en.wikipedia.org

Advanced Vector Extensions

Contributors to Wikimedia projects

34–43 minutes

From Wikipedia, the free encyclopedia

Advanced Vector Extensions (**AVX**, also known as **Gesher New Instructions** and then **Sandy Bridge New Instructions**) are SIMD extensions to the x86 instruction set architecture for microprocessors from Intel and Advanced Micro Devices (AMD). They were proposed by Intel in March 2008 and first supported by Intel with the Sandy Bridge[1] processor shipping in Q1 2011 and later by AMD with the Bulldozer[2] processor shipping in Q3 2011.

AVX provides new features, new instructions, and a new coding scheme.

AVX2 (also known as **Haswell New Instructions**) expands most integer commands to 256 bits and introduces new instructions. They were first supported by Intel with the Haswell processor, which shipped in 2013.

AVX-512 expands AVX to 512-bit support using a new EVEX prefix encoding proposed by Intel in July 2013 and first supported by Intel with the Knights Landing co-processor, which shipped in 2016.[3][4] In conventional processors, AVX-512 was introduced with Skylake server and HEDT processors in 2017.

Advanced Vector Extensions[edit]

AVX uses sixteen YMM registers to perform a single instruction on multiple pieces of data (see SIMD). Each YMM register can hold and do simultaneous operations (math) on:

- eight 32-bit single-precision floating point numbers or
- four 64-bit double-precision floating point numbers.

The width of the SIMD registers is increased from 128 bits to 256 bits, and renamed from XMM0–XMM7 to YMM0–YMM7 (in x86-64 mode, from XMM0–XMM15 to YMM0–YMM15). The legacy SSE instructions can be still utilized via the VEX prefix to operate on the lower 128 bits of the YMM registers.

AVX-512 register scheme as extension from the AVX (YMM0-YMM15) and SSE (XMM0-XMM15) registers

511 256	255 128	127 0
ZMM0	YMM0	XMM0
ZMM1	YMM1	XMM1

ZMM2	YMM2	XMM2
ZMM3	YMM3	XMM3
ZMM4	YMM4	XMM4
ZMM5	YMM5	XMM5
ZMM6	YMM6	XMM6
ZMM7	YMM7	XMM7
ZMM8	YMM8	XMM8
ZMM9	YMM9	XMM9
ZMM10	YMM10	XMM10
ZMM11	YMM11	XMM11
ZMM12	YMM12	XMM12
ZMM13	YMM13	XMM13
ZMM14	YMM14	XMM14

ZMM15	YMM15	XMM15
ZMM16	YMM16	XMM16
ZMM17	YMM17	XMM17
ZMM18	YMM18	XMM18
ZMM19	YMM19	XMM19
ZMM20	YMM20	XMM20
ZMM21	YMM21	XMM21
ZMM22	YMM22	XMM22
ZMM23	YMM23	XMM23
ZMM24	YMM24	XMM24
ZMM25	YMM25	XMM25
ZMM26	YMM26	XMM26
ZMM27	YMM27	XMM27

ZMM28	YMM28	XMM28
ZMM29	YMM29	XMM29
ZMM30	YMM30	XMM30
ZMM31	YMM31	XMM31

AVX introduces a three-operand SIMD instruction format called VEX coding scheme, where the destination register is distinct from the two source operands. For example, an SSE instruction using the conventional two-operand form $a \leftarrow a + b$ can now use a non-destructive three-operand form $c \leftarrow a + b$, preserving both source operands. Originally, AVX's three-operand format was limited to the instructions with SIMD operands (YMM), and did not include instructions with general purpose registers (e.g. EAX). It was later used for coding new instructions on general purpose registers in later extensions, such as BMI. VEX coding is also used for instructions operating on the k0-k7 mask registers that were introduced with AVX-512.

The alignment requirement of SIMD memory operands is relaxed.[5] Unlike their non-VEX coded counterparts, most VEX coded vector instructions no longer require their memory operands to be aligned to the vector size. Notably, the `VMOVDQA` instruction still requires its memory operand to be aligned.

The new VEX coding scheme introduces a new set of code prefixes that extends the opcode space, allows instructions to have more than two operands, and allows SIMD vector registers to be longer than 128 bits. The VEX prefix can also be used on the legacy SSE instructions giving them a three-operand form, and making them interact more efficiently with AVX instructions without the need for `VZEROUPPER` and `VZEROALL`.

The AVX instructions support both 128-bit and 256-bit SIMD. The 128-bit versions can be useful to improve old code without needing to widen the vectorization, and avoid the penalty of going from SSE to AVX, they are also faster on some early AMD implementations of AVX. This mode is sometimes known as AVX-128.[6]

New instructions[edit]

These AVX instructions are in addition to the ones that are 256-bit extensions of the legacy 128-bit SSE instructions; most are usable on both 128-bit and 256-bit operands.

Instruction	Description
`VBROADCASTSS`, `VBROADCASTSD`, `VBROADCASTF128`	Copy a 32-bit, 64-bit or 128-bit memory operand to all elements of a XMM or YMM vector register.
`VINSERTF128`	Replaces either the lower half or the upper half of a 256-bit YMM register with the value of a 128-bit source operand. The other half of the destination is unchanged.

`VEXTRACTF128`	Extracts either the lower half or the upper half of a 256-bit YMM register and copies the value to a 128-bit destination operand.
`VMASKMOVPS`, `VMASKMOVPD`	Conditionally reads any number of elements from a SIMD vector memory operand into a destination register, leaving the remaining vector elements unread and setting the corresponding elements in the destination register to zero. Alternatively, conditionally writes any number of elements from a SIMD vector register operand to a vector memory operand, leaving the remaining elements of the memory operand unchanged. On the AMD Jaguar processor architecture, this instruction with a memory source operand

	takes more than 300 clock cycles when the mask is zero, in which case the instruction should do nothing. This appears to be a design flaw.[7]
`VPERMILPS`, `VPERMILPD`	Permute In-Lane. Shuffle the 32-bit or 64-bit vector elements of one input operand. These are in-lane 256-bit instructions, meaning that they operate on all 256 bits with two separate 128-bit shuffles, so they can not shuffle across the 128-bit lanes.[8]
`VPERM2F128`	Shuffle the four 128-bit vector elements of two 256-bit source operands into a 256-bit destination operand, with an immediate constant as selector.
`VTESTPS`, `VTESTPD`	Packed bit test of the packed single-precision or double-precision floating-point

	sign bits, setting or clearing the ZF flag based on AND and CF flag based on ANDN.
`VZEROALL`	Set all YMM registers to zero and tag them as unused. Used when switching between 128-bit use and 256-bit use.
`VZEROUPPER`	Set the upper half of all YMM registers to zero. Used when switching between 128-bit use and 256-bit use.

CPUs with AVX[edit]

- Intel
- Sandy Bridge processors, Q1 2011[9]
- Sandy Bridge E processors, Q4 2011[10]
- Ivy Bridge processors, Q1 2012

- Ivy Bridge E processors, Q3 2013
- Haswell processors, Q2 2013
- Haswell E processors, Q3 2014
- Broadwell processors, Q4 2014
- Skylake processors, Q3 2015
- Broadwell E processors, Q2 2016
- Kaby Lake processors, Q3 2016 (ULV mobile)/Q1 2017 (desktop/mobile)
- Skylake-X processors, Q2 2017
- Coffee Lake processors, Q4 2017
- Cannon Lake processors, Q2 2018
- Whiskey Lake processors, Q3 2018
- Cascade Lake processors, Q4 2018

- Ice Lake processors, Q3 2019
- Comet Lake processors (only Core and Xeon branded), Q3 2019
- Tiger Lake (Core, Pentium and Celeron branded[11]) processors, Q3 2020
- Rocket Lake processors, Q1 2021
- Alder Lake (Core, Pentium and Celeron branded) processors, Q4 2021. Supported both in Golden Cove P-cores and Gracemont E-cores.
- Raptor Lake processors, Q4 2022
- Sapphire Rapids processors, Q1 2023
- Meteor Lake processors
- Arrow Lake processors
- Lunar Lake processors

Not all CPUs from the listed families support AVX. Generally, CPUs with the commercial denomination Core i3/i5/i7/i9 support them, whereas Pentium and Celeron CPUs before Tiger Lake[12] do not.

- AMD:
- Jaguar-based processors and newer
- Puma-based processors and newer
- "Heavy Equipment" processors
- Bulldozer-based processors, Q4 2011[13]
- Piledriver-based processors, Q4 2012[14]
- Steamroller-based processors, Q1 2014
- Excavator-based processors and newer, 2015
- Zen-based processors, Q1 2017
- Zen+-based processors, Q2 2018

- Zen 2-based processors, Q3 2019
- Zen 3 processors, Q4 2020
- Zen 4 processors, Q4 2022

  Issues regarding compatibility between future Intel and AMD processors are discussed under XOP instruction set.

- VIA:
- Nano QuadCore
- Eden X4
- Zhaoxin:
- WuDaoKou-based processors (KX-5000 and KH-20000)

Compiler and assembler support[edit]

- Absoft supports with -mavx flag.
- The Free Pascal compiler supports AVX and AVX2 with the -CfAVX

and -CfAVX2 switches from version 2.7.1.

- RAD studio (v11.0 Alexandria) supports AVX2 and AVX512.[15]
- The GNU Assembler (GAS) inline assembly functions support these instructions (accessible via GCC), as do Intel primitives and the Intel inline assembler (closely compatible to GAS, although more general in its handling of local references within inline code).
- GCC starting with version 4.6 (although there was a 4.3 branch with certain support) and the Intel Compiler Suite starting with version 11.1 support AVX.
- The Open64 compiler version 4.5.1 supports AVX with -mavx flag.
- PathScale supports via the -mavx flag.
- The Vector Pascal compiler supports AVX via the -cpuAVX32 flag.
- The Visual Studio 2010/2012 compiler supports AVX via intrinsic and /arch:AVX switch.

- Other assemblers such as MASM VS2010 version, YASM,[16] FASM, NASM and JWASM.

Operating system support[edit]

AVX adds new register-state through the 256-bit wide YMM register file, so explicit operating system support is required to properly save and restore AVX's expanded registers between context switches. The following operating system versions support AVX:

- DragonFly BSD: support added in early 2013.
- FreeBSD: support added in a patch submitted on January 21, 2012,[17] which was included in the 9.1 stable release[18]
- Linux: supported since kernel version 2.6.30,[19] released on June 9, 2009.[20]
- macOS: support added in 10.6.8 (Snow Leopard) update[21]

[*unreliable source?*] released on June 23, 2011. In fact, macOS Ventura does not support processors without the AVX2 instruction set. [22]

- OpenBSD: support added on March 21, 2015.[23]
- Solaris: supported in Solaris 10 Update 10 and Solaris 11
- Windows: supported in Windows 7 SP1, Windows Server 2008 R2 SP1,[24] Windows 8, Windows 10
- Windows Server 2008 R2 SP1 with Hyper-V requires a hotfix to support AMD AVX (Opteron 6200 and 4200 series) processors, KB2568088

Advanced Vector Extensions 2[edit]

Advanced Vector Extensions 2 (AVX2), also known as **Haswell New Instructions**,[25] is an expansion of the AVX instruction set introduced in Intel's Haswell microarchitecture. AVX2 makes the following additions:

- expansion of most vector integer SSE and AVX instructions to 256 bits
- Gather support, enabling vector elements to be loaded from non-contiguous memory locations
- DWORD- and QWORD-granularity any-to-any permutes
- vector shifts.

Sometimes three-operand fused multiply-accumulate (FMA3) extension is considered part of AVX2, as it was introduced by Intel in the same processor microarchitecture. This is a separate extension using its own CPUID flag and is described on its own page and not below.

New instructions[edit]

Instruction	Description
`VBROADCASTSS`, `VBROADCASTSD`	Copy a 32-bit or 64-bit register operand to all elements of a XMM or YMM vector register. These are register versions of the same instructions in AVX1. There is no 128-bit version however, but the same effect can be simply achieved using VINSERTF128.
`VPBROADCASTB`, `VPBROADCASTW`, `VPBROADCASTD`, `VPBROADCASTQ`	Copy an 8, 16, 32 or 64-bit integer register or memory operand to all elements of a XMM or YMM vector register.
`VBROADCASTI128`	Copy a 128-bit memory operand to all elements of a YMM vector register.
`VINSERTI128`	Replaces either the lower half or the

	upper half of a 256-bit YMM register with the value of a 128-bit source operand. The other half of the destination is unchanged.
`VEXTRACTI128`	Extracts either the lower half or the upper half of a 256-bit YMM register and copies the value to a 128-bit destination operand.
`VGATHERDPD`, `VGATHERQPD`, `VGATHERDPS`, `VGATHERQPS`	Gathers single or double precision floating point values using either 32 or 64-bit indices and scale.
`VPGATHERDD`, `VPGATHERDQ`, `VPGATHERQD`,	Gathers 32 or 64-bit integer values using either 32 or 64-bit indices and scale.

`VPGATHERQQ`	
`VPMASKMOVD`, `VPMASKMOVQ`	Conditionally reads any number of elements from a SIMD vector memory operand into a destination register, leaving the remaining vector elements unread and setting the corresponding elements in the destination register to zero. Alternatively, conditionally writes any number of elements from a SIMD vector register operand to a vector memory operand, leaving the remaining elements of the memory operand unchanged.
`VPERMPS`, `VPERMD`	Shuffle the eight 32-bit vector elements of one 256-bit source operand into a 256-bit destination operand, with a

	register or memory operand as selector.
`VPERMPD`, `VPERMQ`	Shuffle the four 64-bit vector elements of one 256-bit source operand into a 256-bit destination operand, with a register or memory operand as selector.
`VPERM2I128`	Shuffle (two of) the four 128-bit vector elements of *two* 256-bit source operands into a 256-bit destination operand, with an immediate constant as selector.
`VPBLENDD`	Doubleword immediate version of the PBLEND instructions from SSE4.
`VPSLLVD`, `VPSLLVQ`	Shift left logical. Allows variable shifts where each element is shifted according to the packed input.
`VPSRLVD`, `VPSRLVQ`	Shift right logical. Allows variable shifts

	where each element is shifted according to the packed input.
`VPSRAVD`	Shift right arithmetically. Allows variable shifts where each element is shifted according to the packed input.

CPUs with AVX2[edit]

- Intel
- Haswell processors (only Core and Xeon branded), Q2 2013
- Haswell E processors, Q3 2014
- Broadwell processors, Q4 2014
- Broadwell E processors, Q3 2016
- Skylake processors, Q3 2015
- Kaby Lake processors, Q3 2016 (ULV mobile)/Q1 2017 (desktop/

mobile)

- Skylake-X processors, Q2 2017
- Coffee Lake processors, Q4 2017
- Cannon Lake processors, Q2 2018
- Whiskey Lake processors, Q3 2018
- Cascade Lake processors, Q2 2019
- Ice Lake processors, Q3 2019
- Comet Lake processors, Q3 2019
- Tiger Lake (Core, Pentium and Celeron branded[11]) processors, Q3 2020
- Rocket Lake processors, Q1 2021
- Alder Lake (Xeon, Core, Pentium and Celeron branded[11]) processors, Q4 2021. Supported both in Golden Cove P-cores and

Gracemont E-cores.

- Raptor Lake processors, Q4 2022
- Sapphire Rapids processors, Q1 2023
- Meteor Lake processors
- Arrow Lake processors
- Lunar Lake processors
- AMD
- Excavator processor and newer, Q2 2015
- Zen processors, Q1 2017
- Zen+ processors, Q2 2018
- Zen 2 processors, Q3 2019
- Zen 3 processors, Q4 2020
- Zen 4 processors, Q4 2022

- Zen 5 processors, 2024
- VIA:
- Nano QuadCore
- Eden X4

AVX-512[edit]

AVX-512 are 512-bit extensions to the 256-bit Advanced Vector Extensions SIMD instructions for x86 instruction set architecture proposed by Intel in July 2013, and are supported with Intel's Knights Landing processor.[3]

AVX-512 instructions are encoded with the new EVEX prefix. It allows 4 operands, 8 new 64-bit opmask registers, scalar memory mode with automatic broadcast, explicit rounding control, and compressed displacement memory addressing mode. The width of the register file is increased to 512 bits and total register count increased to 32 (registers ZMM0-ZMM31) in x86-64 mode.

AVX-512 consists of multiple instruction subsets, not all of which are meant to be supported by all processors implementing them. The instruction set consists of the following:

- AVX-512 Foundation (F) – adds several new instructions and expands most 32-bit and 64-bit floating point SSE-SSE4.1 and AVX/AVX2 instructions with EVEX coding scheme to support the 512-bit registers, operation masks, parameter broadcasting, and embedded rounding and exception control
- AVX-512 Conflict Detection Instructions (CD) – efficient conflict detection to allow more loops to be vectorized, supported by Knights Landing[3]
- AVX-512 Exponential and Reciprocal Instructions (ER) – exponential and reciprocal operations designed to help implement transcendental operations, supported by Knights Landing[3]

- AVX-512 Prefetch Instructions (PF) – new prefetch capabilities, supported by Knights Landing[3]
- AVX-512 Vector Length Extensions (VL) – extends most AVX-512 operations to also operate on XMM (128-bit) and YMM (256-bit) registers (including XMM16-XMM31 and YMM16-YMM31 in x86-64 mode)[26]
- AVX-512 Byte and Word Instructions (BW) – extends AVX-512 to cover 8-bit and 16-bit integer operations[26]
- AVX-512 Doubleword and Quadword Instructions (DQ) – enhanced 32-bit and 64-bit integer operations[26]
- AVX-512 Integer Fused Multiply Add (IFMA) – fused multiply add for 512-bit integers.[27]:746
- AVX-512 Vector Byte Manipulation Instructions (VBMI) adds vector byte permutation instructions which are not present in AVX-512BW.

- AVX-512 Vector Neural Network Instructions Word variable precision (4VNNIW) – vector instructions for deep learning.
- AVX-512 Fused Multiply Accumulation Packed Single precision (4FMAPS) – vector instructions for deep learning.
- VPOPCNTDQ – count of bits set to 1.[28]
- VPCLMULQDQ – carry-less multiplication of quadwords.[28]
- AVX-512 Vector Neural Network Instructions (VNNI) – vector instructions for deep learning.[28]
- AVX-512 Galois Field New Instructions (GFNI) – vector instructions for calculating Galois field.[28]
- AVX-512 Vector AES instructions (VAES) – vector instructions for AES coding.[28]
- AVX-512 Vector Byte Manipulation Instructions 2 (VBMI2) – byte/

word load, store and concatenation with shift.[28]

- AVX-512 Bit Algorithms (BITALG) – byte/word bit manipulation instructions expanding VPOPCNTDQ.[28]
- AVX-512 Bfloat16 Floating-Point Instructions (BF16) – vector instructions for AI acceleration.
- AVX-512 Half-Precision Floating-Point Instructions (FP16) – vector instructions for operating on floating-point and complex numbers with reduced precision.

Only the core extension AVX-512F (AVX-512 Foundation) is required by all implementations, though all current implementations also support CD (conflict detection). All central processors with AVX-512 also support VL, DQ and BW. The ER, PF, 4VNNIW and 4FMAPS instruction set extensions are currently only implemented in Intel computing coprocessors.

The updated SSE/AVX instructions in AVX-512F use the same mnemonics as AVX versions; they can operate on 512-bit ZMM registers, and will also support 128/256 bit XMM/YMM registers (with AVX-512VL) and byte, word, doubleword and quadword integer operands (with AVX-512BW/DQ and VBMI).[27]:23

CPUs with AVX-512[edit]

AVX-512 Subset	F	CD	ER	PF	4FMAPS	4VNNIW	VPOPCNTDQ	VL	D
Intel Knights Landing (2016)	Yes		Yes		No				
Intel Knights Mill					Yes			No	

(2017)				
Intel Skylake-SP, Skylake-X (2017)	No		No	Yes
Intel Cannon Lake (2018)				
Intel Cascade Lake-SP (2019)				
Intel Cooper				

Lake (2020)				
Intel Ice Lake (2019)			Yes	
Intel Tiger Lake (2020)				
Intel Rocket Lake (2021)				
Intel Alder Lake			Not officially support	

(2021)			
AMD Zen 4 **(2022)**	Yes		Yes
Intel Sapphire Rapids **(2023)**			
AMD Zen 5 **(2024)**			

[29]

^Note 1 : AVX-512 is disabled by default in Alder Lake processors. On some motherboards with some BIOS versions, AVX-512 can be enabled in the BIOS, but this requires disabling E-cores.[30]

However, Intel has begun fusing AVX-512 off on newer Alder Lake processors.[31]

Compilers supporting AVX-512[edit]

- GCC 4.9 and newer[32]
- Clang 3.9 and newer[33]
- ICC 15.0.1 and newer[34]
- Microsoft Visual Studio 2017 C++ Compiler[35]

AVX-VNNI, AVX-IFMA[edit]

AVX-VNNI is a VEX-coded variant of the AVX512-VNNI instruction set extension. Similarly, AVX-IFMA is a VEX-coded variant of AVX512-IFMA. These extensions provide the same sets of operations as their AVX-512 counterparts, but are limited to 256-bit vectors and do not support any additional features of EVEX encoding, such as broadcasting, opmask registers or accessing more than 16 vector registers. These extensions allow support of VNNI and IFMA operations even when full AVX-512 support is not implemented in the processor.

CPUs with AVX-VNNI[edit]

- Intel
- Alder Lake processors, Q4 2021
- Raptor Lake processors, Q4 2022
- Sapphire Rapids processors, Q1 2023
- Meteor Lake processors
- Emerald Rapids processors
- Arrow Lake processors

- Lunar Lake processors
- AMD
- Zen 5 processors,[36] 2024

CPUs with AVX-IFMA[edit]

- Intel
- Sierra Forest processors
- Grand Ridge processors
- Meteor Lake processors

AVX10[edit]

AVX10, announced in August 2023, is a new, "converged" AVX instruction set. It addresses several issues of AVX-512, in particular that it is split into too many parts[37] (20 feature flags) and that it makes 512-bit vectors mandatory to support. AVX10 presents a simplified CPUID interface to test for instruction support, consisting of the AVX10 version number (indicating the set of instructions supported, with later versions always being a superset of an earlier one) and the available maximum vector length (256 or 512 bits).[38] A combined notation is used to indicate the version and vector length: for example, AVX10.2/256 indicates that a CPU is capable of the second version of AVX10 with a maximum vector width of 256 bits.[39]

The first and "early" version of AVX10, notated AVX10.1, will *not* introduce any instructions or encoding features beyond what is already in AVX-512 (specifically, in Intel Sapphire Rapids: AVX-512F, CD, VL, DQ, BW, IFMA, VBMI, VBMI2, BITALG, VNNI, GFNI, VPOPCNTDQ, VPCLMULQDQ, VAES, BF16, FP16). The second and "fully-featured" version, AVX10.2, introduces new features such as YMM embedded rounding and Suppress All

Exception. For CPUs supporting AVX10 and 512-bit vectors, all legacy AVX-512 feature flags will remain set to facilitate applications supporting AVX-512 to continue using AVX-512 instructions.[39]

AVX10.1/512 will be available on Granite Rapids.[39]

APX[edit]

APX is a new extension. It is not focused on vector computation, but provides RISC-like extensions to the x86-64 architecture by doubling the number of general purpose registers to 32 and introducing three-operand instruction formats. AVX is only tangentially affected as APX introduces extended operands.[40][41]

Applications[edit]

- Suitable for floating point-intensive calculations in multimedia, scientific and financial applications (AVX2 adds support for integer operations).

- Increases parallelism and throughput in floating point SIMD calculations.
- Reduces register load due to the non-destructive instructions.
- Improves Linux RAID software performance (required AVX2, AVX is not sufficient)[42]

Software[edit]

- Blender uses AVX, AVX2 and AVX-512 in the Cycles render engine.[43]
- Bloombase uses AVX, AVX2 and AVX-512 in their Bloombase Cryptographic Module (BCM).
- Botan uses both AVX and AVX2 when available to accelerate some algorithms, like ChaCha.

- BSAFE C toolkits uses AVX and AVX2 where appropriate to accelerate various cryptographic algorithms.[44]
- Crypto++ uses both AVX and AVX2 when available to accelerate some algorithms, like Salsa and ChaCha.
- Esri ArcGIS Data Store uses AVX2 for graph storage.[45]
- OpenSSL uses AVX- and AVX2-optimized cryptographic functions since version 1.0.2.[46] Support for AVX-512 was added in version 3.0.0.[47] Some of these optimizations are also present in various clones and forks, like LibreSSL.
- Prime95/MPrime, the software used for GIMPS, started using the AVX instructions since version 27.1, AVX2 since 28.6 and AVX-512 since 29.1.[48]
- dav1d AV1 decoder can use AVX2 and AVX-512 on supported CPUs.[49][50]

- SVT-AV1 AV1 encoder can use AVX2 and AVX-512 to accelerate video encoding.[51]
- dnetc, the software used by distributed.net, has an AVX2 core available for its RC5 project and will soon release one for its OGR-28 project.
- Einstein@Home uses AVX in some of their distributed applications that search for gravitational waves.[52]
- Folding@home uses AVX on calculation cores implemented with GROMACS library.
- Helios uses AVX and AVX2 hardware acceleration on 64-bit x86 hardware.[53]
- Horizon: Zero Dawn uses AVX in its Decima game engine.
- PCSX2 and RPCS3, are open source PS2 and PS3 emulators respectively that use AVX2 and AVX-512 instructions to emulate

games.

- Network Device Interface, an IP video/audio protocol developed by NewTek for live broadcast production, uses AVX and AVX2 for increased performance.
- TensorFlow since version 1.6 and tensorflow above versions requires CPU supporting at least AVX.[54]
- x264, x265 and VTM video encoders can use AVX2 or AVX-512 to speed up encoding.
- Various CPU-based cryptocurrency miners (like pooler's cpuminer for Bitcoin and Litecoin) use AVX and AVX2 for various cryptography-related routines, including SHA-256 and scrypt.
- libsodium uses AVX in the implementation of scalar multiplication for Curve25519 and Ed25519 algorithms, AVX2 for BLAKE2b, Salsa20, ChaCha20, and AVX2 and AVX-512 in implementation of Argon2 algorithm.

- libvpx open source reference implementation of VP8/VP9 encoder/decoder, uses AVX2 or AVX-512 when available.
- libjpeg-turbo uses AVX2 to accelerate image processing.
- FFTW can utilize AVX, AVX2 and AVX-512 when available.
- LLVMpipe, a software OpenGL renderer in Mesa using Gallium and LLVM infrastructure, uses AVX2 when available.
- glibc uses AVX2 (with FMA) and AVX-512 for optimized implementation of various mathematical (i.e. `expf`, `sinf`, `powf`, `atanf`, `atan2f`) and string (`memmove`, `memcpy`, etc.) functions in libc.
- Linux kernel can use AVX or AVX2, together with AES-NI as optimized implementation of AES-GCM cryptographic algorithm.
- Linux kernel uses AVX or AVX2 when available, in optimized implementation of multiple other cryptographic ciphers: Camellia, CAST5, CAST6, Serpent, Twofish, MORUS-1280, and other

primitives: Poly1305, SHA-1, SHA-256, SHA-512, ChaCha20.

- POCL, a portable Computing Language, that provides implementation of OpenCL, makes use of AVX, AVX2 and AVX-512 when possible.
- .NET and .NET Framework can utilize AVX, AVX2 through the generic `System.Numerics.Vectors` namespace.
- .NET Core, starting from version 2.1 and more extensively after version 3.0 can directly use all AVX, AVX2 intrinsics through the `System.Runtime.Intrinsics.X86` namespace.
- EmEditor 19.0 and above uses AVX2 to speed up processing.[55]
- Native Instruments' Massive X softsynth requires AVX.[56]
- Microsoft Teams uses AVX2 instructions to create a blurred or custom background behind video chat participants,[57] and for background noise suppression.[58]

- Pale Moon custom Windows builds greatly increase browsing speed due to the use of AVX2.
- simdjson, a JSON parsing library, uses AVX2 and AVX-512 to achieve improved decoding speed.[59][60]
- x86-simd-sort, a library with sorting algorithms for 16, 32 and 64-bit numeric data types, uses AVX2 and AVX-512. The library is used in NumPy and OpenJDK to accelerate sorting algorithms.[61]
- zlib-ng, an optimized version of zlib, contains AVX2 and AVX-512 versions of some data compression algorithms.
- Tesseract OCR engine uses AVX, AVX2 and AVX-512 to accelerate character recognition.[62]

Downclocking[edit]

Since AVX instructions are wider and generate more heat, some

Intel processors have provisions to reduce the Turbo Boost frequency limit when such instructions are being executed. On Skylake and its derivatives, the throttling is divided into three levels: [63][64]

- L0 (100%): The normal turbo boost limit.
- L1 (~85%): The "AVX boost" limit. Soft-triggered by 256-bit "heavy" (floating-point unit: FP math and integer multiplication) instructions. Hard-triggered by "light" (all other) 512-bit instructions.
- L2 (~60%):[*dubious* – *discuss*] The "AVX-512 boost" limit. Soft-triggered by 512-bit heavy instructions.

The frequency transition can be soft or hard. Hard transition means the frequency is reduced as soon as such an instruction is spotted; soft transition means that the frequency is reduced only after reaching a threshold number of matching instructions. The limit is per-thread.[63]

In Ice Lake, only two levels persist:[65]

- L0 (100%): The normal turbo boost limit.
- L1 (~97%): Triggered by any 512-bit instructions, but only when single-core boost is active; not triggered when multiple cores are loaded.

Rocket Lake processors do not trigger frequency reduction upon executing any kind of vector instructions regardless of the vector size.[65] However, downclocking can still happen due to other reasons, such as reaching thermal and power limits.

Downclocking means that using AVX in a mixed workload with an Intel processor can incur a frequency penalty. Avoiding the use of wide and heavy instructions help minimize the impact in these cases. AVX-512VL allows for using 256-bit or 128-bit operands in AVX-512, making it a sensible default for mixed loads.[66]

On supported and unlocked variants of processors that down-clock, the ratios are adjustable and may be turned off (set to 0x) entirely via Intel's Overclocking / Tuning utility or in BIOS if supported there. [67]

See also[edit]

- F16C instruction set extension
- Memory Protection Extensions
- Scalable Vector Extension for ARM - a new vector instruction set (supplementing VFP and NEON) similar to AVX-512, with some additional features.

References[edit]

1. ^ *Kanter, David (September 25, 2010). "Intel's Sandy Bridge Microarchitecture". www.realworldtech.com. Retrieved February 17, 2018.*

2. ^ Hruska, Joel (October 24, 2011). "Analyzing Bulldozer: Why AMD's chip is so disappointing - Page 4 of 5 - ExtremeTech". ExtremeTech. Retrieved February 17, 2018.

3. ^ Jump up to: a b c d e James Reinders (July 23, 2013), AVX-512 Instructions, Intel, retrieved August 20, 2013

4. ^ "Intel Xeon Phi Processor 7210 (16GB, 1.30 GHz, 64 core) Product Specifications". Intel ARK (Product Specs). Retrieved March 16, 2018.

5. ^ "14.9". Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture (PDF) (-051US ed.). Intel Corporation. p. 349. Retrieved August 23, 2014. “Memory arguments for most instructions with VEX prefix operate normally without causing #GP(0) on any byte-granularity alignment (unlike Legacy SSE instructions).”

6. ^ "i386 and x86-64 Options - Using the GNU Compiler Collection

(GCC)". Retrieved February 9, 2014.

7. ^ "The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers" (PDF). Retrieved October 17, 2016.
8. ^ "Chess programming AVX2". Archived from the original on July 10, 2017. Retrieved October 17, 2016.
9. ^ "Intel Offers Peek at Nehalem and Larrabee". ExtremeTech. March 17, 2008.
10. ^ "Intel Core i7-3960X Processor Extreme Edition". Retrieved January 17, 2012.
11. ^ Jump up to: a b c "Intel® Celeron® 6305 Processor (4M Cache, 1.80 GHz, with IPU) Product Specifications". ark.intel.com. Retrieved November 10, 2020.
12. ^ "Does a Processor with AVX2 or AVX-512 Support AVX

Instructions?". ark.intel.com. Retrieved April 27, 2022.

13. ^ *Dave Christie (May 7, 2009), Striking a balance, AMD Developer blogs, archived from the original on November 9, 2013, retrieved January 17, 2012*
14. ^ *New "Bulldozer" and "Piledriver" Instructions (PDF), AMD, October 2012*
15. ^ *"What's New - RAD Studio". docwiki.embarcadero.com. Retrieved September 17, 2021.*
16. ^ *"YASM 0.7.0 Release Notes". yasm.tortall.net.*
17. ^ *Add support for the extended FPU states on amd64, both for native 64bit and 32bit ABIs, svnweb.freebsd.org, January 21, 2012, retrieved January 22, 2012*
18. ^ *"FreeBSD 9.1-RELEASE Announcement". Retrieved May 20, 2013.*

19. ^ *x86: add linux kernel support for YMM state, retrieved July 13, 2009*

20. ^ *Linux 2.6.30 - Linux Kernel Newbies, retrieved July 13, 2009*

21. ^ *Twitter, retrieved June 23, 2010*

22. ^ *"Devs are making progress getting macOS Ventura to run on unsupported, decade-old Macs". August 23, 2022.*

23. ^ *Add support for saving/restoring FPU state using the XSAVE/XRSTOR., retrieved March 25, 2015*

24. ^ *Floating-Point Support for 64-Bit Drivers, retrieved December 6, 2009*

25. ^ *Haswell New Instruction Descriptions Now Available, Software.intel.com, retrieved January 17, 2012*

26. ^ Jump up to: *a* *b* *c* *James Reinders (July 17, 2014). "Additional AVX-512 instructions". Intel. Retrieved August 3, 2014.*

27. ^ Jump up to: [a] [b] "Intel Architecture Instruction Set Extensions Programming Reference" (PDF). Intel. Retrieved January 29, 2014.
28. ^ Jump up to: [a] [b] [c] [d] [e] [f] [g] "Intel® Architecture Instruction Set Extensions and Future Features Programming Reference". *Intel*. Retrieved October 16, 2017.
29. ^ "Intel® Software Development Emulator | Intel® Software". *software.intel.com*. Retrieved June 11, 2016.
30. ^ Cutress, Ian; Frumusanu, Andrei. "The Intel 12th Gen Core i9-12900K Review: Hybrid Performance Brings Hybrid Complexity". *AnandTech*. Retrieved November 5, 2021.
31. ^ Alcorn, Paul (March 2, 2022). "Intel Nukes Alder Lake's AVX-512 Support, Now Fuses It Off in Silicon". *Tom's Hardware*. Retrieved October 3, 2022.
32. ^ "GCC 4.9 Release Series — Changes, New Features, and Fixes –

GNU Project - Free Software Foundation (FSF)". gcc.gnu.org. Retrieved April 3, 2017.

33. ^ *"LLVM 3.9 Release Notes — LLVM 3.9 documentation". releases.llvm.org. Retrieved April 3, 2017.*
34. ^ *"Intel® Parallel Studio XE 2015 Composer Edition C++ Release Notes | Intel® Software". software.intel.com. Retrieved April 3, 2017.*
35. ^ *"Microsoft Visual Studio 2017 Supports Intel® AVX-512". July 11, 2017.*
36. ^ *"AMD Zen 5 Compiler Support Posted For GCC - Confirms New AVX Features & More". www.phoronix.com. Retrieved February 10, 2024.*
37. ^ *Mann, Tobias (August 15, 2023). "Intel's AVX10 promises benefits of AVX-512 without baggage". www.theregister.com. Retrieved August 20, 2023.*

38. ^ "The Converged Vector ISA: Intel® Advanced Vector Extensions 10 Technical Paper". Intel.

39. ^ Jump up to: a b c "Intel® Advanced Vector Extensions 10 (Intel® AVX10) Architecture Specification". Intel.

40. ^ "Intel® Advanced Performance Extensions (Intel® APX) Architecture Specification". Intel.

41. ^ Robinson, Dan (July 26, 2023). "Intel discloses x86 and vector instructions for future chips". www.theregister.com. Retrieved August 20, 2023.

42. ^ "Linux RAID". LWN. February 17, 2013. Archived from the original on April 15, 2013.

43. ^ Jaroš, Milan; Strakoš, Petr; Říha, Lubomír (May 28, 2022). "Rendering in Blender using AVX-512 Vectorization" (PDF). Intel eXtreme Performance Users Group. Technical University of

Ostrava. Retrieved October 28, 2022.

44. ^ *"Comparison of BSAFE cryptographic library implementations". July 25, 2023.*

45. ^ *"ArcGIS Data Store 11.2 System Requirements". ArcGIS Enterprise. Retrieved January 24, 2024.*

46. ^ *"Improving OpenSSL Performance". May 26, 2015. Retrieved February 28, 2017.*

47. ^ *"OpenSSL 3.0.0 release notes". GitHub. September 7, 2021.*

48. ^ *"Prime95 release notes". Retrieved July 10, 2022.*

49. ^ *"dav1d: performance and completion of the first release". November 21, 2018. Retrieved November 22, 2018.*

50. ^ *"dav1d 0.6.0 release notes". March 6, 2020.*

51. ^ *"SVT-AV1 0.7.0 release notes". September 26, 2019.*

52. ^ *"Einstein@Home Applications".*

53. ^ *"FAQ, Helios". Helios. Retrieved July 5, 2021.*

54. ^ *"Tensorflow 1.6". GitHub.*

55. ^ New in Version 19.0 – EmEditor (Text Editor)

56. ^ *"MASSIVE X Requires AVX Compatible Processor". Native Instruments. Retrieved November 29, 2019.*

57. ^ *"Hardware requirements for Microsoft Teams". Microsoft. Retrieved April 17, 2020.*

58. ^ *"Reduce background noise in Teams meetings". Microsoft Support. Retrieved January 5, 2021.*

59. ^ *Langdale, Geoff; Lemire, Daniel (2019). "Parsing Gigabytes of JSON per Second". The VLDB Journal. **28** (6): 941–960. arXiv:1902.08318. doi:10.1007/s00778-019-00578-5. S2CID 67856679.*

60. ^ *"simdjson 2.1.0 release notes". GitHub. June 30, 2022.*

61. ^ Larabel, Michael (October 6, 2023). "OpenJDK Merges Intel's x86-simd-sort For Speeding Up Data Sorting 7~15x". Phoronix.
62. ^ Larabel, Michael (July 7, 2022). "Tesseract OCR 5.2 Engine Finds Success With AVX-512F". Phoronix.
63. ^ Jump up to: *a* *b* Lemire, Daniel (September 7, 2018). "AVX-512: when and how to use these new instructions". Daniel Lemire's blog.
64. ^ BeeOnRope. "SIMD instructions lowering CPU frequency". Stack Overflow.
65. ^ Jump up to: *a* *b* Downs, Travis (August 19, 2020). "Ice Lake AVX-512 Downclocking". Performance Matters blog.
66. ^ "x86 - AVX 512 vs AVX2 performance for simple array processing loops". Stack Overflow.
67. ^ "Intel® Extreme Tuning Utility (Intel® XTU) Guide to Overclocking : Advanced Tuning". Intel. Retrieved July 18, 2021.

“See image in linked section, where AVX2 ratio has been set to 0.”

External links[edit]

- Intel Intrinsics Guide
- x86 Assembly Language Reference Manual