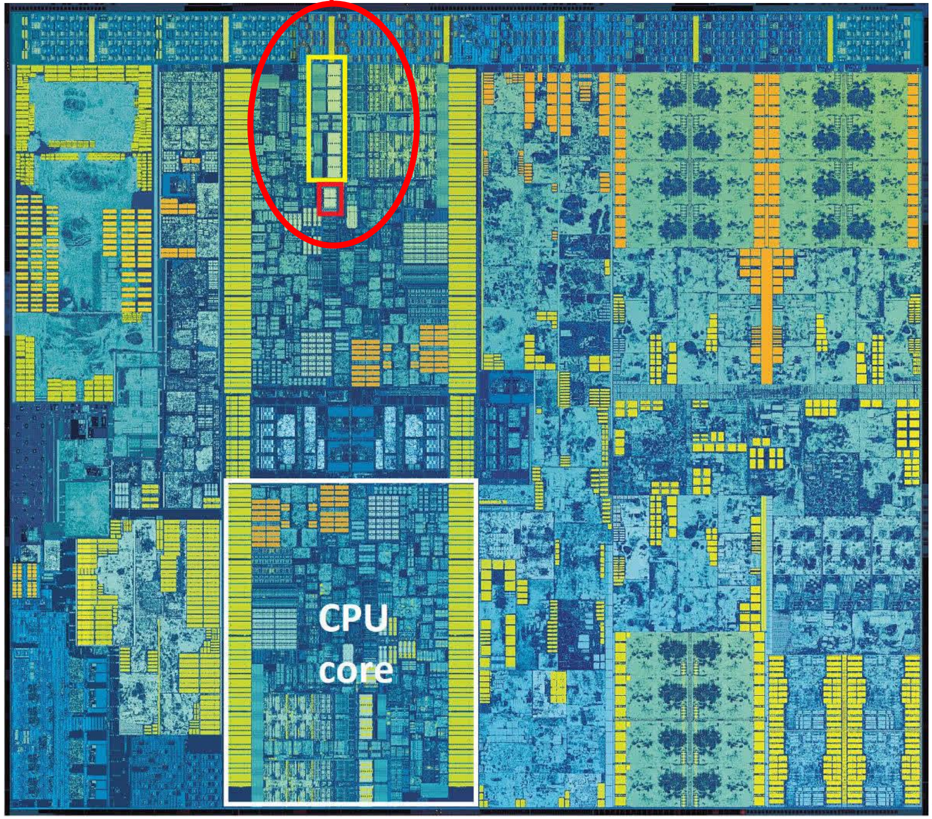


- Principles of parallel computing
- Parallel computing methods
  - Shared Memory – POSIX & OpenMP
  - Distributed Memory – Message Passing Interface
- Clock synchronization
- Deadlock detection and avoidance
- Fault detection
- Advanced Message Passing Interface techniques
  - Scatter & Gather
  - Virtual topologies
- Data parallelism design
  - Simple partitioning strategies
  - Block or tile-based partitioning strategies
  - Fox & cannon techniques for matrix multiplication

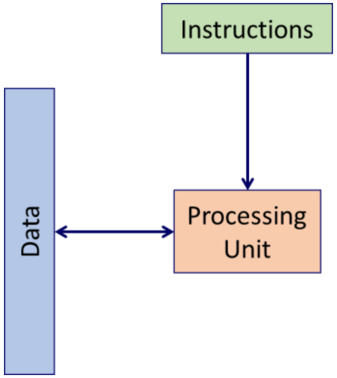
# Parallel computing is also applied deeper inside a core chip!

Additional registers to increase parallelism in vector operations

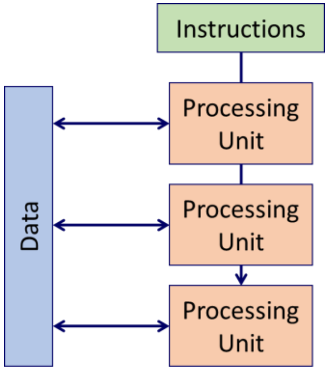




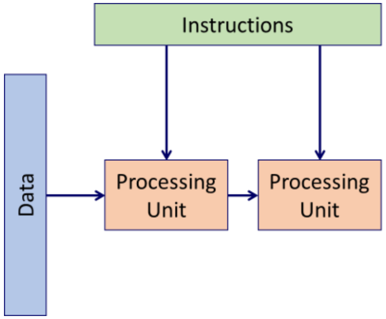
# Flynn Taxonomy (1966)



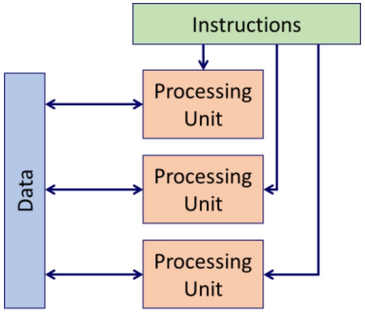
**Single-Instruction**
**Single-Data**
**(Single-Core Processors)**



**Single-Instruction**
**Multi-Data**
**(GPUs, Intel SIMD Extensions)**



**Multi-Instruction**
**Single-Data**
**(Systolic Arrays,…)**



**Multi-Instruction**
**Multi-Data**
**(Parallel Computers)**

# Single Instruction, Multiple Data (SIMD)

- SIMD (vector instructions) is a technique employed to achieve data level parallelism.
- Advantages:
  - **Multiple data** can be loaded at once.
  - **Operations** can be applied to all of the data in one operation.
- Disadvantages:
  - Not all operations can be done with SIMD.
  - Implementation requires human labor in terms of coding (most compilers don't generate SIMD instructions).
  - Programming with particular SIMD instruction sets can involve numerous low-level challenges.

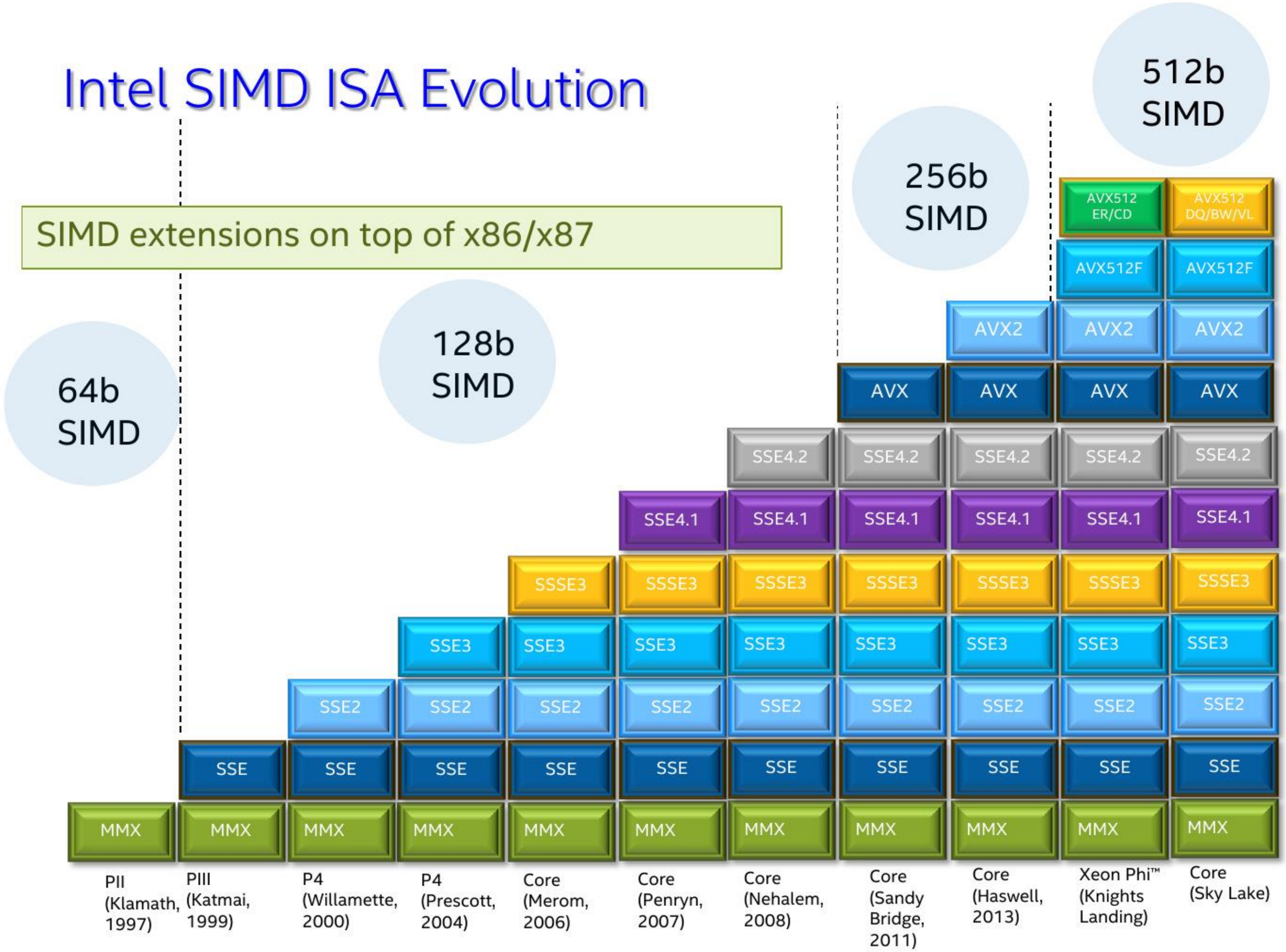
# Intel SIMD Extensions

- Streaming SIMD Extensions (SSE)
  - Single instruction, multiple data (SIMD) instruction set extension to the x86 architecture
- New instructions, new registers
- Introduced in phases/groups of functionality
  - SSE – SSE4 (1999 –2006)
    - 128 bit width operations
  - AVX, FMA, AVX2, AVX-512 (2008 – 2015)
    - 256 – 512 bit width operations

# What about Intel’s MultiMedia eXtension (MMX)?

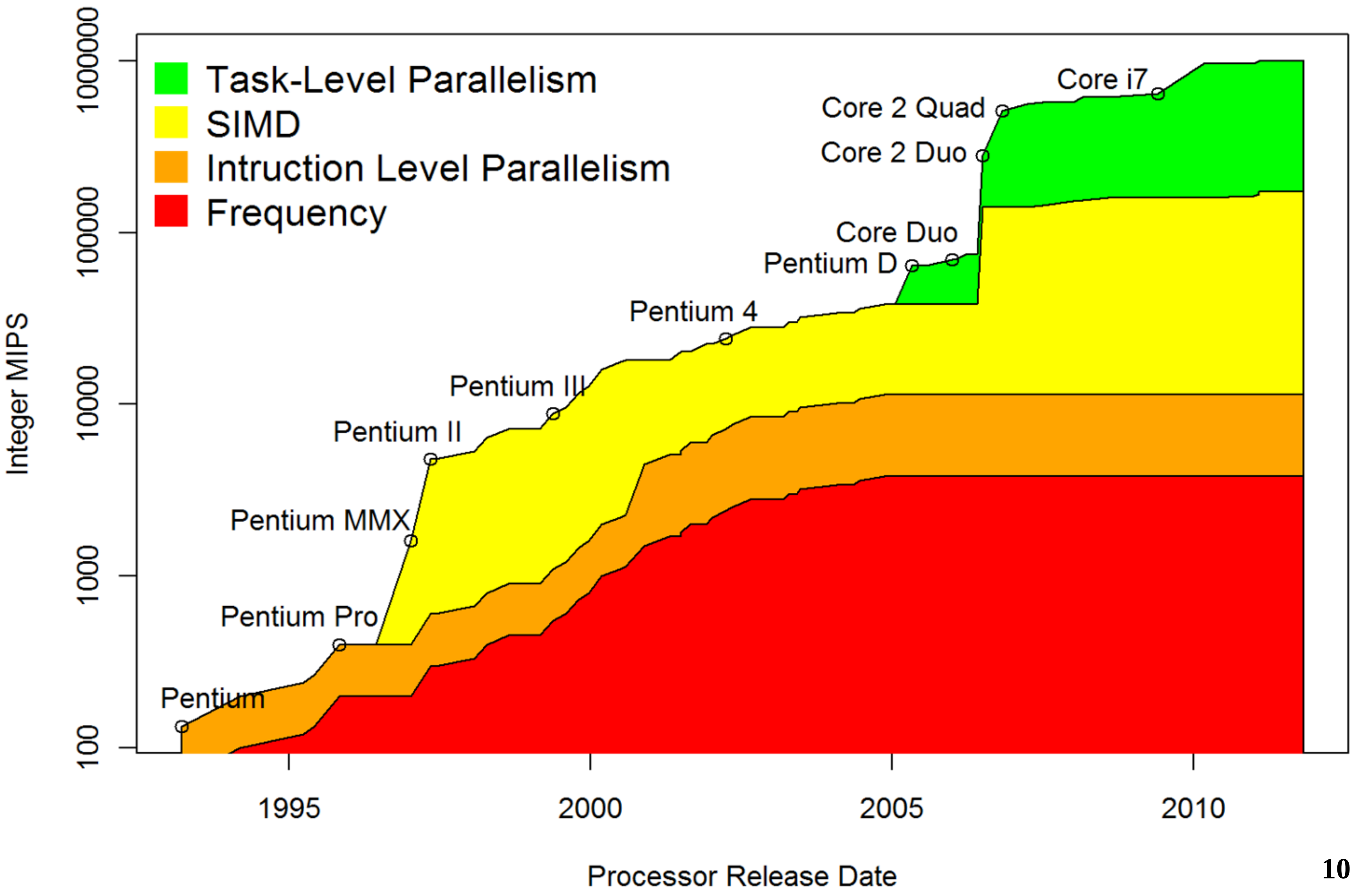
- Intel MMX technology introduced single-instruction multiple-data (SIMD) capability into the IA-32 architecture:
  - ➢ 64-bit MMX registers (MM0 through MM7)
  - ➢ 64-bit packed integer data types
  - ➢ New instructions to perform SIMD operations on packed integers
- These new "registers" were just aliases for the existing x87 FPU stack registers.
- Thus, any changes to the floating point stack would also affect the MMX registers.
- This made it difficult to work with floating point and SIMD data at the same time.
- Another problem for MMX is that it only provides integer operations.

# Intel SIMD ISA Evolution



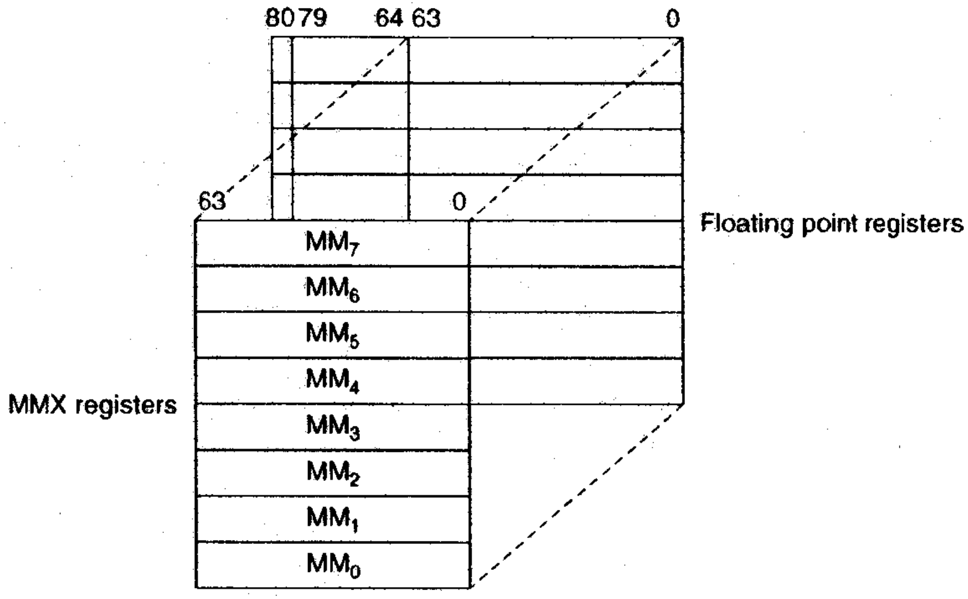
| PII (Klamath, 1997) | PIII (Katmai, 1999) | P4 (Willamette, 2000) | P4 (Prescott, 2004) | Core (Merom, 2006) | Core (Penryn, 2007) | Core (Nehalem, 2008) | Core (Sandy Bridge, 2011) | Core (Haswell, 2013) | Xeon Phi™ (Knights Landing) | Core (Sky Lake) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | AVX512 ER/CD | AVX512 DQ/BW/VL |
| | | | | | | | | | AVX512F | AVX512F |
| | | | | | | | | AVX2 | AVX2 | AVX2 |
| | | | | | | | AVX | AVX | AVX | AVX |
| | | | | | | SSE4.2 | SSE4.2 | SSE4.2 | SSE4.2 | SSE4.2 |
| | | | | | SSE4.1 | SSE4.1 | SSE4.1 | SSE4.1 | SSE4.1 | SSE4.1 |
| | | | | SSSE3 | SSSE3 | SSSE3 | SSSE3 | SSSE3 | SSSE3 | SSSE3 |
| | | | SSE3 | SSE3 | SSE3 | SSE3 | SSE3 | SSE3 | SSE3 | SSE3 |
| | | SSE2 | SSE2 | SSE2 | SSE2 | SSE2 | SSE2 | SSE2 | SSE2 | SSE2 |
| | SSE | SSE | SSE | SSE | SSE | SSE | SSE | SSE | SSE | SSE |
| MMX | MMX | MMX | MMX | MMX | MMX | MMX | MMX | MMX | MMX | MMX |

## Components of peak performance of the top Intel x86 desktop CPU



# Multimedia Register File

- Eight 64-bit registers are provided for storage / hold of (source and destination) operands processed by MMX inst.
  - They are not NEW, but are aliases with the existing floating-point register stack



# Streaming SIMD Extensions (SSE)

- Intel addressed the shortcomings of the MMX technology through SSE, a greatly expanded set of SIMD instructions with 32-bit floating point support.
- Designed by Intel and introduced in 1999 in their Pentium III series processors.
- SSE added eight new 128-bit registers known as XMM0 through XMM7, which made it easy to perform SIMD and FPU operations at the same time
- The x86-64 extensions (AMD64 and Intel 64) add a further eight registers XMM8 through XMM15.

# SSE Features

- SSE adds the following features to the IA-32 architecture, while maintaining backward compatibility with all existing IA-32 processors:
  - Eight 128-bit data registers (XMM registers) in non-64-bit modes; sixteen XMM registers in 64-bit mode.
  - 32-bit MXCSR register, which provides control and status bits for operations performed on XMM registers.
  - 128-bit packed single-precision floating-point data type (four IEEE single-precision floating-point values packed into a double quad-word).
  - Instructions that perform SIMD operations on single-precision floating-point values and that extend SIMD operations that can be performed on integers:
    - 128-bit Packed and scalar single-precision floating-point instructions that operate on data located in XMM registers.
    - 64-bit SIMD integer instructions that support additional operations on packed integer operands located in MMX registers.

  - Instructions that save and restore the state of the MXCSR register.
  - Instructions that support explicit prefetching of data, control of the cacheability of data, and control the ordering of store operations.
  - Extensions to CPUID instruction.
- These features extend the IA-32 architecture’s SIMD programming model in the following ways:
  - The ability to perform SIMD operations on four packed single-precision floating-point values enhances the performance of IA-32 processors for advanced media and communications applications that use computation-intensive algorithms to perform repetitive operations on large arrays of simple, native data elements.
  - The ability to perform SIMD single-precision floating-point operations in XMM registers and SIMD integer operations in MMX registers provides greater flexibility and throughput for executing applications that operate on large arrays of floating-point and integer data.

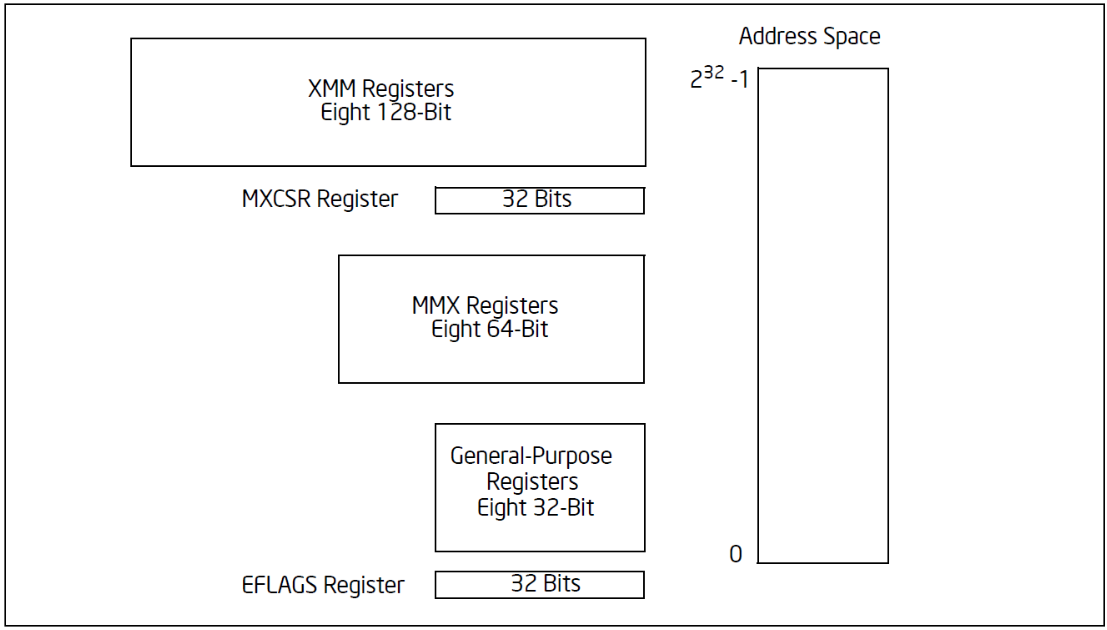
  - Cache control instructions provide the ability to stream data in and out of XMM registers without polluting the caches and the ability to prefetch data to selected cache levels before it is actually used; applications that require regular access to large amounts of data benefit from these prefetching and streaming store capabilities.

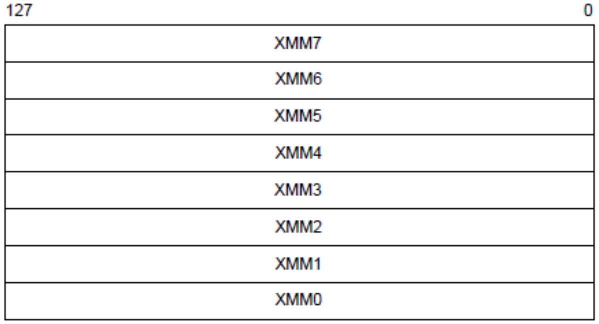
- SSE extensions are fully compatible with all software written for IA-32 processors.
- All existing software continues to run correctly, without modification, on processors that incorporate SSE extensions.
- Enhancements to CPUID permit detection of SSE extensions.
- SSE extensions are accessible from all IA-32 execution modes: protected mode, real address mode, and virtual-8086 mode.

# SSE Programming Environment

- **XMM registers** – Eight 128-bit registers used to operate on packed or scalar single-precision floating-point data.
- **MXCSR register** – 32-bit register provides status and control bits used in SIMD floating-point operations.
- **MMX registers** – Eight 64-bit registers used to perform

data.





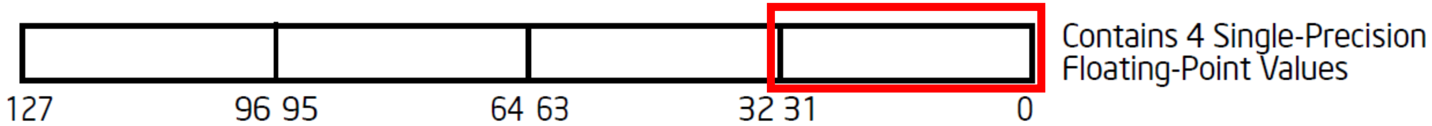
- As the 128-bit XMM registers are additional program states that the operating system must preserve across task switches, they are **disabled by default** until the operating system explicitly enables them.
- This means that the OS must know how to use the FXSAVE and FXRSTOR instructions, which is the extended pair of instructions to save all x87 and SSE register states all at once.
- This support has been added to all major IA-32 operating systems.

# Advantages of SSE over MMX

- Unlike SSE, the eight ‘new’ registers (MM0 through MM7) in MMX are just aliases for the existing x87 FPU.
- Thus, the CPU is unable to work on both floating point and MMX data at the same time.
- MMX only works on integers, while SSE supports both integers and floating point data.
- Hence, SSE is more flexible and has much more use than MMX.

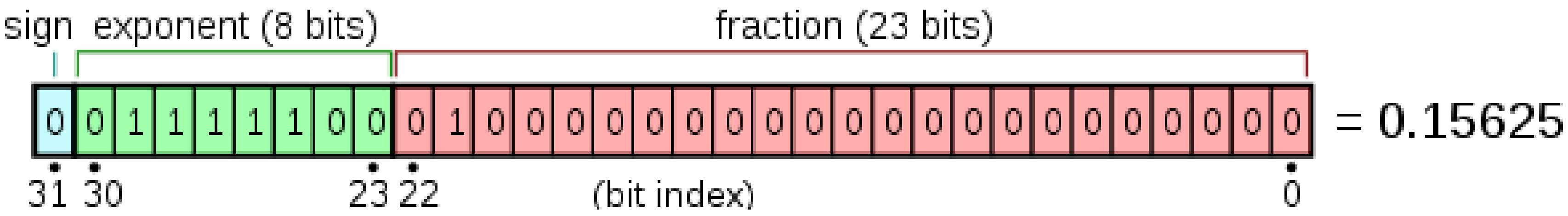
# SSE New Data Type

- SSE extensions introduced one data type, the 128-bit packed single-precision floating-point data type, to the IA-32 architecture.



- This data type consists of four IEEE 32-bit single-precision floating-point values packed into a double quad-word.
- This 128-bit packed single-precision floating-point data type is operated on in the XMM registers or in memory.
- Conversion instructions are provided to convert two packed single-precision floating-point values into two packed double-word integers or a scalar single-precision floating-point value into a double-word integer.

# Single-precision floating-point format



# SSE Instruction Set

- SSE instructions (70 new instructions) are divided into four functional groups:

  i. Packed and scalar single-precision floating-point instructions

  ii. 64-bit SIMD integer instructions

  iii. State management instructions

  iv. Cacheability control, prefetch, and memory ordering instructions

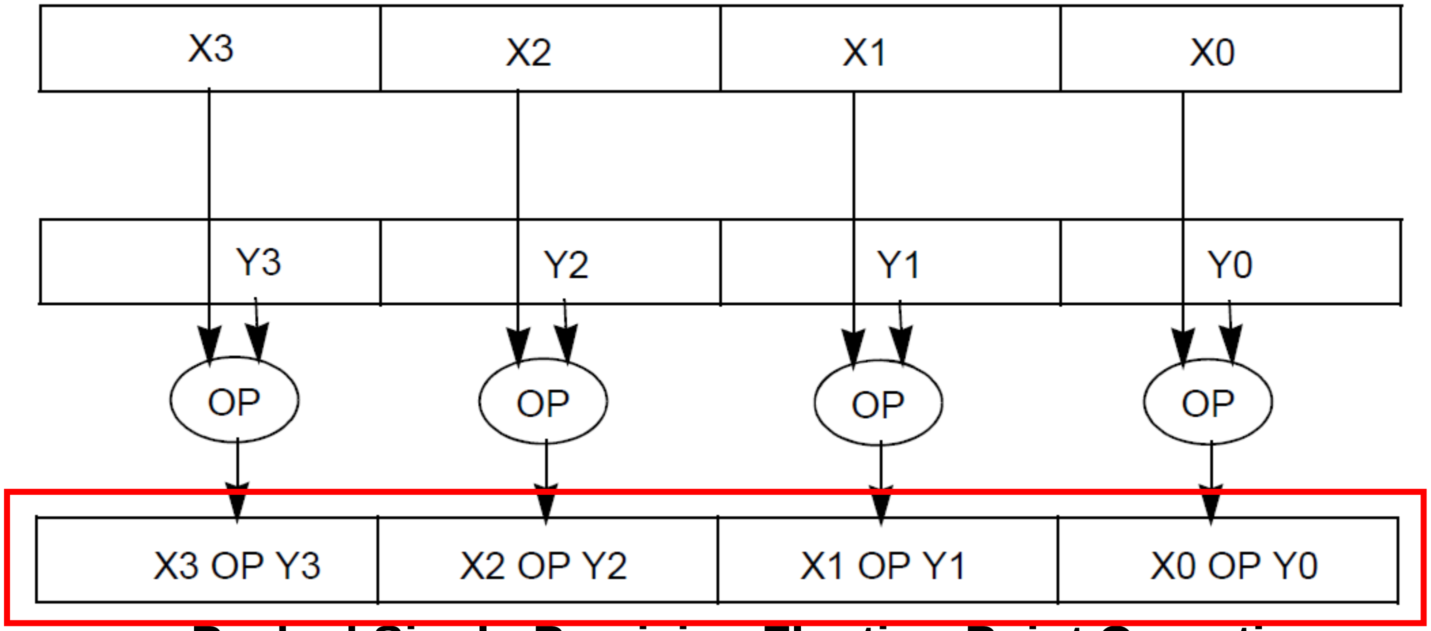
*This chapter restricts the discussion to data movement and arithmetic instructions within the first functional group only. Please refer to "Intel® 64 and IA-32 ArchitecturesSoftware Developer's Manual Volume 1: Basic Architecture" for other instructions.*

http://www.intel.com/products/processor/manuals/

# SSE Packed & Scalar Floating-point Instructions

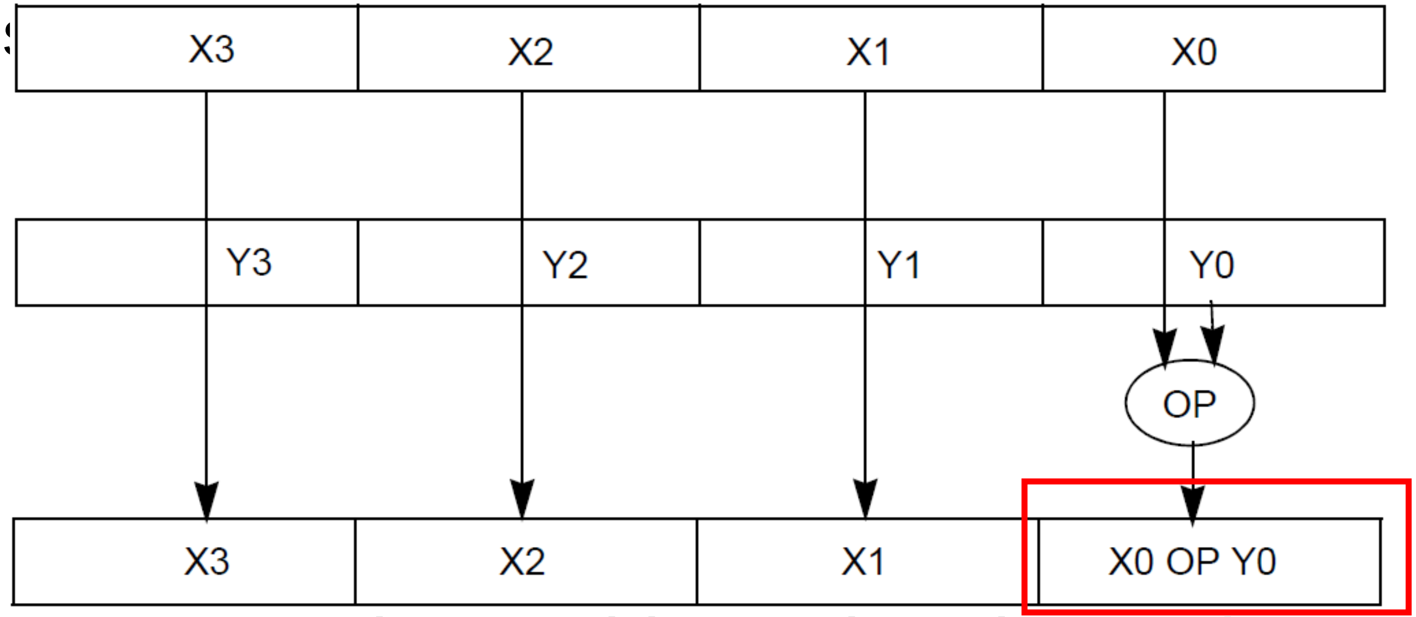
- The packed and scalar single-precision floating-point instructions are divided into the following subgroups:
  - Data movement instructions
  - Arithmetic instructions
  - Logical instructions
  - Comparison instructions
  - Shuffle instructions
  - Conversion instructions
- The packed single-precision floating-point instructions perform SIMD operations on packed single-precision floating-point operands.

- In the SSE **packed** instructions, each source operand contains four single-precision floating-point values, and the destination operand contains the results of the operation performed in parallel on the corresponding values (X0 and Y0, X1 and Y1, X2 and Y2, and X3 and Y3) in each operand.



**Packed Single-Precision Floating-Point Operation**

- The **scalar** single-precision floating-point instructions operate on the low (least significant) doublewords of the two source operands (X0 and Y0).
- The three most significant double-words (X1, X2, and X3) of the first source operand are passed through to the de



**Scalar Single-Precision Floating-Point Operation**

# SSE2

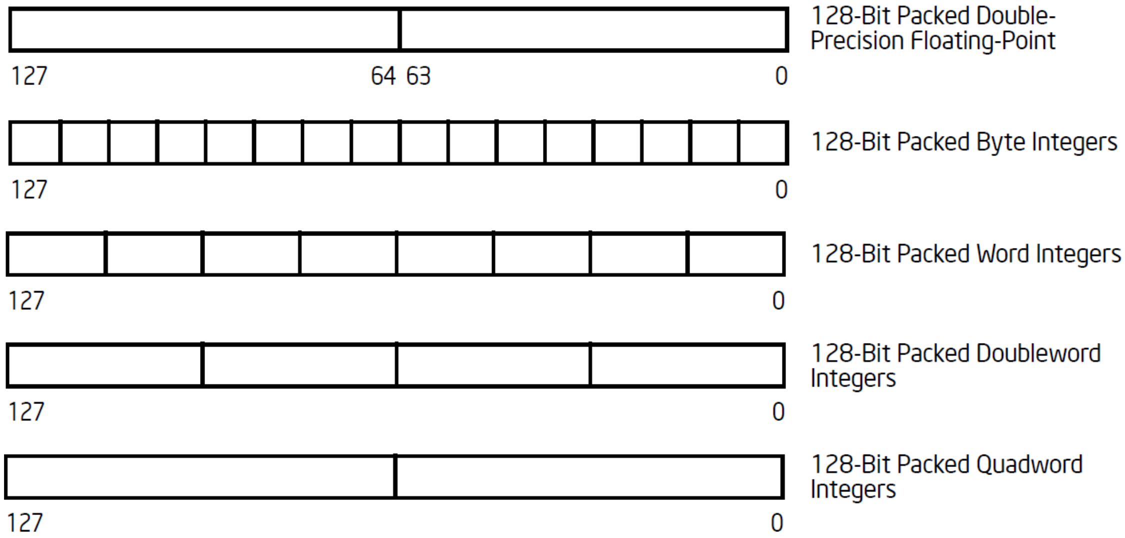
- The streaming SIMD extensions 2 (SSE2) were introduced into the IA-32 architecture in the Pentium 4 and Intel Xeon processors.
- SSE2 adds new math instructions for double-precision (64-bit) floating point and also extends MMX instructions to operate on 128-bit XMM registers.
- SSE2 enables the programmer to perform SIMD math on virtually any data type (from 8-bit integer to 64-bit float) entirely with the XMM registers, without the need to use the (legacy) MMX/FPU registers.

- SSE2 extensions add the following features to the IA-32 architecture:
  - **Five data types**:
    - 128-bit packed double-precision floating-point
    - 128-bit packed byte integers
    - 128-bit packed word integers
    - 128-bit packed doubleword integers
    - 128-bit packed quadword integers
  - **Instructions** to support the additional data types and extend existing SIMD integer operations:
    - Packed and scalar double-precision floating-point instructions
    - Additional 64-bit and 128-bit SIMD integer instructions
    - 128-bit versions of SIMD integer instructions introduced with the MMX technology and the SSE extensions
    - Additional cacheability-control and instruction-ordering instructions.

- These new features extend the IA-32 architecture’s SIMD programming model in three important ways:
  - They provide the ability to perform SIMD operations on pairs of packed double-precision floating-point values.
  - This permits higher precision computations to be carried out in XMM registers, which enhances processor performance in scientific and engineering applications and in applications that use advanced 3-D geometry techniques.
  - Additional flexibility is provided with instructions that operate on single (scalar) double-precision floating-point values located in the low quad-word of an XMM register.
  - They provide the ability to operate on 128-bit packed integers (bytes, words, double-words, and quad-words) in XMM registers.
  - This provides greater flexibility and greater throughput when performing SIMD operations on packed integers. The capability is particularly useful for applications such as authentication and encryption.
  - Using the full set of SIMD registers, data types, and instructions provided with the MMX technology and SSE/SSE2 extensions, programmers can develop algorithms that finely mix packed single- and double-precision floating-point data and 64- and 128-bit packed integer data.
  - SSE2 extensions enhance the support introduced with SSE extensions for controlling the cacheability of SIMD data.

# SSE2 Data Types

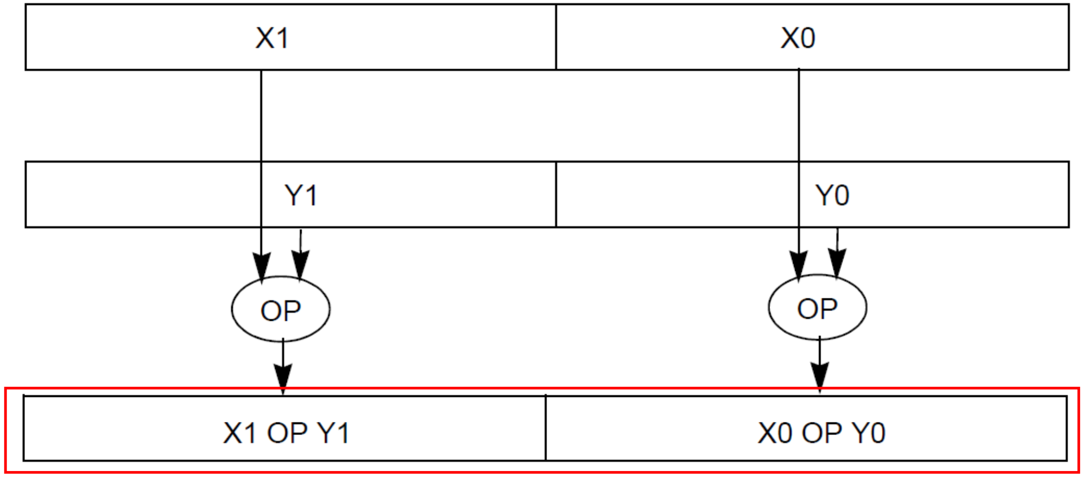
- SSE2 extensions introduced:
  - **Packed double-precision floating-point** — This 128-bit data type consists of two IEEE 64-bit double-precision floating-point values packed into a double quad-word.
  - **128-bit packed integers** — The four 128-bit packed integer data types can contain 16 byte integers, 8 word integers, 4 double-word integers, or 2 quad-word integers.


128-Bit Packed Double-Precision Floating-Point
127
64 63
0
128-Bit Packed Byte Integers
127
0
128-Bit Packed Word Integers
127
0
128-Bit Packed Doubleword Integers
127
0
128-Bit Packed Quadword Integers
127
0

**Data Types Introduced with the SSE2 Extensions**

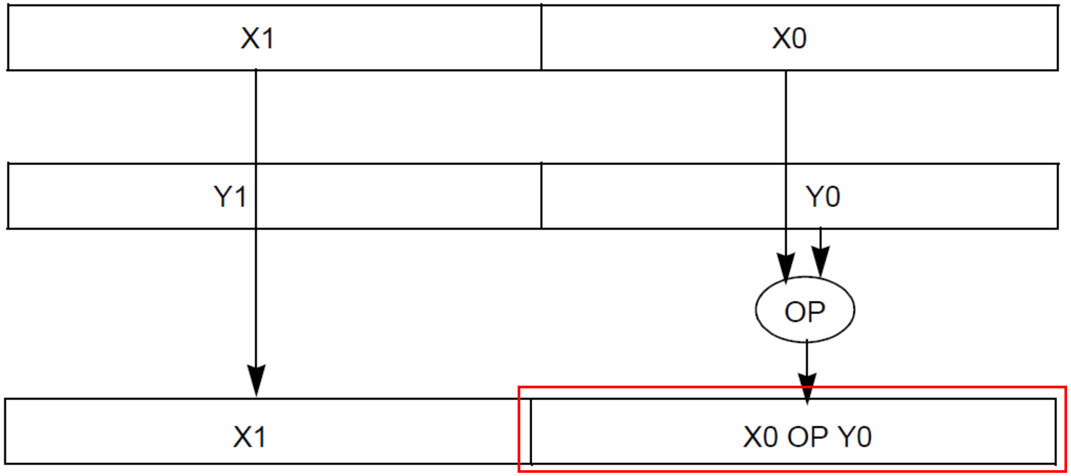
# SSE2 Packed & Scalar Double-Precision Floating-Point Instructions

- The **packed** double-precision floating-point instructions perform SIMD operations similarly to the packed single-precision floating-point instructions.
- Each source operand contains two double-precision floating-point values, and the destination operand contains the results of the operation performed in parallel on the corresponding values (X0 and Y0, and X1 and Y1) in each operand



**Packed Double-Precision Floating-Point Operations**

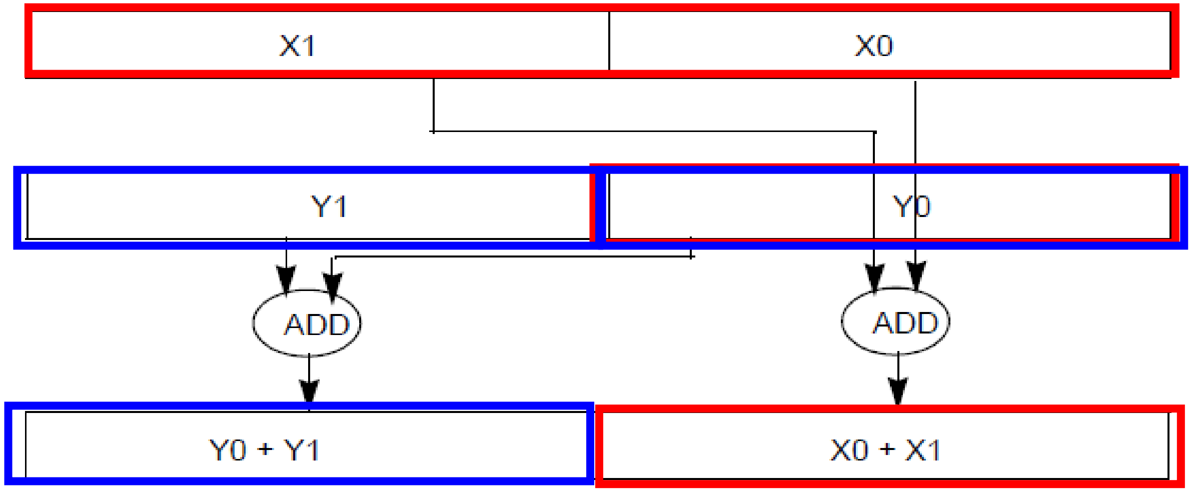
- The **scalar** double-precision floating-point instructions operate on the low (least significant) quad-words of two source operands (X0 and Y0).
- The high quad-word (X1) of the first source operand is passed through to the destination.



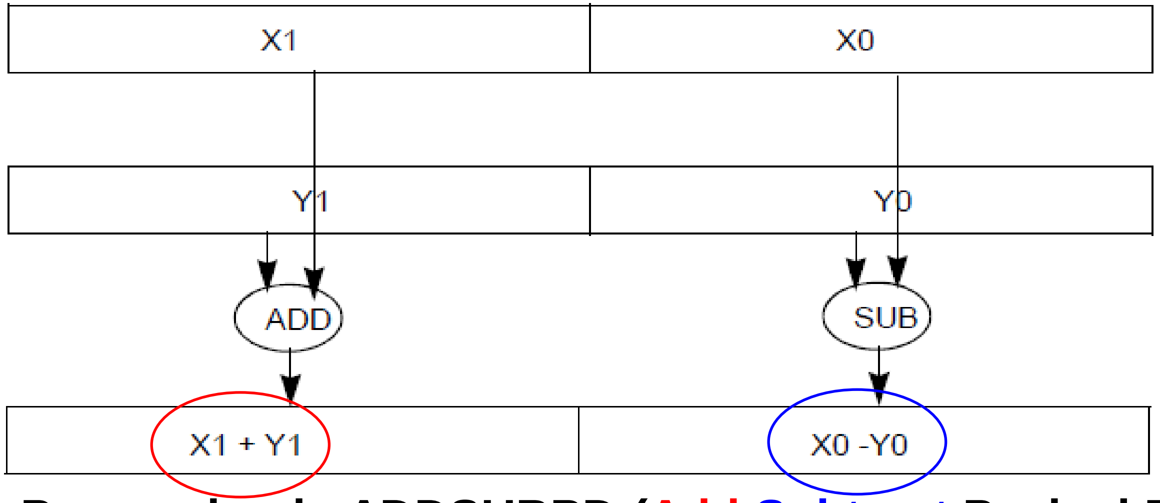
**Scalar Double-Precision Floating-Point Operations**

# SSE3

- The Pentium 4 processor supporting Hyper-Threading Technology (HT Technology) introduces Streaming SIMD Extensions 3 (SSE3).
- 13 new instructions were introduced with SSE3.
- SSE3 did not introduce new data type and programming model.
- Many of the previous SSE/SSE2 instructions accelerate SIMD data processing using a model referred to as vertical computation.
- The most notable change in SSE3 is its capability to work **horizontally** in a register, as opposed to the more or less strictly vertical operation of all previous SSE instructions.
- SSE3 also added instructions to perform different operations on the SIMD data, e.g. subtraction for the 1st pair of data and addition for the 2nd pair (asymmetric processing).



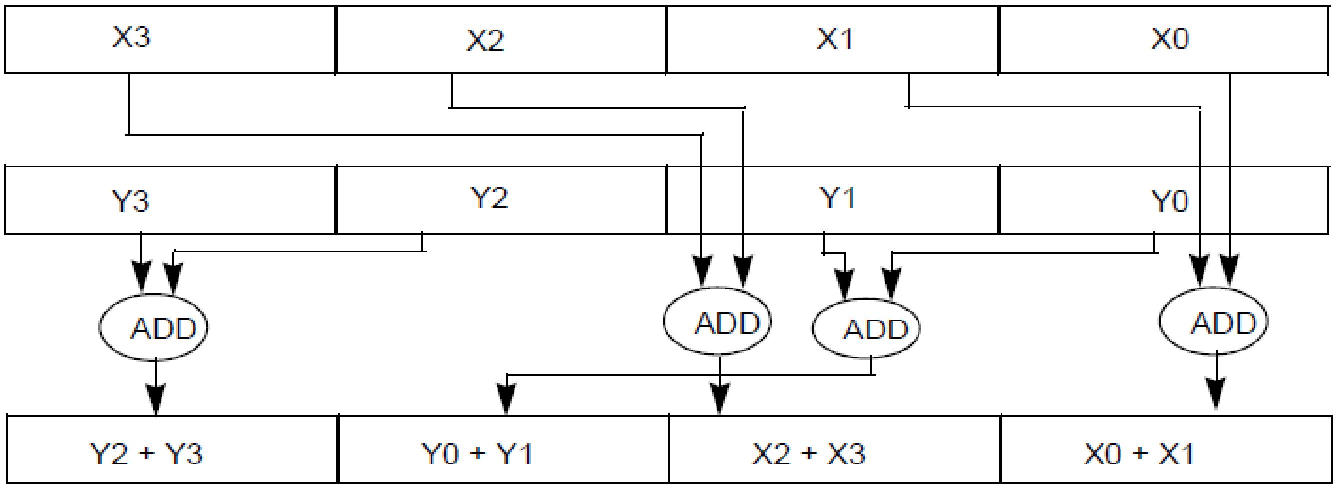
**Horizontal Computation in HADDPD (Horizontal Add Packed Double Precision)**



**Asymmetric Processing in ADDSUBPD (Add Subtract Packed Double Precision)**

# Supplemental SSE3 (SSSE3)

- The Intel Xeon processor 5100 series, Intel Core 2 processor families introduced Supplemental Streaming SIMD Extensions 3 (SSSE3), which is SSSE3 was an incremental upgrade to SSE3.
- Intel have added an S rather than increment the version number, as they appear to consider it merely a revision of SSE3.
- SSSE3 provides 32 instructions to accelerate a variety of multimedia and signal processing applications employing SIMD integer data.
- In analogy to the packed, floating-point horizontal add and subtract instructions in SSE3, SSSE3 offers similar capabilities on packed integer data.



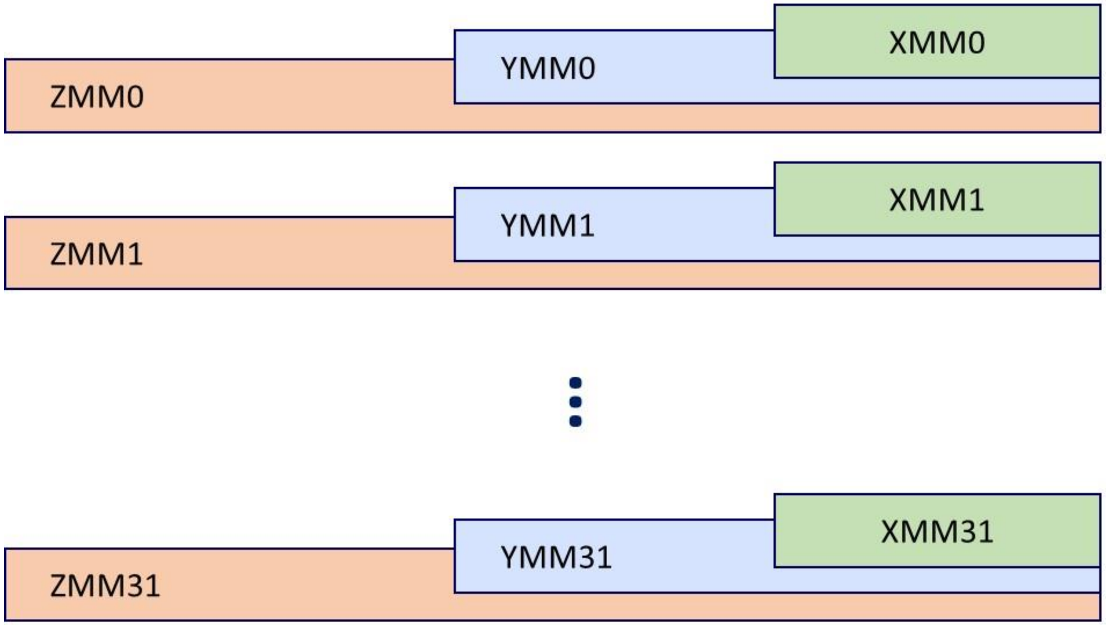
**Horizontal computation in PHADDD (Packed Horizonal Add Doublewords)**

# SSE4

- SSE4 are introduced in Intel processor generations built from 45nm process technology (Penryn and Core i7).
- SSE4 comprises of two sets of extensions: SSE4.1 & SSE4.2.
- SSE4 does not introduce any new data types.
- 47 instructions is available in SSE4.1 (available in Penryn).
- SSE4.1 adds instructions that improve compiler vectorization and significantly increase support for packed double-word computation.
- SSE4.2 consisting of the 7 remaining instructions, is first available in Core i7 (formerly Nehalem).
- SSE4.2 instructions improve performance in string and text processing that can take advantage of single-instruction multiple-data programming techniques.

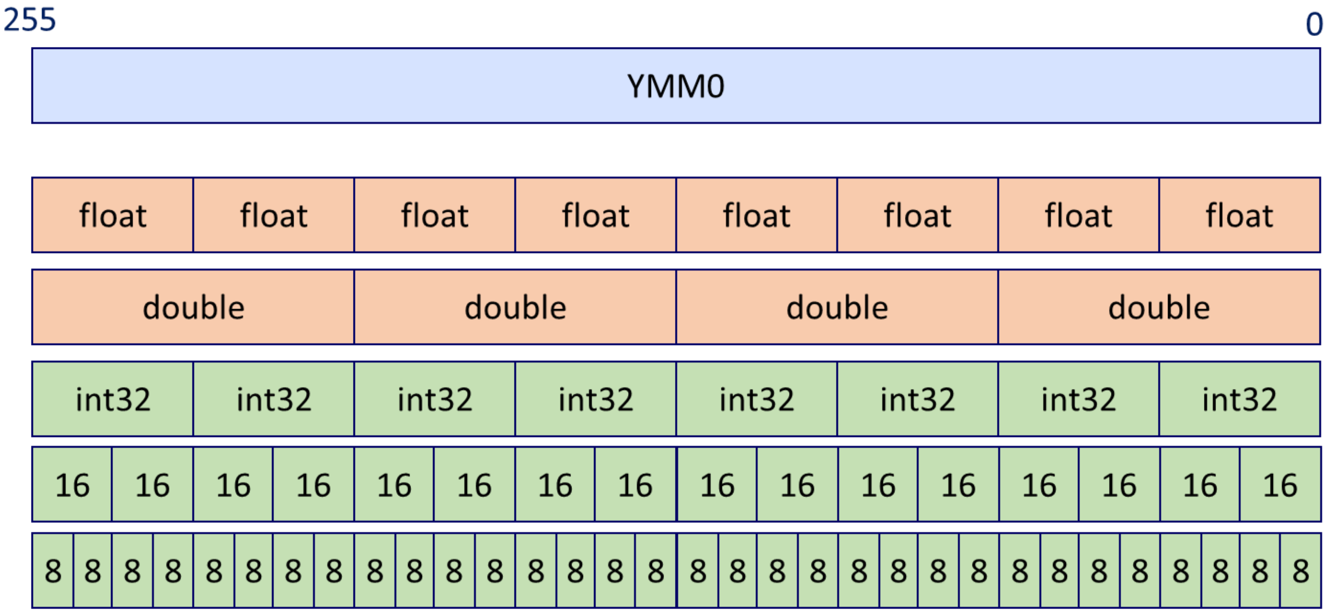
- SSE4 does not support operations on 64-bit MMX registers; SIMD integer operations can be carried out on 128-bit XMM registers only.
- Next - AVX (Advanced Vector Extensions) is an advanced version of SSE announced by Intel featuring a widened data path from 128 bits to 256 bits and 3-operand instructions (up from 2).

# Intel SIMD Registers (AVX-512)



- ❑ XMM0 – XMM15
  - ○ 128-bit registers
  - ○ SSE
- ❑ YMM0 – YMM15
  - ○ 256-bit registers
  - ○ AVX, AVX2
- ❑ ZMM0 – ZMM31
  - ○ 512-bit registers
  - ○ AVX-512

# SSE/AVX Data Types



255 0

YMM0

float float float float float float float float

double double double double

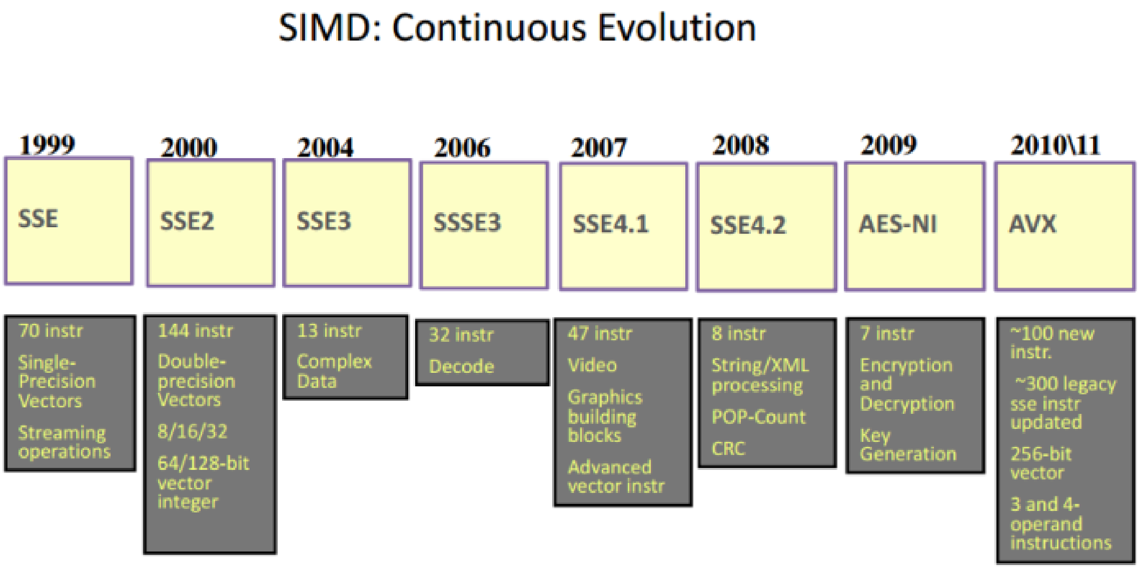
int32 int32 int32 int32 int32 int32 int32 int32

16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16

8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

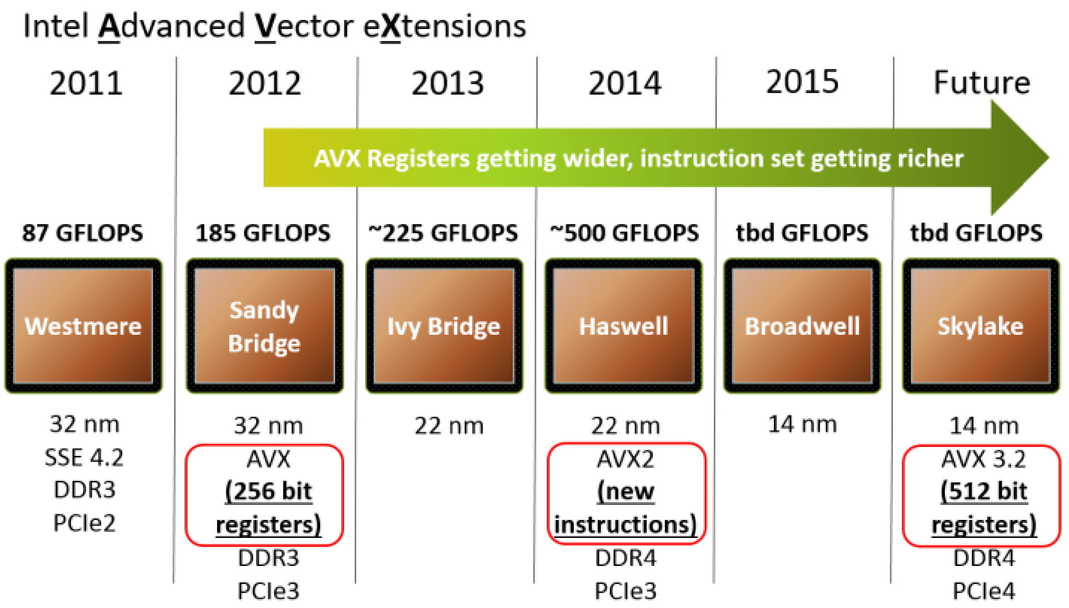
Operation on
32 8-bit values
in one instruction!

# Evolution Summary of SSE & its future



SIMD: Continuous Evolution

| 1999 | 2000 | 2004 | 2006 | 2007 | 2008 | 2009 | 2010\11 |
|---|---|---|---|---|---|---|---|
| SSE | SSE2 | SSE3 | SSSE3 | SSE4.1 | SSE4.2 | AES-NI | AVX |
| 70 instr<br>Single-Precision Vectors<br>Streaming operations | 144 instr<br>Double-precision Vectors<br>8/16/32<br>64/128-bit vector integer | 13 instr<br>Complex Data | 32 instr<br>Decode | 47 instr<br>Video<br>Graphics building blocks<br>Advanced vector instr | 8 instr<br>String/XML processing<br>POP-Count<br>CRC | 7 instr<br>Encryption and Decryption<br>Key Generation | ~100 new instr.<br>~300 legacy sse instr updated<br>256-bit vector<br>3 and 4-operand instructions |



Intel Advanced Vector eXtensions

AVX Registers getting wider, instruction set getting richer

| 2011 | 2012 | 2013 | 2014 | 2015 | Future |
|---|---|---|---|---|---|
| 87 GFLOPS | 185 GFLOPS | ~225 GFLOPS | ~500 GFLOPS | tbd GFLOPS | tbd GFLOPS |
| Westmere | Sandy Bridge | Ivy Bridge | Haswell | Broadwell | Skylake |
| 32 nm<br>SSE 4.2<br>DDR3<br>PCIe2 | 32 nm<br>AVX<br>(256 bit registers)<br>DDR3<br>PCIe3 | 22 nm | 22 nm<br>AVX2<br>(new instructions)<br>DDR4<br>PCIe3 | 14 nm | 14 nm<br>AVX 3.2<br>(512 bit registers)<br>DDR4<br>PCIe4 |

https://chrisadkin.io/2015/06/04/under-the-hood-of-the-batch-engine-simd-with-sql-server-2016-ctp/

# Programming for SSE

Ways to exploit SSE in your code

- Program in assembly (highest performance, but would take another semester to learn)
- C++ vector classes (Available with Intel Compiler, or from Agner Fog)
- Optimized function libraries (Intel IPP & MKL, AMD ACML & LibM, ATLAS, FFTW)
- DSL compilers (Intel SPMD Compiler, Oil Runtime Compiler, AMD/Intel OpenCL library)
- Intrinsic functions
- Compiler autovectorization

# Documentation on x86 SIMD

a. MSDN
- MMX, SSE, and SSE2 intrinsics
- 3dnow! intrinsics
- SSSE3 intrinsics
- SSE4A and ABM intrinsics
- SSE4.1 and SSE4.2 intrinsics

b. Intrinsics supported by Intel Compiler
- Intel extensions only

c. Intel Intrinsics Guide
- Intel extensions only

d. Instruction set manuals from Intel and AMD
- Document instructions, not intrinsics

# References

- 64-BIT PROGRAMMING ARCHITECTURE
  http://download.intel.com/technology/architecture/new-instructions-paper.pdf

- Intel® 64 and IA-32 Architectures Software Developer's Manual
  http://www.intel.com/products/processor/manuals/

- Sang-Woo Jun, CS 295: Modern Systems
  Modern Processors – SIMD Extensions, UCI, 2019

- Marat Dukhan, Performance Tuning for CPU, Part 1: SIMD Optimization