Intel® CPUID Passthrough Virtualization Considerations

Technical Paper

August 2025

Revision 3.0

Document Number: 356709-003US

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not “commercial” names and not intended to function as trademarks.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exception that a) you may publish an unmodified copy and b) code included in this document is licensed subject to the Zero-Clause BSD open source license (0BSD), https://opensource.org/licenses/0BSD. You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

CONTENTS

PAGE

TABLES

Revision History

Revision	Description	Date
1.0	Initial release.	Sept 2023
2.0	Added clarification for VMM passthrough.	March 2024
3.0	Changed title of document to Intel® CPUID Passthrough Virtualization Considerations (from Intel® Advanced Vector Extensions 10 Virtualization Considerations). Added Chapter 1, "Virtualization CPUID Passthrough Introduction." Added Chapter 3, "Intel® Advanced Matrix Extensions Virtualization Passthrough."	Aug 2025

CHAPTER 1
VIRTUALIZATION CPUID PASSTHROUGH INTRODUCTION

Virtual machine monitors (VMMs) were primarily in servers but have grown to be standard on user desktops and mobile devices. This is a challenge in lining up new hardware releases with system software updates since VMMs do not typically expose features that were not known at the time of their release.

There are many technologies that require updating system software, even prior to VMMs. However, there are features in which an operating system can enable once and extensions to the feature become available to applications without an update. This scenario is affected by the addition of VMMs since they have stricter control on exposure of features by the hardware than the operating system.

The CPUID instruction is a native instruction that applications can use to determine features supported by the hardware. Virtual machine monitors intercept CPUID and filter the bits returned to the set or subset of features that it understands. New features represent feature flags or sets of bits that the VMM does not understand and cannot provide to the guest operating system as it could render the guest unstable or non-functional. The reason is that the VMM is required to add additional support for the guest to operate when enumerating a feature, such as accessing an MSR the VMM does not know to allow access or enabling state the VMM does not comprehend.

Intel has implemented architecture around CPUID to allow a VMM to consider passing through new feature bits that were not known at the time of the VMM's release. The architecture is to pre-define the enabling requirements around CPUID subleaves, and the VMM can make an informed decision if passing through these features would work for their architecture.

There are additional considerations for a live migration scenario as compared to a desktop. The desktop would passthrough all the bits and not worry about compatibility across systems. Creating a compatible guest for live migration could be achieved through an intersection or minimum value set of the features across the systems.

This document contains sections on the current features and definitions surrounding CPUID that allow the VMM to consider passing through given pre-defined architectural knowledge. This knowledge will include the required state enabling and the CPUID subleaf for which the architectural definition applies. These feature definitions do not provide new requirements, such as MSRs or MSR bits. These are user mode features that rely on the base state enabling, for example a new instruction for Intel® AVX10 that requires Intel® AVX-512 state. The base enabling may optionally include the Extended Feature Disable (XFD), the passing through of features does not have any relation to requiring it be implemented.

CHAPTER 2
INTEL® ADVANCED VECTOR EXTENSIONS 10 PASSTHROUGH

Intel® Advanced Vector Extensions 10 (Intel® AVX10) establishes a common, converged vector instruction set across all Intel architectures, incorporating the modern vectorization aspects of Intel AVX-512. The architecture details can be found in the Intel® Advanced Vector Extensions 10.2 (Intel® AVX10.2) Architecture Specification.

VMMs may consider passing through the Intel AVX10 feature to guest VMs, which includes the CPUID enumeration. The traditional method of exposing features is to allow fields known at the time of VMM release to be passed through to guests and, in some cases, fields which contain values would be sanitized to known limits.

The CPUID fields shown in Table 2-1 identify the enumeration for the support and version number of AVX10. Typically, a CPUID field such as "Intel AVX10 Converged Vector ISA Version" would be limited to less than or equal to the version known by the VMM software. However, the architecture of the AVX10 version only requires the support of the states in Table 2-2. VMMs that expose and allow a guest to use the states in Table 2-2 can consider passing through the version number without limits. The version number represents the set of instructions that the platform supports.

Table 2-1. CPUID Enumeration of Intel® AVX10

CPUID Bit	Description	Type
CPUID.07H.01H:EDX[bit 19]	If 1, the Intel AVX10 Converged Vector ISA is supported.	Bit (0/1)
CPUID.24H.00H:EBX[bits 7:0]	Reports the Intel AVX10 Converged Vector ISA version.	Integer (≥ 1)

Table 2-2. Required Set of States for Passing Through Intel® AVX10 to a Guest VM

State Bit	XCR0 Bit Number
XCR0.SSE	1
XCR0.AVX	2
XCR0.opmask	5
XCR0.ZMM_Hi256	6
XCR0.Hi16_ZMM	7

Subleaf 1 as shown in Table 2-3 is architecturally defined to enumerate feature flags that require the state in Table 2-2. Subleaves 2 and beyond are currently reserved and do not have an architectural definition for future usage. A VMM can consider passing through subleaf 1 unfiltered when subleaf 1 is enumerated and the required state is supported. Subleaves greater than 1 are reserved and should not be passed through until they are defined. Refer to the Intel® 64 and IA-32 Architectures Software Developer's Manual or the Intel® Advanced Vector Extensions 10.2 (Intel® AVX10.2) Architecture Specification for new feature enumerations.

Table 2-3. CPUID Enumeraiton of Intel® AVX10 Feature Bits

CPUID Bit	Description	Type
CPUID.24H.00H:EAX[bits 31:0]	Reports the maximum supported subleaf.	Integer ≥ 1
CPUID.24H.01H:EAX[bits 31:0]	Reserved for AVX10 discrete feature flags.	Feature Flags
CPUID.24H.01H:EBX[bits 31:0]	Reserved for AVX10 discrete feature flags.	Feature Flags
CPUID.24H.01H:ECX[bits 31:0]	Reserved for AVX10 discrete feature flags.	Feature Flags
CPUID.24H.01H:EDX[bits 31:0]	Reserved for AVX10 discrete feature flags.	Feature Flags

CHAPTER 3
INTEL® ADVANCED MATRIX EXTENSIONS VIRTUALIZATION PASSTHROUGH

Intel® Advanced Matrix Extensions (Intel® AMX) is a built-in accelerator that improves the performance of deep-learning training and inference on the CPU and is ideal for workloads like natural-language processing, recommendation systems and image recognition. The architecture details can be found in the Intel® 64 and IA-32 Architectures Software Developer's Manual.

The CPUID fields shown in Table 3-1 identify the enumeration of subleaf 1 for AMX. Typically, subleaves and their bits would not be passed onto a guest until the VMM software understands what the bits represent. The passthrough support provides the first level of understanding, which is that subleaf 1 will contain feature bits that can be used by software to enable the state in Table 3-2. A VMM may now additionally consider passing through subleaf 1 without understanding the specific feature details if it is supporting the required state. Subleaves greater than 1 are reserved and should not be passed through until they are defined.

Table 3-1. CPUID Subleaf 1 Enumeration of Intel® AMX

CPUID Bit	Description	Type
CPUID.1EH.00H:EAX[bits 31:0]	Reports the maximum supported subleaf.	Integer ≥ 1
CPUID.1EH.01H:EAX[bits 31:0]	Reserved for AMX discrete feature flags.	Feature Flags
CPUID.1EH.01H:EBX[bits 31:0]	Reserved for AMX discrete feature flags.	Feature Flags
CPUID.1EH.01H:ECX[bits 31:0]	Reserved for AMX discrete feature flags.	Feature Flags
CPUID.1EH.01H:EDX[bits 31:0]	Reserved for AMX discrete feature flags.	Feature Flags

Table 3-2. Required Set of States for Passing Through Intel® AMX to a Guest VM

State Bit	XCR0 Bit Number
XCR0.SSE	1
XCR0.AVX	2
XCR0.opmask	5
XCR0.ZMM_Hi256	6
XCR0.Hi16_ZMM	7
XCR0.TILECFG	17
XCR0.TILEDATA	18

This represents a change from the existing AMX enumerations which were enumerated in CPUID Leaf 7 as shown in Table 3-3. These enumerations will remain to support backwards compatibility; however, they will additionally be found in the new subleaf as shown in Table 3-4 to allow new software to use a single location to verify when that subleaf is available. Both locations will always exist except on legacy platforms that do not enumerate subleaf 1. New features will not be enumerated in two locations; these will only be enumerated in subleaf 1 if they meet the criteria. Refer to the Intel® 64 and IA-32 Architectures Software Developer's Manual for new feature enumerations.

Table 3-3. Existing CPUID Leaf 7 AMX Feature Enumerations

CPUID Leaf 7 Bit	Description	Type
CPUID.07H.00H:EAX[bit 22]	AMX-BF16	Bit (0/1)
CPUID.07H.00H:EDX[bit 25]	AMX-INT8	Bit (0/1)

Table 3-3. Existing CPUID Leaf 7 AMX Feature Enumerations

CPUID Leaf 7 Bit	Description	Type
CPUID.07H.01H:EAX[bit 21]	AMX-FP16	Bit (0/1)
CPUID.07H.01H:EBX[bit 8]	AMX-COMPLEX	Bit (0/1)

Table 3-4. New AMX CPUID Enumerations

Description	CPUID Leaf 1EH Subleaf 1	CPUID Leaf 7
AMX-INT8	CPUID.1EH.01H:EAX[bit 0]	CPUID.07H.00H:EDX[bit 25]
AMX-BF16	CPUID.1EH.01H:EAX[bit 1]	CPUID.07H.00H:EAX[bit 22]
AMX-COMPLEX	CPUID.1EH.01H:EAX[bit 2]	CPUID.07H.01H:EBX[bit 8]
AMX-FP16	CPUID.1EH.01H:EAX[bit 3]	CPUID.07H.01H:EAX[bit 21]
Reserved for AMX discrete feature flags.	CPUID.1EH.01H:EAX[bits 31:4]	N/A
Reserved for AMX discrete feature flags.	CPUID.1EH.01H:EBX[bits 31:0]	N/A
Reserved for AMX discrete feature flags.	CPUID.1EH.01H:ECX[bits 31:0]	N/A
Reserved for AMX discrete feature flags.	CPUID.1EH.01H:EDX[bits 31:0]	N/A